1-1-2012

Modelling surface evolution in abrasive jet micromachining using level set methods

Tom Burzynski
Ryerson University

Recommended Citation

Burzynski, Tom, "Modelling surface evolution in abrasive jet micromachining using level set methods" (2012). *Theses and dissertations.* Paper 654.

MODELLING SURFACE EVOLUTION IN ABRASIVE JET MICROMACHINING USING LEVEL SET METHODS

by

Tom Burzynski

MASc, Mechanical Engineering, Ryerson University, 2007

BEng, Mechanical Engineering, Ryerson University, 2005

A dissertation
presented to Ryerson University

in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy
in the Program of
Mechanical Engineering

Toronto, Ontario, Canada, 2012

Author’s Declaration

AUTHOR'S DECLARATION FOR ELECTRONIC SUBMISSION OF A DISSERTATION

I hereby declare that I am the sole author of this dissertation. This is a true copy of the dissertation, including any required final revisions, as accepted by my examiners.

I authorize Ryerson University to lend this dissertation to other institutions or individuals for the purpose of scholarly research.

I further authorize Ryerson University to reproduce this dissertation by photocopying or by other means, in total or in part, at the request of other institutions or individuals for the purpose of scholarly research.

I understand that my dissertation may be made electronically available to the public.

MODELLING SURFACE EVOLUTION IN ABRASIVE JET MICROMACHINING
USING LEVEL SET METHODS

Tom Burzynski

Doctor of Philosophy

2012

Mechanical Engineering

Ryerson University

Abstract

The time dependent surface evolution in abrasive jet micromachining (AJM) is described by a partial differential equation which is difficult to solve using analytical or traditional numerical techniques. These techniques can yield incorrect predicted profile evolution or fail altogether under certain conditions. More recently developed particle tracking cellular automaton simulations can address some of these limitations but are difficult to implement and are computationally expensive.

In this work, level set methods (LSM) were introduced to develop novel surface evolution models to predict resulting feature shapes in AJM. Initially, a LSM-based numerical model was developed to predict the surface evolution of unmasked channels machined at normal and oblique jet impact angles (incidence), as well as masked micro-channels and micro-holes at normal incidence, in both brittle and ductile targets.

This model was then extended to allow the prediction of: surface evolution of inclined masked micro-channels made using AJM at oblique incidence, where the developing profiles rapidly become multi-valued necessitating a more complex formulation; mask erosive wear by permitting surface evolution of both the mask and target micro-channels simultaneously at any jet incidence; and surface damage due to secondary particle strikes in brittle target micro-channels resulting from particle mask-to-target and target-to-target ricochets at any jet incidence. For all the models, a general 'masking' function was developed by applying previous concepts to model the adjustment to abrasive mass flux incident to the target or mask surfaces to reflect the range of particle sizes that are ‘visible’ to these surfaces. The models were also optimized for computational efficiency using an adaptive Narrow Band LSM scheme.

All models were experimentally verified and, where possible, compared against existing models. Generally, good predictive capabilities and improvements over previous attempts in terms of feature prediction or execution time, were observed.

The proposed LSM-based models can be practical assistive tools during the micro-fabrication of complex MEMS and microfluidic devices using AJM.

Acknowledgements

I would like to sincerely thank my supervisor, Prof. Marcello Papini, for his continual guidance, advice and support in the research related to and the writing of this dissertation.

I am very thankful to my wife, Magda, and my parents, for their continual emotional and moral support and advice.

I gratefully acknowledge the financial support of the Natural Sciences and Engineering Research Council of Canada (NSERC), the Department of Mechanical and Industrial Engineering and the School of Graduate Studies at Ryerson University.

Finally, I would like to thank my lab peers, Dr. David Ciampini and Levon Larson, for providing unpublished experimental results (UR), i.e. the necessary mask mechanical properties and erosion parameters (Chapter 4).

Dedication

To my wife, Magda,

For her immense patience, emotional support and sacrifices.

I thank her for giving me the courage and strength to persevere through the biggest challenges on this journey.

Most of all, I thank her for her unconditional love and for being my best friend and partner for life.

To my parents,

For their upbringing, love and countless sacrifices.

For their moral and financial support throughout the years.

I thank them for giving me life and for striving to be the best parents that they could be. I am forever grateful to them.

Table of Contents

Chapter 3 Level Set Methodology for Predicting the Surface Evolution of Inclined Masked Micro-Channels Resulting from Abrasive Jet Micromachining at Oblique Incidence……………………………………………………………….50

List of Tables

List of Figures

List of Appendices

Nomenclature

$^{+,\,-}$	Superscripts, indicating that the physical quantity in question is measured to, at, or corresponds to measurement at the right and left mask edges, respectively
$^{\mathrm{1st,\,2nd}}$	Superscripts, indicating first and second strike contributions, respectively
$_{\mathrm{a,\,e}}$	Subscripts, indicating that the quantity in question corresponds to initially eroding surface node and potentially eroding surface node, respectively
$\vec{a}$	Nearest point on the surface, measured from $\vec{x}$
AR	Aspect ratio, feature depth-to-width ratio
$_{\mathrm{b,\,d}}$	Subscripts, indicating ‘brittle’ and ‘ductile’, respectively
$^{\mathrm{c}}$	Superscript, indicating that for the quantity in question, the partial derivatives of Φ are evaluated using central finite difference approximations
C	Empirical brittle erosion constant ($(\mathrm{m\ s^{-1}})^{-k_\mathrm{v}}$)
d_m	Approximate length of the eroding mask edge measured along x' (m)
D_min	Minimum distance between Φ^o and the upper or lower bands required to trigger band re-initialization (m)
D_LB, D_UB	Band width measured from Φ^o to the lower and upper bands, respectively (m)
$_\mathrm{ext}$	Subscript, indicating extension variable (see F_ext)
$E_\mathrm{ef}(\vec{x},t)$	Erosive efficacy (power) (kg $\mathrm{m^{-2}\ s^{-1}}$)
$E_\mathrm{ef,st,}$, $E_\mathrm{ef,t}$ $\overline{E}_\mathrm{ef,t}$	Erosive efficacy for a stationary target/mask, scanning target/mask and its average, respectively (kg $\mathrm{m^{-2}\ s^{-1}}$)
$E_\mathrm{r}(\vec{x},t)$	Erosion rate, the mass of target material removed per mass of incident particles
ET	Execution time (min (hrs.))
f_v, f_θ	Surface rebound parameters defined as a fraction of the arriving particle speed and angle, respectively
$F(\vec{x},t)$	Local surface normal velocity function (m $\mathrm{s^{-1}}$)
F_ext	Extension velocity, the local surface normal velocity function extended from the zero level set, Φ^o, to the closest grid point $\vec{x}$ (m $\mathrm{s^{-1}}$)

FR	Frequency of re-initialization, how often the $\mathrm{SDF}(\vec{x})$ must be recalculated for the entire computational grid during the course of a simulation, per fixed number of time steps
g	Denotes the global x or z spatial variable
h	Nozzle to target standoff distance (mm)
$H(\nabla\Phi(\vec{x}))$, $\hat{H}$	Hamiltonian and numerical Hamiltonian function, respectively
H_{m}	Pre-machined (unworn) mask height (μm)
$H_{\mathrm{m,eff}}$, $H_{\mathrm{m,eff,90}}$	Effective height, measured parallel to the jet centreline, and vertical height, respectively, of the eroding mask (m)
H_{Φ_g}	Partial derivative of H with respect to Φ_g
Hv	Initial Vickers target or mask hardness (GPa)
i ($i_{\max}$), k ($k_{\max}$)	(Maximum) grid indices of the spatial global coordinates x and z, respectively
k_{v}	Velocity exponent
K	Surface curvature
l	Grid index i or k of the spatial direction x or z, of the partial derivative of Φ
$l_{\mathrm{m,L}}$, $l_{\mathrm{m,R}}$	Pre-machined length of the left and right hand side of the unworn mask, respectively (μm)
L	Target location that an infinitely small particle can reach without undergoing collision with the mask edge, measured along the x' direction at a given z' (m)
m	Iteration number, 0,1,2,…
$_{\mathrm{M}}$	Subscript, indicating 'mask surface'
M	Particle mass at a given x' (kg)
$\dot{M}$	Particle mass flow rate through the nozzle (kg s^{-1} (g min^{-1}))
$M_{x/0}$	Proportion of the total incoming particle mass, M, arriving to the surface at a given x', or masking function, i.e. the adjustment to the incoming particle mass flux incident to the surface at a given x'
$M_{x/0}\big\vert_{\mathrm{M,edge}}, \overline{M}_{x/0}\big\vert_{\mathrm{M,edge}}$	Masking function for the mask edge surface and its mean over the distance d_{m}
$M_{x/0}\big\vert_{\mathrm{M,gen}}, M_{x/0}\big\vert_{\mathrm{T,gen}}$	General masking function for the entire mask and the target surface, respectively

$M_{x/0}\big\|_{\mathrm{T}}$	Masking function for the target surface
$M_{x/0}\big\|_{\mathrm{Unif}}$	Unified masking function for both the mask and target surface
$\vec{n}$	Surface normal
n_1, n_2	Empirical ductile erosion constants
N_{p}	Number of passes $M_{x/0}\big\|_{\mathrm{M,gen}}$ is assumed to be 1 for the entire RM mask
$^{\mathrm{o}}$	Superscript, indicating that the quantity in question is obtained at the point on the zero level set, Φ^{o}, which is closest to the grid point $\vec{x}$
$p_{r\mathrm{s}}$	The proportion of r_{s}, measured along the scanning direction, used in defining $\bar{y}$
r_{m}	Mask edge radius (μm)
r_{p}	Particle radius (m)
r_{s}	Radius of the impact area of the jet on the unmasked target surface measured in the y - z plane (m)
$\mathrm{SDF}(\vec{x})$	Signed distance function of $\vec{x}$, the minimum positive or negative distance between $\vec{x}$ and $\vec{a}$ (positive if $\vec{x}$ is in front of the surface propagation direction, and negative if behind it)
t	Time variable (s)
$\vec{t}$	Surface tangent
$_{\mathrm{T}}$	Subscript, indicating 'target surface'
T_{pass}	The time it takes for the surface to propagate along the profile centreline to a depth defined by the experimental first pass profile (s pass^{-1})
U_{crit}	Critical minimum distance between potentially eroding surface node and initially eroding surface node (μm)
$\vec{U}_{\mathrm{ea}}$	Unit vector distance from initially eroding surface node to potentially eroding surface node (m)
%USV	Percentage of unmachined surface visibility (%)
v_t	Scan speed, measured in the positive y direction (mm s^{-1})

$V(\vec{x},t)$, V_o	Particle velocity (distribution) incident to the surface and maximum jet centre particle velocity, respectively (m s^{-1})
$\vec{V}$	Particle impact velocity vector (m s^{-1})
$\vec{V}_\mathrm{d}$	'Actual' surface departing particle velocity vector, accounting for energy losses (m s^{-1})
W_m	Pre-machined, i.e. unworn, mask opening width (μm)
$W_\mathrm{m,eff}/2$, $W_\mathrm{m,eff,90}/2$, $W_\mathrm{m,eff,90}$	Effective half opening width measured along x', and effective half and full horizontal opening width, respectively, of the eroding mask (m)
$W_\mathrm{m,min}$	The minimum pre-machined, i.e. unworn, mask opening width, necessary so that the surface is 'visible' to the nozzle (m)
$x, y, z; x', y', z'$	Global Cartesian and local transformed spatial coordinates, respectively (m)
$\vec{x}$	Vector representation of a point on the grid in terms of the global spatial variables, i.e. x, y, and z (m)
x''	Horizontal spatial coordinate, measured from the nozzle tip (m)
x'_lim	The limit, measured along the x' direction at a given z', that an infinitesimally small particle can surpass and reach the target, without colliding with the left edge of the mask (m)
x_m	The mask shadow width, measured from the left hand edge of the mask opening (m)
x_max, x_min	Maximum and minimum horizontal grid limit, respectively (m)
x'_max, x'_min	Maximum and minimum x' to the mask surface, respectively, measured from the jet centreline (m)
x_off	Horizontal offset distance between the global and nozzle axes (m)
x'_tran	x' at transition between the mask edge and top of mask (m)
xyz	Global axis, defined to the left of the nozzle axis, offset by x_off
$x''yz$	Nozzle axis, defined at the nozzle tip
$x'y'z'$	Transformed axis, defined at the nozzle tip
$x_\mathrm{o}y_\mathrm{o}z_\mathrm{o}$	Original, i.e. previously used, axis, defined on the unmachined target surface
$\bar{y}$	Transverse variable, defined in terms of p_{rs}, locating the point at which $E_\mathrm{ef,st}$ best approximates $\overline{E}_\mathrm{ef,t}$ (m)

z_{am}	Vertical grid distance above the mask (μm)
z_{max}, z_{min}	Maximum and minimum vertical grid limit, respectively (m)
z_{surf}	Maximum expected feature depth (μm)
$z_{\mathrm{T,adj}}$	Vertical distance over which $M_{x/0}\big\|_{\mathrm{T,gen}} = \overline{M}_{x/0}\big\|_{\mathrm{M,edge}}$ to avoid unrealistic slowdown of surface propagation of the mask edges (μm)
α	Impact angle (angle of incidence), measured from the horizontal axis
β	'Focus coefficient', measuring the spread of the abrasive jet
γ^{g}	Bound of the partial derivative of H with respect to Φ_g
Δx, Δz	Horizontal and vertical spatial grid step, respectively (μm)
Δs_{crit}	Critical displacement dictating whether a secondary collision will occur at a potentially eroding surface node (μm)
Δs_{min}	Minimum displacement between particle trajectories defined by the 'actual' surface departing particle velocity vector accounting for energy losses and surface departing particle velocity vector arriving to a potentially eroding surface node (m)
Δt	Time step (s)
ε	Free parameter regulating the predicted speed of surface evolution in regions containing elevated curvature
ζ	Angle defining the maximum particle trajectory incident to the target surface through the mask opening, measured from the jet centreline to the mask edge
ζ_{lim}	Angle used in defining x'_{lim}, measured from the jet centreline to the left mask edge
θ, θ_{avg}	Angle between the surface normal and the particle impact velocity vector, and its average of all the local slopes at the mask edge, respectively
θ_{d}	Angle between the surface normal at an initially eroding surface node and the 'actual' surface departing particle velocity vector accounting for energy losses
$\theta_{\mathrm{e'd}}$	Angle between the 'actual' surface departing particle velocity vector accounting for energy losses and surface departing particle velocity vector arriving to a potentially eroding surface node
μ_{l}, σ_{l}	Log-normal mean and standard deviation of r_{p}, respectively

ξ	Angle between the velocity vector and surface tangent
ρ	Target or mask mass density (kg m^{-3})
$\phi(\vec{x},t)$	Particle mass flux (distribution), the mass of particles per unit time arriving to a unit surface area at a given surface location (kg m^{-2} s^{-1})
$\Phi(\vec{x},t)$, Φ^{o}	Level set function and zero level set, $\Phi(\vec{x},t) = 0$, i.e. where the surface is located, respectively
Φ_g , Φ_{gg}	First and second partial derivative of Φ with respect to g, respectively
Φ_g^{c} , Φ_{gg}^{c}	Central finite difference approximation to the first and second partial derivative of Φ with respect to g, respectively
Φ_g^{+} , Φ_g^{-}	Forward and backward finite difference approximation to the first partial derivative of Φ with respect to g, respectively
ψ	Maximum jet spread angle for an unmasked surface
Ψ	Heaviside function
$\Omega(r_{\mathrm{p}})$	Particle radial size distribution
$\Omega(r_{\mathrm{p}})dr_{\mathrm{p}}$	Proportion of particles having a radius between r_{p} and $r_{\mathrm{p}} + dr_{\mathrm{p}}$
$\nabla\Phi(\vec{x})$	Gradient of $\Phi(\vec{x},t)$

Chapter 1 Introduction

1.1 Motivation

Abrasive jet micromachining (AJM) is a relatively novel top-down technique for the micro-machining of features such as micro-channels, micro-holes, etc. in glass and polymers. Pressurized air is mixed with small particles and passed through a small nozzle, to produce a jet of abrasive particles which is directed towards a target surface. The resulting mechanical erosion of the target substrate can be controlled by applying patterned masks and by varying jet parameters such as the angle that the nozzle makes with the surface, the impact angle, α, also known as angle of incidence, the standoff distance, the nozzle to target distance, h (Figure 1.1), the particle mass flux, the mass of particles per unit area per unit time, the particle velocity, shape and size as well as the substrate properties [1].

AJM has been used to machine micro-components for use in the electronic, e.g. LCD or plasma flat panel displays, microfluidic, Micro Electro Mechanical Systems (MEMS), and opto-electronic industries [1-3]. For example, AJM can be used to micro-machine glass to produce three-dimensional (3D) suspended micro-cantilever beams for inertial sensors [4], microfluidic channels [5] and other features with aspect ratios (AR), feature depth-to-width ratios, as high as 7 [6]. AJM can also be used to machine polymers, such as poly-methyl-methacrylate (PMMA) and acrylonitrile-butadiene-styrene (ABS) [7], and with a recently developed cryogenic cooling technique, elastomers such as poly-dimethyl-siloxane (PDMS) [8]. Polymers are of great interest for microfluidic and MEMS applications due to their low cost and the fact that they are available with a wide variety of properties [9]. AJM may also be suitable for the machining of micro-moulding dies for the mass production of micro-components [10]. AJM can be an attractive micro-fabrication alternative to traditional wet etch technologies due to its relatively low capital cost, extremely high etch speed [11], and its ability to easily create multi-depth, anisotropic patterns and structures [6,12].

The development of process models that are able to predict the surface evolution, i.e. the size and shape of the machined feature, in AJM as a function of the process parameters is of great interest. Analytical and semi-empirical/computational models have been developed to predict the evolution of masked and unmasked features of relatively low aspect ratios and relatively simple geometrical shape, e.g. micro-holes or channels machined at normal (α = 90°) jet incidence (Figure 1.1). However, the majority of these models suffer from a number of limitations. For instance, they cannot predict the evolution of two-dimensional (2D) features of more complex geometry, such as the asymmetric, i.e. machined at oblique jet incidence (α < 90°), masked micro-channels that are required to fabricate suspended micro-features, such as cantilever beams [4]. Furthermore, the majority of these models ignore mask wear, particle mask-to-target ricochet and second strike, i.e. particle ricochet from the target with a subsequent second impact on the surface, effects, and do not consider curvature-based surface evolution. Finally, presently utilized computer models that account for some of these effects by tracking individual particles are relatively slow and cannot readily be made more computationally efficient. It is the aim of the proposed research to develop more efficient and generally applicable computational models that are able to address these shortcomings so that they could be used in the industry to predict the surface evolution of complex features made using AJM under a wide variety of jet and impact conditions.

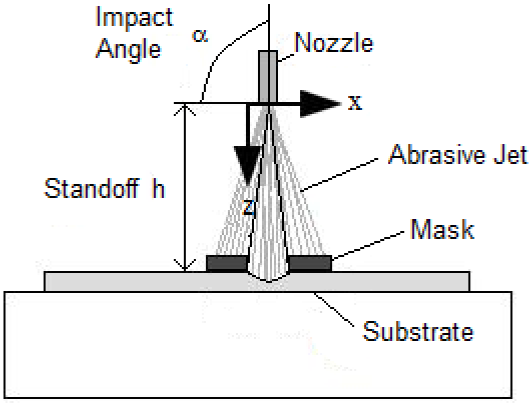


Figure 1.1. Depiction of the AJM process.

1.2 Literature review: Abrasive jet micromachining modelling and level set methods

In this section, a critical literature review is presented to provide a succinct summary of existing AJM modelling efforts and to highlight areas that require further improvement. In addition, Level set methods (LSM), a set of robust numerical techniques for studying evolving interfaces for a variety of different settings, will be discussed, and the reason behind choosing these techniques over other available computational approaches will be highlighted.

1.2.1 Analytical and semi-empirical/computational AJM modelling

Fundamental analytical and semi-empirical models

Analytical models for profile evolution in masked glass (brittle) substrates were first developed by ten Thije Boonkkamp-Jansen [1] and Slikkerveer and in't Veld [13]. The models assume that the surface evolution depends on the relationship between the erosion rate, E_r, the amount of substrate mass removed per mass of abrasive media used, the local impact angle, α, and the particle velocity. The interdependency of these parameters generally is considered constant for a given 'erosive system', consisting of a given erosive particle size and shape distribution, and a given substrate [14]. For brittle erosion, which is characterized by deformation wear and fracture, the erosion rate can be assumed to increase with increasing impact angle, reaching its maximum at normal incidence [1,14,15] (Figure 1.2). In these early surface evolution models, the local surface evolution velocity as the surface erodes could thus be related to the local surface slope, the particle mass flux and the velocity. The result was a partial differential (surface evolution) equation that could be used to predict the shape and size of AJM features as a function of time [1]. To model the decrease in particle flux near the mask edge due to particle-to-mask collisions, a first order approximation, i.e. a linear decrease, was assumed [1]. Moreover, the models did not consider the effect of local surface curvature on surface evolution. The models predicted the evolution of low aspect ratio features (< 0.5) fairly well, although the resulting channel and hole profiles had unrealistically sharp cusps at their centres that over-predicted the experimentally measured profile depths. In addition, both models were developed for masked features only, using constant particle velocity and simplified linear particle flux distributions. For unmasked features, the particle flux and spatial distributions play a fundamental role in feature evolution. The effects of the particle size distribution which affect the mass flux near the mask edge were ignored by the model of [13], whereas the model of [1] ignored the effect of secondary particle impacts, which will be discussed in detail below. Both models ignored mask wear, which can also affect the resulting surface evolution of the target.

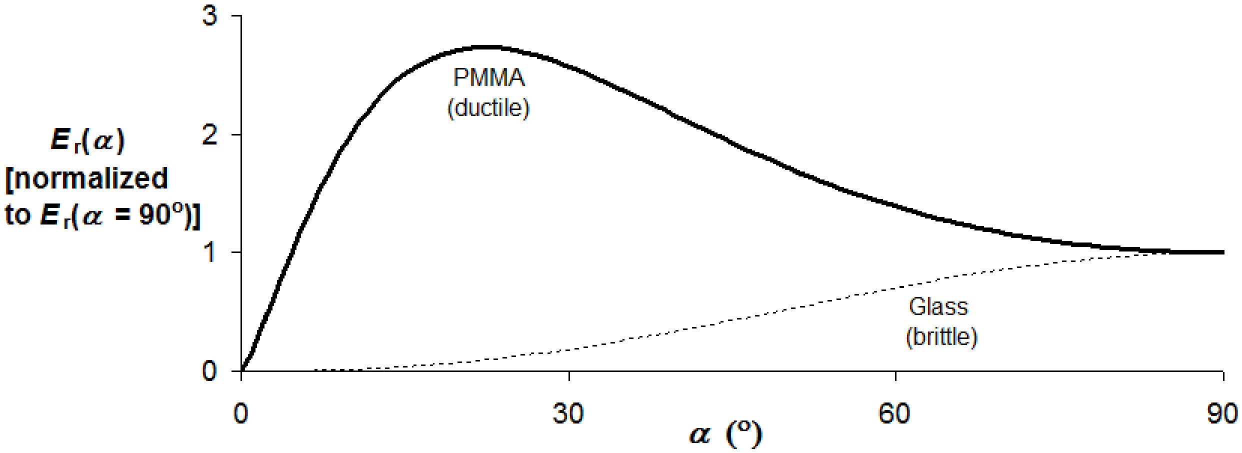


Figure 1.2. Schematic of typical brittle and ductile erosion rate, E_r, as a function of α.

A semi-empirical analytical surface evolution model for brittle materials, based on the model of ten Thije Boonkkamp-Jansen [1], which accounts for some of the above mentioned limitations, was recently developed by Ghobeity et al. [14]. The governing equation describing the surface evolution for a symmetric hole or channel cross-section was given by as [1,14]

$$z_t - \frac{C}{\rho} V(x)^{k_v} \phi(x)(1+{z_x}^2)^{-k_v/2} = 0 \tag{1.1}$$

where z_t and z_x are the partial derivatives of the profile depth z with respect to exposure time t and the coordinate x, defining the width dimension over which the profile is expected to develop. $\phi(x)$ and $V(x)$ are the particle mass flux and velocity distributions, respectively, C is constant for a given erosive system, k_v is the velocity exponent and ρ is the density of the substrate. For a scanning nozzle, the particle mass flux and velocity distributions in eq. (1.1) are also a function of time and scanning speed v_t. It was shown that the eroded profile generated by a scanning nozzle is very close to that generated across the diameter of a stationary nozzle, if the scanning speed is relatively high. This effect was modelled by fitting an empirical exponential function for the net erosive power, the product of velocity to the power k_v and the particle mass flux, in eq. (1.1) from the experimentally obtained first pass profile for both masked and unmasked channels. This approach made the analysis less complicated but deviated from a physical model to a more empirical one. In addition, the model could not be extended to the oblique incidence case for masked substrates, and it neglected secondary effects, such as the effects of mask wear and particle secondary strikes. The predictions of the model compared much better to experimental profiles than the model of [1] for aspect ratios up to 0.5 but still exhibited the cusp at the centre of the channels.

The model of [14], developed for brittle erosive systems, was extended to ductile systems such as the AJM of masked and unmasked channels and holes in PMMA [2]. For ductile erosion, which is characterized by cutting wear, the erosion rate is highest at oblique impact angles [2,15] (Figure 1.2). Ductile erosion rate and system depends on the same parameters and consists of the same components, respectively, as in the brittle case discussed above. A semi-empirical function, $g(\alpha)$, describing the dependence of erosion rate on the angle of incidence for ductile materials was incrorporated into the equation of motion in [2], which can be obtained by multiplying the second term on the left-hand side of eq. (1.1) by $g(\alpha)\,(1+z_x^2)^{(k_v+1)/2}$. The model predicted the centreline depth up to an aspect ratio of 0.6 for masked channels and 0.25 for unmasked channels, but it successively over-predicted the channel width with each pass [2]. The model had the same limitations as that of [14].

The models of [2] and [14] were improved by adding a smoothing parameter into the governing equation defined by eq. (1.1), equivalent to a viscosity term [16], to regulate the predicted speed of surface evolution in regions containing elevated curvature for masked and unmasked holes in glass [17]. This resulted in a significant improvement in feature shape prediction by smoothing out sharp corners up to an aspect ratio of 1 [17]. The smoothing term is a free parameter which is used in numerical models to stabilize the convergence of iterative solutions of partial differential equations [16]. In a related work, Moktadir et al. [18] derived a continuum equation based on the change of surface free energy upon particle impact that included this effect of curvature smoothing in masked glass substrates. However, the model could only be used to predict the qualitative shape of machined features, but not the surface evolution as a function of time.

Ghobeity et al. [19] further extended the models of [2], [14] and [17] by developing a relatively simple analytical model to estimate the spatial distribution of erosive power, or erosive efficacy, across the mask opening in the machining of micro-holes and micro-channels in glass and PMMA. The model allows the erosive efficacy close to the mask edge to be obtained as a function of the measured normal and log-normal particle size distributions and the size of the mask opening for normal incidence cases. The results showed good agreement with experimental channels in glass up to an aspect ratio of 1 and in PMMA up to an aspect ratio of 0.2. The poorer fits for higher aspect ratios were likely caused by the inability of the model to account for the particle second strike effect present during machining of glass channels and particle embedding present during machining of PMMA channels [2,8,17]. The model is quite powerful as it allows the surface evolution of micro-features to be predicted without using semi-empirical or computer particle tracking techniques. However, it was only derived for normal incidence cases and did not account for mask wear.

Analytical and semi-computational models and second strike, mask edge and spatial hindering effects

Wensink et al. [20] and Wensink and Elwenspoek [21] showed that different shapes develop as the erosion profile is machined in masked (brittle) glass channels with aspect ratios up to 2.5. Firstly, a bowl shape develops into a 'V' shape, after which necking occurs, which results in a wider pocket, or an 'udder shape' near the bottom of the profile for AR > 1 [13,21], as shown in Figure 1.3. They explained that the first two shapes originated from a 'blast lag' effect which caused narrower features to travel less deep when compared to wider ones. They hypothesized that this resulted from the characteristics of brittle erosion and from the more rapid formation of sloped sidewalls, and hence the 'V' shape, in narrower features due to the inability for particles to impact the target near the mask edge walls, both of which decrease the erosion rate. However, this could also result from the fact that as the profile deepens, the 'neck' prevents larger particles from reaching the bottom of the narrowing profile and collide with the side walls instead (spatial hindering effects) [13,22,23]. In addition, the udder shape results from the fact that as the profile sidewalls become highly sloped, particle ricochets from the sidewalls and consequent second strikes [13,21-24] at the centre of the target are made possible, as shown in Figure 1.3.

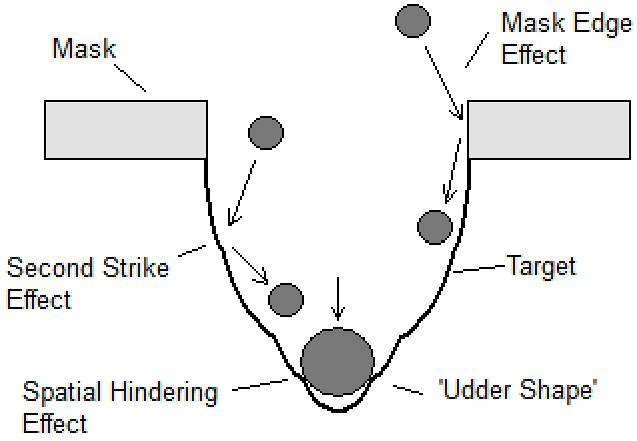


Figure 1.3. Depiction of the second strike, mask edge and spatial hindering effects in glass at normal incidence ($\alpha = 90°$).

This so called 'second strike' effect was initially modelled by Slikkerveer and in't Veld [13] and implemented into their analytical surface evolution model which was described above in *Fundamental analytical and semi-empirical models* section. This was done by using ray tracing to calculate approximate particle trajectories, thus estimating the loss of energy and the rebounding angle of initially striking particles. They found that the model was able to predict the udder shape, but it significantly over-predicted the measured profile depths and the resulting profiles had unrealistically sharp cusps at their centres, as mentioned before. The poor fits were attributed to spatial hindering effects originating from the failure of larger particles to reach the bottom of narrowing deep profiles [13,22,23], as depicted in Figure 1.3. The model provided clear evidence of a second strike effect causing the udder shape. However, in addition to not modelling the particle spatial hindering effects, it did not account for the mask-to-target particle ricochet effect, i.e. mask edge effect, shown in Figure 1.3.

More recently, Ghobeity et al. [22] developed a computer simulation which also utilized numerical ray tracing to model second strike, and incorporated it into an analytical surface evolution model. The model incorporated an 'effective particle flux', extracted from the computer simulation, accounting for the mask edge and second strike effects, and was able to fairly accurately predict the centre depth of the resulting channel profiles. However, although the results showed significant improvements over the model of [13], the model could not predict the udder shape for higher AR, and the unrealistic sharp cusps remained. This was likely because the analytical model did not consider curvature smoothing, spatial hindering effects, and it ignored the effects of mask wear.

Semi-empirical model and mask erosive wear

Several investigators have studied the solid particle erosion of masks used in AJM. For example, Wensink et al. [25] measured the erosion rate of elastic negative resist foil, polyimide, steel and electroplated copper masks under typical AJM conditions, and found that the metallic masks eroded at a lower rate than the polymeric ones. The electroplated copper mask, which was quite thin (~50 μm), was found to provide both good wear resistance and feature edge definition down to a minimum feature size of < 50 μm in glass. For machining deeper features, the effect of mask wear would have likely been much more pronounced. Achtsnick et al. [26] reached similar conclusions by studying the erosive characteristics of 8 metallic, elastomeric and photo-resistive masks. They also found that although elastomeric masks can be used to produce relatively small features (~75 μm) in glass, they are relatively thin (50~100 μm) and prone to elastic deformation, and thus they can only be used to produce a limited range of AR. High AR (>2) features can be achieved through the use of thicker and inherently tougher steel shadow masks [14]; however, they are prone to under-etching, caused by particles entering between the substrate and the mask [26]. More recently, the use of a SU8/PDMS masking technique [27,28] has led to an improvement in attainable AR (~2) and minimum feature size (~30 μm) in glass targets. However, these masks are also quite thin (~50 μm), are difficult to apply and involve a significant capital investment, when compared to standard masking techniques.

In spite of its significant effect on the resolution of features machined using AJM, the modelling of mask wear has thus far been very limited. Although Slikkerveer et al. [29] derived a semi-empirical model that allowed the worn shape of masks made from three elastomeric materials to be predicted, the model was applied to the mask alone, independent of the target erosion, and thus only qualitative conclusions of the influence that mask wear had on surface evolution of the target could be drawn.

Analytical and semi-empirical models and oblique incidence

All of the above mentioned models have been derived for normal incidence cases only. Due to its complexity, the oblique erosion process in AJM has not been studied as extensively. Belloy et al. [30] used AJM to machine low AR (< 0.33) masked glass holes using aluminum oxide particles at oblique angles of attack. They extrapolated curves from experimental profiles in at attempt to quantify the under-etching effect caused by the impact of secondary particles as function of the angle of incidence and etching time. These curves were then used to formulate a simple coordinate transformation, i.e. a correction of each point, from the normal incidence case to oblique case that would be used to match the experimental profiles in the best way possible in order to try to quantify the under-etching effect. This semi-empirical model was not a physical predictor of the oblique erosion process, i.e. it did not model the surface evolution, and the fits to experimental profiles were poor. Park et al. [24] also studied the effect of under-etching in masked glass samples machined at oblique incidence, and schematically showed that mask wear and second strike effects discussed above can have a significant effect on the resulting shape of the machined features. More recently, Getu et al. [31] extended the work of [14] by studying oblique erosion in PMMA as well as in polymers such as LUCITE, an acrylic, and LEXAN, a polycarbonate, in masked and unmasked holes and channels. The analytical model used the technique of [14] by fitting the erosive power distribution across the mask opening with a polynomial function using the first pass experimental profile. The work only considered oblique erosion by varying the angle of incidence along the nozzle centreline and parallel to the scanning direction plane, which would result in symmetrical as opposed to asymmetrical cross-sectional profiles. Asymmetrical profiles are only obtained by varying the angle of incidence in the plane of the channel cross-section and perpendicular to the scanning direction plane. The authors were able to quantify the effect of particle embedding that is present in micro-machining of polymers under certain conditions and found a relation for a net embedding energy flux as a function of scanning direction distance and angle of incidence. However, the model had the same limitations as those in [14].

All of the above mentioned analytical and semi-empirical/computational models cannot be used to predict the evolution of features which have two or more depth values at a given profile location along the width. Such 2D and 3D multi-valued profiles develop, for example, in the AJM of masked micro-channels and holes at oblique jet incidence, which are of interest in the AJM of suspended components, such as micro cantilever beams [4]. More sophisticated techniques must be employed to solve these multi-valued partial differential equations. These will be discussed in the following sections.

1.2.2 Traditional interface tracking techniques and cell-based methods

Traditional interface tracking techniques

The marker/string method and the volume-of-fluid technique are commonly used to numerically track interfaces [16]. The marker/string method uses a Lagrangian approach where the boundary of the interface, i.e. the evolving surface, is discretely parameterized. Surfaces are computed via the use of marker particles in 2D and via nodal triangularization of the interfaces in 3D. The locations of the nodes are updated by obtaining data about the normals and curvature from the representation of the markers. This method can be quite accurate; however, it gives inaccurate solutions when corners and cusps develop in the evolving front and has difficulty in handling complex interface changes. In addition, although the method has been applied to track interfaces in 3D, its extension from 2D to 3D is difficult [16].

The volume-of-fluid technique utilizes a computational grid which is divided into cells, each of which contains ‘volume fractions’ in the range from 0 to 1, which signify the fraction of each cell containing material within the interface [16]. The front is propagated forward in time under a ‘transport velocity’ in each coordinate direction by reconstructing the front based on these volume fractions. This method can handle interface changes with ease and can be extended through adaptive methods to solve 3D problems. However, precise calculation of geometric properties such as the surface normal and curvature can be difficult [16].

Cell-based methods

More recently, a 3D cellular automaton (CA), cell-based computational method used to model AJM has been applied to obtain the shape of a glass channel along with the encompassing polymeric mask [32]. The CA algorithm was composed of combinations of two orthogonal planes of 2D cells along with a solid particle erosion model. The predicted eroded channel and worn mask shapes using both low and high resistant masks showed good qualitative agreement with experiments but no quantitative comparisons were made. In addition, the simulation did not account for a non-uniformly distributed particle flux, oblique incidence, particle scattering and second strike effects, as well as the effect of interference between incident and rebounding particles which becomes important when considering high flux cases [33-35]. A computer simulation for modelling surface evolution in AJM, which accounts for some of these limitations, was recently developed by Shafiei et al. [36]. The simulation can predict the size and shape of resulting unmasked hole and channel profiles as a function of all the process parameters. Target propagation was obtained by discretizing the surface to form a 3D grid of cubic cells, each of which was assigned a 'damage limit' based on the number of particle impacts it experienced. The simulation can be adopted for any substrate and can simulate high flux cases by tracking individual particles and implementing particle-particle and particle-surface collision detection and kinematics. The simulation showed excellent agreement between predicted unmasked profile shapes and experiments at low fluxes and fair agreement at high fluxes, i.e. it overestimated the feature depths.

Most recently, Ciampini and Papini [23] developed a CA-based model for the prediction of the AJM of masked features which included algorithms for launching, tracking, and collision detection of non-uniform particle size distributions. Their model implemented a better algorithm for surface damage distribution than that used in [36], resulting in very good agreement with measured profiles for low and high aspect glass micro-channels and micro-holes. It was the first model to account for all of the effects of second strike, mask edge and spatial hindering described in Section 1.2.1. However, although the CA method is quite powerful since it can emulate real-time conditions, it is very complex and difficult to implement in terms of coding, and is thus less accessible to the machining community. Moreover, the finite size of the cells in this method makes it difficult to calculate geometric quantities such as the surface normal and curvature at sharp corners and complex geometrical shapes. Most importantly, to be effective, this method must incorporate algorithms to track many particles simultaneously [23], making it computationally very expensive, especially at high fluxes and in simulating large masked feature depths. On a standard PC, highly optimized CA/particle tracking simulations may take more than 15 hours to complete, for even low flux cases [23]. In addition to all of this, none of the above CA models considered surface evolution of masked features that result from AJM at oblique incidence.

1.2.3 Level set methods and their advantages over other computational techniques

Level set methods

Level set methods (LSM), developed by Sethian and Osher [16,37], are powerful computational techniques for analyzing and obtaining the evolution of dynamic fronts for a multitude of different situations. LSM numerically estimate the governing equations of motion for a moving front by transforming them into a distinctive solution of an initial value partial differential equation (PDE). LSM are based on a Hamilton-Jacobi type equation for the level set function, and utilize methods formed for solving hyperbolic PDEs which rely on the connection between front propagation and hyperbolic conservation laws [16]. Fundamentally, LSM abandon the Lagrangian (local) geometric perspective for solving interface problems and move towards the Eulerian (global) perspective. They rely on viscosity solutions for fitting PDEs to update the location of the interface, using the velocity of the front [16,37].

LSM for evolving interfaces can be particularly useful for profiles that can: develop sharp corners and cusps where singularities can form, undergo major changes in topology, e.g. where there is a responsive dependence on the direction of the normal to the front and on the surface curvature or when surfaces merge or break apart, undergo sensitive variations in the speed of propagation, and experience complex motion in 3D. Hence, the method is versatile and stable for the formulation of arbitrary geometries. LSM are utilized in computer vision, material science, image processing, computational fluid dynamics, micro-fabrication, i.e. etching, deposition and lithography, and many other fields. The use of LSM for simulating the evolution of machined profiles in AJM was first suggested by ten Thije Boonkkamp [1], but no work on LSM implementation in AJM existed in the literature before the work of the present dissertation. With respect to LSM implementation in the different research fields, the major difference is only the definition of the velocity of surface propagation. As a result, LSM has become a general computational method for solving arbitrary interface propagation problems, and hence it can be extended to AJM [16,37].

The main principle behind LSM is to represent the location of a surface at a particular time t as a zero level set, of a particular implicit function $\Phi(\vec{x},t)$, the level set function, where the initial surface is defined by $\{\vec{x} \mid \Phi(\vec{x},0)=0\}$. The variable $\vec{x}$ can represent all spatial variables, i.e. x, y and z, including the location of the surface, which is embedded in the level set function, and the arrow head indicates a vector quantity. Thus, the desired surface location is implicitly defined by $\Phi(\vec{x},t)$, which has one more dimension than the surface. Surface movement with time results from, for instance, a driving physical force or flux, whose effect is defined by a local surface normal velocity function, which depends on the physics of the particular problem. The local normal velocity of any point on the surface, $F(\vec{x},t)$, depends on both spatial and time variables, and is assumed to be valid for the entire computational domain, not just on the surface. As $\Phi(\vec{x},t)$ evolves in time and the surface propagates, for $t > 0$, it becomes the zero level set of $\Phi(\vec{x},t)$, and it can be defined as a set of points $\{\vec{x} \in \Re^n \mid \Phi(\vec{x},t)=0\}$ (see Figure 1.4).

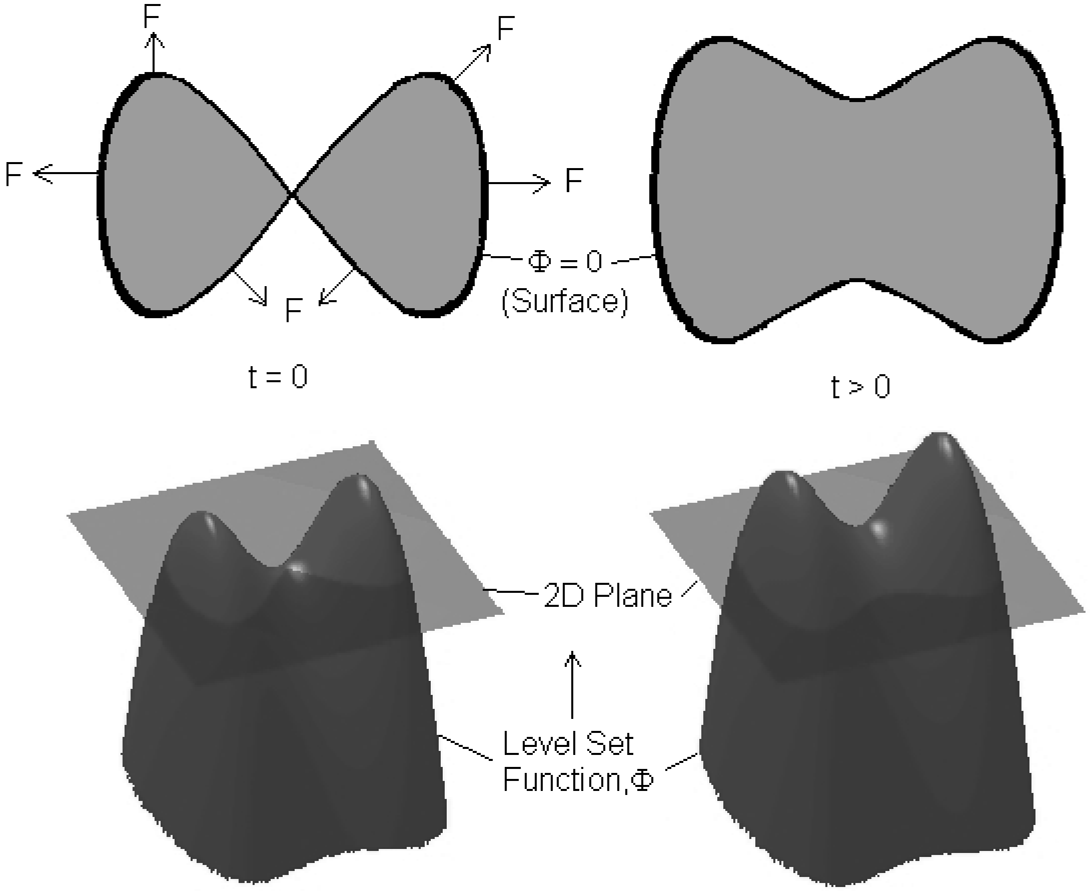


Figure 1.4. Depiction of the level set method (LSM). In LSM, a level set function, $\Phi(\vec{x},t)$, evolves in time and the surface location and shape can be determined by the zero level set, $\Phi(\vec{x},t) = 0$ (in the figure, the intersection of $\Phi(\vec{x},t)$ and the 2D plane). The original image taken from [38] has been modified.

The level set equation is defined as [16,37]

$$\frac{\partial \Phi}{\partial t} + F(\vec{x},t)\,|\nabla\Phi| = 0 \tag{1.2}$$

Since physical models derive the velocity function only at the zero level set, i.e. the surface is located where $\Phi(\vec{x},t) = 0$, it has to be extrapolated appropriately at grid points that are not adjacent to the zero level set. The level set equation can then be iteratively solved to obtain surface profiles for $t > 0$.

Numerically, $\Phi(\vec{x},t)$ is represented by its values on the grid nodes, and the computational domain is approximated by a spatial grid. LSM uses finite differences, FDs, to approximate the solution to $\Phi(\vec{x},t)$ and vital geometric properties, such as the surface normal and curvature. It uses upwind FD schemes to ensure that FD approximations follow the exact-solution PDE theory so that numerical solutions can converge correctly [16,37]. In addition, LSM can be made more computationally efficient through adaptive strategies [16]. These aspects will be explored in more detail in later chapters.

Advantages over other computational techniques

LSM can overcome all the limitations of traditional interface tracking techniques discussed in Section 1.2.2 since it can handle singularities and complex changes in topology. Intrinsic geometric quantities such as the surface normal or curvature can be easily obtained from the level set function and LSM can be readily extended from 2D to 3D by simply extending the size of arrays and gradient operators. Moreover, LSM can overcome all the limitations of cell-based methods since they are less complex and much easier to implement and are more computationally efficient. In addition, geometric quantities can be easily calculated as mentioned above. As a result, LSM were chosen over cell-based methods and traditional interface tracking techniques to model surface evolution in AJM.

1.3 Objectives, significance and organization

The proposed dissertation will utilize LSM to develop surface evolution models in AJM based on less simplifying assumptions than previous analytical and semi-empirical/computational models discussed in Section 1.2.1. These models would also address the limitations of previous computational models discussed in Section 1.2.2. Specifically, the main objective of the research is to develop 2D LSM models, which are relatively easy to implement as well as computationally efficient, for the surface evolution of both brittle and ductile materials during the AJM process. The models would be able to predict 2D features of complex geometry, as well as account for mask wear [24,29,32] and particle behaviour near the mask edge and the second strike effect [13,21-24]. The models will be verified by comparison with experimentally determined AJM feature profiles. Ultimately, such models could be used to perform parametric studies, on how the process parameters such as the standoff distance, angle of incidence, scanning speed, width and height of the mask, etc. affect the resulting feature shapes. This would allow the optimization of AJM operations. The main objective will be accomplished by achieving the following secondary objectives:

1. The development of a 2D LSM numerical model for the AJM of masked and unmasked holes and channels in glass (brittle material) and polymethylmethacrylate (PMMA) (ductile material) at normal incidence ($\alpha = 90°$) and unmasked channels in glass and PMMA at oblique incidence ($\alpha < 90°$). For the masked cases, only ARs of up to approximately 1 will be modelled in this preliminary step. These situations, which do not include secondary effects such as mask wear [24,29,32] and particle second strike [13,21-24], have already been modelled analytically in, e.g. [2,14,17], and thus can serve as a foundation for achieving the main objective and as a means to evaluate the level set methodology, which has not been previously used in modelling the AJM process. This will be considered in Chapter 2.
2. Extension of the LSM model described in objective 1 for the AJM of masked channels in glass and PMMA at normal incidence to oblique incidence. This model will demonstrate the power of LSM for solving the multi-valued PDEs that result from oblique incidence AJM of masked features, which has not been previously considered. This step will also include optimization of the LSM model to increase computational efficiency by utilizing the Narrow Band (NB) LSM [16]. The NB LSM [16] is an adaptive scheme which is based on the notion that calculations need not be performed for points far away from the interface, and hence only for points in its vicinity; hence, up to an order of magnitude decrease in execution time can result [16]. This will demonstrate that the LSM approach is more efficient than other computational approaches in

solving the AJM problem, which would also make it possible to perform parametric studies with more ease. This will be considered in Chapter 3.

3. Extension of the LSM models described in objectives 1 and 2 for the AJM of masked channels in glass and PMMA at any incidence to include mask erosive wear [24,29,32], by modelling the surface as a composite material, where portions of the surface represented by the mask have different material properties than portions represented by the target, and hence erode differently. This approach has never been used in modelling the AJM process, and the effect of mask erosive wear on the development of target features has only been studied at normal incidence [29,32]. This will be considered in Chapter 4.
4. Extension of the LSM models described in objectives 1 to 3 for the AJM of masked channels in glass at any incidence to features having an AR > 1, when the effects of second strike [13,21-24] must be included. These effects will be included by extending the Slikkerveer and in't Veld [13] model of second strike from normal to oblique incidence, which has not been previously considered. It will involve the use of ray-tracing and expressions for reduced rebound velocity and rebound angle, based on the point of impact. This will be considered in Chapter 5.

Chapter 2 Level Set Methods for the Modelling of Surface Evolution in Abrasive Jet Micromachining: Foundational Model

2.1 Motivation

In this chapter, a new methodology for the prediction of surface evolution in AJM based on LSM [16,37] is introduced. It has the potential to address shortcomings of existing analytical and computer models, as explained in Chapter 1, so that it can be used to predict the surface evolution of complex multi-valued AJM features using a wide variety of jet and impact conditions. To demonstrate its feasibility, the methodology is used to predict the surface evolution of unmasked micro-channels machined at normal and oblique jet incidence, and masked micro-channels and micro-holes at normal incidence, in both glass and PMMA, assuming conditions in which secondary effects such as mask wear [24,29,32] and second strike [13,21-24] are neglected. The level set predicted eroded profiles are compared to those experimentally obtained, as well as to those predicted by existing analytical and computer models. The advantages of the current level set methodology over previous modelling efforts are discussed. The majority of the material in this chapter has been published in [39].

2.2 AJM experiments: Unmasked and masked channels

All AJM experiments were conducted using a round 0.76 mm inner diameter nozzle fitted to a commercial microblaster (MB 1005 Microblaster, Comco Inc., Burbank, CA, USA) into which a mixing device was incorporated to prevent particle bed compaction [2]. The blasting pressure was held constant at 200 kPa, and the maximum jet centre velocity of the 25 μm (nominal diameter) alumina powder at the utilized nozzle to target standoff distance, h = 20 mm (Figure 2.1), was approximately 162 m s^{-1}. The 25 μm aluminum oxide particles are the most widely used abrasives because of their excellent hardness and angular shape, which promotes high erosion rates [2,14].

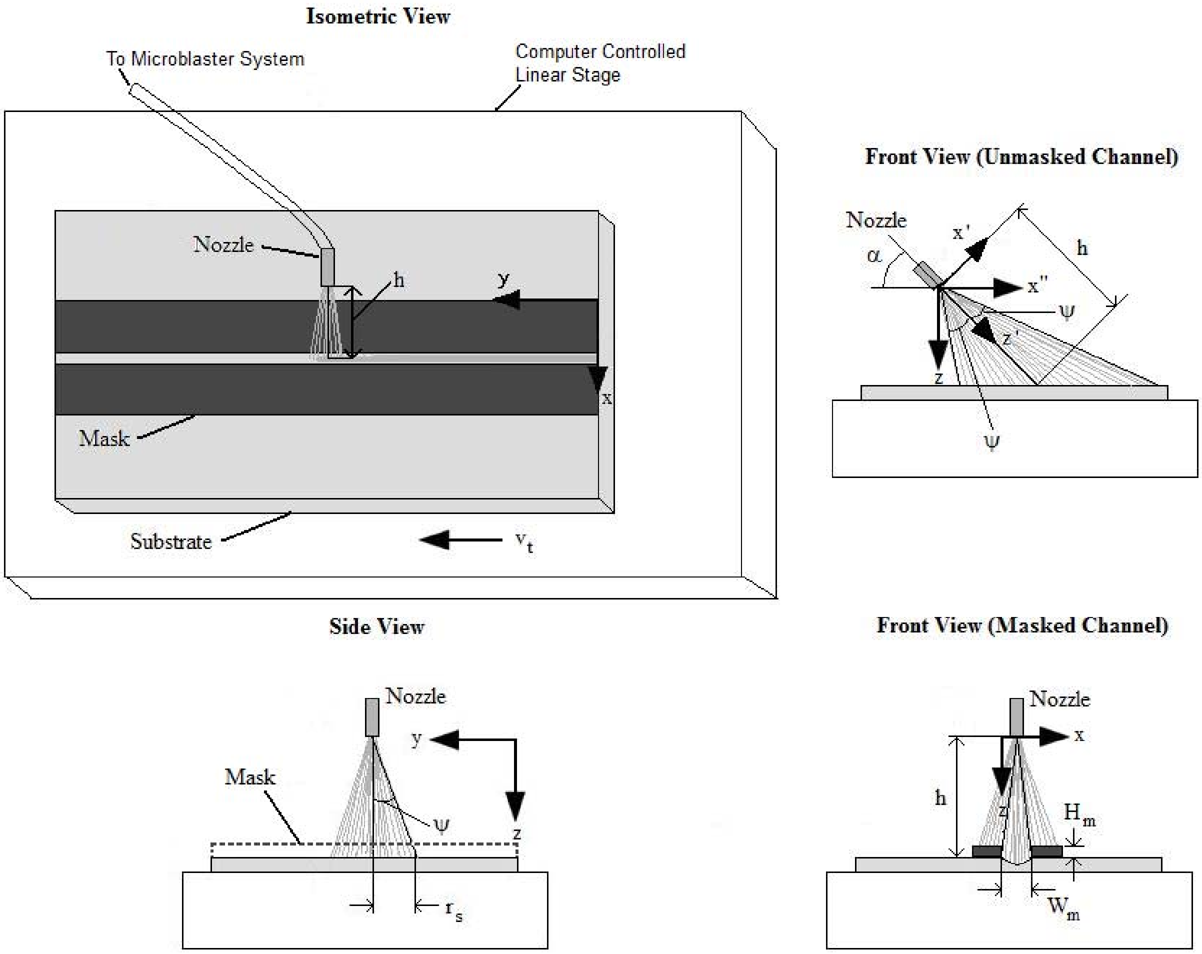


Figure 2.1. Schematic of AJM channel blasting apparatus for the experiments under consideration.

Unmasked and masked channels were machined in 5 mm thick Borofloat (Schott North America Inc., Elmsford, NY, USA) glass plates (density, ρ = 2200 kg m^{-3}), and 1.5 mm thick PMMA (ρ = 1190 kg m^{-3}) sheets (Piedmont Plastics Inc., Brampton, ON, Canada). Glass is the most common material utilized in microfluidics and MEMS applications [6] due to its hardness, transparency and ease of machinability. In AJM, it represents a brittle erosive system having a maximum erosion rate at normal incidence (α = 90°), when the nozzle is perpendicular to the surface (Figure 2.1) [24]. Its erosive behaviour is similar to that of silicon. PMMA is one of the polymeric materials commonly used in microfluidics and MEMS applications [9]. In AJM, it represents a ductile erosive system in which the erosion rate increases with the impact angle to a maximum value of approximately 15–40°, after which it begins to rapidly decrease [2,24] (Figure 1.2). The AJM process models developed for these two materials are thus applicable to a wider class of brittle and ductile erosive systems by simply specifying different erosive system parameters.

The target samples were mounted on parallel screw mounts and clamped to a programmable computer controlled linear stage having an accuracy of 0.5 μm (Aerotech, Pittsburgh, PA, USA) which moved relative to the stationary nozzle. Scanning speeds of 1 mm s^{-1} and 0.5 mm s^{-1} were used in machining the glass and PMMA channels, respectively, over a 15 mm length. The utilized scanning speed ensured that particle embedding [2,8,17,31] and temperature, i.e. target surface heating effects in PMMA were minimized, as in [2]. Nevertheless, even if surface heating had existed, its effect would be accounted for in the surface evolution because, as explained in the next section, the fundamental erosion rate used as an input in the surface evolution model is obtained under the same conditions as the machining. In addition, these scanning speeds ensured that appreciable slopes along the channel length are avoided in both materials. For slow scanning speeds and high erosion rates, i.e. higher mass fluxes, an appreciable slope in the scan direction at the leading edge of the jet can develop. This may affect the local erosion conditions at a particular channel cross-section, and thus the effective erosion rate may change [14]. This is important when modelling a scanning nozzle in 2D, as in the present case. As a result, the scanning speeds were balanced to ensure this was avoided but at the same time guaranteed an efficient erosion rate. All unmasked channels were machined at incidence angles of 90°, 60° and 30° and masked channels at incidence angles of 90°. The average abrasive particle mass flow rate was measured 3 times before and after each channel was machined, by weighing with a micro mass balance of accuracy = 0.01 mg the amount of powder blasted for a minute into a special container that was covered with filter material which prevented the particles escaping, while also preventing back pressure. The measured mass flow rate, i.e. mass flux, range of 2.16-5.38 g min^{-1} (see Appendix A, Tables A-1-A-8) was sufficiently low to ensure that particles rebounding from the surface did not interfere with incoming particles [33-35].

The mask was made by securely taping two parallel tempered steel (Starret Co., Athol, MA, USA) feeler gages to the workpiece a known distance apart, aligned with a reference gage. The mask width, W_m, and height, H_m (Figure 2.1), were approximately 400 μm and 100 μm, respectively. To ensure that the mask adhered to the surface during blasting, an industrial magnet spanning the length of the channel was attached and secured below the 1.5 mm thick PMMA workpiece, and a rare-earth magnet was attached below the 5 mm thick glass workpiece.

The cross-sectional profiles of the machined channels were measured with a non-contact optical profilometer (Nanovea ST400, Micro Photonics Inc, Allentown, PA, USA), which was accurate to within a depth of 10 nm. Between 200-850 and 350-450 data points were obtained over scanning width ranges of 6-10.5 mm and 0.65-0.725 mm for all the unmasked and masked channel profiles, respectively.

2.3 Level set modelling of surface evolution in AJM

2.3.1 Transformation of coordinates

In order to model features machined with the jet at oblique incidence as shown in Figures 2.1 and 2.2, the coordinate system used to describe previous analytical 1D models [1,2,14] of surface evolution described in Section 1.2.1, developed for normal incidence, was modified. The original $x_o - z_o$ axis was moved up to the nozzle and rotated about the y-axis by the angle α (Figure 2.2), to define a new x'- z' axis. To be consistent with the LSM formulation, using geometry, the local x' and z' coordinates can be expressed in terms of the global x and z coordinates as follows

$$\begin{aligned} x' &= x'' \sin\alpha - z\cos\alpha \\ z' &= x'' \cos\alpha + z\sin\alpha \\ x'' &= x - x_{\mathrm{off}} \end{aligned} \tag{2.1}$$

where x_{off} is the offset distance between the global and nozzle axes (Figure 2.2), necessary to ensure generality for both oblique and normal incidence. For the unmasked case, $x_{\mathrm{off}} = h\tan\psi$, where ψ is the maximum jet spread angle (Figure 2.1) [34]. In the equations that follow, x'' will be used instead of $x - x_{\mathrm{off}}$ for brevity.

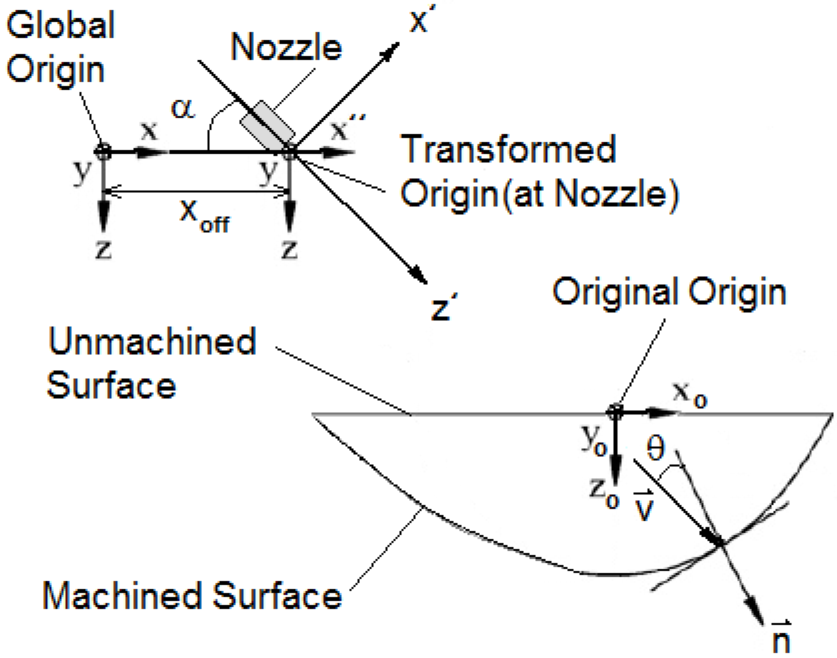


Figure 2.2. Schematic of coordinates used for unmasked channel and/or hole cross-sectional profiles.

2.3.2 Derivation of local normal velocity of evolving surface for unmasked channels and holes

For the AJM of brittle targets such as glass, the velocity of the surface in the direction of the local normal required for the solution of eq. (1.2) can be expressed as [1]

$$F_{\mathrm{b}}(x,y,z,t)=\frac{E_{\mathrm{r,b}}(x,y,z,t)}{\rho}(\vec{\phi}(x,y,z,t)\bullet\vec{n}) \tag{2.2}$$

and for ductile targets, such as PMMA, as [2]

$$F_{\mathrm{d}}(x,y,z,t)=\frac{E_{\mathrm{r,d}}(x,y,z,t)}{\rho}\phi(x,y,z,t) \tag{2.3}$$

where $\phi(x,y,z,t)$ is the particle mass flux, the mass of particles per unit time arriving to a unit surface area at a given spatial location on the surface, $\vec{n}$ is the surface normal (Figure 2.2), and $E_{\mathrm{r}}(x,y,z,t)$ is the erosion rate, the mass of target material removed per mass of incident particles. The subscripts ‘b’ and ‘d’ stand for ‘brittle’ and ‘ductile’, respectively.

The brittle erosion rate can be expressed as [1,14]

$$E_{\mathrm{r,b}}(x,y,z,t) = C(\vec{V}(x,y,z,t) \bullet \vec{n})^{k_{\mathrm{v}}} \tag{2.4}$$

where $V(x,y,z,t)$ is the distribution of particle velocities incident to the surface, k_{v} is a velocity exponent and C is an empirical erosion constant which can be obtained by fitting the modelled erosive efficacy (see Section 2.4.1.1) to the experimentally obtained first pass profile [14,36]. C and k_{v} generally depend on particle and target hardness and toughness, as well as particle size, type and velocity. k_{v} can be obtained by performing erosion test at various angles of attack and curve fitting the results [14].

For ductile erosion, a more complex relationship between erosion rate and angle of attack exists. Following Getu et al. [2], the angular dependence of erosion due to Oka et al. [40] can be adopted for the AJM of many polymers such as PMMA, and E_{r} in these cases can be expressed as

$$E_{\mathrm{r,d}}(x,y,z,t) = C(V(x,y,z,t))^{k_{\mathrm{v}}} \left(\frac{\vec{V}(x,y,z,t) \bullet \vec{n}}{\left|\vec{V}(x,y,z,t)\right|} \right)^{n_1} \left[1 + Hv\left(1 - \frac{\vec{V}(x,y,z,t) \bullet \vec{n}}{\left|\vec{V}(x,y,z,t)\right|} \right) \right]^{n_2} \tag{2.5}$$

where Hv (GPa), the initial Vickers target hardness, and the constants n_1 and n_2 are determined using a microhardness tester and experiments similar to that when obtaining k_{v}, respectively [2,40]. The last two terms in the brackets in the RHS of eq. (2.5) are the surface normal and tangential components of the erosion rate [2,40].

For the AJM of holes machined at $\alpha = 90°$ using a round nozzle, V in eqs. (2.4) and (2.5) is independent of t and $y = y' = 0$ due to symmetry. For the blasting conditions used in the present work, the measured velocity distribution across the jet [14] can be expressed as

$$V(x,z) = V_{\mathrm{o}}\left(1 - 4.92\frac{x''}{z} \right) \tag{2.6}$$

where V_{o} is the maximum jet centre velocity.

For the AJM of channels at any α, the target is scanned relative to the stationary round nozzle at a scan speed of v_{t} in the positive y direction (Figure 2.1), where $y = y' = (r_{\mathrm{s}} - v_{\mathrm{t}}t)$, and $r_{\mathrm{s}} = h\tan\psi$, which is the radius of the impact area of the jet on the unmasked target surface measured in the y - z plane [14] (Figure 2.1). In one pass, each channel cross-section in the x - z plane is exposed to a jet of particles for a total time of $2r_s/v_t$ [14]. Using the coordinate system of eq. (2.1), the velocity distribution for the scanning target in eqs. (2.4) and (2.5) can be expressed as

$$V(x,z,t) = V_o\left[1 - 4.92\frac{\sqrt{(x''\sin\alpha - z\cos\alpha)^2 + (r_s - v_t t)^2}}{x''\cos\alpha + z\sin\alpha}\right] \tag{2.7}$$

The particle mass flux ϕ in eqs. (2.2) and (2.3) for holes machined at $\alpha = 90°$ under the conditions described in Section 2.2 can be obtained as [14]

$$\phi(x,z) = \frac{\dot{M}\beta^2}{\pi z^2} e^{-\beta^2\left(\frac{x''}{z}\right)^2} \tag{2.8}$$

where $\dot{M}$ is the particle mass flow rate through the nozzle and β is the 'focus coefficient' which describes the spread of the abrasive jet. Generally, β depends on the internal nozzle roughness, particle velocity and type [41] and can be obtained through mass measurements at different radial locations of the jet and curve fitting the results [14,42]. A higher β implies a more focused jet [14,41,42].

For channels machined at any α with a scan speed of v_t, the particle mass flux in eqs. (2.2) and (2.3) can be expressed in the coordinates of eq. (2.1) as

$$\phi(x,z,t) = \frac{\dot{M}\beta^2}{\pi(x''\cos\alpha + z\sin\alpha)^2} e^{-\beta^2\frac{(x''\sin\alpha - z\cos\alpha)^2 + (r_s - v_t t)^2}{(x''\cos\alpha + z\sin\alpha)^2}} \tag{2.9}$$

2.3.3 Derivation of local normal velocity of evolving surface for masked holes and channels

For masked features, particle mass flux passing through the mask and striking the surface decreases as the mask edge is approached. This occurs because only smallest incoming particles can avoid colliding with the mask edge as it is approached, as depicted in Figure 2.3. As a result, the proportion of the total particle size distribution that can pass through the mask decreases as the mask edge is approached. The analytical model of [19] for α = 90° describes this effect which depends on the particle size distribution and the size of the mask opening. According to the model, the proportion of the total incoming particle mass, M, that passes through the mask opening and arrives to the surface at a given location, can be expressed as [19]

$$M_{x/0} = \frac{M(x'')}{M(x''=0)} = \frac{\int_0^{W_\mathrm{m}/2-|x''|} r_\mathrm{p}{}^3 \Omega(r_\mathrm{p}) dr_\mathrm{p}}{\int_0^{W_\mathrm{m}/2} r_\mathrm{p}{}^3 \Omega(r_\mathrm{p}) dr_\mathrm{p}} \tag{2.10}$$

where $\Omega(r_\mathrm{p})dr_\mathrm{p}$ is the proportion of particles having a radius, r_p, between r_p and $r_\mathrm{p} + dr_\mathrm{p}$, and W_m is the width of the mask opening (Figure 2.1). x'' is defined in a similar manner as for unmasked cases (eq. (2.1)), where $x_\mathrm{off} = W_\mathrm{m}/2$ (Figure 2.1) so that $x'' = 0$ corresponds to the centre of the mask opening. For the 25 μm alumina used in the experiments of Section 2.2, the size distribution was measured as log-normal [19],

$$\Omega(r_\mathrm{p}) = \left(\frac{1}{r_\mathrm{p} \sigma_\mathrm{l} \sqrt{2\pi}} \right) e^{-\left(\ln(r_\mathrm{p}) - \mu_\mathrm{l}\right)^2 / 2\sigma_\mathrm{l}^2} \tag{2.11}$$

where μ_l and σ_l are the log-normal mean and standard deviation of r_p. This model was applied to masked holes and channels at normal incidence by multiplying the mass flux, eqs. (2.8) or (2.9), by eq. (2.10).

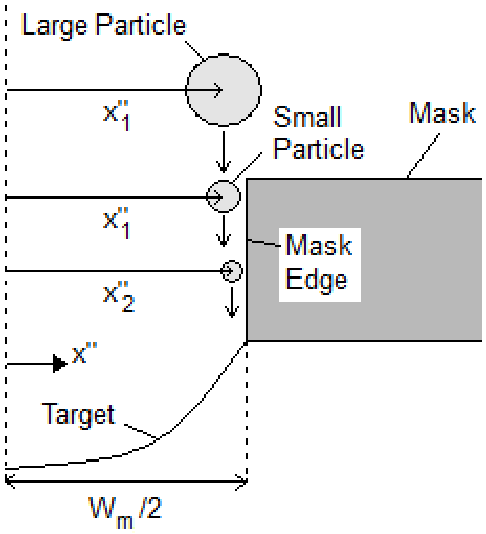


Figure 2.3. Depiction of the decrease in mass flux incident to the target as the mask edge is approached. In the figure, a small particle arriving to the target at a distance x_1'' from the centre of the mask opening will pass through the mask while a larger particle will collide with the mask edge. As the mask edge is approached for $x'' > x_1''$, e.g. when $x'' = x_2''$, only successively smaller particles will arrive to the target without colliding with the mask edge. Due to symmetry, only the right mask edge is shown.

2.3.4 Implementation of the LSM model

2.3.4.1 Finite differences and geometric variables

LSM use finite differences (FDs) to approximate the solution to $\Phi(\vec{x},t)$, the surface normals, and the curvature. Most commonly, first order FDs are used to approximate first and second partial derivatives,

$$\Phi_g^+ = \frac{\partial\Phi}{\partial g}^+ = \frac{\Phi_{l+1} - \Phi_l}{\Delta g},\ \Phi_g^- = \frac{\partial\Phi}{\partial g}^- = \frac{\Phi_l - \Phi_{l-1}}{\Delta g},$$

$$\Phi_g^c = \frac{\partial\Phi}{\partial g}^c = \frac{\Phi_{l+1} - \Phi_{l-1}}{2\Delta g},\Phi_{gg}^c = \frac{\partial^2\Phi}{\partial g^2}^c = \frac{\Phi_{l+1} - 2\Phi_l + \Phi_{l-1}}{(\Delta g)^2} \tag{2.12}$$

where g denotes the x or z spatial variable for the symmetric 2D problems considered in the present work. The superscripts '+', '-' and 'c' denote forward, backward and central FD, respectively, and l denotes the appropriate grid index (i, k) of the spatial direction (x, z) of the partial derivative. First order approximations are much simpler to implement and require less stringent boundary conditions than higher order FDs. The loss of accuracy can be compensated for by increasing the grid resolution. Using FDs, the gradient of $\Phi(\vec{x},t)$ can be defined by [16,37]

$$\nabla\Phi(\vec{x}) = \nabla\Phi(x,z) = \left(\frac{\partial\Phi}{\partial x},\frac{\partial\Phi}{\partial z}\right) = (\Phi_x,\Phi_z) \tag{2.13}$$

From this, the surface normal, $\vec{n}$, and curvature, *K*, can be obtained as [16,37]

$$\vec{n} = \frac{\nabla\Phi}{|\nabla\Phi|} = \frac{(\Phi_x,\Phi_z)}{\sqrt{\Phi_x{}^2+\Phi_z{}^2}}\ ,\ K = \nabla\cdot\frac{\nabla\Phi}{|\nabla\Phi|} = \frac{\Phi_x{}^2\Phi_{zz} - 2\Phi_x\Phi_z\Phi_{xz} + \Phi_z{}^2\Phi_{xx}}{\left(\Phi_x{}^2+\Phi_z{}^2\right)^{3/2}} \tag{2.14}$$

The motion of the surface can be regulated by multiplying *F*(*x,z,t*) in eqs. (2.2) or (2.3) by (1- εK), where ε is a free parameter that regulates, i.e. smoothes out, the predicted speed of surface evolution in regions containing elevated curvature [16,17,37].

2.3.4.2 LSM for non-convex Hamiltonians

Equation (1.2) relates the change in time to the gradient of $\Phi(x,z,t)$ using *F*(*x,z,t*) and can be recast to Hamilton-Jacobi form, and generalized to include curvature, as [16,37]

$$\frac{\partial\Phi}{\partial t} + H(\nabla\Phi(x,z)) = (\varepsilon K)F(x,z,t)\left|\nabla\Phi(x,z)\right| \tag{2.15}$$

where the Hamiltonian, *H*, is defined by

$$H(\nabla\Phi(x,z)) = F(x,z,t)\left|\nabla\Phi(x,z)\right| \tag{2.16}$$

Here, use of the term ‘Hamiltonian’ refers to the Hamiltonian function and not an operator. Combining eqs. (2.2)-(2.5), (2.10), (2.14), and (2.16), and expressing the result in the x'' and z components, the Hamiltonians for brittle erosive (glass) and ductile erosive (PMMA) targets, respectively, can be expressed as

$$H_{\mathrm{b}} = M_{x/0} \frac{C}{\rho} V^{k_{\mathrm{v}}} \phi \frac{\left(x''\Phi_x + z\Phi_z\right)^{k_{\mathrm{v}}+1} \left(\Phi_x^2 + \Phi_z^2\right)^{-k_{\mathrm{v}}/2}}{\left(x''^2 + z^2\right)^{(k_{\mathrm{v}}+1)/2}} \tag{2.17}$$

and

$$H_{\mathrm{d}} = M_{x/0} \frac{C}{\rho} V^{k_{\mathrm{v}}} \phi (\cos\theta)^{n_1} \left[1 + Hv(1-\cos\theta)\right]^{n_2} \sqrt{\Phi_x^2 + \Phi_z^2} \tag{2.18}$$

with

$$\cos\theta = \frac{x''\Phi_x + z\Phi_z}{\sqrt{\Phi_x^2 + \Phi_z^2}\sqrt{x''^2 + z^2}} \tag{2.19}$$

where θ is the angle between the surface normal $\vec{n}$ and the particle impact velocity vector $\vec{V}$ (Figure 2.2), and V and ϕ are solved with eqs. (2.6) and (2.8) for a stationary target, and eqs. (2.7) and (2.9) for a scanning target, respectively.

To determine whether H is convex, and if $F(x,z,t)$ is smooth for all time and position, the following condition must be fulfilled [16]:

$$\frac{\partial^2 H}{\partial\Phi_x \partial\Phi_z} \geq 0 \tag{2.20}$$

For more than one dimension, the Jacobian matrix must be evaluated. For the present cases of eqs. (2.17) and (2.18), the inequality in eq. (2.20) is not satisfied, and the H is said to be non-convex. This implies that the $F(x,z,t)$ depends on $\Phi(x,z,t)$ and H is non-smooth or singular [16,37]. As a result, a special class of numerical schemes for dealing with these more complex non-convex cases is used. Equation (2.15) must be redefined as follows [16,37]

$$\frac{\partial\Phi}{\partial t} + \hat{H} = (\varepsilon K) F(x,z,t) \left|\nabla\Phi(x,z)\right| \tag{2.21}$$

where $\hat{H}$ is the numerical Hamiltonian [16],

$$\hat{H} = H(\nabla\Phi^{\mathrm{c}}(x,z)) - \sum_{g=x,z} \gamma^{g}\left(\frac{\Phi_g^{+} - \Phi_g^{-}}{2}\right) \tag{2.22}$$

H is evaluated using Φ_g^{c}, i.e. central FDs, and γ^g are bounds of the partial derivative of H with respect to Φ_g [16],

$$\gamma^{g} = \max\left|H_{\Phi_g}(\nabla\Phi^{\pm}(x,z))\right| \tag{2.23}$$

where the maximum of H_{Φ_g} is evaluated using all the combinations of Φ_g^{+} and Φ_g^{-}, and the superscript 'g' differentiates between the spatial variables. These terms are second-order linear smoothing viscosity terms which act as second derivatives. If they are too large, the results will yield unrealistic smoothing of sharp corners; if they are too small, numerical instabilities will result [37]. All the quantities on the RHS of eq. (2.21) are evaluated using Φ_g^{c}, since these terms are parabolic contributions to the equation of motion and information propagates in both directions [16]. The temporal partial derivative in eq. (2.21) is evaluated using appropriate first order forward FD.

The numerical schemes for evaluating $\hat{H}$ differ from each other in the manner in which γ^g is found. For example, the Lax-Fredrichs scheme evaluates eq. (2.23) by searching for the maximum in the entire computational domain. This is very computationally expensive, and can also result in unnecessary smoothing which can lead to inaccurate results. To avoid this, the present work utilized the Local Local Lax-Fredrichs scheme, which evaluates eq. (2.23) by using Φ_g^{+} and Φ_g^{-} at a specific grid point [37]. To evaluate eq. (2.23), H_{Φ_x} and H_{Φ_z} for glass and PMMA can be obtained by taking the partial derivatives of eqs. (2.17) and (2.18) with respect to Φ_x and Φ_z,

$$(H_{\mathrm{b}})_{\Phi_x;\Phi_z} = H_{\mathrm{b}}\left[\frac{(k_{\mathrm{v}}+1)(x'';z)}{(x''\Phi_x + z\Phi_z)} - \frac{k_{\mathrm{v}}(\Phi_x;\Phi_z)}{\Phi_x^2 + \Phi_z^2}\right] \tag{2.24}$$

$$(H_{\mathrm{d}})_{\Phi_x;\Phi_z} = \frac{H_{\mathrm{d}}}{\sqrt{\Phi_x^2 + \Phi_z^2}}\left[n_1(A)_{\Phi_x;\Phi_z} + n_2 Hv(B)_{\Phi_x;\Phi_z} + \frac{(\Phi_x;\Phi_z)}{\sqrt{\Phi_x^2 + \Phi_z^2}}\right] \tag{2.25}$$

with

$$(A)_{\Phi_x;\Phi_z} = \frac{(x'';z)(\Phi_z;\Phi_x)^2 - (z;x'')\Phi_x\Phi_z}{(x''\Phi_x + z\Phi_z)\sqrt{\Phi_x^2 + \Phi_z^2}} \tag{2.26}$$

$$(B)_{\Phi_x;\Phi_z} = \frac{-\left\lfloor (x'';z)(\Phi_z;\Phi_x)^2 - (z;x'')\Phi_x\Phi_z \right\rfloor}{(\Phi_x^2 + \Phi_z^2)\sqrt{x''^2 + z^2}\left[1 + Hv(1 - \cos\theta)\right]} \tag{2.27}$$

where the notations (x''; z) and (Φ_x; Φ_z) in eq. (2.24) indicate that x'' and Φ_x are used to calculate $(H_\mathrm{b})_{\Phi_x}$ while z and Φ_z are used to calculate $(H_\mathrm{b})_{\Phi_z}$, etc.

2.3.4.3 Grid formulation, boundary conditions and CFL condition

For all cases, the vertical grid limits were set at $z_\mathrm{min} = h\sin\alpha$ and $z_\mathrm{max} = h\sin\alpha + z_\mathrm{surf}$, where z_surf is the maximum expected feature depth. For unmasked cases, using the geometry of Figure 2.1 (top, right), the horizontal grid limits were defined by $x_\mathrm{min/max} = h\tan\psi + h\sin\alpha/\tan(\alpha \pm \psi)$. For a spatial particle distribution corresponding to eq. (2.8), the jet spread angle at which 99.9% of particles arrive to the surface was assumed to be $\psi = \tan^{-1}(\sqrt{-\ln(0.001)}/\beta)$ [34]. For masked cased at $\alpha = 90°$, $x_\mathrm{min} = 0$ and $x_\mathrm{max} = W_\mathrm{m}$. These limits were used to obtain appropriate spatial grid steps; i.e. $\Delta x = (x_\mathrm{max} - x_\mathrm{min})/(i_\mathrm{max} - 1)$, and $\Delta z = (z_\mathrm{max} - z_\mathrm{min})/(k_\mathrm{max} - 1)$, where $i_\mathrm{max} \cdot k_\mathrm{max}$ is the grid size. The global spatial coordinates at the grid nodes were thus obtained as $x = (i - 1)\Delta x + x_\mathrm{min}$ and $z = (k - 1)\Delta z + z_\mathrm{min}$. The boundary conditions were assumed such that the partial derivatives of $\Phi(x,z,t)$ directed towards outside the computational domain were zero. Finally, the time step Δt was limited by the Courant-Friedreichs-Lewy (CFL) condition, i.e. that the numerical wave speed should be greater than or equal to the physical wave speed to ensure stability [37],

$$\Delta t \cdot \max\left[\sum_{g=x,z}\left(\frac{\left|H_{\Phi_g}(\nabla\Phi^{\pm})\right|}{\Delta g} + \frac{2\varepsilon}{\Delta g^2}\right)\right] < 1 \tag{2.28}$$

where the superscript ‘g’ differentiates between spatial variables. The maximum is calculated by searching the entire computational domain.

2.3.4.4 Surface initialization, re-initialization and interpolation

The initial surface was represented by a horizontal line, and the level set function was initialized at $t = 0$ using the signed distance function, SDF, of a point $\vec{x}$ from the surface [37],

$$\Phi(\vec{x}, t) = \mathrm{SDF}(\vec{x}) = \pm \min(|\vec{x} - \vec{a}|), \vec{a} \in (\Phi = 0) \tag{2.29}$$

where $\vec{x}$ is the position of some grid node and $\vec{a}$ is the nearest point on the surface, and the function is positive if $\vec{x}$ is in front of the surface propagation direction, and negative if behind the surface propagation direction. The zero level set evolves naturally unless it encounters discontinuities, which can occur when the surface is propagated with curvature smoothing [16,37], since the SDF is not differentiable. In this case, the SDF must be re-initialized, i.e. re-calculated, every fixed number of time steps to ensure the level sets are evenly spaced around the front, using eq. (2.29) for $t > 0$ [16]. In order to do this, and to visualize the evolving surface profile for $t > 0$, the surface, i.e. the zero level set, must be interpolated since it is usually located between the grid nodes, as shown in Figure 2.4. The entry and exit points of the zero level set were interpolated linearly in order to maintain monotonicity and hence stability.

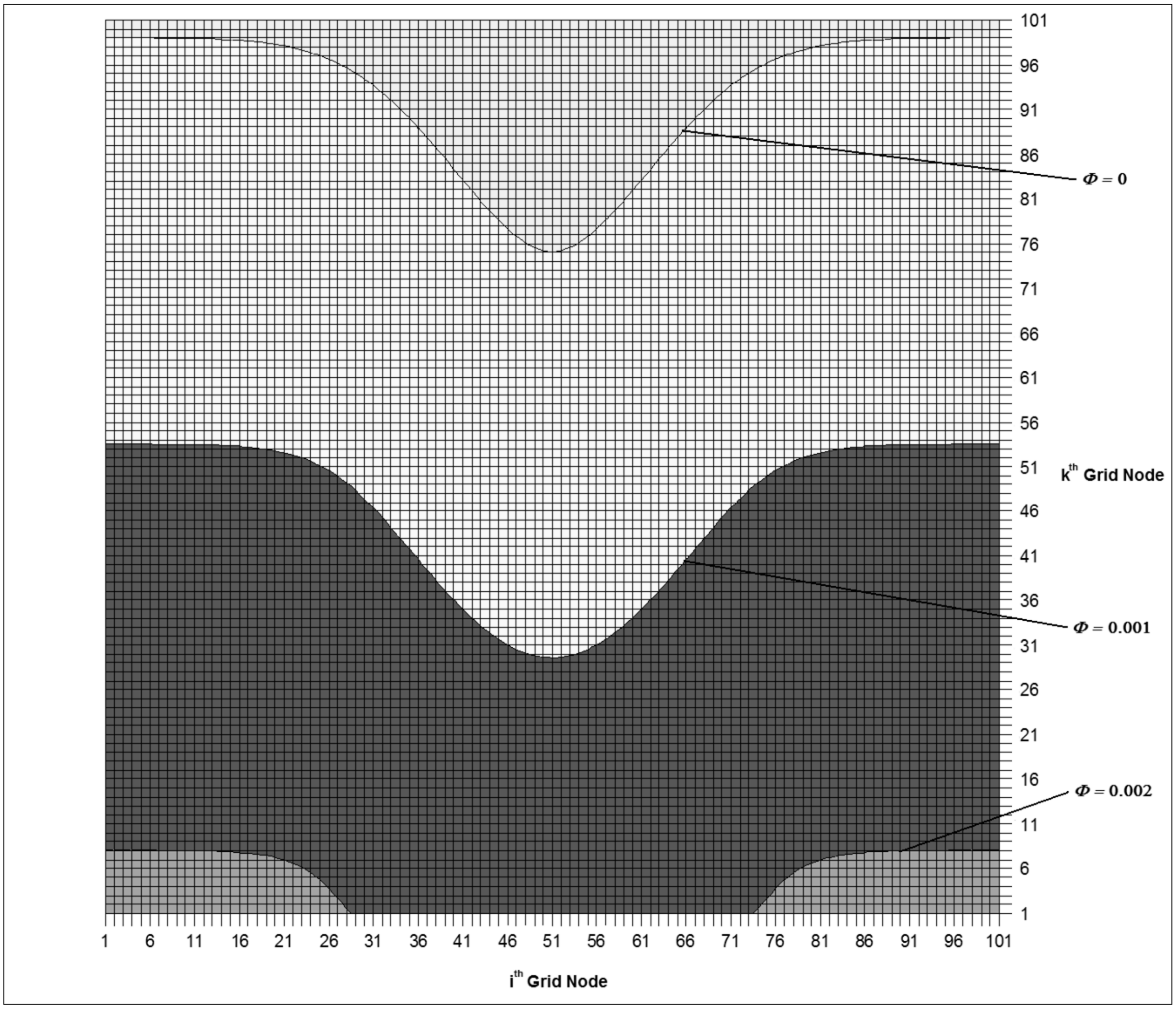


Figure 2.4. Visual representation of Φ, including Φ = 0, i.e. the surface, and the computational grid for the case in Figure 2.5 after 6 passes (see below).

2.4 Results and discussion

2.4.1 Comparison with experiments of Section 2.2

2.4.1.1 Model execution and inputs

The LSM model presented in Section 2.3 was implemented in MATLAB 7.7 (The MathWorks, Inc., Natick, MA, USA). MATLAB was chosen as the programming platform as it has many built-in standard functions. Although LSM libraries have been (scarcely) made available commercially, they are not easily adaptable to difficult LSM problems. As a result, LSM was applied to the current problem from the ground up, i.e. from ‘scratch’.

The resulting predicted surface evolution profiles are compared to the measured ones in Figures 2.5-2.12. On a 2.6 GHz Quad-core Intel CPU with 4 MB of RAM, the execution times, ETs, for most cases were between 16 min and 2.5 hrs. For the majority of cases, the average particle mass flow rate was used since repeatability over the course of machining was good. However, for cases where the mass flow rate fluctuation over the course of machining was significant (Figures 2.7-2.10), it was modelled as a linearly decreasing function of time (see Appendix A, Tables A-3-A-6). In addition to particle compaction in the particle reservoir of the microblaster mentioned in Section 2.2, three other factors that can affect mass flow rate repeatability are particle size stratification, relative humidity of storage air and the effect of powder level in the reservoir [43]. It is difficult to control for all these factors during the course of experimental runs. Thus, for the cases in Figures 2.8-2.10, the mass flow rate decrease with time was most likely caused by the decrease in powder level in the reservoir since a longer machining time was required for those cases by using a slower scan speed of 0.5 mm s^{-1} to obtain a desired depth up to 30 passes for the slowly eroding PMMA, when compared to glass. For the case in Figure 2.7, the decrease in the flow rate could have been due to a combination of all these factors.

For the present nozzle, particle, and jet conditions, β=15 in eqs. (2.8) and (2.9) and V_o= 162 m s^{-1} in eqs. (2.6) and (2.7) were assumed, based on values measured in [14]. For glass targets, k_v = 1.43 in eqs. (2.17) and (2.24) [14], while for PMMA targets, k_v = 2.0, n_1 = 1.27, n_2 = 15.5 and Hv = 0.25 GPa were assumed in eqs. (2.18), (2.25) and (2.27), based on the measurements in [2]. For masked cases, values of μ_l = -11.6 and σ_l = 0.5 were assumed in eq. (2.11) from the measurements made in [19], for the 25 μm alumina particles utilized in the present work. The erosion constants C = 8.0 x 10^{-6} (m $s^{-1})^{-k_v}$ (glass) and C = 5.7 x 10^{-8} (m $s^{-1})^{-k_v}$ (PMMA) used in eqs. (2.17) and (2.18) were obtained from [36].

For most cases, it was not necessary to smooth the results based on curvature, and $\varepsilon = 0$ was assumed. This is especially true for unmasked and masked PMMA features, where the profile bottoms remain rounded and flat, respectively. However, for glass targets, the modelled profiles in Figures 2.5 and 2.11 resulted in converging sidewalls at the profile bottom that tended to create a pointed profile, i.e. a high K. In these cases, beyond AR of roughly 0.2 for unmasked and 0.5 for masked profiles, the smoothing parameter ε [16,17,37] described in Section 2.3.4.1 was estimated based on the recommendations of [17], which gives a range of ε of (0.2-1.0) x 10^{-4} and (0.09-2.25) x 10^{-5} for unmasked and masked cases, respectively. In these cases, the frequency of re-initialization, FR, (Section 2.3.4.4) was obtained through a numerical convergence study by ensuring that numerical stability was maintained, but at the same time the FR minimized, since this step was computationally expensive [16,37]. Similarly, for all cases, $i_{\max} \cdot k_{\max} = 101 \cdot 101$ and the spatial grid steps Δx and Δz, calculated as described in Section 2.3.4.3, were in the range of 4–220 μm, which ensured the convergence and accuracy of the numerical solution, i.e. the grid steps for each case were decreased until the profile plots between successive simulations varied by no more than 1%. The mean representative time step calculated with eq. (2.28) for all cases was in the range of (0.53-5.3) x 10^{-2} s.

For the majority of cases, V and ϕ were calculated using eqs. (2.6) and (2.8) for the micro-machining of holes, and eqs. (2.7) and (2.9) for the micro-machining of channels. However, for the unmasked PMMA channels in Figures 2.8-2.10, eqs. (2.7) and (2.9) had to be slightly modified by assuming a stationary target approach, i.e. $y = y' = (r_s - v_t t) = 0$. This modification was necessary because, for ductile erosive systems, the 2D approximation for the scanning target, i.e. $y' = (r_s - v_t t)$, used in eq. (2.18) introduces a component of the erosive efficacy, $E_{\text{ef}} = CV^{k_v}\phi$, in the y scanning direction that incorrectly cause the surface to grow in the x - z plane. This y component, originating from eqs. (2.3) and (2.5), cannot be eliminated from the 2D channel formulation, and ultimately causes the channel cross-section in the x - z plane to erode and widen, due to the surface tangential component of the erosion law in eq. (2.5). In reality, this tangential component should mostly represent damage done by cutting and ploughing mechanisms in the y direction, and thus should primarily cause the surface to erode in along the channel in the y direction. The net result is the y direction erosive efficacy causes the channels to grow too rapidly wide. This 'widening' effect is minimal in glass since its erosion law depends only on the components of erosive efficacy related to energy transfers normal to the surface (eq. (2.17)), and hence eqs. (2.7) and (2.9) are valid.

The use of a stationary target assumption for channels is acceptable for ductile targets because [14] demonstrated that, to a first approximation, the dose, i.e. mass, of particles that a given channel cross-section receives in the direction normal to the surface during one scanning pass over a time of $2r_s/v_t$

is roughly equivalent to the dose that the cross-section at the centre of a hole receives for a stationary target during the time it takes to reach the same depth. However, since the scanning target erosive efficacy has a smooth bell shape, whereas the stationary target erosive efficacy has a bell shape with a cusp at *x'* = 0, the use of eqs. (2.7) and (2.9) delays the onset of high curvature, K , i.e. a pointed profile, and hence avoids the use of curvature smoothing in many cases, which is computationally expensive. It is therefore efficient to use it for glass targets that develop such sharp profiles. For example, Figure 2.13 compares the LSM model which used the scanning target erosive efficacy, with the model of [14] which used the stationary target erosive efficacy, both with $\varepsilon = 0$. The use of the model of [14] resulted in pointed profiles, while the use of the LSM model resulted in smooth profiles. However, since the pointed profile only occurs in glass, this difference was not significant for PMMA. For masked PMMA channels, the modification was not required since the small mask opening limited the range of incident angles of attack over a narrow range near to perpendicular.

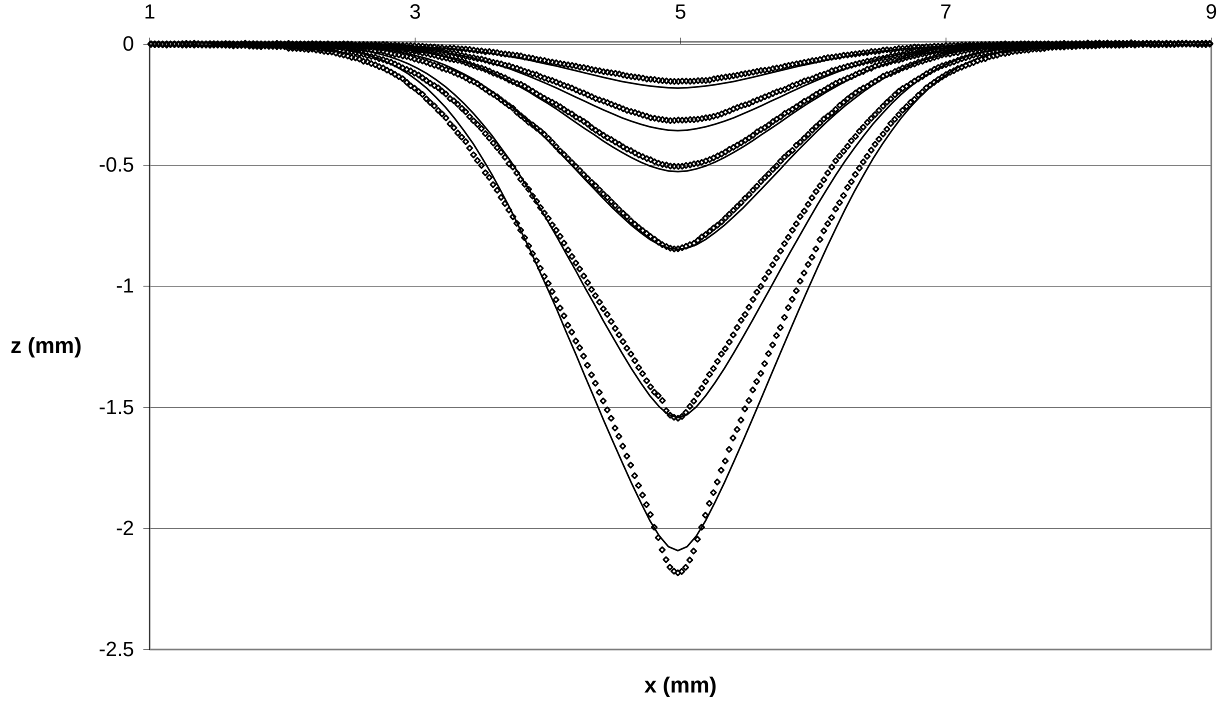


Figure 2.5. LSM predicted (—) and measured (◊) surface evolution of unmasked channels machined in glass at α = 90° after 2, 4, 6, 10, 20 and 30 passes of the nozzle. $\dot{M}$ = 3.30 g min^{-1}, ε= 5.0 x 10^{-5}, FR = 1/20 time steps, ET = 25 min. All other model inputs are specified in Sections 2.4.1.1 and 2.2.

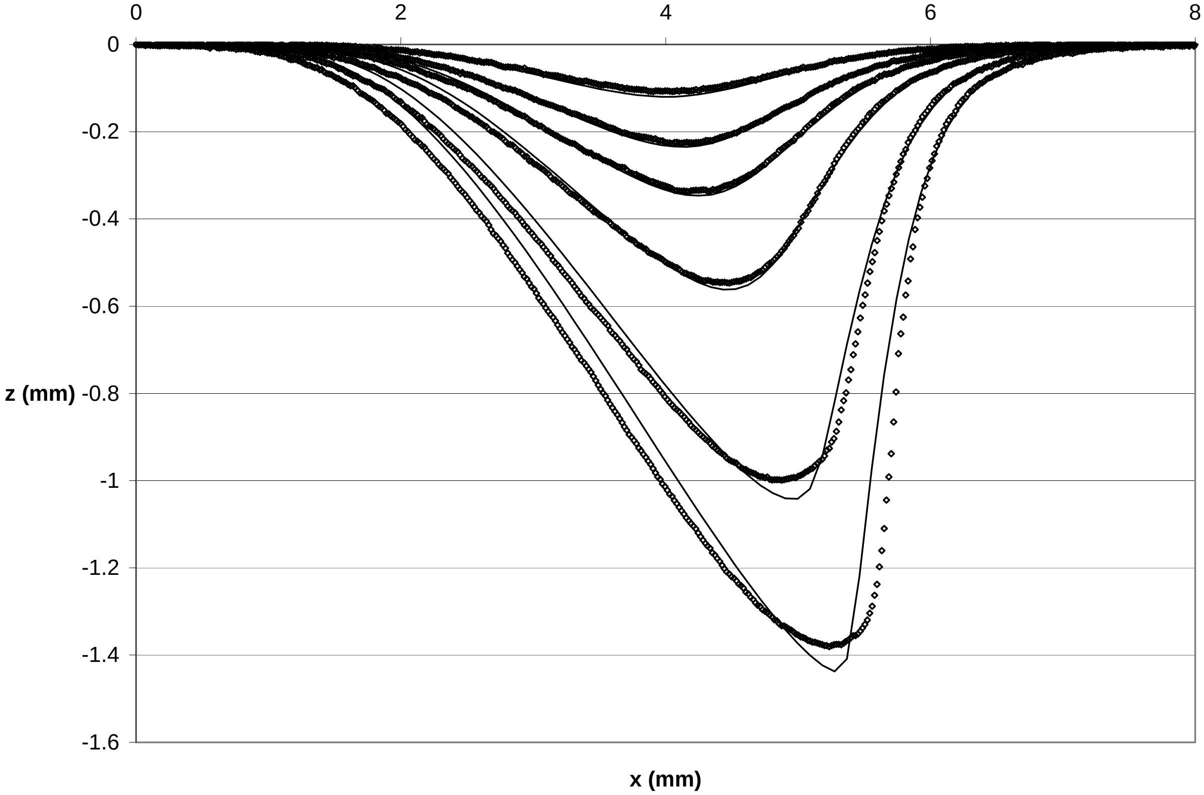


Figure 2.6. LSM predicted (—) and measured (◊) surface evolution of unmasked channels machined in glass at $\alpha = 60°$ after 2, 4, 6, 10, 20 and 30 passes of the nozzle. $\dot{M} = 3.30$ g min^{-1}, ET = 16 min. All other model inputs are specified in Sections 2.4.1.1 and 2.2.

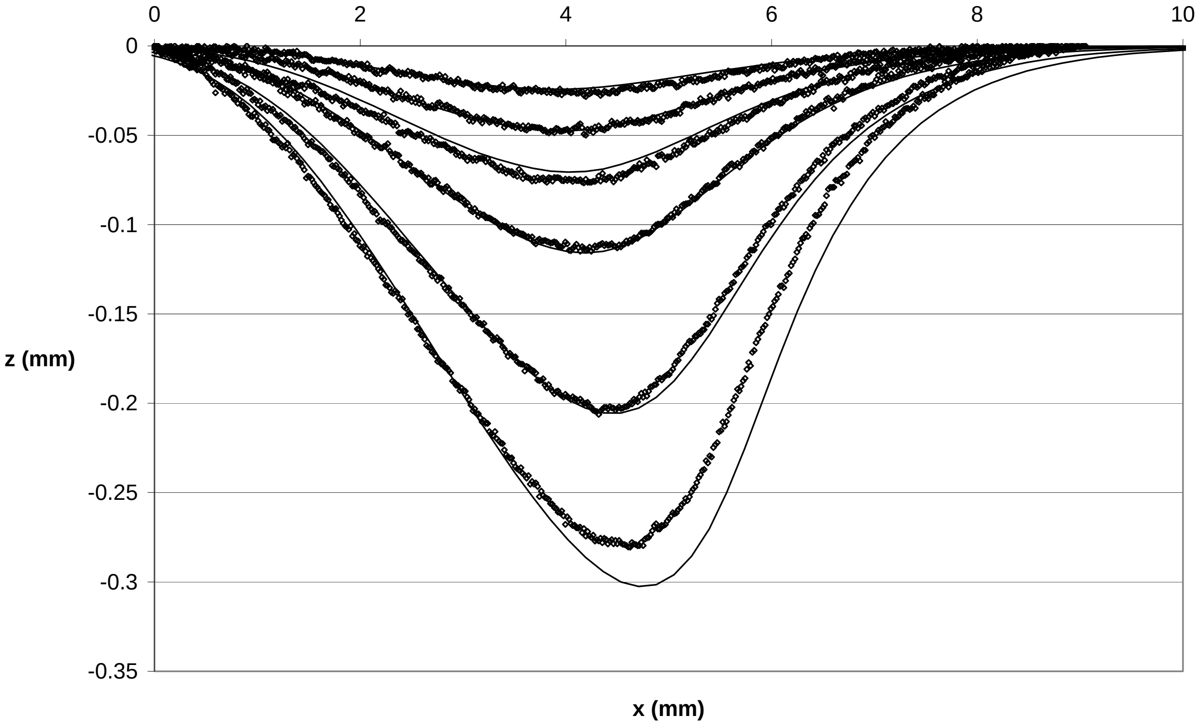


Figure 2.7. LSM predicted (—) and measured (◊) surface evolution of unmasked channels machined in glass at $\alpha = 30°$ after 2, 4, 6, 10, 20 and 30 passes of the nozzle. $\dot{M}$ = (2.43 -1.19 x $10^{-3}t$ (s)) g min^{-1}, ET = 16 min. All other model inputs are specified in Sections 2.4.1.1 and 2.2.

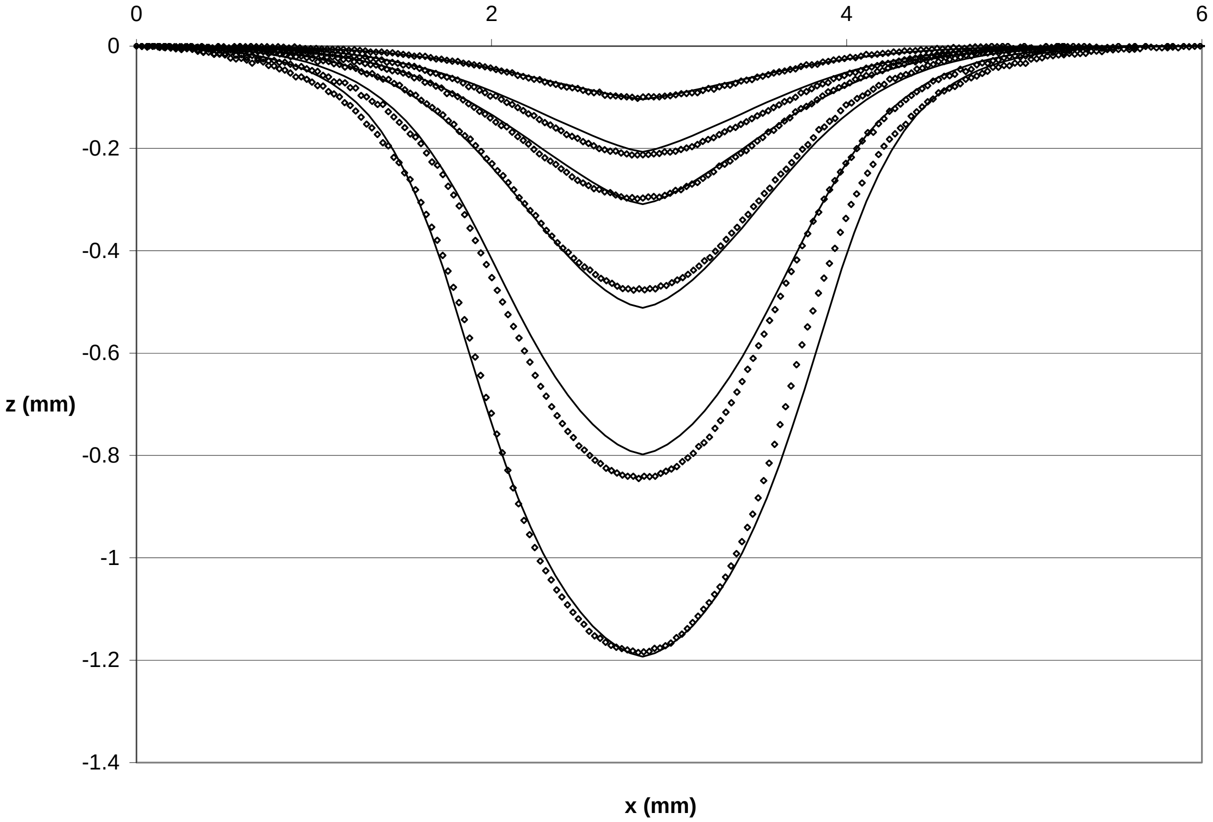


Figure 2.8. LSM predicted (—) and measured (◊) surface evolution of unmasked channels machined in PMMA at $\alpha = 90°$ after 2, 4, 6, 10, 20 and 30 passes of the nozzle. $\dot{M}$ = (3.97 -2.05 x $10^{-3}t$ (s)) g min^{-1}, ET = 141 min. All other model inputs are specified in Sections 2.4.1.1 and 2.2.

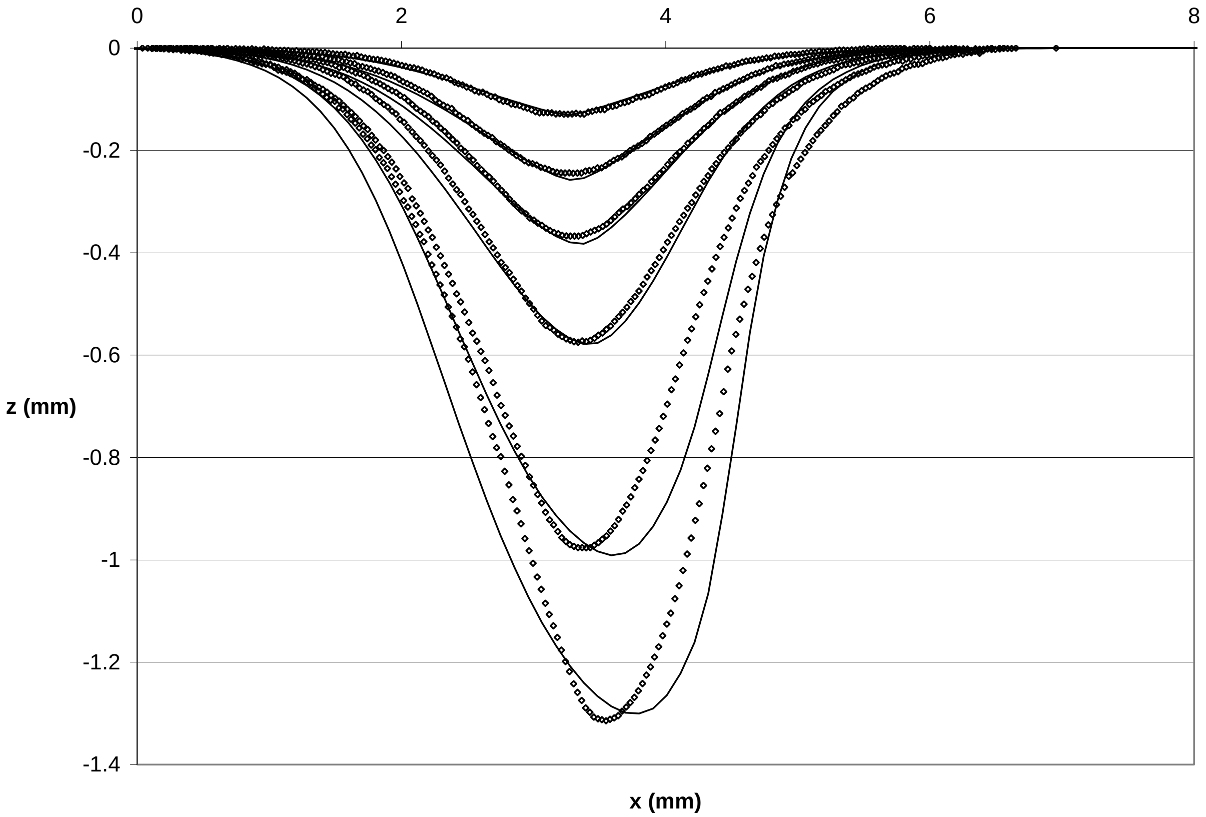


Figure 2.9. LSM predicted (—) and measured (◊) surface evolution of unmasked channels machined in PMMA at $\alpha = 60°$ after 2, 4, 6, 10, 20 and 30 passes of the nozzle. $\dot{M} = (3.88 - 3.47 \times 10^{-3} t\ (\text{s}))\ \text{g min}^{-1}$, ET = 138 min. All other model inputs are specified in Sections 2.4.1.1 and 2.2.

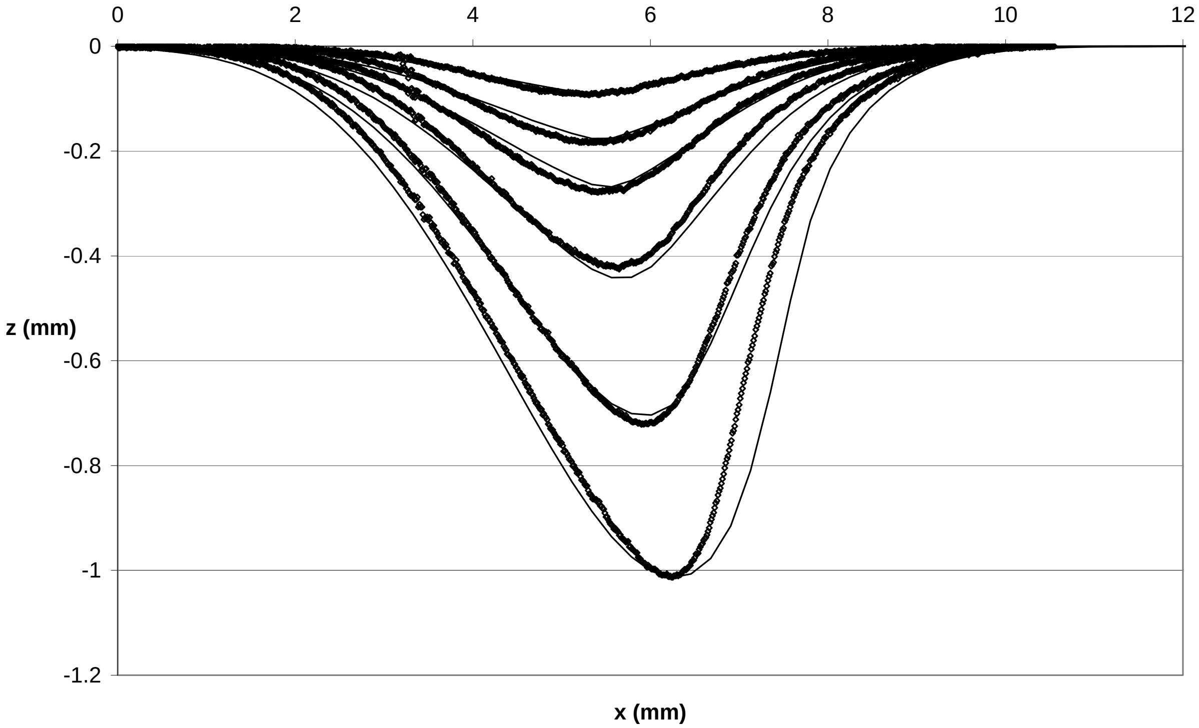


Figure 2.10. LSM predicted (—) and measured (◊) surface evolution of unmasked channels machined in PMMA at $\alpha = 30°$ after 2, 4, 6, 10, 20 and 30 passes of the nozzle. $\dot{M} = (2.66 - 1.19 \times 10^{-3} t(\mathrm{s}))$ g min^{-1}, ET = 135 min. All other model inputs are specified in Sections 2.4.1.1 and 2.2.

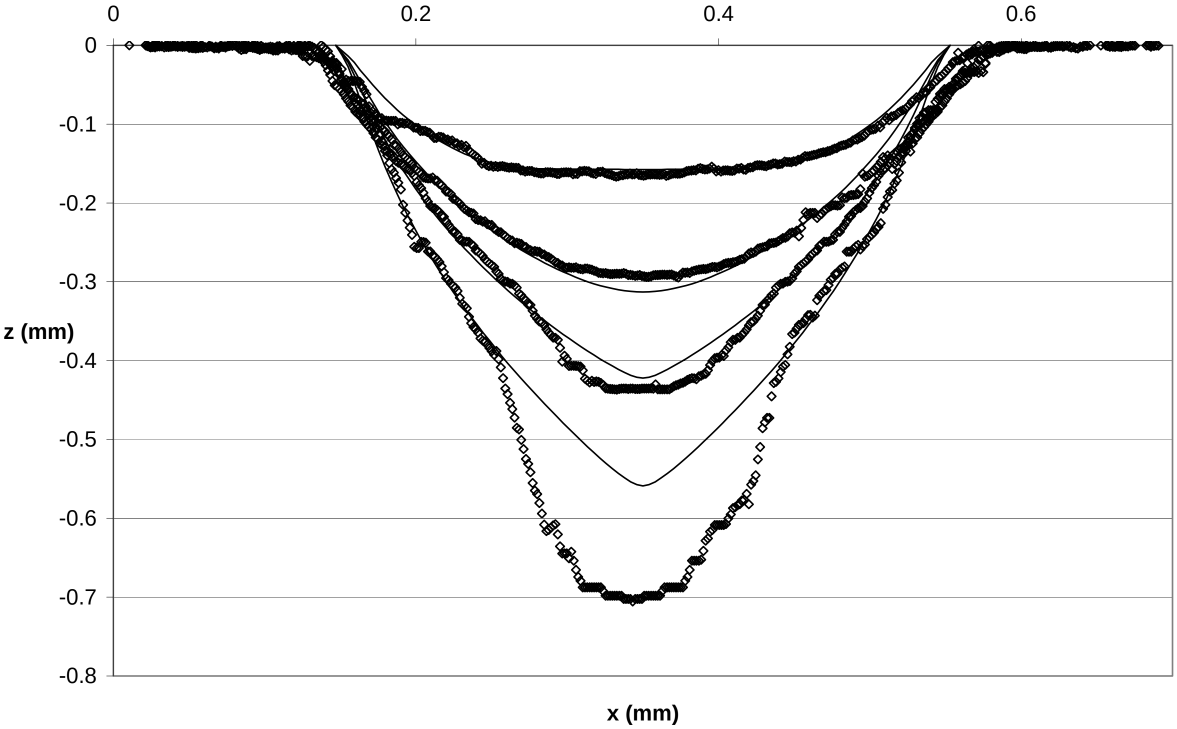


Figure 2.11. LSM predicted (—) and measured (◊) surface evolution of masked channels machined in glass at $\alpha = 90°$ after 2, 4, 6 and 10 passes of the nozzle. $\dot{M} = 2.63$ g min^{-1}, $\varepsilon = 2.0 \times 10^{-6}$, FR = 1/20 time steps, ET = 21 min. All other model inputs are specified in Sections 2.4.1.1 and 2.2.

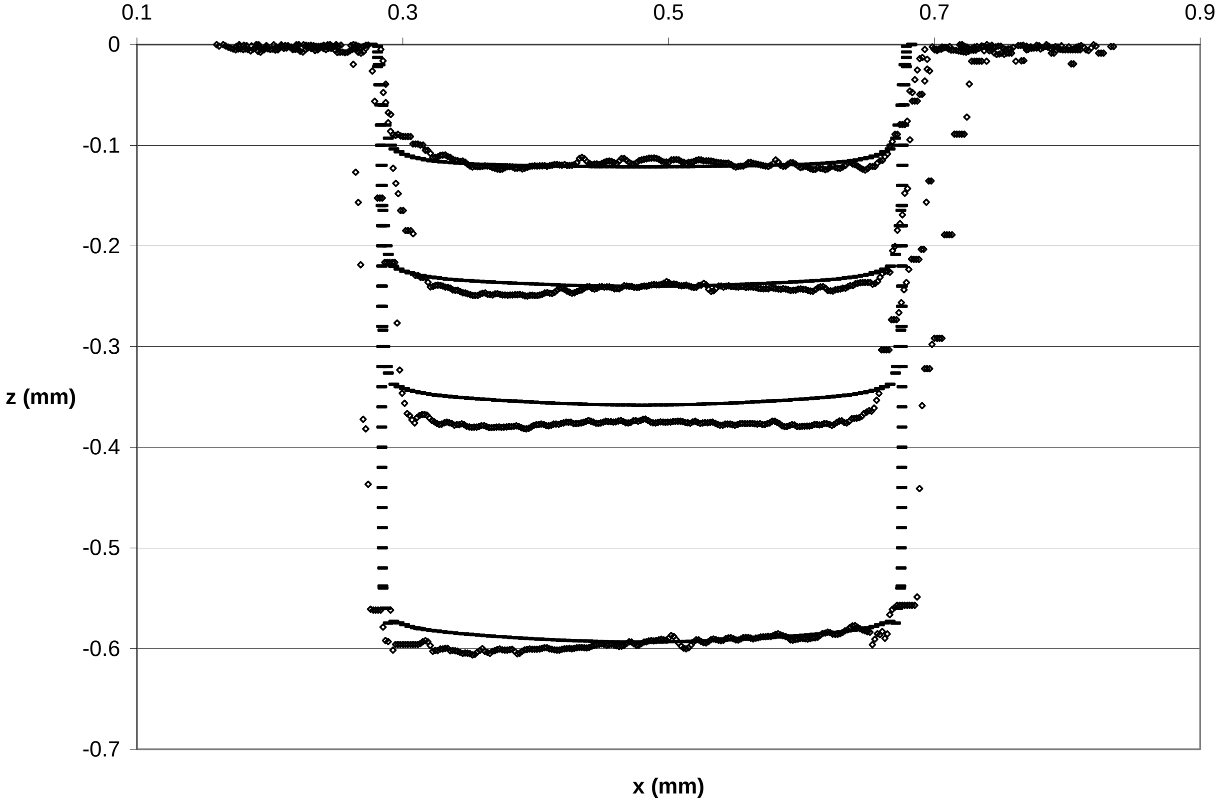


Figure 2.12. LSM predicted (**—**) and measured (◊) surface evolution of masked channels machined in PMMA at $\alpha = 90°$ after 2, 4, 6 and 10 passes of the nozzle. $\dot{M} = 5.38$ g min^{-1}, ET = 245 min. All other model inputs are specified in Sections 2.4.1.1 and 2.2.

2.4.1.2 Fit of LSM model to experiments

Comparisons of the LSM model with experiments of Section 2.2 (Figures 2.5-2.12) showed excellent agreement for the majority of cases, showing the promise of the LSM for modelling surface evolution in AJM. For all cases, relatively minor discrepancies between the model and experiments could have been caused by localized mass flow rate fluctuations, as seen in Figure 2.5 (pass 30), Figure 2.6 (passes 20 and 30), Figure 2.7 (pass 30), Figure 2.8 (passes 10 and 20) and Figure 2.12 (pass 6), and by nozzle misalignments, as seen in Figure 2.9 (passes 20 and 30) and Figure 2.10 (pass 30). However, for the deep profiles in Figure 2.11, the modelled predicted profiles began to significantly deviate from experimental ones beyond AR > 1. The poor fit for deep masked profiles can be understood in the context of two effects that were not considered by the present LSM model: second strikes of particles, and spatial hindering [13,21-24]. These were discussed in Chapter 1 and will be considered in more detail in Chapter 5. In addition, in Figure 2.12, the experimental profiles were wider for deeper profiles than the modelled profiles due to mask wear [24,29,32], which was not considered in the present model. This effect will be considered in detail in Chapter 4.

2.4.2 Comparisons with previously published models and experiments

2.4.2.1 Model execution and inputs

To further validate the predictions of the present LSM methodology, comparisons were also made to previously published models and experiments for selected cases at α = 90°. The ETs for most cases were between 3 min and 1 hr. (Figures 2.13-2.16). In all cases, the $\dot{M}$, v_t, W_m, and H_m values that were obtained based on data presented in [2], [14], [17], [19] and [36], are indicated in the figure captions. For all cases, $\dot{M}$ had to be slightly adjusted to match the initial pass profile due to a lack of published experimental details, and $\dot{M}$ variability. In all cases, except Figure 2.15, where a rectangular 0.3 mm x 3.8 mm nozzle was used, β, V_o, k_v, n_1, n_2, Hv, μ_l, and σ_l were obtained from the same references as specified in Section 2.4.1.1, since all the previous models assumed the same nozzle, particles and jet conditions as those used in Section 2.2. For the predictions of Figure 2.15, a uniform flux along the mask width and a scanning target were assumed; i.e. $x'' = 0$ in eq. (2.8) since the target was oscillated during the machining of the PMMA hole [17]. Also, for Figure 2.15, V_o = 148 m s^{-1} and β = 15 were used for the rectangular nozzle, obtained from unpublished measurements [44]. All other variables were the same as for the circular nozzle. C and ε (= 0) (Figures 2.13-2.16) were obtained and used in the same manner as described in Section 2.4.1.1. For all cases, the same grid size was used as for Figures 2.5-2.12, and Δx and Δz were in the range of 7.6–70 μm, obtained in the same way as described in Section 2.4.1.1. The mean representative time step calculated with eq. (2.28) for all cases was in the range of (0.13-5.6) x 10^{-2} s. Finally, for the unmasked PMMA channels in Figure 2.16, the stationary target approach was used for the same reasons as described in Section 2.4.1.1.

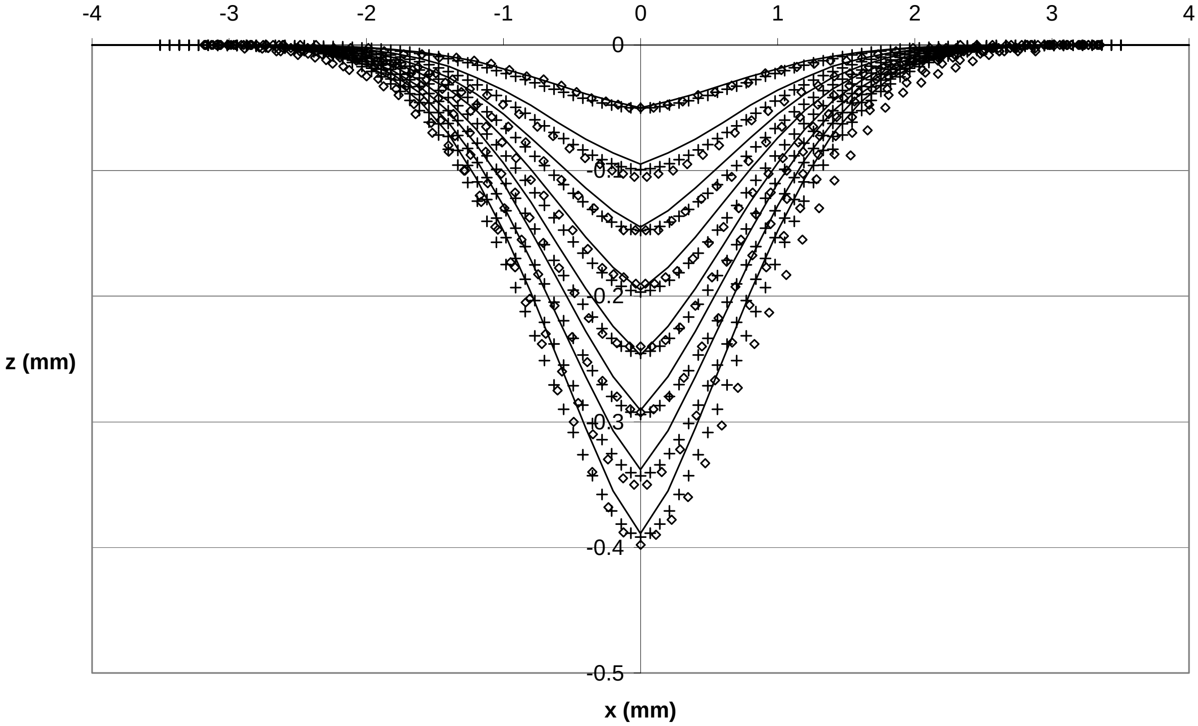


Figure 2.13. LSM predicted (+), analytical model predicted (—; ML) and measured (◊) surface evolution for unmasked channels machined in glass at *α* = 90° after 1, 2, 3, 4, 5, 6, 7 and 8 passes of the nozzle. Measured profile and analytical model data from [14]. $\dot{M}$ = 1.91 g min^{-1}, v_t = 1.0 mm s^{-1}, ET = 3 min. All other model inputs are specified in Section 2.4.2.1.

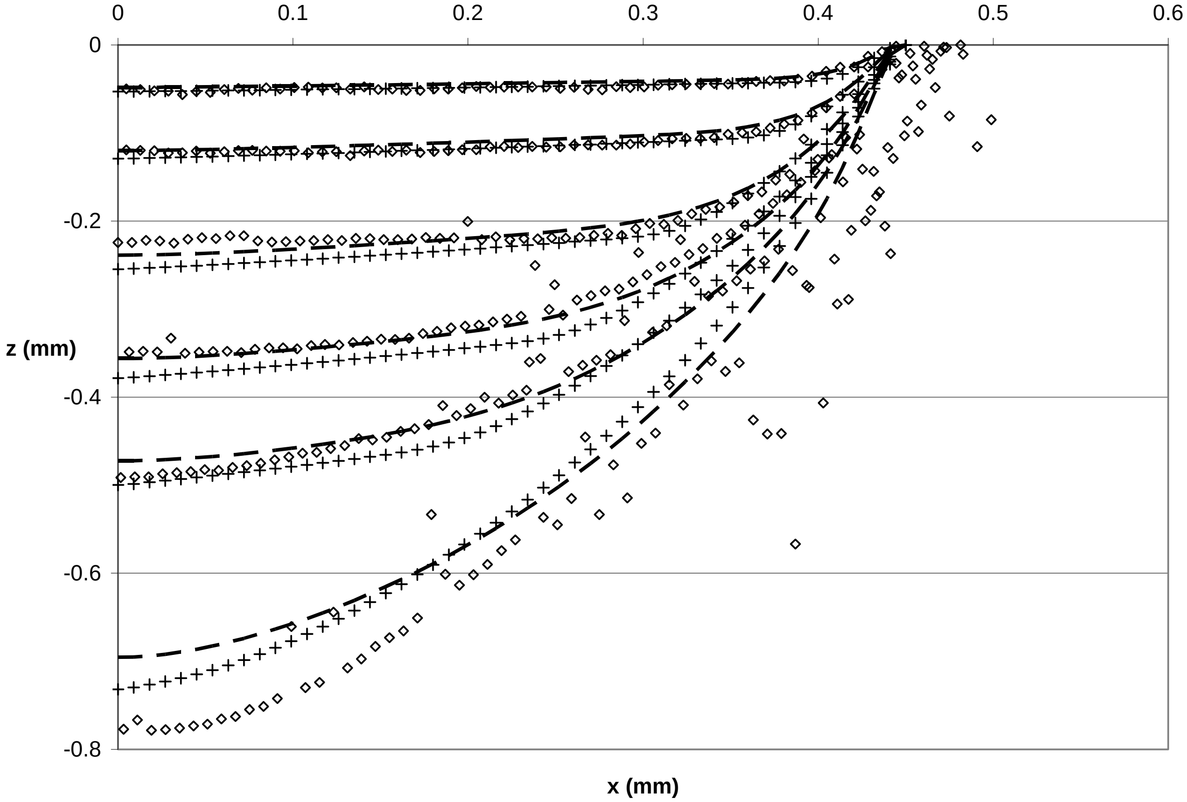


Figure 2.14. LSM predicted (+), analytical model predicted (— —; ML) and measured (◊) surface evolution for masked holes machined in glass at $\alpha = 90°$ after 2, 5, 10, 15, 20, 30 s. Measured profile data from [17] and analytical model data from [19]. $\dot{M} = 2.43$ g min^{-1}, $W_m = 900$ μm, $H_m = 1000$ μm, ET = 62 min. All other model inputs are specified in Section 2.4.2.1. Only half of the symmetric hole is shown.

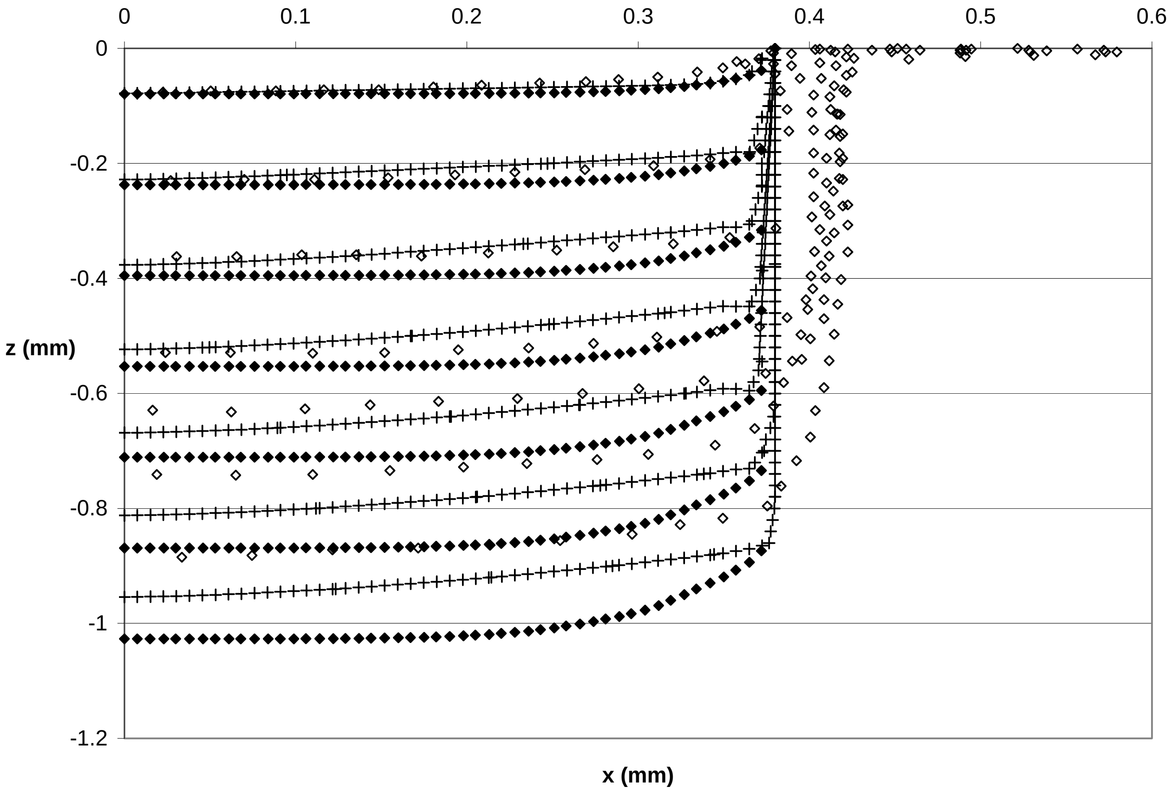


Figure 2.15. LSM predicted (+), analytical model predicted (♦; ML) and measured (◊) surface evolution for masked holes machined in PMMA at α = 90° after 1, 3, 5, 7, 9, 11 and 13 passes of the nozzle. Measured profile and analytical model data from [17]. $\dot{M}$ = 6.90 g min^{-1}, v_t = 0.25 mm s^{-1}, W_m = 760 μm, H_m = 1000 μm, ET = 737 min. All other model inputs are specified in Section 2.4.2.1. Only half of the symmetric hole is shown.

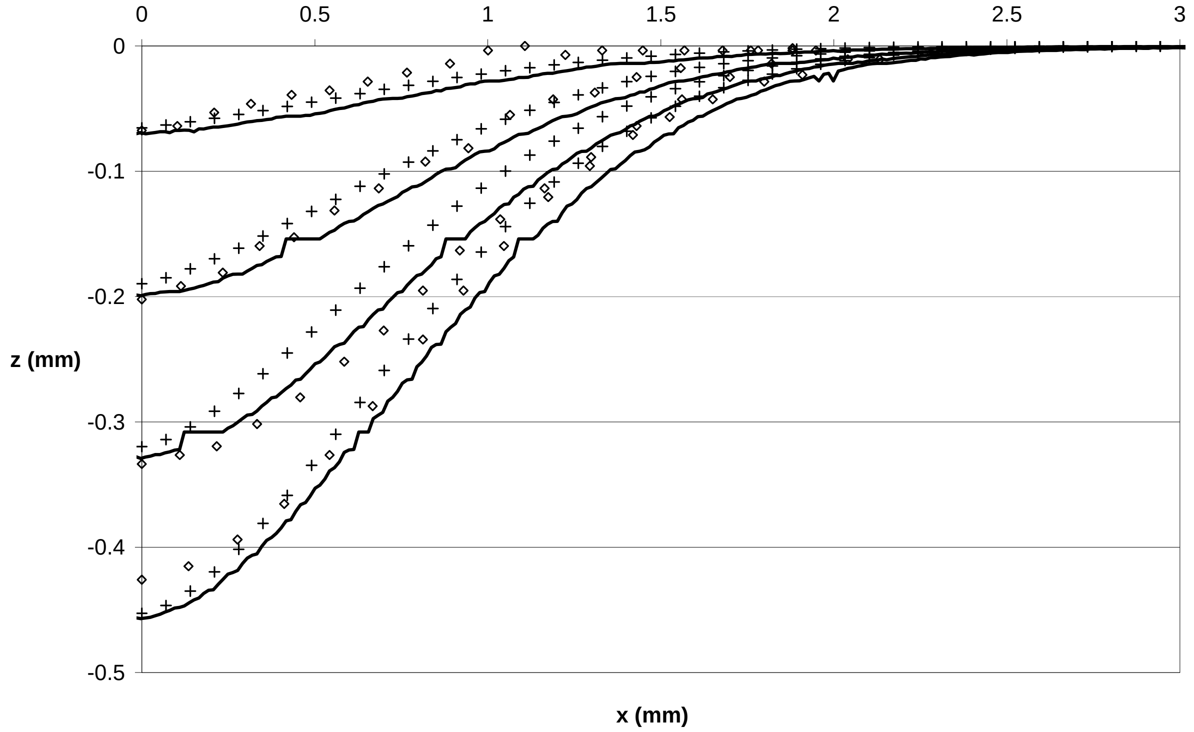


Figure 2.16. LSM predicted (+), computer simulation predicted (▬; CS) and measured (◊) surface evolution for unmasked channels machined in PMMA at α = 90° after 1, 3, 5 and 7 passes of the nozzle. Measured profile from [2] and computer simulation data from [36]. $\dot{M}$ = 2.68 g min^{-1}, v_t = 0.25 mm s^{-1}, ET = 55 min. All other model inputs are specified in Section 2.4.2.1. Only half of the symmetric hole is shown.

2.4.2.2 Comparisons with previous analytical models

The LSM model was compared against previous experiments and analytical models [14,17,19] in Figures 2.13-2.15. The analytical models of [2], [14], [17], and [19] used Mathcad software (Mathsoft, Inc., Cambridge, MA, USA), which applies a pre-coded method of lines (ML) [45] to solve the equations of motion. The ML method, is similar to LSM in that it also uses FDs to approximate spatial and temporal variables. Both methods utilize upwind FD schemes to ensure that FD approximations follow the exact-solution PDE theory so that numerical solutions can converge correctly; i.e. either forward, backward or central FD are used, depending on the situation [16,37].

The underlying difference between the ML and LSM methods is that LSM defines the surface implicitly [16,37], whereas ML defines the surface explicitly, i.e. the surface is directly defined by the spatial variable. LSM can thus be used to solve more complex cases and can be more easily extended to higher dimensions [16] than ML. For the present study, the general difference between the LSM model and the ML solutions to previous analytical models stem from this implicit surface formulation, described in Section 2.3.4, and the difference in modelling the physical aspects of the erosion process, described in Sections 2.3.1-2.3.3. These are summarized as follows:

i. The present LSM model formulation allows the modelling of features which are multi-valued, i.e. there are two or more depth values at a given profile location along the width, or with approximately vertical sidewalls, something which is not possible with previous analytical models.

ii. The present model incorporated a transformed coordinate system (Section 2.3.1) allowing the modelling of obliquely shaped features in the cross-sectional plane (Figures 2.6, 2.7, 2.9 and 2.10), something not previously considered in the analytical models.

iii. The previous analytical models assumed that particles were all incident at the nominal angle of incidence, and thus neglected the variation in incident angles of attack brought about by the nozzle divergence. The present model took this spread in angles of attack into account in eqs. (2.17)-(2.19) and (2.24)-(2.27).

iv. In previous models, the incident particle flux and velocity at the surface were assigned based on the coordinates of the unmachined flat surface. In the present model, eqs. (2.6)-(2.9) utilized a depth spatial coordinate, i.e. z or z', that accounted for the change in particle velocities and fluxes from those at the flat surface (Figure 2.2).

v. For glass masked and unmasked channels, masked PMMA channels and holes, the 2D approximation of a scanning target described in Section 2.3.2 was used, whereas all previous analytical models assumed a stationary target approach for modelling such features.

vi. For masked features, the analytical model of [19] along with actual erosive efficacy, defined by $E_{\mathrm{ef}} = CV^{k_{\mathrm{v}}}\phi$, were used to model the normal incidence cases. The previous analytical models of [2], [14] and [17] (Figure 2.15) utilized an empirical exponential function for the net erosive efficacy from the experimentally obtained first pass profile. However, this did not physically account for the effect of particle size near the mask edge.

Refinement ii is relevant to features machined with oblique nozzles, something that has not been previously attempted with analytical models. Refinements iii and iv, when implemented in the present LSM scheme, were found to make very little improvement over analytical modelling of the surface evolution of the channels and holes considered in [2], [14], [17] and [19]. These refinements are likely more important for future modelling efforts of the machining of: 1) very deep features, where the implementation of refinement iv would account for the more drastic change in particle velocities and fluxes with increasing depth; and 2) very wide features, where the implementation of refinement iii would account for the more drastic variation in θ across the width of feature with increasing $|x'|$. This variation in θ is even more significant for wide features machined at oblique incidence (refinement ii). For instance, for an initially flat target attacked by the jet at oblique incidence, the θ to the left and right of the jet centreline becomes smaller and larger, respectively with increasing jet divergence. In both cases 1) and 2), there would be a more profound effect on the local speed of surface evolution, especially near the bottom or periphery of the very deep or wide features, respectively. It should also be emphasized that refinements iii and iv are necessary when second strike [13,21-24] and mask wear [24,29,32] are to be considered, i.e. when exact particle trajectories need to be specified. Refinements i, v and vi, however, were found to significantly improve the predicted surface evolution, and are therefore discussed in more detail. Although refinements ii-v have not been previously implemented, in theory they could be incorporated into the analytical modelling framework presented in [2], [14], [17] and [19].

Figure 2.13 shows that use of the scanning nozzle technique (refinement v) resulted in an improvement in the prediction of the shape of the experimental unmasked glass channel profiles over the analytical model of [14], mainly due the smoothing that the LSM model introduces, as explained in Section 2.4.1.1.

In Figure 2.14, the LSM model was comparable with that of [19] in predicting the experimental masked glass hole profile, since both models accounted for particle size near the mask edge (refinement vi), where the minor variations in the model fits could be due to the way the initial pass fit was obtained and due to the implementation of refinements iii and iv, since refinements i, ii and v play no role in this case. The LSM model matched the experimental profiles quite well, providing an improvement in the under-prediction of the profile depth after 30 s. The under-prediction at 30 s is likely due to a localized mass flow rate fluctiation during the experiment.

Use of refinements i, v and vi resulted in a large improvement in the prediction of the experimental masked PMMA hole profiles shown in Figure 2.15, over the existing analytical model of [17], in terms of both depth and shape. The most likely reason for the improvement is refinement i, which because of the nature of LSM, allows for the modelling of feature sidewalls that are multi-valued or vertical, such as in Figure 2.15 (or Figure 2.12). This is not possible with the ML solution of the analytical model. The over-predicted depths beyond 7 passes could be the result of particle embedding [2,8,17,31], which can be significant for hole profiles and could decrease the effective erosion rate.

2.4.2.3 Comparisons with previous computer simulation

In Figure 2.16, the LSM model was also compared against the computer simulation (CS) of [36], described in Section 1.2.2. Figure 2.16 shows excellent agreement with experiments and the CS of [36] for unmasked PMMA channels, and in most cases, a slight improvement over the CS (see passes 1, 3 and 7).

The simulation of [36] is very computationally expensive, more so than the LSM model proposed here. For the LSM model, ETs for the majority of cases varied approximately between 3 min and 2.5 hrs. (Figures 2.5-2.16), using a standard PC platform, which is quite efficient. However, some of the masked cases, e.g. such as in Figures 2.12 and 2.15, took quite a long time. This problem will be addressed in the next chapter.

The LSM model presented in this chapter provides a foundation for modelling more complex cases, such as the modelling of obliquely shaped masked features which will be considered in the next chapter.

Chapter 3 Level Set Methodology for Predicting the Surface Evolution of Inclined Masked Micro-Channels Resulting from Abrasive Jet Micromachining at Oblique Incidence

3.1 Motivation

In this chapter, a novel implementation of narrow band (NB) LSM [16,37] (Section 1.3) is used to predict the surface evolution of inclined masked micro-channels in glass and PMMA made using AJM at oblique incidence. The formulation extends the LSM model presented in Chapter 2 for masked features machined at normal incidence to the never before considered case of masked features machined at oblique incidence. The resulting inclined micro-channels rapidly become multi-valued, and the Hamilton-Jacobi type partial differential equation describing their evolution cannot be solved using traditional analytical or semi-empirical/computational techniques such as those mentioned in Section 1.2.1. To predict the decrease in particle flux at the mask edge, the previously developed analytical model of [19] described in Section 2.3.3 is generalized from the normal to the oblique incidence case. The local surface velocity function is non-convex (Section 2.3.4.2), necessitating the development of a modified extension velocity methodology to address the problem of grid 'visibility' of the particle flux. The formulation developed in the present chapter ignores mask wear [24,29,32] and particle second strike effects [13,21-24], to be considered in Chapters 4 and 5, respectively. The agreement between LSM-predicted and measured surface evolution, as well as the feasibility of the model in predicting AJM surface evolution of inclined masked features given its present assumptions, is discussed. The majority of the material in this chapter has been published in [46].

3.2 AJM experiments: Masked channels machined at oblique incidence

All AJM experiments were conducted using the same channel blasting apparatus and almost the same experimental conditions as those described in Section 2.2. Some differences and extra details are outlined below.

The masked glass and PMMA channels were machined with the jet at a 45° angle to the surface (Figure 3.1), measured using an angular level which was accurate to approximately 1°. To ensure correct alignment with the small mask opening, a micro-drill bit with a diameter similar to that of the nozzle was temporarily placed in the nozzle, with a protruding length of 20 mm (= h) and aligned with the mask opening through contact. The mask widths, W_m (Figure 3.1), were approximately 450 μm and 430 μm in the machining of glass and PMMA channels, respectively. The measured mass flow rate was in the range of 0.67-2.70 g min^{-1} (see Appendix A, Tables A-9 and A-10), which was low enough for particle interference to be neglected, as in Section 2.2 [33-35].

The machined samples were cross-sectioned at two locations along the channel using a low speed diamond saw, and the cross-sections were compared to ensure repeatability, i.e. to ensure that no chipping occurred during the cross-sectioning. Where necessary, the cross-sections were polished with corundum abrasive paper (type AW-C, grit P-1200) to obtain clear edges. Images of the cross-sections were obtained using a 5 megapixel digital camera attached to a 40X magnification optical microscope. Examples of typical channel cross-sections in glass and PMMA are shown in Figure 3.2. Digital image analysis software (ImageJ, http://rsb.info.nih.gov/ij/) was used to digitize the coordinates of the cross-sectional channel profiles. Between 30-50 and 50-70 data points were obtained for all the resulting glass and PMMA channel profiles, respectively.

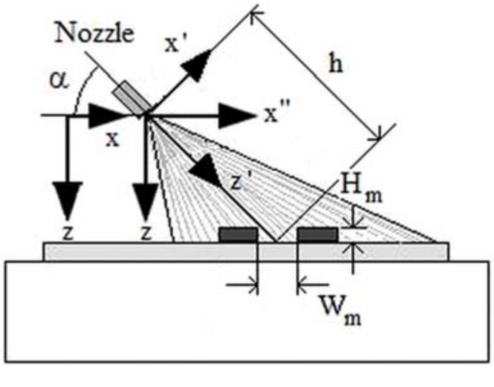
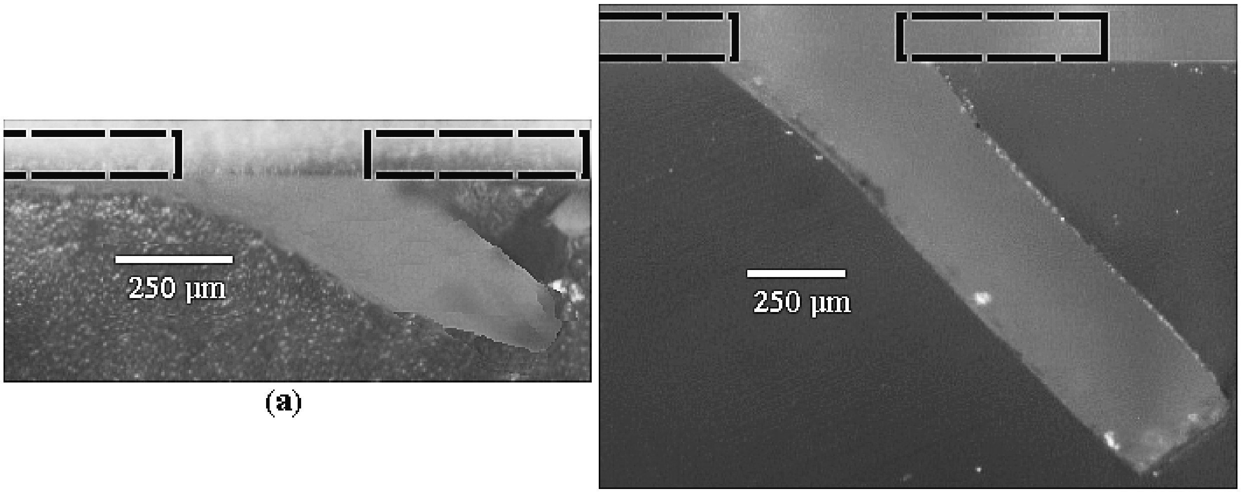


Figure 3.1. Front view schematic in the AJM of oblique incidence masked channels (see Figure 2.1 for more details on the channel blasting apparatus).



(a)

(b)

Figure 3.2. Cross-sections of oblique ($\alpha = 45°$) masked channels in: (a) Glass and (b) PMMA, after 30 passes of the nozzle under the conditions described in Sections 2.2 and 3.2. Dashed lines show approximate original locations of the masks.

3.3 Level set modelling of surface evolution in AJM of oblique masked channels

3.3.1 Local normal velocity function of evolving surface for oblique masked channels

In order to model the masked channels machined at oblique incidence (Figure 3.1), the same transformed coordinate system defined by eq. (2.1) was used as that in Section 2.3.1 and Figure 2.2 for unmasked channels, but with $x_{\mathrm{off}} = W_{\mathrm{m}}/2$. By combining eqs. (2.2) and (2.4), and eqs. (2.3) and (2.5), the velocity of the surface in the direction of the local normal in eq. (1.2) can be re-written conveniently as

$$F_{\mathrm{b}}(x,y,z,t) = \frac{C(\vec{V}(x,y,z,t)\bullet\vec{n})^{k_{\mathrm{v}}}}{\rho}(\vec{\phi}(x,y,z,t)\bullet\vec{n}) \tag{3.1}$$

and

$$F_{\mathrm{d}}(x,y,z,t) = \frac{C(V(x,y,z,t))^{k_{\mathrm{v}}}}{\rho}\left(\frac{\vec{V}(x,y,z,t)\bullet\vec{n}}{\left|\vec{V}(x,y,z,t)\right|}\right)^{n_1}\left[1+Hv\left(1-\frac{\vec{V}(x,y,z,t)\bullet\vec{n}}{\left|\vec{V}(x,y,z,t)\right|}\right)\right]^{n_2}\phi(x,y,z,t) \tag{3.2}$$

for the AJM of glass and PMMA channels, respectively, where all the variables were previously defined in Sections 2.2 and 2.3.2. For the AJM of glass channels, V and ϕ in eq. (3.1) are defined by eqs. (2.7) and (2.9), i.e. scanning target approach, since for the present problem the same blasting conditions were used as in Section 2.2. However, as discussed in Section 2.4.1.1, for the AJM of ductile targets such as the PMMA channels, the 2D approximation for the scanning target used in eq. (3.2) cannot correctly account for the portion of the total erosive efficacy in the y scanning direction that is due to the surface tangential component of velocity; therefore, the stationary target approach whereby $y = y' = (r_{\mathrm{s}} - v_{\mathrm{t}}t) = 0$ in eqs. (2.7) and (2.9) is instead used.

3.3.2 Approximation of decrease in mass flux near the mask edge at oblique incidence

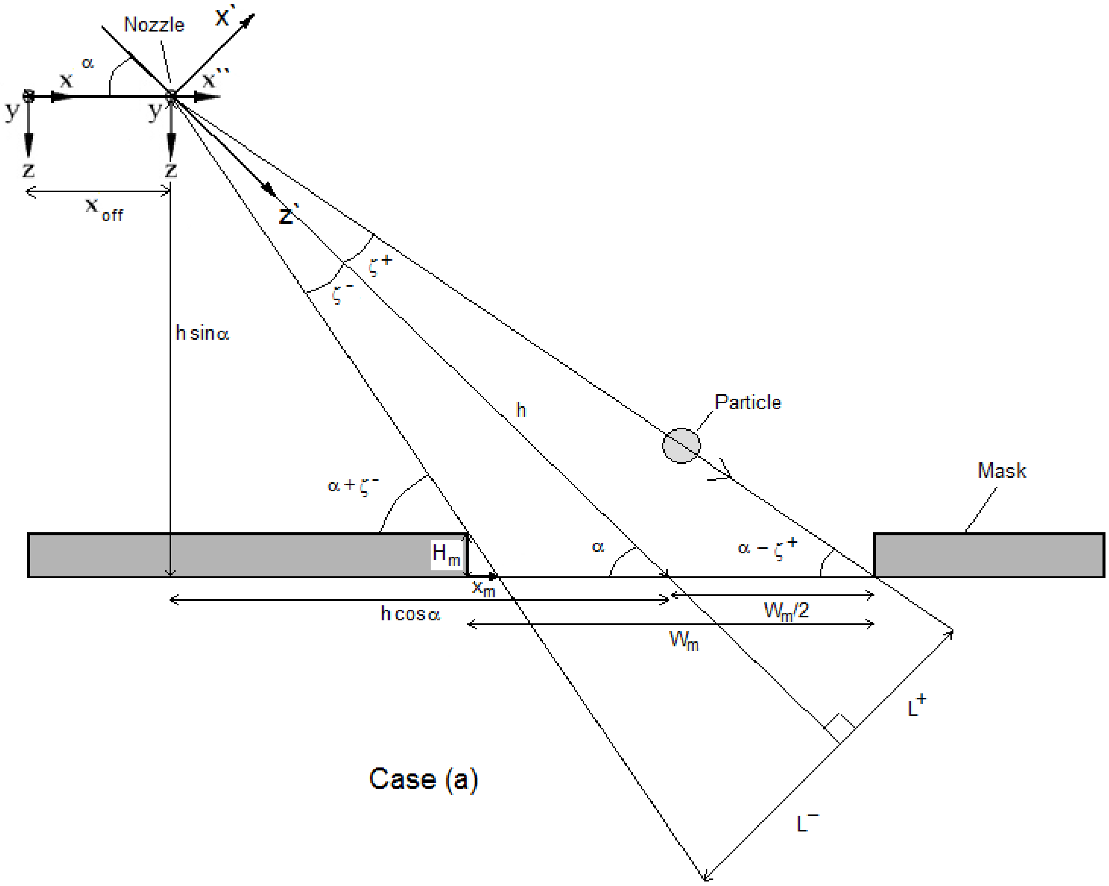
As described in Section 2.3.3 and depicted in Figure 2.3, as the mask edge is approached, only progressively smaller particles can pass through the mask opening without colliding with the mask, and the particle mass flux incident to the surface thus decreases. The effect was modelled in [19] for the case of a non-diverging jet incident perpendicular to the surface, and is now generalized to the oblique incidence case including the effect of the jet divergence. To do this, it was necessary to employ the transformed coordinates of eq. (2.1), and consider the effect of the mask shadow and height, as shown in Figure 3.3. By re-writing eq. (2.10), the resulting expression for the proportion of the total incoming *M* that passes through the mask opening and arrives to the surface at a given *x'* is

$$M_{x/0} = \frac{M(x')}{M(x'=0)} = \frac{\int_0^{L-|x'|} r_\mathrm{p}{}^3 \Omega(r_\mathrm{p}) dr_\mathrm{p}}{\int_0^{L} r_\mathrm{p}{}^3 \Omega(r_\mathrm{p}) dr_\mathrm{p}} \tag{3.3}$$

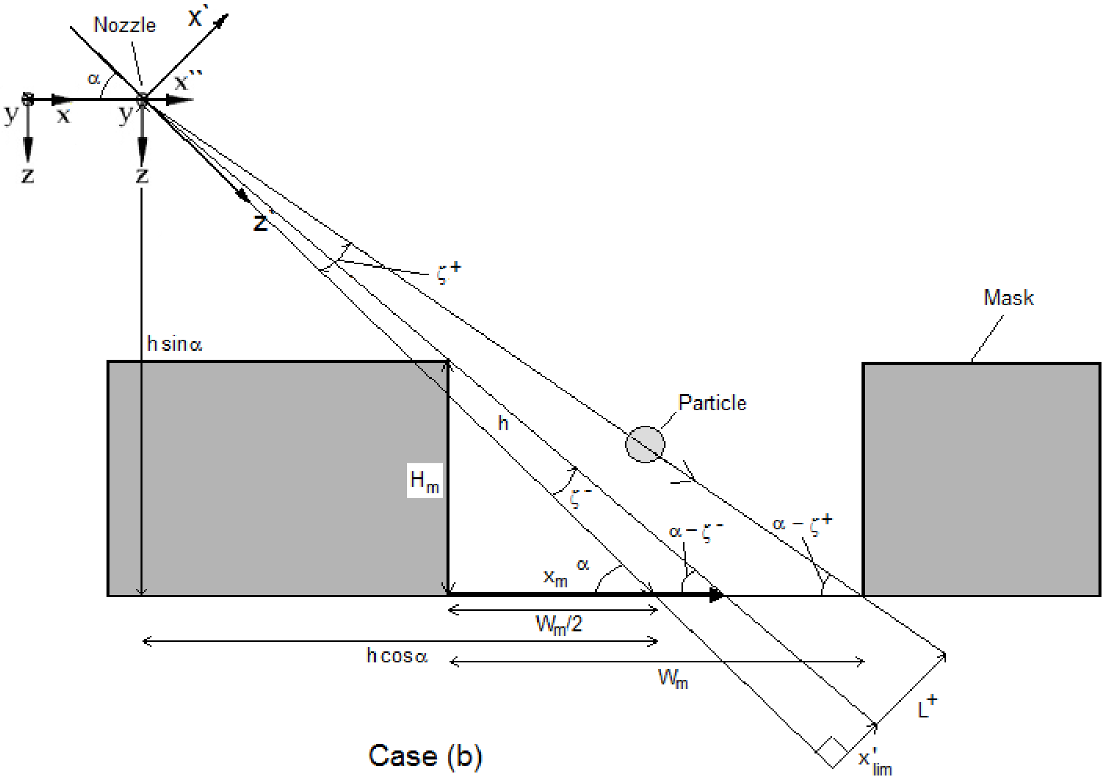
where the same lognormal particle size distribution was used as in eq. (2.11) since the same abrasives were used as in Section 2.2. *L* is the target location that an infinitely small particle can reach without undergoing collision with the edge of the mask, measured along the *x'* direction at a given *z'* (Figure 3.3),

$$L = \begin{cases} L^+ & (x' \geq 0) \\ L^- & (x' < 0) \end{cases} \tag{3.4}$$

The model assumes that the jet centre is aligned with the centre of the mask opening.


Nozzle
x`
α
x
x``
y
y
z
z
z`
x off
ζ+
ζ−
h sinα
h
Particle
Mask
α + ζ−
H m
α
α − ζ+
x m
h cosα
W m/2
W m
L+
L−

Case (a)


Nozzle
x`
α
x``
y
x
y
z
z
z`
ζ+
Mask
h sinα
h
Particle
H m
ζ−
α − ζ−
α − ζ+
x m
α
W m/2
h cosα
W m
L+
x'lim

Case (b)

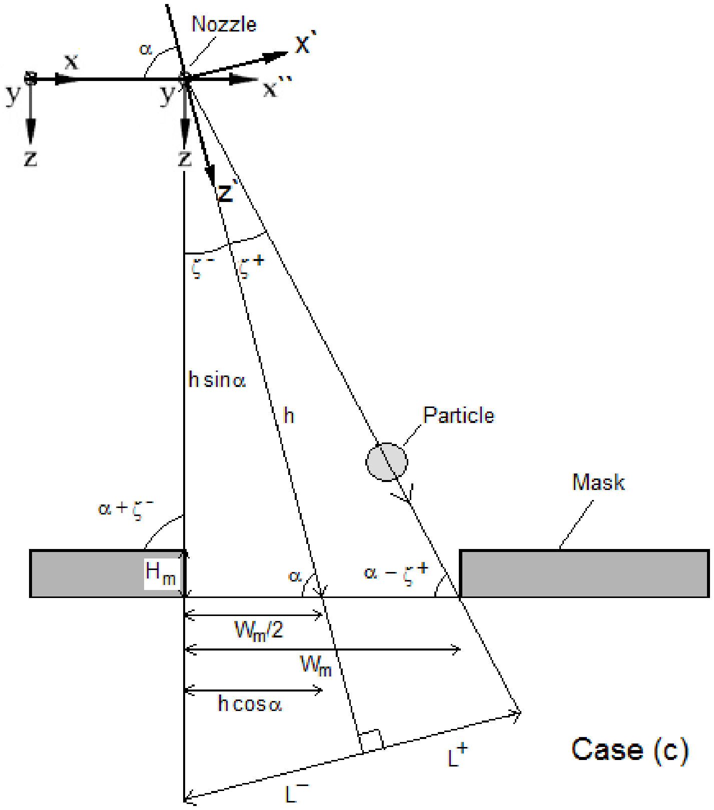


Figure 3.3. Geometry used for modelling flux reduction near mask edges for an inclined jet. x_m, the mask shadow width, measured from the left hand side of the mask opening, reduces the proportion of the target surface in the mask opening that can see incoming particles. Case (a) $0 < x_m < W_m/2$; Case (b) $W_m/2 \le x_m \le W_m$; Case (c) $x_m \le 0$, i.e. no mask shadow. The mask opening width, W_m, is exaggerated with respect to the standoff, h, for clarity.

Depending on the values of W_m, α, h, and H_m, the three cases in Figure 3.3 must be considered, so that L^+ and L^- can be obtained as

$$\left.\begin{aligned} L^+ &= z'\tan(\zeta^+) \\ L^- &= z'\tan(\zeta^-) \end{aligned}\right\} \quad (0 < x_m < W_m/2) \quad (\text{Case a}) \quad \text{or} \quad (x_m \le 0) \quad (\text{Case c})$$

$$\left.\begin{aligned} L^+ &= \begin{cases} z'\tan(\zeta^+) - x'_{\text{lim}} & (x' \ge x'_{\text{lim}}) \\ 0 & (x' < x'_{\text{lim}}) \end{cases} \\ L^- &= \qquad\quad 0 \end{aligned}\right\} \quad (W_m/2 \le x_m \le W_m) \quad (\text{Case b}) \tag{3.5}$$

where x_m is the mask shadow width, measured horizontally from the left hand edge of the mask opening,

$$x_{\mathrm{m}} = \frac{H_{\mathrm{m}}(h\cos\alpha - W_{\mathrm{m}}/2)}{h\sin\alpha - H_{\mathrm{m}}} \tag{3.6}$$

x'_{lim} is defined in Figure 3.3 (b),

$$x'_{\mathrm{lim}} = z'\tan(\zeta^{-}) \tag{3.7}$$

and ζ^{-} and ζ^{+} are the angles defining the maximum particle trajectories that may be incident to the surface through the mask, measured from the jet centreline to the left and right mask edges, respectively,

$$\zeta^{-} = \begin{cases} \tan^{-1}\left(\dfrac{H_{\mathrm{m}}}{x_{\mathrm{m}}}\right) - \alpha & \text{(Case a)} \\ \alpha - \tan^{-1}\left(\dfrac{H_{\mathrm{m}}}{x_{\mathrm{m}}}\right) & \text{(Case b)} \\ \tan^{-1}\left(\dfrac{\sin\alpha}{2h/W_{\mathrm{m}} - \cos\alpha}\right) & \text{(Case c)} \end{cases} \tag{3.8}$$

$$\zeta^{+} = \alpha - \tan^{-1}\left(\frac{h\sin\alpha}{h\cos\alpha + W_{\mathrm{m}}/2}\right) \tag{3.9}$$

The variation of the flux across the mask opening can be obtained by multiplying the mass flux, eq. (2.9), by eq. (3.3). It should be noted that when $x_{\mathrm{m}} < 0$ in eq. (3.6), the negative value holds no physical significance, i.e. $x_{\mathrm{m}} = 0$; however, it is useful to identify the transition to where Case (c) must be used in solving eqs. (3.5) and (3.8).

3.3.2.1 Surface visibility

The modified model presented in Section 3.3.2 can be used to calculate the percentage of unmachined surface visibility, %USV,

$$\%\mathrm{USV} = 100\% \frac{(W_{\mathrm{m}} - x_{\mathrm{m}})}{W_{\mathrm{m}}} \tag{3.10}$$

This quantity is useful for setting up machining runs, since it is difficult to visually discern whether any of the surface is exposed to the jet. A smaller %USV will result in a narrower feature shape and a %USV of 0% will result in no machining of the surface. For example, using eqs. (3.6) and (3.10), with h, α, H_m and W_m (for glass) from Sections 2.2 and 3.2, %USV = 77%, showing that the exposed surface was 'seen' fairly well for the case presented here. Equation (3.10) can also be used in determining the minimum mask width, $W_\mathrm{m,min}$, necessary so that the surface is 'visible' to the nozzle for a given h, α and H_m, for $x_\mathrm{m} > 0$. Substituting %USV = 0% and eqs. (3.6) in (3.10), $W_\mathrm{m,min}$ can be obtained as

$$W_\mathrm{m,min}\Big|_{x_\mathrm{m}>0} = \frac{H_\mathrm{m} h \cos\alpha}{h\sin\alpha - H_\mathrm{m}/2} \tag{3.11}$$

For example, using eq. (3.11), with h, α and H_m from Sections 2.2 and 3.2, $W_\mathrm{m,min} = 100.4\ \mu\mathrm{m}$. Thus, the mask opening must be larger than $W_\mathrm{m,min}$ to allow for machining of the surface.

3.3.3 LSM model implementation

3.3.3.1 Finite differences, signed distance function and geometric variables

In order to approximate the solution to $\Phi(x,z,t)$ and geometric variables, first order FDs in eq. (2.12) were used to approximate the partial derivatives, as explained in Section 2.3.4.1. The initial surface was represented by a horizontal line, and the level set function was initialized at $t = 0$ by using the signed distance function (SDF) defined by eq. (2.29). A useful property of SDFs that can greatly simplify the analysis is that the norm of the gradient (eq. (2.13)) of the level set function must be unity [37],

$$\left|\nabla\Phi(x,z)\right| = 1 \tag{3.12}$$

However, as explained in Section 2.3.4.4, as the surface evolves for $t > 0$, $\Phi(x,z,t)$ generally deviates from the initial value of the signed distance due to numerical instability. In order to ensure that $\Phi(x,z,t)$ remains equal to the signed distance and hence ensure that eq. (3.12) remains valid, $\Phi(x,z,t)$ must be re-initialized at fixed time intervals [37]. To ensure a high degree of accuracy, $\Phi(x,z,t)$ was re-initialized every time step; although this is computationally expensive, it is simple to implement. Using eq. (3.12), $\vec{n}$ and K in eq. (2.14) can be re-defined simply as [37]

$$\vec{n} = \frac{\nabla\Phi}{|\nabla\Phi|} = (\Phi_x, \Phi_z)\,,\ K = \nabla\cdot\frac{\nabla\Phi}{|\nabla\Phi|} = \Phi_{xx} + \Phi_{zz} \tag{3.13}$$

3.3.3.2 Simplified LSM for non-convex Hamiltonians

As was shown in Section 2.3.4.2, the LSM equation of motion for non-convex Hamiltonians, H, is defined by eq. (2.21). The methodology presented here is the same as that used in Section 2.3.4.2; however, it is greatly simplified due to the definition in eq. (3.12). In addition, this definition allows the extension velocity methodology presented in Section 3.3.3.3, and necessary in the present formulation, to be extended to cases where the Hamiltonian function, i.e. the velocity function, is non-convex. For the present case, the surface evolution did not depend on curvature, i.e. $\varepsilon = 0$, since the smoothing of the evolving surface profiles was not necessary. This resulted due to the application of the scanning target approach in modelling the erosive efficacy for glass, which delays the formation of cusps at the profile centres, and the fact that curvature-based surface evolution need not be considered in modelling ductile targets, i.e. PMMA, as explained in Section 2.4.1.1. Thus, eq. (2.21) can be reduced to

$$\frac{\partial\Phi}{\partial t} + \hat{H} = 0 \tag{3.14}$$

where once again, $\hat{H}$ is the numerical Hamiltonian defined by eq. (2.22) along with eq. (2.23). Using eq. (3.12), H in eq. (2.22) can be reduced to [37]

$$H = F(x, z, t) \tag{3.15}$$

Combining eqs. (3.1)-(3.3), (3.12), (3.13), and (3.15), and expressing the result in the x'' and z components, the H for glass and PMMA defined by eqs. (2.17)-(2.19), respectively, can be reduced to

$$H_{\mathrm{b}} = M_{x/0}\frac{C}{\rho}V^{k_{\mathrm{v}}}\phi(\cos\theta)^{k_{\mathrm{v}}+1} \tag{3.16}$$

and

$$H_{\mathrm{d}} = M_{x/0} \frac{C}{\rho} V^{k_{\mathrm{v}}} \phi (\cos\theta)^{n_1} \left[1 + Hv(1-\cos\theta)\right]^{n_2} \tag{3.17}$$

with

$$\cos\theta = \frac{x''\Phi_x + z\Phi_z}{\sqrt{x''^2 + z^2}} \tag{3.18}$$

where θ is the angle between $\vec{n}$ and the particle impact velocity vector $\vec{V}$ (Figure 2.2), V and ϕ are obtained using eqs. (2.7) and (2.9), respectively, as described in Section 3.3.1, and $M_{x/0}$ is now defined by eq. (3.3).

Finally, the partial derivatives of eq. (3.16) and eq. (3.17) with respect to Φ_x and Φ_z, H_{Φ_x} and H_{Φ_z}, used in eq. (2.23), are obtained as

$$(H_{\mathrm{b}})_{\Phi_x;\Phi_z} = H_{\mathrm{b}} \left[\frac{(x'';z)(k_{\mathrm{v}}+1)}{\sqrt{x''^2+z^2}\cos\theta} \right] \tag{3.19}$$

and

$$(H_{\mathrm{d}})_{\Phi_x;\Phi_z} = H_{\mathrm{d}} \left[\frac{(x'';z)}{\sqrt{x''^2+z^2}} \left(\frac{n_1}{\cos\theta} - \frac{n_2 Hv}{1+Hv(1-\cos\theta)} \right) \right] \tag{3.20}$$

where the notation (*x''*; *z*) in eqs. (3.19) and (3.20) indicates that *x''* is used to calculate H_{Φ_x} and *z* to calculate H_{Φ_z}. Equations (3.19) and (3.20) are the simplified versions of eqs. (2.24)-(2.27) due to eq. (3.12).

3.3.3.3 Extension velocity methodology for non-convex Hamiltonians

The velocity function $F(\vec{x},t)$ has physical meaning only for the zero level set, Φ^{o}, i.e. $\Phi(\vec{x},t) =$ 0 (Section 1.2.3), which defines the actual surface evolution [16]. However, since eq. (1.2) is written for the function $\Phi(\vec{x},t)$ defined over the entire computational space, the equation must have a consistent physical meaning for all the level sets on the grid, i.e. at every $\vec{x}$, so that Φ^{0} is allowed to propagate naturally [16]. The velocity function $F(\vec{x},t)$ obtains its physical meaning from the position of Φ^{0} and not the geometry of $\Phi(\vec{x},t)$. The use of a mask (Section 3.3.2) introduces a boundary beyond which the level sets are slowed down since only the portion of the incoming particle jet which is outlined by the mask is 'visible' to the grid. As a result, Φ^{0} will stop once that boundary is reached, as depicted in Figure 3.4. For the present formulation, where only the surface evolution of the target is considered, as opposed to the surface evolution of both the target and the mask which will be considered in Chapter 4, this effect occurs only at oblique incidence, since at normal incidence, the entire jet is 'visible' to the grid, and visibility is thus not a problem. As a result, for the oblique case, the velocity function must be *extended* from the surface, i.e. Φ^{0}, to all other level sets [16]. Henceforth, this procedure will be referred to as the extension velocity methodology (EVM).

When there is no natural choice available for an extension velocity, as in the present case, the most common method of implementing EVM is to assign to every point $\vec{x}$ in the domain, the velocity of the nearest point on the surface, $\vec{a}$ [16],

$$F_{\text{ext}}(\vec{x}) = F(\vec{a}), \vec{a} \in (\Phi = 0) \tag{3.21}$$

where F_{ext} is the extension velocity. Although this step is computationally expensive, it is necessary in order to ensure that the level sets do not collide. It should be emphasized that the sole purpose of F_{ext} is to force the motion of the level sets in the vicinity of Φ^{0}, and it need not correspond to the velocity implied by the physics of the problem [16]. The only requirement is that it must equal the velocity at Φ^{0}, as the distance between $\vec{x}$ and $\vec{a}$ approaches zero [16], which eq. (3.21) implies.

EVM is usually applied to cases where the H is convex, meaning that eq. (2.15) (Section 2.3.4.2) without considering motion with curvature, i.e. $\varepsilon = 0$, can be used to propagate the surface with time, instead of eq. (3.14). Equation (2.15) with $\varepsilon = 0$ can be solved by using simpler upwind FD schemes that ensure FD approximations follow the exact-solution PDE theory [16,37], where the evaluation of the partial derivative of H with respect to Φ_g is not required. In such cases, the extension velocity can be easily applied by substituting F_{ext}, eq. (3.21), for F in eq. (3.15) [16,37]. However, for the present problem, H is non-convex, meaning that F_{ext} must also be applied to determine eqs. (3.19) and (3.20) in solving eq. (2.23) defined in Section 2.3.4.2. This necessitated a novel extension of EVM theory in order to treat these non-convex H cases, by factoring out F in eqs. (3.19) and (3.20), using eq. (3.15) and applying eq. (3.21), to obtain

$$H_{\text{ext}} = F_{\text{ext}}(x, z, \Phi_x, \Phi_z) = F(x^{\text{o}}, z^{\text{o}}, \Phi_x^{\text{o}}, \Phi_z^{\text{o}}), \quad (x^{\text{o}}, z^{\text{o}}, \Phi_x^{\text{o}}, \Phi_z^{\text{o}}) \in (\Phi = 0) \tag{3.22}$$

where the superscript ‘o’ indicates that the quantity in question is obtained at the point on Φ^{o} which is closest to the grid point (x, z). H_{b} and H_{d} in eqs. (3.19) and (3.20) are replaced with $H_{\text{b,ext}}$ and $H_{\text{d,ext}}$, according to eq. (3.22), in order to obtain $(H_{\text{b,ext}})_{\Phi_x;\Phi_z}$ and $(H_{\text{d,ext}})_{\Phi_x;\Phi_z}$, respectively. $H_{\text{b,ext}}$ and $H_{\text{d,ext}}$ are evaluated using eqs. (3.16) and (3.17) but with x^{o}, z^{o}, Φ_x^{o} and Φ_z^{o} in place of x, z, Φ_x and Φ_z, respectively. All remaining quantities in eqs. (3.19) and (3.20) are evaluated with x, z, Φ_x and Φ_z, i.e. the grid node values. These new expressions are then used in place of H and H_{Φ_g} in eqs. (2.22) and (2.23), in order to solve eq. (3.14).

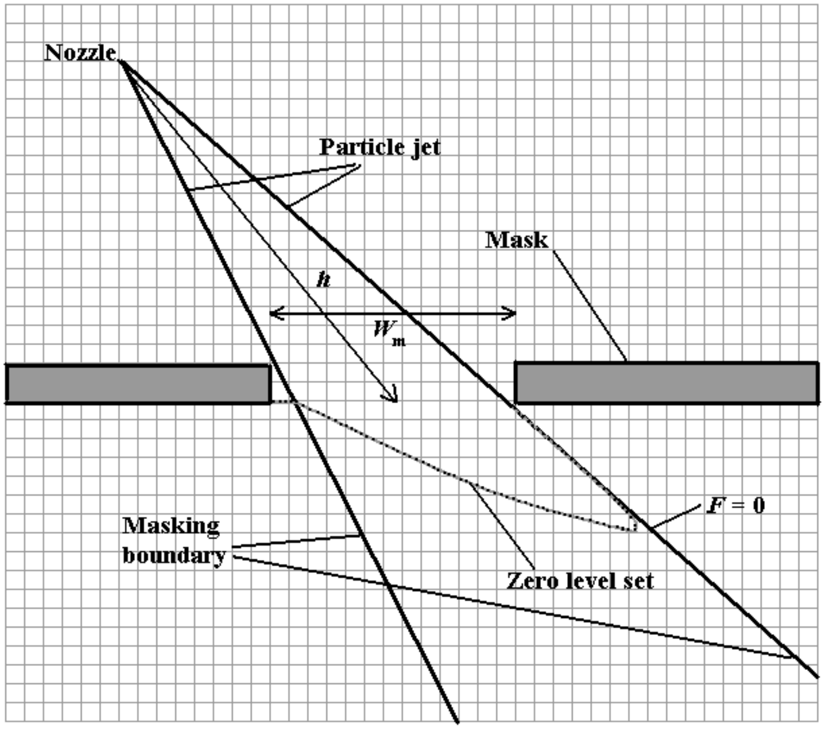


Figure 3.4. Portion of the jet which is outlined by the mask that is ‘visible’ to the grid. Without velocity extension, the level sets are stationary beyond the masking boundary. W_{m} is exaggerated with respect to h for clarity.

3.3.3.4 Optimization using the narrow band LSM

As mentioned in Section 1.3, the narrow band (NB) LSM [16] is an adaptive scheme that is based on the notion that calculations need only be performed for points in the vicinity of the surface, since only the Φ^{0} has any physical meaning. Thus, in the NB LSM, a narrow band is defined around the Φ^{0} where all computations are only performed at grid points, (x, z), within this boundary, or 'tube'. As a result, when compared to the full-grid LSM approach, less grid points are considered for each iteration step, thus greatly improving the computational efficiency [16]. The NB LSM also has the advantage that the extension velocity described in Section 3.3.3.3 need only be calculated for points situated within the narrow band [16], which makes the method even more computationally efficient for cases such as the present, where EVM is required.

The band is initialized based on the position and shape of Φ^{0}, by searching the grid for $\Phi = D_{\mathrm{UB}}$, which forms the 'upper band' ahead of Φ^{0}, and for $\Phi = -D_{\mathrm{LB}}$, which forms the 'lower band' behind Φ^{0}. D_{UB} and D_{LB} are the band widths measured from the Φ^{0} to the upper and lower bands, respectively. As Φ^{0} evolves and approaches the boundary, calculations are temporarily halted and a new band is re-initialized around Φ^{0}. $\Phi(x,z,t)$ for all the (x, z) within the new band are calculated with the SDF defined by eq. (2.29), while $\Phi(x,z,t)$ for all (x, z) on or outside the 'tube' are frozen. Since re-initialization is computationally expensive, the size of the band is chosen as a compromise between assigning a band width large enough to prevent recurrent re-initialization, and small enough to not include too large a domain. In the present work, $D_{\mathrm{UB}} = 4\Delta z$ ($= 4\Delta x$) and $D_{\mathrm{LB}} = 2\Delta z$, where Δx and Δz are the horizontal and vertical spatial grid steps, respectively, were found to be appropriate by maintaining a good balance between computational effort and band re-initialization. Re-initialization occurred when a minimum distance between Φ^{0} and the upper or lower bands, D_{min}, was reached. A $D_{\mathrm{min}} = \Delta z/2$ was used, which ensured that Φ^{0} did not pass the boundary while also minimizing the frequency of re-initialization. D_{min} was evaluated at each time step. The whole process was repeated until the required propagation time was reached.

3.3.3.5 Grid formulation, boundary conditions and time step

For the present problem, the vertical grid limits were the same as that presented in Section 2.3.4.3. Using the geometry of Figure 3.3, the horizontal grid limits were defined as $x_{\mathrm{min}} = x_{\mathrm{off}} + h\cos\alpha - W_{\mathrm{m}}/2 + x_{\mathrm{m}}$ and $x_{\mathrm{max}} = x_{\mathrm{off}} + h\cos\alpha + W_{\mathrm{m}}/2 + z_{\mathrm{surf}}/\tan(\alpha - \zeta^{+})$. These limits were then used to calculate the spatial grid steps and global spatial coordinates at the grid nodes as shown in Section 2.3.4.3. The boundary conditions were obtained in the same way as in Section 2.3.4.3, where now the computational domain was defined by the narrow band boundary, as opposed to the entire grid. Finally, Δt was once again calculated using the CFL condition [37], and with $\varepsilon = 0$ and extension, eq. (2.28) can be re-defined as

$$\Delta t \cdot \max\left(\sum_{g=x,z} \frac{\left| (H_{\text{ext}})_{\Phi_g} (\nabla\Phi^{\pm}) \right|}{\Delta g} \right) < 1 \tag{3.23}$$

The maximum was evaluated by searching within the entire narrow band.

3.3.3.6 Surface partial derivatives and interpolation

As mentioned in Section 2.3.4.4, in order to re-initialize the level set function using the SDF of eq. (2.29), and to visualize the evolving surface profile for $t > 0$, the surface, i.e. Φ^{o}, must be interpolated since it is usually located between the grid nodes as shown in Figure 3.5. The entry/exit point of the zero level set shown in Figure 3.6 was interpolated linearly in order to maintain monotonicity and hence stability (Section 2.3.4.4). In addition, in order to extend the velocity from Φ^{o} to other level sets and thus evaluate the $H_{\text{b,ext}}$ and $H_{\text{d,ext}}$ described in Section 3.3.3.3, it was necessary to obtain FD approximations of the partial derivatives, Φ_x^{o} and Φ_z^{o}, and coordinates, x^{o} and z^{o}, for the surface entry/exit point. x^{o} and z^{o} were obtained with linear interpolation based on the value of Φ^{o} and adjacent nodes. Following eq. (2.12), forward, backward and central FD approximations of Φ_x^{o} and Φ_z^{o} were obtained for Case (a) and Case (b) in Figure 3.6 using the following relations

$$\left(\Phi_x^+\right)^{\mathrm{o}} = \begin{cases} \dfrac{\Phi^{\mathrm{GH}} - 0}{\Delta x} & \text{(Case a)} \\[2ex] \dfrac{\Phi^{\mathrm{RU}} - 0}{\Delta x} & \text{(Case b)} \end{cases} \qquad \left(\Phi_z^+\right)^{\mathrm{o}} = \begin{cases} \dfrac{\Phi^{\mathrm{JK}} - 0}{\Delta z} & \text{(Case a)} \\[2ex] \dfrac{\Phi^{\mathrm{TW}} - 0}{\Delta z} & \text{(Case b)} \end{cases}$$

$$\left(\Phi_x^-\right)^{\mathrm{o}} = \begin{cases} \dfrac{0 - \Phi^{\mathrm{EF}}}{\Delta x} & \text{(Case a)} \\[2ex] \dfrac{0 - \Phi^{\mathrm{PS}}}{\Delta x} & \text{(Case b)} \end{cases} \qquad \left(\Phi_z^-\right)^{\mathrm{o}} = \begin{cases} \dfrac{0 - \Phi^{\mathrm{BC}}}{\Delta z} & \text{(Case a)} \\[2ex] \dfrac{0 - \Phi^{\mathrm{NQ}}}{\Delta z} & \text{(Case b)} \end{cases}$$

$$\left(\Phi_x^{\mathrm{c}}\right)^{\mathrm{o}} = \begin{cases} \dfrac{\Phi^{\mathrm{GH}} - \Phi^{\mathrm{EF}}}{2\Delta x} & \text{(Case a)} \\[2ex] \dfrac{\Phi^{\mathrm{RU}} - \Phi^{\mathrm{PS}}}{2\Delta x} & \text{(Case b)} \end{cases} \qquad \left(\Phi_z^{\mathrm{c}}\right)^{\mathrm{o}} = \begin{cases} \dfrac{\Phi^{\mathrm{JK}} - \Phi^{\mathrm{BC}}}{2\Delta z} & \text{(Case a)} \\[2ex] \dfrac{\Phi^{\mathrm{TW}} - \Phi^{\mathrm{NQ}}}{2\Delta z} & \text{(Case b)} \end{cases} \qquad (3.24)$$

where, for example, Φ^{GH} was obtained using linear interpolation between the nodes G and H, etc.

Finally, in construction of the narrow band of Section 3.3.3.4, the upper and lower bands were also located through linear interpolation between the grid nodes, based on the position of Φ^{o}.

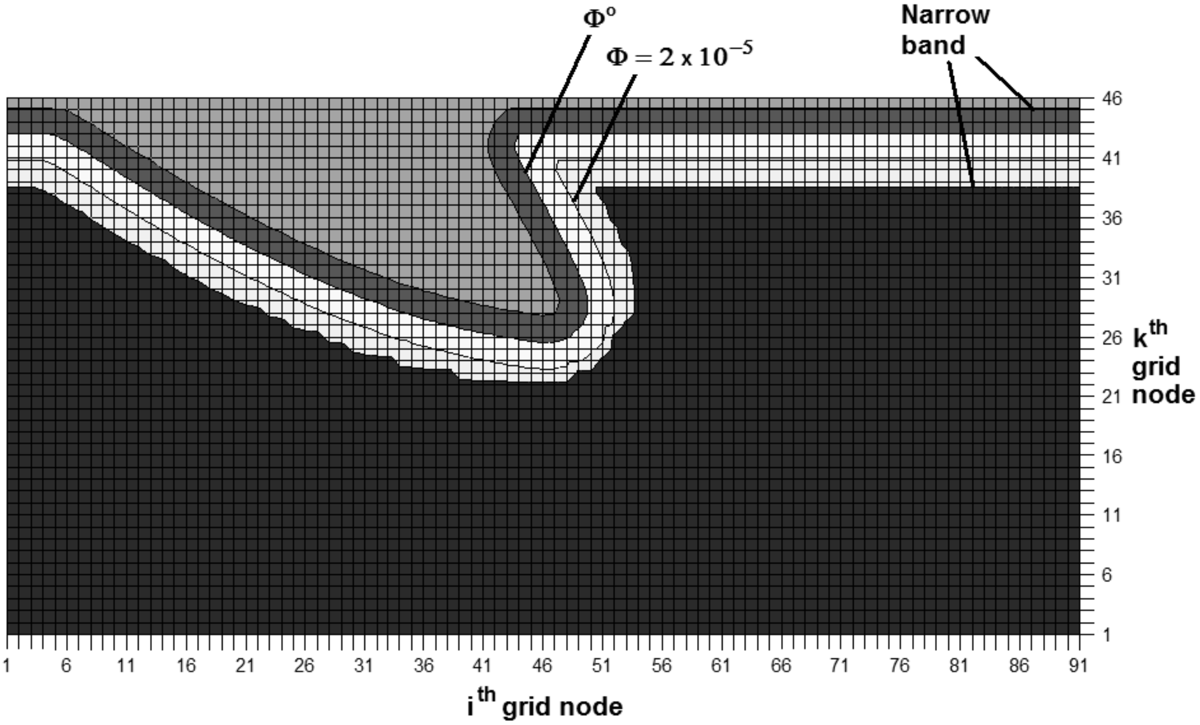
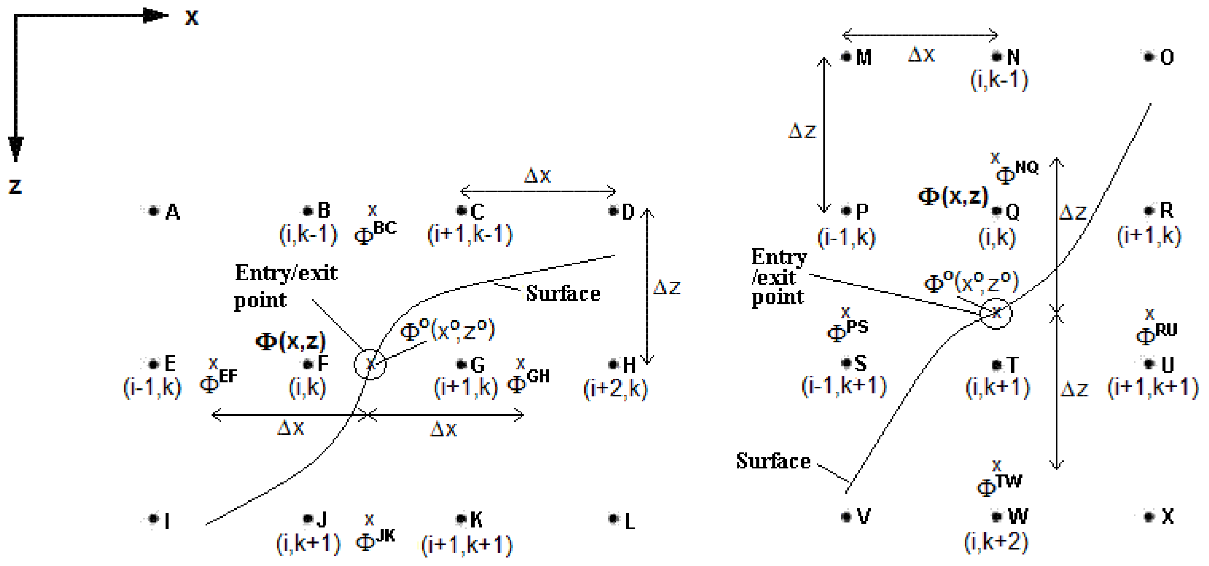


Figure 3.5. The level set function, Φ, on the computational grid, along with the zero level set, Φ^o, i.e. the location of the machined surface, and the narrow band, after 6 passes in the machining of an inclined masked channel in glass at $\alpha = 45°$.



Case (a) Case (b)

Figure 3.6. FD approximation of partial derivatives for the zero level set, Φ^o, for a surface entry/exit point located in between the grid nodes: Case (a) along the *x*-direction; Case (b) along the *z*-direction, used in eq. (3.24). The dots labelled A to X represent grid points, where F and Q are the reference points, and the x's represent locations of interpolation. The value of Φ^o at the circled entry/exit point is obtained using linear interpolation. When Φ^o passes through a grid point, e.g. F, eq. (3.24) reduces to eq. (2.12) at that point, and no interpolation is necessary.

3.3.3.7 Summary of algorithm

The algorithm used in solving eq. (3.14) can be summarized as follows:

1. Initialize Φ with the SDF, eq. (2.29), at each grid point (*i*,*k*). Build a narrow band around the Φ^o, as described in Section 3.3.3.4.
2. For the initial iteration, $m = 0$, where $m = 0,1,2,\ldots$ is the iteration number, compute H using either eq. (3.16) or (3.17), and H_{Φ_g} using either eq. (3.19) or (3.20), for each grid point (*i*,*k*) inside the narrow band without the use of extension. For $m > 0$, compute the extended Hamiltonian H_{ext} using eq. (3.16) or (3.17), with x^o, z^o, $\Phi_x{}^o$ and $\Phi_z{}^o$ obtained as described in Section 3.3.3.6, and $(H_{\text{ext}})_{\Phi_g}$ using eq. (3.19) or (3.20), with x^o, z^o, $\Phi_x{}^o$ and $\Phi_z{}^o$ and x, z, Φ_x and Φ_z, as explained in Section 3.3.3.3, for each grid point (*i*,*k*) inside the narrow band.

3. Solve eq. (3.14) with eqs. (2.22) and (2.23) (Section 2.3.4.2) using the unextended ($m = 0$) or extended ($m > 0$) quantities from step 2, along with Φ_m, to obtain Φ_{m+1}.
4. Using linear interpolation, obtain a set of points that represent the physical surface (i.e. Φ^{o}), $\{(x^{\mathrm{o}}, z^{\mathrm{o}}) \mid \Phi^{\mathrm{o}}\}$, as explained in Section 3.3.3.6.
5. If Φ^{o} reaches $D_{\min}$ defined in Section 3.3.3.4, stop the computation and re-build a new upper and lower band around the Φ^{o}; otherwise, continue to the next step.
6. Re-initialize Φ with the SDF, eq. (2.29), at each grid point (i,k) in the narrow band.
7. Repeat steps 2-6 until the simulation reaches the desired time.

3.4 Results and discussion

3.4.1 Model execution and inputs

The model presented in Section 3.3 was implemented in MATLAB 7.7 (The MathWorks, Inc., Natick, MA, USA), as in Chapter 2. The LSM predicted surface evolution profiles are compared to the measured ones in Figures 3.7 and 3.8. Using the same PC as outlined in Section 2.4.1.1, the ETs were 9 min and 76 min for the simulation of channels machined in glass (Figure 3.7) and PMMA (Figure 3.8), respectively, using the NB LSM approach described in Section 3.3.3.4. The implementation of the NB LSM approach decreased the ETs by approximately 7 and 3 times for the glass and PMMA channels, respectively, when compared to simulations using the full-grid approach. Using the D_{UB}, D_{LB} and $D_{\min}$ presented in Section 3.3.3.4, the band was re-initialized 12 and 20 times, approximately every 220 and 1200 time steps, for the glass and PMMA channels shown in Figures 3.7 and 3.8, respectively.

A summary of the required model inputs appears in Table 3.1. The parameters h, H_{m}, W_{m}, α, v_t, and ρ were obtained based on experimental conditions of Section 3.2 and are specified in Table 3.1. The measured particle mass flow rate, $\dot{M}$, given in Table 3.1, was assumed to linearly decrease with time to account for the mass flow rate fluctuations, as described in Section 2.4.1.1 (see Appendix A, Tables A-9 and A-10). The parameters β, V_{o}, μ_1, σ_1, k_{v}, C, n_1, n_2 and Hv specified in Table 3.1 were obtained from Section 2.4.1.1 based on previous measurements for the same nozzle, jet conditions, abrasives, and target materials as presently used.

The grid dimensions, $i_{\max} \cdot k_{\max}$, and z_{surf} defined in Section 2.3.4.3, given in Table 3.1, were chosen such that, for the channels machined in glass, $\Delta x = \Delta z = 8.8$ μm, while for PMMA, $\Delta x = \Delta z = 20$ μm, which ensured the convergence and accuracy of the numerical solution as described in Section 2.4.1.1. For the present simulations, representative mean values of Δt determined by eq. (3.23) were 5.6 x 10^{-2} s and 1.8 x 10^{-2} s for glass and PMMA, respectively. The simulation time was set based on the maximum number of required nozzle passes (Table 3.1), in multiples of $2r_{\mathrm{s}}/v_t$ (Section 2.3.2).

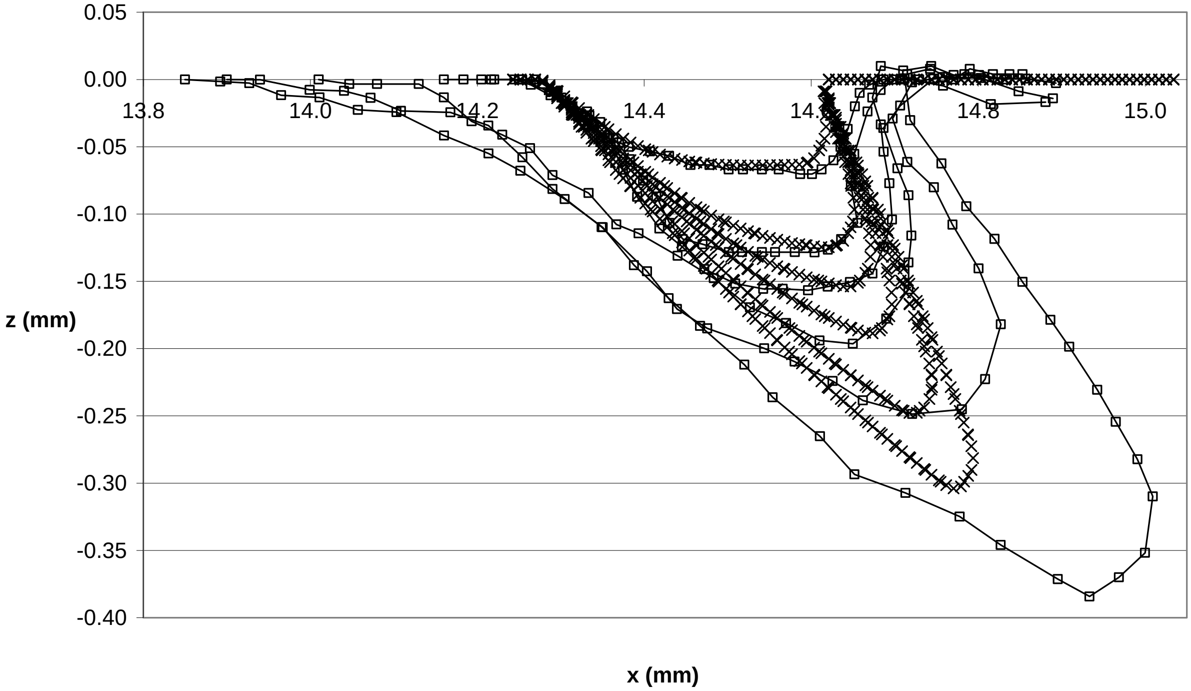


Figure 3.7. Predicted (×) and measured (-□-) surface evolution of masked channels machined in glass at $\alpha = 45°$ after 2, 4, 6, 10, 20 and 30 passes of the nozzle. $\dot{M} = (2.70 - 9.65 \times 10^{-3} \cdot t\ (\mathrm{s}))$ g min^{-1}. All other model inputs are specified in Table 3.1.

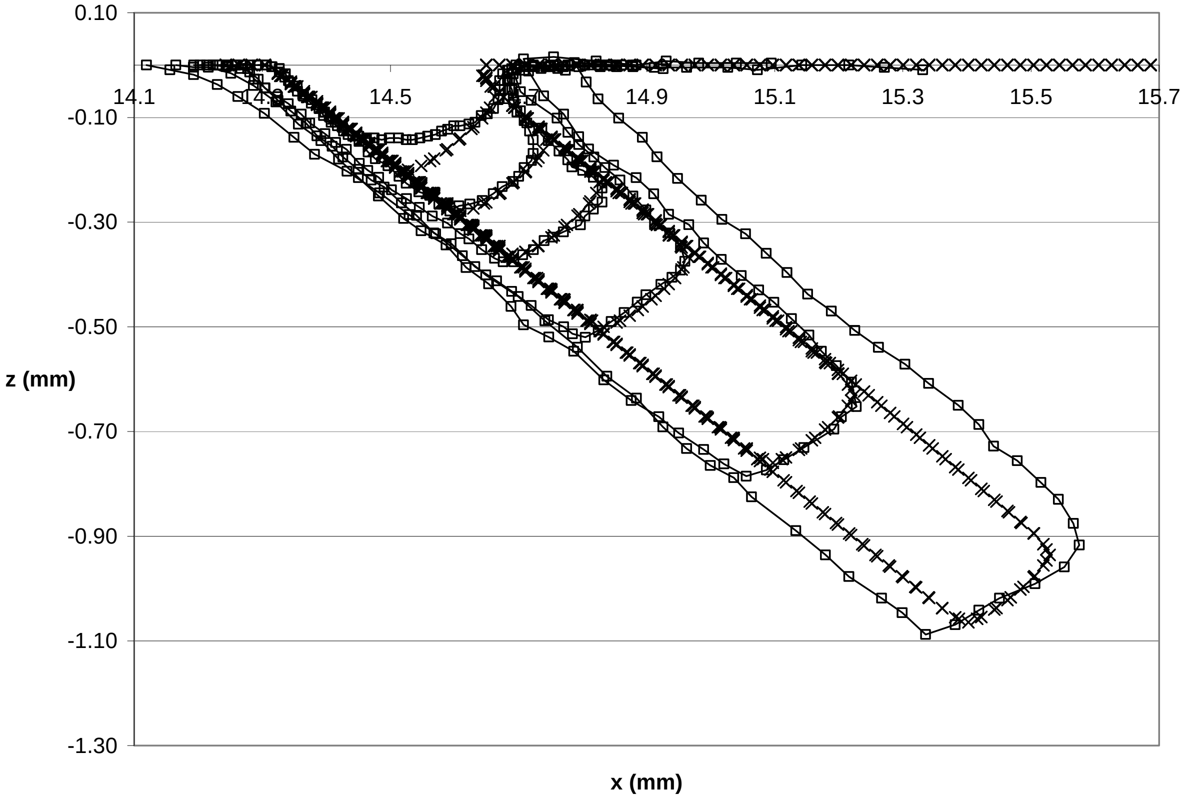


Figure 3.8. Predicted (×) and measured (-☐-) surface evolution of masked channels machined in PMMA at $\alpha = 45°$ after 2, 4, 6, 10, 20 and 30 passes of the nozzle. $\dot{M} = (1.86 - 1.44 \times 10^{-3} \cdot t\ (\text{s}))\ \text{g min}^{-1}$. All other model inputs are specified in Table 3.1.

Table 3.1. Model inputs.

Model inputs	Figure 3.7 (glass)	Figure 3.8 (PMMA)	Figure 3.9 (glass)
h (mm)	20	20	20
H_m (μm)	100	100	100
W_m (μm)	450	430	550
α (°)	45	45	45
v_t (mm s^{-1})	1.0	0.5	1.0
ρ (kg m^{-3})	2200	1190	2200
$\dot{M}$ (g min^{-1})	2.70 -9.65 x $10^{-3}\cdot t$ (s)	1.86 -1.44 x $10^{-3}\cdot t$ (s)	2.70 -9.65 x $10^{-3}\cdot t$ (s)
β ( )	15 [14]	15 [14]	15 [14]
V_o (m s^{-1})	162 [14]	162 [14]	162 [14]
μ_l ( )	-11.6 [19]	-11.6 [19]	-11.6 [19]
σ_l ( )	0.5 [19]	0.5 [19]	0.5 [19]
k_v ( )	1.43 [14]	2.0 [2]	1.43 [14]
C (m s^{-1})$^{-k_\mathrm{v}}$	8.0 x 10^{-6} [36]	5.7 x 10^{-8} [36]	8.0 x 10^{-6} [36]
n_1 ( )	-	1.27 [2]	-
n_2 ( )	-	15.5 [2]	-
Hv (GPa)	-	0.25 [2]	-
i_max ( )	91	76	115
k_max ( )	90	60	90
z_surf (μm)	780	1200	780
Max. no. of nozzle passes ( )	30	30	30

3.4.2 Fits of model to experiments

It is evident from Figures 3.2, 3.7 and 3.8 that the measured profiles of the inclined masked channels for glass and PMMA differ in shape, as was also noted previously for channels machined at normal incidence [2]. The glass profiles in Figure 3.2 have curved walls and rounded bottoms, while the PMMA profiles have straighter walls and more rectangular bottoms. This difference in shape arises due to the different erosion laws that govern the surface evolution of the two target materials, i.e. eqs. (3.1) and (3.2), or in more expanded form, eqs. (3.16) and (3.17). For PMMA, the local velocity of surface erosion is highest when the particle incident velocity vectors are at a shallow angle to the surface, i.e. when θ in Figure 2.2 is large. This has the tendency to rapidly create side walls which are approximately parallel to the incident velocity vectors, i.e. approximately parallel to the nozzle inclination angle, while the bottom of the channel, which is approximately at normal incidence to the particles, advances relatively slowly. This alignment of the walls with the inclination angle of the nozzle does not occur for glass because its maximum erosion rate occurs at normal incidence, i.e. at the bottom of the channel, and not at the shallower angles on the inclined side walls. The inclined straight walls in the PMMA targets may be desirable in the micro-fabrication of ‘V’ grooves in microstructures such as hydraulic resistors [47] and pressure-flow sensors [48].

For the case of the glass micro-channels in Figure 3.7, there was quite a significant discrepancy between the measured and modelled profiles, both in terms of depth and overall shape, although the depths were predicted quite well up to 10 passes. The discrepancy in shape is likely due to a combination of mask wear [24,29,32] and particle second strike effects [13,21-24], that the present model cannot account for. It is hypothesized that the particles arriving to one side of the target profile ricochet, and strike the opposite side, resulting in the significant under-etching seen in the experimental profiles, as well as deeper experimental profiles than those predicted by the model. In addition, particle mask-to-target ricochet, i.e. mask edge effect, [23] can further contribute to this effect.

As the surface became deeper, the glass experimental profiles tended to shift their direction of propagation to the right, likely caused by the combination of increase in mask width due to mask wear with time and second strike particles. This can be seen by comparing the predicted and experimental profiles after 20 and 30 passes of the nozzle in Figure 3.7. The effect of mask wear in causing this change in propagation direction is partially demonstrated in Figure 3.9, where the modelled profiles of Figure 3.7 with W_m = 450 μm are compared against modelled profiles with a representative worn mask having a width of W_m = 550 μm. This approximate value was obtained by measuring the mask width after machining using a reference gage and micro callipers. It can be seen that the profiles with the larger mask opening are shifted more to the right and travel deeper. Assuming that most of the mask wear occurred between 10 to 20 passes, and W_m increased from 450 μm to 550 μm during that time, the change in surface propagation direction is also indicated by the solid line shown in Figure 3.9. This is, of course, a simplified view of the increase in mask width, which, in reality, would be a more complex function of the erosion mechanism of the mask and second strike effects, as indicated by the experimental profiles in Figure 3.7.

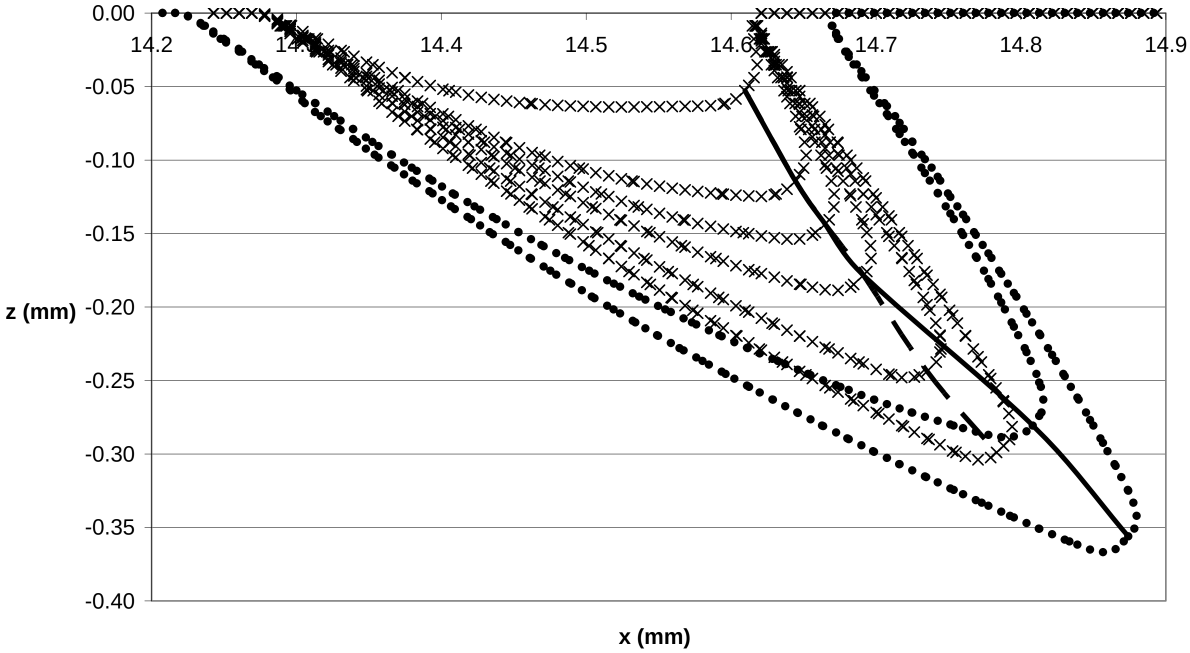


Figure 3.9. Depiction of the relation between mask wear and profile propagation direction. Model simulation of masked glass channels at $\alpha = 45°$ for $W_m = 450$ μm (×: Figure 3.7) and $W_m = 550$ μm (•: 20 and 30 passes. ET = 12 min, with band re-initialized 14 times approximately every 180 time steps and mean $\Delta t = 6.4 \times 10^{-2}$ s. All other model inputs are specified in Table 3.1). The solid and dashed line indicates the propagation direction with and without mask wear, respectively. The lines connect the points of highest curvature of each profile.

Figure 3.8 shows a good agreement between the predicted and measured profiles of the micro-channels machined in PMMA, both in terms of depth and overall shape. The agreement is better when compared to the predicted and measured profiles in glass (Figure 3.7). Since the PMMA channels develop straight sidewalls which are approximately aligned with the incoming particle velocity vectors, ricochet and second strike of particles to the opposite side wall is unlikely. In addition, the effect of mask wear in causing a shift in the propagation direction of the profiles does not occur in PMMA because the profile bottoms remain flat, as opposed to curved as in glass. However, the effect of mask wear in increasing the mask width size did result in a discrepancy between the measured and modelled profile width in Figure 3.8.

The present model was only implemented for one representative inclination angle. A change in the incident angle would likely change the inclination of the micro-channels; however, experiments were performed at only 45°, which represents a realistic balance between feature width and inclination angle.

Moreover, for impact angles other than 45°, the general shape of the PMMA channels should not significantly change, i.e. they will always have straight walls and rectangular bottoms, regardless of angle of incidence, as also was shown in Chapter 2 at normal incidence, e.g. as seen in Figure 2.12. On the other hand, the shape of glass would most likely change with the angle. However, it would not be practical to machine brittle masked substrates at angles < 45°, because, for brittle targets, the erosion rate rapidly decreases with decreasing incident angle (Section 2.2), thus requiring much longer machining times which also increases the mask wear. In the more practical range of 45° - 90°, the shape would still have curved walls and round bottoms, approaching symmetry as normal incidence is approached.

It was demonstrated in Chapter 2 that LSM can fairly accurately predict the surface evolution of micro-channels machined in glass at 90° up to AR = 1, e.g. as seen in Figure 2.11. Beyond that, the modelled predicted profiles begin to deviate from experimental ones due to particle second strike and spatial hindering effects [13,21-24] described in Section 1.2.1. For the present case in Figure 3.7 (α = 45°), the effective AR, the ratio of maximum feature depth to width at zero depth multiplied by $1/(\sin\alpha)^2$, approaches 1 after just 6 passes, which is much more rapidly than at 90°. Hence these effects, and consequently deviations in the model predictions, are expected to be more significant in the case of oblique blasting. In addition, the effect of mask wear is likely more significant at oblique incidence since masks are normally made from ductile materials that have a peak erosion rate at oblique incidence, as in the present case. Hence, for the model to accurately predict the experimental profiles in glass, the effects of mask wear and second strike must be included. Thus, the current formulation is suitable only in cases where these effects are minimal, and so is generally not applicable for predictions of the AJM in brittle substrates.

In the next chapter, the formulation will be extended to include mask erosive wear.

Chapter 4 Level Set Methodology for Predicting the Effect of Mask Wear on Surface Evolution of Features in Abrasive Jet Micromachining

4.1 Motivation

As shown in Chapters 2 and 3, mask wear effects affect the surface evolution of target features during AJM, e.g. such as can be seen in Figures 2.15, 3.7 and 3.9. In many cases mask wear cannot be avoided, i.e. when (thin) polymeric masks are used or when large particle doses, and hence long machining times, are necessary to create high aspect ratio features.

In spite of its significant effect on the resolution of features machined using AJM, the modelling of mask wear has thus far been limited to only two previous models: Slikkerveer at al. [29], described in Section 1.2.1, and Yagyu and Tabata [32], described in Section 1.2.2. However, in both cases, the work did not make direct comparisons between the model predictions and experiments on both the target and the mask together; thus only qualitative conclusions in terms of influence of mask wear on surface evolution of the target could be drawn. In addition, these models did not consider the influence of mask wear on features machined at oblique incidence such as those presented in Chapter 3.

This chapter extends the NB LSM-based methodology presented in Chapter 3 to allow the surface evolution of both the mask and target to be predicted simultaneously, by representing them as a hybrid, yet continuous, mask-target surface. The general methodology is based on a previous level set approach developed for ion beam milling a masked substrate at normal incidence [49]. The extension of this methodology to make it suitable for the modelling of mask wear in AJM presented unique challenges because, in contrast to ion beam milling, it considers finite size particles, and thus requires an estimate of the change in abrasive mass flux incident to the target through the mask opening, such as in Sections 2.3.3 and 3.3.2, and, for the first time, onto the eroding mask edge itself. The modelling of oblique incidence also presents additional complications that have not been previously considered. The predicted channel and eroded mask shapes were directly compared against measurements on channels machined in both glass and PMMA targets, using two different masks, thus verifying the predictive capability of the methodology. The majority of the material in this chapter has been submitted for publication and is currently under review [50].

4.2 AJM experiments: Channels and eroding masks

All AJM experiments were conducted using the same channel blasting apparatus and similar experimental conditions as those described in Section 2.2. Some differences, especially with respect to the masking technique, and extra details, are outlined below.

Two mask materials were used for the glass and PMMA targets: tempered steel feeler gauge (FG) stock (Starret Co., Athol, MA, USA) and High Tack RapidMask^{TM} (RM) (IKONICS Imaging, Duluth, MN, USA). The densities of FG and RM were ρ = 7712 kg m^{-3} and ρ = 1292 kg m^{-3}, respectively, obtained from unpublished experimental results (UR) from the author's laboratory (see Acknowledgements). The initial hardness of the FG stock was measured as Hv = 5.67 GPa using a micro hardness tester, and a value of Hv = 0.1 GPa was assumed for RM (UR). Both mask materials exhibited ductile erosive properties, as in the case of PMMA, although RM is an elastomer.

The FG masks were made by securely clamping two gauges to the target a specified distance apart, parallel to each other. The RM masks were made by placing a patterned photomask (Fine Line Imaging, Colorado Springs, CO, USA) over an RM sheet, and exposing it to a fused quartz silicon dioxide curing lamp (5.04 W/mm) for 40 passes at 76 mm/s using a mini conveyor UV cure system (American Ultraviolet Company, Lebabon, IN, USA). A squeegee was used to ensure that no air bubbles formed around the exposed feature, which acquired brittle properties after exposure. The resulting pre-machined FG and RM masks had a height, H_m, of approximately 100 μm and opening widths, W_m of approximately 180 μm and 130 μm, respectively. Figure 4.1 shows a schematic of the mask arrangement.

The glass masked channels were machined at impact angles, α, of 90° and 45°, and the PMMA masked channels at impact angles of 90°, as shown in Figures 2.1 and 3.1. Scanning speeds of 1 mm s^{-1} and 2 mm s^{-1} were used in machining the channels with FG and RM masks, respectively. The measured abrasive mass flow rate was in the range of 2.24-3.84 g min^{-1} (see Appendix A, Tables A-11-A-16), which again was low enough for particle interference to be neglected, as in Section 2.2 [33-35].

Cross-sectional target channel profiles with the masks attached were measured before and after machining using a non-contact optical profilometer described in Section 2.2. After machining, the masks were removed, the targets cleaned with alcohol, and the cross-sectional profiles of the target channels were measured. Approximately between 33-456 and 26-111 data points were obtained over scanning width ranges of 0.33-3.30 mm and 0.23-0.62 mm for all the mask profiles and channel profiles machined at α = 90°, respectively. As in Section 3.2, the inclined channel profiles were multi-valued, and could not be scanned using the optical profilometer. Instead, they were sectioned, and their cross-sectional profiles were measured and digitized using a 1.3 megapixel CMOS digital camera mounted to a 50X magnification optical microscope (ViewMet, Buehler Ltd., Lake Bluff, IL, USA), together with ImageJ software. Between 21-80 data points were obtained for all the resulting inclined channel profiles. A FG mask edge radius, r_m, depicted in Figure 4.6 and given in Table 4.1 (see below), was measured before machining using the same hardware as that used in obtaining the cross-sectional profiles.

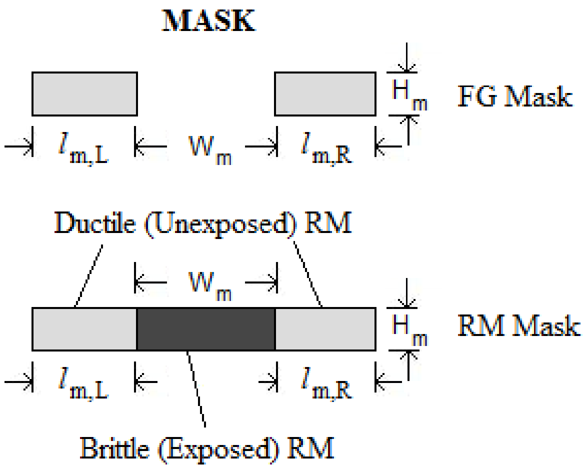


Figure 4.1. Front view schematic of masks used in the AJM of the micro-channels.

4.3 Level set modelling of surface evolution in AJM including mask erosive wear

4.3.1 Local normal velocity function of evolving surface for masked channels including mask erosive wear

In order to model masked channels including mask erosive wear, as shown in Figure 4.2, the same transformed coordinate system defined in eq. (2.1) from Section 2.3.1 and shown Figure 3.1 was used, but with $x_{\text{off}} = W_m/2 + l_{m,L}$, where $l_{m,L}$ is the pre-machined length of the left hand side of the mask shown in Figure 4.1. The velocity of the surface in the direction of the local normal for the AJM of brittle materials (glass) and ductile materials (PMMA, FG and RM) is defined by eqs. (3.1) and (3.2), respectively, with the particle velocity, V, and particle flux, ϕ, distributions defined as previously in eqs. (2.7) and (2.9), since similar blasting conditions were used as those in Sections 2.2 and 3.2.

As explained in Sections 2.4.1.1 and 3.3.1, eqs. (2.7) and (2.9) can be used to express the erosive efficacy of the jet, $E_{\text{ef}} = CV^{k_v}\phi$, for the AJM of channels in brittle targets, but for ductile materials in general, a stationary approach that approximates E_{ef} of a moving target/mask by a stationary one must be used. Since for the present problem, the hybrid surface consists of both the target (glass, brittle or PMMA, ductile) and the mask (FG or RM, ductile), for consistency and without loss of accuracy, in the present formulation the stationary target/mask approach was also used for the brittle glass targets.

In general, to use the stationary approach for a channel, the time dependencies of eqs. (2.7) and (2.9), represented by the $(r_s - v_t t)$ terms, are removed by calculating a single y location along the scanning direction where the erosive efficacy distribution seen by a cross-section of a hole, $E_{\text{ef,st}}$, is closest to the erosive efficacy distribution seen by a cross-section of a channel, $E_{\text{ef,t}}$, over a full pass. For instance, in Chapters 2 and 3, $y = 0$ was used in place of $y = (r_s - v_t t)$, as explained in Sections 2.4.1.1 and 3.3.1. However, in the present case, a slightly different approach was taken. In a single pass of the nozzle, the average erosive efficacy for a scanning target with the mask, $\overline{E}_{\text{ef,t}}$, is obtained as

$$\overline{E}_{\text{ef,t}} = \frac{v_t}{2r_s} \int_0^{2r_s/v_t} CV^{k_v}\phi\, dt \tag{4.1}$$

where V and ϕ are defined by eqs. (2.7) and (2.9), respectively, with the time dependencies in place. The equivalent stationary erosive efficacy was then obtained by calculating the location $\bar{y} = p_{rs} \cdot r_s$ where the $E_{\text{ef,st}}$ best fit eq. (4.1) over the full range of x' and z', i.e. p_{rs} (i.e. y) was varied until the mean ratio of $\overline{E}_{\text{ef,t}} / E_{\text{ef,st}}(y)$ over the full range of x' and z' was equal to 1. $\bar{y}$ was then used in place of $y = (r_s - v_t t)$ in eqs. (2.7) and (2.9) to represent the equivalent stationary erosive efficacy, $E_{\mathrm{ef,st}}(\bar{y})$. This approach better approximates the mean $E_{\mathrm{ef,t}}$ for the present formulation than the stationary approach used in Chapters 2 and 3. $p_{r\mathrm{s}}$ can vary with v_t, k_v, α and r_s (see Table 4.1).

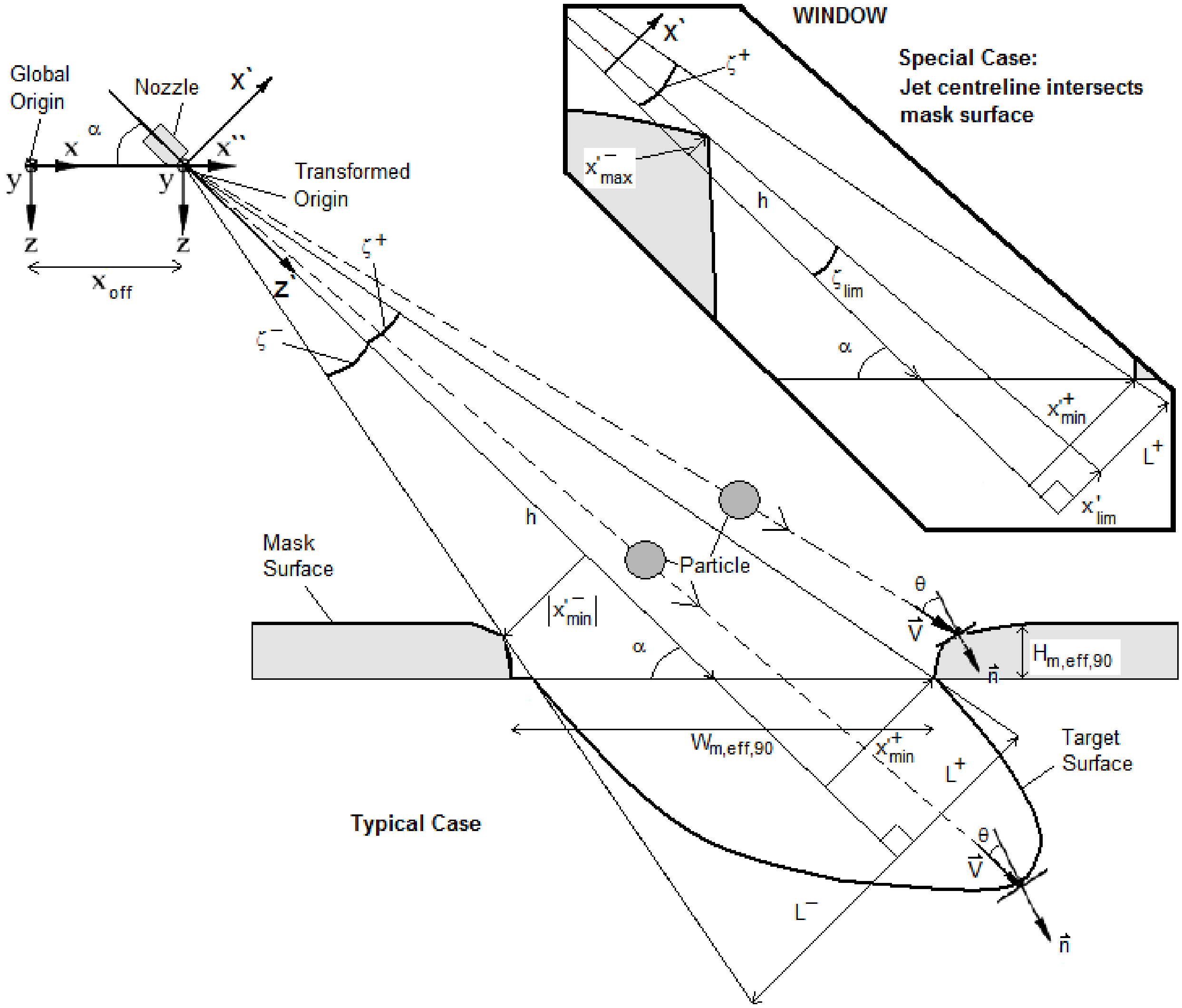


Figure 4.2. Geometry for modelling of flux adjustment for the target near the eroding mask edges at any incidence angle, α. The window shows the special case where the jet centreline intersects the eroding mask surface. The effective vertical height and horizontal opening width of the eroding mask, $H_{\mathrm{m,eff,90}}$, and $W_{\mathrm{m,eff,90}}$, are exaggerated with respect to h.

4.3.2 Masking function: Adjustment of particle mass flux to the target and the mask

In Sections 2.3.3 and 3.3.2, the reduction in particle mass flux incident to the target surface in the vicinity of the mask edges was modelled. In the present approach, since the surface is composed of both the target and the mask, the formulation is generalized by introducing the concept of a masking function, i.e. the adjustment to the incoming particle mass flux incident to the hybrid mask-target surface at a given *x'* to reflect the range of particle sizes that are 'visible' to this surface. The following sections derive this function.

4.3.2.1 Adjustment of mass flux to the target

In Sections 2.3.3 and 3.3.2, the effect of reduction in mass flux through the mask opening and incident to the target surface near the mask edges was modelled for non-eroding masks. In contrast to the previous approach, the present involves an eroding mask, so that the mass flux incident to the target surface adjacent to the edge changes with time and must thus be updated at each time step. The following formulation generalizes the approach presented in Section 3.3.2 to eroding masks.

At any instant, the masking function for the target surface, $M_{x/0}\big|_{\mathrm{T}}$, defined as the ratio of the mass of particles, *M*, that can reach the target surface at *x'* to the *M* that can reach the target surface at *x'* = 0, across the eroding mask opening, is

$$M_{x/0}\Big|_{\mathrm{T}} = \left.\frac{M(x')}{M(x'=0)}\right|_{\mathrm{T}} = \frac{\int_0^{L-|x'|} r_{\mathrm{p}}^3 \Omega(r_{\mathrm{p}})\, dr_{\mathrm{p}}}{\int_0^{L} r_{\mathrm{p}}^3 \Omega(r_{\mathrm{p}})\, dr_{\mathrm{p}}} \tag{4.2}$$

where eq. (4.2) is the same as eq. (3.3), and is repeated here to maintain continuity, and the subscript 'T' indicates 'target surface' (Figure 4.2). The same lognormal particle size distribution was assumed in eq. (4.2) as in eq. (2.11) since the same abrasives were used as in Sections 2.2. Note that eq. (4.2) can also be interpreted as the ratio of the expected value of the particle volume incident to the target at a given *x'* to the expected value of the particle volume at *x'* = 0, where particles of any size can reach the target. Following Section 3.3.2, *L* represents the target location that an infinitely small particle can reach without undergoing collision with the right (+) or left (-) edge of the eroding mask, measured along the *x'* direction at a given *z'* (Figure 4.2),

$$L = \begin{cases} L^+ = & z'\tan(\zeta^+) \quad (x' \geq 0) \\ L^- = & z'\tan(\zeta^-) \quad (x' < 0) \end{cases} \tag{4.3}$$

ζ^+ and ζ^- are the angles defining the maximum particle trajectories incident to the target surface through the eroding mask opening, measured from the jet centreline to the right (+) and left (-) mask edges. For the typical case presented in Figure 4.2 and at normal incidence, the angles are calculated at each time step by searching for the minimum point *x'* to the right (+)/left (-) mask surface, $x'^{+/-}_{\min}$, measured from the jet centreline,

$$\zeta^{+/-} = \tan^{-1}\left(\frac{\left|x'^{+/-}_{\min}\right|}{z'\big|_{\left|x'^{+/-}_{\min}\right|}}\right) \tag{4.4}$$

where $z'\big|_{\left|x'^{+/-}_{\min}\right|}$ is the *z'* that corresponds to $\left|x'^{+/-}_{\min}\right|$. The analysis assumes that the jet centre coincides with the centre of the pre-machined mask opening. For the special case presented in the window of Figure 4.2, when the jet centreline intersects the mask surface on the left hand side, eq. (4.3) must be modified as follows

$$L = \begin{cases} L^+ = & \begin{cases} z'\tan(\zeta^+) - x'_{\text{lim}} & (x' \geq x'_{\text{lim}}) \\ 0 & (x' < x'_{\text{lim}}) \end{cases} \\ L^- = & 0 \end{cases} \tag{4.5}$$

where ζ^+ is obtained using eq. (4.4) and x'_{lim} is defined in the window of Figure 4.2,

$$x'_{\text{lim}} = z'\tan(\zeta_{\text{lim}}) \tag{4.6}$$

The angle ζ_{lim} is calculated by searching for the maximum *x'* to the mask surface on the left hand side, $x'^{-}_{\max}$ defined in the window of Figure 4.2, measured from the jet centreline,

$$\zeta_{\text{lim}} = \tan^{-1}\left(\frac{x'^{-}_{\max}}{z'\big|_{x'^{-}_{\max}}}\right) \tag{4.7}$$

where $z'\big|_{x'^{-}_{\max}}$ is the *z'* that corresponds to $x'^{-}_{\max}$. It should be noted that when $|x'| \geq L$ in eq. (4.2), $M_{x/0}\big|_{\text{T}} = 0$, as expected.

4.3.2.2 Adjustment of mass flux to the mask edges

In addition to the adjustment to particle mass flux ‘visible’ to the target surface near the mask edges, the finite size of the particles also limits the flux ‘seen’ by the mask edges themselves. For example, the particle shown in Figure 4.3 (a) is too large to strike the inclined edge of the mask, and would instead strike the top of the mask near x'^{+}_{tran}. Similarly, only small particles can strike the mask edge at its bottom near the surface. This effect has never been previously modelled.

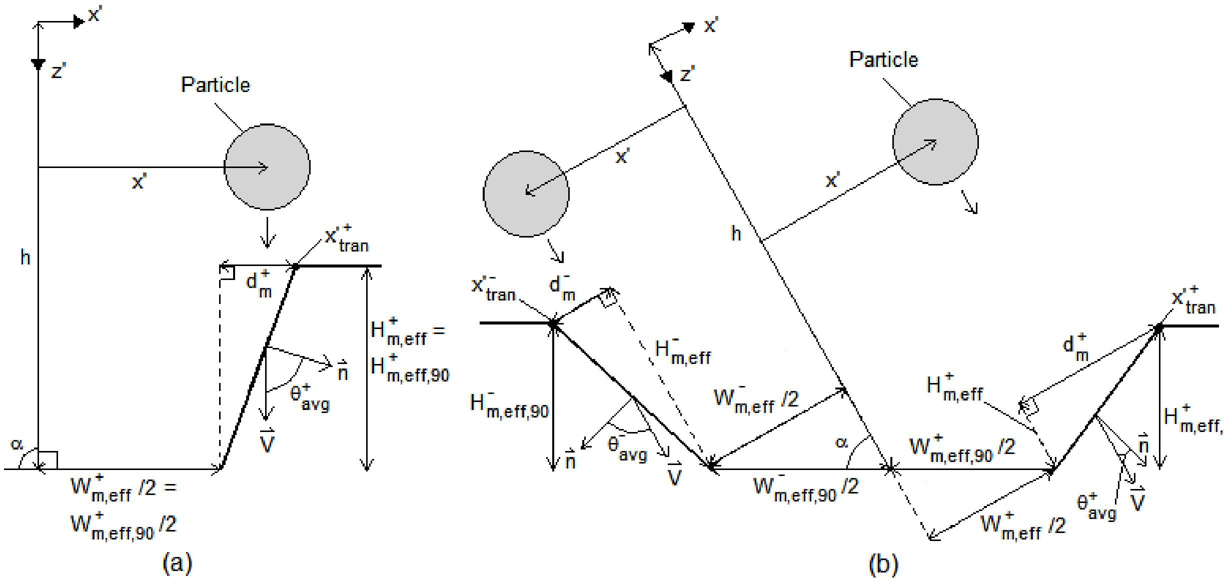


Figure 4.3. Modelling of adjustment to mass flux striking the mask edges for (a) $\alpha = 90°$ and (b) $\alpha \leq 90°$, i.e. the general case. Due to symmetry, only the right mask edge is shown in (a). The effective eroding mask heights and opening widths are exaggerated with respect to *h*.

Modifying eq. (4.2), the masking function for the mask edge surface, $M_{x/0}\big|_{\text{M,edge}}$, defined as the ratio of the portion of incident *M* that can arrive at *x'* and strike the mask edge surface, to the *M* that can arrive at *x'* = 0, across the eroding mask opening and edge, can be expressed as

$$M_{x/0}\Big|_{\text{M,edge}} = \left.\frac{M(x')}{M(x'=0)}\right|_{\text{M,edge}} = \frac{\displaystyle\int_{\left(\frac{W_{\text{m,eff}}}{2}-|x'|\right)\cdot(1-\Psi)}^{\frac{W_{\text{m,eff}}}{2}+d_{\text{m}}-|x'|} r_{\text{p}}^{3}\Omega(r_{\text{p}})dr_{\text{p}}}{\displaystyle\int_{0}^{\frac{W_{\text{m,eff}}}{2}+d_{\text{m}}} r_{\text{p}}^{3}\Omega(r_{\text{p}})dr_{\text{p}}} \tag{4.8}$$

where the subscript ‘M’ indicates ‘mask surface’ (Figure 4.2). $W_{\text{m,eff}}/2$ is the effective half opening width of the eroding mask measured along x', and d_{m} is the approximated length of the eroding mask edge measured along x' (Figure 4.3). The Heaviside function, $\Psi = \Psi(|x'| - W_{\text{m,eff}}/2)$ is defined as $\Psi = 0$ if $|x'| < W_{\text{m,eff}}/2$ and $\Psi = 1$ if $|x'| \geq W_{\text{m,eff}}/2$. It should be noted that when $|x'| \geq W_{\text{m,eff}}/2 + d_{\text{m}}$ in eq. (4.8), $M_{x/0}\big|_{\text{M,edge}} = 0$, as expected.

The LSM model allows for the mask edge to evolve a curved profile, i.e. with a non-uniform θ (defined in Figures 2.2 and 4.2). As a first-order approximation, for the purposes of determining $W_{\text{m,eff}}/2$ and d_{m} at any given time step, it was nevertheless assumed that the eroded mask edge could be represented by a single line (Figure 4.3) having a slope determined by the average of all the local θ over the mask edge, θ_{avg}. With this approximation, for the typical case (Figure 4.2) for the right (+) or left (-) mask,

$$W_{\text{m,eff}}/2 = \begin{cases} W^{+}_{\text{m,eff}}/2 = \quad x'^{+}_{\text{min}} & (x' \geq 0) \\ W^{-}_{\text{m,eff}}/2 = \quad \left|x'^{-}_{\text{min}}\right| & (x' < 0) \end{cases} \tag{4.9}$$

and for the special case depicted in the window of Figure 4.2,

$$W_{\text{m,eff}}/2 = \begin{cases} W^{+}_{\text{m,eff}}/2 = & \begin{cases} x'^{+}_{\text{min}} - x'_{\text{lim}} & (x' \geq x'_{\text{lim}}) \\ 0 & (x' < x'_{\text{lim}}) \end{cases} \\ W^{-}_{\text{m,eff}}/2 = & 0 \end{cases} \tag{4.10}$$

where $x'^{+/-}_{\text{min}}$ and x'_{lim} were defined in Section 4.3.2.1 and Figure 4.2. The distance d_{m} in Figure 4.3 is approximated as

$$d_{\mathrm{m}} = \begin{cases} d_{\mathrm{m}}^{+} = \quad H_{\mathrm{m,eff}}^{+} \cot(\theta_{\mathrm{avg}}^{+}) & (x' \geq 0) \\ d_{\mathrm{m}}^{-} = \begin{cases} 0 & \theta_{\mathrm{avg}}^{-} \geq \pi/2 \\ H_{\mathrm{m,eff}}^{-} \cot(\theta_{\mathrm{avg}}^{-}) & \theta_{\mathrm{avg}}^{-} < \pi/2 \end{cases} & (x' < 0) \end{cases} \tag{4.11}$$

where $H_{\mathrm{m,eff}}^{+}$ and $H_{\mathrm{m,eff}}^{-}$ are the effective heights of the eroding right and left masks, respectively, measured parallel to the jet centreline (Figure 4.3),

$$H_{\mathrm{m,eff}}^{+/-} = \frac{H_{\mathrm{m,eff,90}}^{+/-} \sin(\theta_{\mathrm{avg}}^{+/-})}{\sin(\theta_{\mathrm{avg}}^{+/-} \pm \pi/2 \mp \alpha)} \tag{4.12}$$

The value of $\left|x'^{-}_{\mathrm{min}}\right|$ in eq. (4.9) only needs to be evaluated when d_{m} is non-zero. For a FG mask, and for a RM mask after the 'brittle' RM (BRM) layer (Figure 4.1) has been fully etched through to the target surface, $H_{\mathrm{m,eff,90}}^{+/-}$ (Figures 4.2 and 4.3) is calculated as the distance from the transition point between the eroding mask edge and the top of the mask, $x'^{+/-}_{\mathrm{tran}}$ (Figure 4.3), to the unmachined target surface. For a RM mask before the BRM layer has been fully etched through, $H_{\mathrm{m,eff,90}}^{+/-}$ is calculated numerically from $x'^{+/-}_{\mathrm{tran}}$ to the assumed flat eroding bottom of the BRM surface (see Section 4.3.2.3 and bottom of Figure 4.4). x'_{tran} is obtained by searching the mask surface and locating a point of sharp transition in local θ.

In the analysis, particles incident to $|x'| < W_{\mathrm{m,eff}}/2$ were considered incident to the target surface, whereas ones incident to $|x'| \geq W_{\mathrm{m,eff}}/2$ were considered incident to the eroding mask edge, thus ignoring unlikely glancing collisions. In addition, the eroded mask edge was represented by a line having a single average slope, θ_{avg}. As a result, the overall adjustment to the flux at the sloped mask edge was modelled as the mean of the masking function in eq. (4.8) over the range of $W_{\mathrm{m,eff}}/2 \leq |x'| \leq W_{\mathrm{m,eff}}/2 + d_{\mathrm{m}}$,

$$\overline{M}_{x/0}\Big|_{\mathrm{M,edge}} = \frac{1}{d_{\mathrm{m}}} \int_{W_{\mathrm{m,eff}}/2}^{W_{\mathrm{m,eff}}/2 + d_{\mathrm{m}}} M_{x/0}\Big|_{\mathrm{M,edge}} \, dx' \tag{4.13}$$

Since the distance d_{m} is generally small, the error introduced by this averaging is also small.

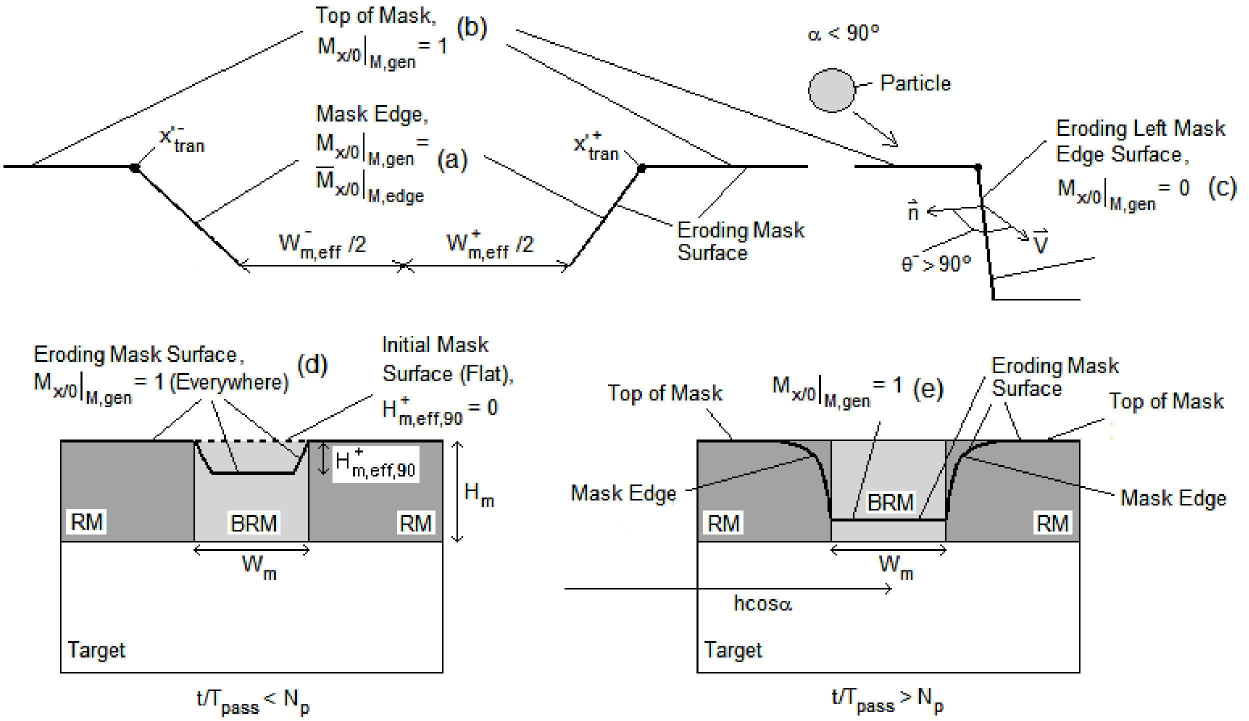
4.3.2.3 General masking function for the entire mask

The general masking function for the entire mask surface, $M_{x/0}\big|_{\mathrm{M,gen}}$, defined within the global range of $h\sin\alpha - H_{\mathrm{m}} \le z < h\sin\alpha$ (Figure 4.2), where H_{m} is the pre-machined mask height, and the full horizontal length of the computational space is

$$M_{x/0}\big|_{\mathrm{M,gen}} = \begin{cases} \overline{M}_{x/0}\big|_{\mathrm{M,edge}} & (\mathrm{W_{m,eff}}/2 \le |x'| \le x'^{+/-}_{\mathrm{tran}}) & (\text{eq. (4.13) for Mask Edges}) & \text{(a)} \\ 1 & (|x'| > x'^{+/-}_{\mathrm{tran}}) & (\text{Top of Mask}) & \text{(b)} \\ 0 & (\theta^{-} \ge \pi/2) & (\text{Only for } \alpha < \pi/2\text{, Left Mask Edge}) & \text{(c)} \\ \text{Special Cases:} & & & \\ 1 & (t/T_{\mathrm{pass}} < N_p) & (\text{Only for RM Mask}) & \text{(d)} \\ 1 & (h\cos\alpha - \mathrm{W_m}/2 \le x \le h\cos\alpha + \mathrm{W_m}/2) & (\text{Centre Mask Region}) & \text{(e)} \end{cases} \tag{4.14}$$

The various cases that must be considered in eq. (4.14) are shown in Figure 4.4. Equation (4.14 (a)) is defined in eq. (4.13), and eq. (4.14 (b)) describes the masking function for the top of the mask that was originally horizontal. Equation (4.14 (c)) reflects the fact that, at oblique incidence, any point on the left mask edge surface does not see the incoming particle flux until its local slope is at least parallel with the incident velocity vector.

In the case of the RM, the entire initial surface is flat, and the region of width W_{m} exposed to the UV light becomes brittle (BRM). Thus, the initial surface is composed of two different eroding materials: RM and BRM (see Section 4.3.3.1). The use of a hybrid surface with continuous connectivity can introduce complications at the points of intersection of the surfaces; in this case, the points connecting RM and BRM. This results solely from the introduction of a masking function for the mask edges. If the conditions of eq. (4.14 (a)) were assumed from the beginning, although the BRM surface would propagate downwards towards the target relatively quickly, the lateral propagation of the eroding RM mask edge surfaces would be unrealistically slowed down due to the low initial $H^{+/-}_{\mathrm{m,eff,90}}$ in eqs. (4.11) and (4.12), and hence $\overline{M}_{x/0}\big|_{\mathrm{M,edge}} \approx 0$. To overcome this difficulty, $M_{x/0}\big|_{\mathrm{M,gen}} = 1$ was assumed in eq. (4.14 (d)), for both the BRM and RM, until the RM began to develop an inclined mask edge, after N_p passes of the nozzle. After N_p passes, the eroded profile was partially in the RM and the BRM, and eq. (4.14 (a)) was assumed on the inclined RM surface, and $M_{x/0}\big|_{\mathrm{M,gen}} = 1$ was assumed in eq. (4.14 (e)) for the remaining BRM. As a result, for the RM-BRM profile surface, the mask edge effect was partially accounted for until the target region was reached. N_p corresponds to the experimental initial pass profile used to determine the time, T_{pass}, that it takes for the surface to propagate to a depth defined by the first pass profile (see Section 4.4.1).



Top of Mask,
$M_{x/0}|_{\text{M,gen}} = 1$
(b)
$\alpha < 90°$
Particle
x'^{-}_{tran}
Mask Edge,
$M_{x/0}|_{\text{M,gen}} = \overline{M}_{x/0}|_{\text{M,edge}}$
(a)
x'^{+}_{tran}
Eroding Left Mask Edge Surface,
$M_{x/0}|_{\text{M,gen}} = 0$
(c)
$\vec{n}$
$\vec{V}$
$W^{-}_{\text{m,eff}}/2$
$W^{+}_{\text{m,eff}}/2$
Eroding Mask Surface
$\theta^{-} > 90°$
Eroding Mask Surface,
$M_{x/0}|_{\text{M,gen}} = 1$ (Everywhere)
(d)
Initial Mask Surface (Flat),
$H^{+}_{\text{m,eff,90}} = 0$
$H^{+}_{\text{m,eff,90}}$
H_m
RM
BRM
RM
W_m
Target
$t/T_{\text{pass}} < N_p$
Top of Mask
$M_{x/0}|_{\text{M,gen}} = 1$
(e)
Eroding Mask Surface
Top of Mask
Mask Edge
Mask Edge
RM
BRM
RM
W_m
hcosα
Target
$t/T_{\text{pass}} > N_p$

Figure 4.4. Schematic representation of cases corresponding to eq. (4.14). Top left: eq. (4.14 (a)) and eq. (4.14 (b)); top right: eq. (4.14 (c)); bottom left: eq. (4.14 (d)); and bottom right: eq. (4.14 (e)). The rectangular regions in the bottom two schematics differentiate the different materials in the numerical grid.

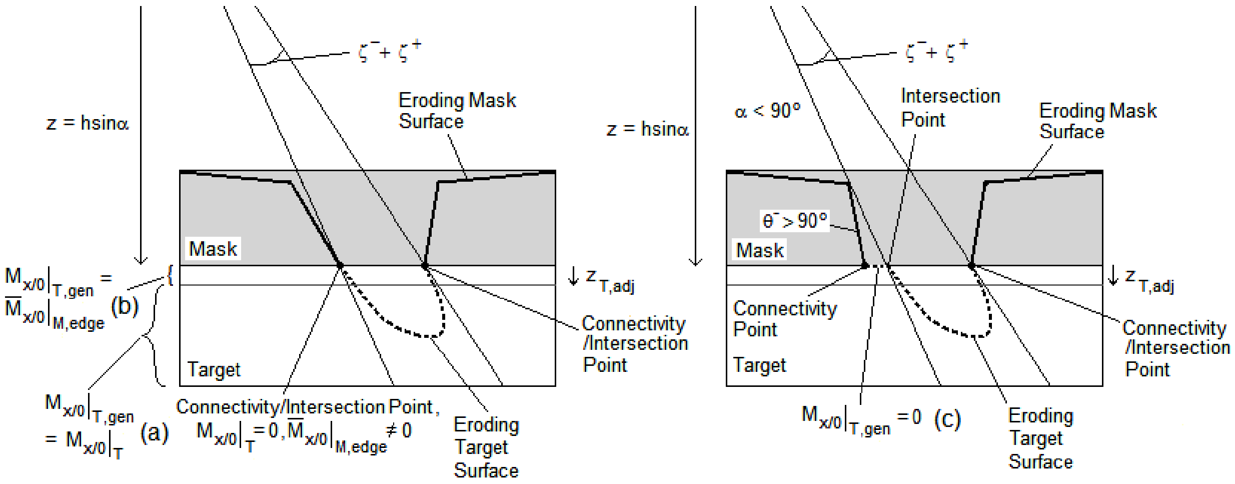
4.3.2.4 General masking function for the target

Equation (4.2) is used to model the adjustment to the mass flux incident to the target surface near the mask edges, and greatly improves the fits of the modelled target profiles to measured ones [19]. However, when the intersection points between the lines of the maximum particle trajectories incident to the target defined by ζ^- and ζ^+ and the horizontal line at $z = h\sin(\alpha)$ match the connectivity points between the target and the mask, the target masking function $M_{x/0}|_{\text{T}}$ = 0, while the mask edge masking function $\overline{M}_{x/0}|_{\text{M,edge}}$ is generally non-zero (Figure 4.5). If uncorrected, this causes an unrealistic slowdown in the surface propagation of the mask edges similar to that encountered for the RM-BRM surface in Section 4.3.2.3, however this time due to the target masking function. This problem was not encountered in previous LSM studies of e.g. ion milling when applied to source etching of a substrate and a mask [49], because the masking function in that case was 1 at such locations, since the particles were all considered infinitely small. Thus, in the present work, it was necessary to adopt a procedure whereby the target was allowed to erode to a depth $z_{\mathrm{T,adj}}$ under the same flux adjustment, i.e. masking function, as the mask shown on the left hand side of Figure 4.5. This allowed the target and mask edges to evolve together over the depth $z_{\mathrm{T,adj}}$. Beyond that depth, the target evolved under the usual flux adjustment according to eq. (4.2). Hence, the general masking function for the target surface, $M_{x/0}\big|_{\mathrm{T,gen}}$, defined globally for $z \geq h\sin\alpha$ (Figures 4.2 and 4.5) and over the full computational horizontal length, is

$$M_{x/0}\big|_{\mathrm{T,gen}} = \begin{cases} M_{x/0}\big|_{\mathrm{T}} & (z \geq h\sin\alpha + z_{\mathrm{T,adj}}, \text{All Possible } x) \qquad (\text{eq.}(4.2)) \qquad (\text{a}) \\ \text{Special Cases:} & \\ \overline{M}_{x/0}\big|_{\mathrm{M,edge}} & (h\sin\alpha \leq z < h\sin\alpha + z_{\mathrm{T,adj}}, \text{All Possible } x) \qquad (\text{eq.}(4.14(\text{a}))) \qquad (\text{b}) \\ 0 & (z = h\sin\alpha, x \text{ over Left Mask Shadow Length}) \qquad (\text{c}) \end{cases} \tag{4.15}$$

The value of $z_{\mathrm{T,adj}}$ was chosen through numerical experimentation to be as large as possible so as not to affect the propagation of the mask edges, but at the same time small enough to not significantly affect the shape of the target profile within $h\sin\alpha \leq z < h\sin\alpha + z_{\mathrm{T,adj}}$. To do this, the simulation was successively run to different $z_{\mathrm{T,adj}}$ depths with no flux adjustment to the target, i.e. infinitely small particles impacting the target with $M_{x/0}\big|_{\mathrm{T,gen}} = 1$. $z_{\mathrm{T,adj}}$ was decreased until a transition between rapid mask propagation and slow mask propagation was noted, i.e. such that there was an insignificant difference in mask propagation between the chosen $z_{\mathrm{T,adj}}$ and a large $z_{\mathrm{T,adj}}$. In general, $z_{\mathrm{T,adj}}$ was found to be in the range of 2 to 3 grid steps (see Table 4.1). Since the resulting $z_{\mathrm{T,adj}}$ was generally only on the order of 10 - 20 μm (see Table 4.1), which is less than the maximum depth of the resulting target profile (see Section 4.4.1), the resulting error in the target profile evolution was relatively small.

For the special case when $\alpha < 90°$ and $\theta^{-} \geq 90°$ shown on the right hand side of Figure 4.5, a portion of the target to the right of the connectivity point is in the shadow of the left hand side of the mask, and $M_{x/0}\big|_{\mathrm{T,gen}} = 0$ was assumed in eq. (4.15 (c)) over this shadow length.



ζ⁻ + ζ⁺
Eroding Mask Surface
z = hsinα
Mask
Target
$M_{x/0}|_{T,gen} = M_{x/0}|_{M,edge}$ (b)
$M_{x/0}|_{T,gen} = M_{x/0}|_T$ (a)
Connectivity/Intersection Point, $M_{x/0}|_T = 0, M_{x/0}|_{M,edge} \neq 0$
Eroding Target Surface
$z_{T,adj}$
Connectivity /Intersection Point
ζ⁻ + ζ⁺
α < 90°
Intersection Point
Eroding Mask Surface
z = hsinα
θ⁻ > 90°
Mask
Connectivity Point
Target
$M_{x/0}|_{T,gen} = 0$ (c)
Eroding Target Surface
$z_{T,adj}$
Connectivity /Intersection Point

Figure 4.5. Schematic representation of cases corresponding to eq. (4.15). Left: eq. (4.15 (a)) and eq. (4.15 (b)); and right: eq. (4.15 (c)). The rectangular regions differentiate the different materials in the numerical grid.

4.3.2.5 Unified masking function

The unified masking function, $M_{x/0}\big|_{\text{Unif}}$, for the hybrid surface consisting of both the mask and the target surface, can be summarized as

$$M_{x/0}\big|_{\text{Unif}} = \begin{cases} M_{x/0}\big|_{\text{M,gen}} & (\text{eq. (4.14) for Mask Surface}) \quad (a) \\ M_{x/0}\big|_{\text{T,gen}} & (\text{eq. (4.15) for Target Surface}) \quad (b) \end{cases} \qquad (4.16)$$

where $M_{x/0}\big|_{\text{M,gen}}$ and $M_{x/0}\big|_{\text{T,gen}}$ are the general masking functions for the entire mask and target surfaces, respectively. The effective flux for the entire hybrid mask-target surface at each time step can be obtained by multiplying the modified eq. (2.9) (Section 4.3.1) by eq. (4.16).

4.3.3 LSM model implementation

4.3.3.1 Partial derivatives, surface initialization and geometric variables

Following the approach of Sections 2.3.4.1 and 3.3.3.1, eq. (2.12) was used to approximate partial derivatives used in obtaining the solution to $\Phi(x,z,t)$ and geometric variables. The initial hybrid surface was represented either by a horizontal line consisting of the RM and 'brittle' RM (BRM) mask surfaces at the same height, or, in the case of the FG, an inverse 'hat' shape, consisting of the FG mask and the target which is at a lower height over the mask opening width, as shown in Figure 4.6. Following the approach of [49], the grid was divided into regions, each of which represents a different material. For simulations involving a FG mask, there were 2 regions consisting of the FG and the target. For simulations involving a RM mask, there were 3 regions consisting of the RM, BRM, and the target (Figure 4.6). For this case, the hybrid surface consisted of the RM mask and the target once the BRM layer was etched through to the target. The $\Phi(x, z, t)$ was initialized and re-initialized using the SDF defined by eq. (2.29) as explained in Section 3.3.3.1. The geometric variables $\vec{n}$ and K were obtained using eq. (3.13).

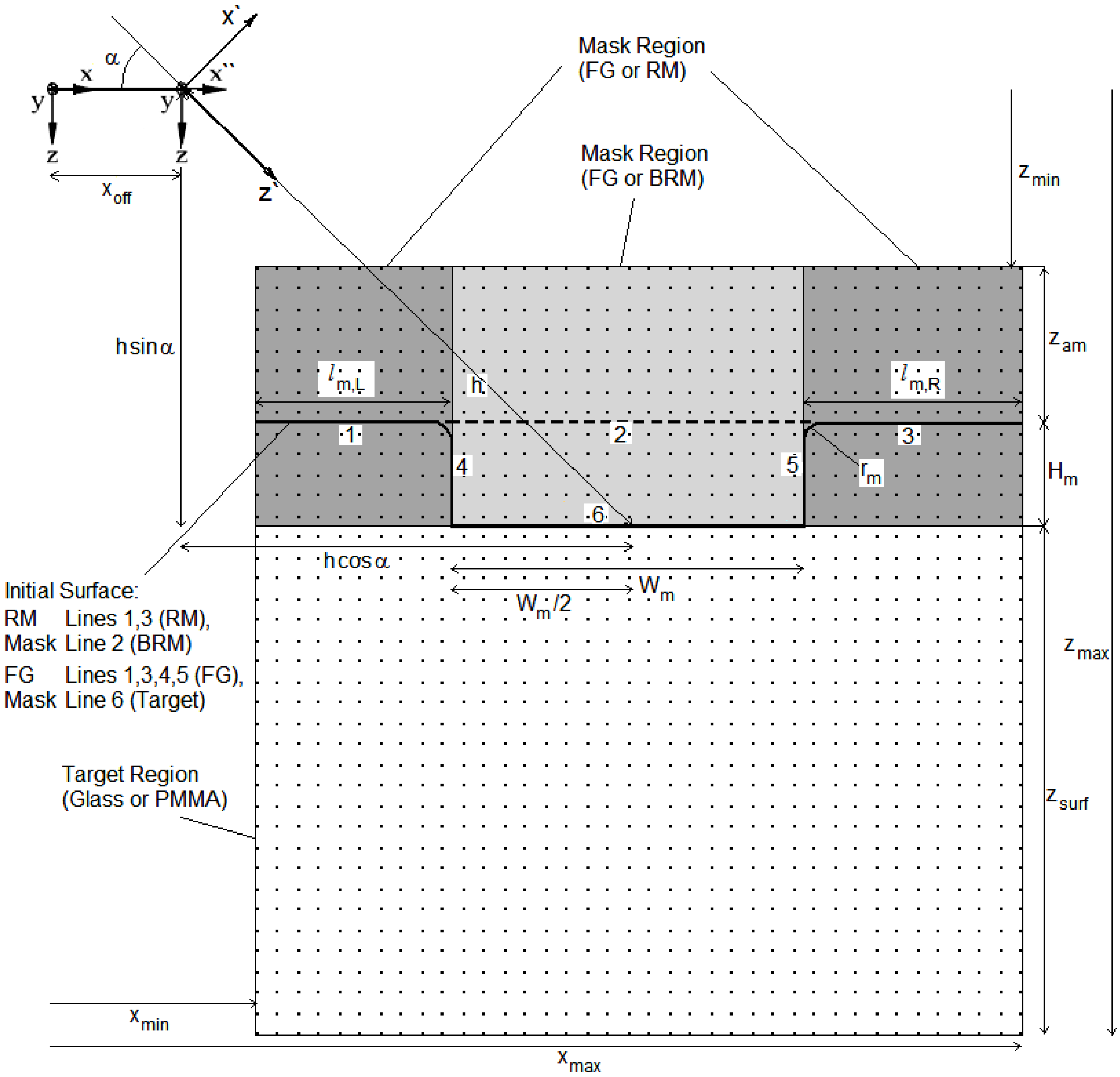


Figure 4.6. Initial surface and grid formulation. The grid is divided into regions, each of which represents a different material. The grid spacing and mask dimensions are exaggerated with respect to *h*.

4.3.3.2 Simplified NB LSM for non-convex Hamiltonians with EVM

The same simplified LSM for non-convex *H* presented in Section 3.3.3.2 is used in the present formulation, however, with two exceptions: 1) Equation (2.21) with $|\nabla\Phi(x,z)|=1$ (see eq. (3.12)) is used instead of eq. (3.14) since the present formulation considers motion with curvature, i.e. $\varepsilon \neq 0$; and 2)

$M_{x/0}\big|_{\text{Unif}}$ defined by eq. (4.16) is used in eqs. (3.16) (for glass) and (3.17) (for PMMA, FG and RM), instead of $M_{x/0}$ defined by eq. (3.3), since the present formulation considers eroding masks.

The same EVM presented in Section 3.3.3.3 is used in the present formulation but it should be added that the 'jet visibility' effect (Figure 3.4) which necessitates the application of EVM described in that section now occurs at any α, i.e. both oblique and normal incidence. This occurs since the present formulation includes the mask as part of the hybrid zero level set surface, which affects the visibility of the grid 'below' the mask at any incidence. In other words, the use of a mask (Section 4.3.2) introduces a boundary for the target for $z \geq h \sin\alpha$ formed by lines of the maximum particle trajectories incident to the target defined with ζ^- and ζ^+ in Figure 4.2, beyond which $F(x,z,t) = 0$ for the level sets below the mask to the left or right of the lines, respectively, at any incidence. Thus, EVM was used at any incidence, and for both the target and the mask to maintain consistency and numerical stability, i.e. by extending from the entire hybrid mask-target surface to all the material regions defined in Figure 4.6.

Finally, the same NB LSM presented in Section 3.3.3.4 is applied in the present formulation. The NB-extension algorithm was the same as outlined in Section 3.3.3.7.

4.3.3.3 Grid formulation, boundary conditions, time step and surface interpolation

For the present problem, using the geometry of Figure 4.6, the vertical grid limits were obtained as $z_{\min} = h\sin\alpha - (H_{\text{m}} + z_{\text{am}})$ and $z_{\max} = h\sin\alpha + z_{\text{surf}}$, where z_{surf} was defined in Section 2.3.4.3 and z_{am} is the vertical grid distance above the mask necessary to initiate $\Phi(x,z,t)$ with the SDF in eq. (2.29) and adjusted to maintain uniform grid spacing, i.e. $\Delta x = \Delta z$. The horizontal grid limits were obtained as $x_{\min} = x_{\text{off}} + h\cos\alpha - (W_{\text{m}}/2 + l_{\text{m,L}})$ and $x_{\max} = x_{\text{off}} + h\cos\alpha + (W_{\text{m}}/2 + l_{\text{m,R}})$, where $l_{\text{m,L}}$ and $l_{\text{m,R}}$ are the pre-machined lengths of the left and right hand sides of the mask, respectively, shown in Figures 4.1 and 4.6. These limits were then used to calculate the spatial grid steps and global spatial coordinates at the grid nodes as described in Section 2.3.4.3. The boundary conditions were obtained in the same way as described in Sections 2.3.4.3 and 3.3.3.5. The time step was once again calculated using the CFL condition [37] by combining eqs. (2.28) and (3.23) with a slight modification,

$$\Delta t \cdot \max\left[\sum_{g=x,z}\left(\frac{\left|\left(H_{\text{ext}}\right)_{\Phi_g}(\nabla\Phi^{\pm})\right|}{\Delta g} + \frac{2\varepsilon\left|\left(H_{\text{ext}}\right)_{\Phi_g}(\nabla\Phi^{\text{c}})\right|}{\Delta g^2}\right)\right] < 1 \tag{4.17}$$

where the maximum was obtained in the same way as in Section 3.3.3.5. Equation (4.17) differs from the approach originally suggested by Osher and Fedkiw [37], and which was adapted in Chapter 2 and eq. (2.28), only through the introduction of $\left|H_{\Phi_g}(\nabla\Phi^{\mathrm{c}})\right|$ in the second term, related to the motion due to curvature in eq. (2.21) with $|\nabla\Phi| = 1$ (Section 4.3.3.2). In Section 2.3.4.2 the $\varepsilon K^{\mathrm{c}}$ term multiplying $H^{\mathrm{c}} = F^{\mathrm{c}}\left|\nabla\Phi^{\mathrm{c}}\right|$ in eq. (2.21) was applied to the $\left|\nabla\Phi^{\mathrm{c}}\right|$ term, whereas now it is applied directly to the F^{c} term which more accurately reflects the motion due to curvature defined in eq. (2.21) with $\left|\nabla\Phi^{\mathrm{c}}\right| = 1$. This greatly reduces the time step restriction required in eq. (4.17) for cases where curvature is used, and hence increases the computational efficiency.

Finally, the location of the surface, used in obtaining $\Phi(x,z,t)$ with the SDF in eq. (2.29), in calculating $M_{x/0}\big|_{\mathrm{Unif}}$ defined in Sections 4.3.2.5 and in locating the upper and lower bands described in Section 3.3.3.4, as well as FD approximations of the partial derivatives at surface nodes (Figure 3.6 and eq. (3.24)) used in EVM described in Section 4.3.3.2, were obtained using interpolation as described in Section 3.3.3.6.

4.4 Results and discussion

4.4.1 Model execution and inputs/outputs

The model presented in Section 4.3 was implemented in MATLAB 7.9 (The MathWorks, Inc., Natick, MA, USA) and experimentally verified by comparing the LSM predicted surface evolution profiles to the measured ones in Figures 4.7-4.11. The required model inputs and resulting numerical outputs corresponding to simulations in Figures 4.7-4.11 are specified in Table 4.1. The parameters h, H_m, W_m, r_m, α, v_t, $\dot{M}$ and ρ were obtained based on experimental conditions of Sections 2.2 and 4.2 and are specified in Table 4.1. For BRM, ρ was assumed to be the same as for RM (UR). *Hv* for PMMA was obtained from Section 2.4.1.1 and for FG and RM from Section 4.2. Attempts to measure the hardness of the thin BRM sheet failed due to its tendency to fracture into small pieces upon contact with the indenter when using a micro hardness tester. Therefore, the *Hv* for this material was assumed to increase from the unexposed RM value of 0.1 (UR) to 0.25 GPa after exposure, reflecting the transition from ductile to brittle behaviour. The higher value for BRM was estimated based on the previously measured value for PMMA [2], which is harder than the unexposed RM (elastomer).

The parameters V_o, β, μ_l, σ_l, k_v, C, n_1 and n_2 specified in Table 4.1 were obtained from Section 2.4.1.1 or UR based on previous measurements for the same nozzle, jet conditions, abrasives, target and mask materials as presently utilized. Although the RM after exposure was labelled as 'brittle', the resulting experimental BRM profiles implied an intermediate brittle and ductile erosive response as shown in the Figure 4.11 measured profile after 2 passes since, as shown in Section 3.4.2, brittle target profiles generally have sloped walls and rounded bottoms, while ductile target profiles have straight walls and rectangular bottoms. Thus, the general erosion-angle of incidence model proposed by Oka et al. [40] and used in eq. (3.2) was assumed for BRM. The constant n_1 in eq. (3.2) for BRM can be calculated knowing $k_\mathrm{v} = 1.41$ (Table 4.1) from the relationship due to Oka et al., $n_1 = k_\mathrm{v} + 1$ [40]. The constant $n_2 = 9$ was obtained by numerically fitting the model-predicted to the measured sidewall slope of the BRM mask after the initial pass, in terms of side wall slope, analogous to the approach used to obtain T_pass (see below). Thus, the particular value of $Hv = 0.25$ GPa used for BRM, as explained in the previous paragraph, is not of great importance as long as it is of the correct order of magnitude, since it is a curve fitting parameter used along with n_1 and n_2 to obtain a best fit of the model-predicted to measured initial pass profile shape. Use of a different value would only yield a new n_2 without significantly affecting the resulting profile shape.

The parameters p_{rs}, N_p and $z_{\mathrm{T,adj}}$ specified in Table 4.1 were obtained as explained in Sections 4.3.1, 4.3.2.3 and 4.3.2.4, respectively. The parameter ε [16,17,37] was used to smooth the results in modelling glass targets and RM masks, estimated based on the recommendations of [17] as explained in Section 2.4.1.1.

For all the present simulations, the grid dimensions $i_{\max} \cdot k_{\max}$ and geometrical parameters z_{surf}, z_{am}, $l_{\mathrm{m,L}}$ and $l_{\mathrm{m,R}}$ defined in Section 4.3.3.3 and Figure 4.6 were chosen such that $\Delta x = \Delta z$, which ensured the convergence and accuracy of the numerical solution as described in Section 2.4.1.1 (Table 4.1).

Following the approach taken in [2,14,17], the time it takes for the surface to propagate along the profile centreline to a depth defined by the experimental first pass profile, T_{pass}, was used in the simulations to obtain an equivalent time, i.e. $2r_{\mathrm{s}}/v_t$, that a given masked channel cross-section is exposed to the particle jet during a single pass of the nozzle, as explained in Section 2.3.2. T_{pass} for the target (glass and PMMA) and the mask (FG, RM and BRM) in Table 4.1 was obtained from a best fit of the simulated and experimental initial pass profile depths, by matching the depths within the total percentage standard deviations of the corresponding mass flow rate measurements (see Appendix A, Tables A-11-A-16). This is similar to the method used in [36] to estimate the erosive efficacy due to a single pass of the nozzle and hence C, as explained in Sections 2.3.2. For all cases in Figures 4.7-4.11, T_{pass} for glass, PMMA, FG, RM and BRM varied between 2.5-5.1 s pass^{-1}, 3.9-11.2 s pass^{-1}, 2.7-18.9 s pass^{-1}, 1.1-2.8 s pass^{-1} and 5.6-7.5 s pass^{-1}, respectively. The different values resulted from experimental fluctuations in flux and/or velocity between experiments, and the fact that different scanning speeds were used in Figures 4.7, 4.8 and 4.10, and Figures 4.9 and 4.11.

The total simulation times for each case in Figures 4.7-4.11 were obtained by multiplying the maximum number of nozzle passes (Table 4.1) by (the chosen) T_{pass} for the target. Thus, since the target and the mask surfaces evolve in the simulation simultaneously, and, as shown in Table 4.1, the T_{pass} for the target and the mask in each case were generally not equal, the H_{b} and H_{d} in eqs. (3.16), (3.17), (3.19) and (3.20) for the mask had to be multiplied by the ratio of $T_{\mathrm{pass,mask}}/T_{\mathrm{pass,target}}$ in order to solve the equation of motion defined by eq. (2.21) as described in Section 4.3.3.2. The resulting representative mean values of Δt determined by eq. (4.17) and the number of iterations and band re-initializations (Section 3.3.3.4) are listed in Table 4.1. On a 2.93 GHz Quad-core Intel i7 CPU with 8 GB of RAM, the ETs for all the cases were approximately between 8 to 135 min.

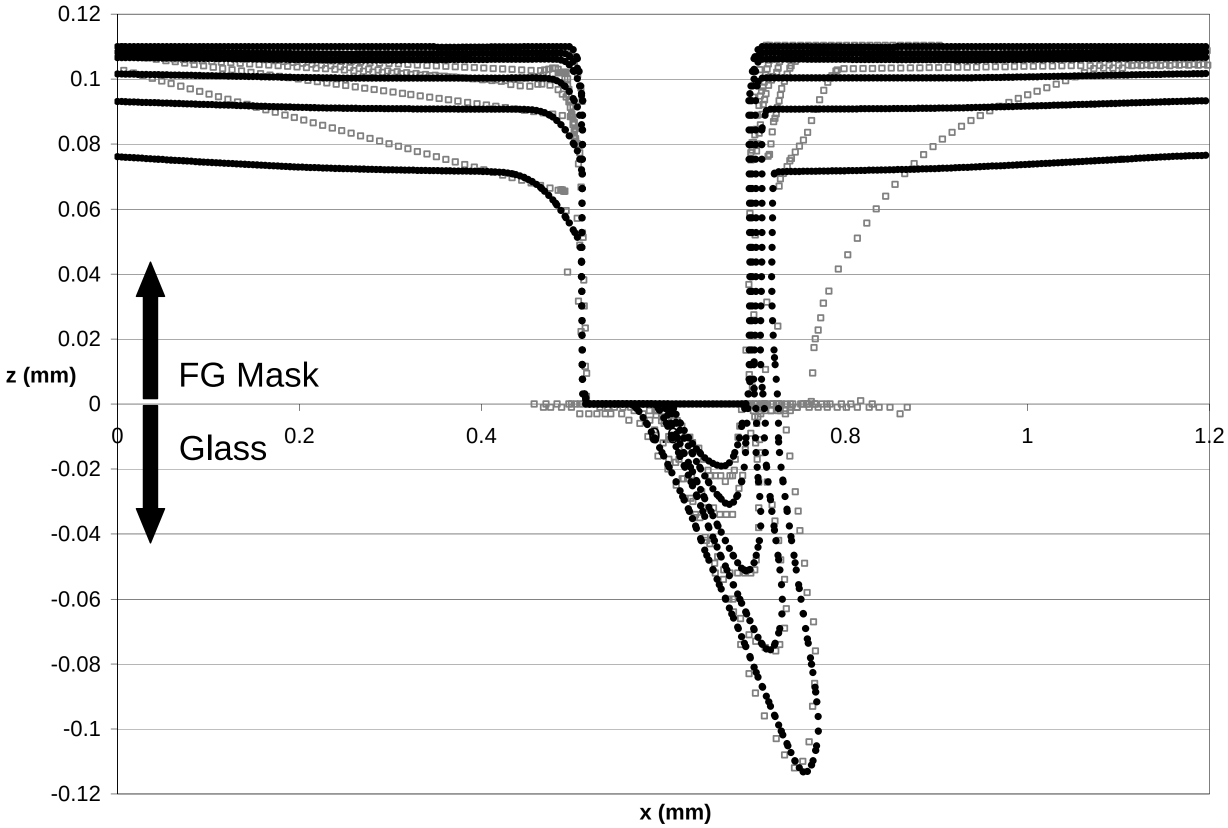


Figure 4.7. Predicted (●) and measured (☐) surface evolution of glass channels ($z \leq 0$) with FG mask ($z \geq 0$) machined at $\alpha = 45°$ after 0, 2, 4, 10, 20 and 40 passes of the nozzle. All model inputs are specified in Table 4.1.

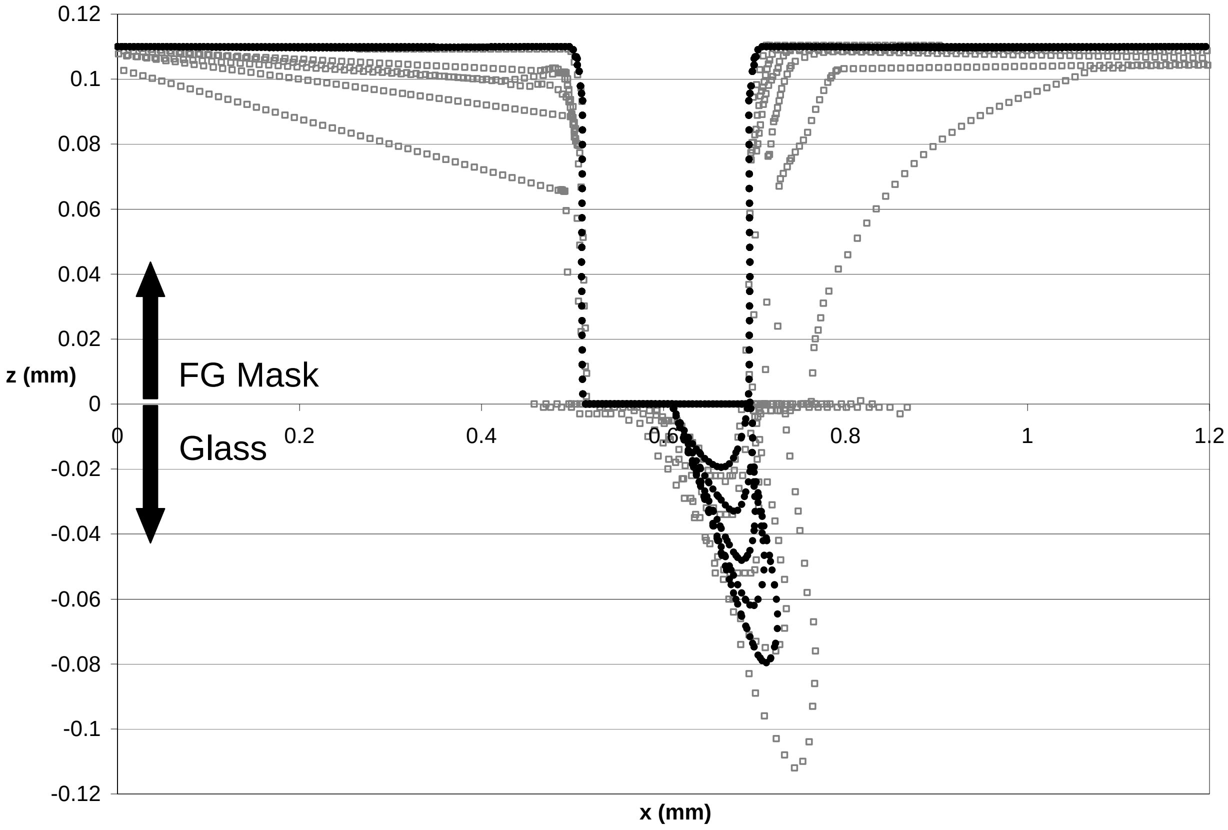


Figure 4.8. The case of Figure 4.7 re-plotted for the case where the predicted (•) surface evolution does not consider mask wear.

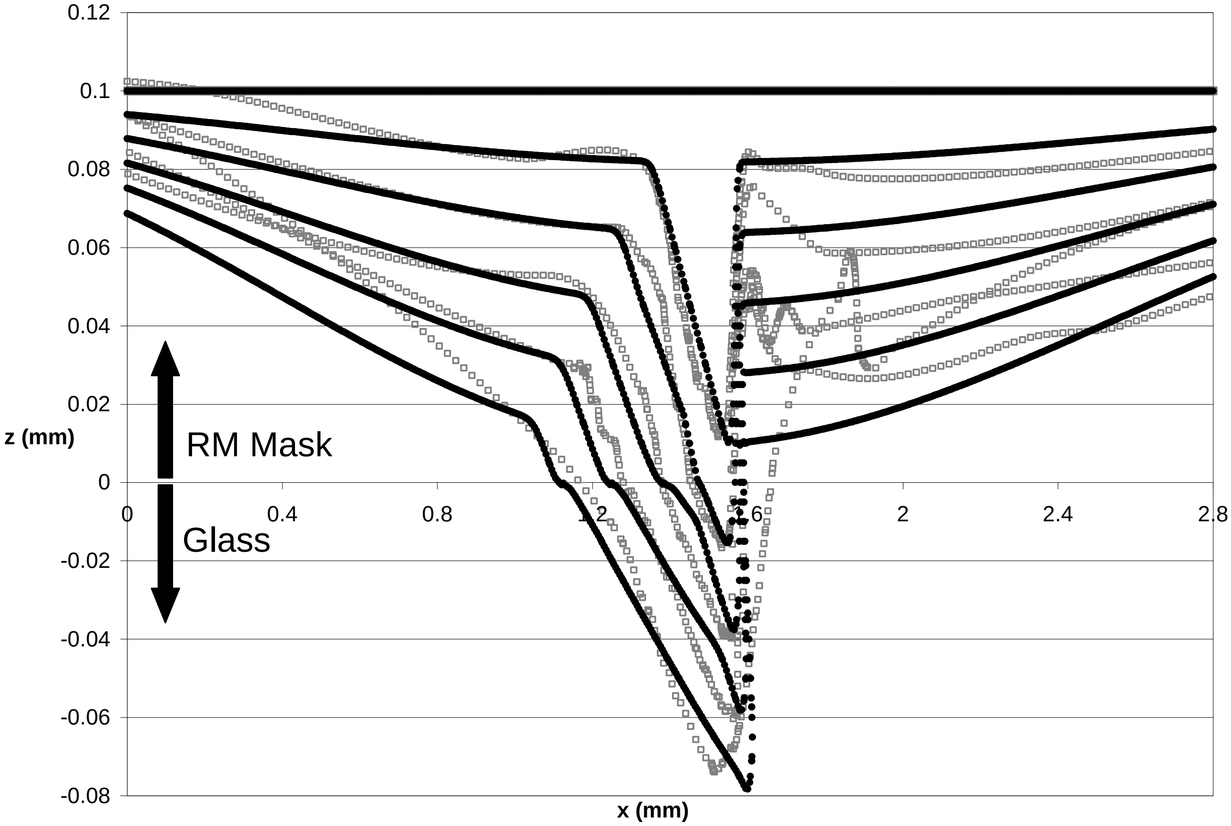


Figure 4.9. Predicted (●) and measured (□) surface evolution of glass channels ($z \leq 0$) with RM mask ($z \geq 0$) machined at $\alpha = 45°$ after 0, 2, 4, 6, 8 and 10 passes of the nozzle. All model inputs are specified in Table 4.1.

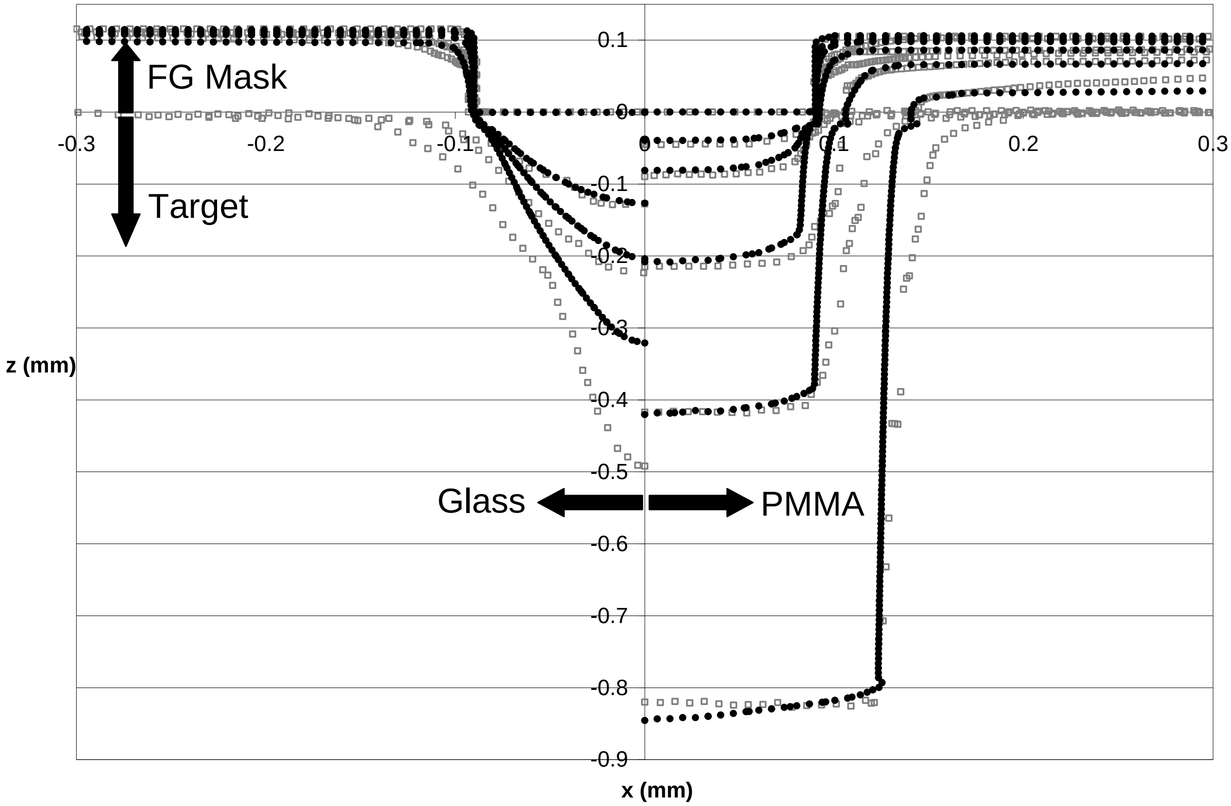


Figure 4.10. Predicted (•) and measured (□) surface evolution of glass ($x \leq 0$) and PMMA ($x \geq 0$) channels ($z \leq 0$) with FG masks ($z \geq 0$) machined at $\alpha = 90°$ after 0, 2, 4, 10 and 0, 2, 4, 10, 20, 40 passes of the nozzle, respectively. Only half the profiles are shown due to symmetry. All model inputs are specified in Table 4.1.

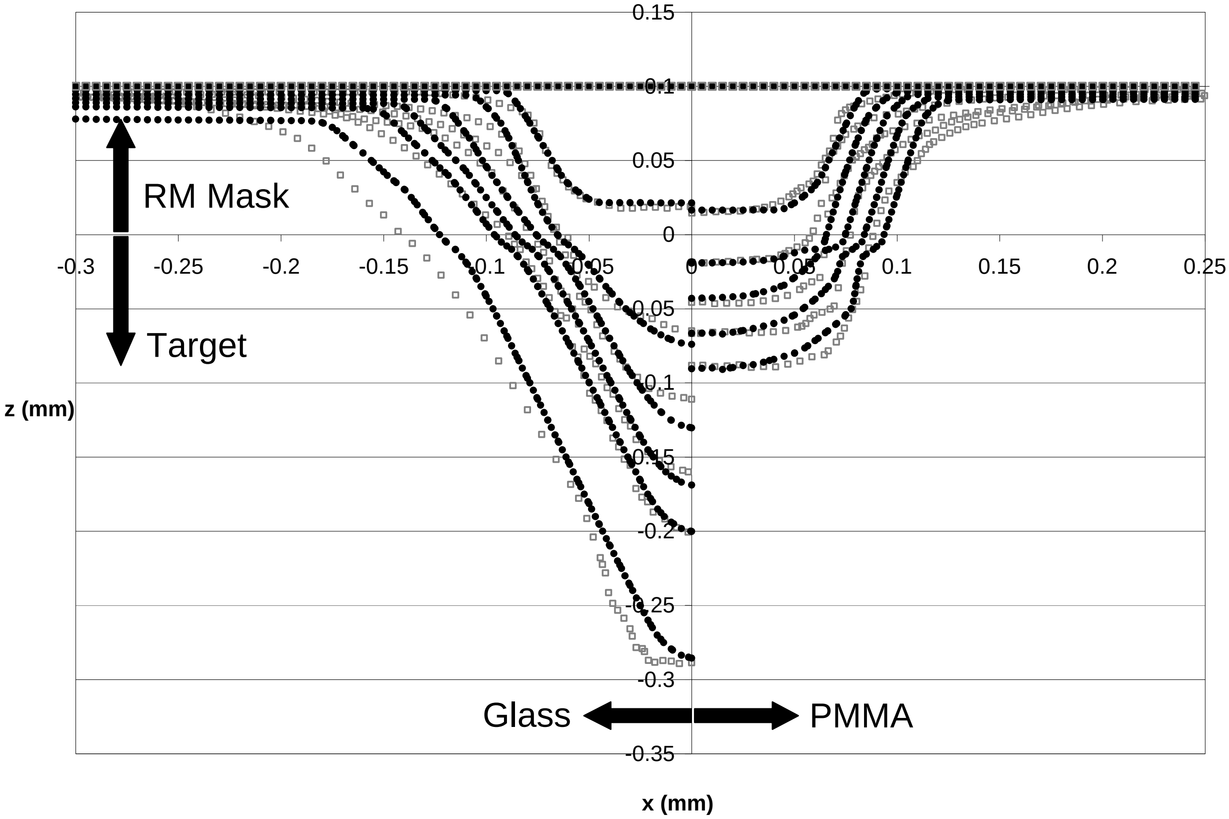


Figure 4.11. Predicted (●) and measured (□) surface evolution of glass ($x \leq 0$) and PMMA ($x \geq 0$) channels ($z \leq 0$) with RM masks ($z \geq 0$) machined at $\alpha = 90°$ after 0, 2, 4, 6, 8, 10, 16 and 0, 2, 4, 6, 8, 10 passes of the nozzle, respectively. Only half the profiles are shown due to symmetry. All model inputs are specified in Table 4.1.

Table 4.1. Model inputs and numerical outputs.

Model parameters	Figure 4.7 {Figure 4.8}*	Figure 4.9	Figure 4.10 ($x \leq 0$)	Figure 4.10 ($x \geq 0$)	Figure 4.11 ($x \leq 0$)	Figure 4.11 ($x \geq 0$)
Model inputs						
h (mm)	20	20	20	20	20	20
H_m (μm)	110	100	115	105	100	100
W_m (μm)	185	130	180	180	130	130
α (°)	45	45	90	90	90	90
v_t (mm s^{-1})	1.0	2.0	1.0	1.0	2.0	2.0
$\dot{M}$ (avg.) (g min^{-1})	2.56	3.84	3.27	2.24	3.60	3.56
ρ (kg m^{-3})	2200 (glass), 7712 (UR) (FG)	2200 (glass),1292 (UR) (RM/BRM)	2200 (glass), 7712 (UR) (FG)	1190 (PMMA), 7712 (UR) (FG)	2200 (glass),1292 (UR) (RM/BRM)	1190 (PMMA),1292 (UR) (RM/BRM)
Hv (GPa)	- (glass), 5.67 (FG)	- (glass),0.1 (UR) (RM),0.25 (BRM)	- (glass), 5.67 (FG)	0.25 [2] (PMMA) 5.67 (FG)	- (glass),0.1 (UR) (RM), 0.25 (BRM)	0.25 [2] (PMMA), 0.1 (UR) (RM),0.25 (BRM)
V_0 (m s^{-1}), β ( )	162 [14],15 [14]	162 [14],15 [14]	162 [14],15 [14]	162 [14],15 [14]	162 [14],15 [14]	162 [14],15 [14]
μ_l ( ), σ_l ( )	-11.6 [19],0.5 [19]	-11.6 [19],0.5 [19]	-11.6 [19],0.5 [19]	-11.6 [19],0.5 [19]	-11.6 [19],0.5 [19]	-11.6 [19],0.5 [19]
k_v ( )	1.43 [14] (glass), 1.73 (UR) (FG)	1.43 [14] (glass), 2.05 (UR) (RM), 1.41 (UR) (BRM)	1.43 [14] (glass), 1.73 (UR) (FG)	2.00 [2] (PMMA), 1.73 (UR) (FG)	1.43 [14] (glass), 2.05 (UR) (RM), 1.41 (UR) (BRM)	2.00 [2] (PMMA), 2.05 (UR) (RM), 1.41 (UR) (BRM)
C (m s^{-1})$^{-k_v}$	8.0 x 10^{-6} [36] (glass), 7.9 x 10^{-8} {0} (UR) (FG)	8.0 x 10^{-6} [36] (glass), 1.4 x 10^{-8} (UR) (RM), 2.5 x 10^{-6} (UR) (BRM)	8.0 x 10^{-6} [36] (glass), 7.9 x 10^{-8} (UR) (FG)	5.7 x 10^{-8} [36] (PMMA), 7.9 x 10^{-8} (UR) (FG)	8.0 x 10^{-6} [36] (glass), 1.4 x 10^{-8} (UR) (RM), 2.5 x 10^{-6} (UR) (BRM)	5.7 x 10^{-8} [36] (PMMA), 1.4 x 10^{-8} (UR) (RM), 2.5 x 10^{-6} (UR) (BRM)
n_1 ( )	- (glass), 0.69 (UR) (FG)	- (glass),1.10 (UR) (RM),2.41 (BRM)	- (glass), 0.69 (UR) (FG)	1.27 [2] (PMMA), 0.69 (UR) (FG)	- (glass),1.10 (UR) (RM),2.41 (BRM)	1.27 [2] (PMMA),1.10 (UR) (RM),2.41 (BRM)
n_2 ( )	- (glass), 1.40 (UR) (FG)	- (glass),58.0 (UR) (RM),9.00 (BRM)	- (glass), 1.40 (UR) (FG)	15.5 [2] (PMMA), 1.40 (UR) (FG)	- (glass),58.0 (UR) (RM),9.00 (BRM)	15.5 [2] (PMMA),58.0 (UR) (RM),9.00 (BRM)
$i_{max} \cdot k_{max}$ ( )	268·83	601·61	115·115	115·163	131·131	201·61
z_{surf} (μm)	200	150	600	900	500	150
$l_{m,L}$ (μm) = $l_{m,R}$ (μm)	510	1435	292	292	260	436
r_m (μm)	14	-	14	14	-	-
z_{am} (μm)	60.1	50.0	48.2	79.5	50.0	50.3
Δx (μm) = Δz (μm)	4.5	5	6.7	6.7	5	5
p_{rs} ( )	0.3501	0.3490	0.3536	0.3437	0.3497	0.3384
N_p ( )	0	2	0	0	2	2
$z_{T,adj}$ (μm)	10	10	20	20	10	10
ε ( )	3.7 x 10^{-6} (glass), 0 (FG)	0 (glass/BRM), 5.2 x 10^{-6} (RM)	3.6 x 10^{-6} (glass), 0 (FG)	0 (PMMA/FG)	5.2 x 10^{-6} (glass/RM), 0 (BRM)	0 (PMMA/BRM), 5.2 x 10^{-6} (RM)
T_{pass} (s pass^{-1})	2.5 (glass),2.7 (FG)	2.7 (glass),2.8 (RM), 7.5 (BRM)	5.1 (glass),12.2 (FG)	11.2 (PMMA),18.9 (FG)	4.0 (glass),1.8 (RM), 5.6 (BRM)	3.9 (PMMA),1.1 (RM), 5.6 (BRM)
Max. no. of nozzle passes ( )	40	10	10	40	16	10
Numerical outputs						
Δt (mean) (s)	2.26 x 10^{-2} {2.64 x 10^{-2}}	1.21 x 10^{-2}	5.60 x 10^{-2}	5.73 x 10^{-2}	2.48 x 10^{-2}	1.46 x 10^{-2}
ET (min)	79.1 {68.5}	134.9	7.6	100.7	24.7	28.1
No. iterations ( )	4344 {3720}	2203	914	7835	2599	2662
No. band re-initializations ( )	9 {6}	13	13	36	21	10

*{Quantity} in the column entries indicates respective input/output for Figure 4.8 if different than that for Figure 4.7.

4.4.2 Comparison of predicted and measured surface evolution

Figures 4.10 and 4.11 show profiles for a given mask, FG and RM, respectively, and two target materials, glass and PMMA, to compare the extent of mask wear and the differences in the evolution of the two features. From these figures, it can be directly seen that: 1) the glass profiles had curved walls and rounded bottoms, while the PMMA profiles had relatively straight walls and rectangular bottoms, consistent with observations in Chapters 2 and 3; and 2) the etch rate of glass was greater than that of PMMA as has been noted in several other studies, e.g. in [17].

In Section 3.4.2, the lack of a mask wear model was identified as a major contributor to the inaccurate prediction of the surface evolution of inclined channels in glass at oblique incidence. For the presently considered cases, the necessity of modelling mask wear can be seen by comparing Figure 4.7, which shows the model-predicted profiles when mask wear is considered, to Figure 4.8, which shows the predictions when mask wear is not modelled, for the oblique incidence case of a FG mask on a glass target. Modelling the mask wear resulted in an increase in width and depth of the target profiles resulting from the surface evolution of the mask. Without mask wear, the portion of the jet outlined by the mask that is 'visible' to the target defined with $\zeta^{+} + \zeta^{-}$ in Figure 4.2 does not increase with time, and hence the modelled target profiles maintain a constant width. As a result, their propagation begins to slow down beyond a certain depth, as also noted in Section 3.4.2. An analogous argument explains the increase in width and depth of the glass target profiles, as well as the increase in width of the PMMA target profiles as explained in Section 3.4.2, for the $\alpha = 90°$ cases in Figures 4.10 and 4.11.

In general, the agreement between the predicted and measured glass and PMMA target profiles of the micro-channels machined at $\alpha = 45°$ and $\alpha = 90°$ in Figures 4.7, 4.9, 4.10 and 4.11 were good, both in terms of depth and overall shape. In the majority of cases, e.g. Figures 4.7 and 4.11, the model fairly accurately predicted the increase in width (and depth) of the target profiles resulting from the surface evolution of the FG and RM masks. However, in Figure 4.9, there was a slight discrepancy between the measured and modelled glass profile shapes. This was likely due to the deformation of the RM mask on the right hand side caused by the poor structural integrity of the elastomeric RM mask resulting from the exposure of interface between the mask and the substrate to the incoming jet at oblique incidence.

On the left hand side of Figure 4.10, discrepancies between the measured and modelled profiles of the channels machined in glass at $\alpha = 90°$ when using a FG mask likely resulted from particle second strike and spatial hindering effects [13,21-24], as also explained in Sections 2.4.1.2 and 3.4.2. As a result of this, in comparison to the modelled profiles in glass, the experimental profiles were deeper beyond 4 passes. Additional discrepancies in Figure 4.10 resulted from mask under-etch, as also seen in [26] when using steel masks as explained in Section 1.2.1, where the measured profiles were wider near the mask-target interface, i.e. $z = 0$ in the figure, and tapered off towards the bottom beyond 2 and 4 passes in glass and PMMA, respectively, when compared to the modelled profiles.

In the modelled PMMA profiles on the right hand side of Figure 4.10, there was a slight 'under-etching effect' at the target side walls, just below the zero depth, due to a combination of the rapid lateral target propagation caused by the high local erosion rate of PMMA at shallow impact angles and the application of $z_{\mathrm{T,adj}}$ in eq. (4.15 (b)), as opposed to eq. (4.15 (a)), necessary so that the slowdown in the numerical surface propagation of the mask edges is avoided, as explained in Section 4.3.2.4. This was not evident in the predicted glass profiles since the local erosion rate of glass is minimal at shallow angles, as explained in Sections 2.2 and 3.4.2.

Overall, the agreement between the predicted and measured FG and RM mask profiles machined on glass and PMMA targets at $\alpha = 45°$ and $\alpha = 90°$ in Figures 4.7, 4.9, 4.10 and 4.11 was fair, both in terms of depth and overall shape. Figure 4.7 shows relatively fair agreement up to ~ 4 passes, and poor agreement beyond that, especially at the right mask. Figure 4.9 shows good agreement up to 8 passes on the left mask, and poor agreement on the right mask. This is partially due to the RM mask deformation on the right side that was mentioned above. Another reason for the discrepancies between the measured and predicted mask profiles in Figures 4.7, 4.9, 4.10 and 4.11 may be the first-order approximation discussed in Section 4.3.2.2 that assumes the masking function at the sloped mask edge to be constant at an average value, determined by assuming a linear mask edge. Nevertheless, it is the prediction of the surface evolution of the target profiles incorporating mask wear that is of practical importance, rather than the shape of the eroding mask itself. Since the model predicts the target profiles quite well, in the majority of cases, it is nevertheless useful.

As a final practical note, RM masks erode relatively fast, when compared to FG masks. From Figures 4.9 and 4.11, it can deduced that in machining features using RM, i.e. elastomeric, masks, the maximum achievable AR is ~ 1, taking into account the increase in feature width due to mask wear. Thus RM masks can be viewed as being more suited to machine small shallow features, whereas FG, i.e. steel, masks can be used to achieve higher ARs such as the PMMA channels in Figure 4.10 where AR > 2.

The next chapter considers the effect of particle second strikes [13,21-24] in an attempt to improve feature predictions in glass (brittle) features for AR > 1.

Chapter 5 Modelling of Surface Evolution in Abrasive Jet Micromachining Including Particle Second Strikes: A Level Set Methodology

5.1 Motivation

As shown in Figures 2.11, 3.7 and 4.10 ($x \leq 0$), the particle second strike effect [13,21-24] can significantly affect the surface evolution of brittle (glass) target features with AR > 1. As discussed in Section 1.2.1, this effect was initially modelled and implemented into an analytical surface evolution model by Slikkerveer and in't Veld [13], which was later extended to include the mask edge effect by Ghobeity et al. [22]. These previous models could not account for all the pertinent effects in order to accurately model the second strike effect. In addition, as discussed in Section 1.2.2, Ciampini and Papini [23] developed a CA computer simulation which addressed the previous limitations of the above mentioned models. However, these types of computer simulations are very computationally expensive. In addition, none of the previous models have considered second strike and mask edge effects for cases in which the nozzle is incident at oblique angles, such as the case depicted in Figure 5.1.

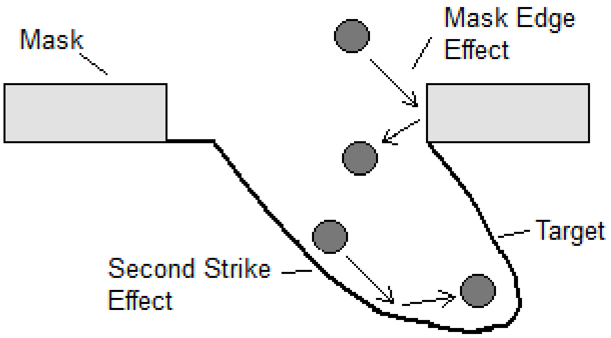


Figure 5.1. Depiction of the particle second strike and mask edge effects in glass at oblique incidence ($\alpha < 90°$).

In the present chapter, the LSM model developed in Chapter 4 is extended to include second strike and mask edge effects in brittle (glass) features made using AJM, using a ray tracing/node tracking algorithm. The second strike model is generalized to any impact angle, and thus presents an improvement upon previous modelling attempts, i.e. for the first time, the prediction of the particle second strike effects from inclined masked brittle features is made possible. The model is verified against previous measurements on masked micro-channels in glass with AR > 1. The present model is also compared against the model of Ghobeity et al. [22] and Ciampini and Papini [23], as well as previous LSM formulations developed in Chapters 2 and 3, which consider and do not consider the effects of second strikes, respectively. Particle second strike effects do not need to be considered in the surface evolution of ductile, e.g. PMMA, features made using AJM since, as explained in Section 3.4.2, ductile features develop straight sidewalls which are approximately aligned with the incoming particle velocity vectors. Thus, ricochet and second strike of particles to the opposite side walls is highly unlikely. As shown in Figures 2.12, 3.8 and 4.10 ($x \geq 0$), there was no significant discrepancy between predicted and measured surface evolution of ductile (PMMA) features in terms of profile depth and overall shape well beyond AR of 1, without consideration of second strike effects. The majority of the material in this chapter has been accepted for publication in [51].

5.2 Level set modelling of surface evolution in AJM including particle second strikes

5.2.1 Local normal velocity and masking functions

In order to model masked channels including mask erosive wear, as shown in Figure 4.2, and second strikes, the same transformed coordinate system defined by eq. (2.1) was used as that in Section 2.3.1, Figure 3.1 and Section 4.3.1. The velocity of the surface in the direction of the local normal for the AJM of brittle target (glass) and ductile mask (FG) presently considered is defined by eqs. (3.1) and (3.2), respectively, where V and ϕ are defined by eqs. (2.7) and (2.9) with $\bar{y}$ used in place of $y = (r_s - v_t t)$ from Section 4.3.1, since similar blasting conditions were used as those in Sections 2.2, 3.2 and 4.2. In addition, the same masking function $M_{x/0}\big|_{\text{Unif}}$ defined by eq. (4.16) is used to adjust the particle mass flux ϕ incident to the glass target and the FG mask as that in Section 4.3.2.

5.2.2 LSM model implementation

5.2.2.1 Partial derivatives, surface initialization and geometric variables

Following the approach of Sections 2.3.4.1, 3.3.3.1 and 4.3.3.1, eq. (2.12) was used to approximate partial derivatives used in obtaining the solution to $\Phi(x,z,t)$ and geometric variables. The initial hybrid mask-target surface was represented by an inverse 'hat' shape, and the grid was divided into 2 material regions, FG and glass, as shown in Figure 4.6. The $\Phi(x,z,t)$ was initialized and re-initialized using the SDF defined by eq. (2.29) as explained in Section 3.3.3.1. The geometric variables $\vec{n}$ and K were obtained using eq. (3.13). In addition, with $|\nabla\Phi(x,z)| = 1$ from eq. (3.12) and Section 3.3.3.1, the tangent to the surface, $\vec{t}$ (Figure 5.2), becomes

$$\vec{t} = (-\Phi_z, \Phi_x) \tag{5.1}$$

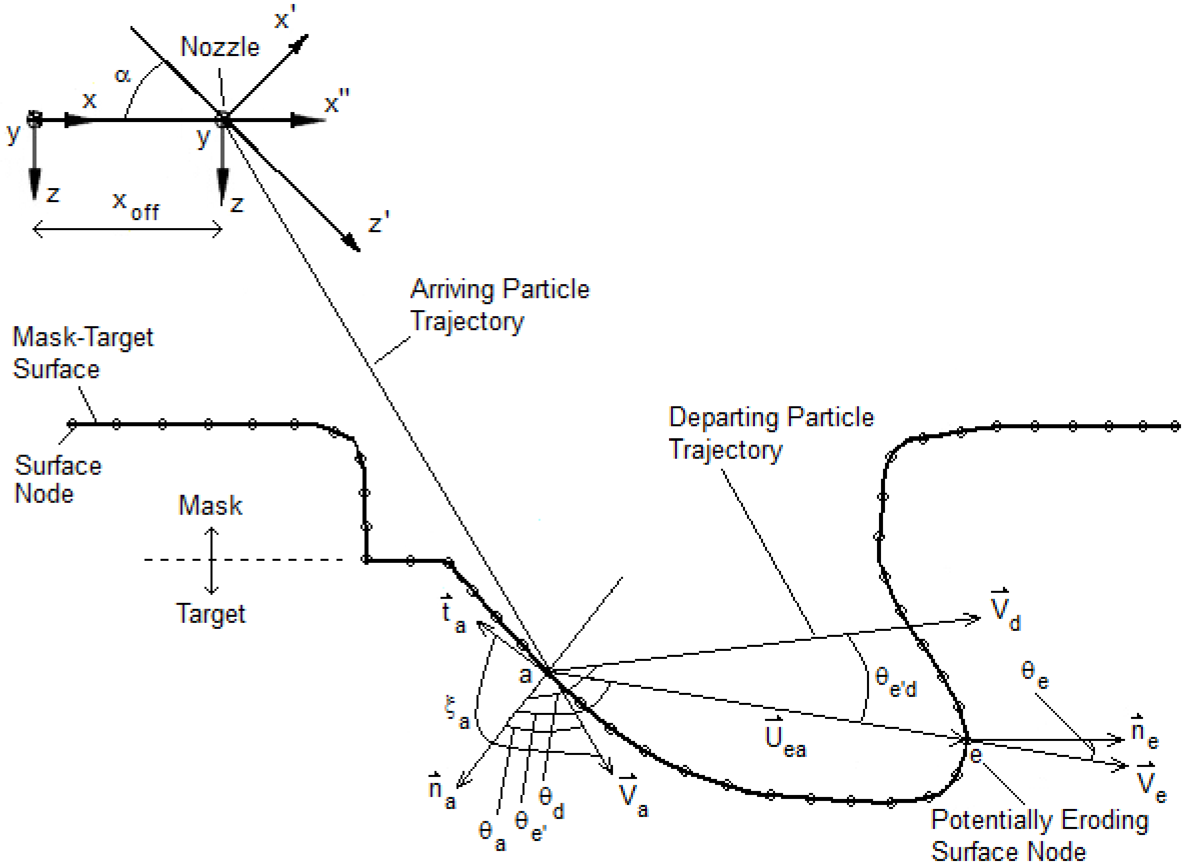


Figure 5.2. Geometry for modelling particle second strike.

5.2.2.2 Particle second strike formulation

The following assumptions were made in modelling the particle second strike effect:

a. First and secondary particle strike contributions occurred over the same time step. The same modelling approach was used in [13,22]. Since the particle velocities were on the order of 150 m s^{-1}, and the maximum distance a particle can travel after ricocheting is on the order of the mask opening width (~500 μm), the time for a particle to ricochet is on the order of 3.3 μs, which was much smaller than the average time step, which was on the order of 14-40 ms (see Section 5.3.1). Therefore, this assumption introduces negligible error.

b. All particles travelled in the *x*-*z* plane, at a $\bar{y}$ location along the scanning direction that approximates the total flux seen in a single scan of the nozzle by an equivalent 2 dimensional system, as discussed in Sections 4.3.1 and 5.2.1.

c. Only first and secondary strike contributions were considered, assuming that third and higher ricochets would carry negligible energy towards erosion, as in [13,22].

d. Inter-particle collision were ignored since the range of the particle mass flow rate typically used in AJM, and considered in the present study (see Section 5.3.1) was low enough such that particles ricocheting from the surface did not interfere with arriving particles [33-35], as in Sections 2.2, 3.2 and 4.2.

In the analysis, the hybrid mask-target surface is composed of surface nodes (Figure 5.2), or a collection of points which are not the same as the grid nodes, since the actual surface is usually in between the grid nodes, as explained in Sections 2.3.4.4 and 3.3.3.6 and shown in Figures 2.4 and 3.5. The initial particle strike is calculated by tracking particle trajectories originating from the nozzle and arriving to each of the surface nodes, which could be the mask or the target. The example in Figure 5.2 only shows an initial particle arriving to the target, but can be generalized to the entire mask-target surface. Surface nodes which define the 'top' of the mask as shown in Figure 4.4, and which are not visible to the initial particle trajectory, as shown in Figure 4.4 along with eq. (4.14 (c)) and Figure 4.5 along with eq. (4.15 (c)), resulting from the application of a mask as described in Sections 4.3.2 and 5.2.1, will obviously not contribute to the secondary strike effect.

The secondary impact can result from particle target-to-target or mask-to-target ricochet, i.e. mask edge effect. The algorithm uses ray tracing and iterates through each of the surface nodes where an initial particle strike already occurred, e.g. node 'e' in Figure 5.2, and checks whether a secondary collision is plausible from any of the other surface nodes, e.g. node 'a' in Figure 5.2. Following the example in Figure 5.2, a particle arrives with a velocity vector $\vec{V}_\mathrm{a}$ to a node 'a' where, using eqs. (3.13) and (5.1), the surface normal and tangent are defined as

$$\vec{n}_\mathrm{a} = (\Phi_x, \Phi_z)_\mathrm{a} \tag{5.2}$$

and

$$\vec{t}_\mathrm{a} = (-\Phi_z, \Phi_x)_\mathrm{a} \tag{5.3}$$

and with this, the angles between $\vec{V}_\mathrm{a}$ and $\vec{n}_\mathrm{a}$, θ_a, and between $\vec{V}_\mathrm{a}$ and $\vec{t}_\mathrm{a}$, ξ_a, are defined as

$$\theta_{\mathrm{a}} = \cos^{-1}\left(\frac{\vec{V}_{\mathrm{a}}}{\left|\vec{V}_{\mathrm{a}}\right|} \bullet \vec{n}_{\mathrm{a}}\right) = \cos^{-1}\left[\frac{x''_{\mathrm{a}}\left(\Phi_x\right)_{\mathrm{a}} + z_{\mathrm{a}}\left(\Phi_z\right)_{\mathrm{a}}}{\sqrt{x''^2_{\mathrm{a}} + z^2_{\mathrm{a}}}}\right] \tag{5.4}$$

and

$$\xi_{\mathrm{a}} = \cos^{-1}\left(\frac{\vec{V}_{\mathrm{a}}}{\left|\vec{V}_{\mathrm{a}}\right|} \bullet \vec{t}_{\mathrm{a}}\right) = \cos^{-1}\left[\frac{-x''_{\mathrm{a}}\left(\Phi_z\right)_{\mathrm{a}} + z_{\mathrm{a}}\left(\Phi_x\right)_{\mathrm{a}}}{\sqrt{x''^2_{\mathrm{a}} + z^2_{\mathrm{a}}}}\right] \tag{5.5}$$

Ideally, the particle will rebound from node ‘a’ and arrive at a potentially eroding surface node ‘e’ under consideration with a velocity vector $\vec{V}_{\mathrm{e}}$ where with eq. (3.13) the surface normal is defined as

$$\vec{n}_{\mathrm{e}} = (\Phi_x, \Phi_z)_{\mathrm{e}} \tag{5.6}$$

and the unit vector distance from nodes ‘a’ to ‘e’, $\vec{U}_{\mathrm{ea}}$, can be obtained as

$$\vec{U}_{\mathrm{ea}} = \frac{\vec{x}_{\mathrm{e}} - \vec{x}_{\mathrm{a}}}{\left|\vec{x}_{\mathrm{e}} - \vec{x}_{\mathrm{a}}\right|} = \frac{(x''_{\mathrm{e}} - x''_{\mathrm{a}}, z_{\mathrm{e}} - z_{\mathrm{a}})}{\sqrt{(x''_{\mathrm{e}} - x''_{\mathrm{a}})^2 + (z_{\mathrm{e}} - z_{\mathrm{a}})^2}} \tag{5.7}$$

The angle between $\vec{V}_{\mathrm{e}}$ and $\vec{n}_{\mathrm{e}}$, θ_{e}, can then be calculated as

$$\theta_{\mathrm{e}} = \cos^{-1}(\vec{U}_{\mathrm{ea}} \bullet \vec{n}_{\mathrm{e}}) = \cos^{-1}\left[\frac{(x''_{\mathrm{e}} - x''_{\mathrm{a}})\left(\Phi_x\right)_{\mathrm{e}} + (z_{\mathrm{e}} - z_{\mathrm{a}})\left(\Phi_z\right)_{\mathrm{e}}}{\sqrt{(x''_{\mathrm{e}} - x''_{\mathrm{a}})^2 + (z_{\mathrm{e}} - z_{\mathrm{a}})^2}}\right] \tag{5.8}$$

However, the ‘actual’ departing velocity from node ‘a’, $\vec{V}_{\mathrm{d}}$, must be determined using particle kinematics accounting for energy losses. The method of Slikkerveer and in’t Veld [13] was adopted for this purpose, where the departing speed $\left|\vec{V}_{\mathrm{d}}\right|$ was defined as a fraction, f_{v}, of the arriving speed $\left|\vec{V}_{\mathrm{a}}\right|$,

$$\left|\vec{V}_{\mathrm{d}}\right| = f_{\mathrm{v}}\left|\vec{V}_{\mathrm{a}}\right| \tag{5.9}$$

and the departing angle, defined as 180° - θ_d in the present formulation, was defined as a fraction, f_θ, of the arriving angle θ_a, where θ_d is the angle between $\vec{V}_\mathrm{d}$ and $\vec{n}_\mathrm{a}$ in Figure 5.2,

$$\theta_\mathrm{d} = \pi - f_\theta \theta_\mathrm{a} \tag{5.10}$$

It is noted that f_v and f_θ can be different for the mask and the target. Thus, a particle is assumed to strike node ‘e’ a second time only when the particle trajectory defined by $\vec{V}_\mathrm{d}$ ‘closely’ matches that defined by $\vec{V}_\mathrm{e}$ in Figure 5.2. For each node ‘e’, the algorithm iterates through each plausible node ‘a’ and calculates a single minimum displacement, Δs_min, between these trajectories as

$$\Delta s_\mathrm{min} = \min(\tan\theta_\mathrm{e'd} \cdot |\vec{x}_\mathrm{e} - \vec{x}_\mathrm{a}|) \tag{5.11}$$

where $\theta_\mathrm{e'd}$ is defined in Figure 5.2,

$$\theta_\mathrm{e'd} = |\theta_\mathrm{e'} - \theta_\mathrm{d}| \tag{5.12}$$

and using eqs. (5.2) and (5.7), the angle between $\vec{V}_\mathrm{e}$ and $\vec{n}_\mathrm{a}$, $\theta_\mathrm{e'}$, can be calculated as

$$\theta_\mathrm{e'} = \cos^{-1}(\vec{U}_\mathrm{ea} \bullet \vec{n}_\mathrm{a}) = \cos^{-1}\left[\frac{(x''_\mathrm{e} - x''_\mathrm{a})(\Phi_x)_\mathrm{a} + (z_\mathrm{e} - z_\mathrm{a})(\Phi_z)_\mathrm{a}}{\sqrt{(x''_\mathrm{e} - x''_\mathrm{a})^2 + (z_\mathrm{e} - z_\mathrm{a})^2}}\right] \tag{5.13}$$

A secondary collision will occur at node ‘e’ if the following conditions are met:

$$\Delta s_\mathrm{min} < \Delta s_\mathrm{crit} \tag{5.14}$$

and

$$|\vec{x}_\mathrm{e} - \vec{x}_\mathrm{a}| > U_\mathrm{crit} \tag{5.15}$$

where Δs_{crit} in eq. (5.14) is a critical displacement on the order of 1 node spacing dictating whether a secondary collision will occur at node 'e' and U_{crit} in eq. (5.15) is a critical minimum distance between nodes 'e' and 'a' on the order of mean particle diameter. The condition in eq. (5.15) ensures that eq. (5.11) remains valid and approximates a realistic scenario such that the distance between nodes 'e' and 'a' must be larger than the mean particle diameter so that a particle has enough room to rebound. Since a secondary particle trajectory will usually pass between nodes, the secondary strike contribution between successive 'e' surface nodes needs to be linearly weighed to evenly distribute the erosive energy across the nodes. Thus, eq. (5.9) is re-written to obtain the speed $\left|\vec{V}_{\text{e}}\right|$ as

$$\left|\vec{V}_{\text{e}}\right| = \left(\frac{\Delta s_{\text{crit}} - \Delta s_{\text{min}}}{\Delta s_{\text{crit}}}\right)\left|\vec{V}_{\text{d}}\right| = f_{\text{v}}\left(\frac{\Delta s_{\text{crit}} - \Delta s_{\text{min}}}{\Delta s_{\text{crit}}}\right)\left|\vec{V}_{\text{a}}\right| \tag{5.16}$$

where $\left|\vec{V}_{\text{a}}\right|$ is defined by eq. (2.7) as described in Section 5.2.1 but with x''_{a} and z_{a}, i.e. surface nodal values, in place of x'' and z, respectively. The particle mass flux simply becomes

$$\left|\vec{\phi}_{\text{e}}\right| = \left|\vec{\phi}_{\text{a}}\right| \tag{5.17}$$

where $\left|\vec{\phi}_{\text{a}}\right|$ is defined by eq. (2.9) as described in Section 5.2.1 but with x''_{a} and z_{a} in place of x'' and z, respectively.

This above procedure is repeated for all the nodes 'e'. Only a single secondary strike contribution from a given node 'a' is possible to a given node 'e' if conditions in eqs. (5.14) and (5.15) hold; otherwise there is no secondary strike at node 'e'. Although at first glance the algorithm seems computationally expensive, the algorithm avoids many redundant calculations by checking the visibility between nodes 'a' and 'e' and the rebound direction of the potential secondary strike trajectory. Thus, node 'e' is visible to the potential secondary strike trajectory from node 'a' only if $\theta_{\text{e}'} > 90°$ and $\theta_{\text{e}} < 90°$ in Figure 5.2, which ensures that glancing collisions never occur. The assumption is valid for smooth surfaces that do not have excessive irregularities. In addition, by checking the rebound direction of the potential secondary strike trajectory, many nodes can be ignored. For instance, if $\xi_{\text{a}} > 90°$, as shown in Figure 5.2, the particle will rebound from left to right, with respect to the x'-z' axis. Thus, node 'e' must be to the right of node 'a', or $x'_{\text{e}} > x'_{\text{a}}$, for any calculations to be performed. Similarly, for $\xi_{\text{a}} < 90°$, a particle will rebound from right to left, thus node 'e' must be to the left of node 'a', or $x'_{\text{a}} > x'_{\text{e}}$, for any calculations to be performed. Finally, for ξ_a = 90°, the particle will rebound back along the same arrival trajectory from the nozzle, and thus no secondary strike is possible.

The entire procedure is repeated at each time step and is valid for any impact angle, α.

5.2.2.3 Simplified LSM for non-convex Hamiltonians including particle second strikes

The simplified LSM for non-convex H presented in Sections 3.3.3.2 and 4.3.3.2, and based on Section 2.3.4.2, is now generalized to include the effect of particle second strike. Thus, eq. (2.21) with $|\nabla\Phi(x,z)| = 1$ can be rewritten as

$$\frac{\partial\Phi}{\partial t} + \hat{H} = (\varepsilon K^{\mathrm{c}})\left[H(\nabla\Phi^{\mathrm{c}})^{1\mathrm{st}} + H(\nabla\Phi^{\mathrm{c}})^{2\mathrm{nd}}\right] \tag{5.18}$$

where the superscripts 1st and 2nd indicate first and second strike contributions. $\hat{H}$ from eq. (2.22) now includes the effect of second strike,

$$\hat{H} = H(\nabla\Phi^{\mathrm{c}})^{1\mathrm{st}} + H(\nabla\Phi^{\mathrm{c}})^{2\mathrm{nd}} - \sum_{g=x,z}\gamma^{g}\left(\frac{\Phi_g^{+} - \Phi_g^{-}}{2}\right) \tag{5.19}$$

Combining eqs. (3.2), (3.12), (3.13), (4.16), (5.8), (5.16) and (5.17), the Hamiltonians for the first and second strike contributions on a ductile FG mask, respectively, can be expressed as

$$H^{1\mathrm{st}} = M_{x/0}\Big|_{\mathrm{Unif}}\frac{C}{\rho}(V_{\mathrm{e}}^{1\mathrm{st}})^{k_{\mathrm{v}}}\,\phi_{\mathrm{e}}^{1\mathrm{st}}(\cos\theta_{\mathrm{e}}^{1\mathrm{st}})^{n_1}[1 + Hv(1-\cos\theta_{\mathrm{e}}^{1\mathrm{st}})]^{n_2} \tag{5.20}$$

and

$$H^{2\mathrm{nd}} = \frac{C}{\rho}(V_{\mathrm{e}}^{2\mathrm{nd}})^{k_{\mathrm{v}}}\,\phi_{\mathrm{e}}^{2\mathrm{nd}}(\cos\theta_{\mathrm{e}}^{2\mathrm{nd}})^{n_1}[1 + Hv(1-\cos\theta_{\mathrm{e}}^{2\mathrm{nd}})]^{n_2} \tag{5.21}$$

where $\theta_{\mathrm{e}}^{\mathrm{2nd}}, \phi_{\mathrm{e}}^{\mathrm{2nd}}$ and $V_{\mathrm{e}}^{\mathrm{2nd}}$ are defined by eqs. (5.8), (5.16) and (5.17), respectively, $\theta_{\mathrm{e}}^{\mathrm{1st}}$ is the angle between the initial particle trajectory velocity, $\vec{V}_{\mathrm{e}}^{\mathrm{1st}}$, arriving from the nozzle to node 'e' and $\vec{n}_{\mathrm{e}}$ (Figure 5.2),

$$\theta_{\mathrm{e}}^{\mathrm{1st}} = \cos^{-1}\left(\frac{\vec{V}_{\mathrm{e}}^{\mathrm{1st}}}{\left|\vec{V}_{\mathrm{e}}^{\mathrm{1st}}\right|} \bullet \vec{n}_{\mathrm{e}}\right) = \cos^{-1}\left[\frac{x''_{\mathrm{e}}\left(\Phi_x\right)_{\mathrm{e}} + z_{\mathrm{e}}\left(\Phi_z\right)_{\mathrm{e}}}{\sqrt{x''^{2}_{\mathrm{e}} + z_{\mathrm{e}}^{2}}}\right] \tag{5.22}$$

and $V_{\mathrm{e}}^{\mathrm{1st}}$ and $\phi_{\mathrm{e}}^{\mathrm{1st}}$ are obtained using eqs. (2.7) and (2.9) as described in Section 5.2.1, respectively, but with x''_{e} and z_{e} in place of x'' and z, respectively. It is noted that the Hamiltonians for the first and second strike contributions on a brittle glass target corresponding to eq. (3.1) can be obtained by substituting both $n_1 = k_{\mathrm{v}} + 1$ and $n_2 = 0$ in eqs. (5.20) and (5.21). It should also be noted that the masking function $M_{x/0}\big|_{\mathrm{Unif}}$ only applies to the first strike in eq. (5.20), and not eq. (5.21). γ^g in eq. (5.19), originally defined by eq. (2.23), now includes the effect of second strike,

$$\gamma^{g} = \max\left(\left|H_{\Phi_g}^{\mathrm{1st}}(\nabla\Phi^{\pm})\right| + \left|H_{\Phi_g}^{\mathrm{2nd}}(\nabla\Phi^{\pm})\right|\right) \tag{5.23}$$

From the results of eqs. (3.18) and (3.20), in eq. (5.23), the partial derivatives of eqs. (5.20) and (5.21) with respect to Φ_x and Φ_z, can be expressed as

$$(H^{\mathrm{1st}})_{\Phi_x;\Phi_z} = H^{\mathrm{1st}}\left[\frac{(x'';z)}{\sqrt{x''^{2}+z^{2}}}\left(\frac{n_1}{\cos\theta} - \frac{n_2 Hv}{1 + Hv(1-\cos\theta)}\right)\right] \tag{5.24}$$

and

$$(H^{\mathrm{2nd}})_{\Phi_x;\Phi_z} = H^{\mathrm{2nd}}\left[\frac{(x'';z)}{\sqrt{x''^{2}+z^{2}}}\left(\frac{n_1}{\cos\theta} - \frac{n_2 Hv}{1 + Hv(1-\cos\theta)}\right)\right] \tag{5.25}$$

where

$$\theta = \cos^{-1}\left[\frac{x''\Phi_x + z\Phi_z}{\sqrt{x''^2 + z^2}}\right] \tag{5.26}$$

It is again noted that the brittle target versions of eqs. (5.24) and (5.25) can be obtained by substituting in $n_1 = k_v + 1$ and $n_2 = 0$. The notation (x''; z) in eqs. (5.24) and (5.25) indicates that x'' and z were used when evaluating H_{Φ_x} and H_{Φ_z}, respectively. All the non-constant quantities in eq. (5.26), in the square brackets on the right hand side of eqs. (5.24) and (5.25), in K from eq. (3.13) used in eq. (5.18), and Φ_g^+ and Φ_g^- from eq. (2.12) used in eq. (5.19), are evaluated at the grid nodes, rather than the surface nodes. In contrast, eqs. (5.20)-(5.22) are evaluated at the surface nodes, which is consistent with EVM from Sections 3.3.3.3 and 4.3.3.2 (and 5.2.2.4).

5.2.2.4 EVM and NB LSM

The same EVM presented in Sections 3.3.3.3 and 4.3.3.2 is used in the present formulation. Using eq. (3.22), H^{1st} and H^{2nd} in eqs. (5.20), (5.21), (5.24) and (5.25) at the surface nodes become $H_{\text{ext}}^{\text{1st}}$ and $H_{\text{ext}}^{\text{2nd}}$ at the grid nodes. Equations (5.24) and (5.25) thus become $(H_{\text{ext}}^{\text{1st}})_{\Phi_x;\Phi_z}$ and $(H_{\text{ext}}^{\text{2nd}})_{\Phi_x;\Phi_z}$. These new expressions now replace $H^{\text{1st}}, H^{\text{2nd}}, H_{\Phi_g}^{\text{1st}}$ and $H_{\Phi_g}^{\text{2nd}}$ in eqs. (5.18), (5.19) and (5.23) in solving eq. (5.18) for $t > 0$. In addition, the same NB LSM presented in Section 3.3.3.4 is applied in the present formulation. The NB-extension algorithm in solving eq. (5.18) was analogous to the one outlined in Section 3.3.3.7 in solving eq. (3.14).

5.2.2.5 Grid formulation, boundary conditions, time step and surface interpolation

For the present problem, using the geometry of Figure 4.6, the vertical and horizontal grid limits were obtained in the same way as described in Section 4.3.3.3. These limits were then used to calculate the spatial grid steps and global spatial coordinates at the grid nodes as described in Section 2.3.4.3. The boundary conditions were obtained in the same way as described in Sections 2.3.4.3 and 3.3.3.5. The time step was calculated using the CFL condition [37] described by eq. (4.17), which is now modified to include the effect of second strike,

$$\Delta t \cdot \max\left[\sum_{g=x,z}\left(\frac{\left(\left|\left(H_{\text{ext}}^{\text{1st}}\right)_{\Phi_g}(\nabla\Phi^{\pm})\right|+\left|\left(H_{\text{ext}}^{\text{2nd}}\right)_{\Phi_g}(\nabla\Phi^{\pm})\right|\right)}{\Delta g}+\frac{2\varepsilon\left(\left|\left(H_{\text{ext}}^{\text{1st}}\right)_{\Phi_g}(\nabla\Phi^{\text{c}})\right|+\left|\left(H_{\text{ext}}^{\text{2nd}}\right)_{\Phi_g}(\nabla\Phi^{\text{c}})\right|\right)}{\Delta g^2}\right)\right]<1 \tag{5.27}$$

where the maximum was obtained in the same way as described in Section 3.3.3.5.

Finally, as described in Section 4.3.3.3, the location of the surface, used in obtaining $\Phi(x,z,t)$, $M_{x/0}\big|_{\text{Unif}}$ and the upper and lower bands, as well as FD approximations of the partial derivatives at surface nodes, used in EVM and the presently considered second strike formulation, were obtained using interpolation as described in Section 3.3.3.6.

5.3 Results and discussion

5.3.1 Model execution and inputs/outputs

The second strike LSM-based model presented in Section 5.2 was verified by comparison with experimental profiles and LSM models from Sections 2.4.1.1 (Figure 2.11) and 3.4.1 (Figure 3.7) in Figures 5.3-5.5, as well as previously published experimental profiles and computer simulation based analytical model from [22] along with CA based model from [23] in Figure 5.6. The necessary model inputs and resulting outputs for the current model simulations corresponding to Cases 1-5 in Figures 5.3-5.6 are summarized in Table 5.1. In all figures, a zero depth, i.e. $z = 0$, represents the mask-target interface.

For Cases 1-4 in Table 5.1 (Figures 5.3-5.5), the parameters h, H_{m}, W_{m}, r_{m}, α, v_t, $\dot{M}$, Hv and ρ (for glass) were obtained based on experimental conditions and measurements as specified in Sections 2.2, 2.4.1.1, 3.2, 3.4.1 and 4.2. The parameters V_{o}, β, μ_{l}, σ_{l}, ρ (for FG), k_{v}, C, n_1 and n_2 were obtained from Sections 2.4.1.1, 4.2 and 4.4.1 based on previous measurements for the same nozzle, jet conditions, abrasives, target and mask materials as presently utilized.

For Case 5 in Table 5.1 (Figure 5.6), almost the same experimental setup, jet conditions, abrasives and target material were used in [22] as that specified in Sections 2.2, 2.4.1.1, 3.2 and 3.4.1. Thus, the parameters μ_{l}, σ_{l}, h, α and v_t, as well as ρ, k_{v} and C for the glass target were the same as that for Cases 1-4, with the exception of α for Cases 3 and 4, where the jet was at oblique incidence to the target. However, the abrasives were blasted from a 3.8 x 0.3 mm rectangular nozzle in [22] instead of a 0.76 mm inner diameter nozzle, as in Cases 1-4. Thus, for this case, a constant velocity across the target and the mask was assumed, i.e. $V = V_{\mathrm{o}}$ rather than that defined by eq. (2.7) as described in Section 5.2.1, in order to be consistent with the approach in [22]. Also, V_{o} and β were obtained from measurements in [44] as described in Section 2.4.2.1, and $\dot{M}$ from measurements in [17]. In addition, the assumed tempered steel FG mask attached to the target via clamps in [22] had different dimensions than that for Cases 1-4, and due to a lack of published data, erosion and material constants for this mask could not be verified to be the same as that for Cases 1-4. Thus, it was assumed that no mask wear occurred in Case 5, as was also done in [22], and thus the parameters ρ, Hv, k_{v}, C, n_1 and n_2 for the mask were not specified.

For all cases in Table 5.1, the parameters p_{rs} and $z_{\mathrm{T,adj}}$ were obtained as explained in Sections 4.3.1 and 4.3.2.4, respectively. The parameter ε [16,17,37] was used to smooth the results in modelling glass targets, estimated based on the recommendations of [17] described in Section 2.4.1.1. The parameter ε for Cases 1-4 in Figures 5.3-5.5 was different than that used previously in Figure 2.11 and

Figure 3.7 where the scanning target approach described in Sections 2.4.1.1 and 3.3.1 was used, since the present formulation used the stationary target approach described in Sections 4.3.1 and 5.2.1.

For all cases in Table 5.1, the parameters f_θ and f_v were adjusted to obtain a best fit to the experimental glass profiles for AR > 1 where second strike effects begin to take effect, as was also done in [13,23]. These values were within the ranges of $0.8 \leq f_\theta \leq 1.2$ and $0.2 \leq f_v \leq 0.5$ quoted by Slikkerveer and in't Veld [13], and almost the same as those assumed in [13,23], under similar experimental conditions. The parameters Δs_{crit} and U_{crit} were obtained as defined in Section 5.2.2.2.

For all the present simulations in Table 5.1, the grid dimensions $i_{max} \cdot k_{max}$ and geometrical parameters z_{surf}, z_{am}, $l_{m,L}$ and $l_{m,R}$ defined in Section 4.3.3.3 and Figure 4.6 were chosen such that $\Delta x = \Delta z$, which ensured the convergence and accuracy of the numerical solution as described in Section 2.4.1.1. T_{pass} for the glass target and the FG mask was obtained in the same way as explained in Section 4.4.1, except for Case 5 where it was assumed that no mask wear resulted, as explained above, and thus T_{pass} for the mask was not specified. For all cases in Table 5.1, T_{pass} for the target and the mask varied between 1.5-7.6 s pass^{-1} and 9.1-12.0 s pass^{-1}, respectively, which resulted from possible variations in flux and/or velocity between experiments, and the fact that different nozzles were used for Cases 1-4 and Case 5.

As explained in Section 4.4.1, the total simulation times for all the cases were obtained by multiplying the maximum number of nozzle passes (Table 5.1) by (the chosen) T_{pass} for the target. The H^{1st} and H^{2nd} in eqs. (5.20), (5.21), (5.24) and (5.25) for the mask were multiplied by the ratio of $T_{pass,mask}/T_{pass,target}$. The resulting mean Δt values obtained with eq. (5.27) and the number of iterations and band re-initializations (Section 3.3.3.4) are listed in Table 5.1.

The model was implemented in MATLAB 7.9 (The MathWorks, Inc., Natick, MA, USA). All cases were simulated on a 2.93 GHz Quad-core Intel i7 CPU with 8 GB of RAM and the resulting ETs were approximately between 0.5-2.5 hours. An example program, corresponding to Case 2 in Figure 5.3, is given in Appendix B.

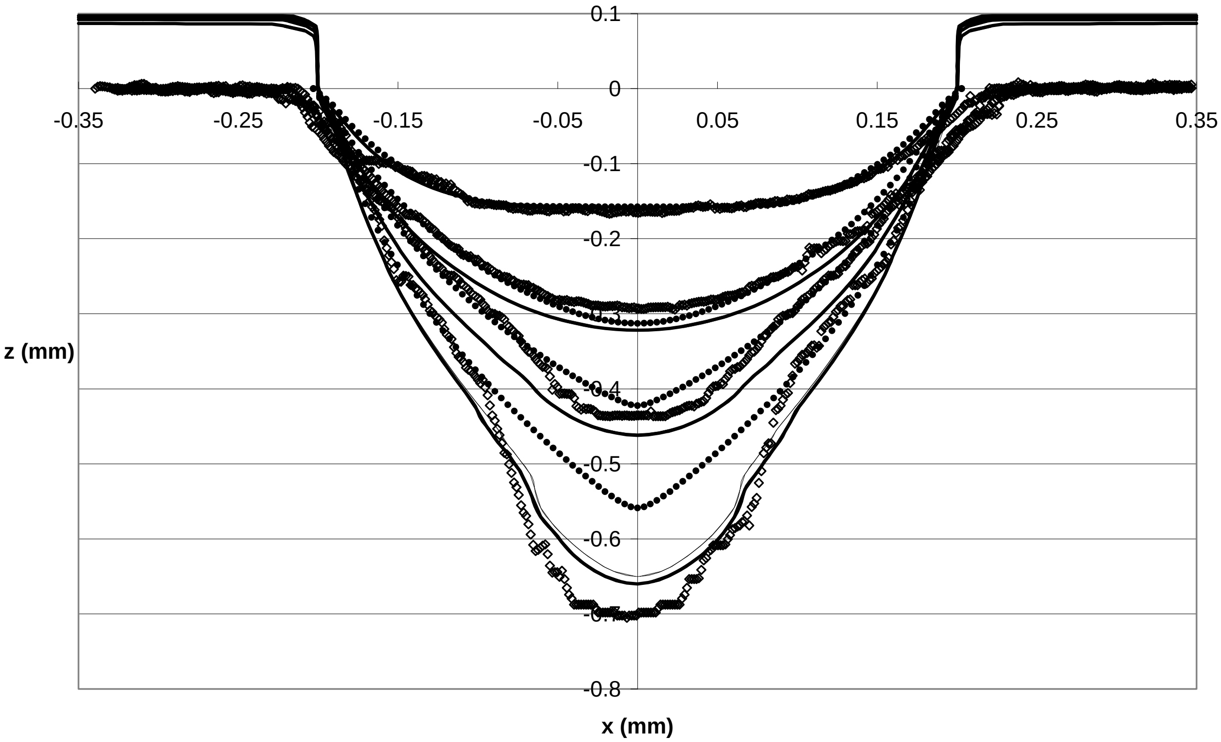


Figure 5.3. Comparison of measured (Section 2.4.1.1; Figure 2.11) (◊) surface evolution of glass FG masked channels machined at α = 90° after 2, 4, 6 and 10 passes of the nozzle with predictions of: (●) previous LSM model (Section 2.4.1.1; Figure 2.11) that did not consider mask wear and second strikes; (——, Case 1) present model that considers mask wear and second strikes off the target only; (▬▬, Case 2) present model that considers mask wear and second strikes both off the target and the mask. All model inputs are specified in Table 5.1.

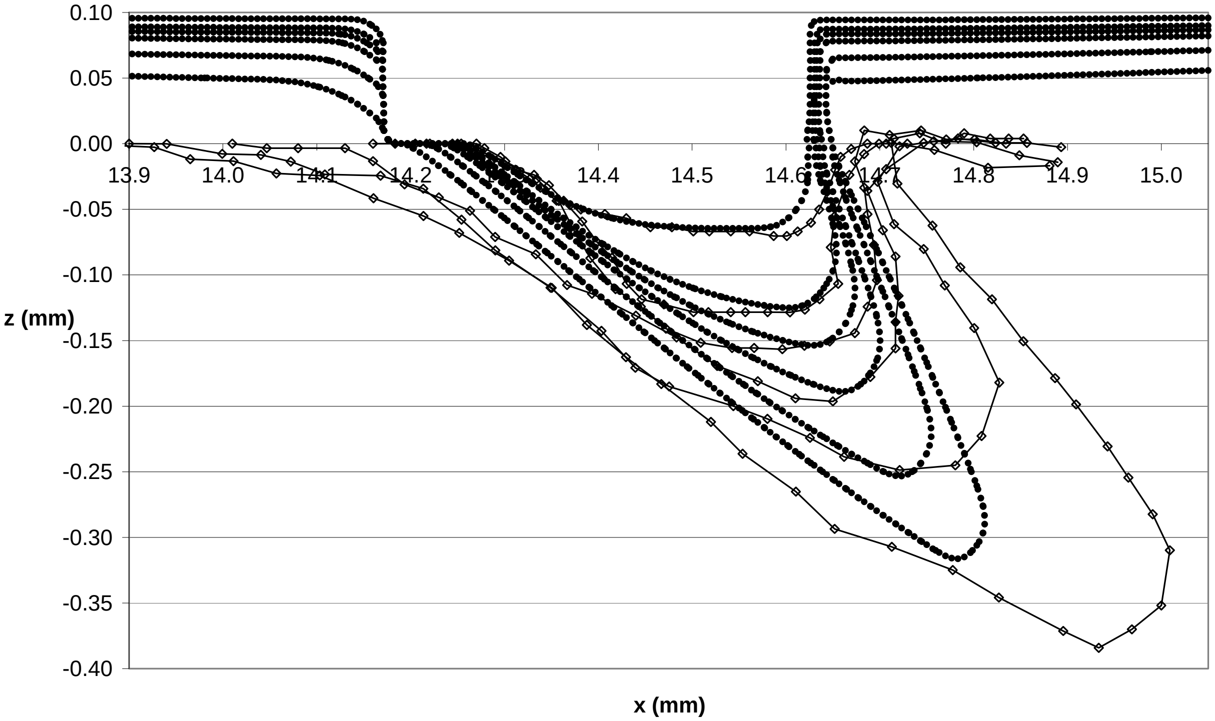


Figure 5.4. Comparison of measured (Section 3.4.1; Figure 3.7) (—◊—) surface evolution of glass FG masked channels machined at $\alpha = 45°$ after 2, 4, 6, 10, 20 and 30 passes of the nozzle with predictions of: (●, Case 3) present model that considers mask wear and does not consider second strikes. All model inputs are specified in Table 5.1.

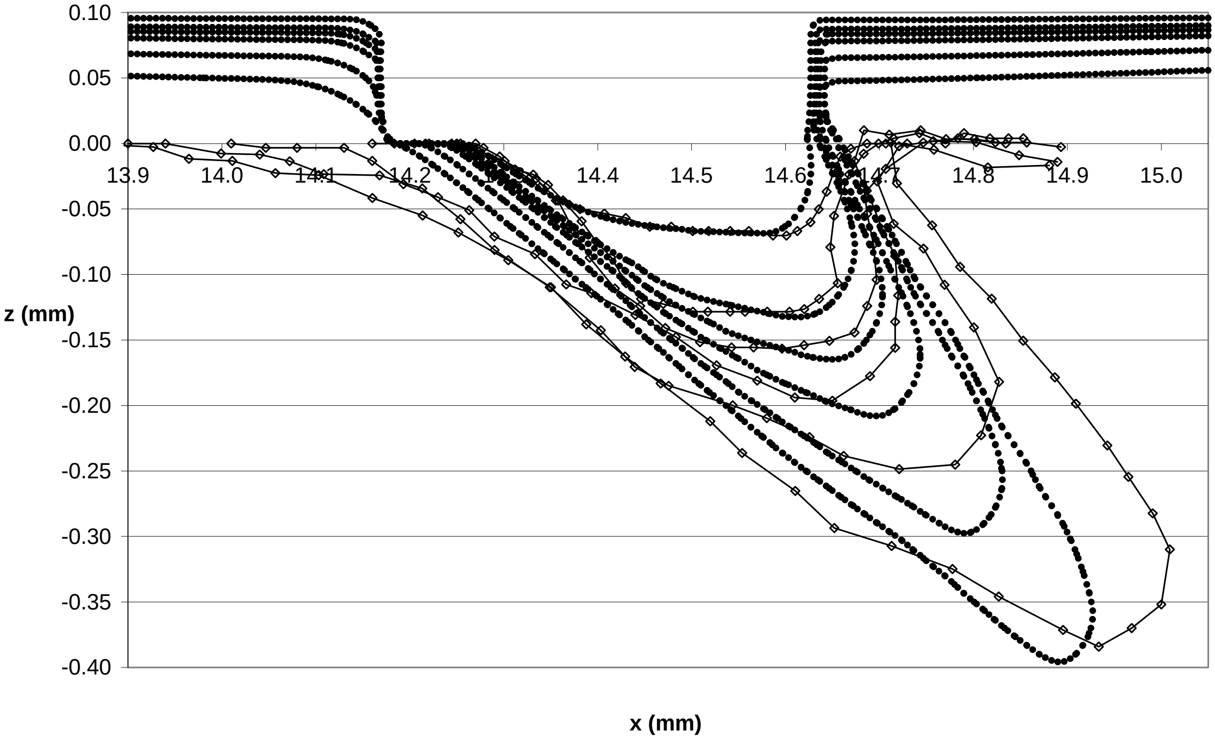


Figure 5.5. Comparison of measured (Section 3.4.1; Figure 3.7) (—◊—) surface evolution of glass FG masked channels machined at $\alpha = 45°$ after 2, 4, 6, 10, 20 and 30 passes of the nozzle with predictions of: (●, Case 4) present model that considers mask wear and second strikes both off the target and the mask. All model inputs are specified in Table 5.1.

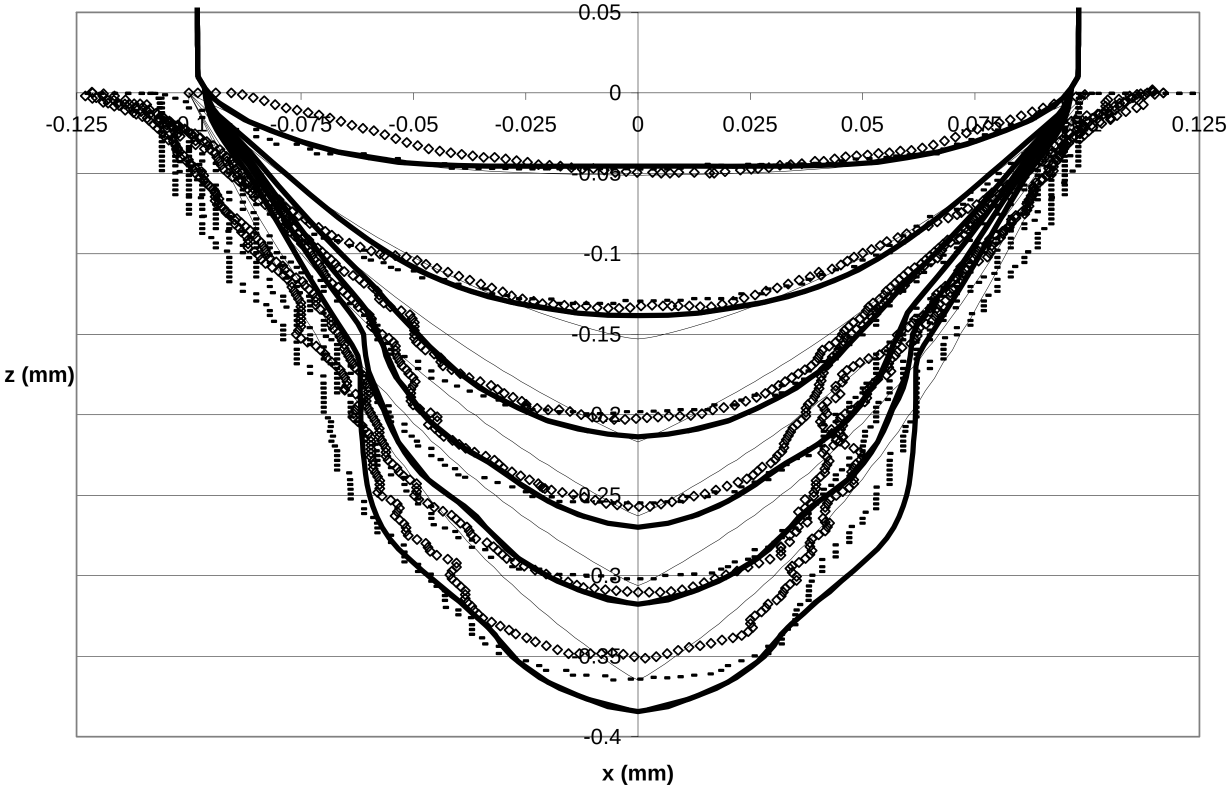


Figure 5.6. Comparison of measured [22] (◊) surface evolution of glass FG masked channels machined at $\alpha = 90°$ after 1, 3, 5, 7, 9 and 12 passes of the nozzle with predictions of: (——) previous [22] computer based analytical model that considered second strikes; (–) previous [23] CA based model that considered second strikes; (▬▬, Case 5) present model that does not consider mask wear and considers second strikes both off the target and the mask. Only a small portion of the simulated mask profiles are shown for ease of comparisons. All model inputs are specified in Table 5.1.

Table 5.1. Model inputs and numerical outputs.

Model parameters	Case 1 and {Case 2}* (Figure 5.3)	Case 3 (Figure 5.4)	Case 4 (Figure 5.5)	Case 5 (Figure 5.6)
Model inputs				
V_0 (m s^{-1}), β( )	162 [14],15 [14]	162 [14],15 [14]	162 [14],15 [14]	148 [44],15 [44]
μ_l ( ), σ_l ( )	-11.6 [19],0.5 [19]	-11.6 [19],0.5 [19]	-11.6 [19],0.5 [19]	-11.6 [19],0.5 [19]
h (mm)	20	20	20	20 [22]
H_m (μm)	100	100	100	1000 [22]
W_m (μm)	400	450	450	200 [22]
r_m (μm)	14	14	14	300 [22]
α (°)	90	45	45	90 [22]
v_t (mm s^{-1})	1.0	1.0	1.0	1.0 [22]
$\dot{M}$ (g min^{-1})	2.63	2.70 - 9.65 x $10^{-3}\cdot t$ (s)	2.70 - 9.65 x $10^{-3}\cdot t$ (s)	5.21 [17]
ρ (kg m^{-3})	2200 (glass), 7712 (UR) (FG)	2200 (glass), 7712 (UR) (FG)	2200 (glass), 7712 (UR) (FG)	2200 (glass), - (FG)
Hv (GPa)	- (glass),5.67 (FG)	- (glass),5.67 (FG)	- (glass),5.67 (FG)	- (glass),- (FG)
k_v ( )	1.43 [14] (glass), 1.73 (UR) (FG)	1.43 [14] (glass), 1.73 (UR) (FG)	1.43 [14] (glass), 1.73 (UR) (FG)	1.43 [14] (glass), - (FG)
C (m s^{-1})$^{-k_v}$	8.00 x 10^{-6} [36] (glass), 7.90 x 10^{-8} (UR) (FG)	8.00 x 10^{-6} [36] (glass), 7.90 x 10^{-8} (UR) (FG)	8.00 x 10^{-6} [36] (glass), 7.90 x 10^{-8} (UR) (FG)	8.00 x 10^{-6} [36] (glass), - (FG)
n_1 ( )	- (glass),0.69 (UR) (FG)	- (glass),0.69 (UR) (FG)	- (glass),0.69 (UR) (FG)	- (glass),- (FG)
n_2 ( )	- (glass),1.40 (UR) (FG)	- (glass),1.40 (UR) (FG)	- (glass),1.40 (UR) (FG)	- (glass),- (FG)
p_{rs} ( )	0.3536	0.3501	0.3501	0.3536
$z_{T,adj}$ (μm)	20	10	10	20
ε( )	8.0 x 10^{-6} (glass),0 (FG)	9.0 x 10^{-6} (glass),0 (FG)	9.0 x 10^{-6} (glass),0 (FG)	4.0 x 10^{-6} (glass),0 (FG)
f_v ( )	0.4 (glass),0 {0.4} (FG)	0 (glass),0 (FG)	0.4 (glass),0.4 (FG)	0.4 (glass),0.4 (FG)
f_θ ( )	0.9 (glass),1.0 (FG)	0.9 (glass),1.0 (FG)	0.9 (glass),1.0 (FG)	0.8 (glass),1.0 (FG)
Δs_{crit} (μm), U_{crit} (μm)	6.7,25	6.7,25	6.7,25	6.7,25
$i_{max}\cdot k_{max}$ ( )	105·151	211·92	211·92	121·226
z_{surf} (μm)	850	450	450	450
$l_{m,L}$ (μm), $l_{m,R}$ (μm)	150,150	400,549	400,549	300,300
z_{am} (μm)	50.0	54.7	54.7	50.0
Δx (μm) = Δz (μm)	6.7	6.7	6.7	6.7
T_{pass} (s pass^{-1})	7.6 (glass),12.0 (FG)	7.0 (glass),9.1 (FG)	7.0 (glass),9.1 (FG)	1.5 (glass),- (FG)
Max. no. of nozzle passes ( )	10	30	30	12
Numerical outputs				
Δt (mean) (s)	4.06 x 10^{-2} {4.16 x 10^{-2}}	3.80 x 10^{-2}	1.81 x 10^{-2}	1.40 x 10^{-2}
ET (min)	32.5 {30.6}	55.7	154.2	39.8
No. iterations ( )	3324 {3318}	3870	8921	1729
No. band re-initializations ( )	27 {28}	16	22	17

*{Quantity} in the column entries indicates respective input/output for Case 2 if different than that for Case 1.

5.3.2 Comparisons with LSM model and experiments at α = 90° from Section 2.4.1.1 (Figure 2.11)

In Figure 5.3, the predictions of the current LSM model which includes the effects of particle second strike and mask wear are compared to experiments at α = 90° and LSM model which did not consider particle second strikes and mask wear from Section 2.4.1.1 (Figure 2.11). The effect of mask wear in increasing the width and depth of the feature was insignificant for this case. However, the importance of modelling the second strike effect can be seen in Figure 5.3. Whereas the previous LSM model failed to predict the udder shape and under-predicted the measured profile depths for AR > 1, the present model successfully predicted the udder shape and had a much better overall agreement with the measured glass profiles both in terms of centre depth and overall shape. Figure 5.3 also compared the present model predicted surface evolution considering ricochet and second particles strikes from both the target and the mask (Table 5.1, Case 2), with that considering second strikes off the target only (Table 5.1, Case 1). The mask edge effects were minor in this case due to the low mask height which resulted in only a slightly deeper and wider target profile after 10 passes when compared to the case which did not consider mask edge effects. For 2-6 passes, the simulated profiles with and without consideration of the mask edge effects overlapped.

5.3.3 Comparisons with LSM model and experiments at $\alpha = 45°$ from Section 3.4.1 (Figure 3.7)

In Figures 5.4 and 5.5, the predictions of the present LSM model were compared to experiments at $\alpha = 45°$ from Section 3.4.1 (Figure 3.7). The predictions of the LSM model which did not consider particle second strikes and mask wear are shown in Figure 3.7. As can be seen in Figure 3.7, the overall agreement between the previous model predicted and the measured glass profiles of the micro-channels machined at $\alpha = 45°$ was poor both in terms of the overall shape and the depth, which were under-predicted beyond 10 passes where the effective AR > 1. In Figure 5.4, in order to isolate the effects of second strike, the present model, but considering only mask wear effects (Case 3) and not second strike, was compared against the same experiments. Figure 5.4 showed some improvement in agreement between predicted and measured profiles over the previous LSM model of Figure 3.7 in terms of the overall shape, i.e. the simulated profiles in Figure 5.4 were slightly wider and deeper than those in Figure 3.7. Finally, in Figure 5.5, the present model considering both mask wear and second strike effects from both the mask and the target (Case 4) was again compared to the same experimental results as Figures 3.7 and 5.4. In this case, a significant improvement in agreement between predicted and measured profiles was achieved. The simulated profiles in Figure 5.5 were not only much wider and deeper than those in Figure 3.7, but also deeper than those in Figure 5.4. The simulated profiles in Figure 5.5 over-predicted the measured profile centre depths beyond 6 passes, more so than those in Figure 3.7 or 5.4. This perhaps resulted from the fact that the present model did not consider spatial hindering effects. Nonetheless, the overall model predictions which considered both effect of second strike and mask wear were much better than those which did not consider both of these effects. It is thus clear that both the effect of second strike and mask wear must be considered for a fair prediction of surface evolution of inclined micro-channels.

In Section 3.4.2, it was stated that the glass experimental profiles in Figure 3.7 tended to shift their direction of propagation to the right, likely caused by the combination of increase in mask width due to mask wear with time, as shown in Figure 3.9, and second strike particles. Figure 5.5 shows that this effect was actually less significant than previously thought, i.e. the simulated profiles did slightly propagate to the right, but less so than the experimental profiles. The additional discrepancy between predicted and measured profiles beyond 10 passes shown in Figure 5.5 was likely caused by mask under-etch that resulted in a wider profile, and thus a more significant shift in profile propagation direction than that predicted by the model. Mask under-etch, discussed in Section 1.2.1, was also seen in Figure 4.10 in Section 4.4.2 and in [26] when using steel masks.

It should be noted that mask edge effects were very minor for the case presented in Figure 5.5, even more so than for the $\alpha = 90°$ case in Figure 5.3, and hence simulated profiles considering mask wear and second strikes off the target only were not shown, i.e. the simulated profiles with and without consideration of the mask edge effects overlapped.

Finally, for the case presented in Figure 4.7, where the model of Chapter 4 which only considered mask wear effects was compared against similar experiments as that presently considered in Figures 3.7, 5.4, and 5.5 but using a narrower W_m, second strike effects were not significant. This can be said since the agreement between predicted and measured target profiles in Figure 4.7 was very good. This likely resulted since for the case in Figure 4.7, the mask wore and target etched at a rate such that the resulting increase in target width preserved an effective AR such that second strikes were not significant, when compared to the present cases.

5.3.4 Comparisons with previously published semi-computational and cellular automaton based models and experiments at α = 90°

Figure 5.6 shows the predictions of the current LSM model compared to previously published experimental profiles at $\alpha = 90°$ [22] and to semi-computational [22] and cellular automaton (CA) based models [23] described in Sections 1.2.1 and 1.2.2, respectively, which considered particle second strikes. The overall agreement between the predictions of both the current model (Case 5), and those of the CA model with the measured glass profiles of the micro-channels in Figure 5.6 were quite good, in terms of overall profile shape. Both models were able to predict the udder shape for AR > 1, since both models account for the second strike and mask edge effects. However, the present model slightly over-predicted the centre profile depths beyond 5 passes. This likely resulted from the fact that the present model did not consider spatial hindering effects, while the CA model of [23] did.

The computer simulation based analytical model of [22] predicted the measured centre profile depths well in Figure 5.6, but did not predict the overall profile shape well for AR > 1. Instead, the resulting simulated profiles revealed sharp cusps at their centres. In addition, the udder shape was not predicted. This is likely because the model of [22] did not directly model the second strike effect; instead an additional ‘effective particle flux’ originating from the second strike contribution was obtained from the computer simulation and incorporated into the analytical model. In addition, the model of [22] did not consider curvature smoothing, i.e. when $\varepsilon \neq 0$ in eq. (5.18), which the current LSM model took into account, and spatial hindering effects, which the CA model of [23] took into account.

The present LSM formulation presented an improvement over the model of [22] in terms of feature shape prediction as can be seen in Figure 5.6. Also, the current formulation is more computationally efficient than the CA approach, as mentioned in Sections 1.2.2 and 1.2.3. For the case in Figure 5.6, ET ≈ 40 min for the current LSM model, whereas ET ≈ 15 hrs. for the CA model of [23], using similar hardware. In addition, the current model considers particle second strike effects in inclined masked features, as shown in Section 5.3.3, and thus presents a significant improvement over previous modelling work of [13,22,23], which only considered normal incidence cases.

Finally, it should be noted that mask edge effects were very minor for Case 5 in Figure 5.6, as was the case in Figure 5.5, and thus simulated profiles considering second strikes off the target only were not shown. The same conclusion was reached in [22], where the computer simulation showed that the combination of large mask thickness and narrow mask opening width used in this case resulted in particles ricocheting off the mask edge wall and striking the mask edge wall a second time, as opposed to the target surface. The mask edge effect is likely to become important in cases where the ratio of mask thickness, i.e. height, to mask opening width, i.e. feature width, approaches unity.

Chapter 6 Conclusions and Future Work

6.1 Summary

In Chapter 2, a LSM-based approach was presented to model the surface evolution in AJM of unmasked channels machined at normal and oblique jet incidence, as well as masked micro-channels and micro-holes at normal incidence, in both brittle (glass) and ductile (PMMA) targets. The approach was developed to address the limitations of previous modelling approaches, as described in Chapter 1. A previously developed analytical model of the AJM surface advancement problem was recast into level set form. The level set predicted eroded profiles were compared to those experimentally obtained, and to those predicted by existing analytical models and a computer simulation. The proposed model generally showed good predictive capability and improvements over previous modelling attempts. The model developed provided a foundation for simulation of more complicated cases that followed.

In Chapter 3, the formulation presented in Chapter 2 for masked features was extended to allow the prediction of surface evolution in AJM of masked micro-channels in glass and PMMA machined at oblique incidence. The resulting profiles were multi-valued, which necessitated the development of a more complex and computationally efficient NB LSM-based formulation. An extension of a previously developed analytical model from the normal to oblique incidence case allowed the decrease in particle flux near the mask edge to be predicted as a function of the mask height and the jet angular spread. Utilizing the NB LSM approach resulted in, on average, 5 times faster execution times. The general agreement between measured and predicted profiles was fair due to the fact that the proposed model ignored particle second strikes and mask wear effects.

In Chapter 4, the formulation presented in Chapter 3 was extended to allow for the modelling of mask erosive wear in AJM. The model permitted the prediction of the surface evolution at any jet incidence of both the mask and the target simultaneously, by representing them as a hybrid and continuous mask-target surface. The concept of a masking function was introduced to model the adjustment to abrasive mass flux incident to the hybrid surface to reflect the range of particle sizes that are ‘visible’ to this surface. The predictions of the channel surface and eroded mask profiles were compared with measurements on micro-channels machined in both glass and PMMA targets at both normal and oblique incidence, using tempered steel and elastomeric masks. Taking mask wear into account showed significant improvement in agreement between measured and predicted target profiles.

In Chapter 5, the formulation presented in Chapter 4 was extended to include the effect of particle second strikes off a brittle target and the mask, i.e. mask edge effects, at any jet incidence. When compared to LSM models developed in Chapters 2 and 3 that did not account for mask wear and second strike effects, the model significantly improved the prediction of measured masked micro-channels machined in glass. The model also showed improvement over previous semi-computational and cellular automaton based models which considered particle second strikes, either in terms of feature prediction or in execution time.

6.2 Limitations of model

The LSM-based model presented in Chapter 5 predicts the surface evolution of masked micro-channels abrasive jet micromachined at any jet incidence in brittle targets considering particle second strikes, mask edge effects and erosive wear of masks where the target is exposed to the jet from time zero, e.g. FG mask. This model also applies to: 1) cases using ductile targets, e.g. PMMA, by simply changing the target material and erosive parameters and by ignoring the effects of second strike; 2) cases using masks which are initially flat where the target is initially unexposed, e.g. RM mask, by simply changing the initial surface conditions for the mask; 3) surface evolution of masked micro-holes at normal incidence since the model of Chapter 5 uses the *stationary* target/mask approach to approximate the erosive efficacy for the AJM of micro-channels; and 4) surface evolution of unmasked channels at any incidence and unmasked holes at normal incidence, by simply allowing the masking function to be unity everywhere and by modifying the initial surface conditions and grid limits. Thus, the model of Chapter 5 is general and can be used to model any of the previous cases presented in Chapters 2-4. It has the following limitations:

1. The model can only be used in cases where the particle mass flux, i.e. flow rate, is relatively low. For higher mass fluxes, particles ricocheting from the surface can interfere with arriving particles [33-35], thus decreasing the effective flux incident to the surface, and hence the erosion rate. However, the analytical model of Burzynski and Papini [34] showed that the ranges of the particle mass flux used in the present study, typical of those used in AJM operations, were sufficiently low for these effects to be negligible.
2. For the AJM of ductile targets, the model can only be used in cases where particle embedding and temperature, i.e. target surface heating, effects are minimal [2,8,17,31], as shown in Section 2.2. For example, in the AJM of PMMA holes at a scan speed of 0.25 mm s^{-1} in Figure 2.15, the profile depths were over-predicted by the model beyond 7 passes. This was likely due to particle embedding which decreases the effective erosion rate [2,8,17,31]. In addition, for much higher fluxes and longer blasting times than that considered in the present study, the PMMA targets can begin to experience surface heating [2,31] which can affect the erosion rate. Furthermore, the present model cannot be used for ductile polymeric targets such as ABS or soft PDMS that can be very difficult to machine or have very low erosion rates when abrasive jet machined at room temperature [8]. However, the model is still applicable to these materials when machining at cryogenic temperatures, and to a wide range of typical experimental conditions and ductile targets used in AJM, such as, e.g. LUCITE and LEXAN [31].

3. As discussed in Section 4.3.2.2, the model uses a the first-order approximation that assumes the masking function at the sloped mask edge to be constant at an average value, determined by assuming a linear mask edge. This leads to discrepancies between the measured and predicted mask profiles, as was seen in, for instance, Figure 4.7. Nevertheless, as explained in Section 4.4.2, it is the prediction of the surface evolution of the target profiles incorporating mask wear that is of practical importance, which in most cases, the model does fairly well.
4. As discussed in Sections 5.3.3 and 5.3.4, the model ignores spatial hindering effects [13,22,23], and thus can only be used for cases where such effects are minimal, i.e. when the width at the top of the developing udder shape is significantly larger than the mean particle size. For example, for the same mean particle size, the udder shape width in Figure 5.3 was approximately twice as large as that in Figure 5.6, and this was reflected in the better model predictions in Figure 5.3 when compared to Figure 5.6. Nevertheless, for the case presented in Figure 5.6, the CA simulation of [23] showed that the maximum percentage of spatially hindered particles at the profile centres was only approximately 9%, and thus this effect is likely to become significant only at much higher ARs in brittle features than those presently considered.
5. The model cannot be used for prediction of the AJM of holes at oblique incidence, or channels at very slow scan speeds or high mass fluxes, where an appreciable slope in the scan direction at the leading edge of the jet can develop [14], as discussed in Section 2.2. These cases require a 3D formulation because, in the former case, there is no profile symmetry across the scanning direction, and in the latter case, the slope on the leading edge can strongly affect the erosion rate under the given circumstances. However, scan speeds and mass fluxes can be chosen such that the effect of the development of an appreciable slope in the scan direction can be ignored, while still maintaining an efficient erosion rate, as was done in the present study.

6.3 Conclusions and contributions

The main conclusions and contributions of the present study can be summarized as follows:

1. In Chapter 2, a novel approach was presented to model the surface evolution in AJM of unmasked and masked features in both brittle and ductile targets using LSM.
2. For most of the cases considered in Chapter 2, excellent agreement with the experimental surface profiles was obtained using the foundational LSM-based model. The model also showed improvement in terms of feature prediction over existing analytical models and a computer simulation, especially in cases where the evolving feature developed steep sidewalls, e.g. as seen in the surface evolution of PMMA, which could not be predicted by the analytical models.
3. In Chapters 2 and 3, the scanning target approach was used for the first time in modelling the erosive efficacy for the AJM of micro-channels, whereas all previous analytical models assumed a stationary target approach. This approach proved to be useful in modelling brittle, e.g., glass, targets that develop sharp profiles since it delays the onset of high curvature, and hence in many cases avoids the use of curvature smoothing which is computationally expensive.
4. In Chapter 3, an NB LSM-based formulation was developed that, for the first time, allowed the prediction of surface evolution in AJM of inclined masked features.
5. In Chapter 3, it was shown that the developed model for predicting the decrease in particle flux near the mask edge at oblique incidence could also be used in setting up successful experimental runs by calculating the portion of the target that is exposed to the jet prior to machining.
6. In the LSM formulation of Chapter 3, the resulting surface velocity function was non-convex, necessitating a novel extension of an existing extension velocity methodology (EVM) to allow the problem of grid 'visibility' of the particle flux to be properly addressed.
7. The results of Chapter 3 showed that the inclined masked PMMA micro-channels had straight walls and rectangular bottoms, while the glass micro-channels had curved walls and rounded bottoms. It was also shown that changing the angle of incidence does not significantly affect the general shape of masked PMMA channels.
8. Since the LSM-based model developed in Chapter 3 ignored particle second strike and mask wear effects, it was best suited for predicting surface evolution at oblique incidence in ductile substrates with non-eroding masks, since the straight sidewalls in PMMA result in a low likelihood of particle ricochet. The model was not suitable for use on glass, where mask wear and second strike effects significantly affected its ability to accurately predict the resulting surface evolution for AR > 1. In addition, it was shown that the combination of the effects of

mask wear and second strike particles can cause a shift in the propagation direction of the developing glass profiles, although it was later shown that that this effect is actually less significant than originally thought.

9. In Chapter 4, the LSM-based formulation of Chapter 3 was extended to permit the prediction of the surface evolution of both the target and the mask simultaneously. It was the first model that considered the influence of mask wear on features machined using AJM at oblique incidence.
10. The work of Chapter 4 demonstrated for the first time the need to model the adjustment in the mass flux at the mask edge walls themselves, as opposed to the target only, due to the finite size of the particles which limit the flux that is visible to the mask edges. This led to the development of a masking function which generalized the adjustment in the flux to the entire mask-target surface.
11. In Chapter 4, the use of a continuous mask-target surface in the formulation introduced complications that were nevertheless resolved to obtain significant improvement in agreement between measured and predicted target surface profiles when mask wear was taken into account, although the mask profiles were not predicted as well. Mask wear generally resulted in wider and deeper target profiles, when compared to cases where no mask wear was present.
12. In Chapter 5, the LSM-based formulation of Chapter 4 was extended to include second strike and mask edge effects. The model presented was the first to allow prediction of these effects in inclined masked features made using AJM.
13. The work of Chapter 5 demonstrated that the second strike effect becomes important in the prediction of surface evolution of brittle (glass) features abrasive jet micromachined at *any* incidence for AR > 1. The inclusion of the second strike effect into the formulation led to a significant improvement in feature prediction when compared to LSM models which did not account for these effects. The model presented also showed significant improvement over previous semi-computational and cellular automaton based models which accounted for these effects, either in terms of feature prediction or in execution time.
14. The work of Chapter 5 showed that the mask edge effect is not a significant contributor to the overall second strike effect; thus, in the majority of cases, it can be ignored.
15. The proposed LSM-based feature predictive models can prove to be practical assistive tools during the micro-fabrication of complex MEMS and microfluidic devices using AJM.

6.4 Future work

The general LSM-based model presented in Chapter 5 can be improved by addressing the limitation outlined in Section 6.2. Firstly, it can be extended to model high flux cases by incorporating the analytical model of particle interference effects developed in [34]. However, since the model of [34] was developed for divergent erosive jets incident to a flat target surface at normal incidence only, it would have to be generalized to work for cases where the jet is incident to a non-flat, i.e. evolving, target surface at any incidence, as in the present study. Secondly, the model can be extended to account for particle embedding effects in ductile targets by incorporating the model of [31] where a relation for a net embedding energy flux as a function of scanning direction distance and angle of incidence was used. This would enable the modelling of surface evolution in AJM of a wider class of polymeric targets where these effects are significant. Thirdly, the first-order approximation in modelling the mask edge could be improved by extending the formulation of Section 4.3.2.2 to work for cases where the mask edge slope is non-linear. Furthermore, spatial hindering effects could be modelled by extending the formulation of Section 4.3.2.1 for the masking function for the target by assuming that the width of the udder shape is analogous to the mask opening width. This would ensure that particles only as wide as the udder shape itself and smaller can reach the bottom of the profile. If necessary, the model could be extended to 3D by extending the size of arrays and gradient operators for the level set function. This would also require the development of a new 3D formulation to model the erosive efficacy, masking function and the grid, more difficult techniques in visualizing the evolving surface, and more efficient use of computational resources, i.e. by taking advantage of parallel computing to maintain computational efficiency.

The model in its present form could be made more computationally efficient by minimizing the frequency of re-initialization, FR, of the level set function, while still ensuring that eq. (3.12) remains valid (see Section 3.3.3.1). This can be achieved through numerical convergence studies or the introduction of a metric, although this would make the approach more complicated and less accurate, when compared to the present approach where re-initializaton is performed at each time step to ensure the highest degree of accuracy. As a final note, the present model can be used to predict more intricate shapes that can be sculpted for a desired application through superposition of individual features made using AJM, i.e. the resulting surface shape from the initial simulation can be used as an input for the next one, etc. Such sculpted features could be abrasive jet micromachined by using a combination of different nozzle passes, mask patterns, mask dimensions, and jet parameters, e.g. scan speed, angle of incidence, standoff distance, particle flux, velocity, shape and size. This could be considered for future work.

Appendix A Abrasive Mass Flow Rate Measurements for AJM Experiments

In the analysis, the mass flow rate fluctuation over the course of the machining experiment, originally described in Section 2.4.1.1, was deemed to be significant if the percentage change in average $\dot{M}$ before and after the experiment was more than 10%. In such cases, the mass flow rate was modelled as a linearly decreasing function of time t (cases in Tables A-3-A-6, A-9 and A-10). Otherwise, the total average $\dot{M}$ was used (cases in Tables A-1, A-2, A-7, A-8 and A-11-A-16).

For the cases in Tables A-3-A-6, A-9 and A-10, the linear $\dot{M}$ function was obtained using the average $\dot{M}$ before and after the experiment and the total machining time for the respective cases, i.e. the maximum number of nozzle passes multiplied by $2r_s/v_t$ (Section 2.3.2), since all such cases corresponded to the machining of channels. For all these cases, the maximum number of nozzle passes was 30, as indicated in Sections 2.4.1.1 and 3.4.1. From Sections 2.3.2 and 2.3.4.3, r_s can be obtained as $r_s = h\sqrt{-\ln(0.001)}/\beta$, where h = 20 mm from Sections 2.2 and 3.2 and β = 15 from Sections 2.4.1.1 and 3.4.1. Using scanning speeds v_t of 1 mm s^{-1} (glass, Tables A-3 and A-9) and 0.5 mm s^{-1} (PMMA, Tables A-4-A-6 and A-10) from Sections 2.2 and 3.2, the resulting total machining times were obtained approximately as 210 s and 420 s, respectively. The resulting linear $\dot{M}$ functions are listed in Tables A-3-A-6, A-9 and A-10.

Table A-1. Abrasive mass flow rate measurements corresponding to Figure 2.5 and Sections 2.2 and 2.4.1.1.

	Trial			$\dot{M}$ avg. (stand. dev.) (g min^{-1})
	1	2	3	
$\dot{M}$ before experiment (g min^{-1})	3.41	3.32	3.35	3.36 (0.04)
$\dot{M}$ after experiment (g min^{-1})	3.20	3.29	3.23	3.24 (0.04)
$\dot{M}$ total avg. (total stand. dev.) (g min^{-1})				3.30 (0.07)
Change in $\dot{M}$ avg. before/after experiment (%)				3.6 (< 10)

Table A-2. Abrasive mass flow rate measurements corresponding to Figure 2.6 and Sections 2.2 and 2.4.1.1.

	Trial			$\dot{M}$ avg. (stand. dev.) (g min^{-1})
	1	2	3	
$\dot{M}$ before experiment (g min^{-1})	3.37	3.29	3.30	3.32 (0.04)
$\dot{M}$ after experiment (g min^{-1})	3.27	3.31	3.26	3.28 (0.02)
$\dot{M}$ total avg. (total stand. dev.) (g min^{-1})				3.30 (0.04)
Change in $\dot{M}$ avg. before/after experiment (%)				1.2 (< 10)

Table A-3. Abrasive mass flow rate measurements and resulting linear function corresponding to Figure 2.7 and Sections 2.2 and 2.4.1.1.

	Trial			$\dot{M}$ avg. (stand. dev.) (g min^{-1})
	1	2	3	
$\dot{M}$ before experiment (g min^{-1})	2.27	2.46	2.56	2.43 (0.12)
$\dot{M}$ after experiment (g min^{-1})	2.09	2.12	2.33	2.18 (0.11)
$\dot{M}$ total avg. (total stand. dev.) (g min^{-1})				2.31 (0.17)
Change in $\dot{M}$ avg. before/after experiment (%)				10.3 (> 10)
Resulting $\dot{M}$ function (g min^{-1})				$2.43-1.19 \times 10^{-3}t$(s)

Table A-4. Abrasive mass flow rate measurements and resulting linear function corresponding to Figure 2.8 and Sections 2.2 and 2.4.1.1.

	Trial			$\dot{M}$ avg. (stand. dev.) (g min^{-1})
	1	2	3	
$\dot{M}$ before experiment (g min^{-1})	4.17	3.96	3.78	3.97 (0.16)
$\dot{M}$ after experiment (g min^{-1})	2.98	3.21	3.13	3.11 (0.10)
$\dot{M}$ total avg. (total stand. dev.) (g min^{-1})				3.54 (0.45)
Change in $\dot{M}$ avg. before/after experiment (%)				21.7 (> 10)
Resulting $\dot{M}$ function (g min^{-1})				3.97-2.05 x 10^{-3}t(s)

Table A-5. Abrasive mass flow rate measurements and resulting linear function corresponding to Figure 2.9 and Sections 2.2 and 2.4.1.1.

	Trial			$\dot{M}$ avg. (stand. dev.) (g min^{-1})
	1	2	3	
$\dot{M}$ before experiment (g min^{-1})	3.77	4.05	3.82	3.88 (0.12)
$\dot{M}$ after experiment (g min^{-1})	2.21	2.53	2.52	2.42 (0.15)
$\dot{M}$ total avg. (total stand. dev.) (g min^{-1})				3.15 (0.74)
Change in $\dot{M}$ avg. before/after experiment (%)				37.6 (> 10)
Resulting $\dot{M}$ function (g min^{-1})				3.88-3.47 x 10^{-3}t(s)

Table A-6. Abrasive mass flow rate measurements and resulting linear function corresponding to Figure 2.10 and Sections 2.2 and 2.4.1.1.

	Trial			$\dot{M}$ avg. (stand. dev.) (g min^{-1})
	1	2	3	
$\dot{M}$ before experiment (g min^{-1})	2.77	2.43	2.78	2.66 (0.16)
$\dot{M}$ after experiment (g min^{-1})	2.25	2.31	1.92	2.16 (0.17)
$\dot{M}$ total avg. (total stand. dev.) (g min^{-1})				2.41 (0.30)
Change in $\dot{M}$ avg. before/after experiment (%)				18.8 (> 10)
Resulting $\dot{M}$ function (g min^{-1})				2.66-1.19 x $10^{-3}t$(s)

Table A-7. Abrasive mass flow rate measurements corresponding to Figure 2.11 and Sections 2.2 and 2.4.1.1.

	Trial			$\dot{M}$ avg. (stand. dev.) (g min^{-1})
	1	2	3	
$\dot{M}$ before experiment (g min^{-1})	2.63	2.81	2.69	2.71 (0.07)
$\dot{M}$ after experiment (g min^{-1})	2.47	2.58	2.60	2.55 (0.06)
$\dot{M}$ total avg. (total stand. dev.) (g min^{-1})				2.63 (0.10)
Change in $\dot{M}$ avg. before/after experiment (%)				5.9 (< 10)

Table A-8. Abrasive mass flow rate measurements corresponding to Figure 2.12 and Sections 2.2 and 2.4.1.1.

	Trial			$\dot{M}$ avg. (stand. dev.) (g min^{-1})
	1	2	3	
$\dot{M}$ before experiment (g min^{-1})	5.31	5.55	5.61	5.49 (0.13)
$\dot{M}$ after experiment (g min^{-1})	5.29	5.19	5.33	5.27 (0.06)
$\dot{M}$ total avg. (total stand. dev.) (g min^{-1})				5.38 (0.15)
Change in $\dot{M}$ avg. before/after experiment (%)				4.0 (< 10)

Table A-9. Abrasive mass flow rate measurements and resulting linear function corresponding to Figures 3.7 and 3.9, Table 3.1, and Sections 3.2 and 3.4.1.

	Trial			$\dot{M}$ avg. (stand. dev.) (g min^{-1})
	1	2	3	
$\dot{M}$ before experiment (g min^{-1})	2.78	2.49	2.83	2.70 (0.15)
$\dot{M}$ after experiment (g min^{-1})	0.58	0.91	0.52	0.67 (0.17)
$\dot{M}$ total avg. (total stand. dev.) (g min^{-1})				1.69 (1.03)
Change in $\dot{M}$ avg. before/after experiment (%)				75.2 (> 10)
Resulting $\dot{M}$ function (g min^{-1})				2.70-9.65 x $10^{-3}t$(s)

Table A-10. Abrasive mass flow rate measurements and resulting linear function corresponding to Figure 3.8, Table 3.1, and Sections 3.2 and 3.4.1.

	Trial			$\dot{M}$ avg. (stand. dev.) (g min^{-1})
	1	2	3	
$\dot{M}$ before experiment (g min^{-1})	1.71	1.99	1.88	1.86 (0.12)
$\dot{M}$ after experiment (g min^{-1})	1.31	1.12	1.33	1.25 (0.09)
$\dot{M}$ total avg. (total stand. dev.) (g min^{-1})				1.56 (0.32)
Change in $\dot{M}$ avg. before/after experiment (%)				32.8 (> 10)
Resulting $\dot{M}$ function (g min^{-1})				1.86-1.44 x $10^{-3}t$(s)

Table A-11. Abrasive mass flow rate measurements corresponding to Figures 4.7 and 4.8, Table 4.1, and Sections 4.2 and 4.4.1.

	Trial			$\dot{M}$ avg. (stand. dev.) (g min^{-1})
	1	2	3	
$\dot{M}$ before experiment (g min^{-1})	2.70	2.12	3.01	2.61 (0.37)
$\dot{M}$ after experiment (g min^{-1})	2.08	2.62	2.83	2.51 (0.32)
$\dot{M}$ total avg. (total stand. dev.) (g min^{-1})				2.56 (0.35)
Change in $\dot{M}$ avg. before/after experiment (%)				3.8 (< 10)

Table A-12. Abrasive mass flow rate measurements corresponding to Figure 4.9, Table 4.1, and Sections 4.2 and 4.4.1.

	Trial			$\dot{M}$ avg. (stand. dev.) (g min^{-1})
	1	2	3	
$\dot{M}$ before experiment (g min^{-1})	3.84	3.82	3.86	3.84 (0.02)
$\dot{M}$ after experiment (g min^{-1})	3.85	3.84	3.83	3.84 (0.01)
$\dot{M}$ total avg. (total stand. dev.) (g min^{-1})				3.84 (0.01)
Change in $\dot{M}$ avg. before/after experiment (%)				0 (< 10)

Table A-13. Abrasive mass flow rate measurements corresponding to Figure 4.10 ($x \leq 0$), Table 4.1, and Sections 4.2 and 4.4.1.

	Trial			$\dot{M}$ avg. (stand. dev.) (g min^{-1})
	1	2	3	
$\dot{M}$ before experiment (g min^{-1})	2.81	3.39	3.16	3.12 (0.24)
$\dot{M}$ after experiment (g min^{-1})	3.17	3.03	4.06	3.42 (0.46)
$\dot{M}$ total avg. (total stand. dev.) (g min^{-1})				3.27 (0.39)
Change in $\dot{M}$ avg. before/after experiment (%)				9.6 (< 10)

Table A-14. Abrasive mass flow rate measurements corresponding to Figure 4.10 ($x \geq 0$), Table 4.1, and Sections 4.2 and 4.4.1.

	Trial			$\dot{M}$ avg. (stand. dev.) (g min^{-1})
	1	2	3	
$\dot{M}$ before experiment (g min^{-1})	2.13	2.67	2.25	2.35 (0.23)
$\dot{M}$ after experiment (g min^{-1})	2.05	1.79	2.52	2.12 (0.30)
$\dot{M}$ total avg. (total stand. dev.) (g min^{-1})				2.24 (0.29)
Change in $\dot{M}$ avg. before/after experiment (%)				9.8 (< 10)

Table A-15. Abrasive mass flow rate measurements corresponding to Figure 4.11 ($x \leq 0$), Table 4.1, and Sections 4.2 and 4.4.1.

	Trial			$\dot{M}$ avg. (stand. dev.) (g min^{-1})
	1	2	3	
$\dot{M}$ before experiment (g min^{-1})	2.97	3.97	3.86	3.60 (0.45)
$\dot{M}$ after experiment (g min^{-1})	3.07	3.68	4.05	3.60 (0.40)
$\dot{M}$ total avg. (total stand. dev.) (g min^{-1})				3.60 (0.43)
Change in $\dot{M}$ avg. before/after experiment (%)				0 (< 10)

Table A-16. Abrasive mass flow rate measurements corresponding to Figure 4.11 ($x \geq 0$), Table 4.1, and Sections 4.2 and 4.4.1.

	Trial			$\dot{M}$ avg. (stand. dev.) (g min^{-1})
	1	2	3	
$\dot{M}$ before experiment (g min^{-1})	3.45	3.62	3.55	3.54 (0.07)
$\dot{M}$ after experiment (g min^{-1})	3.56	3.58	3.60	3.58 (0.02)
$\dot{M}$ total avg. (total stand. dev.) (g min^{-1})				3.56 (0.05)
Change in $\dot{M}$ avg. before/after experiment (%)				1.1 (< 10)

Appendix B Example Program for the Case Corresponding to Figure 5.3 (Case 2) in Section 5.3.1

```
%2D LLLF LSM with Curvature, F Extended, Narrow Band
%AJM of Masked Glass Channels with Mask Wear, Second Strike
clc
clear all
tic; %Start timer
%______________________________________________________________________
%Target (T) Properties
%Input Constants of Target (all in metric units)
MFR =2.63/60000;          %MFR (kg/s)
C=8.0e-6; %T              %Empirical constant (m/s)^-k_vel
H_slp=4.92;               %Velocity distribution slope
beta=15;                  %Focus coefficient
v_o=162;                  %v(0) (m/s)
v_scan=0.001;             %Scan Velocity (m/s)
rho_s=2200; %T            %Target surface mass density (kg/m^3)
k_vel=1.43; %T            %Velocity exponent
alfa=90*pi/180;           %Angle of incidence (rad)
epsilon=8e-6;             %Curvature coefficient - NOTE has to be MAX if have epsilon for T & M (SEE CFL condition)
h=0.02;                   %Standoff distance (m)
W_m=400e-6;               %Mask width (m)
H_m=100e-6;               %Mask height (m)
%______________________________________________________________________
%Mask (M) Properties
C_M=7.9e-8;               %Empirical constant (Mask)(m/s)^-k_vel_M
rho_s_M=7712;             %Mass density (Mask) (kg/m^3)
k_vel_M=1.73;             %Velocity exponent (Mask)
H_vic_M=5.67;             %Vicker's hardness (Mask) (GPa)
n_1_M=0.69;               %Ductile Erosion constants (Mask)
n_2_M=1.399;
leng_M_L=150e-6;          %Mask length on left side (m)
leng_M_R=150e-6;          %Mask length on right side (m)
%______________________________________________________________________
%Other Inputs
pas_des=10; %Desired number of passes
adj_r_s=1.08;  %r_s (radius of impact area of jet on surface; see r_s) fit, i.e. Tpass_T (the time it takes to reach initial profile depth; see t_in) - applied here for code simplicity
Num_iter=200000;  %Used for initial time step dt=tin/Num_iter (see below)
adj_r_s_M=1.71; %T_pass_M fit (Mask); See adj_r_s
C_M=C_M*(adj_r_s_M/adj_r_s); %Equivalent to Hamiltonian_M*(T_pass_M/T_pass_T) since C_M in Hamiltonian_M - applied here for code simplicity
z_in=850e-6;  %Maximum expected feature(target) depth (m) (ensure large enough to account for upper band size BS)
z_air=50e-6;  %Vertical distance above mask (m); (ensure large enough to account for lower band size)
i_max=105;  %Grid size, x
k_max=151;  %Grid size, z
BS_L=2; %Define LOWER band size (Integer), in multiples of dz (need 1 grid pt. clearance from end of grid)
BS_U=4; %Define UPPER band size (Integer), in multiples of dz (need 1 grid pt. clearance from end of grid)
MT_pt_dist=3;  %Define zTadj distance, in multiples of dz
%______________________________________________________________________
%Specify output name of excel file
if (alfa==90*pi/180)
    fil_name='G_2DMOb_FxNB_MWr_2S_90FGch.xls';
end
if (alfa==60*pi/180)
    fil_name='G_2DMOb_FxNB_MWr_2S_60FGch.xls';
end
if (alfa==45*pi/180)
    fil_name='G_2DMOb_FxNB_MWr_2S_45FGch.xls';
end
if (alfa==30*pi/180)
    fil_name='G_2DMOb_FxNB_MWr_2S_30FGch.xls';
end
%______________________________________________________________________
%Calculate Spread angle (rad), assume 99.9% particles from r=0 to
%(r+dr)_max and radius of impact area (r_s) of jet on surface in scan direction
fi=atan(sqrt(-log(0.001))/beta);
r_s=adj_r_s*h*tan(fi); %Adjusted r_s to obtain Tpass_T - applied here for code simplicity
y_mean=0.3536*(r_s/adj_r_s); %Mean y used in stationary erosive power/efficacy (see below) simulating Avg. scanning erosive power/efficacy
%______________________________________________________________________
%Calculate input 'blasting time'
pass_spec=pas_des;
t_in=pass_spec*(2*r_s)/v_scan;
T_pass_T=(2*r_s)/v_scan; %Tpass_T
T_pass_M=(2*r_s)/v_scan*(adj_r_s_M/adj_r_s); %Tpass_M
%______________________________________________________________________
```

```
%Calculate minimum and maximum x & z grid limits
x_off=W_m/2+leng_M_L; %Offset distance between global and nozzle axes

if (alfa==90*pi/180) %Need this condition since Matlab creates error for cos(90)
x_min_grid=0;
x_max_grid=W_m/2+leng_M_R+x_off;
else
x_min_grid=h*cos(alfa);
x_max_grid=h*cos(alfa)+W_m/2+leng_M_R+x_off;
end

z_min_grid=h*sin(alfa)-(H_m+z_air);
z_max_grid=h*sin(alfa)+z_in;

%Define grid spacing
dx=(x_max_grid-x_min_grid)/(i_max-1);
dz=(z_max_grid-z_min_grid)/(k_max-1);
%_______________________________________________________________________
Crit_D=dz/2; %Define critical distance that must be achieved before bands are rebuilt
%_______________________________________________________________________
%Define x and z coordinates
x_cord=zeros(1,i_max);
for i=1:1:i_max
   x_cord(i)=(i-1).*dx+x_min_grid;
end
%Local x, nozzle origin
x_cord_local=zeros(1,i_max);
for i=1:1:i_max
x_cord_local(i)=x_cord(i)-x_off;
end
z_cord=zeros(k_max,1);
for k=1:1:k_max
z_cord(k)=(k_max-k).*dz+z_min_grid;
end
%_______________________________________________________________________
%INITIALIZATION of phi (level set function)

%Create arrays for initial surface, inc. mask
r_UCt=14e-6;    %Mask radius

%Top mask surface (left and right)
z_rTL=h*sin(alfa)-H_m+r_UCt; %Centre pos'n of radial circle
x_rTL=x_min_grid+leng_M_L-r_UCt;
i_max_1L=1+ceil(leng_M_L/dx);
dx_1L=leng_M_L/(i_max_1L-1);
xz_surf_ini_1L=zeros(i_max_1L,2);
for i_1L=1:1:i_max_1L
   xz_surf_ini_1L(i_1L,1)=(i_1L-1)*dx_1L+x_min_grid;
   if (xz_surf_ini_1L(i_1L,1)<x_rTL)
      xz_surf_ini_1L(i_1L,2)=h*sin(alfa)-H_m;
   else
      xz_surf_ini_1L(i_1L,2)=real(-sqrt(r_UCt^2-(xz_surf_ini_1L(i_1L,1)-x_rTL).^2)+z_rTL);
   end
end

z_rTR=h*sin(alfa)-H_m+r_UCt; %Centre pos'n of radial circle
x_rTR=x_max_grid-leng_M_R+r_UCt;
i_max_1R=1+ceil(leng_M_R/dx);
dx_1R=leng_M_R/(i_max_1R-1);
xz_surf_ini_1R=zeros(i_max_1R,2);
for i_1R=1:1:i_max_1R
   xz_surf_ini_1R(i_1R,1)=(i_1R-1)*dx_1R+(x_max_grid-leng_M_R);
   if (xz_surf_ini_1R(i_1R,1)>x_rTR)
      xz_surf_ini_1R(i_1R,2)=h*sin(alfa)-H_m;
   else
      xz_surf_ini_1R(i_1R,2)=real(-sqrt(r_UCt^2-(xz_surf_ini_1R(i_1R,1)-x_rTR).^2)+z_rTR);
   end
end
%Target surface
i_max_2=1+ceil((W_m)/dx);
dx_2=(W_m)/(i_max_2-1);
xz_surf_ini_2=zeros(i_max_2,2);
for i_2=1:1:i_max_2
   xz_surf_ini_2(i_2,1)=(i_2-1)*dx_2+(x_min_grid+leng_M_L);
   xz_surf_ini_2(i_2,2)=h*sin(alfa);
end
%Vertical Mask edge surface (left and right)
k_max_3=1+ceil((H_m-r_UCt)/dz);
dz_3=(H_m-r_UCt)/(k_max_3-1);
xz_surf_ini_3=zeros(k_max_3,2);
```

```
for k_3=1:1:k_max_3
    xz_surf_ini_3(k_3,1)=x_min_grid+leng_M_L;
    xz_surf_ini_3(k_3,2)=(k_3-1)*dz_3+(h*sin(alfa)-H_m+r_UCt);
end

k_max_4=k_max_3;
dz_4=dz_3;
xz_surf_ini_4=zeros(k_max_4,2);
for k_4=1:1:k_max_4
    xz_surf_ini_4(k_4,1)=x_max_grid-leng_M_R;
    xz_surf_ini_4(k_4,2)=(k_4-1)*dz_4+(h*sin(alfa)-H_m+r_UCt);
end

%Combine all into 1 array
xz_surf_ini=[xz_surf_ini_1L;xz_surf_ini_1R;xz_surf_ini_2;xz_surf_ini_3;xz_surf_ini_4];

%Calculate SDF and hence initialize phi
b_max_ini=i_max_1L+i_max_1R+i_max_2+k_max_3+k_max_4;
SDF_ini=zeros(b_max_ini,1);
phi=zeros(k_max,i_max);
for k=1:1:k_max
    for i=1:1:i_max
        for b_ini=1:1:b_max_ini
            SDF_ini(b_ini)=((x_cord(i)-xz_surf_ini(b_ini,1)).^2+(z_cord(k)-xz_surf_ini(b_ini,2)).^2).^0.5;
        end

        min_SDF_ini=min(SDF_ini);

        %Assign signage (+ or -)
        if (min_SDF_ini==0)
            phi(k,i)=0;
        else
           if (z_cord(k)<(h*sin(alfa)-H_m))
              phi(k,i)=-min_SDF_ini;
           elseif (z_cord(k)>(h*sin(alfa)))
              phi(k,i)=min_SDF_ini;
           else %if (h*sin(alfa)-H_m) <= z <= h*sin(alfa)

               if ((x_cord(i)>=x_min_grid)&&(x_cord(i)<(x_min_grid+leng_M_L-r_UCt)))||...
                       ((x_cord(i)>(x_max_grid-leng_M_R+r_UCt))&&(x_cord(i)<=x_max_grid))
                  phi(k,i)=min_SDF_ini;
               elseif ((x_cord(i)>(x_min_grid+leng_M_L))&&(x_cord(i)<(x_max_grid-leng_M_R))) %if x_min_grid+leng_M_L < x < x_max_grid-leng_M_R
                  phi(k,i)=-min_SDF_ini;
               elseif ((x_cord(i)>=x_min_grid+leng_M_L-r_UCt)&&(x_cord(i)<=(x_min_grid+leng_M_L)))

                     if (z_cord(k)>(h*sin(alfa)-H_m+r_UCt))&&(z_cord(k)<(h*sin(alfa)))
                        phi(k,i)=min_SDF_ini;
                     else
                         if (((x_cord(i)-x_rTL)^2+(z_cord(k)-z_rTL)^2)>=r_UCt^2)
                             phi(k,i)=-min_SDF_ini;
                         else
                             phi(k,i)=min_SDF_ini;
                         end
                     end

               else

                     if (z_cord(k)>(h*sin(alfa)-H_m+r_UCt))&&(z_cord(k)<(h*sin(alfa)))
                        phi(k,i)=min_SDF_ini;
                     else
                         if (((x_cord(i)-x_rTR)^2+(z_cord(k)-z_rTR)^2)>=r_UCt^2)
                             phi(k,i)=-min_SDF_ini;
                         else
                             phi(k,i)=min_SDF_ini;
                         end
                     end

               end
           end
        end

    end
end
%________________________________________________________________________
%Define flags for M and T
flag_T_M=zeros(k_max,i_max);
for k=1:1:k_max
    for i=1:1:i_max
       if (z_cord(k)<(h*sin(alfa)))
           flag_T_M(k,i)=2; %M
        else
            flag_T_M(k,i)=1; %T
        end
    end
end
%_____________________________________________________________________________
%Define (initial) mask "Visibility", Ratio of what flux "sees" at
%initial surface w.r.t mask opening
x_m=H_m*(h*cos(alfa)-W_m/2)/(h*sin(alfa)-H_m); %(Initial) Mask shadow at top of surface

if (x_m<=0)         %i.e., when (alfa+fi_min)>= 90 deg
    Visibility=1;
elseif (x_m>=W_m)   %i.e., when shadow is larger than mask opening
    Visibility=0;
else
    Visibility=(W_m-x_m)/W_m;
end
%_____________________________________________________________________________
%Adjustment to Mass Flux due to Mask Model for Target M(x') for initial iteration

%Calculate visibility angles based on zero level set (M)
x_prime_surf_LM=zeros(b_max_ini,1); %Initial surface for M in local coordinates
z_prime_surf_LM=zeros(b_max_ini,1);
x_prime_surf_RM=zeros(b_max_ini,1);
z_prime_surf_RM=zeros(b_max_ini,1);
for b_ini=1:1:b_max_ini
    if ((xz_surf_ini(b_ini,1)>=x_min_grid)&&(xz_surf_ini(b_ini,1)<=(x_min_grid+leng_M_L)))&&...
            ((xz_surf_ini(b_ini,2)>=(h*sin(alfa)-H_m))&&(xz_surf_ini(b_ini,2)<h*sin(alfa))) %This check also ensures that if surf was not encountered
        % (i.e. xz surfs are 0) then it will ignore those values (this will occur below after interpolation)
        x_prime_surf_LM(b_ini)=(xz_surf_ini(b_ini,1)-x_off).*sin(alfa)-xz_surf_ini(b_ini,2).*cos(alfa); %Rotated local x
        z_prime_surf_LM(b_ini)=(xz_surf_ini(b_ini,1)-x_off).*cos(alfa)+xz_surf_ini(b_ini,2).*sin(alfa); %Rotated local z
    else
        x_prime_surf_LM(b_ini)=NaN;
        z_prime_surf_LM(b_ini)=NaN;
    end

    if ((xz_surf_ini(b_ini,1)>=(x_max_grid-leng_M_R))&&(xz_surf_ini(b_ini,1)<=x_max_grid))&&...
            ((xz_surf_ini(b_ini,2)>=(h*sin(alfa)-H_m))&&(xz_surf_ini(b_ini,2)<h*sin(alfa)))
        x_prime_surf_RM(b_ini)=(xz_surf_ini(b_ini,1)-x_off).*sin(alfa)-xz_surf_ini(b_ini,2).*cos(alfa); %Rotated local x
        z_prime_surf_RM(b_ini)=(xz_surf_ini(b_ini,1)-x_off).*cos(alfa)+xz_surf_ini(b_ini,2).*sin(alfa); %Rotated local z
    else
        x_prime_surf_RM(b_ini)=NaN;
        z_prime_surf_RM(b_ini)=NaN;
    end

end

%By evaluating Max of x_prime_surf_LM, we can check if any entries are
%positive, ignoring NaN's; if any are, then we have case (b),
%where mask shadow is >= W_m/2 from left mask edge, else have case (a) and
%(c), mask shadow is < W_m/2
[max_x_prime_surf_LM,I_max_LM]=max(x_prime_surf_LM); %Will ignore NaN's

%Min tan of left 'spread' angle defined by mask
if (max_x_prime_surf_LM>=0) %Case (b)
    tan_fi_min=max_x_prime_surf_LM/z_prime_surf_LM(I_max_LM);
else %Case (a) and (c)
    %Find min |x_prime_surf_LM|
    [min_x_prime_surf_LM,I_min_LM]=min(abs(x_prime_surf_LM));

    if (alfa==90*pi/180)
        tan_fi_min=min_x_prime_surf_LM/h;
    else
        tan_fi_min=min_x_prime_surf_LM/z_prime_surf_LM(I_min_LM);
    end
end

%Find min x_prime_surf_RM
[min_x_prime_surf_RM,I_min_RM]=min(x_prime_surf_RM);
%Max tan of right 'spread' angle defined by mask
if (alfa==90*pi/180)
    tan_fi_max=min_x_prime_surf_RM/h;
else
    tan_fi_max=min_x_prime_surf_RM/z_prime_surf_RM(I_min_RM);
end

%Define more Mask "Visibility" Parameters
x_prime=zeros(k_max,i_max);
z_prime=zeros(k_max,i_max);
for k=1:1:k_max
```

```
    for i=1:1:i_max
      x_prime(k,i)=x_cord_local(i).*sin(alfa)-z_cord(k).*cos(alfa); %Rotated local x
      z_prime(k,i)=x_cord_local(i).*cos(alfa)+z_cord(k).*sin(alfa); %Rotated local z
    end
end

%Define Parameters for particle size distribution (mean and standard
%deviation)
mu_l=-11.615;
sigma_l=0.5;

%Constants for M(x') closed form fit (T)
P_1=0.11338616706948672477e-14;
P_2=-1.4142135623730950488;
P_3=15.365430355183677705;

%Define Left/Right rotated local x limit for mask (L)
L_mask=zeros(k_max,i_max);
x_lim=zeros(k_max,i_max); %Lower x_prime L_mask limit when x_m>=W_m/2
Int_P_r_x_prime=zeros(k_max,i_max);
Int_P_r_L_mask=zeros(k_max,i_max);
M_r_x_prime=zeros(k_max,i_max); %Initial masking function
for k=1:1:k_max
    for i=1:1:i_max

      if flag_T_M(k,i)==1 %T

      if (max_x_prime_surf_LM<0) %Case (a) and (c)
        if (x_prime(k,i)<0)
          L_mask(k,i)=z_prime(k,i).*tan_fi_min;
        else %i.e., when x_prime>=0
          L_mask(k,i)=z_prime(k,i).*tan_fi_max;
        end
      else %i.e., when x_m>=W_m/2 Case (b)
        x_lim(k,i)=z_prime(k,i).*tan_fi_min;
        if (x_prime(k,i)<x_lim(k,i))
          L_mask(k,i)=0;
        else %i.e., when x_prime>=x_lim
          L_mask(k,i)=z_prime(k,i).*tan_fi_max-x_lim(k,i);
        end
      end

      %Define proportion of mass of particles that pass through mask opening having a
      %specific particle size (of radius r) distribution
      if (abs(x_prime(k,i))>=L_mask(k,i))
        M_r_x_prime(k,i)=0;
      else
        Int_P_r_x_prime(k,i)=real(P_1-P_1*erf(P_2*log(L_mask(k,i)-abs(x_prime(k,i)))-P_3));
        Int_P_r_L_mask(k,i)=real(P_1-P_1*erf(P_2*log(L_mask(k,i))-P_3));
        M_r_x_prime(k,i)=Int_P_r_x_prime(k,i)./Int_P_r_L_mask(k,i);
      end

      else  %M (for initial iteration only over very small dt) %Below, calculate this using theta to check visibility for M and M(x') for edge


       if (alfa==90*pi/180) %90deg case
          M_r_x_prime(k,i)=1;
       else

        %Spread angle covering range not seen by nozzle due to mask
        tan_fi_M=abs(((x_min_grid+leng_M_L)-x_off).*sin(alfa)-h*sin(alfa).*cos(alfa))/...
          abs(((x_min_grid+leng_M_L)-x_off).*cos(alfa)+h*sin(alfa).*sin(alfa));

        if (max_x_prime_surf_LM<0) %Case (a) and (c)
          if (((z_prime(k,i)*tan_fi_min<abs(x_prime(k,i)))&&(z_prime(k,i)*tan_fi_M>abs(x_prime(k,i))))&&...
              (x_cord(i)>=(x_min_grid+leng_M_L))&&(x_prime(k,i)<0))
            M_r_x_prime(k,i)=0;
          else
            M_r_x_prime(k,i)=1;
          end


        else %Case (b)
          if ((((-z_prime(k,i)*tan_fi_M)<x_prime(k,i))&&((z_prime(k,i)*tan_fi_min)>x_prime(k,i)))&&...
              (x_cord(i)>=(x_min_grid+leng_M_L)))
            M_r_x_prime(k,i)=0;
          else
            M_r_x_prime(k,i)=1;
          end
```

```
            end

           end


         end

      end
end
%___________________________________________________________________________
%Initialize time and counters
time=0;
counter=0;
No_RE=0; %Initialize band re-initialization counter

%###########################################################################
%INITIAL ITERATION (W/O F_EXT & NB) TO CALCULATE/INITIALIZE F_EXT FOR WHILE
%LOOP;
%###########################################################################

%___________________________________________________________________________
%USE STATIONARY APPROACH
%Define velocity v(x,z) at each grid node
v=zeros(k_max,i_max);
   for k=1:1:k_max
      for i=1:1:i_max
            v(k,i)=v_o*(1-H_slp*((x_cord_local(i)*sin(alfa)-z_cord(k)*cos(alfa)).^2+(y_mean).^2).^0.5./(x_cord_local(i)*cos(alfa)+z_cord(k)*sin(alfa)));
         if (v(k,i)<0)
            v(k,i)=0;
         end
      end
   end

%Define particle mass flux(x,z) at each grid node
 flux=zeros(k_max,i_max);
   for k=1:1:k_max
      for i=1:1:i_max
            flux(k,i)=(MFR/pi)*(beta./(x_cord_local(i)*cos(alfa)+z_cord(k)*sin(alfa))).^2....
            *exp(-(beta^2.*((x_cord_local(i)*sin(alfa)-z_cord(k)*cos(alfa)).^2+(y_mean).^2)./(x_cord_local(i)*cos(alfa)+z_cord(k)*sin(alfa)).^2));
      end
   end

%Define Erosive Power Eros_pow(k,i) at each grid node
Eros_pow=zeros(k_max,i_max);
   for k=1:1:k_max
      for i=1:1:i_max
         if flag_T_M(k,i)==1 %T
            Eros_pow(k,i)=M_r_x_prime(k,i).*v(k,i).^k_vel.*flux(k,i);
         else %M
            Eros_pow(k,i)=M_r_x_prime(k,i).*v(k,i).^k_vel_M.*flux(k,i);
         end
      end
   end
%___________________________________________________________________________
%Define FD's and BC's
%Initialization (preallocation) to increase computational speed
phi_x_pos=zeros(k_max,i_max);
phi_x_neg=zeros(k_max,i_max);
phi_x_cen=zeros(k_max,i_max);
phi_x_x_cen=zeros(k_max,i_max);
phi_z_pos=zeros(k_max,i_max);
phi_z_neg=zeros(k_max,i_max);
phi_z_cen=zeros(k_max,i_max);
phi_z_z_cen=zeros(k_max,i_max);
phi_x_z_cen=zeros(k_max,i_max);
for k=1:1:k_max
      for i=1:1:i_max
            if i==i_max
               phi_x_pos(k,i)=0;
            else
               phi_x_pos(k,i)=(phi(k,i+1)-phi(k,i))./dx;
            end

            if i==1
               phi_x_neg(k,i)=0;
            else
               phi_x_neg(k,i)=(phi(k,i)-phi(k,i-1))./dx;
            end

            if (i==i_max)||(i==1)
```

```
                phi_x_cen(k,i)=0;
                phi_x_x_cen(k,i)=0;
            else
                phi_x_cen(k,i)=(phi(k,i+1)-phi(k,i-1))./(2*dx);
                phi_x_x_cen(k,i)=(phi(k,i+1)-2*phi(k,i)+phi(k,i-1))./(dx^2);
            end
            %Note direction (k+1 is k-1) and
            %BC (k==k_max is k==1)) since z is +ve 'downward'
            if k==1
                phi_z_pos(k,i)=0;
            else
                phi_z_pos(k,i)=(phi(k-1,i)-phi(k,i))./dz;
            end
            %Note direction (k-1 is k+1) and
            %BC (k==1 is k==k_max)
            if k==k_max
                phi_z_neg(k,i)=0;
            else
                phi_z_neg(k,i)=(phi(k,i)-phi(k+1,i))./dz;
            end
            %Note direction (k+1 is k-1)
            if (k==k_max)||(k==1)
                phi_z_cen(k,i)=0;
                phi_z_z_cen(k,i)=0;
            else
                phi_z_cen(k,i)=(phi(k-1,i)-phi(k+1,i))./(2*dz);
                phi_z_z_cen(k,i)=(phi(k-1,i)-2*phi(k,i)+phi(k+1,i))./(dz^2);
            end
        end
end
%___________________________________________________________________________
%Define Curvature K
K=zeros(k_max,i_max);
 if (epsilon==0) %If Epsilon=0,there is no need to compute K
     for k=1:1:k_max
        for i=1:1:i_max
                K(k,i)=0;
        end
     end
 else
     for k=1:1:k_max
         for i=1:1:i_max
                K(k,i)=phi_x_x_cen(k,i)+phi_z_z_cen(k,i);
         end
     end
 end
%___________________________________________________________________________
%Define Partial Hamiltonians H for LLLF Scheme
cos_t_pfx_pfz=zeros(k_max,i_max); %cos(theta) with all combinations of +/- FD's
cos_t_pfx_nfz=zeros(k_max,i_max);
cos_t_nfx_pfz=zeros(k_max,i_max);
cos_t_nfx_nfz=zeros(k_max,i_max);
H1_LLLF_pfx_pfz=zeros(k_max,i_max); %Partial H wrt phi_x (all +/- FD combinations)
H1_LLLF_pfx_nfz=zeros(k_max,i_max);
H1_LLLF_nfx_pfz=zeros(k_max,i_max);
H1_LLLF_nfx_nfz=zeros(k_max,i_max);
H3_LLLF_pfx_pfz=zeros(k_max,i_max); %Partial H wrt phi_z (all +/- FD combinations)
H3_LLLF_pfx_nfz=zeros(k_max,i_max);
H3_LLLF_nfx_pfz=zeros(k_max,i_max);
H3_LLLF_nfx_nfz=zeros(k_max,i_max);
for k=1:1:k_max
        for i=1:1:i_max
           cos_t_pfx_pfz(k,i)=(x_cord_local(i).*(phi_x_pos(k,i))+z_cord(k).*(phi_z_pos(k,i)))./sqrt(x_cord_local(i).^2+z_cord(k).^2);
           cos_t_pfx_nfz(k,i)=(x_cord_local(i).*(phi_x_pos(k,i))+z_cord(k).*(phi_z_neg(k,i)))./sqrt(x_cord_local(i).^2+z_cord(k).^2);
           cos_t_nfx_pfz(k,i)=(x_cord_local(i).*(phi_x_neg(k,i))+z_cord(k).*(phi_z_pos(k,i)))./sqrt(x_cord_local(i).^2+z_cord(k).^2);
           cos_t_nfx_nfz(k,i)=(x_cord_local(i).*(phi_x_neg(k,i))+z_cord(k).*(phi_z_neg(k,i)))./sqrt(x_cord_local(i).^2+z_cord(k).^2);
           if (cos_t_pfx_pfz(k,i)>1) %Limit cos(theta)
                cos_t_pfx_pfz(k,i)=1;
           end
           if (cos_t_pfx_nfz(k,i)>1)
                cos_t_pfx_nfz(k,i)=1;
           end
           if (cos_t_nfx_pfz(k,i)>1)
                cos_t_nfx_pfz(k,i)=1;
           end
           if (cos_t_nfx_nfz(k,i)>1)
                cos_t_nfx_nfz(k,i)=1;
           end
           %_________
           if cos_t_pfx_pfz(k,i)==0 %Done to ensure 0/0 doesn't results and hence an error - F=H=0 when this occurs
        H3_LLLF_pfx_pfz(k,i)=0;
        H1_LLLF_pfx_pfz(k,i)=0;
      else
        if flag_T_M(k,i)==1 %T
          H3_LLLF_pfx_pfz(k,i)=real((C/rho_s)*Eros_pow(k,i).*((cos_t_pfx_pfz(k,i))^(k_vel+1))...
            .*(z_cord(k)./sqrt(x_cord_local(i).^2+z_cord(k).^2)).*((k_vel+1)./cos_t_pfx_pfz(k,i)));
        else %M
          H3_LLLF_pfx_pfz(k,i)=real((C_M/rho_s_M).*Eros_pow(k,i).*((cos_t_pfx_pfz(k,i)).^n_1_M).*((1+H_vic_M*(1-cos_t_pfx_pfz(k,i))).^n_2_M)...
            .*(z_cord(k)./sqrt(x_cord_local(i).^2+z_cord(k).^2)).*(n_1_M./cos_t_pfx_pfz(k,i)-n_2_M*H_vic_M./(1+H_vic_M*(1-cos_t_pfx_pfz(k,i))))));
        end

        H1_LLLF_pfx_pfz(k,i)=(x_cord_local(i)./z_cord(k)).*H3_LLLF_pfx_pfz(k,i);
      end
      %_________
      if cos_t_pfx_nfz(k,i)==0
        H3_LLLF_pfx_nfz(k,i)=0;
        H1_LLLF_pfx_nfz(k,i)=0;
      else
        if flag_T_M(k,i)==1 %T
          H3_LLLF_pfx_nfz(k,i)=real((C/rho_s)*Eros_pow(k,i).*((cos_t_pfx_nfz(k,i))^(k_vel+1))...
            .*(z_cord(k)./sqrt(x_cord_local(i).^2+z_cord(k).^2)).*((k_vel+1)./cos_t_pfx_nfz(k,i)));
        else %M
          H3_LLLF_pfx_nfz(k,i)=real((C_M/rho_s_M).*Eros_pow(k,i).*((cos_t_pfx_nfz(k,i)).^n_1_M).*((1+H_vic_M*(1-cos_t_pfx_nfz(k,i))).^n_2_M)...
            .*(z_cord(k)./sqrt(x_cord_local(i).^2+z_cord(k).^2)).*(n_1_M./cos_t_pfx_nfz(k,i)-n_2_M*H_vic_M./(1+H_vic_M*(1-cos_t_pfx_nfz(k,i))))));
        end

        H1_LLLF_pfx_nfz(k,i)=(x_cord_local(i)./z_cord(k)).*H3_LLLF_pfx_nfz(k,i);
      end
      %_________
      if cos_t_nfx_pfz(k,i)==0
        H3_LLLF_nfx_pfz(k,i)=0;
        H1_LLLF_nfx_pfz(k,i)=0;
      else
        if flag_T_M(k,i)==1 %T
          H3_LLLF_nfx_pfz(k,i)=real((C/rho_s)*Eros_pow(k,i).*((cos_t_nfx_pfz(k,i))^(k_vel+1))...
            .*(z_cord(k)./sqrt(x_cord_local(i).^2+z_cord(k).^2)).*((k_vel+1)./cos_t_nfx_pfz(k,i)));
        else %M
          H3_LLLF_nfx_pfz(k,i)=real((C_M/rho_s_M).*Eros_pow(k,i).*((cos_t_nfx_pfz(k,i)).^n_1_M).*((1+H_vic_M*(1-cos_t_nfx_pfz(k,i))).^n_2_M)...
            .*(z_cord(k)./sqrt(x_cord_local(i).^2+z_cord(k).^2)).*(n_1_M./cos_t_nfx_pfz(k,i)-n_2_M*H_vic_M./(1+H_vic_M*(1-cos_t_nfx_pfz(k,i))))));
        end

        H1_LLLF_nfx_pfz(k,i)=(x_cord_local(i)./z_cord(k)).*H3_LLLF_nfx_pfz(k,i);
      end
      %_________
      if cos_t_nfx_nfz(k,i)==0
        H3_LLLF_nfx_nfz(k,i)=0;
        H1_LLLF_nfx_nfz(k,i)=0;
      else
        if flag_T_M(k,i)==1 %T
          H3_LLLF_nfx_nfz(k,i)=real((C/rho_s)*Eros_pow(k,i).*((cos_t_nfx_nfz(k,i))^(k_vel+1))...
            .*(z_cord(k)./sqrt(x_cord_local(i).^2+z_cord(k).^2)).*((k_vel+1)./cos_t_nfx_nfz(k,i)));
        else %M
          H3_LLLF_nfx_nfz(k,i)=real((C_M/rho_s_M).*Eros_pow(k,i).*((cos_t_nfx_nfz(k,i)).^n_1_M).*((1+H_vic_M*(1-cos_t_nfx_nfz(k,i))).^n_2_M)...
            .*(z_cord(k)./sqrt(x_cord_local(i).^2+z_cord(k).^2)).*(n_1_M./cos_t_nfx_nfz(k,i)-n_2_M*H_vic_M./(1+H_vic_M*(1-cos_t_nfx_nfz(k,i))))));
        end

        H1_LLLF_nfx_nfz(k,i)=(x_cord_local(i)./z_cord(k)).*H3_LLLF_nfx_nfz(k,i);
      end

    end
end
%__________________________________________________________________________
%LLLF Scheme
%Initialization (preallocation) to increase computational speed
phi_x_star=zeros(k_max,i_max); %FD used in numerical Hamiltonian H (star)
phi_z_star=zeros(k_max,i_max);
alpha_x=zeros(k_max,i_max); %Bounds of partial deriv. of H
alpha_z=zeros(k_max,i_max);
Ham=zeros(k_max,i_max); %Hamiltonian
Ham_num=zeros(k_max,i_max); %Numerical Hamiltonian
cos_t_star=zeros(k_max,i_max);
 for k=1:1:k_max
    for i=1:1:i_max
        phi_x_star(k,i)=(phi_x_pos(k,i)+phi_x_neg(k,i))/2;
        phi_z_star(k,i)=(phi_z_pos(k,i)+phi_z_neg(k,i))/2;
        H1_LLLF_array=[abs(H1_LLLF_pfx_pfz(k,i)),abs(H1_LLLF_pfx_nfz(k,i)),abs(H1_LLLF_nfx_pfz(k,i)),abs(H1_LLLF_nfx_nfz(k,i))];
        alpha_x(k,i)=max(H1_LLLF_array);
        H3_LLLF_array=[abs(H3_LLLF_pfx_pfz(k,i)),abs(H3_LLLF_pfx_nfz(k,i)),abs(H3_LLLF_nfx_pfz(k,i)),abs(H3_LLLF_nfx_nfz(k,i))];
        alpha_z(k,i)=max(H3_LLLF_array);
      %Define Numerical Hamiltonian
       cos_t_star(k,i)=(x_cord_local(i).*(phi_x_star(k,i))+z_cord(k).*(phi_z_star(k,i)))./sqrt(x_cord_local(i).^2+z_cord(k).^2);
       if (cos_t_star(k,i)>1)
          cos_t_star(k,i)=1;
       end

       if flag_T_M(k,i)==1 %T
            Ham(k,i)=real((C/rho_s)*Eros_pow(k,i).*((cos_t_star(k,i))^(k_vel+1)));
       else %M
            Ham(k,i)=real((C_M/rho_s_M).*Eros_pow(k,i).*((cos_t_star(k,i)).^n_1_M).*((1+H_vic_M*(1-cos_t_star(k,i))).^n_2_M));
       end

       Ham_num(k,i)=Ham(k,i)-(alpha_x(k,i)/2).*(phi_x_pos(k,i)-phi_x_neg(k,i))-(alpha_z(k,i)/2).*(phi_z_pos(k,i)-phi_z_neg(k,i));

    end
 end
%_________________________________________________________________________
%Define Central Difference Hamiltonian
Ham_cen=zeros(k_max,i_max);
cos_t_cen=zeros(k_max,i_max);
 if (epsilon==0)
   for k=1:1:k_max
     for i=1:1:i_max
        Ham_cen(k,i)=0;
     end
   end
 else
        for k=1:1:k_max
            for i=1:1:i_max
               cos_t_cen(k,i)=(x_cord_local(i).*(phi_x_cen(k,i))+z_cord(k).*(phi_z_cen(k,i)))./sqrt(x_cord_local(i).^2+z_cord(k).^2);
               if (cos_t_cen(k,i)>1)
                  cos_t_cen(k,i)=1;
               end

               if flag_T_M(k,i)==1 %T
                    Ham_cen(k,i)=real((C/rho_s)*Eros_pow(k,i).*((cos_t_cen(k,i))^(k_vel+1)));
               else %M
                    Ham_cen(k,i)=real((C_M/rho_s_M).*Eros_pow(k,i).*((cos_t_cen(k,i)).^n_1_M).*((1+H_vic_M*(1-cos_t_cen(k,i))).^n_2_M));
               end
            end
        end
 end
%_________________________________________________________________________
%Initial small dt
dt=t_in/Num_iter;

%Curvature coefficient
%NOTE can scale up/down 'epsilon' to use different epsilons for M and T
%but 'epsilon' must be maximum for CFL condition
epsilon_gen=zeros(k_max,i_max);
for k=1:1:k_max
     for i=1:1:i_max
        if flag_T_M(k,i)==1 %T
           epsilon_gen(k,i)=epsilon;
        else %M
           epsilon_gen(k,i)=0;
        end
     end
end

%_________________________________________________________________________
%Solve EOM (1st iteration) for phi's to pass onto while loop
phi_1=zeros(k_max,i_max);
for k=1:1:k_max
     for i=1:1:i_max
        phi_1(k,i)=phi(k,i)+dt.*(-Ham_num(k,i)+epsilon_gen(k,i).*K(k,i).*Ham_cen(k,i));
     end
end
%_________________________________________________________________________
%Surface Interpolation Algorithm (where phi=0)

z_surf_1=zeros(i_max,1); %Initial z surface value at a given x
z_surf_2=zeros(i_max,1); %Second z surface value at a given x (if multi-valued surface)
z_surf_3=zeros(i_max,1); %Third z surface value at a given x (if multi-valued surface)
for i=1:1:i_max %Order matters
   flag_1=0;
   flag_2=0;
    for k=1:1:k_max %Order matters
         if ((flag_1==0)&&(((phi_1(k,i)>0)&&(phi_1(k+1,i)<0))||(phi_1(k,i)==0)))
            z_surf_1(i)=((phi_1(k,i).*(z_cord(k)-z_cord(k+1)))./(phi_1(k+1,i)-phi_1(k,i)))+z_cord(k);
            flag_1=i; %Done to ensure that the rest of code executes only if this 'if condition' occurs (see below)
            continue %go to next iteration in the for loop, skipping whatever remains below for this iteration
        end
        if (((flag_2==0)&&(flag_1==i)&&(k~=k_max))&&(((phi_1(k,i)<0)&&(phi_1(k+1,i)>0))||(phi_1(k,i)==0))) %put in AND expression for k~=kmax since grid
ends (no z surface there)
            z_surf_2(i)=((phi_1(k,i).*(z_cord(k)-z_cord(k+1)))./(phi_1(k+1,i)-phi_1(k,i)))+z_cord(k);
            flag_2=i;
            continue
        end
        if ((flag_2==i)&&(((phi_1(k,i)>0)&&(phi_1(k+1,i)<0))||(phi_1(k,i)==0)))
            z_surf_3(i)=((phi_1(k,i).*(z_cord(k)-z_cord(k+1)))./(phi_1(k+1,i)-phi_1(k,i)))+z_cord(k);
        end
    end
end

x_surf_1=zeros(1,k_max); %First x surface value at a given z
x_surf_2=zeros(1,k_max); %Second x surface value at a given z
for k=1:1:k_max %Order matters
   flag_3=0;
    for i=1:1:i_max %Order matters
         if (((flag_3==0)&&(i~=i_max)&&(i~=1))&&(((phi_1(k,i)>0)&&(phi_1(k,i+1)<0))||(phi_1(k,i)==0))) %put in AND expression for i~=imax or 1 since grid
ends (no x surface there)
            x_surf_1(k)=((phi_1(k,i).*(x_cord(i)-x_cord(i+1)))./(phi_1(k,i+1)-phi_1(k,i)))+x_cord(i);
            flag_3=k;
            continue
         end
         if (((flag_3==k)&&(i~=i_max))&&(((phi_1(k,i)<0)&&(phi_1(k,i+1)>0))||(phi_1(k,i)==0)))
            x_surf_2(k)=((phi_1(k,i).*(x_cord(i)-x_cord(i+1)))./(phi_1(k,i+1)-phi_1(k,i)))+x_cord(i);
         end
    end
end

% Convert all z_surf and x_surf to one array
xz_surf=[x_cord', z_surf_1;x_cord', z_surf_2;x_cord', z_surf_3;x_surf_1',z_cord;x_surf_2',z_cord];
b_max=3*i_max+2*k_max;
for b=1:1:(3*i_max)
      if xz_surf(b,2)==0 %Ignore this part of array-see below (no surface there)
      xz_surf(b,1)=0;
      xz_surf(b,2)=0;
      else
      xz_surf(b,1)=xz_surf(b,1);
      xz_surf(b,2)=xz_surf(b,2);
      end
end

for b=(3*i_max+1):1:b_max
      if xz_surf(b,1)==0 %Ignore this part of array-see below (no surface there)
      xz_surf(b,1)=0;
      xz_surf(b,2)=0;
      else
      xz_surf(b,1)=xz_surf(b,1);
      xz_surf(b,2)=xz_surf(b,2);
      end
end
%____________________________________________________________________________
%Calculate Upper and Lower Band

%UPPER BAND

z_surf_1_U=zeros(i_max,1); %Analogous to phi=0 surface
z_surf_2_U=zeros(i_max,1);
z_surf_3_U=zeros(i_max,1);
for i=1:1:i_max
   flag_1_U=0;
   flag_2_U=0;
    for k=1:1:k_max
        if ((flag_1_U==0)&&(((phi_1(k,i)>(BS_U*dz))&&(phi_1(k+1,i)<(BS_U*dz)))||(phi_1(k,i)==(BS_U*dz))))
            z_surf_1_U(i)=(((BS_U*dz-phi_1(k,i)).*(z_cord(k+1)-z_cord(k)))./(phi_1(k+1,i)-phi_1(k,i)))+z_cord(k);
            flag_1_U=i;
            continue
        end
        if (((flag_2_U==0)&&(flag_1_U==i)&&(k~=k_max))&&(((phi_1(k,i)<(BS_U*dz))&&(phi_1(k+1,i)>(BS_U*dz)))||(phi_1(k,i)==(BS_U*dz))))
            z_surf_2_U(i)=(((BS_U*dz-phi_1(k,i)).*(z_cord(k+1)-z_cord(k)))./(phi_1(k+1,i)-phi_1(k,i)))+z_cord(k);
            flag_2_U=i;
            continue
        end
        if ((flag_2_U==i)&&(((phi_1(k,i)>(BS_U*dz))&&(phi_1(k+1,i)<(BS_U*dz)))||(phi_1(k,i)==(BS_U*dz))))
            z_surf_3_U(i)=(((BS_U*dz-phi_1(k,i)).*(z_cord(k+1)-z_cord(k)))./(phi_1(k+1,i)-phi_1(k,i)))+z_cord(k);
        end
    end
```

```
end

x_surf_1_U=zeros(1,k_max);
x_surf_2_U=zeros(1,k_max);
for k=1:1:k_max
    flag_3_U=0;
     for i=1:1:i_max
           if (((flag_3_U==0)&&(i~=i_max)&&(i~=1))&&(((phi_1(k,i)>(BS_U*dz))&&(phi_1(k,i+1)<(BS_U*dz)))||(phi_1(k,i)==(BS_U*dz))))
               x_surf_1_U(k)=(((BS_U*dz-phi_1(k,i)).*(x_cord(i+1)-x_cord(i)))./(phi_1(k,i+1)-phi_1(k,i)))+x_cord(i);
               flag_3_U=k;
               continue
           end
           if (((flag_3_U==k)&&(i~=i_max))&&(((phi_1(k,i)<(BS_U*dz))&&(phi_1(k,i+1)>(BS_U*dz)))||(phi_1(k,i)==(BS_U*dz))))
               x_surf_2_U(k)=(((BS_U*dz-phi_1(k,i)).*(x_cord(i+1)-x_cord(i)))./(phi_1(k,i+1)-phi_1(k,i)))+x_cord(i);
           end
     end
end

% Convert all z_surf_U and x_surf_U to one array
xz_surf_U=[x_cord', z_surf_1_U;x_cord', z_surf_2_U;x_cord', z_surf_3_U;x_surf_1_U',z_cord;x_surf_2_U',z_cord];
for b=1:1:(3*i_max)
        if xz_surf_U(b,2)==0 %Means it was not called up (no surface there)
        xz_surf_U(b,1)=0;
        xz_surf_U(b,2)=0;
        end
end

for b=(3*i_max+1):1:b_max
        if xz_surf_U(b,1)==0
        xz_surf_U(b,1)=0;
        xz_surf_U(b,2)=0;
        end
end

%LOWER BAND

z_surf_1_L=zeros(i_max,1); %Analogous to phi=0 surface
z_surf_2_L=zeros(i_max,1);
z_surf_3_L=zeros(i_max,1);
for i=1:1:i_max
    flag_1_L=0;
    flag_2_L=0;
     for k=1:1:k_max
          if ((flag_1_L==0)&&(((phi_1(k,i)>(-dz*BS_L))&&(phi_1(k+1,i)<(-dz*BS_L)))||(phi_1(k,i)==(-dz*BS_L))))
              z_surf_1_L(i)=(((-dz*BS_L-phi_1(k,i)).*(z_cord(k+1)-z_cord(k)))./(phi_1(k+1,i)-phi_1(k,i)))+z_cord(k);
              flag_1_L=i;
              continue
          end
          if (((flag_2_L==0)&&(flag_1_L==i)&&(k~=k_max))&&(((phi_1(k,i)<(-dz*BS_L))&&(phi_1(k+1,i)>(-dz*BS_L)))||(phi_1(k,i)==(-dz*BS_L))))
              z_surf_2_L(i)=(((-dz*BS_L-phi_1(k,i)).*(z_cord(k+1)-z_cord(k)))./(phi_1(k+1,i)-phi_1(k,i)))+z_cord(k);
              flag_2_L=i;
              continue
          end
          if ((flag_2_L==i)&&(((phi_1(k,i)>(-dz*BS_L))&&(phi_1(k+1,i)<(-dz*BS_L)))||(phi_1(k,i)==(-dz*BS_L))))
              z_surf_3_L(i)=(((-dz*BS_L-phi_1(k,i)).*(z_cord(k+1)-z_cord(k)))./(phi_1(k+1,i)-phi_1(k,i)))+z_cord(k);
          end
     end
end

x_surf_1_L=zeros(1,k_max);
x_surf_2_L=zeros(1,k_max);
for k=1:1:k_max
    flag_3_L=0;
     for i=1:1:i_max
           if (((flag_3_L==0)&&(i~=i_max)&&(i~=1))&&(((phi_1(k,i)>(-dz*BS_L))&&(phi_1(k,i+1)<(-dz*BS_L)))||(phi_1(k,i)==(-dz*BS_L))))
               x_surf_1_L(k)=(((-dz*BS_L-phi_1(k,i)).*(x_cord(i+1)-x_cord(i)))./(phi_1(k,i+1)-phi_1(k,i)))+x_cord(i);
               flag_3_L=k;
               continue
           end
           if (((flag_3_L==k)&&(i~=i_max))&&(((phi_1(k,i)<(-dz*BS_L))&&(phi_1(k,i+1)>(-dz*BS_L)))||(phi_1(k,i)==(-dz*BS_L))))
               x_surf_2_L(k)=(((-dz*BS_L-phi_1(k,i)).*(x_cord(i+1)-x_cord(i)))./(phi_1(k,i+1)-phi_1(k,i)))+x_cord(i);
           end
     end
end

% Convert all z_surf_L and x_surf_L to one array
xz_surf_L=[x_cord', z_surf_1_L;x_cord', z_surf_2_L;x_cord', z_surf_3_L;x_surf_1_L',z_cord;x_surf_2_L',z_cord];
for b=1:1:(3*i_max)
        if xz_surf_L(b,2)==0
        xz_surf_L(b,1)=0;
```

```
      xz_surf_L(b,2)=0;
      end
end

for b=(3*i_max+1):1:b_max
      if xz_surf_L(b,1)==0
      xz_surf_L(b,1)=0;
      xz_surf_L(b,2)=0;
      end
end
%_____________________________________________________________________________
%Create flags for points IN the Narrow Band
%These flags will only change after band is re-initialized
flag_NB=zeros(k_max,i_max);
for k=1:1:k_max
   for i=1:1:i_max
      if (((phi_1(k,i)>=0)&&(abs(phi_1(k,i))<BS_U*dz))||((phi_1(k,i)<0)&&(abs(phi_1(k,i))<BS_L*dz))) %Don't consider points on boundary
         flag_NB(k,i)=1; %else they will remain 0 (are outside the band)
      end
   end
end

%Create flags to indicate BC pts (adjacent to NB boundary)
%These flags will only change after band is re-initialized
for k=1:1:k_max
   for i=1:1:i_max

      if (i~=1)&&(i~=i_max)&&(k~=1)&&(k~=k_max)
         if (flag_NB(k,i)==0)&&((flag_NB(k,i+1)==1)||(flag_NB(k,i-1)==1)||(flag_NB(k+1,i)==1)||(flag_NB(k-1,i)==1))
            flag_NB(k,i)=2;
         end
      elseif (i==1)&&(k~=1)&&(k~=k_max)
         if (flag_NB(k,i)==0)&&((flag_NB(k,i+1)==1)||(flag_NB(k+1,i)==1)||(flag_NB(k-1,i)==1))
            flag_NB(k,i)=2;
         end
      elseif (i==i_max)&&(k~=1)&&(k~=k_max)
         if (flag_NB(k,i)==0)&&((flag_NB(k,i-1)==1)||(flag_NB(k+1,i)==1)||(flag_NB(k-1,i)==1))
            flag_NB(k,i)=2;
         end
      elseif (k==1)&&(i~=1)&&(i~=i_max)
         if (flag_NB(k,i)==0)&&((flag_NB(k,i+1)==1)||(flag_NB(k,i-1)==1)||(flag_NB(k+1,i)==1))
            flag_NB(k,i)=2;
         end
      elseif (k==k_max)&&(i~=1)&&(i~=i_max)
         if (flag_NB(k,i)==0)&&((flag_NB(k,i+1)==1)||(flag_NB(k,i-1)==1)||(flag_NB(k-1,i)==1))
            flag_NB(k,i)=2;
         end
      elseif  (i==1)&&(k==1)
         if (flag_NB(k,i)==0)&&((flag_NB(k,i+1)==1)||(flag_NB(k+1,i)==1))
            flag_NB(k,i)=2;
         end
      elseif  (i==i_max)&&(k==1)
         if (flag_NB(k,i)==0)&&((flag_NB(k,i-1)==1)||(flag_NB(k+1,i)==1))
            flag_NB(k,i)=2;
         end
      elseif  (i==1)&&(k==k_max)
         if (flag_NB(k,i)==0)&&((flag_NB(k,i+1)==1)||(flag_NB(k-1,i)==1))
            flag_NB(k,i)=2;
         end
      else %(i==i_max)&&(k==k_max)
         if (flag_NB(k,i)==0)&&((flag_NB(k,i-1)==1)||(flag_NB(k-1,i)==1))
            flag_NB(k,i)=2;
         end
      end

   end
end

%Create flags for the remaining pts. so when rebuild band, we know where
%phi's should be +ve or -ve; i.e. assign flag_NB(k,i) = 4 and 3 so as to
%later indicate +ve and -ve phi
for k=1:1:k_max
   for i=1:1:i_max
      if (flag_NB(k,i)~=1)&&(flag_NB(k,i)~=2)
         if (phi_1(k,i)>0) %we only need to check +ve phi_1 since these are outside the band
            flag_NB(k,i)=4; %label +ve phi_1's outside band
         else
            flag_NB(k,i)=3; %label -ve phi_1's outside band
         end
      end
```

```
    end
end
%_____________________________________________________________________________
%Algorithm to obtain mask angles for T

%Adjustment to Mass Flux due to Mask Model for Target M(x')

%Calculate visibility angles based on zero level set (M)
x_prime_surf_LM=zeros(b_max,1); %Initial surface for M in local coordinates
z_prime_surf_LM=zeros(b_max,1);
x_prime_surf_RM=zeros(b_max,1);
z_prime_surf_RM=zeros(b_max,1);
for b=1:1:b_max
    if ((xz_surf(b,1)>=x_min_grid)&&(xz_surf(b,1)<=(x_min_grid+leng_M_L)))&&...
            ((xz_surf(b,2)>=(h*sin(alfa)-H_m))&&(xz_surf(b,2)<h*sin(alfa)))
        x_prime_surf_LM(b)=(xz_surf(b,1)-x_off).*sin(alfa)-xz_surf(b,2).*cos(alfa); %Rotated local x
        z_prime_surf_LM(b)=(xz_surf(b,1)-x_off).*cos(alfa)+xz_surf(b,2).*sin(alfa); %Rotated local z
    else
        x_prime_surf_LM(b)=NaN;
        z_prime_surf_LM(b)=NaN;
    end

    if ((xz_surf(b,1)>=(x_max_grid-leng_M_R))&&(xz_surf(b,1)<=x_max_grid))&&...
            ((xz_surf(b,2)>=(h*sin(alfa)-H_m))&&(xz_surf(b,2)<h*sin(alfa)))
        x_prime_surf_RM(b)=(xz_surf(b,1)-x_off).*sin(alfa)-xz_surf(b,2).*cos(alfa); %Rotated local x
        z_prime_surf_RM(b)=(xz_surf(b,1)-x_off).*cos(alfa)+xz_surf(b,2).*sin(alfa); %Rotated local z
    else
        x_prime_surf_RM(b)=NaN;
        z_prime_surf_RM(b)=NaN;
    end

end

%Find min x_prime_surf_LM
[max_x_prime_surf_LM,I_max_LM]=max(x_prime_surf_LM); %Will ignore NaN's

%Min tan of left 'spread' angle defined by mask
if (max_x_prime_surf_LM>=0) %Case (b)
    tan_fi_min=max_x_prime_surf_LM/z_prime_surf_LM(I_max_LM);
else %Case (b) and (c)
    %Find min |x_prime_surf_LM|
    [min_x_prime_surf_LM,I_min_LM]=min(abs(x_prime_surf_LM));

    if (alfa==90*pi/180)
        tan_fi_min=min_x_prime_surf_LM/h;
    else
        tan_fi_min=min_x_prime_surf_LM/z_prime_surf_LM(I_min_LM);
    end
end

%Find min x_prime_surf_RM
[min_x_prime_surf_RM,I_min_RM]=min(x_prime_surf_RM);
%Max tan of right 'spread' angle defined by mask
if (alfa==90*pi/180)
    tan_fi_max=min_x_prime_surf_RM/h;
else
    tan_fi_max=min_x_prime_surf_RM/z_prime_surf_RM(I_min_RM);
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%START of 2nd Strike Algorithm %%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%Kinematic rebound parameters (angle and velocity)
f_alfa_AR_T=0.9; %angle (T)
f_v_AR_T=0.4; %velocity (T)
f_alfa_AR_M=1; %angle (M)
f_v_AR_M=0.4; %velocity (M)

%Obtain arriving node 'a'='AR' theta_AR, gamma_AR & departing 'd'='DE' theta_DE at surface

%Initialize variables before loop entry
J_k_AR=0; %Nearest reference node index for interpolation
J_i_AR=0;
phi_x_pos_AR=zeros(b_max,1); %FD's at 'AR' node
phi_x_neg_AR=zeros(b_max,1);
phi_x_cen_AR=zeros(b_max,1);
phi_z_pos_AR=zeros(b_max,1);
phi_z_neg_AR=zeros(b_max,1);
phi_z_cen_AR=zeros(b_max,1);
```

```
phi_x_star_AR=zeros(b_max,1);
phi_z_star_AR=zeros(b_max,1);
cos_t_star_AR=zeros(b_max,1); %Cos(theta) star
cos_g_star_AR=zeros(b_max,1); %Cos(gamma) star

theta_AR=zeros(b_max,1);
theta_AR_deg=zeros(b_max,1);
gamma_AR=zeros(b_max,1);
gamma_AR_deg=zeros(b_max,1);
x_prime_surf_AR=zeros(b_max,1); %Local x of AR node
f_alfa_AR=zeros(b_max,1); %f_alfa for entire surface
theta_DE=zeros(b_max,1);
theta_DE_deg=zeros(b_max,1);

 for b=1:1:b_max

    if ((xz_surf(b,1)~=0)&&(xz_surf(b,2)~=0))||((alfa==(90*pi/180))&&((b==1)||(b==(i_max+1))||(b==(2*i_max+1)))&&(xz_surf(b,1)==0)&&(xz_surf(b,2)~=0))
      %if surface pt; Need 2nd condition since AT 90deg xcord=0 for 1,imax+1,2imax+1 and code thinks no surface there

       if (b>=1)&&(b<=(3*i_max)) %zsurf used

        if (b>=1)&&(b<=i_max)  %%Here, we use I_min_ik="i", acts as reference for phi interpolation to know "i" value (see below)
          I_min_ik=b; %z_surf_1 used
        end
        if (b>=(i_max+1))&&(b<=(2*i_max))
          I_min_ik=b-i_max;  %z_surf_2 used
        end

        if (b>=(2*i_max+1))&&(b<=(3*i_max))
          I_min_ik=b-2*i_max;  %z_surf_3 used
        end

            %Calculate nearest k index to surface
            J_k_AR=floor(k_max-(xz_surf(b,2)-z_min_grid)/dz);

            %FD's at 'AR' node using surface nodes (in b/w grid nodes)
            %i.e. Calculation of dphi/dx,dphi/dz; Define BC's
            if I_min_ik==i_max
               phi_x_pos_AR(b)=0;
            else
               phi_B_z_surf=((xz_surf(b,2)-z_cord(J_k_AR))/(z_cord(J_k_AR+1)-z_cord(J_k_AR))).*(phi_1(J_k_AR+1,I_min_ik+1)-
phi_1(J_k_AR,I_min_ik+1))+phi_1(J_k_AR,I_min_ik+1);
               phi_x_pos_AR(b)=(phi_B_z_surf-0)./dx;
            end

            if I_min_ik==1
               phi_x_neg_AR(b)=0;
            else
               phi_A_z_surf=((xz_surf(b,2)-z_cord(J_k_AR))/(z_cord(J_k_AR+1)-z_cord(J_k_AR))).*(phi_1(J_k_AR+1,I_min_ik-1)-phi_1(J_k_AR,I_min_ik-
1))+phi_1(J_k_AR,I_min_ik-1);
               phi_x_neg_AR(b)=(0-phi_A_z_surf)./dx;
            end

            if (I_min_ik==i_max)||(I_min_ik==1)
               phi_x_cen_AR(b)=0;
            else
               phi_A_z_surf=((xz_surf(b,2)-z_cord(J_k_AR))/(z_cord(J_k_AR+1)-z_cord(J_k_AR))).*(phi_1(J_k_AR+1,I_min_ik-1)-phi_1(J_k_AR,I_min_ik-
1))+phi_1(J_k_AR,I_min_ik-1);
               phi_B_z_surf=((xz_surf(b,2)-z_cord(J_k_AR))/(z_cord(J_k_AR+1)-z_cord(J_k_AR))).*(phi_1(J_k_AR+1,I_min_ik+1)-
phi_1(J_k_AR,I_min_ik+1))+phi_1(J_k_AR,I_min_ik+1);
               phi_x_cen_AR(b)=(phi_B_z_surf-phi_A_z_surf)./(2*dx);
            end

            if J_k_AR==1
               phi_z_pos_AR(b)=0;
            else
               phi_C_z_surf=((xz_surf(b,2)+dz-z_cord(J_k_AR-1))/(z_cord(J_k_AR)-z_cord(J_k_AR-1))).*(phi_1(J_k_AR,I_min_ik)-phi_1(J_k_AR-
1,I_min_ik))+phi_1(J_k_AR-1,I_min_ik);
               phi_z_pos_AR(b)=(phi_C_z_surf-0)./dz;
            end

            if J_k_AR==k_max
               phi_z_neg_AR(b)=0;
            else
               phi_D_z_surf=((xz_surf(b,2)-dz-z_cord(J_k_AR+1))/(z_cord(J_k_AR+2)-z_cord(J_k_AR+1))).*(phi_1(J_k_AR+2,I_min_ik)-
phi_1(J_k_AR+1,I_min_ik))+phi_1(J_k_AR+1,I_min_ik);
               phi_z_neg_AR(b)=(0-phi_D_z_surf)./dz;
            end

            if (J_k_AR==k_max)||(J_k_AR==1)
          phi_z_cen_AR(b)=0;
        else
          phi_C_z_surf=((xz_surf(b,2)+dz-z_cord(J_k_AR-1))/(z_cord(J_k_AR)-z_cord(J_k_AR-1))).*(phi_1(J_k_AR,I_min_ik)-phi_1(J_k_AR-
1,I_min_ik))+phi_1(J_k_AR-1,I_min_ik);
          phi_D_z_surf=((xz_surf(b,2)-dz-z_cord(J_k_AR+1))/(z_cord(J_k_AR+2)-z_cord(J_k_AR+1))).*(phi_1(J_k_AR+2,I_min_ik)-
phi_1(J_k_AR+1,I_min_ik))+phi_1(J_k_AR+1,I_min_ik);
          phi_z_cen_AR(b)=(phi_C_z_surf-phi_D_z_surf)./(2*dz);
        end

        %Calculate phi_stars_AR
        phi_x_star_AR(b)=(phi_x_pos_AR(b)+phi_x_neg_AR(b))/2;
        phi_z_star_AR(b)=(phi_z_pos_AR(b)+phi_z_neg_AR(b))/2;

        %Calculate Angles and x_surf'_AR
          cos_t_star_AR(b)=((xz_surf(b,1)-x_off).*(phi_x_star_AR(b))+xz_surf(b,2).*(phi_z_star_AR(b)))./(sqrt((xz_surf(b,1)-
x_off).^2+xz_surf(b,2).^2).*sqrt((phi_x_star_AR(b)).^2+(phi_z_star_AR(b)).^2));
          theta_AR(b)=acos(cos_t_star_AR(b));
          theta_AR_deg(b)=theta_AR(b)*(180/pi);
          cos_g_star_AR(b)=((xz_surf(b,1)-x_off).*(-phi_z_star_AR(b))+xz_surf(b,2).*(phi_x_star_AR(b)))./(sqrt((xz_surf(b,1)-
x_off).^2+xz_surf(b,2).^2).*sqrt((phi_x_star_AR(b)).^2+(phi_z_star_AR(b)).^2));
          gamma_AR(b)=acos(cos_g_star_AR(b));
          gamma_AR_deg(b)=gamma_AR(b)*(180/pi);
          if (alfa==90*pi/180)
            x_prime_surf_AR(b)=(xz_surf(b,1)-x_off);
          else
            x_prime_surf_AR(b)=(xz_surf(b,1)-x_off).*sin(alfa)-xz_surf(b,2).*cos(alfa);
          end

          if (xz_surf(b,2)>=(h*sin(alfa))) %T
            f_alfa_AR(b)=f_alfa_AR_T;
          else %M
            f_alfa_AR(b)=f_alfa_AR_M;
          end
          theta_DE(b)=pi-f_alfa_AR(b).*theta_AR(b);
          theta_DE_deg(b)=theta_DE(b)*(180/pi);
%___________________________________________________________________________
     else  %xsurf used

      if (b>=(3*i_max+1))&&(b<=(3*i_max+k_max))  %%Here, we use I_min_ik="k", acts as reference for phi interpolation to know "k" value (see below)
        I_min_ik=b-3*i_max; %x_surf_1 used
      end
      if (b>=(3*i_max+k_max+1))&&(b<=b_max)
        I_min_ik=b-3*i_max-k_max;  %x_surf_2 used
      end

        %Repeat above algorithm but for xsurf
        %Calculate nearest i index to surface
        J_i_AR=floor(1+(xz_surf(b,1)-x_min_grid)/dx);

        %Calculation of dphi/dx,dphi/dz; Define BC's
        if J_i_AR==i_max
          phi_x_pos_AR(b)=0;
        else
          phi_D_x_surf=((xz_surf(b,1)+dx-x_cord(J_i_AR+1))/(x_cord(J_i_AR+2)-x_cord(J_i_AR+1))).*(phi_1(I_min_ik,J_i_AR+2)-
phi_1(I_min_ik,J_i_AR+1))+phi_1(I_min_ik,J_i_AR+1);
          phi_x_pos_AR(b)=(phi_D_x_surf-0)./dx;
        end

        if J_i_AR==1
          phi_x_neg_AR(b)=0;
        else
          phi_C_x_surf=((xz_surf(b,1)-dx-x_cord(J_i_AR-1))/(x_cord(J_i_AR)-x_cord(J_i_AR-1))).*(phi_1(I_min_ik,J_i_AR)-phi_1(I_min_ik,J_i_AR-
1))+phi_1(I_min_ik,J_i_AR-1);
          phi_x_neg_AR(b)=(0-phi_C_x_surf)./dx;
        end

        if (J_i_AR==i_max)||(J_i_AR==1)
          phi_x_cen_AR(b)=0;
        else
          phi_C_x_surf=((xz_surf(b,1)-dx-x_cord(J_i_AR-1))/(x_cord(J_i_AR)-x_cord(J_i_AR-1))).*(phi_1(I_min_ik,J_i_AR)-phi_1(I_min_ik,J_i_AR-
1))+phi_1(I_min_ik,J_i_AR-1);
          phi_D_x_surf=((xz_surf(b,1)+dx-x_cord(J_i_AR+1))/(x_cord(J_i_AR+2)-x_cord(J_i_AR+1))).*(phi_1(I_min_ik,J_i_AR+2)-
phi_1(I_min_ik,J_i_AR+1))+phi_1(I_min_ik,J_i_AR+1);
          phi_x_cen_AR(b)=(phi_D_x_surf-phi_C_x_surf)./(2*dx);
        end

        if I_min_ik==1
          phi_z_pos_AR(b)=0;
        else
```

```
               phi_A_x_surf=((xz_surf(b,1)-x_cord(J_i_AR))/(x_cord(J_i_AR+1)-x_cord(J_i_AR))).*(phi_1(I_min_ik-1,J_i_AR+1)-phi_1(I_min_ik-1,J_i_AR))+phi_1(I_min_ik-1,J_i_AR);
               phi_z_pos_AR(b)=(phi_A_x_surf-0)./dz;
            end

            if I_min_ik==k_max
               phi_z_neg_AR(b)=0;
            else
               phi_B_x_surf=((xz_surf(b,1)-x_cord(J_i_AR))/(x_cord(J_i_AR+1)-x_cord(J_i_AR))).*(phi_1(I_min_ik+1,J_i_AR+1)-phi_1(I_min_ik+1,J_i_AR))+phi_1(I_min_ik+1,J_i_AR);
               phi_z_neg_AR(b)=(0-phi_B_x_surf)./dz;
            end

            if (I_min_ik==k_max)||(I_min_ik==1)
               phi_z_cen_AR(b)=0;
            else
               phi_A_x_surf=((xz_surf(b,1)-x_cord(J_i_AR))/(x_cord(J_i_AR+1)-x_cord(J_i_AR))).*(phi_1(I_min_ik-1,J_i_AR+1)-phi_1(I_min_ik-1,J_i_AR))+phi_1(I_min_ik-1,J_i_AR);
               phi_B_x_surf=((xz_surf(b,1)-x_cord(J_i_AR))/(x_cord(J_i_AR+1)-x_cord(J_i_AR))).*(phi_1(I_min_ik+1,J_i_AR+1)-phi_1(I_min_ik+1,J_i_AR))+phi_1(I_min_ik+1,J_i_AR);
               phi_z_cen_AR(b)=(phi_A_x_surf-phi_B_x_surf)./(2*dz);
            end

            %Calculate phi_stars_AR
            phi_x_star_AR(b)=(phi_x_pos_AR(b)+phi_x_neg_AR(b))/2;
            phi_z_star_AR(b)=(phi_z_pos_AR(b)+phi_z_neg_AR(b))/2;

            %Calculate Angles and x_surf'_AR
               cos_t_star_AR(b)=((xz_surf(b,1)-x_off).*(phi_x_star_AR(b))+xz_surf(b,2).*(phi_z_star_AR(b)))./(sqrt((xz_surf(b,1)-x_off).^2+xz_surf(b,2).^2).*sqrt((phi_x_star_AR(b)).^2+(phi_z_star_AR(b)).^2));
               theta_AR(b)=acos(cos_t_star_AR(b));
               theta_AR_deg(b)=theta_AR(b)*(180/pi);
               cos_g_star_AR(b)=((xz_surf(b,1)-x_off).*(-phi_z_star_AR(b))+xz_surf(b,2).*(phi_x_star_AR(b)))./(sqrt((xz_surf(b,1)-x_off).^2+xz_surf(b,2).^2).*sqrt((phi_x_star_AR(b)).^2+(phi_z_star_AR(b)).^2));
               gamma_AR(b)=acos(cos_g_star_AR(b));
               gamma_AR_deg(b)=gamma_AR(b)*(180/pi);
               if (alfa==90*pi/180)
                  x_prime_surf_AR(b)=(xz_surf(b,1)-x_off);
               else
                  x_prime_surf_AR(b)=(xz_surf(b,1)-x_off).*sin(alfa)-xz_surf(b,2).*cos(alfa);
               end

               if (xz_surf(b,2)>=(h*sin(alfa))) %T
                  f_alfa_AR(b)=f_alfa_AR_T;
               else %M
                  f_alfa_AR(b)=f_alfa_AR_M;
               end
               theta_DE(b)=pi-f_alfa_AR(b).*theta_AR(b);
               theta_DE_deg(b)=theta_DE(b)*(180/pi);

        end

    else  %Need this since it accounts for cases where surface wasn't encountered (i.e. x_surf and z_surf = 0 numerically)
        %and theta_AR(b) would stay 0 since pre-allocated with 0's and the fact that theta_AR can actually = 0
       theta_AR(b)=NaN;
       theta_AR_deg(b)=NaN;
       gamma_AR(b)=NaN;
       gamma_AR_deg(b)=NaN;
       x_prime_surf_AR(b)=NaN;
       f_alfa_AR(b)=NaN;
       theta_DE(b)=NaN;
       theta_DE_deg(b)=NaN;
    end

 end
NaN_Chk_theta_AR=isnan(theta_AR); %If any entries are NaN, returns 1 for that entry, else 0

%2nd Strike Detection Algorithm

%NOTE: All values in array for phi_AR (arriving node 'AR'='a') or phi_D (damaged node 'D'='e') or corresponding xz_surf
%values, or thetas , x'surf and others are the same, so can reuse above found values but the indices will
%correspond to the right nodes (AR or D)

f_v_AR=zeros(b_max,1); %f_v_AR for entire surface
f_v_AR_fin=zeros(b_max,1); %Corrected f_v_AR for entire surface
v_AR=zeros(b_max,1); %Velocity at node AR
flux_AR=zeros(b_max,1); %Flux at node AR
theta_D=zeros(b_max,1); %theta at node D
theta_D_deg=zeros(b_max,1);
```

```
cos_t_pfx_pfz_D=zeros(b_max,1); %cos(theta) at node D
cos_t_pfx_nfz_D=zeros(b_max,1);
cos_t_nfx_pfz_D=zeros(b_max,1);
cos_t_nfx_nfz_D=zeros(b_max,1);
cos_t_star_D=zeros(b_max,1);
cos_t_cen_D=zeros(b_max,1);

c_max=b_max;
ds_crit=dz; %Critical spacing to invoke inclusion of 2nd strike to node D
No_ds_cr=3.75;%4; %No. of ds_crit to define limit of min U_D_AR_dist (see below)

U_D_AR_dist=zeros(c_max,1); %Distance between nodes D and AR
theta_D_prime=zeros(c_max,1); %Angle between 'actual' surface departing vel. vector and that arriving to node D
theta_D_1=zeros(c_max,1); %theta_D only for checking in c=1...cmax loop
ds_pre=zeros(c_max,1); %See below; used in obtaining min_ds

I_min_ds=zeros(b_max,1); %Index of min_ds
min_ds=zeros(b_max,1); %Min spacing, after 2nd strk. for each node AR (not necessarily small enough to include 2nd strk. yet)

for b=1:1:b_max %Check each D node
  if (((xz_surf(b,1)~=0)&&(xz_surf(b,2)~=0))||((alfa==(90*pi/180))&&((b==1)||(b==(i_max+1))||(b==(2*i_max+1)))&&(xz_surf(b,1)==0)&&(xz_surf(b,2)~=0)))...
      &&(xz_surf(b,2)>(h*sin(alfa)-H_m+dz/2)) %Check if surf. found (numerically) AND if surf. is not top of mask

    for c=1:1:c_max %Check AR nodes for each D node
        if (NaN_Chk_theta_AR(c)==0)&&(xz_surf(c,2)>(h*sin(alfa)-
H_m+dz/2))&&(b~=c)&&((theta_AR(c)<(pi/2))&&(theta_AR(c)>0))&&((theta_DE(c)<pi)&&(theta_DE(c)>(pi/2)))...
&&(((gamma_AR(c)>(pi/2))&&(x_prime_surf_AR(b)>x_prime_surf_AR(c)))||((gamma_AR(c)<(pi/2))&&(x_prime_surf_AR(c)>x_prime_surf_AR(b))))
          %Check if surf. found (numerically)-this check supercedes next checks; if surf is not top of mask; ignore check at node D=AR;
          %limit range of theta_AR; limit range of theta_DE; check if rebound direction makes sense

          %Calculate theta_D' and theta_D
          theta_D_prime(c)=acos(((xz_surf(b,1)-xz_surf(c,1)).*(phi_x_star_AR(c))+(xz_surf(b,2)-xz_surf(c,2)).*(phi_z_star_AR(c)))./...
            (sqrt((xz_surf(b,1)-xz_surf(c,1)).^2+(xz_surf(b,2)-xz_surf(c,2)).^2).*sqrt((phi_x_star_AR(c)).^2+(phi_z_star_AR(c)).^2)));

          theta_D_1(c)=acos(((xz_surf(b,1)-xz_surf(c,1)).*(phi_x_star_AR(b))+(xz_surf(b,2)-xz_surf(c,2)).*(phi_z_star_AR(b)))./...
            (sqrt((xz_surf(b,1)-xz_surf(c,1)).^2+(xz_surf(b,2)-xz_surf(c,2)).^2).*sqrt((phi_x_star_AR(b)).^2+(phi_z_star_AR(b)).^2)));

          %Calculate distance between nodes AR and D
          U_D_AR_dist(c)=sqrt((xz_surf(b,1)-xz_surf(c,1)).^2+(xz_surf(b,2)-xz_surf(c,2)).^2);

          if ((theta_D_prime(c)>(pi/2))&&(theta_D_1(c)<(pi/2))) %If node D 'seen' by node AR
            ds_pre(c)=U_D_AR_dist(c).*tan(abs(theta_D_prime(c)-theta_DE(c)));
          else
            ds_pre(c)=NaN;
          end

        else
          ds_pre(c)=NaN;
        end
    end
     [min_ds(b),I_min_ds(b)]=min(ds_pre); %Find min_ds, Will ignore NaN's

     if (I_min_ds(b)~=0)&&(min_ds(b)<ds_crit)&&(U_D_AR_dist(I_min_ds(b))>(No_ds_cr*ds_crit))
       %Check if possibility of 2nd strike even occurred, ds_min<ds_crit and if U_D_AR_dist is large enough

                %Calculate 2nd strike values - f_v_AR_fin, v_AR,
                %flux_AR, assign calc'd theta_D

                %f_v_AR_fin
                if (xz_surf(b,2)>=(h*sin(alfa))) %T
                   f_v_AR(b)=f_v_AR_T;
                else %M
                   f_v_AR(b)=f_v_AR_M;
                end
                f_v_AR_fin(b)=f_v_AR(b).*((ds_crit-min_ds(b))./ds_crit);

                %Define particle velocity at AR node
                v_AR(b)=v_o*(1-H_slp*(((xz_surf(I_min_ds(b),1)-x_off)*sin(alfa)-xz_surf(I_min_ds(b),2)*cos(alfa)).^2+(y_mean).^2).^0.5./...
                  ((xz_surf(I_min_ds(b),1)-x_off)*cos(alfa)+xz_surf(I_min_ds(b),2)*sin(alfa)));
                if (v_AR(b)<0)
                v_AR(b)=0;
                end

                %Define particle mass flux at AR node
                flux_AR(b)=(MFR/pi)*(beta./((xz_surf(I_min_ds(b),1)-x_off)*cos(alfa)+xz_surf(I_min_ds(b),2)*sin(alfa))).^2....
                *exp(-(beta^2.*(((xz_surf(I_min_ds(b),1)-x_off)*sin(alfa)-xz_surf(I_min_ds(b),2)*cos(alfa)).^2+(y_mean).^2)./...
                ((xz_surf(I_min_ds(b),1)-x_off)*cos(alfa)+xz_surf(I_min_ds(b),2)*sin(alfa)).^2));

                %theta_D
                theta_D(b)=acos(((xz_surf(b,1)-xz_surf(I_min_ds(b),1)).*(phi_x_star_AR(b))+(xz_surf(b,2)-xz_surf(I_min_ds(b),2)).*(phi_z_star_AR(b)))./...
                         (sqrt((xz_surf(b,1)-xz_surf(I_min_ds(b),1)).^2+(xz_surf(b,2)-
xz_surf(I_min_ds(b),2)).^2).*sqrt((phi_x_star_AR(b)).^2+(phi_z_star_AR(b)).^2)));
                       theta_D_deg(b)=theta_D(b)*(180/pi);

                       %Calculate +/-,c,* cos_theta_D for F_ext Algorithm (Note, if 2nd strike not called
                       %up, cos_thetas will remain 0 from pre-allocation, so F_2nd=0 in Fext


                       cos_t_pfx_pfz_D(b)=((xz_surf(b,1)-xz_surf(I_min_ds(b),1)).*(phi_x_pos_AR(b))+(xz_surf(b,2)-
xz_surf(I_min_ds(b),2)).*(phi_z_pos_AR(b)))./...
                         (sqrt((xz_surf(b,1)-xz_surf(I_min_ds(b),1)).^2+(xz_surf(b,2)-
xz_surf(I_min_ds(b),2)).^2).*sqrt((phi_x_pos_AR(b)).^2+(phi_z_pos_AR(b)).^2));

                       cos_t_pfx_nfz_D(b)=((xz_surf(b,1)-xz_surf(I_min_ds(b),1)).*(phi_x_pos_AR(b))+(xz_surf(b,2)-
xz_surf(I_min_ds(b),2)).*(phi_z_neg_AR(b)))./...
                         (sqrt((xz_surf(b,1)-xz_surf(I_min_ds(b),1)).^2+(xz_surf(b,2)-
xz_surf(I_min_ds(b),2)).^2).*sqrt((phi_x_pos_AR(b)).^2+(phi_z_neg_AR(b)).^2));

                       cos_t_nfx_pfz_D(b)=((xz_surf(b,1)-xz_surf(I_min_ds(b),1)).*(phi_x_neg_AR(b))+(xz_surf(b,2)-
xz_surf(I_min_ds(b),2)).*(phi_z_pos_AR(b)))./...
                         (sqrt((xz_surf(b,1)-xz_surf(I_min_ds(b),1)).^2+(xz_surf(b,2)-
xz_surf(I_min_ds(b),2)).^2).*sqrt((phi_x_neg_AR(b)).^2+(phi_z_pos_AR(b)).^2));

                       cos_t_nfx_nfz_D(b)=((xz_surf(b,1)-xz_surf(I_min_ds(b),1)).*(phi_x_neg_AR(b))+(xz_surf(b,2)-
xz_surf(I_min_ds(b),2)).*(phi_z_neg_AR(b)))./...
                         (sqrt((xz_surf(b,1)-xz_surf(I_min_ds(b),1)).^2+(xz_surf(b,2)-
xz_surf(I_min_ds(b),2)).^2).*sqrt((phi_x_neg_AR(b)).^2+(phi_z_neg_AR(b)).^2));

                       cos_t_star_D(b)=cos(theta_D(b)); %Calculated above already

                       cos_t_cen_D(b)=((xz_surf(b,1)-xz_surf(I_min_ds(b),1)).*(phi_x_cen_AR(b))+(xz_surf(b,2)-xz_surf(I_min_ds(b),2)).*(phi_z_cen_AR(b)))./...
                         (sqrt((xz_surf(b,1)-xz_surf(I_min_ds(b),1)).^2+(xz_surf(b,2)-
xz_surf(I_min_ds(b),2)).^2).*sqrt((phi_x_cen_AR(b)).^2+(phi_z_cen_AR(b)).^2));

          else
            f_v_AR_fin(b)=0;
            v_AR(b)=0;
            flux_AR(b)=0;
            theta_D(b)=NaN;
            theta_D_deg(b)=NaN;
          end

      else
        f_v_AR_fin(b)=0;
        v_AR(b)=0;
        flux_AR(b)=0;
        theta_D(b)=NaN;
        theta_D_deg(b)=NaN;
      end

end
NaN_Chk_theta_D=isnan(theta_D); %If any entries are NaN, returns 1 for that entry, else 0

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%END of 2nd Strike Algorithm %%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
        %START OF SDF AND F_EXT (Extension Velocity) ALGORITHM
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%RE-initialize SDF (update phi)

%Initialize SDF (NOTE: x_surf and z_surf entries should never be 0 in
%reality since in array it is 0 numerically if NO surface is encountered for that
%row or column of phi's)
SDF=zeros(b_max,1);
%Initialize F_ext's and corresponding cos(theta)'s
cos_t_pfx_pfz_ext=zeros(k_max,i_max);
cos_t_pfx_nfz_ext=zeros(k_max,i_max);
cos_t_nfx_pfz_ext=zeros(k_max,i_max);
cos_t_nfx_nfz_ext=zeros(k_max,i_max);
cos_t_star_ext=zeros(k_max,i_max);
cos_t_cen_ext=zeros(k_max,i_max);
F_ext_pfx_pfz=zeros(k_max,i_max);
F_ext_pfx_nfz=zeros(k_max,i_max);
F_ext_nfx_pfz=zeros(k_max,i_max);
F_ext_nfx_nfz=zeros(k_max,i_max);
F_ext_star=zeros(k_max,i_max);
F_ext_cen=zeros(k_max,i_max);
%Initialize dphi/dx,dphi/dz (FD's)
```

```
phi_x_pos_ext=zeros(k_max,i_max);
phi_x_neg_ext=zeros(k_max,i_max);
phi_x_cen_ext=zeros(k_max,i_max);
phi_z_pos_ext=zeros(k_max,i_max);
phi_z_neg_ext=zeros(k_max,i_max);
phi_z_cen_ext=zeros(k_max,i_max);
phi_x_star_ext=zeros(k_max,i_max);
phi_z_star_ext=zeros(k_max,i_max);
%Initialize Erosive Power and Masking Function Properties - Extended
x_prime_ext=zeros(k_max,i_max);
z_prime_ext=zeros(k_max,i_max);
L_mask_ext=zeros(k_max,i_max);
x_lim_ext=zeros(k_max,i_max);
M_r_x_prime_ext=zeros(k_max,i_max);
Eros_pow_ext=zeros(k_max,i_max);
v_ext=zeros(k_max,i_max);
flux_ext=zeros(k_max,i_max);
Int_P_r_x_prime_ext=zeros(k_max,i_max);
Int_P_r_L_mask_ext=zeros(k_max,i_max);
J_k_ext=0; %Initialize (see below)
J_i_ext=0;
I_min=zeros(k_max,i_max); %Index of SDF (see below)

%2nd strike erosive power
Eros_pow_ext_2nd=zeros(k_max,i_max);
%Initial strike F_ext's
F_ext_pfx_pfz_1st=zeros(k_max,i_max);
F_ext_pfx_nfz_1st=zeros(k_max,i_max);
F_ext_nfx_pfz_1st=zeros(k_max,i_max);
F_ext_nfx_nfz_1st=zeros(k_max,i_max);
F_ext_star_1st=zeros(k_max,i_max);
F_ext_cen_1st=zeros(k_max,i_max);

for k=1:1:k_max
  for i=1:1:i_max

    if ((flag_NB(k,i)==1)||(flag_NB(k,i)==2)) %#####*****#####$$$$$#####****##### IF IN NB
      %NOTE, no BC's (i.e. where flag_NB==2) need to be specified since we
      %calculate SDF and Fext only for grid pts IN the band (i.e., flag_NB==1) and all calculations are based on surface points
      % which are always in the band. flag_NB==2 comes into play when the EOM
      % is solved in the while loop: they must be specified at the
      % beginning of each loop.  There is no issue here since band
      % is re-initialized before boundary (where flag_NB==2) is hit
      % so surface never reaches boundary so no checks need to be
      % performed where boundary phi_x or phi_z would have free end B.C.'s.

    for b=1:1:b_max
      if (phi_1(k,i)==0)
        SDF(b)=0; %we are on the surface
      elseif
((xz_surf(b,1)~=0)&&(xz_surf(b,2)~=0))||((alfa==(90*pi/180))&&((b==1)||(b==(i_max+1))||(b==(2*i_max+1)))&&(xz_surf(b,1)==0)&&(xz_surf(b,2)~=0))
        SDF(b)=((x_cord(i)-xz_surf(b,1)).^2+(z_cord(k)-xz_surf(b,2)).^2).^0.5;
      else
        SDF(b)=NaN; %Need this since it accounts for cases where surface wasn't encountered (i.e. x_surf or z_surf are 0 numerically)
      end           %and the fact that SDF can be actually 0, if we are on the surface (and if surface not encountered, SDF(b) would stay 0
              %since it is pre-allocated with zeros for speed
    end
    %Obtain value and index at which SDF is MIN (ignores NaN's)
    [min_SDF,I_min(k,i)]=min(SDF);

    %Update phi
    if (phi_1(k,i)>0)          %Checking sign of pre re-initialized phi's and thus new phi's (no change in sign)
        phi(k,i)=min_SDF;
    elseif (phi_1(k,i)<0)
        phi(k,i)=-min_SDF;
    else
        phi(k,i)=phi_1(k,i); %i.e., phi(k,i)=0, we are on the surface
    end


    %--------------------------------------------------------------
    %%%%%%%%%%%%%%%%%%%%%%F_ext Algorithm%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    %--------------------------------------------------------------

    if (I_min(k,i)>=1)&&(I_min(k,i)<=(3*i_max))&&(phi_1(k,i)~=0) %zsurf used; phi_1(k,i)~=0, since if it is 0,
      %use regular eq'ns to calculate Fext where surface node = grid node (see below)

     if (I_min(k,i)>=1)&&(I_min(k,i)<=i_max)  %%Here, we use I_min_ik="i", acts as reference for phi interpolation to know "i" value (see below)
       I_min_ik=I_min(k,i); %z_surf_1 used
     end
```

```
        if (I_min(k,i)>=(i_max+1))&&(I_min(k,i)<=(2*i_max))
          I_min_ik=I_min(k,i)-i_max;  %z_surf_2 used
        end

        if (I_min(k,i)>=(2*i_max+1))&&(I_min(k,i)<=(3*i_max))
          I_min_ik=I_min(k,i)-2*i_max;  %z_surf_3 used
        end

          %Calculate nearest (reference) k index to surface
          J_k_ext=floor(k_max-(xz_surf(I_min(k,i),2)-z_min_grid)/dz);
%_______________________________________________________________________
          %Calculation of dphi/dx,dphi/dz (in b/w grid nodes); Define BC's

          if I_min_ik==i_max
             phi_x_pos_ext(k,i)=0;
          else
             phi_B_z_surf=((xz_surf(I_min(k,i),2)-z_cord(J_k_ext))/(z_cord(J_k_ext+1)-z_cord(J_k_ext))).*(phi_1(J_k_ext+1,I_min_ik+1)-
phi_1(J_k_ext,I_min_ik+1))+phi_1(J_k_ext,I_min_ik+1);
             phi_x_pos_ext(k,i)=(phi_B_z_surf-0)./dx;
          end

          if I_min_ik==1
             phi_x_neg_ext(k,i)=0;
          else
             phi_A_z_surf=((xz_surf(I_min(k,i),2)-z_cord(J_k_ext))/(z_cord(J_k_ext+1)-z_cord(J_k_ext))).*(phi_1(J_k_ext+1,I_min_ik-1)-phi_1(J_k_ext,I_min_ik-
1))+phi_1(J_k_ext,I_min_ik-1);
             phi_x_neg_ext(k,i)=(0-phi_A_z_surf)./dx;
          end

          if (I_min_ik==i_max)||(I_min_ik==1)
             phi_x_cen_ext(k,i)=0;
          else
             phi_A_z_surf=((xz_surf(I_min(k,i),2)-z_cord(J_k_ext))/(z_cord(J_k_ext+1)-z_cord(J_k_ext))).*(phi_1(J_k_ext+1,I_min_ik-1)-phi_1(J_k_ext,I_min_ik-
1))+phi_1(J_k_ext,I_min_ik-1);
             phi_B_z_surf=((xz_surf(I_min(k,i),2)-z_cord(J_k_ext))/(z_cord(J_k_ext+1)-z_cord(J_k_ext))).*(phi_1(J_k_ext+1,I_min_ik+1)-
phi_1(J_k_ext,I_min_ik+1))+phi_1(J_k_ext,I_min_ik+1);
             phi_x_cen_ext(k,i)=(phi_B_z_surf-phi_A_z_surf)./(2*dx);
          end

          if J_k_ext==1
             phi_z_pos_ext(k,i)=0;
          else
             phi_C_z_surf=((xz_surf(I_min(k,i),2)+dz-z_cord(J_k_ext-1))/(z_cord(J_k_ext)-z_cord(J_k_ext-1))).*(phi_1(J_k_ext,I_min_ik)-phi_1(J_k_ext-
1,I_min_ik))+phi_1(J_k_ext-1,I_min_ik);
             phi_z_pos_ext(k,i)=(phi_C_z_surf-0)./dz;
          end

          if J_k_ext==k_max
             phi_z_neg_ext(k,i)=0;
          else
             phi_D_z_surf=((xz_surf(I_min(k,i),2)-dz-z_cord(J_k_ext+1))/(z_cord(J_k_ext+2)-z_cord(J_k_ext+1))).*(phi_1(J_k_ext+2,I_min_ik)-
phi_1(J_k_ext+1,I_min_ik))+phi_1(J_k_ext+1,I_min_ik);
             phi_z_neg_ext(k,i)=(0-phi_D_z_surf)./dz;
          end

          if (J_k_ext==k_max)||(J_k_ext==1)
             phi_z_cen_ext(k,i)=0;
          else
             phi_C_z_surf=((xz_surf(I_min(k,i),2)+dz-z_cord(J_k_ext-1))/(z_cord(J_k_ext)-z_cord(J_k_ext-1))).*(phi_1(J_k_ext,I_min_ik)-phi_1(J_k_ext-
1,I_min_ik))+phi_1(J_k_ext-1,I_min_ik);
             phi_D_z_surf=((xz_surf(I_min(k,i),2)-dz-z_cord(J_k_ext+1))/(z_cord(J_k_ext+2)-z_cord(J_k_ext+1))).*(phi_1(J_k_ext+2,I_min_ik)-
phi_1(J_k_ext+1,I_min_ik))+phi_1(J_k_ext+1,I_min_ik);
             phi_z_cen_ext(k,i)=(phi_C_z_surf-phi_D_z_surf)./(2*dz);
          end

          %Now calculate phi_stars
          phi_x_star_ext(k,i)=(phi_x_pos_ext(k,i)+phi_x_neg_ext(k,i))/2;
          phi_z_star_ext(k,i)=(phi_z_pos_ext(k,i)+phi_z_neg_ext(k,i))/2;
%_______________________________________________________________________
          %Masking Function for T and M
          x_prime_ext(k,i)=(xz_surf(I_min(k,i),1)-x_off).*sin(alfa)-xz_surf(I_min(k,i),2).*cos(alfa); %Rotated local x
          z_prime_ext(k,i)=(xz_surf(I_min(k,i),1)-x_off).*cos(alfa)+xz_surf(I_min(k,i),2).*sin(alfa); %Rotated local z

               if (xz_surf(I_min(k,i),2)>=(h*sin(alfa))) %T

               if (max_x_prime_surf_LM<0) %Case (a) and (c)
                 if (x_prime_ext(k,i)<0)
                    L_mask_ext(k,i)=z_prime_ext(k,i).*tan_fi_min;
                 else %i.e., when x_prime>=0
                    L_mask_ext(k,i)=z_prime_ext(k,i).*tan_fi_max;
                    end
                else %i.e., when x_m>=W_m/2 Case (b)
                    x_lim_ext(k,i)=z_prime_ext(k,i).*tan_fi_min;
                    if (x_prime_ext(k,i)<x_lim_ext(k,i))
                        L_mask_ext(k,i)=0;
                    else %i.e., when x_prime>=x_lim
                        L_mask_ext(k,i)=z_prime_ext(k,i).*tan_fi_max-x_lim_ext(k,i);
                    end
                end

                %Define proportion of mass of particles that pass through mask opening having a
                %specific particle size (of radius r) distribution
                if (abs(x_prime_ext(k,i))>=L_mask_ext(k,i))
                    M_r_x_prime_ext(k,i)=0;
                else
                    Int_P_r_x_prime_ext(k,i)=real(P_1-P_1*erf(P_2*log(L_mask_ext(k,i)-abs(x_prime_ext(k,i)))-P_3));
                    Int_P_r_L_mask_ext(k,i)=real(P_1-P_1*erf(P_2*log(L_mask_ext(k,i))-P_3));
                    M_r_x_prime_ext(k,i)=Int_P_r_x_prime_ext(k,i)./Int_P_r_L_mask_ext(k,i);
                end

                else  %M; Below for F, will check if M(x')=0 if cos(theta)<0 so apply it there (initial iteration assumption for small initial dt)
                    M_r_x_prime_ext(k,i)=1;
                end

%_____________________________________________________________________________
                %Define velocity v(x,z) at each grid node
                    v_ext(k,i)=v_o*(1-H_slp*(((xz_surf(I_min(k,i),1)-x_off)*sin(alfa)-xz_surf(I_min(k,i),2)*cos(alfa)).^2+(y_mean).^2).^0.5./((xz_surf(I_min(k,i),1)-x_off)*cos(alfa)+xz_surf(I_min(k,i),2)*sin(alfa)));
                if (v_ext(k,i)<0)
                    v_ext(k,i)=0;
                end

                %Define particle mass flux(x,z) at each grid node
                    flux_ext(k,i)=(MFR/pi)*(beta./((xz_surf(I_min(k,i),1)-x_off)*cos(alfa)+xz_surf(I_min(k,i),2)*sin(alfa))).^2....
                    *exp(-(beta^2.*(((xz_surf(I_min(k,i),1)-x_off)*sin(alfa)-xz_surf(I_min(k,i),2)*cos(alfa)).^2+(y_mean).^2)./((xz_surf(I_min(k,i),1)-x_off)*cos(alfa)+xz_surf(I_min(k,i),2)*sin(alfa)).^2));

                %Define Erosive Power Eros_pow(k,i) at each grid
                %node (1st Strike)
                if (xz_surf(I_min(k,i),2)>=(h*sin(alfa))) %T
                    Eros_pow_ext(k,i)=M_r_x_prime_ext(k,i).*v_ext(k,i).^k_vel.*flux_ext(k,i);
                else %M
                    Eros_pow_ext(k,i)=M_r_x_prime_ext(k,i).*v_ext(k,i).^k_vel_M.*flux_ext(k,i);
                end

                %Define Erosive Power for 2nd strike
                %NOTE: No Mask here
                if (xz_surf(I_min(k,i),2)>=(h*sin(alfa))) %T
                    if (NaN_Chk_theta_D(I_min(k,i))==0) %If second strike occurred
                        Eros_pow_ext_2nd(k,i)=(f_v_AR_fin(I_min(k,i)).*v_AR(I_min(k,i))).^k_vel.*flux_AR(I_min(k,i));
                    else
                        Eros_pow_ext_2nd(k,i)=0;
                    end
                else %M
                    if (NaN_Chk_theta_D(I_min(k,i))==0)
                        Eros_pow_ext_2nd(k,i)=(f_v_AR_fin(I_min(k,i)).*v_AR(I_min(k,i))).^k_vel_M.*flux_AR(I_min(k,i));
                    else
                        Eros_pow_ext_2nd(k,i)=0;
                    end
                end

%_____________________________________________________________________________
        %Calculate F_extensions

            cos_t_pfx_pfz_ext(k,i)=((xz_surf(I_min(k,i),1)-x_off).*(phi_x_pos_ext(k,i))+xz_surf(I_min(k,i),2).*(phi_z_pos_ext(k,i)))./sqrt((xz_surf(I_min(k,i),1)-x_off).^2+xz_surf(I_min(k,i),2).^2);
            cos_t_pfx_nfz_ext(k,i)=((xz_surf(I_min(k,i),1)-x_off).*(phi_x_pos_ext(k,i))+xz_surf(I_min(k,i),2).*(phi_z_neg_ext(k,i)))./sqrt((xz_surf(I_min(k,i),1)-x_off).^2+xz_surf(I_min(k,i),2).^2);
            cos_t_nfx_pfz_ext(k,i)=((xz_surf(I_min(k,i),1)-x_off).*(phi_x_neg_ext(k,i))+xz_surf(I_min(k,i),2).*(phi_z_pos_ext(k,i)))./sqrt((xz_surf(I_min(k,i),1)-x_off).^2+xz_surf(I_min(k,i),2).^2);
            cos_t_nfx_nfz_ext(k,i)=((xz_surf(I_min(k,i),1)-x_off).*(phi_x_neg_ext(k,i))+xz_surf(I_min(k,i),2).*(phi_z_neg_ext(k,i)))./sqrt((xz_surf(I_min(k,i),1)-x_off).^2+xz_surf(I_min(k,i),2).^2);
            cos_t_star_ext(k,i)=((xz_surf(I_min(k,i),1)-x_off).*(phi_x_star_ext(k,i))+xz_surf(I_min(k,i),2).*(phi_z_star_ext(k,i)))./sqrt((xz_surf(I_min(k,i),1)-x_off).^2+xz_surf(I_min(k,i),2).^2);
            cos_t_cen_ext(k,i)=((xz_surf(I_min(k,i),1)-x_off).*(phi_x_cen_ext(k,i))+xz_surf(I_min(k,i),2).*(phi_z_cen_ext(k,i)))./sqrt((xz_surf(I_min(k,i),1)-x_off).^2+xz_surf(I_min(k,i),2).^2);
            if (cos_t_pfx_pfz_ext(k,i)>1) %limit cos(theta) to be b/w -1 and 1
                cos_t_pfx_pfz_ext(k,i)=1;
            end
            if (cos_t_pfx_nfz_ext(k,i)>1)
                cos_t_pfx_nfz_ext(k,i)=1;
            end
            if (cos_t_nfx_pfz_ext(k,i)>1)
                cos_t_nfx_pfz_ext(k,i)=1;
            end
            if (cos_t_nfx_nfz_ext(k,i)>1)
                cos_t_nfx_nfz_ext(k,i)=1;
            end
            if (cos_t_star_ext(k,i)>1)
                cos_t_star_ext(k,i)=1;
            end
            if (cos_t_cen_ext(k,i)>1)
                cos_t_cen_ext(k,i)=1;
            end

            if (xz_surf(I_min(k,i),2)>=(h*sin(alfa))) %T
                F_ext_pfx_pfz_1st(k,i)=real((C/rho_s)*Eros_pow_ext(k,i).*((cos_t_pfx_pfz_ext(k,i))^(k_vel+1)));
            else %M
                if (cos_t_pfx_pfz_ext(k,i)<=0) %Apply mask visibility for M
                F_ext_pfx_pfz_1st(k,i)=0;
                else
                F_ext_pfx_pfz_1st(k,i)=real((C_M/rho_s_M).*Eros_pow_ext(k,i).*((cos_t_pfx_pfz_ext(k,i)).^n_1_M).*((1+H_vic_M*(1-
cos_t_pfx_pfz_ext(k,i))).^n_2_M));
                end
            end

            %Note: cos_theta_D = 0 if no 2nd strk, so OK (no NaNs)
            if (xz_surf(I_min(k,i),2)>=(h*sin(alfa))) %T
                F_ext_pfx_pfz(k,i)=F_ext_pfx_pfz_1st(k,i)+real((C/rho_s)*Eros_pow_ext_2nd(k,i).*((cos_t_pfx_pfz_D(I_min(k,i)))^(k_vel+1)));
            else %M

F_ext_pfx_pfz(k,i)=F_ext_pfx_pfz_1st(k,i)+real((C_M/rho_s_M).*Eros_pow_ext_2nd(k,i).*((cos_t_pfx_pfz_D(I_min(k,i))).^n_1_M).*((1+H_vic_M*(1-
cos_t_pfx_pfz_D(I_min(k,i)))).^n_2_M));
            end

            if (xz_surf(I_min(k,i),2)>=(h*sin(alfa))) %T
                F_ext_pfx_nfz_1st(k,i)=real((C/rho_s)*Eros_pow_ext(k,i).*((cos_t_pfx_nfz_ext(k,i))^(k_vel+1)));
            else %M
                if (cos_t_pfx_nfz_ext(k,i)<=0) %Apply mask visibility for M
                F_ext_pfx_nfz_1st(k,i)=0;
                else
                F_ext_pfx_nfz_1st(k,i)=real((C_M/rho_s_M).*Eros_pow_ext(k,i).*((cos_t_pfx_nfz_ext(k,i)).^n_1_M).*((1+H_vic_M*(1-
cos_t_pfx_nfz_ext(k,i))).^n_2_M));
                end
            end

            %Note: cos_theta_D = 0 if no 2nd strk, so OK (no NaNs)
            if (xz_surf(I_min(k,i),2)>=(h*sin(alfa))) %T
                F_ext_pfx_nfz(k,i)=F_ext_pfx_nfz_1st(k,i)+real((C/rho_s)*Eros_pow_ext_2nd(k,i).*((cos_t_pfx_nfz_D(I_min(k,i)))^(k_vel+1)));
            else %M

F_ext_pfx_nfz(k,i)=F_ext_pfx_nfz_1st(k,i)+real((C_M/rho_s_M).*Eros_pow_ext_2nd(k,i).*((cos_t_pfx_nfz_D(I_min(k,i))).^n_1_M).*((1+H_vic_M*(1-
cos_t_pfx_nfz_D(I_min(k,i)))).^n_2_M));
            end

            if (xz_surf(I_min(k,i),2)>=(h*sin(alfa))) %T
                F_ext_nfx_pfz_1st(k,i)=real((C/rho_s)*Eros_pow_ext(k,i).*((cos_t_nfx_pfz_ext(k,i))^(k_vel+1)));
            else %M
                if (cos_t_nfx_pfz_ext(k,i)<=0) %Apply mask visibility for M
                F_ext_nfx_pfz_1st(k,i)=0;
                else
                F_ext_nfx_pfz_1st(k,i)=real((C_M/rho_s_M).*Eros_pow_ext(k,i).*((cos_t_nfx_pfz_ext(k,i)).^n_1_M).*((1+H_vic_M*(1-
cos_t_nfx_pfz_ext(k,i))).^n_2_M));
                end
            end

            %Note: cos_theta_D = 0 if no 2nd strk, so OK (no NaNs)
            if (xz_surf(I_min(k,i),2)>=(h*sin(alfa))) %T
                F_ext_nfx_pfz(k,i)=F_ext_nfx_pfz_1st(k,i)+real((C/rho_s)*Eros_pow_ext_2nd(k,i).*((cos_t_nfx_pfz_D(I_min(k,i)))^(k_vel+1)));
            else %M

F_ext_nfx_pfz(k,i)=F_ext_nfx_pfz_1st(k,i)+real((C_M/rho_s_M).*Eros_pow_ext_2nd(k,i).*((cos_t_nfx_pfz_D(I_min(k,i))).^n_1_M).*((1+H_vic_M*(1-
cos_t_nfx_pfz_D(I_min(k,i)))).^n_2_M));
            end

            if (xz_surf(I_min(k,i),2)>=(h*sin(alfa))) %T
                F_ext_nfx_nfz_1st(k,i)=real((C/rho_s)*Eros_pow_ext(k,i).*((cos_t_nfx_nfz_ext(k,i))^(k_vel+1)));
            else %M
                if (cos_t_nfx_nfz_ext(k,i)<=0) %Apply mask visibility for M
                F_ext_nfx_nfz_1st(k,i)=0;
```

```
                else
                F_ext_nfx_nfz_1st(k,i)=real((C_M/rho_s_M).*Eros_pow_ext(k,i).*((cos_t_nfx_nfz_ext(k,i)).^n_1_M).*((1+H_vic_M*(1-
cos_t_nfx_nfz_ext(k,i))).^n_2_M));
                end
            end

            %Note: cos_theta_D = 0 if no 2nd strk, so OK (no NaNs)
            if (xz_surf(I_min(k,i),2)>=(h*sin(alfa))) %T
                F_ext_nfx_nfz(k,i)=F_ext_nfx_nfz_1st(k,i)+real((C/rho_s)*Eros_pow_ext_2nd(k,i).*((cos_t_nfx_nfz_D(I_min(k,i)))^(k_vel+1)));
            else %M

F_ext_nfx_nfz(k,i)=F_ext_nfx_nfz_1st(k,i)+real((C_M/rho_s_M).*Eros_pow_ext_2nd(k,i).*((cos_t_nfx_nfz_D(I_min(k,i))).^n_1_M).*((1+H_vic_M*(1-
cos_t_nfx_nfz_D(I_min(k,i)))).^n_2_M));
            end

            if (xz_surf(I_min(k,i),2)>=(h*sin(alfa))) %T
                F_ext_star_1st(k,i)=real((C/rho_s)*Eros_pow_ext(k,i).*((cos_t_star_ext(k,i))^(k_vel+1)));
            else %M
                if (cos_t_star_ext(k,i)<=0) %Apply mask visibility for M
                F_ext_star_1st(k,i)=0;
                else
                F_ext_star_1st(k,i)=real((C_M/rho_s_M).*Eros_pow_ext(k,i).*((cos_t_star_ext(k,i)).^n_1_M).*((1+H_vic_M*(1-cos_t_star_ext(k,i))).^n_2_M));
                end
            end

            %Note: cos_theta_D = 0 if no 2nd strk, so OK (no NaNs)
            if (xz_surf(I_min(k,i),2)>=(h*sin(alfa))) %T
                F_ext_star(k,i)=F_ext_star_1st(k,i)+real((C/rho_s)*Eros_pow_ext_2nd(k,i).*((cos_t_star_D(I_min(k,i)))^(k_vel+1)));
            else %M
                F_ext_star(k,i)=F_ext_star_1st(k,i)+real((C_M/rho_s_M).*Eros_pow_ext_2nd(k,i).*((cos_t_star_D(I_min(k,i))).^n_1_M).*((1+H_vic_M*(1-
cos_t_star_D(I_min(k,i)))).^n_2_M));
            end

            if (xz_surf(I_min(k,i),2)>=(h*sin(alfa))) %T
                F_ext_cen_1st(k,i)=real((C/rho_s)*Eros_pow_ext(k,i).*((cos_t_cen_ext(k,i))^(k_vel+1)));
            else %M
                if (cos_t_cen_ext(k,i)<=0) %Apply mask visibility for M
                F_ext_cen_1st(k,i)=0;
                else
                F_ext_cen_1st(k,i)=real((C_M/rho_s_M).*Eros_pow_ext(k,i).*((cos_t_cen_ext(k,i)).^n_1_M).*((1+H_vic_M*(1-cos_t_cen_ext(k,i))).^n_2_M));
                end
            end

            %Note: cos_theta_D = 0 if no 2nd strk, so OK (no NaNs)
            if (xz_surf(I_min(k,i),2)>=(h*sin(alfa))) %T
                F_ext_cen(k,i)=F_ext_cen_1st(k,i)+real((C/rho_s)*Eros_pow_ext_2nd(k,i).*((cos_t_cen_D(I_min(k,i)))^(k_vel+1)));
            else %M
                F_ext_cen(k,i)=F_ext_cen_1st(k,i)+real((C_M/rho_s_M).*Eros_pow_ext_2nd(k,i).*((cos_t_cen_D(I_min(k,i))).^n_1_M).*((1+H_vic_M*(1-
cos_t_cen_D(I_min(k,i)))).^n_2_M));
            end


    elseif (I_min(k,i)>=(3*i_max+1))&&(I_min(k,i)<=b_max)&&(phi_1(k,i)~=0) %xsurf used; phi_1(k,i)~=0 (see above)

        if (I_min(k,i)>=(3*i_max+1))&&(I_min(k,i)<=(3*i_max+k_max))  %%Here, we use I_min_ik="k", acts as reference for phi interpolation to know "k" value
(see below)
          I_min_ik=I_min(k,i)-3*i_max; %x_surf_1 used
        end
        if (I_min(k,i)>=(3*i_max+k_max+1))&&(I_min(k,i)<=b_max)
          I_min_ik=I_min(k,i)-3*i_max-k_max;  %x_surf_2 used
        end

          %Repeat above algorithm but for xsurf
          %Calculate nearest i index to surface
          J_i_ext=floor(1+(xz_surf(I_min(k,i),1)-x_min_grid)/dx);
%_____________________________________________________________________________
          %Calculation of dphi/dx,dphi/dz, Define BC's
          if J_i_ext==i_max
             phi_x_pos_ext(k,i)=0;
          else
             phi_D_x_surf=((xz_surf(I_min(k,i),1)+dx-x_cord(J_i_ext+1))/(x_cord(J_i_ext+2)-x_cord(J_i_ext+1))).*(phi_1(I_min_ik,J_i_ext+2)-
phi_1(I_min_ik,J_i_ext+1))+phi_1(I_min_ik,J_i_ext+1);
             phi_x_pos_ext(k,i)=(phi_D_x_surf-0)./dx;
          end

          if J_i_ext==1
             phi_x_neg_ext(k,i)=0;
          else
             phi_C_x_surf=((xz_surf(I_min(k,i),1)-dx-x_cord(J_i_ext-1))/(x_cord(J_i_ext)-x_cord(J_i_ext-1))).*(phi_1(I_min_ik,J_i_ext)-phi_1(I_min_ik,J_i_ext-
1))+phi_1(I_min_ik,J_i_ext-1);
```

```
          phi_x_neg_ext(k,i)=(0-phi_C_x_surf)./dx;
        end

        if (J_i_ext==i_max)||(J_i_ext==1)
          phi_x_cen_ext(k,i)=0;
        else
          phi_C_x_surf=((xz_surf(I_min(k,i),1)-dx-x_cord(J_i_ext-1))/(x_cord(J_i_ext)-x_cord(J_i_ext-1))).*(phi_1(I_min_ik,J_i_ext)-phi_1(I_min_ik,J_i_ext-
1))+phi_1(I_min_ik,J_i_ext-1);
          phi_D_x_surf=((xz_surf(I_min(k,i),1)+dx-x_cord(J_i_ext+1))/(x_cord(J_i_ext+2)-x_cord(J_i_ext+1))).*(phi_1(I_min_ik,J_i_ext+2)-
phi_1(I_min_ik,J_i_ext+1))+phi_1(I_min_ik,J_i_ext+1);
          phi_x_cen_ext(k,i)=(phi_D_x_surf-phi_C_x_surf)./(2*dx);
        end

        if I_min_ik==1
          phi_z_pos_ext(k,i)=0;
        else
          phi_A_x_surf=((xz_surf(I_min(k,i),1)-x_cord(J_i_ext))/(x_cord(J_i_ext+1)-x_cord(J_i_ext))).*(phi_1(I_min_ik-1,J_i_ext+1)-phi_1(I_min_ik-
1,J_i_ext))+phi_1(I_min_ik-1,J_i_ext);
          phi_z_pos_ext(k,i)=(phi_A_x_surf-0)./dz;
        end

        if I_min_ik==k_max
          phi_z_neg_ext(k,i)=0;
        else
          phi_B_x_surf=((xz_surf(I_min(k,i),1)-x_cord(J_i_ext))/(x_cord(J_i_ext+1)-x_cord(J_i_ext))).*(phi_1(I_min_ik+1,J_i_ext+1)-
phi_1(I_min_ik+1,J_i_ext))+phi_1(I_min_ik+1,J_i_ext);
          phi_z_neg_ext(k,i)=(0-phi_B_x_surf)./dz;
        end

        if (I_min_ik==k_max)||(I_min_ik==1)
          phi_z_cen_ext(k,i)=0;
        else
          phi_A_x_surf=((xz_surf(I_min(k,i),1)-x_cord(J_i_ext))/(x_cord(J_i_ext+1)-x_cord(J_i_ext))).*(phi_1(I_min_ik-1,J_i_ext+1)-phi_1(I_min_ik-
1,J_i_ext))+phi_1(I_min_ik-1,J_i_ext);
          phi_B_x_surf=((xz_surf(I_min(k,i),1)-x_cord(J_i_ext))/(x_cord(J_i_ext+1)-x_cord(J_i_ext))).*(phi_1(I_min_ik+1,J_i_ext+1)-
phi_1(I_min_ik+1,J_i_ext))+phi_1(I_min_ik+1,J_i_ext);
          phi_z_cen_ext(k,i)=(phi_A_x_surf-phi_B_x_surf)./(2*dz);
        end

        %Calculate phi_stars
        phi_x_star_ext(k,i)=(phi_x_pos_ext(k,i)+phi_x_neg_ext(k,i))/2;
        phi_z_star_ext(k,i)=(phi_z_pos_ext(k,i)+phi_z_neg_ext(k,i))/2;
%_________________________________________________________________________
        %Masking function for T, M (analogous to above)
        x_prime_ext(k,i)=(xz_surf(I_min(k,i),1)-x_off).*sin(alfa)-xz_surf(I_min(k,i),2).*cos(alfa); %Rotated local x
        z_prime_ext(k,i)=(xz_surf(I_min(k,i),1)-x_off).*cos(alfa)+xz_surf(I_min(k,i),2).*sin(alfa); %Rotated local z

            if (xz_surf(I_min(k,i),2)>=(h*sin(alfa))) %T

            if (max_x_prime_surf_LM<0) %Case (a) and (c)
              if (x_prime_ext(k,i)<0)
                L_mask_ext(k,i)=z_prime_ext(k,i).*tan_fi_min;
              else %i.e., when x_prime>=0
                L_mask_ext(k,i)=z_prime_ext(k,i).*tan_fi_max;
              end
            else %i.e., when x_m>=W_m/2 Case (b)
              x_lim_ext(k,i)=z_prime_ext(k,i).*tan_fi_min;
              if (x_prime_ext(k,i)<x_lim_ext(k,i))
                L_mask_ext(k,i)=0;
              else %i.e., when x_prime>=x_lim
                L_mask_ext(k,i)=z_prime_ext(k,i).*tan_fi_max-x_lim_ext(k,i);
              end
            end

            %Define proportion of mass of particle that pass through mask opening having a
            %specific particle size (of radius r) distribution
            if (abs(x_prime_ext(k,i))>=L_mask_ext(k,i))
              M_r_x_prime_ext(k,i)=0;
            else
              Int_P_r_x_prime_ext(k,i)=real(P_1-P_1*erf(P_2*log(L_mask_ext(k,i)-abs(x_prime_ext(k,i)))-P_3));
              Int_P_r_L_mask_ext(k,i)=real(P_1-P_1*erf(P_2*log(L_mask_ext(k,i))-P_3));
              M_r_x_prime_ext(k,i)=Int_P_r_x_prime_ext(k,i)./Int_P_r_L_mask_ext(k,i);
            end

            else  %M
              M_r_x_prime_ext(k,i)=1;
            end

%_________________________________________________________________________
            %Define velocity v(x,z) at each grid node
```

```
              v_ext(k,i)=v_o*(1-H_slp*(((xz_surf(I_min(k,i),1)-x_off)*sin(alfa)-xz_surf(I_min(k,i),2)*cos(alfa)).^2+(y_mean).^2).^0.5./((xz_surf(I_min(k,i),1)-
x_off)*cos(alfa)+xz_surf(I_min(k,i),2)*sin(alfa)));
            if (v_ext(k,i)<0)
              v_ext(k,i)=0;
            end

            %Define particle mass flux(x,z) at each grid node
              flux_ext(k,i)=(MFR/pi)*(beta./((xz_surf(I_min(k,i),1)-x_off)*cos(alfa)+xz_surf(I_min(k,i),2)*sin(alfa))).^2....
              *exp(-(beta^2.*(((xz_surf(I_min(k,i),1)-x_off)*sin(alfa)-xz_surf(I_min(k,i),2)*cos(alfa)).^2+(y_mean).^2)./((xz_surf(I_min(k,i),1)-
x_off)*cos(alfa)+xz_surf(I_min(k,i),2)*sin(alfa)).^2));

            %Define Erosive Power Eros_pow(k,i) at each grid
            %node (1st Strike)
            if (xz_surf(I_min(k,i),2)>=(h*sin(alfa))) %T
              Eros_pow_ext(k,i)=M_r_x_prime_ext(k,i).*v_ext(k,i).^k_vel.*flux_ext(k,i);
            else %M
              Eros_pow_ext(k,i)=M_r_x_prime_ext(k,i).*v_ext(k,i).^k_vel_M.*flux_ext(k,i);
            end

            %Define Erosive Power for 2nd strike
            %NOTE: No Mask here
            if (xz_surf(I_min(k,i),2)>=(h*sin(alfa))) %T
              if (NaN_Chk_theta_D(I_min(k,i))==0)
                Eros_pow_ext_2nd(k,i)=(f_v_AR_fin(I_min(k,i)).*v_AR(I_min(k,i))).^k_vel.*flux_AR(I_min(k,i));
              else
                Eros_pow_ext_2nd(k,i)=0;
              end
            else %M
              if (NaN_Chk_theta_D(I_min(k,i))==0)
                Eros_pow_ext_2nd(k,i)=(f_v_AR_fin(I_min(k,i)).*v_AR(I_min(k,i))).^k_vel_M.*flux_AR(I_min(k,i));
              else
                Eros_pow_ext_2nd(k,i)=0;
              end
            end

%________________________________________________________________________
        %Calculate F_extensions

          cos_t_pfx_pfz_ext(k,i)=((xz_surf(I_min(k,i),1)-x_off).*(phi_x_pos_ext(k,i))+xz_surf(I_min(k,i),2).*(phi_z_pos_ext(k,i)))./sqrt((xz_surf(I_min(k,i),1)-
x_off).^2+xz_surf(I_min(k,i),2).^2);
          cos_t_pfx_nfz_ext(k,i)=((xz_surf(I_min(k,i),1)-x_off).*(phi_x_pos_ext(k,i))+xz_surf(I_min(k,i),2).*(phi_z_neg_ext(k,i)))./sqrt((xz_surf(I_min(k,i),1)-
x_off).^2+xz_surf(I_min(k,i),2).^2);
          cos_t_nfx_pfz_ext(k,i)=((xz_surf(I_min(k,i),1)-x_off).*(phi_x_neg_ext(k,i))+xz_surf(I_min(k,i),2).*(phi_z_pos_ext(k,i)))./sqrt((xz_surf(I_min(k,i),1)-
x_off).^2+xz_surf(I_min(k,i),2).^2);
          cos_t_nfx_nfz_ext(k,i)=((xz_surf(I_min(k,i),1)-x_off).*(phi_x_neg_ext(k,i))+xz_surf(I_min(k,i),2).*(phi_z_neg_ext(k,i)))./sqrt((xz_surf(I_min(k,i),1)-
x_off).^2+xz_surf(I_min(k,i),2).^2);
          cos_t_star_ext(k,i)=((xz_surf(I_min(k,i),1)-x_off).*(phi_x_star_ext(k,i))+xz_surf(I_min(k,i),2).*(phi_z_star_ext(k,i)))./sqrt((xz_surf(I_min(k,i),1)-
x_off).^2+xz_surf(I_min(k,i),2).^2);
          cos_t_cen_ext(k,i)=((xz_surf(I_min(k,i),1)-x_off).*(phi_x_cen_ext(k,i))+xz_surf(I_min(k,i),2).*(phi_z_cen_ext(k,i)))./sqrt((xz_surf(I_min(k,i),1)-
x_off).^2+xz_surf(I_min(k,i),2).^2);
          if (cos_t_pfx_pfz_ext(k,i)>1)
            cos_t_pfx_pfz_ext(k,i)=1;
          end
          if (cos_t_pfx_nfz_ext(k,i)>1)
            cos_t_pfx_nfz_ext(k,i)=1;
          end
          if (cos_t_nfx_pfz_ext(k,i)>1)
            cos_t_nfx_pfz_ext(k,i)=1;
          end
          if (cos_t_nfx_nfz_ext(k,i)>1)
            cos_t_nfx_nfz_ext(k,i)=1;
          end
          if (cos_t_star_ext(k,i)>1)
            cos_t_star_ext(k,i)=1;
          end
          if (cos_t_cen_ext(k,i)>1)
            cos_t_cen_ext(k,i)=1;
          end

          if (xz_surf(I_min(k,i),2)>=(h*sin(alfa))) %T
            F_ext_pfx_pfz_1st(k,i)=real((C/rho_s)*Eros_pow_ext(k,i).*((cos_t_pfx_pfz_ext(k,i))^(k_vel+1)));
          else %M
            if (cos_t_pfx_pfz_ext(k,i)<=0) %Apply mask visibility for M
            F_ext_pfx_pfz_1st(k,i)=0;
            else
            F_ext_pfx_pfz_1st(k,i)=real((C_M/rho_s_M).*Eros_pow_ext(k,i).*((cos_t_pfx_pfz_ext(k,i)).^n_1_M).*((1+H_vic_M*(1-
cos_t_pfx_pfz_ext(k,i))).^n_2_M));
            end
          end
          %Note: cos_theta_D = 0 if no 2nd strk, so OK (no NaNs)
          if (xz_surf(I_min(k,i),2)>=(h*sin(alfa))) %T
            F_ext_pfx_pfz(k,i)=F_ext_pfx_pfz_1st(k,i)+real((C/rho_s)*Eros_pow_ext_2nd(k,i).*((cos_t_pfx_pfz_D(I_min(k,i)))^(k_vel+1)));
          else %M

F_ext_pfx_pfz(k,i)=F_ext_pfx_pfz_1st(k,i)+real((C_M/rho_s_M).*Eros_pow_ext_2nd(k,i).*((cos_t_pfx_pfz_D(I_min(k,i))).^n_1_M).*((1+H_vic_M*(1-
cos_t_pfx_pfz_D(I_min(k,i)))).^n_2_M));
          end

          if (xz_surf(I_min(k,i),2)>=(h*sin(alfa))) %T
            F_ext_pfx_nfz_1st(k,i)=real((C/rho_s)*Eros_pow_ext(k,i).*((cos_t_pfx_nfz_ext(k,i))^(k_vel+1)));
          else %M
            if (cos_t_pfx_nfz_ext(k,i)<=0) %Apply mask visibility for M
            F_ext_pfx_nfz_1st(k,i)=0;
            else
            F_ext_pfx_nfz_1st(k,i)=real((C_M/rho_s_M).*Eros_pow_ext(k,i).*((cos_t_pfx_nfz_ext(k,i)).^n_1_M).*((1+H_vic_M*(1-
cos_t_pfx_nfz_ext(k,i))).^n_2_M));
            end
          end

          %Note: cos_theta_D = 0 if no 2nd strk, so OK (no NaNs)
          if (xz_surf(I_min(k,i),2)>=(h*sin(alfa))) %T
            F_ext_pfx_nfz(k,i)=F_ext_pfx_nfz_1st(k,i)+real((C/rho_s)*Eros_pow_ext_2nd(k,i).*((cos_t_pfx_nfz_D(I_min(k,i)))^(k_vel+1)));
          else %M

F_ext_pfx_nfz(k,i)=F_ext_pfx_nfz_1st(k,i)+real((C_M/rho_s_M).*Eros_pow_ext_2nd(k,i).*((cos_t_pfx_nfz_D(I_min(k,i))).^n_1_M).*((1+H_vic_M*(1-
cos_t_pfx_nfz_D(I_min(k,i)))).^n_2_M));
          end

          if (xz_surf(I_min(k,i),2)>=(h*sin(alfa))) %T
            F_ext_nfx_pfz_1st(k,i)=real((C/rho_s)*Eros_pow_ext(k,i).*((cos_t_nfx_pfz_ext(k,i))^(k_vel+1)));
          else %M
            if (cos_t_nfx_pfz_ext(k,i)<=0) %Apply mask visibility for M
            F_ext_nfx_pfz_1st(k,i)=0;
            else
            F_ext_nfx_pfz_1st(k,i)=real((C_M/rho_s_M).*Eros_pow_ext(k,i).*((cos_t_nfx_pfz_ext(k,i)).^n_1_M).*((1+H_vic_M*(1-
cos_t_nfx_pfz_ext(k,i))).^n_2_M));
            end
          end

          %Note: cos_theta_D = 0 if no 2nd strk, so OK (no NaNs)
          if (xz_surf(I_min(k,i),2)>=(h*sin(alfa))) %T
            F_ext_nfx_pfz(k,i)=F_ext_nfx_pfz_1st(k,i)+real((C/rho_s)*Eros_pow_ext_2nd(k,i).*((cos_t_nfx_pfz_D(I_min(k,i)))^(k_vel+1)));
          else %M

F_ext_nfx_pfz(k,i)=F_ext_nfx_pfz_1st(k,i)+real((C_M/rho_s_M).*Eros_pow_ext_2nd(k,i).*((cos_t_nfx_pfz_D(I_min(k,i))).^n_1_M).*((1+H_vic_M*(1-
cos_t_nfx_pfz_D(I_min(k,i)))).^n_2_M));
          end

          if (xz_surf(I_min(k,i),2)>=(h*sin(alfa))) %T
            F_ext_nfx_nfz_1st(k,i)=real((C/rho_s)*Eros_pow_ext(k,i).*((cos_t_nfx_nfz_ext(k,i))^(k_vel+1)));
          else %M
            if (cos_t_nfx_nfz_ext(k,i)<=0) %Apply mask visibility for M
            F_ext_nfx_nfz_1st(k,i)=0;
            else
            F_ext_nfx_nfz_1st(k,i)=real((C_M/rho_s_M).*Eros_pow_ext(k,i).*((cos_t_nfx_nfz_ext(k,i)).^n_1_M).*((1+H_vic_M*(1-
cos_t_nfx_nfz_ext(k,i))).^n_2_M));
            end
          end

          %Note: cos_theta_D = 0 if no 2nd strk, so OK (no NaNs)
          if (xz_surf(I_min(k,i),2)>=(h*sin(alfa))) %T
            F_ext_nfx_nfz(k,i)=F_ext_nfx_nfz_1st(k,i)+real((C/rho_s)*Eros_pow_ext_2nd(k,i).*((cos_t_nfx_nfz_D(I_min(k,i)))^(k_vel+1)));
          else %M

F_ext_nfx_nfz(k,i)=F_ext_nfx_nfz_1st(k,i)+real((C_M/rho_s_M).*Eros_pow_ext_2nd(k,i).*((cos_t_nfx_nfz_D(I_min(k,i))).^n_1_M).*((1+H_vic_M*(1-
cos_t_nfx_nfz_D(I_min(k,i)))).^n_2_M));
          end

          if (xz_surf(I_min(k,i),2)>=(h*sin(alfa))) %T
            F_ext_star_1st(k,i)=real((C/rho_s)*Eros_pow_ext(k,i).*((cos_t_star_ext(k,i))^(k_vel+1)));
          else %M
            if (cos_t_star_ext(k,i)<=0) %Apply mask visibility for M
            F_ext_star_1st(k,i)=0;
            else
            F_ext_star_1st(k,i)=real((C_M/rho_s_M).*Eros_pow_ext(k,i).*((cos_t_star_ext(k,i)).^n_1_M).*((1+H_vic_M*(1-cos_t_star_ext(k,i))).^n_2_M));
            end
          end

          %Note: cos_theta_D = 0 if no 2nd strk, so OK (no NaNs)
```

```
                if (xz_surf(I_min(k,i),2)>=(h*sin(alfa))) %T
                  F_ext_star(k,i)=F_ext_star_1st(k,i)+real((C/rho_s)*Eros_pow_ext_2nd(k,i).*((cos_t_star_D(I_min(k,i)))^(k_vel+1)));
                else %M
                  F_ext_star(k,i)=F_ext_star_1st(k,i)+real((C_M/rho_s_M).*Eros_pow_ext_2nd(k,i).*((cos_t_star_D(I_min(k,i))).^n_1_M).*((1+H_vic_M*(1-
cos_t_star_D(I_min(k,i)))).^n_2_M));
                end

                if (xz_surf(I_min(k,i),2)>=(h*sin(alfa))) %T
                  F_ext_cen_1st(k,i)=real((C/rho_s)*Eros_pow_ext(k,i).*((cos_t_cen_ext(k,i))^(k_vel+1)));
                else %M
                  if (cos_t_cen_ext(k,i)<=0) %Apply mask visibility for M
                  F_ext_cen_1st(k,i)=0;
                  else
                  F_ext_cen_1st(k,i)=real((C_M/rho_s_M).*Eros_pow_ext(k,i).*((cos_t_cen_ext(k,i)).^n_1_M).*((1+H_vic_M*(1-cos_t_cen_ext(k,i))).^n_2_M));
                  end
                end

                %Note: cos_theta_D = 0 if no 2nd strk, so OK (no NaNs)
                if (xz_surf(I_min(k,i),2)>=(h*sin(alfa))) %T
                  F_ext_cen(k,i)=F_ext_cen_1st(k,i)+real((C/rho_s)*Eros_pow_ext_2nd(k,i).*((cos_t_cen_D(I_min(k,i)))^(k_vel+1)));
                else %M
                  F_ext_cen(k,i)=F_ext_cen_1st(k,i)+real((C_M/rho_s_M).*Eros_pow_ext_2nd(k,i).*((cos_t_cen_D(I_min(k,i))).^n_1_M).*((1+H_vic_M*(1-
cos_t_cen_D(I_min(k,i)))).^n_2_M));
                end

        else %we are on surface and grid node = surface node (and phi=0)
            %Repeat above algorithm but using grid nodes

%__________________________________________________________________________
            %Calculation of dphi/dx,dphi/dz; Define BC's
            if i==i_max
                phi_x_pos_ext(k,i)=0;
            else
                phi_x_pos_ext(k,i)=(phi_1(k,i+1)-phi_1(k,i))./dx;
            end

            if i==1
                phi_x_neg_ext(k,i)=0;
            else
                phi_x_neg_ext(k,i)=(phi_1(k,i)-phi_1(k,i-1))./dx;
            end

            if (i==i_max)||(i==1)
                phi_x_cen_ext(k,i)=0;
            else
                phi_x_cen_ext(k,i)=(phi_1(k,i+1)-phi_1(k,i-1))./(2*dx);
            end

            if k==1
                phi_z_pos_ext(k,i)=0;
            else
                phi_z_pos_ext(k,i)=(phi_1(k-1,i)-phi_1(k,i))./dz;
            end

            if k==k_max
                phi_z_neg_ext(k,i)=0;
            else
                phi_z_neg_ext(k,i)=(phi_1(k,i)-phi_1(k+1,i))./dz;
            end

            if (k==k_max)||(k==1)
                phi_z_cen_ext(k,i)=0;
            else
                phi_z_cen_ext(k,i)=(phi_1(k-1,i)-phi_1(k+1,i))./(2*dz);
            end

            %Calculate phi_stars
            phi_x_star_ext(k,i)=(phi_x_pos_ext(k,i)+phi_x_neg_ext(k,i))/2;
            phi_z_star_ext(k,i)=(phi_z_pos_ext(k,i)+phi_z_neg_ext(k,i))/2;
%__________________________________________________________________________
            %Masking function for T, M (analogous to above)
            x_prime_ext(k,i)=x_cord_local(i).*sin(alfa)-z_cord(k).*cos(alfa); %Rotated local x
            z_prime_ext(k,i)=x_cord_local(i).*cos(alfa)+z_cord(k).*sin(alfa); %Rotated local z

                  if (z_cord(k)>=(h*sin(alfa))) %T

                  if (max_x_prime_surf_LM<0) %Case (a) and (c)
                     if (x_prime_ext(k,i)<0)
                        L_mask_ext(k,i)=z_prime_ext(k,i).*tan_fi_min;
                     else %i.e., when x_prime>=0
                L_mask_ext(k,i)=z_prime_ext(k,i).*tan_fi_max;
              end
            else %i.e., when x_m>=W_m/2 Case (b)
              x_lim_ext(k,i)=z_prime_ext(k,i).*tan_fi_min;
              if (x_prime_ext(k,i)<x_lim_ext(k,i))
                L_mask_ext(k,i)=0;
              else %i.e., when x_prime>=x_lim
                L_mask_ext(k,i)=z_prime_ext(k,i).*tan_fi_max-x_lim_ext(k,i);
              end
            end

            %Define proportion of mass of particle that pass through mask opening having a
            %specific particle size (of radius r) distribution
            if (abs(x_prime_ext(k,i))>=L_mask_ext(k,i))
              M_r_x_prime_ext(k,i)=0;
            else
              Int_P_r_x_prime_ext(k,i)=real(P_1-P_1*erf(P_2*log(L_mask_ext(k,i)-abs(x_prime_ext(k,i)))-P_3));
              Int_P_r_L_mask_ext(k,i)=real(P_1-P_1*erf(P_2*log(L_mask_ext(k,i))-P_3));
              M_r_x_prime_ext(k,i)=Int_P_r_x_prime_ext(k,i)./Int_P_r_L_mask_ext(k,i);
            end

            else  %M
              M_r_x_prime_ext(k,i)=1;
            end

%____________________________________________________________________________
            %Define velocity v(x,z) at each grid node
              v_ext(k,i)=v_o*(1-H_slp*((x_cord_local(i)*sin(alfa)-z_cord(k)*cos(alfa)).^2+(y_mean).^2).^0.5./(x_cord_local(i)*cos(alfa)+z_cord(k)*sin(alfa)));
            if (v_ext(k,i)<0)
              v_ext(k,i)=0;
            end

            %Define particle mass flux(x,z) at each grid node
              flux_ext(k,i)=(MFR/pi)*(beta./(x_cord_local(i)*cos(alfa)+z_cord(k)*sin(alfa))).^2....
              *exp(-(beta^2.*((x_cord_local(i)*sin(alfa)-z_cord(k)*cos(alfa)).^2+(y_mean).^2)./(x_cord_local(i)*cos(alfa)+z_cord(k)*sin(alfa)).^2));

            %Define Erosive Power Eros_pow(k,i) at each grid
            %node (1st strike)
            if (z_cord(k)>=(h*sin(alfa))) %T
              Eros_pow_ext(k,i)=M_r_x_prime_ext(k,i).*v_ext(k,i).^k_vel.*flux_ext(k,i);
            else %M
              Eros_pow_ext(k,i)=M_r_x_prime_ext(k,i).*v_ext(k,i).^k_vel_M.*flux_ext(k,i);
            end

            %Define Erosive Power for 2nd strike
            %NOTE: No Mask here
            if (z_cord(k)>=(h*sin(alfa))) %T
              if (NaN_Chk_theta_D(I_min(k,i))==0)
                Eros_pow_ext_2nd(k,i)=(f_v_AR_fin(I_min(k,i)).*v_AR(I_min(k,i))).^k_vel.*flux_AR(I_min(k,i));
              else
                Eros_pow_ext_2nd(k,i)=0;
              end
            else %M
              if (NaN_Chk_theta_D(I_min(k,i))==0)
                Eros_pow_ext_2nd(k,i)=(f_v_AR_fin(I_min(k,i)).*v_AR(I_min(k,i))).^k_vel_M.*flux_AR(I_min(k,i));
              else
                Eros_pow_ext_2nd(k,i)=0;
              end
            end


%____________________________________________________________________________
        %Calculate F_extensions

          cos_t_pfx_pfz_ext(k,i)=(x_cord_local(i).*(phi_x_pos_ext(k,i))+z_cord(k).*(phi_z_pos_ext(k,i)))./sqrt(x_cord_local(i).^2+z_cord(k).^2);
          cos_t_pfx_nfz_ext(k,i)=(x_cord_local(i).*(phi_x_pos_ext(k,i))+z_cord(k).*(phi_z_neg_ext(k,i)))./sqrt(x_cord_local(i).^2+z_cord(k).^2);
          cos_t_nfx_pfz_ext(k,i)=(x_cord_local(i).*(phi_x_neg_ext(k,i))+z_cord(k).*(phi_z_pos_ext(k,i)))./sqrt(x_cord_local(i).^2+z_cord(k).^2);
          cos_t_nfx_nfz_ext(k,i)=(x_cord_local(i).*(phi_x_neg_ext(k,i))+z_cord(k).*(phi_z_neg_ext(k,i)))./sqrt(x_cord_local(i).^2+z_cord(k).^2);
          cos_t_star_ext(k,i)=(x_cord_local(i).*(phi_x_star_ext(k,i))+z_cord(k).*(phi_z_star_ext(k,i)))./sqrt(x_cord_local(i).^2+z_cord(k).^2);
          cos_t_cen_ext(k,i)=(x_cord_local(i).*(phi_x_cen_ext(k,i))+z_cord(k).*(phi_z_cen_ext(k,i)))./sqrt(x_cord_local(i).^2+z_cord(k).^2);
          if (cos_t_pfx_pfz_ext(k,i)>1)
            cos_t_pfx_pfz_ext(k,i)=1;
          end
          if (cos_t_pfx_nfz_ext(k,i)>1)
            cos_t_pfx_nfz_ext(k,i)=1;
          end
          if (cos_t_nfx_pfz_ext(k,i)>1)
            cos_t_nfx_pfz_ext(k,i)=1;
          end
          if (cos_t_nfx_nfz_ext(k,i)>1)
```

```
               cos_t_nfx_nfz_ext(k,i)=1;
            end
            if (cos_t_star_ext(k,i)>1)
               cos_t_star_ext(k,i)=1;
            end
            if (cos_t_cen_ext(k,i)>1)
               cos_t_cen_ext(k,i)=1;
            end

            if (z_cord(k)>=(h*sin(alfa))) %T
              F_ext_pfx_pfz_1st(k,i)=real((C/rho_s)*Eros_pow_ext(k,i).*((cos_t_pfx_pfz_ext(k,i))^(k_vel+1)));
            else %M
              if (cos_t_pfx_pfz_ext(k,i)<=0) %Apply mask visibility for M
              F_ext_pfx_pfz_1st(k,i)=0;
              else
              F_ext_pfx_pfz_1st(k,i)=real((C_M/rho_s_M).*Eros_pow_ext(k,i).*((cos_t_pfx_pfz_ext(k,i)).^n_1_M).*((1+H_vic_M*(1-
cos_t_pfx_pfz_ext(k,i))).^n_2_M));
              end
            end

            %Note: cos_theta_D = 0 if no 2nd strk, so OK (no NaNs)
            if (z_cord(k)>=(h*sin(alfa))) %T
              F_ext_pfx_pfz(k,i)=F_ext_pfx_pfz_1st(k,i)+real((C/rho_s)*Eros_pow_ext_2nd(k,i).*((cos_t_pfx_pfz_D(I_min(k,i)))^(k_vel+1)));
            else %M

F_ext_pfx_pfz(k,i)=F_ext_pfx_pfz_1st(k,i)+real((C_M/rho_s_M).*Eros_pow_ext_2nd(k,i).*((cos_t_pfx_pfz_D(I_min(k,i))).^n_1_M).*((1+H_vic_M*(1-
cos_t_pfx_pfz_D(I_min(k,i)))).^n_2_M));
            end

            if (z_cord(k)>=(h*sin(alfa))) %T
              F_ext_pfx_nfz_1st(k,i)=real((C/rho_s)*Eros_pow_ext(k,i).*((cos_t_pfx_nfz_ext(k,i))^(k_vel+1)));
            else %M
              if (cos_t_pfx_nfz_ext(k,i)<=0) %Apply mask visibility for M
              F_ext_pfx_nfz_1st(k,i)=0;
              else
              F_ext_pfx_nfz_1st(k,i)=real((C_M/rho_s_M).*Eros_pow_ext(k,i).*((cos_t_pfx_nfz_ext(k,i)).^n_1_M).*((1+H_vic_M*(1-
cos_t_pfx_nfz_ext(k,i))).^n_2_M));
              end
            end

            %Note: cos_theta_D = 0 if no 2nd strk, so OK (no NaNs)
            if (z_cord(k)>=(h*sin(alfa))) %T
              F_ext_pfx_nfz(k,i)=F_ext_pfx_nfz_1st(k,i)+real((C/rho_s)*Eros_pow_ext_2nd(k,i).*((cos_t_pfx_nfz_D(I_min(k,i)))^(k_vel+1)));
            else %M

F_ext_pfx_nfz(k,i)=F_ext_pfx_nfz_1st(k,i)+real((C_M/rho_s_M).*Eros_pow_ext_2nd(k,i).*((cos_t_pfx_nfz_D(I_min(k,i))).^n_1_M).*((1+H_vic_M*(1-
cos_t_pfx_nfz_D(I_min(k,i)))).^n_2_M));
            end

            if (z_cord(k)>=(h*sin(alfa))) %T
              F_ext_nfx_pfz_1st(k,i)=real((C/rho_s)*Eros_pow_ext(k,i).*((cos_t_nfx_pfz_ext(k,i))^(k_vel+1)));
            else %M
              if (cos_t_nfx_pfz_ext(k,i)<=0) %Apply mask visibility for M
              F_ext_nfx_pfz_1st(k,i)=0;
              else
              F_ext_nfx_pfz_1st(k,i)=real((C_M/rho_s_M).*Eros_pow_ext(k,i).*((cos_t_nfx_pfz_ext(k,i)).^n_1_M).*((1+H_vic_M*(1-
cos_t_nfx_pfz_ext(k,i))).^n_2_M));
              end
            end

            %Note: cos_theta_D = 0 if no 2nd strk, so OK (no NaNs)
            if (z_cord(k)>=(h*sin(alfa))) %T
              F_ext_nfx_pfz(k,i)=F_ext_nfx_pfz_1st(k,i)+real((C/rho_s)*Eros_pow_ext_2nd(k,i).*((cos_t_nfx_pfz_D(I_min(k,i)))^(k_vel+1)));
            else %M

F_ext_nfx_pfz(k,i)=F_ext_nfx_pfz_1st(k,i)+real((C_M/rho_s_M).*Eros_pow_ext_2nd(k,i).*((cos_t_nfx_pfz_D(I_min(k,i))).^n_1_M).*((1+H_vic_M*(1-
cos_t_nfx_pfz_D(I_min(k,i)))).^n_2_M));
            end

            if (z_cord(k)>=(h*sin(alfa))) %T
              F_ext_nfx_nfz_1st(k,i)=real((C/rho_s)*Eros_pow_ext(k,i).*((cos_t_nfx_nfz_ext(k,i))^(k_vel+1)));
            else %M
              if (cos_t_nfx_nfz_ext(k,i)<=0) %Apply mask visibility for M
              F_ext_nfx_nfz_1st(k,i)=0;
              else
              F_ext_nfx_nfz_1st(k,i)=real((C_M/rho_s_M).*Eros_pow_ext(k,i).*((cos_t_nfx_nfz_ext(k,i)).^n_1_M).*((1+H_vic_M*(1-
cos_t_nfx_nfz_ext(k,i))).^n_2_M));
              end
            end
```

```
                %Note: cos_theta_D = 0 if no 2nd strk, so OK (no NaNs)
                if (z_cord(k)>=(h*sin(alfa))) %T
                  F_ext_nfx_nfz(k,i)=F_ext_nfx_nfz_1st(k,i)+real((C/rho_s)*Eros_pow_ext_2nd(k,i).*((cos_t_nfx_nfz_D(I_min(k,i)))^(k_vel+1)));
                else %M

F_ext_nfx_nfz(k,i)=F_ext_nfx_nfz_1st(k,i)+real((C_M/rho_s_M).*Eros_pow_ext_2nd(k,i).*((cos_t_nfx_nfz_D(I_min(k,i))).^n_1_M).*((1+H_vic_M*(1-
cos_t_nfx_nfz_D(I_min(k,i)))).^n_2_M));
                end

                if (z_cord(k)>=(h*sin(alfa))) %T
                  F_ext_star_1st(k,i)=real((C/rho_s)*Eros_pow_ext(k,i).*((cos_t_star_ext(k,i))^(k_vel+1)));
                else %M
                  if (cos_t_star_ext(k,i)<=0) %Apply mask visibility for M
                  F_ext_star_1st(k,i)=0;
                  else
                  F_ext_star_1st(k,i)=real((C_M/rho_s_M).*Eros_pow_ext(k,i).*((cos_t_star_ext(k,i)).^n_1_M).*((1+H_vic_M*(1-cos_t_star_ext(k,i))).^n_2_M));
                  end
                end

                %Note: cos_theta_D = 0 if no 2nd strk, so OK (no NaNs)
                if (z_cord(k)>=(h*sin(alfa))) %T
                  F_ext_star(k,i)=F_ext_star_1st(k,i)+real((C/rho_s)*Eros_pow_ext_2nd(k,i).*((cos_t_star_D(I_min(k,i)))^(k_vel+1)));
                else %M
                  F_ext_star(k,i)=F_ext_star_1st(k,i)+real((C_M/rho_s_M).*Eros_pow_ext_2nd(k,i).*((cos_t_star_D(I_min(k,i))).^n_1_M).*((1+H_vic_M*(1-
cos_t_star_D(I_min(k,i)))).^n_2_M));
                end

                if (z_cord(k)>=(h*sin(alfa))) %T
                  F_ext_cen_1st(k,i)=real((C/rho_s)*Eros_pow_ext(k,i).*((cos_t_cen_ext(k,i))^(k_vel+1)));
                else %M
                  if (cos_t_cen_ext(k,i)<=0) %Apply mask visibility for M
                  F_ext_cen_1st(k,i)=0;
                  else
                  F_ext_cen_1st(k,i)=real((C_M/rho_s_M).*Eros_pow_ext(k,i).*((cos_t_cen_ext(k,i)).^n_1_M).*((1+H_vic_M*(1-cos_t_cen_ext(k,i))).^n_2_M));
                  end
                end

                %Note: cos_theta_D = 0 if no 2nd strk, so OK (no NaNs)
                if (z_cord(k)>=(h*sin(alfa))) %T
                  F_ext_cen(k,i)=F_ext_cen_1st(k,i)+real((C/rho_s)*Eros_pow_ext_2nd(k,i).*((cos_t_cen_D(I_min(k,i)))^(k_vel+1)));
                else %M
                  F_ext_cen(k,i)=F_ext_cen_1st(k,i)+real((C_M/rho_s_M).*Eros_pow_ext_2nd(k,i).*((cos_t_cen_D(I_min(k,i))).^n_1_M).*((1+H_vic_M*(1-
cos_t_cen_D(I_min(k,i)))).^n_2_M));
                end

        end


       end  %#####*****#####$$$$$#####*****#####
    end
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
                    %END OF SDF AND F_EXT ALGORITHM
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%Time Counter
time=time+dt;

%Iteration counter
counter=counter+1;

%Initialize variables for NB collision detection (see below)
flag_RE=0;
Num_iter_RE_TOT=0;

%#######################################################################
%                      END OF INITIAL ITERATION
%#######################################################################

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
                        %Main Loop
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

while (time<=t_in)

%______________________________________________________________________
%Define FD's and BC's
phi_x_pos=zeros(k_max,i_max);
phi_x_neg=zeros(k_max,i_max);
phi_x_cen=zeros(k_max,i_max);
```

```
phi_x_x_cen=zeros(k_max,i_max);
phi_z_pos=zeros(k_max,i_max);
phi_z_neg=zeros(k_max,i_max);
phi_z_cen=zeros(k_max,i_max);
phi_z_z_cen=zeros(k_max,i_max);

%FD's where there is a NB boundary 1st
for k=1:1:k_max
        for i=1:1:i_max

            if (flag_NB(k,i)==2) %#####*****#####$$$$$#####*****#####

                if i==i_max
                    phi_x_pos(k,i)=0;
                elseif ((flag_NB(k,i+1)==3)||(flag_NB(k,i+1)==4))
                    phi_x_pos(k,i)=0;
                else
                    phi_x_pos(k,i)=(phi(k,i+1)-phi(k,i))./dx;
                end

                if i==1
                    phi_x_neg(k,i)=0;
                elseif ((flag_NB(k,i-1)==3)||(flag_NB(k,i-1)==4))
                    phi_x_neg(k,i)=0;
                else
                    phi_x_neg(k,i)=(phi(k,i)-phi(k,i-1))./dx;
                end

                if (i==i_max)||(i==1)
                    phi_x_cen(k,i)=0;
                    phi_x_x_cen(k,i)=0;
                elseif ((flag_NB(k,i+1)==3)||(flag_NB(k,i+1)==4))||((flag_NB(k,i-1)==3)||(flag_NB(k,i-1)==4))
                    phi_x_cen(k,i)=0;
                    phi_x_x_cen(k,i)=0;
                else
                    phi_x_cen(k,i)=(phi(k,i+1)-phi(k,i-1))./(2*dx);
                    phi_x_x_cen(k,i)=(phi(k,i+1)-2*phi(k,i)+phi(k,i-1))./(dx^2);
                end

                if k==1
                    phi_z_pos(k,i)=0;
                elseif ((flag_NB(k-1,i)==3)||(flag_NB(k-1,i)==4))
                    phi_z_pos(k,i)=0;
                else
                    phi_z_pos(k,i)=(phi(k-1,i)-phi(k,i))./dz;
                end

                if k==k_max
                    phi_z_neg(k,i)=0;
                elseif ((flag_NB(k+1,i)==3)||(flag_NB(k+1,i)==4))
                    phi_z_neg(k,i)=0;
                else
                    phi_z_neg(k,i)=(phi(k,i)-phi(k+1,i))./dz;
                end

                if (k==k_max)||(k==1)
                    phi_z_cen(k,i)=0;
                    phi_z_z_cen(k,i)=0;
                elseif ((flag_NB(k-1,i)==3)||(flag_NB(k-1,i)==4))||((flag_NB(k+1,i)==3)||(flag_NB(k+1,i)==4))
                    phi_z_cen(k,i)=0;
                    phi_z_z_cen(k,i)=0;
                else
                    phi_z_cen(k,i)=(phi(k-1,i)-phi(k+1,i))./(2*dz);
                    phi_z_z_cen(k,i)=(phi(k-1,i)-2*phi(k,i)+phi(k+1,i))./(dz^2);
                end

            end %#####*****#####$$$$$#####*****#####
        end
end

%FD's inside the NB
for k=1:1:k_max
        for i=1:1:i_max

            if (flag_NB(k,i)==1) %#####*****#####$$$$$#####*****#####

                if i==i_max
                    phi_x_pos(k,i)=0;
                else
                    phi_x_pos(k,i)=(phi(k,i+1)-phi(k,i))./dx;
                end

                if i==1
                    phi_x_neg(k,i)=0;
                else
                    phi_x_neg(k,i)=(phi(k,i)-phi(k,i-1))./dx;
                end

                if (i==i_max)||(i==1)
                    phi_x_cen(k,i)=0;
                    phi_x_x_cen(k,i)=0;
                else
                    phi_x_cen(k,i)=(phi(k,i+1)-phi(k,i-1))./(2*dx);
                    phi_x_x_cen(k,i)=(phi(k,i+1)-2*phi(k,i)+phi(k,i-1))./(dx^2);
                end

                if k==1
                    phi_z_pos(k,i)=0;
                else
                    phi_z_pos(k,i)=(phi(k-1,i)-phi(k,i))./dz;
                end

                if k==k_max
                    phi_z_neg(k,i)=0;
                else
                    phi_z_neg(k,i)=(phi(k,i)-phi(k+1,i))./dz;
                end

                if (k==k_max)||(k==1)
                    phi_z_cen(k,i)=0;
                    phi_z_z_cen(k,i)=0;
                else
                    phi_z_cen(k,i)=(phi(k-1,i)-phi(k+1,i))./(2*dz);
                    phi_z_z_cen(k,i)=(phi(k-1,i)-2*phi(k,i)+phi(k+1,i))./(dz^2);
                end

            end %#####*****#####$$$$$#####****#####
        end
end

%______________________________________________________________________
%Define Curvature K
K=zeros(k_max,i_max);
 if (epsilon==0)
     for k=1:1:k_max
        for i=1:1:i_max
            if ((flag_NB(k,i)==1)||(flag_NB(k,i)==2)) %#####*****#####$$$$$#####****##### IN NB, etc.
                    K(k,i)=0;
            end %#####*****#####$$$$$#####****#####
        end
     end
 else
     for k=1:1:k_max
         for i=1:1:i_max
                if ((flag_NB(k,i)==1)||(flag_NB(k,i)==2)) %#####*****#####$$$$$#####****#####
                        K(k,i)=phi_x_x_cen(k,i)+phi_z_z_cen(k,i);
                end  %#####*****#####$$$$$#####****#####
              end
     end
 end

%______________________________________________________________________
%Define Partial Hamiltonians for LLLF Scheme (analogous to initial iteration)
cos_t_pfx_pfz=zeros(k_max,i_max);
cos_t_pfx_nfz=zeros(k_max,i_max);
cos_t_nfx_pfz=zeros(k_max,i_max);
cos_t_nfx_nfz=zeros(k_max,i_max);
H1_LLLF_pfx_pfz=zeros(k_max,i_max); %Partial H wrt phi_x (all +/- FD combinations)
H1_LLLF_pfx_nfz=zeros(k_max,i_max);
H1_LLLF_nfx_pfz=zeros(k_max,i_max);
H1_LLLF_nfx_nfz=zeros(k_max,i_max);
H3_LLLF_pfx_pfz=zeros(k_max,i_max); %Partial H wrt phi_z (all +/- FD combinations)
H3_LLLF_pfx_nfz=zeros(k_max,i_max);
H3_LLLF_nfx_pfz=zeros(k_max,i_max);
H3_LLLF_nfx_nfz=zeros(k_max,i_max);
for k=1:1:k_max
        for i=1:1:i_max
        if ((flag_NB(k,i)==1)||(flag_NB(k,i)==2)) %#####*****#####$$$$$#####****#####

            if (flag_NB(k,i)==1)
        cos_t_pfx_pfz(k,i)=(x_cord_local(i).*(phi_x_pos(k,i))+z_cord(k).*(phi_z_pos(k,i)))./sqrt(x_cord_local(i).^2+z_cord(k).^2);
        cos_t_pfx_nfz(k,i)=(x_cord_local(i).*(phi_x_pos(k,i))+z_cord(k).*(phi_z_neg(k,i)))./sqrt(x_cord_local(i).^2+z_cord(k).^2);
        cos_t_nfx_pfz(k,i)=(x_cord_local(i).*(phi_x_neg(k,i))+z_cord(k).*(phi_z_pos(k,i)))./sqrt(x_cord_local(i).^2+z_cord(k).^2);
        cos_t_nfx_nfz(k,i)=(x_cord_local(i).*(phi_x_neg(k,i))+z_cord(k).*(phi_z_neg(k,i)))./sqrt(x_cord_local(i).^2+z_cord(k).^2);
        else %Need this condition since NB boundary pts can incorrectly evaluate cos() since either phi_x or phi_z is forced to 0 there

         if (phi_x_pos(k,i)==0)&&(phi_z_pos(k,i)==0)
         cos_t_pfx_pfz(k,i)=0;
         else

cos_t_pfx_pfz(k,i)=(x_cord_local(i).*(phi_x_pos(k,i))+z_cord(k).*(phi_z_pos(k,i)))./(sqrt(x_cord_local(i).^2+z_cord(k).^2).*sqrt((phi_x_pos(k,i)).^2+(phi_z_pos(k,i)
).^2));
         end

         if (phi_x_pos(k,i)==0)&&(phi_z_neg(k,i)==0)
         cos_t_pfx_nfz(k,i)=0;
         else

cos_t_pfx_nfz(k,i)=(x_cord_local(i).*(phi_x_pos(k,i))+z_cord(k).*(phi_z_neg(k,i)))./(sqrt(x_cord_local(i).^2+z_cord(k).^2).*sqrt((phi_x_pos(k,i)).^2+(phi_z_neg(k,i
)).^2));
         end

         if (phi_x_neg(k,i)==0)&&(phi_z_pos(k,i)==0)
         cos_t_nfx_pfz(k,i)=0;
         else

cos_t_nfx_pfz(k,i)=(x_cord_local(i).*(phi_x_neg(k,i))+z_cord(k).*(phi_z_pos(k,i)))./(sqrt(x_cord_local(i).^2+z_cord(k).^2).*sqrt((phi_x_neg(k,i)).^2+(phi_z_pos(k,i
)).^2));
         end

         if (phi_x_neg(k,i)==0)&&(phi_z_neg(k,i)==0)
         cos_t_nfx_nfz(k,i)=0;
         else

cos_t_nfx_nfz(k,i)=(x_cord_local(i).*(phi_x_neg(k,i))+z_cord(k).*(phi_z_neg(k,i)))./(sqrt(x_cord_local(i).^2+z_cord(k).^2).*sqrt((phi_x_neg(k,i)).^2+(phi_z_neg(k,i
)).^2));
         end

        end

        if (cos_t_pfx_pfz(k,i)>1) %limit cos(theta) to be b/w -1 and 1
              cos_t_pfx_pfz(k,i)=1;
        end
        if (cos_t_pfx_nfz(k,i)>1)
               cos_t_pfx_nfz(k,i)=1;
        end
        if (cos_t_nfx_pfz(k,i)>1)
               cos_t_nfx_pfz(k,i)=1;
        end
        if (cos_t_nfx_nfz(k,i)>1)
               cos_t_nfx_nfz(k,i)=1;
        end
        %_________
        if cos_t_pfx_pfz(k,i)==0 %Done to ensure 0/0 doesn't results and hence an error - F=H=0 when this occurs
            H3_LLLF_pfx_pfz(k,i)=0;
            H1_LLLF_pfx_pfz(k,i)=0;
        else
            if flag_T_M(k,i)==1 %T
               H3_LLLF_pfx_pfz(k,i)=real(F_ext_pfx_pfz(k,i)...
                   .*(z_cord(k)./sqrt(x_cord_local(i).^2+z_cord(k).^2)).*((k_vel+1)./cos_t_pfx_pfz(k,i)));
            else %M
               H3_LLLF_pfx_pfz(k,i)=real(F_ext_pfx_pfz(k,i)...
                   .*(z_cord(k)./sqrt(x_cord_local(i).^2+z_cord(k).^2)).*(n_1_M./cos_t_pfx_pfz(k,i)-n_2_M*H_vic_M./(1+H_vic_M*(1-cos_t_pfx_pfz(k,i)))));
            end

            H1_LLLF_pfx_pfz(k,i)=(x_cord_local(i)./z_cord(k)).*H3_LLLF_pfx_pfz(k,i);
        end
        %_________
        if cos_t_pfx_nfz(k,i)==0
            H3_LLLF_pfx_nfz(k,i)=0;
            H1_LLLF_pfx_nfz(k,i)=0;
        else
            if flag_T_M(k,i)==1 %T
               H3_LLLF_pfx_nfz(k,i)=real(F_ext_pfx_nfz(k,i)...
                   .*(z_cord(k)./sqrt(x_cord_local(i).^2+z_cord(k).^2)).*((k_vel+1)./cos_t_pfx_nfz(k,i)));
            else %M
               H3_LLLF_pfx_nfz(k,i)=real(F_ext_pfx_nfz(k,i)...
                   .*(z_cord(k)./sqrt(x_cord_local(i).^2+z_cord(k).^2)).*(n_1_M./cos_t_pfx_nfz(k,i)-n_2_M*H_vic_M./(1+H_vic_M*(1-cos_t_pfx_nfz(k,i)))));
            end
        H1_LLLF_pfx_nfz(k,i)=(x_cord_local(i)./z_cord(k)).*H3_LLLF_pfx_nfz(k,i);
        end
        %_________
        if cos_t_nfx_pfz(k,i)==0
        H3_LLLF_nfx_pfz(k,i)=0;
        H1_LLLF_nfx_pfz(k,i)=0;
        else
        if flag_T_M(k,i)==1 %T
            H3_LLLF_nfx_pfz(k,i)=real(F_ext_nfx_pfz(k,i)...
                .*(z_cord(k)./sqrt(x_cord_local(i).^2+z_cord(k).^2)).*((k_vel+1)./cos_t_nfx_pfz(k,i)));
        else %M
            H3_LLLF_nfx_pfz(k,i)=real(F_ext_nfx_pfz(k,i)...
                .*(z_cord(k)./sqrt(x_cord_local(i).^2+z_cord(k).^2)).*(n_1_M./cos_t_nfx_pfz(k,i)-n_2_M*H_vic_M./(1+H_vic_M*(1-cos_t_nfx_pfz(k,i)))));
        end

        H1_LLLF_nfx_pfz(k,i)=(x_cord_local(i)./z_cord(k)).*H3_LLLF_nfx_pfz(k,i);
        end
        %_________
        if cos_t_nfx_nfz(k,i)==0
        H3_LLLF_nfx_nfz(k,i)=0;
        H1_LLLF_nfx_nfz(k,i)=0;
        else
        if flag_T_M(k,i)==1 %T
            H3_LLLF_nfx_nfz(k,i)=real(F_ext_nfx_nfz(k,i)...
                .*(z_cord(k)./sqrt(x_cord_local(i).^2+z_cord(k).^2)).*((k_vel+1)./cos_t_nfx_nfz(k,i)));
        else %M
            H3_LLLF_nfx_nfz(k,i)=real(F_ext_nfx_nfz(k,i)...
                .*(z_cord(k)./sqrt(x_cord_local(i).^2+z_cord(k).^2)).*(n_1_M./cos_t_nfx_nfz(k,i)-n_2_M*H_vic_M./(1+H_vic_M*(1-cos_t_nfx_nfz(k,i)))));
        end

        H1_LLLF_nfx_nfz(k,i)=(x_cord_local(i)./z_cord(k)).*H3_LLLF_nfx_nfz(k,i);
        end
     end %#####*****#####$$$$$#####*****#####

        end
end

%________________________________________________________________________
%LLLF Scheme
%Initialization (preallocation) to increase computational speed
alpha_x=zeros(k_max,i_max);
alpha_z=zeros(k_max,i_max);
Ham=zeros(k_max,i_max);
Ham_num=zeros(k_max,i_max);
 for k=1:1:k_max
        for i=1:1:i_max
        if ((flag_NB(k,i)==1)||(flag_NB(k,i)==2)) %#####*****#####$$$$$#####*****#####
        H1_LLLF_array=[abs(H1_LLLF_pfx_pfz(k,i)),abs(H1_LLLF_pfx_nfz(k,i)),abs(H1_LLLF_nfx_pfz(k,i)),abs(H1_LLLF_nfx_nfz(k,i))];
        alpha_x(k,i)=max(H1_LLLF_array);
        H3_LLLF_array=[abs(H3_LLLF_pfx_pfz(k,i)),abs(H3_LLLF_pfx_nfz(k,i)),abs(H3_LLLF_nfx_pfz(k,i)),abs(H3_LLLF_nfx_nfz(k,i))];
        alpha_z(k,i)=max(H3_LLLF_array);

        %Define Numerical Hamiltonian

        Ham(k,i)=real(F_ext_star(k,i)); %No need to differentiate b/w cases T & M since taken care of in Fext Algorithm

        Ham_num(k,i)=Ham(k,i)-(alpha_x(k,i)/2).*(phi_x_pos(k,i)-phi_x_neg(k,i))-(alpha_z(k,i)/2).*(phi_z_pos(k,i)-phi_z_neg(k,i));

        end %#####*****#####$$$$$#####*****#####
        end
 end

%________________________________________________________________________
 %Define Central Difference Hamiltonian
Ham_cen=zeros(k_max,i_max);
 if (epsilon==0)
   for k=1:1:k_max
     for i=1:1:i_max
       if ((flag_NB(k,i)==1)||(flag_NB(k,i)==2)) %#####*****#####$$$$$#####*****#####
         Ham_cen(k,i)=0;
       end %#####*****#####$$$$$#####*****#####
     end
   end
 else
        for k=1:1:k_max
            for i=1:1:i_max
              if ((flag_NB(k,i)==1)||(flag_NB(k,i)==2)) %#####*****#####$$$$$#####*****#####
                    Ham_cen(k,i)=real(F_ext_cen(k,i)); %No need to differentiate b/w cases T & M since taken care of in Fext Algorithm
              end %#####*****#####$$$$$#####*****#####
            end
          end
 end

%_____________________________________________________________________________
%Calculate dt (CFL Condition)

%Find absolute Max of alpha_x and alpha_z used to obtain time step
alpha_x_vector=zeros(1,k_max*i_max);
alpha_z_vector=zeros(1,k_max*i_max);
for W=1:1:(k_max*i_max)
    alpha_x_vector(1,W)=alpha_x(W);
end
for X=1:1:(k_max*i_max)
    alpha_z_vector(1,X)=alpha_z(X);
end
%No need to take absolute value of above vectors since alphas already
%absolute positive (from previous evaluation)
max_alpha_x=max(alpha_x_vector);
max_alpha_z=max(alpha_z_vector);

%CFL condition
Ham_cen1=zeros(k_max,i_max); %Partial H wrt phi_x (all central FD) used to multiply epsilon
Ham_cen3=zeros(k_max,i_max); %Partial H wrt phi_z (all central FD)
cos_t_cen=zeros(k_max,i_max);
 if (epsilon==0) %No need to evaluate the Partial H's
   for k=1:1:k_max
     for i=1:1:i_max
       if ((flag_NB(k,i)==1)||(flag_NB(k,i)==2)) %#####*****#####$$$$$#####*****#####
         Ham_cen1(k,i)=0;
         Ham_cen3(k,i)=0;
       end %#####*****#####$$$$$#####*****#####
     end
   end
 else
         for k=1:1:k_max
           for i=1:1:i_max
               if ((flag_NB(k,i)==1)||(flag_NB(k,i)==2)) %#####*****#####$$$$$#####*****#####

                 if (flag_NB(k,i)==1)
                 cos_t_cen(k,i)=(x_cord_local(i).*(phi_x_cen(k,i))+z_cord(k).*(phi_z_cen(k,i)))./sqrt(x_cord_local(i).^2+z_cord(k).^2);

                 else %Need this condition since NB boundary pts can incorrectly evaluate cos() since either phi_x or phi_z is forced to 0 there

                 if (phi_x_cen(k,i)==0)&&(phi_z_cen(k,i)==0) %Done to ensure 0/0 doesn't results
                 cos_t_cen(k,i)=0;
                 else

cos_t_cen(k,i)=(x_cord_local(i).*(phi_x_cen(k,i))+z_cord(k).*(phi_z_cen(k,i)))./(sqrt(x_cord_local(i).^2+z_cord(k).^2).*sqrt((phi_x_cen(k,i)).^2+(phi_z_cen(k,i)).^2
));
                 end

                 end

                 if (cos_t_cen(k,i)>1) %limit cos(theta) to be b/w -1 and 1
                   cos_t_cen(k,i)=1;
                 end

                 %_________
                 if (cos_t_cen(k,i)==0)||(cos_t_cen(k,i)<=0.01) %Done to ensure 0/0 doesn't results and hence an error - F=H=0 when this occurs
                   Ham_cen3(k,i)=0;
                   Ham_cen1(k,i)=0;
                 else
                   if flag_T_M(k,i)==1 %T
                     Ham_cen3(k,i)=real(F_ext_cen(k,i)...
                       .*(z_cord(k)./sqrt(x_cord_local(i).^2+z_cord(k).^2)).*((k_vel+1)./cos_t_cen(k,i)));
                   else %M
                     Ham_cen3(k,i)=real(F_ext_cen(k,i)...
                       .*(z_cord(k)./sqrt(x_cord_local(i).^2+z_cord(k).^2)).*(n_1_M./cos_t_cen(k,i)-n_2_M*H_vic_M./(1+H_vic_M*(1-cos_t_cen(k,i)))));
                   end

                   Ham_cen1(k,i)=(x_cord_local(i)./z_cord(k)).*Ham_cen3(k,i);
                 end


               end %#####*****#####$$$$$#####*****#####
           end
         end
 end

 max_Ham_cen1=max(max(Ham_cen1)); %Max Ham_cen's
 max_Ham_cen3=max(max(Ham_cen3));

dt_alpha=0.9;%Used to scale down dt if necessary

if ((max_alpha_x==0)&&(max_alpha_z==0)&&(epsilon==0))
   dt=t_in/Num_iter;
else
    dt=dt_alpha/(max_alpha_x/dx+max_alpha_z/dz+2*epsilon*max_Ham_cen1/dx^2+2*epsilon*max_Ham_cen3/dz^2);
end

%Curvature coefficient
%NOTE can scale up/down 'epsilon' to obtain different epsilons for M and T
%BUT 'epsilon' must be maximum
epsilon_gen=zeros(k_max,i_max);
for k=1:1:k_max
        for i=1:1:i_max
            if ((flag_NB(k,i)==1)||(flag_NB(k,i)==2)) %#####*****#####$$$$$#####*****#####
            if flag_T_M(k,i)==1 %T
                epsilon_gen(k,i)=epsilon;
            else %M
                epsilon_gen(k,i)=0;
            end
            end %#####*****#####$$$$$#####*****#####
        end
end

%______________________________________________________________________________
%Solve EOM for phi's to pass onto while loop
phi_1=zeros(k_max,i_max);
for k=1:1:k_max
        for i=1:1:i_max
            if ((flag_NB(k,i)==1)||(flag_NB(k,i)==2)) %#####*****#####$$$$$#####*****#####
                phi_1(k,i)=phi(k,i)+dt.*(-Ham_num(k,i)+epsilon_gen(k,i).*K(k,i).*Ham_cen(k,i));
            end %#####*****#####$$$$$#####*****#####
        end
end

%______________________________________________________________________________
%Surface Interpolation algorithm (analogous to initial iteration)

z_surf_1=zeros(i_max,1);
z_surf_2=zeros(i_max,1);
z_surf_3=zeros(i_max,1);
for i=1:1:i_max
    flag_1=0;
    flag_2=0;
     for k=1:1:k_max
         if ((flag_NB(k,i)==1)||(flag_NB(k,i)==2)) %#####*****#####$$$$$#####*****#####
           if ((flag_1==0)&&(((phi_1(k,i)>0)&&(phi_1(k+1,i)<0))||(phi_1(k,i)==0)))
               z_surf_1(i)=((phi_1(k,i).*(z_cord(k)-z_cord(k+1)))./(phi_1(k+1,i)-phi_1(k,i)))+z_cord(k);
               flag_1=i;
               continue
           end
           if (((flag_2==0)&&(flag_1==i)&&(k~=k_max))&&(((phi_1(k,i)<0)&&(phi_1(k+1,i)>0))||(phi_1(k,i)==0)))
               z_surf_2(i)=((phi_1(k,i).*(z_cord(k)-z_cord(k+1)))./(phi_1(k+1,i)-phi_1(k,i)))+z_cord(k);
               flag_2=i;
               continue
           end
           if ((flag_2==i)&&(((phi_1(k,i)>0)&&(phi_1(k+1,i)<0))||(phi_1(k,i)==0)))
               z_surf_3(i)=((phi_1(k,i).*(z_cord(k)-z_cord(k+1)))./(phi_1(k+1,i)-phi_1(k,i)))+z_cord(k);
           end
         end %#####*****#####$$$$$#####*****#####
     end
end

x_surf_1=zeros(1,k_max);
x_surf_2=zeros(1,k_max);
for k=1:1:k_max
    flag_3=0;
     for i=1:1:i_max
         if ((flag_NB(k,i)==1)||(flag_NB(k,i)==2)) %#####*****#####$$$$$#####*****#####
            if (((flag_3==0)&&(i~=i_max)&&(i~=1))&&(((phi_1(k,i)>0)&&(phi_1(k,i+1)<0))||(phi_1(k,i)==0)))
                x_surf_1(k)=((phi_1(k,i).*(x_cord(i)-x_cord(i+1)))./(phi_1(k,i+1)-phi_1(k,i)))+x_cord(i);
                flag_3=k;
                continue
            end
            if (((flag_3==k)&&(i~=i_max))&&(((phi_1(k,i)<0)&&(phi_1(k,i+1)>0))||(phi_1(k,i)==0)))
                x_surf_2(k)=((phi_1(k,i).*(x_cord(i)-x_cord(i+1)))./(phi_1(k,i+1)-phi_1(k,i)))+x_cord(i);
            end
         end %#####*****#####$$$$$#####*****#####
    end
end

%________________________________________________________________________
% Convert all z_surf and x_surf to one array
xz_surf=[x_cord', z_surf_1;x_cord', z_surf_2;x_cord', z_surf_3;x_surf_1',z_cord;x_surf_2',z_cord];
b_max=3*i_max+2*k_max;
for b=1:1:(3*i_max)
        if xz_surf(b,2)==0
        xz_surf(b,1)=0;
        xz_surf(b,2)=0;
        else
        xz_surf(b,1)=xz_surf(b,1);
        xz_surf(b,2)=xz_surf(b,2);
        end
end

for b=(3*i_max+1):1:b_max
        if xz_surf(b,1)==0
        xz_surf(b,1)=0;
        xz_surf(b,2)=0;
        else
        xz_surf(b,1)=xz_surf(b,1);
        xz_surf(b,2)=xz_surf(b,2);
        end
end

%________________________________________________________________________
%COLLISSION DETECTION ALGORITHM

%Calculate (roughly) how many iterations it will take to hit band before
%Band Re-initialization (RE)
Num_iter_RE=round(((BS_U-Crit_D/dz)*Num_iter)/(z_in/dz));

%Re-initialization counter; goes to 0 as band is rebuilt and starts
%over.  flag_RE and Num_iter_RE_TOT are pre-allocated to 0 before while
%loop and afterwards are defined by values in REBUILD BAND ALGORITHM
counter_RE=counter-flag_RE*Num_iter_RE_TOT;

%Pre-allocate variables for CHECK DISTANCE ALGORITHM
c_max=b_max;
D_o_U=zeros(b_max,1);
D_o_L=zeros(b_max,1);
min_D_o_U=zeros(c_max,1);
min_D_o_L=zeros(c_max,1);
MIN_D_o_U=0;
MIN_D_o_L=0;
D_o_LU=0; %Min. distance b/w surface and Upper and Lower bands

%Pre-allocate variables for REBUILD BAND ALGORITHM
SDF_RE=zeros(b_max,1);

prop_Num_iter_RE=0.0001; %Defines proportion of Num_iter_RE before begin checking distance for every
%iteration.  The larger this value, the less check are performed but run
%the risk of surface passing band; Makes insignificant difference wrt comp. efficiency so
%assumed ~0 so checks performed every iteration so surface does not collide
%with the band

counter_RE_chk=round(prop_Num_iter_RE*Num_iter_RE); %Round the result (to get integer)
if (counter_RE>=counter_RE_chk) %Start to perform checks after some specified time

   %CHECK DISTANCE ALGORITHM
      for c=1:1:c_max %Check Upper Band
         if  ((xz_surf_U(c,1)==0)&&(xz_surf_U(c,2)==0))
            min_D_o_U(c)=NaN; %Need this since values could be 0 from pre-allocation
         else
            for b=1:1:b_max
               if
((xz_surf(b,1)~=0)&&(xz_surf(b,2)~=0))||((alfa==(90*pi/180))&&((b==1)||(b==(i_max+1))||(b==(2*i_max+1)))&&(xz_surf(b,1)==0)&&(xz_surf(b,2)~=0))
                  D_o_U(b)=((xz_surf_U(c,1)-xz_surf(b,1)).^2+(xz_surf_U(c,2)-xz_surf(b,2)).^2).^0.5;
               else
                  D_o_U(b)=NaN;
               end
            end
            min_D_o_U(c)=min(D_o_U); %Will ignore the NaN's; Min. distances from surface to Upper band pts.
         end
      end

      for c=1:1:c_max %Check Lower Band
         if  ((xz_surf_L(c,1)==0)&&(xz_surf_L(c,2)==0))
```

```
            min_D_o_L(c)=NaN;
         else
            for b=1:1:b_max
               if
((xz_surf(b,1)~=0)&&(xz_surf(b,2)~=0))||((alfa==(90*pi/180))&&((b==1)||(b==(i_max+1))||(b==(2*i_max+1)))&&(xz_surf(b,1)==0)&&(xz_surf(b,2)~=0))
                  D_o_L(b)=((xz_surf_L(c,1)-xz_surf(b,1)).^2+(xz_surf_L(c,2)-xz_surf(b,2)).^2).^0.5;
               else
                  D_o_L(b)=NaN;
               end
            end
            min_D_o_L(c)=min(D_o_L); %Will ignore the NaN's; Min. distances from surface to Lower band pts.
         end
      end

      MIN_D_o_U=min(min_D_o_U); %Will ignore the NaN's; Absolute Min. dist. from surf. to Upper band pts.
      MIN_D_o_L=min(min_D_o_L); %Will ignore the NaN's; Absolute Min. dist. from surf. to Lower band pts.
      MIN_D_o_LU_matrix=[MIN_D_o_U,MIN_D_o_L]; %Make matrix to evaluate Min.
      D_o_LU=min(MIN_D_o_LU_matrix); %Min. distance b/w surface and Upper AND Lower bands

   if (D_o_LU<=Crit_D) %If surface 'near' band

      %REBUILD BAND ALGORITHM

      for k=1:1:k_max
         for i=1:1:i_max

            if ((flag_NB(k,i)==1)||(flag_NB(k,i)==2))
                for b=1:1:b_max
                  if (phi_1(k,i)==0)
                     SDF_RE(b)=0;
                  elseif
((xz_surf(b,1)~=0)&&(xz_surf(b,2)~=0))||((alfa==(90*pi/180))&&((b==1)||(b==(i_max+1))||(b==(2*i_max+1)))&&(xz_surf(b,1)==0)&&(xz_surf(b,2)~=0))
                     SDF_RE(b)=((x_cord(i)-xz_surf(b,1)).^2+(z_cord(k)-xz_surf(b,2)).^2).^0.5;
                  else
                     SDF_RE(b)=NaN; %Need this since it accounts for cases where surface wasn't encountered (i.e. x_surf or z_surf are 0 numerically)
                  end
               end
               min_SDF_RE=min(SDF_RE); %Will ignore the NaN's

               if (phi_1(k,i)>0)
                  phi_1(k,i)=min_SDF_RE;
               elseif (phi_1(k,i)<0)
                  phi_1(k,i)=-min_SDF_RE;
               else
                  phi_1(k,i)=phi_1(k,i); %i.e.,phi(k,i)=0; i.e., we are on the surface!!!
               end

            else
               for b=1:1:b_max %No need to check if are on the surface since these are band or outer band pts
                  if
((xz_surf(b,1)~=0)&&(xz_surf(b,2)~=0))||((alfa==(90*pi/180))&&((b==1)||(b==(i_max+1))||(b==(2*i_max+1)))&&(xz_surf(b,1)==0)&&(xz_surf(b,2)~=0))
                     SDF_RE(b)=((x_cord(i)-xz_surf(b,1)).^2+(z_cord(k)-xz_surf(b,2)).^2).^0.5;
                  else
                     SDF_RE(b)=NaN;
                  end
               end
               min_SDF_RE=min(SDF_RE);

               if (((flag_NB(k,i)==4)))
                  phi_1(k,i)=min_SDF_RE; %Positive phi_1's for upper band and in front upper band pts.
               else
                  phi_1(k,i)=-min_SDF_RE; %Accounts for all other cases when flag_NB=3
               end
            end
         end
      end

      %Calculate NEW Upper and Lower Band

      %NEW UPPER BAND

      z_surf_1_U=zeros(i_max,1);
      z_surf_2_U=zeros(i_max,1);
      z_surf_3_U=zeros(i_max,1);
      for i=1:1:i_max
         flag_1_U=0;
         flag_2_U=0;
         for k=1:1:k_max
            if ((flag_1_U==0)&&(((phi_1(k,i)>(BS_U*dz))&&(phi_1(k+1,i)<(BS_U*dz)))||(phi_1(k,i)==(BS_U*dz))))
               z_surf_1_U(i)=(((BS_U*dz-phi_1(k,i)).*(z_cord(k+1)-z_cord(k)))./(phi_1(k+1,i)-phi_1(k,i)))+z_cord(k);
            flag_1_U=i;
            continue
        end
        if (((flag_2_U==0)&&(flag_1_U==i)&&(k~=k_max))&&(((phi_1(k,i)<(BS_U*dz))&&(phi_1(k+1,i)>(BS_U*dz)))||(phi_1(k,i)==(BS_U*dz))))
            z_surf_2_U(i)=(((BS_U*dz-phi_1(k,i)).*(z_cord(k+1)-z_cord(k)))./(phi_1(k+1,i)-phi_1(k,i)))+z_cord(k);
            flag_2_U=i;
            continue
        end
        if ((flag_2_U==i)&&(((phi_1(k,i)>(BS_U*dz))&&(phi_1(k+1,i)<(BS_U*dz)))||(phi_1(k,i)==(BS_U*dz))))
            z_surf_3_U(i)=(((BS_U*dz-phi_1(k,i)).*(z_cord(k+1)-z_cord(k)))./(phi_1(k+1,i)-phi_1(k,i)))+z_cord(k);
        end
    end
end

x_surf_1_U=zeros(1,k_max);
x_surf_2_U=zeros(1,k_max);
for k=1:1:k_max
    flag_3_U=0;
    for i=1:1:i_max
            if (((flag_3_U==0)&&(i~=i_max)&&(i~=1))&&(((phi_1(k,i)>(BS_U*dz))&&(phi_1(k,i+1)<(BS_U*dz)))||(phi_1(k,i)==(BS_U*dz))))
                    x_surf_1_U(k)=(((BS_U*dz-phi_1(k,i)).*(x_cord(i+1)-x_cord(i)))./(phi_1(k,i+1)-phi_1(k,i)))+x_cord(i);
                flag_3_U=k;
                continue
            end
            if (((flag_3_U==k)&&(i~=i_max))&&(((phi_1(k,i)<(BS_U*dz))&&(phi_1(k,i+1)>(BS_U*dz)))||(phi_1(k,i)==(BS_U*dz))))
                    x_surf_2_U(k)=(((BS_U*dz-phi_1(k,i)).*(x_cord(i+1)-x_cord(i)))./(phi_1(k,i+1)-phi_1(k,i)))+x_cord(i);
            end
    end
end

% Convert all z_surf_U and x_surf_U to one array
xz_surf_U=[x_cord', z_surf_1_U;x_cord', z_surf_2_U;x_cord', z_surf_3_U;x_surf_1_U',z_cord;x_surf_2_U',z_cord];
for b=1:1:(3*i_max)
        if xz_surf_U(b,2)==0 %Means it was not called up
        xz_surf_U(b,1)=0;
        xz_surf_U(b,2)=0;
        end
end

for b=(3*i_max+1):1:b_max
        if xz_surf_U(b,1)==0
        xz_surf_U(b,1)=0;
        xz_surf_U(b,2)=0;
        end
end

%NEW LOWER BAND

z_surf_1_L=zeros(i_max,1);
z_surf_2_L=zeros(i_max,1);
z_surf_3_L=zeros(i_max,1);
for i=1:1:i_max
    flag_1_L=0;
    flag_2_L=0;
    for k=1:1:k_max
        if ((flag_1_L==0)&&(((phi_1(k,i)>(-dz*BS_L))&&(phi_1(k+1,i)<(-dz*BS_L)))||(phi_1(k,i)==(-dz*BS_L))))
            z_surf_1_L(i)=(((-dz*BS_L-phi_1(k,i)).*(z_cord(k+1)-z_cord(k)))./(phi_1(k+1,i)-phi_1(k,i)))+z_cord(k);
            flag_1_L=i;
            continue
        end
        if (((flag_2_L==0)&&(flag_1_L==i)&&(k~=k_max))&&(((phi_1(k,i)<(-dz*BS_L))&&(phi_1(k+1,i)>(-dz*BS_L)))||(phi_1(k,i)==(-dz*BS_L))))
            z_surf_2_L(i)=(((-dz*BS_L-phi_1(k,i)).*(z_cord(k+1)-z_cord(k)))./(phi_1(k+1,i)-phi_1(k,i)))+z_cord(k);
            flag_2_L=i;
            continue
        end
        if ((flag_2_L==i)&&(((phi_1(k,i)>(-dz*BS_L))&&(phi_1(k+1,i)<(-dz*BS_L)))||(phi_1(k,i)==(-dz*BS_L))))
            z_surf_3_L(i)=(((-dz*BS_L-phi_1(k,i)).*(z_cord(k+1)-z_cord(k)))./(phi_1(k+1,i)-phi_1(k,i)))+z_cord(k);
        end
    end
end

x_surf_1_L=zeros(1,k_max);
x_surf_2_L=zeros(1,k_max);
for k=1:1:k_max
    flag_3_L=0;
    for i=1:1:i_max
            if (((flag_3_L==0)&&(i~=i_max)&&(i~=1))&&(((phi_1(k,i)>(-dz*BS_L))&&(phi_1(k,i+1)<(-dz*BS_L)))||(phi_1(k,i)==(-dz*BS_L))))
                    x_surf_1_L(k)=(((-dz*BS_L-phi_1(k,i)).*(x_cord(i+1)-x_cord(i)))./(phi_1(k,i+1)-phi_1(k,i)))+x_cord(i);
                flag_3_L=k;
                continue
            end
            if (((flag_3_L==k)&&(i~=i_max))&&(((phi_1(k,i)<(-dz*BS_L))&&(phi_1(k,i+1)>(-dz*BS_L)))||(phi_1(k,i)==(-dz*BS_L))))
                    x_surf_2_L(k)=(((-dz*BS_L-phi_1(k,i)).*(x_cord(i+1)-x_cord(i)))./(phi_1(k,i+1)-phi_1(k,i)))+x_cord(i);
            end
    end
end

% Convert all z_surf_L and x_surf_L to one array
xz_surf_L=[x_cord', z_surf_1_L;x_cord', z_surf_2_L;x_cord', z_surf_3_L;x_surf_1_L',z_cord;x_surf_2_L',z_cord];
for b=1:1:(3*i_max)
        if xz_surf_L(b,2)==0 %Means it was not called up
        xz_surf_L(b,1)=0;
        xz_surf_L(b,2)=0;
        end
end

for b=(3*i_max+1):1:b_max
        if xz_surf_L(b,1)==0
        xz_surf_L(b,1)=0;
        xz_surf_L(b,2)=0;
        end
end

%Create flags for points IN the NEW Narrow Band
%These flags will only change after band is re-initialized
flag_NB=zeros(k_max,i_max); %Initialize flag_NB to zeros again
for k=1:1:k_max
    for i=1:1:i_max
        if (((phi_1(k,i)>=0)&&(abs(phi_1(k,i))<BS_U*dz))||((phi_1(k,i)<0)&&(abs(phi_1(k,i))<BS_L*dz))) %Don't consider points on boundary
            flag_NB(k,i)=1; %else they will remain 0 (are outside the band)
        end
    end
end

%Create flags to indicate NEW BC pts (adjacent to NB boundary)
%These will be later used to give phi_i=0 for values where flag_NB=2 in the
%beginning of while loop
%These flags will only change after band is re-initialized
for k=1:1:k_max
    for i=1:1:i_max

        if (i~=1)&&(i~=i_max)&&(k~=1)&&(k~=k_max)
            if (flag_NB(k,i)==0)&&((flag_NB(k,i+1)==1)||(flag_NB(k,i-1)==1)||(flag_NB(k+1,i)==1)||(flag_NB(k-1,i)==1))
                flag_NB(k,i)=2;
            end
        elseif (i==1)&&(k~=1)&&(k~=k_max)
            if (flag_NB(k,i)==0)&&((flag_NB(k,i+1)==1)||(flag_NB(k+1,i)==1)||(flag_NB(k-1,i)==1))
                flag_NB(k,i)=2;
            end
        elseif (i==i_max)&&(k~=1)&&(k~=k_max)
            if (flag_NB(k,i)==0)&&((flag_NB(k,i-1)==1)||(flag_NB(k+1,i)==1)||(flag_NB(k-1,i)==1))
                flag_NB(k,i)=2;
            end
        elseif (k==1)&&(i~=1)&&(i~=i_max)
            if (flag_NB(k,i)==0)&&((flag_NB(k,i+1)==1)||(flag_NB(k,i-1)==1)||(flag_NB(k+1,i)==1))
                flag_NB(k,i)=2;
            end
        elseif (k==k_max)&&(i~=1)&&(i~=i_max)
            if (flag_NB(k,i)==0)&&((flag_NB(k,i+1)==1)||(flag_NB(k,i-1)==1)||(flag_NB(k-1,i)==1))
                flag_NB(k,i)=2;
            end
        elseif  (i==1)&&(k==1)
            if (flag_NB(k,i)==0)&&((flag_NB(k,i+1)==1)||(flag_NB(k+1,i)==1))
                flag_NB(k,i)=2;
            end
        elseif  (i==i_max)&&(k==1)
            if (flag_NB(k,i)==0)&&((flag_NB(k,i-1)==1)||(flag_NB(k+1,i)==1))
                flag_NB(k,i)=2;
            end
        elseif  (i==1)&&(k==k_max)
            if (flag_NB(k,i)==0)&&((flag_NB(k,i+1)==1)||(flag_NB(k-1,i)==1))
                flag_NB(k,i)=2;
            end
        else %(i==i_max)&&(k==k_max)
            if (flag_NB(k,i)==0)&&((flag_NB(k,i-1)==1)||(flag_NB(k-1,i)==1))
                flag_NB(k,i)=2;
            end
        end

    end
```

```
        end

        %Create flags for the NEW remaining pts. so when rebuild band, we know where
        %phi's should be +ve or -ve; i.e. assign flag_NB(k,i) = 4 and 3 so as to
        %later indicate +ve and -ve phi
        for k=1:1:k_max
            for i=1:1:i_max
                if (flag_NB(k,i)~=1)&&(flag_NB(k,i)~=2)
                    if (phi_1(k,i)>0) %we only need to check +ve phi_1 since these are outside the band
                        flag_NB(k,i)=4; %label +ve phi_1's outside band
                    else
                        flag_NB(k,i)=3; %label -ve phi_1's outside band
                    end
                end
            end
        end

        flag_RE=1; %will not change until the rest of simulation
        counter_current=counter; %Stores current counter value at re-building
        Num_iter_RE_TOT=counter_current; %Assigns current counter value until next re-building

        No_RE=No_RE+1; %Counts number of re-initializations
        phi=zeros(k_max,i_max); %Done to ensure "old" phi's don't interfere with new ones in the update process

    end
end

%_____________________________________________________________________________
%Adjustment to Mass Flux due to Mask Model for Target M(x'): Algorithm to obtain mask angles for T

%Calculate visibility angles based on zero level set (M)
x_prime_surf_LM=zeros(b_max,1); %M surface in local coordinates
z_prime_surf_LM=zeros(b_max,1);
x_prime_surf_RM=zeros(b_max,1);
z_prime_surf_RM=zeros(b_max,1);

for b=1:1:b_max
    if ((xz_surf(b,1)>=x_min_grid)&&(xz_surf(b,1)<=(x_min_grid+leng_M_L)))&&...
            ((xz_surf(b,2)>=(h*sin(alfa)-H_m))&&(xz_surf(b,2)<h*sin(alfa))) %This check also ensures that if surf was not encountered
        % (i.e. xz surfs are 0) then it will ignore those values
        x_prime_surf_LM(b)=(xz_surf(b,1)-x_off).*sin(alfa)-xz_surf(b,2).*cos(alfa); %Rotated local x
        z_prime_surf_LM(b)=(xz_surf(b,1)-x_off).*cos(alfa)+xz_surf(b,2).*sin(alfa); %Rotated local z
    else
        x_prime_surf_LM(b)=NaN;
        z_prime_surf_LM(b)=NaN;
    end

    if ((xz_surf(b,1)>=(x_max_grid-leng_M_R))&&(xz_surf(b,1)<=x_max_grid))&&...
            ((xz_surf(b,2)>=(h*sin(alfa)-H_m))&&(xz_surf(b,2)<h*sin(alfa)))
        x_prime_surf_RM(b)=(xz_surf(b,1)-x_off).*sin(alfa)-xz_surf(b,2).*cos(alfa); %Rotated local x
        z_prime_surf_RM(b)=(xz_surf(b,1)-x_off).*cos(alfa)+xz_surf(b,2).*sin(alfa); %Rotated local z
    else
        x_prime_surf_RM(b)=NaN;
        z_prime_surf_RM(b)=NaN;
    end

end

%By evaluating Max of x_prime_surf_LM, we can check if any entries are
%positive, ignoring NaN's; if any are, then we have case (b),
%where mask shadow is >= W_m/2 from left mask edge, else have case (a) and
%(c), mask shadow is < W_m/2
[max_x_prime_surf_LM,I_max_LM]=max(x_prime_surf_LM); %Will ignore NaN's

%Min tan of left 'spread' angle defined by mask
if (max_x_prime_surf_LM>=0) %Case (b)
    tan_fi_min=max_x_prime_surf_LM/z_prime_surf_LM(I_max_LM);
else %Case (a) and (c)
    %Find min |x_prime_surf_LM|
    [min_x_prime_surf_LM,I_min_LM]=min(abs(x_prime_surf_LM));

    if (alfa==90*pi/180)
        tan_fi_min=min_x_prime_surf_LM/h;
    else
        tan_fi_min=min_x_prime_surf_LM/z_prime_surf_LM(I_min_LM);
    end
end

%Find min x_prime_surf_RM
[min_x_prime_surf_RM,I_min_RM]=min(x_prime_surf_RM);
```

```
%Max tan of right 'spread' angle defined by mask
if (alfa==90*pi/180)
    tan_fi_max=min_x_prime_surf_RM/h;
else
    tan_fi_max=min_x_prime_surf_RM/z_prime_surf_RM(I_min_RM);
end

%_________________________________________________________________
%Adjustment to Mass Flux due to Mask Model for Mask Edges M(x')

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Initialize variables before loop entry
NaN_Chk_xps_LM=isnan(x_prime_surf_LM); %If any entries are NaN, returns 1 for that entry, else 0
NaN_Chk_zps_LM=isnan(z_prime_surf_LM); %Mask surface in local coordinates (see above)
NaN_Chk_xps_RM=isnan(x_prime_surf_RM);
NaN_Chk_zps_RM=isnan(z_prime_surf_RM);

SDF_LM=zeros(b_max,1); %See below
SDF_RM=zeros(b_max,1);
J_k_M=0;
J_i_M=0;
%theta, cos(theta) at mask edges (left and right) and corresponding FDs
theta_LM=zeros(b_max,1);
cos_t_star_LM=zeros(b_max,1);
phi_x_pos_LM=zeros(b_max,1);
phi_x_neg_LM=zeros(b_max,1);
phi_z_pos_LM=zeros(b_max,1);
phi_z_neg_LM=zeros(b_max,1);
phi_x_star_LM=zeros(b_max,1);
phi_z_star_LM=zeros(b_max,1);

theta_RM=zeros(b_max,1);
cos_t_star_RM=zeros(b_max,1);
phi_x_pos_RM=zeros(b_max,1);
phi_x_neg_RM=zeros(b_max,1);
phi_z_pos_RM=zeros(b_max,1);
phi_z_neg_RM=zeros(b_max,1);
phi_x_star_RM=zeros(b_max,1);
phi_z_star_RM=zeros(b_max,1);
%Variables and limits used to evaluate M(x')edge (Masking function for M
%edge); See below
n_max=1001;
x_n_LM=zeros(1,n_max);
x_n_RM=zeros(1,n_max);
L_NU_LM=zeros(1,n_max);
L_NL_LM=zeros(1,n_max);
L_NU_RM=zeros(1,n_max);
L_NL_RM=zeros(1,n_max);
Mrx_M_edge_LM=zeros(1,n_max);
Mrx_M_edge_RM=zeros(1,n_max);

No_p_MrxM_ON=0; %Number of passes before turn on Masking Function for edge (non-0 only for RapidMask mask)
%Loop
if ((time*v_scan/(2*r_s))<=No_p_MrxM_ON)

        M_r_x_prime_LM=1;
        M_r_x_prime_RM=1;

else


%Calculate Wm_eff/2
if (max_x_prime_surf_LM>=0) %Case (b) (see above Flux adjustment model for T)
    x_prime_Wm_eff_2_LM=0; %rotated local x (used below)
    x_prime_Wm_eff_2_RM=min_x_prime_surf_RM-max_x_prime_surf_LM;  %Used below
else %Case (a) and (c)
    x_prime_Wm_eff_2_LM=min_x_prime_surf_LM; %rotated local x  (used below)
    x_prime_Wm_eff_2_RM=min_x_prime_surf_RM; % Used below
end
%Need to change variables to global values since theta calc. with global
%variables - value of x where mask edge starts
if (max_x_prime_surf_LM>=0) %Case (b) (see above Flux adjustment model for T)
    x_Wm_eff_2_LM=max_x_prime_surf_LM*sin(alfa)+z_prime_surf_LM(I_max_LM)*cos(alfa)+x_off; %Global x
    x_Wm_eff_2_RM=min_x_prime_surf_RM*sin(alfa)+z_prime_surf_RM(I_min_RM)*cos(alfa)+x_off; %See above results
else %Case (a) and (c)
    %For 1st expression below added a -1* at beginning since x_prime_Wm_eff_2_LM should be -ve but is +ve from above calcs.
    %and needs to stay +ve
    x_Wm_eff_2_LM=-1*x_prime_Wm_eff_2_LM*sin(alfa)+z_prime_surf_LM(I_min_LM)*cos(alfa)+x_off; %Global x
    x_Wm_eff_2_RM=x_prime_Wm_eff_2_RM*sin(alfa)+z_prime_surf_RM(I_min_RM)*cos(alfa)+x_off;
end
```

```
%Find x' limit beyond which have top of mask; for less than this limit
%we have mask edge - the point on M surface with min. dist. from
%(x',z')=(0,hsin(alfa)-Hm)= ref. pt.

%Define reference pt.
if (alfa==90*pi/180) %Need this condition to avoid small Matlab numerical error
   x_prime_ref=0; %Rotated local x
   z_prime_ref=(h*sin(alfa)-H_m); %Rotated local z
else
   x_prime_ref=((x_min_grid+leng_M_L+W_m/2)-x_off)*sin(alfa)-(h*sin(alfa)-H_m)*cos(alfa); %Rotated local x
   z_prime_ref=((x_min_grid+leng_M_L+W_m/2)-x_off)*cos(alfa)+(h*sin(alfa)-H_m)*sin(alfa); %Rotated local z
end
%Calculate min. distance from ref. pt. to mask edge
for b=1:1:b_max
      if (NaN_Chk_xps_LM(b)==0)&&(NaN_Chk_zps_LM(b)==0) %if Entry is NOT NaN
         SDF_LM(b)=((x_prime_ref-x_prime_surf_LM(b)).^2+(z_prime_ref-z_prime_surf_LM(b)).^2).^0.5;
      else
         SDF_LM(b)=NaN;
      end

      if (NaN_Chk_xps_RM(b)==0)&&(NaN_Chk_zps_RM(b)==0)
         SDF_RM(b)=((x_prime_ref-x_prime_surf_RM(b)).^2+(z_prime_ref-z_prime_surf_RM(b)).^2).^0.5;
      else
         SDF_RM(b)=NaN;
      end
end
[min_SDF_LM,J_min_LM]=min(SDF_LM); %Will ignore the NaNs
[min_SDF_RM,J_min_RM]=min(SDF_RM);
%Calculate x'tran
x_prime_lim_LM=x_prime_surf_LM(J_min_LM); %rotated local x
x_prime_lim_RM=x_prime_surf_RM(J_min_RM);
x_lim_LM=x_prime_lim_LM*sin(alfa)+z_prime_surf_LM(J_min_LM)*cos(alfa)+x_off; %Global x
x_lim_RM=x_prime_lim_RM*sin(alfa)+z_prime_surf_RM(J_min_RM)*cos(alfa)+x_off;

%Obtain theta avg. over mask edges

%%Avg. theta for Left Mask (LM) Edge - calculations analogous to 2nd strike algorithm

 for b=1:1:b_max

    if (((xz_surf(b,1)>=x_min_grid)&&(xz_surf(b,1)<=(x_min_grid+leng_M_L)))&&...
          ((xz_surf(b,2)>=(h*sin(alfa)-H_m))&&(xz_surf(b,2)<h*sin(alfa))))&&((xz_surf(b,1)>=(x_lim_LM-dx))&&(xz_surf(b,1)<=(x_Wm_eff_2_LM+dx))) %LM
edge & within dx
      %to obtain entire edge. This check also ensures that if surf. was not encountered (i.e. xz_surfs are 0) then it will ignore those values

        if (b>=1)&&(b<=(3*i_max)) %zsurf used

          if (b>=1)&&(b<=i_max)
             I_min_ik=b; %z_surf_1 used
          end
          if (b>=(i_max+1))&&(b<=(2*i_max))
             I_min_ik=b-i_max;  %z_surf_2 used
          end

          if (b>=(2*i_max+1))&&(b<=(3*i_max))
             I_min_ik=b-2*i_max;  %z_surf_3 used
          end

            %Calculate nearest k index to surface
            J_k_M=floor(k_max-(xz_surf(b,2)-z_min_grid)/dz);

            %Calculation of dphi/dx,dphi/dz; Define BC's
            if I_min_ik==i_max
               phi_x_pos_LM(b)=0;
            else
               phi_B_z_surf=((xz_surf(b,2)-z_cord(J_k_M))/(z_cord(J_k_M+1)-z_cord(J_k_M))).*(phi_1(J_k_M+1,I_min_ik+1)-
phi_1(J_k_M,I_min_ik+1))+phi_1(J_k_M,I_min_ik+1);
               phi_x_pos_LM(b)=(phi_B_z_surf-0)./dx;
            end

            if I_min_ik==1
               phi_x_neg_LM(b)=0;
            else
               phi_A_z_surf=((xz_surf(b,2)-z_cord(J_k_M))/(z_cord(J_k_M+1)-z_cord(J_k_M))).*(phi_1(J_k_M+1,I_min_ik-1)-phi_1(J_k_M,I_min_ik-
1))+phi_1(J_k_M,I_min_ik-1);
               phi_x_neg_LM(b)=(0-phi_A_z_surf)./dx;
            end

            if J_k_M==1
```

```
                phi_z_pos_LM(b)=0;
              else
                phi_C_z_surf=((xz_surf(b,2)+dz-z_cord(J_k_M-1))/(z_cord(J_k_M)-z_cord(J_k_M-1))).*(phi_1(J_k_M,I_min_ik)-phi_1(J_k_M-1,I_min_ik))+phi_1(J_k_M-1,I_min_ik);
                phi_z_pos_LM(b)=(phi_C_z_surf-0)./dz;
              end

              if J_k_M==k_max
                phi_z_neg_LM(b)=0;
              else
                phi_D_z_surf=((xz_surf(b,2)-dz-z_cord(J_k_M+1))/(z_cord(J_k_M+2)-z_cord(J_k_M+1))).*(phi_1(J_k_M+2,I_min_ik)-phi_1(J_k_M+1,I_min_ik))+phi_1(J_k_M+1,I_min_ik);
                phi_z_neg_LM(b)=(0-phi_D_z_surf)./dz;
              end

              %Calculate phi_stars
              phi_x_star_LM(b)=(phi_x_pos_LM(b)+phi_x_neg_LM(b))/2;
              phi_z_star_LM(b)=(phi_z_pos_LM(b)+phi_z_neg_LM(b))/2;

                cos_t_star_LM(b)=((xz_surf(b,1)-x_off).*(phi_x_star_LM(b))+xz_surf(b,2).*(phi_z_star_LM(b)))./(sqrt((xz_surf(b,1)-x_off).^2+xz_surf(b,2).^2).*sqrt((phi_x_star_LM(b)).^2+(phi_z_star_LM(b)).^2));

                if (cos_t_star_LM(b)>1)
                  cos_t_star_LM(b)=1;
                end
                theta_LM(b)=acos(cos_t_star_LM(b));
%_____________________________________________________________________________
          else  %xsurf used

            if (b>=(3*i_max+1))&&(b<=(3*i_max+k_max))
              I_min_ik=b-3*i_max; %x_surf_1 used
            end
            if (b>=(3*i_max+k_max+1))&&(b<=b_max)
              I_min_ik=b-3*i_max-k_max;  %x_surf_2 used
            end

              %Repeat above algorithm but for xsurf
              %Calculate nearest i index to surface
              J_i_M=floor(1+(xz_surf(b,1)-x_min_grid)/dx);

              %Calculation of dphi/dx,dphi/dz; Define BC's
              if J_i_M==i_max
                phi_x_pos_LM(b)=0;
              else
                phi_D_x_surf=((xz_surf(b,1)+dx-x_cord(J_i_M+1))/(x_cord(J_i_M+2)-x_cord(J_i_M+1))).*(phi_1(I_min_ik,J_i_M+2)-phi_1(I_min_ik,J_i_M+1))+phi_1(I_min_ik,J_i_M+1);
                phi_x_pos_LM(b)=(phi_D_x_surf-0)./dx;
              end

              if J_i_M==1
                phi_x_neg_LM(b)=0;
              else
                phi_C_x_surf=((xz_surf(b,1)-dx-x_cord(J_i_M-1))/(x_cord(J_i_M)-x_cord(J_i_M-1))).*(phi_1(I_min_ik,J_i_M)-phi_1(I_min_ik,J_i_M-1))+phi_1(I_min_ik,J_i_M-1);
                phi_x_neg_LM(b)=(0-phi_C_x_surf)./dx;
              end

              if I_min_ik==1
                phi_z_pos_LM(b)=0;
              else
                phi_A_x_surf=((xz_surf(b,1)-x_cord(J_i_M))/(x_cord(J_i_M+1)-x_cord(J_i_M))).*(phi_1(I_min_ik-1,J_i_M+1)-phi_1(I_min_ik-1,J_i_M))+phi_1(I_min_ik-1,J_i_M);
                phi_z_pos_LM(b)=(phi_A_x_surf-0)./dz;
              end

              if I_min_ik==k_max
                phi_z_neg_LM(b)=0;
              else
                phi_B_x_surf=((xz_surf(b,1)-x_cord(J_i_M))/(x_cord(J_i_M+1)-x_cord(J_i_M))).*(phi_1(I_min_ik+1,J_i_M+1)-phi_1(I_min_ik+1,J_i_M))+phi_1(I_min_ik+1,J_i_M);
                phi_z_neg_LM(b)=(0-phi_B_x_surf)./dz;
              end

              %Calculate phi_stars
              phi_x_star_LM(b)=(phi_x_pos_LM(b)+phi_x_neg_LM(b))/2;
              phi_z_star_LM(b)=(phi_z_pos_LM(b)+phi_z_neg_LM(b))/2;

                cos_t_star_LM(b)=((xz_surf(b,1)-x_off).*(phi_x_star_LM(b))+xz_surf(b,2).*(phi_z_star_LM(b)))./(sqrt((xz_surf(b,1)-x_off).^2+xz_surf(b,2).^2).*sqrt((phi_x_star_LM(b)).^2+(phi_z_star_LM(b)).^2));
```

```
                   if (cos_t_star_LM(b)>1)
                       cos_t_star_LM(b)=1;
                   end
                   theta_LM(b)=acos(cos_t_star_LM(b));

         end

     else
        theta_LM(b)=NaN;
     end

 end

 NaN_Chk_theta_LM=isnan(theta_LM); %If any entries are NaN, returns 1 for that entry, else 0
 count_theta_LM=0; %Mask edge points
 for b=1:1:b_max
     if (NaN_Chk_theta_LM(b)==0) %if Entry is NOT NaN (i.e. if surface encountered)
        theta_LM(b)=theta_LM(b);
        count_theta_LM=count_theta_LM+1;
     else
        theta_LM(b)=0; %It will not count towards the sum avg. if no surface
     end
 end
%Average theta (Left Mask Edge)
 if count_theta_LM==0 %To avoid 0/0 error
     theta_LM_avg=0; %This occurs if no mask edges seen yet
 else
     theta_LM_avg=sum(theta_LM)/count_theta_LM;
 end

 %Avg. theta for Right Mask (RM) Edge (analogous to LM calcs.)

%Initialize variables
J_k_M=0; %See below
J_i_M=0;

 for b=1:1:b_max

     if (((xz_surf(b,1)>=(x_max_grid-leng_M_R))&&(xz_surf(b,1)<=x_max_grid))&&...
            ((xz_surf(b,2)>=(h*sin(alfa)-H_m))&&(xz_surf(b,2)<h*sin(alfa))))&&((xz_surf(b,1)>=(x_Wm_eff_2_RM-dx))&&(xz_surf(b,1)<=(x_lim_RM+dx)))

         if (b>=1)&&(b<=(3*i_max)) %zsurf used

           if (b>=1)&&(b<=i_max)
              I_min_ik=b; %z_surf_1 used
           end
           if (b>=(i_max+1))&&(b<=(2*i_max))
              I_min_ik=b-i_max;  %z_surf_2 used
           end

           if (b>=(2*i_max+1))&&(b<=(3*i_max))
              I_min_ik=b-2*i_max;  %z_surf_3 used
           end

               %Calculate nearest k index to surface
               J_k_M=floor(k_max-(xz_surf(b,2)-z_min_grid)/dz);

               %Calculation of dphi/dx,dphi/dz; Define BC's
               if I_min_ik==i_max
                   phi_x_pos_RM(b)=0;
               else
                   phi_B_z_surf=((xz_surf(b,2)-z_cord(J_k_M))/(z_cord(J_k_M+1)-z_cord(J_k_M))).*(phi_1(J_k_M+1,I_min_ik+1)-
phi_1(J_k_M,I_min_ik+1))+phi_1(J_k_M,I_min_ik+1);
                   phi_x_pos_RM(b)=(phi_B_z_surf-0)./dx;
               end

               if I_min_ik==1
                   phi_x_neg_RM(b)=0;
               else
                   phi_A_z_surf=((xz_surf(b,2)-z_cord(J_k_M))/(z_cord(J_k_M+1)-z_cord(J_k_M))).*(phi_1(J_k_M+1,I_min_ik-1)-phi_1(J_k_M,I_min_ik-
1))+phi_1(J_k_M,I_min_ik-1);
                   phi_x_neg_RM(b)=(0-phi_A_z_surf)./dx;
               end

               if J_k_M==1
                   phi_z_pos_RM(b)=0;
               else
                   phi_C_z_surf=((xz_surf(b,2)+dz-z_cord(J_k_M-1))/(z_cord(J_k_M)-z_cord(J_k_M-1))).*(phi_1(J_k_M,I_min_ik)-phi_1(J_k_M-
1,I_min_ik))+phi_1(J_k_M-1,I_min_ik);
                   phi_z_pos_RM(b)=(phi_C_z_surf-0)./dz;
```

```
          end

          if J_k_M==k_max
            phi_z_neg_RM(b)=0;
          else
            phi_D_z_surf=((xz_surf(b,2)-dz-z_cord(J_k_M+1))/(z_cord(J_k_M+2)-z_cord(J_k_M+1))).*(phi_1(J_k_M+2,I_min_ik)-
phi_1(J_k_M+1,I_min_ik))+phi_1(J_k_M+1,I_min_ik);
            phi_z_neg_RM(b)=(0-phi_D_z_surf)./dz;
          end

          %Calculate phi_stars
          phi_x_star_RM(b)=(phi_x_pos_RM(b)+phi_x_neg_RM(b))/2;
          phi_z_star_RM(b)=(phi_z_pos_RM(b)+phi_z_neg_RM(b))/2;

            cos_t_star_RM(b)=((xz_surf(b,1)-x_off).*(phi_x_star_RM(b))+xz_surf(b,2).*(phi_z_star_RM(b)))./(sqrt((xz_surf(b,1)-
x_off).^2+xz_surf(b,2).^2).*sqrt((phi_x_star_RM(b)).^2+(phi_z_star_RM(b)).^2));

            if (cos_t_star_RM(b)>1)
              cos_t_star_RM(b)=1;
            end
            theta_RM(b)=acos(cos_t_star_RM(b));
%_____________________________________________________________________________
      else  %xsurf used

        if (b>=(3*i_max+1))&&(b<=(3*i_max+k_max))
          I_min_ik=b-3*i_max; %x_surf_1 used
        end
        if (b>=(3*i_max+k_max+1))&&(b<=b_max)
          I_min_ik=b-3*i_max-k_max;  %x_surf_2 used
        end

          %Repeat above algorithm but for xsurf
          %Calculate nearest i index to surface
          J_i_M=floor(1+(xz_surf(b,1)-x_min_grid)/dx);

          %Calculation of dphi/dx,dphi/dz; Define BC's
          if J_i_M==i_max
            phi_x_pos_RM(b)=0;
          else
            phi_D_x_surf=((xz_surf(b,1)+dx-x_cord(J_i_M+1))/(x_cord(J_i_M+2)-x_cord(J_i_M+1))).*(phi_1(I_min_ik,J_i_M+2)-
phi_1(I_min_ik,J_i_M+1))+phi_1(I_min_ik,J_i_M+1);
            phi_x_pos_RM(b)=(phi_D_x_surf-0)./dx;
          end

          if J_i_M==1
            phi_x_neg_RM(b)=0;
          else
            phi_C_x_surf=((xz_surf(b,1)-dx-x_cord(J_i_M-1))/(x_cord(J_i_M)-x_cord(J_i_M-1))).*(phi_1(I_min_ik,J_i_M)-phi_1(I_min_ik,J_i_M-
1))+phi_1(I_min_ik,J_i_M-1);
            phi_x_neg_RM(b)=(0-phi_C_x_surf)./dx;
          end

          if I_min_ik==1
            phi_z_pos_RM(b)=0;
          else
            phi_A_x_surf=((xz_surf(b,1)-x_cord(J_i_M))/(x_cord(J_i_M+1)-x_cord(J_i_M))).*(phi_1(I_min_ik-1,J_i_M+1)-phi_1(I_min_ik-
1,J_i_M))+phi_1(I_min_ik-1,J_i_M);
            phi_z_pos_RM(b)=(phi_A_x_surf-0)./dz;
          end

          if I_min_ik==k_max
            phi_z_neg_RM(b)=0;
          else
            phi_B_x_surf=((xz_surf(b,1)-x_cord(J_i_M))/(x_cord(J_i_M+1)-x_cord(J_i_M))).*(phi_1(I_min_ik+1,J_i_M+1)-
phi_1(I_min_ik+1,J_i_M))+phi_1(I_min_ik+1,J_i_M);
            phi_z_neg_RM(b)=(0-phi_B_x_surf)./dz;
          end

          %Calculate phi_stars
          phi_x_star_RM(b)=(phi_x_pos_RM(b)+phi_x_neg_RM(b))/2;
          phi_z_star_RM(b)=(phi_z_pos_RM(b)+phi_z_neg_RM(b))/2;

            cos_t_star_RM(b)=((xz_surf(b,1)-x_off).*(phi_x_star_RM(b))+xz_surf(b,2).*(phi_z_star_RM(b)))./(sqrt((xz_surf(b,1)-
x_off).^2+xz_surf(b,2).^2).*sqrt((phi_x_star_RM(b)).^2+(phi_z_star_RM(b)).^2));

            if (cos_t_star_RM(b)>1)
              cos_t_star_RM(b)=1;
            end
            theta_RM(b)=acos(cos_t_star_RM(b));
          end

      else
         theta_RM(b)=NaN;
      end

  end

   NaN_Chk_theta_RM=isnan(theta_RM); %If any entries are NaN, returns 1 for that entry, else 0
  count_theta_RM=0;
  for b=1:1:b_max
      if (NaN_Chk_theta_RM(b)==0) %if Entry is NOT NaN
         theta_RM(b)=theta_RM(b);
         count_theta_RM=count_theta_RM+1;
      else
         theta_RM(b)=0; %It will not count towards the sum avg. (no surface)
      end
  end

 %Average theta (Right Mask Edge)
  if count_theta_RM==0 %To avoid 0/0 error
      theta_RM_avg=0; %This occurs if no mask edges seen yet
  else
      theta_RM_avg=sum(theta_RM)/count_theta_RM;
  end

 %Effective height (LM,RM)

 %Need to obtain global z_lim LM/RM to calculate Heff90 LM/RM from local
 %x'lim, z'lim (see above)
 z_lim_LM=-x_prime_surf_LM(J_min_LM)*cos(alfa)+z_prime_surf_LM(J_min_LM)*sin(alfa); %Global x
 z_lim_RM=-x_prime_surf_RM(J_min_RM)*cos(alfa)+z_prime_surf_RM(J_min_RM)*sin(alfa);

 if (alfa==90*pi/180)
        Heff_LM=h+dz-z_prime_surf_LM(J_min_LM); %Effective height
        Heff_RM=h+dz-z_prime_surf_RM(J_min_RM); %within dz to obtain top of mask
 else %alfa<90deg

        Heff_LM_90=h*sin(alfa)+dz-z_lim_LM; %Vertical Effective height
        Heff_RM_90=h*sin(alfa)+dz-z_lim_RM;

        Heff_LM=(Heff_LM_90*sin(theta_LM_avg))/(sin(-pi/2+alfa+theta_LM_avg));
        Heff_RM=(Heff_RM_90*sin(theta_RM_avg))/(sin(pi/2-alfa+theta_RM_avg));
 end

 %Using avg. theta, Hmeff and Wmeff/2 calculate the masking
 %function for the mask edges

 %Calculate d_m
 if ((theta_LM_avg==0)||(theta_LM_avg>(pi/2))) %When LM edge not seen yet (for alfa=90deg and alfa<90deg), thus d_m=0
     d_m_LM=0; %Need this so cot(0) does not result in infinity
 else
     d_m_LM=Heff_LM*cot(theta_LM_avg);
 end

 if theta_RM_avg==0 %When RM edge not seen yet (alfa=90deg only), thus d_m=0
     d_m_RM=0; %Need this so cot(0) does not result in infinity
 else
     d_m_RM=Heff_RM*cot(theta_RM_avg);
 end

 %LM x' integration range
 x_n_min_LM=0;
 x_n_max_LM=d_m_LM+x_prime_Wm_eff_2_LM;
 dx_n_LM=(x_n_max_LM-x_n_min_LM)/(n_max-1);

 for n=1:1:n_max
     x_n_LM(n)=(n-1).*dx_n_LM;
 end
 %RM x' integration range
 x_n_min_RM=0;
 x_n_max_RM=d_m_RM+x_prime_Wm_eff_2_RM;
 dx_n_RM=(x_n_max_RM-x_n_min_RM)/(n_max-1);

 for n=1:1:n_max
     x_n_RM(n)=(n-1).*dx_n_RM;
 end

 %Define limits of integration
 %LM
```

```
%Upper numerator limit
for n=1:1:n_max
   L_NU_LM(n)=d_m_LM+x_prime_Wm_eff_2_LM-x_n_LM(n);
end

 %Lower numerator limit
for n=1:1:n_max
   L_NL_LM(n)=x_prime_Wm_eff_2_LM-x_n_LM(n);
end

L_DU_LM=d_m_LM+x_prime_Wm_eff_2_LM; %Upper denominator limit (Note: lower denom. limit is 0)
%RM
 %Upper numerator limit
for n=1:1:n_max
   L_NU_RM(n)=d_m_RM+x_prime_Wm_eff_2_RM-x_n_RM(n);
end

 %Lower numerator limit
for n=1:1:n_max
   L_NL_RM(n)=x_prime_Wm_eff_2_RM-x_n_RM(n);
end

L_DU_RM=d_m_RM+x_prime_Wm_eff_2_RM; %Upper denominator limit (Note: lower denom. limit is 0)

%LM Integral: closed form fit
n_max_new_LM=0;
for n=1:1:n_max-1
   if (x_n_LM(n)<=x_prime_Wm_eff_2_LM)
      Mrx_M_edge_LM(n)=0;
   else
      Mrx_M_edge_LM(n)=real((erf(-P_2*log(L_NU_LM(n))+P_3)+1)./(1-erf(P_2*log(L_DU_LM)-P_3)));
      n_max_new_LM=n_max_new_LM+1;
   end
end
Mrx_M_edge_LM(n_max)=0; %Need this since error results for L_NU_LM(1001) = very small

%RM Integral: closed form fit
n_max_new_RM=0;
for n=1:1:n_max-1
   if (x_n_RM(n)<=x_prime_Wm_eff_2_RM)
      Mrx_M_edge_RM(n)=0;
   else
      Mrx_M_edge_RM(n)=real((erf(-P_2*log(L_NU_RM(n))+P_3)+1)./(1-erf(P_2*log(L_DU_RM)-P_3)));
      n_max_new_RM=n_max_new_RM+1;
   end
end
Mrx_M_edge_RM(n_max)=0; %Need this since error results for L_NU_RM(1001) = very small

%Define avg. M(x')_edge over edge
M_r_x_prime_LM=sum(Mrx_M_edge_LM)/n_max_new_LM;
M_r_x_prime_RM=sum(Mrx_M_edge_RM)/n_max_new_RM;


end

%____________________________________________________________________________
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%____________________________________________________________________________


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%START of 2nd Strike Algorithm %%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%See initial iteration (analogous here)

%Initialize variables before loop entry
J_k_AR=0;
J_i_AR=0;
phi_x_pos_AR=zeros(b_max,1);
phi_x_neg_AR=zeros(b_max,1);
phi_x_cen_AR=zeros(b_max,1);
phi_z_pos_AR=zeros(b_max,1);
phi_z_neg_AR=zeros(b_max,1);
phi_z_cen_AR=zeros(b_max,1);
phi_x_star_AR=zeros(b_max,1);
phi_z_star_AR=zeros(b_max,1);
cos_t_star_AR=zeros(b_max,1);
cos_g_star_AR=zeros(b_max,1);
```

```
theta_AR=zeros(b_max,1);
theta_AR_deg=zeros(b_max,1);
gamma_AR=zeros(b_max,1);
gamma_AR_deg=zeros(b_max,1);
x_prime_surf_AR=zeros(b_max,1);
f_alfa_AR=zeros(b_max,1);
theta_DE=zeros(b_max,1);
theta_DE_deg=zeros(b_max,1);

 for b=1:1:b_max

     if ((xz_surf(b,1)~=0)&&(xz_surf(b,2)~=0))||((alfa==(90*pi/180))&&((b==1)||(b==(i_max+1))||(b==(2*i_max+1)))&&(xz_surf(b,1)==0)&&(xz_surf(b,2)~=0))
%If surface

          if (b>=1)&&(b<=(3*i_max)) %zsurf used

            if (b>=1)&&(b<=i_max)
               I_min_ik=b; %z_surf_1 used
            end
            if (b>=(i_max+1))&&(b<=(2*i_max))
               I_min_ik=b-i_max;  %z_surf_2 used
            end

            if (b>=(2*i_max+1))&&(b<=(3*i_max))
               I_min_ik=b-2*i_max;  %z_surf_3 used
            end

                %Calculate nearest k index to surface
                J_k_AR=floor(k_max-(xz_surf(b,2)-z_min_grid)/dz);

                %Calculation of dphi/dx,dphi/dz; Define BC's
                if I_min_ik==i_max
                    phi_x_pos_AR(b)=0;
                else
                    phi_B_z_surf=((xz_surf(b,2)-z_cord(J_k_AR))/(z_cord(J_k_AR+1)-z_cord(J_k_AR))).*(phi_1(J_k_AR+1,I_min_ik+1)-
phi_1(J_k_AR,I_min_ik+1))+phi_1(J_k_AR,I_min_ik+1);
                    phi_x_pos_AR(b)=(phi_B_z_surf-0)./dx;
                end

                if I_min_ik==1
                    phi_x_neg_AR(b)=0;
                else
                    phi_A_z_surf=((xz_surf(b,2)-z_cord(J_k_AR))/(z_cord(J_k_AR+1)-z_cord(J_k_AR))).*(phi_1(J_k_AR+1,I_min_ik-1)-phi_1(J_k_AR,I_min_ik-
1))+phi_1(J_k_AR,I_min_ik-1);
                    phi_x_neg_AR(b)=(0-phi_A_z_surf)./dx;
                end

                if (I_min_ik==i_max)||(I_min_ik==1)
                    phi_x_cen_AR(b)=0;
                else
                    phi_A_z_surf=((xz_surf(b,2)-z_cord(J_k_AR))/(z_cord(J_k_AR+1)-z_cord(J_k_AR))).*(phi_1(J_k_AR+1,I_min_ik-1)-phi_1(J_k_AR,I_min_ik-
1))+phi_1(J_k_AR,I_min_ik-1);
                    phi_B_z_surf=((xz_surf(b,2)-z_cord(J_k_AR))/(z_cord(J_k_AR+1)-z_cord(J_k_AR))).*(phi_1(J_k_AR+1,I_min_ik+1)-
phi_1(J_k_AR,I_min_ik+1))+phi_1(J_k_AR,I_min_ik+1);
                    phi_x_cen_AR(b)=(phi_B_z_surf-phi_A_z_surf)./(2*dx);
                end

                if J_k_AR==1
                    phi_z_pos_AR(b)=0;
                else
                    phi_C_z_surf=((xz_surf(b,2)+dz-z_cord(J_k_AR-1))/(z_cord(J_k_AR)-z_cord(J_k_AR-1))).*(phi_1(J_k_AR,I_min_ik)-phi_1(J_k_AR-
1,I_min_ik))+phi_1(J_k_AR-1,I_min_ik);
                    phi_z_pos_AR(b)=(phi_C_z_surf-0)./dz;
                end

                if J_k_AR==k_max
                    phi_z_neg_AR(b)=0;
                else
                    phi_D_z_surf=((xz_surf(b,2)-dz-z_cord(J_k_AR+1))/(z_cord(J_k_AR+2)-z_cord(J_k_AR+1))).*(phi_1(J_k_AR+2,I_min_ik)-
phi_1(J_k_AR+1,I_min_ik))+phi_1(J_k_AR+1,I_min_ik);
                    phi_z_neg_AR(b)=(0-phi_D_z_surf)./dz;
                end

                if (J_k_AR==k_max)||(J_k_AR==1)
                    phi_z_cen_AR(b)=0;
                else
                    phi_C_z_surf=((xz_surf(b,2)+dz-z_cord(J_k_AR-1))/(z_cord(J_k_AR)-z_cord(J_k_AR-1))).*(phi_1(J_k_AR,I_min_ik)-phi_1(J_k_AR-
1,I_min_ik))+phi_1(J_k_AR-1,I_min_ik);
```

```
          phi_D_z_surf=((xz_surf(b,2)-dz-z_cord(J_k_AR+1))/(z_cord(J_k_AR+2)-z_cord(J_k_AR+1))).*(phi_1(J_k_AR+2,I_min_ik)-
phi_1(J_k_AR+1,I_min_ik))+phi_1(J_k_AR+1,I_min_ik);
          phi_z_cen_AR(b)=(phi_C_z_surf-phi_D_z_surf)./(2*dz);
        end

        %Calculate phi_stars
        phi_x_star_AR(b)=(phi_x_pos_AR(b)+phi_x_neg_AR(b))/2;
        phi_z_star_AR(b)=(phi_z_pos_AR(b)+phi_z_neg_AR(b))/2;

        %Calculate Angles
          cos_t_star_AR(b)=((xz_surf(b,1)-x_off).*(phi_x_star_AR(b))+xz_surf(b,2).*(phi_z_star_AR(b)))./(sqrt((xz_surf(b,1)-
x_off).^2+xz_surf(b,2).^2).*sqrt((phi_x_star_AR(b)).^2+(phi_z_star_AR(b)).^2));
          theta_AR(b)=acos(cos_t_star_AR(b));
          theta_AR_deg(b)=theta_AR(b)*(180/pi);
          cos_g_star_AR(b)=((xz_surf(b,1)-x_off).*(-phi_z_star_AR(b))+xz_surf(b,2).*(phi_x_star_AR(b)))./(sqrt((xz_surf(b,1)-
x_off).^2+xz_surf(b,2).^2).*sqrt((phi_x_star_AR(b)).^2+(phi_z_star_AR(b)).^2));
          gamma_AR(b)=acos(cos_g_star_AR(b));
          gamma_AR_deg(b)=gamma_AR(b)*(180/pi);
          if (alfa==90*pi/180)
            x_prime_surf_AR(b)=(xz_surf(b,1)-x_off);
          else
            x_prime_surf_AR(b)=(xz_surf(b,1)-x_off).*sin(alfa)-xz_surf(b,2).*cos(alfa);
          end

          if (xz_surf(b,2)>=(h*sin(alfa))) %T
            f_alfa_AR(b)=f_alfa_AR_T;
          else %M
            f_alfa_AR(b)=f_alfa_AR_M;
          end
          theta_DE(b)=pi-f_alfa_AR(b).*theta_AR(b);
          theta_DE_deg(b)=theta_DE(b)*(180/pi);
%___________________________________________________________________________
     else  %xsurf used

      if (b>=(3*i_max+1))&&(b<=(3*i_max+k_max))
        I_min_ik=b-3*i_max; %x_surf_1 used
      end
      if (b>=(3*i_max+k_max+1))&&(b<=b_max)
        I_min_ik=b-3*i_max-k_max;  %x_surf_2 used
      end

        %Repeat above algorithm but for xsurf
        %Calculate nearest i index to surface
        J_i_AR=floor(1+(xz_surf(b,1)-x_min_grid)/dx);

        %Calculation of dphi/dx,dphi/dz - Define BC's
        if J_i_AR==i_max
          phi_x_pos_AR(b)=0;
        else
          phi_D_x_surf=((xz_surf(b,1)+dx-x_cord(J_i_AR+1))/(x_cord(J_i_AR+2)-x_cord(J_i_AR+1))).*(phi_1(I_min_ik,J_i_AR+2)-
phi_1(I_min_ik,J_i_AR+1))+phi_1(I_min_ik,J_i_AR+1);
          phi_x_pos_AR(b)=(phi_D_x_surf-0)./dx;
        end

        if J_i_AR==1
          phi_x_neg_AR(b)=0;
        else
          phi_C_x_surf=((xz_surf(b,1)-dx-x_cord(J_i_AR-1))/(x_cord(J_i_AR)-x_cord(J_i_AR-1))).*(phi_1(I_min_ik,J_i_AR)-phi_1(I_min_ik,J_i_AR-
1))+phi_1(I_min_ik,J_i_AR-1);
          phi_x_neg_AR(b)=(0-phi_C_x_surf)./dx;
        end

        if (J_i_AR==i_max)||(J_i_AR==1)
          phi_x_cen_AR(b)=0;
        else
          phi_C_x_surf=((xz_surf(b,1)-dx-x_cord(J_i_AR-1))/(x_cord(J_i_AR)-x_cord(J_i_AR-1))).*(phi_1(I_min_ik,J_i_AR)-phi_1(I_min_ik,J_i_AR-
1))+phi_1(I_min_ik,J_i_AR-1);
          phi_D_x_surf=((xz_surf(b,1)+dx-x_cord(J_i_AR+1))/(x_cord(J_i_AR+2)-x_cord(J_i_AR+1))).*(phi_1(I_min_ik,J_i_AR+2)-
phi_1(I_min_ik,J_i_AR+1))+phi_1(I_min_ik,J_i_AR+1);
          phi_x_cen_AR(b)=(phi_D_x_surf-phi_C_x_surf)./(2*dx);
        end

        if I_min_ik==1
          phi_z_pos_AR(b)=0;
        else
          phi_A_x_surf=((xz_surf(b,1)-x_cord(J_i_AR))/(x_cord(J_i_AR+1)-x_cord(J_i_AR))).*(phi_1(I_min_ik-1,J_i_AR+1)-phi_1(I_min_ik-
1,J_i_AR))+phi_1(I_min_ik-1,J_i_AR);
          phi_z_pos_AR(b)=(phi_A_x_surf-0)./dz;
        end
        if I_min_ik==k_max
          phi_z_neg_AR(b)=0;
        else
          phi_B_x_surf=((xz_surf(b,1)-x_cord(J_i_AR))/(x_cord(J_i_AR+1)-x_cord(J_i_AR))).*(phi_1(I_min_ik+1,J_i_AR+1)-
phi_1(I_min_ik+1,J_i_AR))+phi_1(I_min_ik+1,J_i_AR);
          phi_z_neg_AR(b)=(0-phi_B_x_surf)./dz;
        end

        if (I_min_ik==k_max)||(I_min_ik==1)
          phi_z_cen_AR(b)=0;
        else
          phi_A_x_surf=((xz_surf(b,1)-x_cord(J_i_AR))/(x_cord(J_i_AR+1)-x_cord(J_i_AR))).*(phi_1(I_min_ik-1,J_i_AR+1)-phi_1(I_min_ik-
1,J_i_AR))+phi_1(I_min_ik-1,J_i_AR);
          phi_B_x_surf=((xz_surf(b,1)-x_cord(J_i_AR))/(x_cord(J_i_AR+1)-x_cord(J_i_AR))).*(phi_1(I_min_ik+1,J_i_AR+1)-
phi_1(I_min_ik+1,J_i_AR))+phi_1(I_min_ik+1,J_i_AR);
          phi_z_cen_AR(b)=(phi_A_x_surf-phi_B_x_surf)./(2*dz);
        end

        %Calculate phi_stars
        phi_x_star_AR(b)=(phi_x_pos_AR(b)+phi_x_neg_AR(b))/2;
        phi_z_star_AR(b)=(phi_z_pos_AR(b)+phi_z_neg_AR(b))/2;

        %Calculate Angles and x_surf'_AR
          cos_t_star_AR(b)=((xz_surf(b,1)-x_off).*(phi_x_star_AR(b))+xz_surf(b,2).*(phi_z_star_AR(b)))./(sqrt((xz_surf(b,1)-
x_off).^2+xz_surf(b,2).^2).*sqrt((phi_x_star_AR(b)).^2+(phi_z_star_AR(b)).^2));
          theta_AR(b)=acos(cos_t_star_AR(b));
          theta_AR_deg(b)=theta_AR(b)*(180/pi);
          cos_g_star_AR(b)=((xz_surf(b,1)-x_off).*(-phi_z_star_AR(b))+xz_surf(b,2).*(phi_x_star_AR(b)))./(sqrt((xz_surf(b,1)-
x_off).^2+xz_surf(b,2).^2).*sqrt((phi_x_star_AR(b)).^2+(phi_z_star_AR(b)).^2));
          gamma_AR(b)=acos(cos_g_star_AR(b));
          gamma_AR_deg(b)=gamma_AR(b)*(180/pi);
          if (alfa==90*pi/180)
            x_prime_surf_AR(b)=(xz_surf(b,1)-x_off);
          else
            x_prime_surf_AR(b)=(xz_surf(b,1)-x_off).*sin(alfa)-xz_surf(b,2).*cos(alfa);
          end

          if (xz_surf(b,2)>=(h*sin(alfa))) %T
            f_alfa_AR(b)=f_alfa_AR_T;
          else %M
            f_alfa_AR(b)=f_alfa_AR_M;
          end
          theta_DE(b)=pi-f_alfa_AR(b).*theta_AR(b);
          theta_DE_deg(b)=theta_DE(b)*(180/pi);

     end

  else  %Need this since it accounts for cases where surface wasn't encountered (i.e. x_surf and z_surf = 0 numerically)
     %and theta_AR(b) would stay 0 since pre-allocated with 0's for speed and the fact that theta_AR can actually = 0
    theta_AR(b)=NaN;
    theta_AR_deg(b)=NaN;
    gamma_AR(b)=NaN;
    gamma_AR_deg(b)=NaN;
    x_prime_surf_AR(b)=NaN;
    f_alfa_AR(b)=NaN;
    theta_DE(b)=NaN;
    theta_DE_deg(b)=NaN;
  end

 end
NaN_Chk_theta_AR=isnan(theta_AR); %If any entries are NaN, returns 1 for that entry, else 0

%2nd Strike Detection Algorithm (see initial iteration)

f_v_AR=zeros(b_max,1);
f_v_AR_fin=zeros(b_max,1);
v_AR=zeros(b_max,1);
flux_AR=zeros(b_max,1);
theta_D=zeros(b_max,1);
theta_D_deg=zeros(b_max,1);

cos_t_pfx_pfz_D=zeros(b_max,1);
cos_t_pfx_nfz_D=zeros(b_max,1);
cos_t_nfx_pfz_D=zeros(b_max,1);
cos_t_nfx_nfz_D=zeros(b_max,1);
cos_t_star_D=zeros(b_max,1);
cos_t_cen_D=zeros(b_max,1);

c_max=b_max;
```

```
%See first iteration  for definition of ds_crit and No_ds_cr
U_D_AR_dist=zeros(c_max,1); %Distance between nodes D and AR
theta_D_prime=zeros(c_max,1);
theta_D_1=zeros(c_max,1); %theta_D only for checking in c=1...cmax loop
ds_pre=zeros(c_max,1);

I_min_ds=zeros(b_max,1);
min_ds=zeros(b_max,1); %Min spacing, after 2nd strk. for each node AR (not necessarily small enough to include 2nd strk yet)

for b=1:1:b_max %Check each D node
    if (((xz_surf(b,1)~=0)&&(xz_surf(b,2)~=0))||((alfa==(90*pi/180))&&((b==1)||(b==(i_max+1))||(b==(2*i_max+1)))&&(xz_surf(b,1)==0)&&(xz_surf(b,2)~=0)))...
            &&((xz_surf(b,2)>=(h*sin(alfa)))||((xz_surf(b,2)<(h*sin(alfa)))&&(xz_surf(b,1)>=(x_lim_LM-dx))&&(xz_surf(b,1)<=(x_lim_RM+dx))))
     %Check if surf. found (numerically) AND if surf is not top of mask

        for c=1:1:c_max %Check AR nodes for each D node
                if (NaN_Chk_theta_AR(c)==0)&&((xz_surf(c,2)>=(h*sin(alfa)))||((xz_surf(c,2)<(h*sin(alfa)))&&(xz_surf(c,1)>=(x_lim_LM-
dx))&&(xz_surf(c,1)<=(x_lim_RM+dx))))...
                    &&(b~=c)&&((theta_AR(c)<(pi/2))&&(theta_AR(c)>0))&&((theta_DE(c)<pi)&&(theta_DE(c)>(pi/2)))...

&&(((gamma_AR(c)>(pi/2))&&(x_prime_surf_AR(b)>x_prime_surf_AR(c)))||((gamma_AR(c)<(pi/2))&&(x_prime_surf_AR(c)>x_prime_surf_AR(b))))
                   %Check if surf found (numerically)-this check supercedes next checks; if surf is not top of mask; ignore check at node D=AR;
                   %limit range of theta_AR; limit range of theta_DE; check if rebound direction makes sense

                   %Calculate theta_D' and theta_D
                   theta_D_prime(c)=acos(((xz_surf(b,1)-xz_surf(c,1)).*(phi_x_star_AR(c))+(xz_surf(b,2)-xz_surf(c,2)).*(phi_z_star_AR(c)))./...
                      (sqrt((xz_surf(b,1)-xz_surf(c,1)).^2+(xz_surf(b,2)-xz_surf(c,2)).^2).*sqrt((phi_x_star_AR(c)).^2+(phi_z_star_AR(c)).^2)));

                   theta_D_1(c)=acos(((xz_surf(b,1)-xz_surf(c,1)).*(phi_x_star_AR(b))+(xz_surf(b,2)-xz_surf(c,2)).*(phi_z_star_AR(b)))./...
                      (sqrt((xz_surf(b,1)-xz_surf(c,1)).^2+(xz_surf(b,2)-xz_surf(c,2)).^2).*sqrt((phi_x_star_AR(b)).^2+(phi_z_star_AR(b)).^2)));

                   %Calculate distance between nodes AR and D
                   U_D_AR_dist(c)=sqrt((xz_surf(b,1)-xz_surf(c,1)).^2+(xz_surf(b,2)-xz_surf(c,2)).^2);

                   if ((theta_D_prime(c)>(pi/2))&&(theta_D_1(c)<(pi/2))) %If node D 'seen' by node AR
                      ds_pre(c)=U_D_AR_dist(c).*tan(abs(theta_D_prime(c)-theta_DE(c)));
                   else
                      ds_pre(c)=NaN;
                   end

                else
                   ds_pre(c)=NaN;
                end
        end
         [min_ds(b),I_min_ds(b)]=min(ds_pre); %Find min_ds, Will ignore NaN's

         if (I_min_ds(b)~=0)&&(min_ds(b)<ds_crit)&&(U_D_AR_dist(I_min_ds(b))>(No_ds_cr*ds_crit))
            %Check if possibility of 2nd strike even occurred, ds_min<ds_crit and if U_D_AR_dist is large enough

                        %Calculate 2nd strike values - f_v_AR_fin, v_AR,
                        %flux_AR, assign calc'd theta_D

                        %f_v_AR_fin
                        if (xz_surf(b,2)>=(h*sin(alfa))) %T
                            f_v_AR(b)=f_v_AR_T;
                        else %M
                            f_v_AR(b)=f_v_AR_M;
                        end
                        f_v_AR_fin(b)=f_v_AR(b).*((ds_crit-min_ds(b))./ds_crit);

                        %Define particle velocity at AR node
                        v_AR(b)=v_o*(1-H_slp*(((xz_surf(I_min_ds(b),1)-x_off)*sin(alfa)-xz_surf(I_min_ds(b),2)*cos(alfa)).^2+(y_mean).^2).^0.5./...
                           ((xz_surf(I_min_ds(b),1)-x_off)*cos(alfa)+xz_surf(I_min_ds(b),2)*sin(alfa)));
                        if (v_AR(b)<0)
                        v_AR(b)=0;
                        end

                        %Define particle mass flux at AR node
                        flux_AR(b)=(MFR/pi)*(beta./((xz_surf(I_min_ds(b),1)-x_off)*cos(alfa)+xz_surf(I_min_ds(b),2)*sin(alfa))).^2....
                        *exp(-(beta^2.*(((xz_surf(I_min_ds(b),1)-x_off)*sin(alfa)-xz_surf(I_min_ds(b),2)*cos(alfa)).^2+(y_mean).^2)./...
                        ((xz_surf(I_min_ds(b),1)-x_off)*cos(alfa)+xz_surf(I_min_ds(b),2)*sin(alfa)).^2));

                        %theta_D
                        theta_D(b)=acos(((xz_surf(b,1)-xz_surf(I_min_ds(b),1)).*(phi_x_star_AR(b))+(xz_surf(b,2)-xz_surf(I_min_ds(b),2)).*(phi_z_star_AR(b)))./...
                           (sqrt((xz_surf(b,1)-xz_surf(I_min_ds(b),1)).^2+(xz_surf(b,2)-
xz_surf(I_min_ds(b),2)).^2).*sqrt((phi_x_star_AR(b)).^2+(phi_z_star_AR(b)).^2)));
                        theta_D_deg(b)=theta_D(b)*(180/pi);

                        %Calculate +/-,c,* cos_theta_D for F_ext Algorithm (Note, if 2nd strike not called
                        %up, cos_thetas will remain 0, so F_2nd=0 in Fext
                    cos_t_pfx_pfz_D(b)=((xz_surf(b,1)-xz_surf(I_min_ds(b),1)).*(phi_x_pos_AR(b))+(xz_surf(b,2)-
xz_surf(I_min_ds(b),2)).*(phi_z_pos_AR(b)))./...
                      (sqrt((xz_surf(b,1)-xz_surf(I_min_ds(b),1)).^2+(xz_surf(b,2)-
xz_surf(I_min_ds(b),2)).^2).*sqrt((phi_x_pos_AR(b)).^2+(phi_z_pos_AR(b)).^2));

                    cos_t_pfx_nfz_D(b)=((xz_surf(b,1)-xz_surf(I_min_ds(b),1)).*(phi_x_pos_AR(b))+(xz_surf(b,2)-
xz_surf(I_min_ds(b),2)).*(phi_z_neg_AR(b)))./...
                      (sqrt((xz_surf(b,1)-xz_surf(I_min_ds(b),1)).^2+(xz_surf(b,2)-
xz_surf(I_min_ds(b),2)).^2).*sqrt((phi_x_pos_AR(b)).^2+(phi_z_neg_AR(b)).^2));

                    cos_t_nfx_pfz_D(b)=((xz_surf(b,1)-xz_surf(I_min_ds(b),1)).*(phi_x_neg_AR(b))+(xz_surf(b,2)-
xz_surf(I_min_ds(b),2)).*(phi_z_pos_AR(b)))./...
                      (sqrt((xz_surf(b,1)-xz_surf(I_min_ds(b),1)).^2+(xz_surf(b,2)-
xz_surf(I_min_ds(b),2)).^2).*sqrt((phi_x_neg_AR(b)).^2+(phi_z_pos_AR(b)).^2));

                    cos_t_nfx_nfz_D(b)=((xz_surf(b,1)-xz_surf(I_min_ds(b),1)).*(phi_x_neg_AR(b))+(xz_surf(b,2)-
xz_surf(I_min_ds(b),2)).*(phi_z_neg_AR(b)))./...
                      (sqrt((xz_surf(b,1)-xz_surf(I_min_ds(b),1)).^2+(xz_surf(b,2)-
xz_surf(I_min_ds(b),2)).^2).*sqrt((phi_x_neg_AR(b)).^2+(phi_z_neg_AR(b)).^2));

                    cos_t_star_D(b)=cos(theta_D(b)); %Calculated above already

                    cos_t_cen_D(b)=((xz_surf(b,1)-xz_surf(I_min_ds(b),1)).*(phi_x_cen_AR(b))+(xz_surf(b,2)-xz_surf(I_min_ds(b),2)).*(phi_z_cen_AR(b)))./...
                      (sqrt((xz_surf(b,1)-xz_surf(I_min_ds(b),1)).^2+(xz_surf(b,2)-
xz_surf(I_min_ds(b),2)).^2).*sqrt((phi_x_cen_AR(b)).^2+(phi_z_cen_AR(b)).^2));


      else
        f_v_AR_fin(b)=0;
        v_AR(b)=0;
        flux_AR(b)=0;
        theta_D(b)=NaN;
        theta_D_deg(b)=NaN;
      end

   else
     f_v_AR_fin(b)=0;
     v_AR(b)=0;
     flux_AR(b)=0;
     theta_D(b)=NaN;
     theta_D_deg(b)=NaN;
   end

end
NaN_Chk_theta_D=isnan(theta_D); %If any entries are NaN, returns 1 for that entry, else 0

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%END of 2nd Strike Algorithm %%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
             %START OF SDF AND F_EXT ALGORITHM
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%RE-initialize SDF (update phi)

%Initialize SDF
SDF=zeros(b_max,1);
%Initialize F_ext's and corresponding cos(theta)'s
cos_t_pfx_pfz_ext_1=zeros(k_max,i_max);
cos_t_pfx_nfz_ext_1=zeros(k_max,i_max);
cos_t_nfx_pfz_ext_1=zeros(k_max,i_max);
cos_t_nfx_nfz_ext_1=zeros(k_max,i_max);
cos_t_star_ext_1=zeros(k_max,i_max);
cos_t_cen_ext_1=zeros(k_max,i_max);
F_ext_pfx_pfz_1=zeros(k_max,i_max);
F_ext_pfx_nfz_1=zeros(k_max,i_max);
F_ext_nfx_pfz_1=zeros(k_max,i_max);
F_ext_nfx_nfz_1=zeros(k_max,i_max);
F_ext_star_1=zeros(k_max,i_max);
F_ext_cen_1=zeros(k_max,i_max);
%Initialize dphi/dx,dphi/dz (FD's)
phi_x_pos_ext=zeros(k_max,i_max);
phi_x_neg_ext=zeros(k_max,i_max);
phi_x_cen_ext=zeros(k_max,i_max);
phi_z_pos_ext=zeros(k_max,i_max);
phi_z_neg_ext=zeros(k_max,i_max);
phi_z_cen_ext=zeros(k_max,i_max);
phi_x_star_ext=zeros(k_max,i_max);
phi_z_star_ext=zeros(k_max,i_max);
```

```
%Initialize Erosive Power and Masking Function Properties - Extended
x_prime_ext=zeros(k_max,i_max);
z_prime_ext=zeros(k_max,i_max);
L_mask_ext=zeros(k_max,i_max);
x_lim_ext=zeros(k_max,i_max);
M_r_x_prime_ext=zeros(k_max,i_max);
Eros_pow_ext=zeros(k_max,i_max);
v_ext=zeros(k_max,i_max);
flux_ext=zeros(k_max,i_max);
Int_P_r_x_prime_ext=zeros(k_max,i_max);
Int_P_r_L_mask_ext=zeros(k_max,i_max);
J_k_ext=0; %Initialize
J_i_ext=0;
I_min=zeros(k_max,i_max); %Index of SDF

%2nd strike erosive power
Eros_pow_ext_2nd=zeros(k_max,i_max);
%Initial strike F_ext's
F_ext_pfx_pfz_1_1st=zeros(k_max,i_max);
F_ext_pfx_nfz_1_1st=zeros(k_max,i_max);
F_ext_nfx_pfz_1_1st=zeros(k_max,i_max);
F_ext_nfx_nfz_1_1st=zeros(k_max,i_max);
F_ext_star_1_1st=zeros(k_max,i_max);
F_ext_cen_1_1st=zeros(k_max,i_max);

for k=1:1:k_max
     for i=1:1:i_max

         if ((flag_NB(k,i)==1)||(flag_NB(k,i)==2)) %#####*****#####$$$$$#####*****#####

          for b=1:1:b_max
              if (phi_1(k,i)==0)
                  SDF(b)=0; %we are on the surface
              elseif
((xz_surf(b,1)~=0)&&(xz_surf(b,2)~=0))||((alfa==(90*pi/180))&&((b==1)||(b==(i_max+1))||(b==(2*i_max+1)))&&(xz_surf(b,1)==0)&&(xz_surf(b,2)~=0))
                  SDF(b)=((x_cord(i)-xz_surf(b,1)).^2+(z_cord(k)-xz_surf(b,2)).^2).^0.5;
              else
                  SDF(b)=NaN; %Need this since it accounts for cases where surface wasn't encountered
              end            %and the fact that SDF can be actually 0

          end
          %Obtain value and index at which SDF is MIN (ignores NaN's)
          [min_SDF,I_min(k,i)]=min(SDF);

          %Update phi
          if (phi_1(k,i)>0)
                  phi(k,i)=min_SDF;
          elseif (phi_1(k,i)<0)
                  phi(k,i)=-min_SDF;
          else
                  phi(k,i)=phi_1(k,i); %i.e.,phi(k,i)=0; i.e., we are on the surface
          end


          %----------------------------------------------------------------
          %%%%%%%%%%%%%%%%%%%%F_ext Algorithm%%%%%%%%%%%%%%%%%%%%%%%%%%%%
          %----------------------------------------------------------------

          if (I_min(k,i)>=1)&&(I_min(k,i)<=(3*i_max))&&(phi_1(k,i)~=0) %zsurf used

            if (I_min(k,i)>=1)&&(I_min(k,i)<=i_max)
               I_min_ik=I_min(k,i); %z_surf_1 used
            end
            if (I_min(k,i)>=(i_max+1))&&(I_min(k,i)<=(2*i_max))
               I_min_ik=I_min(k,i)-i_max;  %z_surf_2 used
            end

            if (I_min(k,i)>=(2*i_max+1))&&(I_min(k,i)<=(3*i_max))
               I_min_ik=I_min(k,i)-2*i_max;  %z_surf_3 used
            end

                %Calculate nearest k index to surface
                J_k_ext=floor(k_max-(xz_surf(I_min(k,i),2)-z_min_grid)/dz);
%____________________________________________________________________________________
                %Calculation of dphi/dx,dphi/dz; Define BC's

                if I_min_ik==i_max
                    phi_x_pos_ext(k,i)=0;
                else
          phi_B_z_surf=((xz_surf(I_min(k,i),2)-z_cord(J_k_ext))/(z_cord(J_k_ext+1)-z_cord(J_k_ext))).*(phi_1(J_k_ext+1,I_min_ik+1)-phi_1(J_k_ext,I_min_ik+1))+phi_1(J_k_ext,I_min_ik+1);
          phi_x_pos_ext(k,i)=(phi_B_z_surf-0)./dx;
        end

        if I_min_ik==1
          phi_x_neg_ext(k,i)=0;
        else
          phi_A_z_surf=((xz_surf(I_min(k,i),2)-z_cord(J_k_ext))/(z_cord(J_k_ext+1)-z_cord(J_k_ext))).*(phi_1(J_k_ext+1,I_min_ik-1)-phi_1(J_k_ext,I_min_ik-1))+phi_1(J_k_ext,I_min_ik-1);
          phi_x_neg_ext(k,i)=(0-phi_A_z_surf)./dx;
        end

        if (I_min_ik==i_max)||(I_min_ik==1)
          phi_x_cen_ext(k,i)=0;
        else
          phi_A_z_surf=((xz_surf(I_min(k,i),2)-z_cord(J_k_ext))/(z_cord(J_k_ext+1)-z_cord(J_k_ext))).*(phi_1(J_k_ext+1,I_min_ik-1)-phi_1(J_k_ext,I_min_ik-1))+phi_1(J_k_ext,I_min_ik-1);
          phi_B_z_surf=((xz_surf(I_min(k,i),2)-z_cord(J_k_ext))/(z_cord(J_k_ext+1)-z_cord(J_k_ext))).*(phi_1(J_k_ext+1,I_min_ik+1)-phi_1(J_k_ext,I_min_ik+1))+phi_1(J_k_ext,I_min_ik+1);
          phi_x_cen_ext(k,i)=(phi_B_z_surf-phi_A_z_surf)./(2*dx);
        end

        if J_k_ext==1
          phi_z_pos_ext(k,i)=0;
        else
          phi_C_z_surf=((xz_surf(I_min(k,i),2)+dz-z_cord(J_k_ext-1))/(z_cord(J_k_ext)-z_cord(J_k_ext-1))).*(phi_1(J_k_ext,I_min_ik)-phi_1(J_k_ext-1,I_min_ik))+phi_1(J_k_ext-1,I_min_ik);
          phi_z_pos_ext(k,i)=(phi_C_z_surf-0)./dz;
        end

        if J_k_ext==k_max
          phi_z_neg_ext(k,i)=0;
        else
          phi_D_z_surf=((xz_surf(I_min(k,i),2)-dz-z_cord(J_k_ext+1))/(z_cord(J_k_ext+2)-z_cord(J_k_ext+1))).*(phi_1(J_k_ext+2,I_min_ik)-phi_1(J_k_ext+1,I_min_ik))+phi_1(J_k_ext+1,I_min_ik);
          phi_z_neg_ext(k,i)=(0-phi_D_z_surf)./dz;
        end

        if (J_k_ext==k_max)||(J_k_ext==1)
          phi_z_cen_ext(k,i)=0;
        else
          phi_C_z_surf=((xz_surf(I_min(k,i),2)+dz-z_cord(J_k_ext-1))/(z_cord(J_k_ext)-z_cord(J_k_ext-1))).*(phi_1(J_k_ext,I_min_ik)-phi_1(J_k_ext-1,I_min_ik))+phi_1(J_k_ext-1,I_min_ik);
          phi_D_z_surf=((xz_surf(I_min(k,i),2)-dz-z_cord(J_k_ext+1))/(z_cord(J_k_ext+2)-z_cord(J_k_ext+1))).*(phi_1(J_k_ext+2,I_min_ik)-phi_1(J_k_ext+1,I_min_ik))+phi_1(J_k_ext+1,I_min_ik);
          phi_z_cen_ext(k,i)=(phi_C_z_surf-phi_D_z_surf)./(2*dz);
        end

        %Calculate phi_stars
        phi_x_star_ext(k,i)=(phi_x_pos_ext(k,i)+phi_x_neg_ext(k,i))/2;
        phi_z_star_ext(k,i)=(phi_z_pos_ext(k,i)+phi_z_neg_ext(k,i))/2;
%________________________________________________________________________________
        %Masking function for T, M
        x_prime_ext(k,i)=(xz_surf(I_min(k,i),1)-x_off).*sin(alfa)-xz_surf(I_min(k,i),2).*cos(alfa); %Rotated local x
        z_prime_ext(k,i)=(xz_surf(I_min(k,i),1)-x_off).*cos(alfa)+xz_surf(I_min(k,i),2).*sin(alfa); %Rotated local z

            if (xz_surf(I_min(k,i),2)>=(h*sin(alfa))) %T

            if (xz_surf(I_min(k,i),2)>=(h*sin(alfa)+dz*MT_pt_dist))

            if (max_x_prime_surf_LM<0) %Case (a) and (c)
              if (x_prime_ext(k,i)<0)
                L_mask_ext(k,i)=z_prime_ext(k,i).*tan_fi_min;
              else %i.e., when x_prime>=0
                L_mask_ext(k,i)=z_prime_ext(k,i).*tan_fi_max;
              end
            else %i.e., when x_m>=W_m/2 Case (b)
              x_lim_ext(k,i)=z_prime_ext(k,i).*tan_fi_min;
              if (x_prime_ext(k,i)<x_lim_ext(k,i))
                L_mask_ext(k,i)=0;
              else %i.e., when x_prime>=x_lim
                L_mask_ext(k,i)=z_prime_ext(k,i).*tan_fi_max-x_lim_ext(k,i);
              end
            end

            %Define proportion of mass of particle that pass through mask opening having a
            %specific particle size (of radius r) distribution
            if (abs(x_prime_ext(k,i))>=L_mask_ext(k,i))
              M_r_x_prime_ext(k,i)=0;
```

```
                else
                   Int_P_r_x_prime_ext(k,i)=real(P_1-P_1*erf(P_2*log(L_mask_ext(k,i)-abs(x_prime_ext(k,i)))-P_3));
                   Int_P_r_L_mask_ext(k,i)=real(P_1-P_1*erf(P_2*log(L_mask_ext(k,i))-P_3));
                   M_r_x_prime_ext(k,i)=Int_P_r_x_prime_ext(k,i)./Int_P_r_L_mask_ext(k,i);
                end

                else %T w/in zTadj
                   if ((time*v_scan/(2*r_s))<=No_p_MrxM_ON)
                          M_r_x_prime_ext(k,i)=1;
                   else

                   if ((xz_surf(I_min(k,i),1)>=x_min_grid)&&(xz_surf(I_min(k,i),1)<=(x_min_grid+leng_M_L)))%TL
                      if (xz_surf(I_min(k,i),1)<(x_lim_LM-dx))
                         M_r_x_prime_ext(k,i)=1;
                      else
                        M_r_x_prime_ext(k,i)=M_r_x_prime_LM;
                      end
                   elseif ((xz_surf(I_min(k,i),1)>=(x_max_grid-leng_M_R))&&(xz_surf(I_min(k,i),1)<=x_max_grid)) %TR
                      if (xz_surf(I_min(k,i),1)>(x_lim_RM+dx))
                         M_r_x_prime_ext(k,i)=1;
                      else
                         M_r_x_prime_ext(k,i)=M_r_x_prime_RM;
                      end

                   else
                      M_r_x_prime_ext(k,i)=1;
                   end

                   end
                end

                else  %M
                   if ((time*v_scan/(2*r_s))<=No_p_MrxM_ON)
                          M_r_x_prime_ext(k,i)=1;
                   else

                   if ((xz_surf(I_min(k,i),1)>=x_min_grid)&&(xz_surf(I_min(k,i),1)<=(x_min_grid+leng_M_L)))%ML
                      if (xz_surf(I_min(k,i),1)<(x_lim_LM-dx))
                         M_r_x_prime_ext(k,i)=1;
                      else
                        M_r_x_prime_ext(k,i)=M_r_x_prime_LM;
                      end
                   elseif ((xz_surf(I_min(k,i),1)>=(x_max_grid-leng_M_R))&&(xz_surf(I_min(k,i),1)<=x_max_grid)) %MR
                      if (xz_surf(I_min(k,i),1)>(x_lim_RM+dx))
                         M_r_x_prime_ext(k,i)=1;
                      else
                         M_r_x_prime_ext(k,i)=M_r_x_prime_RM;
                      end

                   else
                      M_r_x_prime_ext(k,i)=1;
                   end

                   end

                end

%________________________________________________________________________________
                %Define velocity v(x,z) at each grid node
                   v_ext(k,i)=v_o*(1-H_slp*(((xz_surf(I_min(k,i),1)-x_off)*sin(alfa)-xz_surf(I_min(k,i),2)*cos(alfa)).^2+(y_mean).^2).^0.5./((xz_surf(I_min(k,i),1)-
x_off)*cos(alfa)+xz_surf(I_min(k,i),2)*sin(alfa)));
                if (v_ext(k,i)<0)
                   v_ext(k,i)=0;
                end

                %Define particle mass flux(x,z) at each grid node
                   flux_ext(k,i)=(MFR/pi)*(beta./((xz_surf(I_min(k,i),1)-x_off)*cos(alfa)+xz_surf(I_min(k,i),2)*sin(alfa))).^2....
                   *exp(-(beta^2.*(((xz_surf(I_min(k,i),1)-x_off)*sin(alfa)-xz_surf(I_min(k,i),2)*cos(alfa)).^2+(y_mean).^2)./((xz_surf(I_min(k,i),1)-
x_off)*cos(alfa)+xz_surf(I_min(k,i),2)*sin(alfa)).^2));

                %Define Erosive Power Eros_pow(k,i) at each grid
                %node (1st Strike)
                if (xz_surf(I_min(k,i),2)>=(h*sin(alfa))) %T
                   Eros_pow_ext(k,i)=M_r_x_prime_ext(k,i).*v_ext(k,i).^k_vel.*flux_ext(k,i);
                else %M
                   Eros_pow_ext(k,i)=M_r_x_prime_ext(k,i).*v_ext(k,i).^k_vel_M.*flux_ext(k,i);
                end

                %Define Erosive Power for 2nd strike
                %NOTE: No Mask here
```

```
            if (xz_surf(I_min(k,i),2)>=(h*sin(alfa))) %T
              if (NaN_Chk_theta_D(I_min(k,i))==0)
                Eros_pow_ext_2nd(k,i)=(f_v_AR_fin(I_min(k,i)).*v_AR(I_min(k,i))).^k_vel.*flux_AR(I_min(k,i));
              else
                Eros_pow_ext_2nd(k,i)=0;
              end
            else %M
              if (NaN_Chk_theta_D(I_min(k,i))==0)
                Eros_pow_ext_2nd(k,i)=(f_v_AR_fin(I_min(k,i)).*v_AR(I_min(k,i))).^k_vel_M.*flux_AR(I_min(k,i));
              else
                Eros_pow_ext_2nd(k,i)=0;
              end
            end

%___________________________________________________________________________
        %Calculate F_extensions

          cos_t_pfx_pfz_ext_1(k,i)=((xz_surf(I_min(k,i),1)-x_off).*(phi_x_pos_ext(k,i))+xz_surf(I_min(k,i),2).*(phi_z_pos_ext(k,i)))./sqrt((xz_surf(I_min(k,i),1)-
x_off).^2+xz_surf(I_min(k,i),2).^2);
          cos_t_pfx_nfz_ext_1(k,i)=((xz_surf(I_min(k,i),1)-x_off).*(phi_x_pos_ext(k,i))+xz_surf(I_min(k,i),2).*(phi_z_neg_ext(k,i)))./sqrt((xz_surf(I_min(k,i),1)-
x_off).^2+xz_surf(I_min(k,i),2).^2);
          cos_t_nfx_pfz_ext_1(k,i)=((xz_surf(I_min(k,i),1)-x_off).*(phi_x_neg_ext(k,i))+xz_surf(I_min(k,i),2).*(phi_z_pos_ext(k,i)))./sqrt((xz_surf(I_min(k,i),1)-
x_off).^2+xz_surf(I_min(k,i),2).^2);
          cos_t_nfx_nfz_ext_1(k,i)=((xz_surf(I_min(k,i),1)-x_off).*(phi_x_neg_ext(k,i))+xz_surf(I_min(k,i),2).*(phi_z_neg_ext(k,i)))./sqrt((xz_surf(I_min(k,i),1)-
x_off).^2+xz_surf(I_min(k,i),2).^2);
          cos_t_star_ext_1(k,i)=((xz_surf(I_min(k,i),1)-x_off).*(phi_x_star_ext(k,i))+xz_surf(I_min(k,i),2).*(phi_z_star_ext(k,i)))./sqrt((xz_surf(I_min(k,i),1)-
x_off).^2+xz_surf(I_min(k,i),2).^2);
          cos_t_cen_ext_1(k,i)=((xz_surf(I_min(k,i),1)-x_off).*(phi_x_cen_ext(k,i))+xz_surf(I_min(k,i),2).*(phi_z_cen_ext(k,i)))./sqrt((xz_surf(I_min(k,i),1)-
x_off).^2+xz_surf(I_min(k,i),2).^2);
          if (cos_t_pfx_pfz_ext_1(k,i)>1)
            cos_t_pfx_pfz_ext_1(k,i)=1;
          end
          if (cos_t_pfx_nfz_ext_1(k,i)>1)
            cos_t_pfx_nfz_ext_1(k,i)=1;
          end
          if (cos_t_nfx_pfz_ext_1(k,i)>1)
            cos_t_nfx_pfz_ext_1(k,i)=1;
          end
          if (cos_t_nfx_nfz_ext_1(k,i)>1)
            cos_t_nfx_nfz_ext_1(k,i)=1;
          end
          if (cos_t_star_ext_1(k,i)>1)
            cos_t_star_ext_1(k,i)=1;
          end
          if (cos_t_cen_ext_1(k,i)>1)
            cos_t_cen_ext_1(k,i)=1;
          end

          if (xz_surf(I_min(k,i),2)>=(h*sin(alfa))) %T
            if (cos_t_pfx_pfz_ext_1(k,i)<=0)
           F_ext_pfx_pfz_1_1st(k,i)=0;
           else
           F_ext_pfx_pfz_1_1st(k,i)=real((C/rho_s)*Eros_pow_ext(k,i).*((cos_t_pfx_pfz_ext_1(k,i))^(k_vel+1)));
            end
          else %M
           if (cos_t_pfx_pfz_ext_1(k,i)<=0) %Apply mask visibility for M
           F_ext_pfx_pfz_1_1st(k,i)=0;
           else
           F_ext_pfx_pfz_1_1st(k,i)=real((C_M/rho_s_M).*Eros_pow_ext(k,i).*((cos_t_pfx_pfz_ext_1(k,i)).^n_1_M).*((1+H_vic_M*(1-
cos_t_pfx_pfz_ext_1(k,i))).^n_2_M));
           end
          end

          %Note: cos_theta_D = 0 if no 2nd strk.
          if (xz_surf(I_min(k,i),2)>=(h*sin(alfa))) %T
           F_ext_pfx_pfz_1(k,i)=F_ext_pfx_pfz_1_1st(k,i)+real((C/rho_s)*Eros_pow_ext_2nd(k,i).*((cos_t_pfx_pfz_D(I_min(k,i)))^(k_vel+1)));
          else %M

F_ext_pfx_pfz_1(k,i)=F_ext_pfx_pfz_1_1st(k,i)+real((C_M/rho_s_M).*Eros_pow_ext_2nd(k,i).*((cos_t_pfx_pfz_D(I_min(k,i))).^n_1_M).*((1+H_vic_M*(1-
cos_t_pfx_pfz_D(I_min(k,i)))).^n_2_M));
          end

          if (xz_surf(I_min(k,i),2)>=(h*sin(alfa))) %T
            if (cos_t_pfx_nfz_ext_1(k,i)<=0)
           F_ext_pfx_nfz_1_1st(k,i)=0;
           else
           F_ext_pfx_nfz_1_1st(k,i)=real((C/rho_s)*Eros_pow_ext(k,i).*((cos_t_pfx_nfz_ext_1(k,i))^(k_vel+1)));
            end
          else %M
           if (cos_t_pfx_nfz_ext_1(k,i)<=0) %Apply mask visibility for M
```

```
                  F_ext_pfx_nfz_1_1st(k,i)=0;
                  else
                  F_ext_pfx_nfz_1_1st(k,i)=real((C_M/rho_s_M).*Eros_pow_ext(k,i).*((cos_t_pfx_nfz_ext_1(k,i)).^n_1_M).*((1+H_vic_M*(1-
cos_t_pfx_nfz_ext_1(k,i))).^n_2_M));
                   end
                end

                %Note: cos_theta_D = 0 if no 2nd strk.)
                if (xz_surf(I_min(k,i),2)>=(h*sin(alfa))) %T
                  F_ext_pfx_nfz_1(k,i)=F_ext_pfx_nfz_1_1st(k,i)+real((C/rho_s)*Eros_pow_ext_2nd(k,i).*((cos_t_pfx_nfz_D(I_min(k,i)))^(k_vel+1)));
                else %M

F_ext_pfx_nfz_1(k,i)=F_ext_pfx_nfz_1_1st(k,i)+real((C_M/rho_s_M).*Eros_pow_ext_2nd(k,i).*((cos_t_pfx_nfz_D(I_min(k,i))).^n_1_M).*((1+H_vic_M*(1-
cos_t_pfx_nfz_D(I_min(k,i)))).^n_2_M));
                end

                if (xz_surf(I_min(k,i),2)>=(h*sin(alfa))) %T
                   if (cos_t_nfx_pfz_ext_1(k,i)<=0)
                  F_ext_nfx_pfz_1_1st(k,i)=0;
                  else
                  F_ext_nfx_pfz_1_1st(k,i)=real((C/rho_s)*Eros_pow_ext(k,i).*((cos_t_nfx_pfz_ext_1(k,i))^(k_vel+1)));
                   end
                else %M
                  if (cos_t_nfx_pfz_ext_1(k,i)<=0) %Apply mask visibility for M
                  F_ext_nfx_pfz_1_1st(k,i)=0;
                  else
                  F_ext_nfx_pfz_1_1st(k,i)=real((C_M/rho_s_M).*Eros_pow_ext(k,i).*((cos_t_nfx_pfz_ext_1(k,i)).^n_1_M).*((1+H_vic_M*(1-
cos_t_nfx_pfz_ext_1(k,i))).^n_2_M));
                   end
                end

                %Note: cos_theta_D = 0 if no 2nd strk.
                if (xz_surf(I_min(k,i),2)>=(h*sin(alfa))) %T
                  F_ext_nfx_pfz_1(k,i)=F_ext_nfx_pfz_1_1st(k,i)+real((C/rho_s)*Eros_pow_ext_2nd(k,i).*((cos_t_nfx_pfz_D(I_min(k,i)))^(k_vel+1)));
                else %M

F_ext_nfx_pfz_1(k,i)=F_ext_nfx_pfz_1_1st(k,i)+real((C_M/rho_s_M).*Eros_pow_ext_2nd(k,i).*((cos_t_nfx_pfz_D(I_min(k,i))).^n_1_M).*((1+H_vic_M*(1-
cos_t_nfx_pfz_D(I_min(k,i)))).^n_2_M));
                end

                if (xz_surf(I_min(k,i),2)>=(h*sin(alfa))) %T
                   if (cos_t_nfx_nfz_ext_1(k,i)<=0)
                  F_ext_nfx_nfz_1_1st(k,i)=0;
                  else
                  F_ext_nfx_nfz_1_1st(k,i)=real((C/rho_s)*Eros_pow_ext(k,i).*((cos_t_nfx_nfz_ext_1(k,i))^(k_vel+1)));
                   end
                else %M
                  if (cos_t_nfx_nfz_ext_1(k,i)<=0) %Apply mask visibility for M
                  F_ext_nfx_nfz_1_1st(k,i)=0;
                  else
                  F_ext_nfx_nfz_1_1st(k,i)=real((C_M/rho_s_M).*Eros_pow_ext(k,i).*((cos_t_nfx_nfz_ext_1(k,i)).^n_1_M).*((1+H_vic_M*(1-
cos_t_nfx_nfz_ext_1(k,i))).^n_2_M));
                   end
                end

                %Note: cos_theta_D = 0 if no 2nd strk.
                if (xz_surf(I_min(k,i),2)>=(h*sin(alfa))) %T
                  F_ext_nfx_nfz_1(k,i)=F_ext_nfx_nfz_1_1st(k,i)+real((C/rho_s)*Eros_pow_ext_2nd(k,i).*((cos_t_nfx_nfz_D(I_min(k,i)))^(k_vel+1)));
                else %M

F_ext_nfx_nfz_1(k,i)=F_ext_nfx_nfz_1_1st(k,i)+real((C_M/rho_s_M).*Eros_pow_ext_2nd(k,i).*((cos_t_nfx_nfz_D(I_min(k,i))).^n_1_M).*((1+H_vic_M*(1-
cos_t_nfx_nfz_D(I_min(k,i)))).^n_2_M));
                end

                if (xz_surf(I_min(k,i),2)>=(h*sin(alfa))) %T
                   if (cos_t_star_ext_1(k,i)<=0)
                  F_ext_star_1_1st(k,i)=0;
                  else
                  F_ext_star_1_1st(k,i)=real((C/rho_s)*Eros_pow_ext(k,i).*((cos_t_star_ext_1(k,i))^(k_vel+1)));
                   end
                else %M
                  if (cos_t_star_ext_1(k,i)<=0) %Apply mask visibility for M
                  F_ext_star_1_1st(k,i)=0;
                  else
                  F_ext_star_1_1st(k,i)=real((C_M/rho_s_M).*Eros_pow_ext(k,i).*((cos_t_star_ext_1(k,i)).^n_1_M).*((1+H_vic_M*(1-
cos_t_star_ext_1(k,i))).^n_2_M));
                   end
                end

                %Note: cos_theta_D = 0 if no 2nd strk.
```

```
            if (xz_surf(I_min(k,i),2)>=(h*sin(alfa))) %T
              F_ext_star_1(k,i)=F_ext_star_1_1st(k,i)+real((C/rho_s)*Eros_pow_ext_2nd(k,i).*((cos_t_star_D(I_min(k,i)))^(k_vel+1)));
            else %M
              F_ext_star_1(k,i)=F_ext_star_1_1st(k,i)+real((C_M/rho_s_M).*Eros_pow_ext_2nd(k,i).*((cos_t_star_D(I_min(k,i))).^n_1_M).*((1+H_vic_M*(1-cos_t_star_D(I_min(k,i)))).^n_2_M));
            end

            if (xz_surf(I_min(k,i),2)>=(h*sin(alfa))) %T
               if (cos_t_cen_ext_1(k,i)<=0)
              F_ext_cen_1_1st(k,i)=0;
              else
              F_ext_cen_1_1st(k,i)=real((C/rho_s)*Eros_pow_ext(k,i).*((cos_t_cen_ext_1(k,i))^(k_vel+1)));
               end
            else %M
              if (cos_t_cen_ext_1(k,i)<=0) %Apply mask visibility for M
              F_ext_cen_1_1st(k,i)=0;
              else
              F_ext_cen_1_1st(k,i)=real((C_M/rho_s_M).*Eros_pow_ext(k,i).*((cos_t_cen_ext_1(k,i)).^n_1_M).*((1+H_vic_M*(1-cos_t_cen_ext_1(k,i))).^n_2_M));
              end
            end

              %Note: cos_theta_D = 0 if no 2nd strk.
            if (xz_surf(I_min(k,i),2)>=(h*sin(alfa))) %T
              F_ext_cen_1(k,i)=F_ext_cen_1_1st(k,i)+real((C/rho_s)*Eros_pow_ext_2nd(k,i).*((cos_t_cen_D(I_min(k,i)))^(k_vel+1)));
            else %M
              F_ext_cen_1(k,i)=F_ext_cen_1_1st(k,i)+real((C_M/rho_s_M).*Eros_pow_ext_2nd(k,i).*((cos_t_cen_D(I_min(k,i))).^n_1_M).*((1+H_vic_M*(1-cos_t_cen_D(I_min(k,i)))).^n_2_M));
            end


      elseif (I_min(k,i)>=(3*i_max+1))&&(I_min(k,i)<=b_max)&&(phi_1(k,i)~=0) %xsurf used

        if (I_min(k,i)>=(3*i_max+1))&&(I_min(k,i)<=(3*i_max+k_max))
          I_min_ik=I_min(k,i)-3*i_max; %x_surf_1 used
        end
        if (I_min(k,i)>=(3*i_max+k_max+1))&&(I_min(k,i)<=b_max)
          I_min_ik=I_min(k,i)-3*i_max-k_max;  %x_surf_2 used
        end

          %Repeat above algorithm but for xsurf
          %Calculate nearest i index to surface
          J_i_ext=floor(1+(xz_surf(I_min(k,i),1)-x_min_grid)/dx);
%________________________________________________________________________
          %Calculation of dphi/dx,dphi/dz; Define BC's
          if J_i_ext==i_max
            phi_x_pos_ext(k,i)=0;
          else
            phi_D_x_surf=((xz_surf(I_min(k,i),1)+dx-x_cord(J_i_ext+1))/(x_cord(J_i_ext+2)-x_cord(J_i_ext+1))).*(phi_1(I_min_ik,J_i_ext+2)-phi_1(I_min_ik,J_i_ext+1))+phi_1(I_min_ik,J_i_ext+1);
            phi_x_pos_ext(k,i)=(phi_D_x_surf-0)./dx;
          end

          if J_i_ext==1
            phi_x_neg_ext(k,i)=0;
          else
            phi_C_x_surf=((xz_surf(I_min(k,i),1)-dx-x_cord(J_i_ext-1))/(x_cord(J_i_ext)-x_cord(J_i_ext-1))).*(phi_1(I_min_ik,J_i_ext)-phi_1(I_min_ik,J_i_ext-1))+phi_1(I_min_ik,J_i_ext-1);
            phi_x_neg_ext(k,i)=(0-phi_C_x_surf)./dx;
          end

          if (J_i_ext==i_max)||(J_i_ext==1)
            phi_x_cen_ext(k,i)=0;
          else
            phi_C_x_surf=((xz_surf(I_min(k,i),1)-dx-x_cord(J_i_ext-1))/(x_cord(J_i_ext)-x_cord(J_i_ext-1))).*(phi_1(I_min_ik,J_i_ext)-phi_1(I_min_ik,J_i_ext-1))+phi_1(I_min_ik,J_i_ext-1);
            phi_D_x_surf=((xz_surf(I_min(k,i),1)+dx-x_cord(J_i_ext+1))/(x_cord(J_i_ext+2)-x_cord(J_i_ext+1))).*(phi_1(I_min_ik,J_i_ext+2)-phi_1(I_min_ik,J_i_ext+1))+phi_1(I_min_ik,J_i_ext+1);
            phi_x_cen_ext(k,i)=(phi_D_x_surf-phi_C_x_surf)./(2*dx);
          end

          if I_min_ik==1
            phi_z_pos_ext(k,i)=0;
          else
            phi_A_x_surf=((xz_surf(I_min(k,i),1)-x_cord(J_i_ext))/(x_cord(J_i_ext+1)-x_cord(J_i_ext))).*(phi_1(I_min_ik-1,J_i_ext+1)-phi_1(I_min_ik-1,J_i_ext))+phi_1(I_min_ik-1,J_i_ext);
            phi_z_pos_ext(k,i)=(phi_A_x_surf-0)./dz;
          end

          if I_min_ik==k_max
            phi_z_neg_ext(k,i)=0;
          else
            phi_B_x_surf=((xz_surf(I_min(k,i),1)-x_cord(J_i_ext))/(x_cord(J_i_ext+1)-x_cord(J_i_ext))).*(phi_1(I_min_ik+1,J_i_ext+1)-
phi_1(I_min_ik+1,J_i_ext))+phi_1(I_min_ik+1,J_i_ext);
            phi_z_neg_ext(k,i)=(0-phi_B_x_surf)./dz;
          end

          if (I_min_ik==k_max)||(I_min_ik==1)
            phi_z_cen_ext(k,i)=0;
          else
            phi_A_x_surf=((xz_surf(I_min(k,i),1)-x_cord(J_i_ext))/(x_cord(J_i_ext+1)-x_cord(J_i_ext))).*(phi_1(I_min_ik-1,J_i_ext+1)-phi_1(I_min_ik-
1,J_i_ext))+phi_1(I_min_ik-1,J_i_ext);
            phi_B_x_surf=((xz_surf(I_min(k,i),1)-x_cord(J_i_ext))/(x_cord(J_i_ext+1)-x_cord(J_i_ext))).*(phi_1(I_min_ik+1,J_i_ext+1)-
phi_1(I_min_ik+1,J_i_ext))+phi_1(I_min_ik+1,J_i_ext);
            phi_z_cen_ext(k,i)=(phi_A_x_surf-phi_B_x_surf)./(2*dz);
          end

          %Calculate phi_stars
          phi_x_star_ext(k,i)=(phi_x_pos_ext(k,i)+phi_x_neg_ext(k,i))/2;
          phi_z_star_ext(k,i)=(phi_z_pos_ext(k,i)+phi_z_neg_ext(k,i))/2;
%______________________________________________________________________________
          %Masking function for T, M
          x_prime_ext(k,i)=(xz_surf(I_min(k,i),1)-x_off).*sin(alfa)-xz_surf(I_min(k,i),2).*cos(alfa); %Rotated local x
          z_prime_ext(k,i)=(xz_surf(I_min(k,i),1)-x_off).*cos(alfa)+xz_surf(I_min(k,i),2).*sin(alfa); %Rotated local z

              if (xz_surf(I_min(k,i),2)>=(h*sin(alfa))) %T

              if (xz_surf(I_min(k,i),2)>=(h*sin(alfa)+dz*MT_pt_dist))

              if (max_x_prime_surf_LM<0) %Case (a) and (c)
                if (x_prime_ext(k,i)<0)
                  L_mask_ext(k,i)=z_prime_ext(k,i).*tan_fi_min;
                else %i.e., when x_prime>=0
                  L_mask_ext(k,i)=z_prime_ext(k,i).*tan_fi_max;
                end
              else %i.e., when x_m>=W_m/2 Case (b)
                x_lim_ext(k,i)=z_prime_ext(k,i).*tan_fi_min;
                if (x_prime_ext(k,i)<x_lim_ext(k,i))
                  L_mask_ext(k,i)=0;
                else %i.e., when x_prime>=x_lim
                  L_mask_ext(k,i)=z_prime_ext(k,i).*tan_fi_max-x_lim_ext(k,i);
                end
              end

              %Define proportion of mass of particle that pass through mask opening having a
              %specific particle size (of radius r) distribution
              if (abs(x_prime_ext(k,i))>=L_mask_ext(k,i))
                M_r_x_prime_ext(k,i)=0;
              else
                Int_P_r_x_prime_ext(k,i)=real(P_1-P_1*erf(P_2*log(L_mask_ext(k,i)-abs(x_prime_ext(k,i)))-P_3));
                Int_P_r_L_mask_ext(k,i)=real(P_1-P_1*erf(P_2*log(L_mask_ext(k,i))-P_3));
                M_r_x_prime_ext(k,i)=Int_P_r_x_prime_ext(k,i)./Int_P_r_L_mask_ext(k,i);
              end

              else %T w/in zTadj
                if ((time*v_scan/(2*r_s))<=No_p_MrxM_ON)
                     M_r_x_prime_ext(k,i)=1;
                else

                if ((xz_surf(I_min(k,i),1)>=x_min_grid)&&(xz_surf(I_min(k,i),1)<=(x_min_grid+leng_M_L)))%TL
                  if (xz_surf(I_min(k,i),1)<(x_lim_LM-dx))
                    M_r_x_prime_ext(k,i)=1;
                  else
                   M_r_x_prime_ext(k,i)=M_r_x_prime_LM;
                  end
                elseif ((xz_surf(I_min(k,i),1)>=(x_max_grid-leng_M_R))&&(xz_surf(I_min(k,i),1)<=x_max_grid)) %TR
                  if (xz_surf(I_min(k,i),1)>(x_lim_RM+dx))
                    M_r_x_prime_ext(k,i)=1;
                  else
                    M_r_x_prime_ext(k,i)=M_r_x_prime_RM;
                  end

                else
                  M_r_x_prime_ext(k,i)=1;
                end

                end
              end

              else  %M
```

```
              if ((time*v_scan/(2*r_s))<=No_p_MrxM_ON)
                  M_r_x_prime_ext(k,i)=1;
              else

              if ((xz_surf(I_min(k,i),1)>=x_min_grid)&&(xz_surf(I_min(k,i),1)<=(x_min_grid+leng_M_L)))%ML
                if (xz_surf(I_min(k,i),1)<(x_lim_LM-dx))
                  M_r_x_prime_ext(k,i)=1;
                else
                 M_r_x_prime_ext(k,i)=M_r_x_prime_LM;
                end
              elseif ((xz_surf(I_min(k,i),1)>=(x_max_grid-leng_M_R))&&(xz_surf(I_min(k,i),1)<=x_max_grid)) %MR
                if (xz_surf(I_min(k,i),1)>(x_lim_RM+dx))
                  M_r_x_prime_ext(k,i)=1;
                else
                  M_r_x_prime_ext(k,i)=M_r_x_prime_RM;
                end

              else
                M_r_x_prime_ext(k,i)=1;
              end

              end

            end

%_________________________________________________________________________
            %Define velocity v(x,z) at each grid node
              v_ext(k,i)=v_o*(1-H_slp*(((xz_surf(I_min(k,i),1)-x_off)*sin(alfa)-xz_surf(I_min(k,i),2)*cos(alfa)).^2+(y_mean).^2).^0.5./((xz_surf(I_min(k,i),1)-
x_off)*cos(alfa)+xz_surf(I_min(k,i),2)*sin(alfa)));
            if (v_ext(k,i)<0)
              v_ext(k,i)=0;
            end

            %Define particle mass flux(x,z) at each grid node
              flux_ext(k,i)=(MFR/pi)*(beta./((xz_surf(I_min(k,i),1)-x_off)*cos(alfa)+xz_surf(I_min(k,i),2)*sin(alfa))).^2....
              *exp(-(beta^2.*(((xz_surf(I_min(k,i),1)-x_off)*sin(alfa)-xz_surf(I_min(k,i),2)*cos(alfa)).^2+(y_mean).^2)./((xz_surf(I_min(k,i),1)-
x_off)*cos(alfa)+xz_surf(I_min(k,i),2)*sin(alfa)).^2));

            %Define Erosive Power Eros_pow(k,i) at each grid
            %node (1st Strike)
            if (xz_surf(I_min(k,i),2)>=(h*sin(alfa))) %T
              Eros_pow_ext(k,i)=M_r_x_prime_ext(k,i).*v_ext(k,i).^k_vel.*flux_ext(k,i);
            else %M
              Eros_pow_ext(k,i)=M_r_x_prime_ext(k,i).*v_ext(k,i).^k_vel_M.*flux_ext(k,i);
            end

            %Define Erosive Power for 2nd strike
            %NOTE: No Mask here
            if (xz_surf(I_min(k,i),2)>=(h*sin(alfa))) %T
              if (NaN_Chk_theta_D(I_min(k,i))==0)
                Eros_pow_ext_2nd(k,i)=(f_v_AR_fin(I_min(k,i)).*v_AR(I_min(k,i))).^k_vel.*flux_AR(I_min(k,i));
              else
                Eros_pow_ext_2nd(k,i)=0;
              end
            else %M
              if (NaN_Chk_theta_D(I_min(k,i))==0)
                Eros_pow_ext_2nd(k,i)=(f_v_AR_fin(I_min(k,i)).*v_AR(I_min(k,i))).^k_vel_M.*flux_AR(I_min(k,i));
              else
                Eros_pow_ext_2nd(k,i)=0;
              end
            end

%_________________________________________________________________________
        %Calculate F_extensions

          cos_t_pfx_pfz_ext_1(k,i)=((xz_surf(I_min(k,i),1)-x_off).*(phi_x_pos_ext(k,i))+xz_surf(I_min(k,i),2).*(phi_z_pos_ext(k,i)))./sqrt((xz_surf(I_min(k,i),1)-
x_off).^2+xz_surf(I_min(k,i),2).^2);
          cos_t_pfx_nfz_ext_1(k,i)=((xz_surf(I_min(k,i),1)-x_off).*(phi_x_pos_ext(k,i))+xz_surf(I_min(k,i),2).*(phi_z_neg_ext(k,i)))./sqrt((xz_surf(I_min(k,i),1)-
x_off).^2+xz_surf(I_min(k,i),2).^2);
          cos_t_nfx_pfz_ext_1(k,i)=((xz_surf(I_min(k,i),1)-x_off).*(phi_x_neg_ext(k,i))+xz_surf(I_min(k,i),2).*(phi_z_pos_ext(k,i)))./sqrt((xz_surf(I_min(k,i),1)-
x_off).^2+xz_surf(I_min(k,i),2).^2);
          cos_t_nfx_nfz_ext_1(k,i)=((xz_surf(I_min(k,i),1)-x_off).*(phi_x_neg_ext(k,i))+xz_surf(I_min(k,i),2).*(phi_z_neg_ext(k,i)))./sqrt((xz_surf(I_min(k,i),1)-
x_off).^2+xz_surf(I_min(k,i),2).^2);
          cos_t_star_ext_1(k,i)=((xz_surf(I_min(k,i),1)-x_off).*(phi_x_star_ext(k,i))+xz_surf(I_min(k,i),2).*(phi_z_star_ext(k,i)))./sqrt((xz_surf(I_min(k,i),1)-
x_off).^2+xz_surf(I_min(k,i),2).^2);
          cos_t_cen_ext_1(k,i)=((xz_surf(I_min(k,i),1)-x_off).*(phi_x_cen_ext(k,i))+xz_surf(I_min(k,i),2).*(phi_z_cen_ext(k,i)))./sqrt((xz_surf(I_min(k,i),1)-
x_off).^2+xz_surf(I_min(k,i),2).^2);
          if (cos_t_pfx_pfz_ext_1(k,i)>1)
            cos_t_pfx_pfz_ext_1(k,i)=1;
          end
          if (cos_t_pfx_nfz_ext_1(k,i)>1)
            cos_t_pfx_nfz_ext_1(k,i)=1;
          end
          if (cos_t_nfx_pfz_ext_1(k,i)>1)
            cos_t_nfx_pfz_ext_1(k,i)=1;
          end
          if (cos_t_nfx_nfz_ext_1(k,i)>1)
            cos_t_nfx_nfz_ext_1(k,i)=1;
          end
          if (cos_t_star_ext_1(k,i)>1)
            cos_t_star_ext_1(k,i)=1;
          end
          if (cos_t_cen_ext_1(k,i)>1)
            cos_t_cen_ext_1(k,i)=1;
          end

          if (xz_surf(I_min(k,i),2)>=(h*sin(alfa))) %T
            if (cos_t_pfx_pfz_ext_1(k,i)<=0)
           F_ext_pfx_pfz_1_1st(k,i)=0;
           else
           F_ext_pfx_pfz_1_1st(k,i)=real((C/rho_s)*Eros_pow_ext(k,i).*((cos_t_pfx_pfz_ext_1(k,i))^(k_vel+1)));
            end
          else %M
           if (cos_t_pfx_pfz_ext_1(k,i)<=0) %Apply mask visibility for M
           F_ext_pfx_pfz_1_1st(k,i)=0;
           else
           F_ext_pfx_pfz_1_1st(k,i)=real((C_M/rho_s_M).*Eros_pow_ext(k,i).*((cos_t_pfx_pfz_ext_1(k,i)).^n_1_M).*((1+H_vic_M*(1-
cos_t_pfx_pfz_ext_1(k,i))).^n_2_M));
            end
          end

          %Note: cos_theta_D = 0 if no 2nd strk.
          if (xz_surf(I_min(k,i),2)>=(h*sin(alfa))) %T
           F_ext_pfx_pfz_1(k,i)=F_ext_pfx_pfz_1_1st(k,i)+real((C/rho_s)*Eros_pow_ext_2nd(k,i).*((cos_t_pfx_pfz_D(I_min(k,i)))^(k_vel+1)));
          else %M

F_ext_pfx_pfz_1(k,i)=F_ext_pfx_pfz_1_1st(k,i)+real((C_M/rho_s_M).*Eros_pow_ext_2nd(k,i).*((cos_t_pfx_pfz_D(I_min(k,i))).^n_1_M).*((1+H_vic_M*(1-
cos_t_pfx_pfz_D(I_min(k,i)))).^n_2_M));
          end

          if (xz_surf(I_min(k,i),2)>=(h*sin(alfa))) %T
            if (cos_t_pfx_nfz_ext_1(k,i)<=0)
           F_ext_pfx_nfz_1_1st(k,i)=0;
           else
           F_ext_pfx_nfz_1_1st(k,i)=real((C/rho_s)*Eros_pow_ext(k,i).*((cos_t_pfx_nfz_ext_1(k,i))^(k_vel+1)));
            end
          else %M
           if (cos_t_pfx_nfz_ext_1(k,i)<=0) %Apply mask visibility for M
           F_ext_pfx_nfz_1_1st(k,i)=0;
           else
           F_ext_pfx_nfz_1_1st(k,i)=real((C_M/rho_s_M).*Eros_pow_ext(k,i).*((cos_t_pfx_nfz_ext_1(k,i)).^n_1_M).*((1+H_vic_M*(1-
cos_t_pfx_nfz_ext_1(k,i))).^n_2_M));
            end
          end

          %Note: cos_theta_D = 0 if no 2nd strk.
          if (xz_surf(I_min(k,i),2)>=(h*sin(alfa))) %T
           F_ext_pfx_nfz_1(k,i)=F_ext_pfx_nfz_1_1st(k,i)+real((C/rho_s)*Eros_pow_ext_2nd(k,i).*((cos_t_pfx_nfz_D(I_min(k,i)))^(k_vel+1)));
          else %M

F_ext_pfx_nfz_1(k,i)=F_ext_pfx_nfz_1_1st(k,i)+real((C_M/rho_s_M).*Eros_pow_ext_2nd(k,i).*((cos_t_pfx_nfz_D(I_min(k,i))).^n_1_M).*((1+H_vic_M*(1-
cos_t_pfx_nfz_D(I_min(k,i)))).^n_2_M));
          end

          if (xz_surf(I_min(k,i),2)>=(h*sin(alfa))) %T
            if (cos_t_nfx_pfz_ext_1(k,i)<=0)
           F_ext_nfx_pfz_1_1st(k,i)=0;
           else
           F_ext_nfx_pfz_1_1st(k,i)=real((C/rho_s)*Eros_pow_ext(k,i).*((cos_t_nfx_pfz_ext_1(k,i))^(k_vel+1)));
            end
          else %M
           if (cos_t_nfx_pfz_ext_1(k,i)<=0) %Apply mask visibility for M
           F_ext_nfx_pfz_1_1st(k,i)=0;
           else
           F_ext_nfx_pfz_1_1st(k,i)=real((C_M/rho_s_M).*Eros_pow_ext(k,i).*((cos_t_nfx_pfz_ext_1(k,i)).^n_1_M).*((1+H_vic_M*(1-
cos_t_nfx_pfz_ext_1(k,i))).^n_2_M));
            end
          end

          %Note: cos_theta_D = 0 if no 2nd strk.
```

```
                if (xz_surf(I_min(k,i),2)>=(h*sin(alfa))) %T
                   F_ext_nfx_pfz_1(k,i)=F_ext_nfx_pfz_1_1st(k,i)+real((C/rho_s)*Eros_pow_ext_2nd(k,i).*((cos_t_nfx_pfz_D(I_min(k,i)))^(k_vel+1)));
                else %M

F_ext_nfx_pfz_1(k,i)=F_ext_nfx_pfz_1_1st(k,i)+real((C_M/rho_s_M).*Eros_pow_ext_2nd(k,i).*((cos_t_nfx_pfz_D(I_min(k,i))).^n_1_M).*((1+H_vic_M*(1-
cos_t_nfx_pfz_D(I_min(k,i)))).^n_2_M));
                end

                if (xz_surf(I_min(k,i),2)>=(h*sin(alfa))) %T
                    if (cos_t_nfx_nfz_ext_1(k,i)<=0)
                   F_ext_nfx_nfz_1_1st(k,i)=0;
                   else
                   F_ext_nfx_nfz_1_1st(k,i)=real((C/rho_s)*Eros_pow_ext(k,i).*((cos_t_nfx_nfz_ext_1(k,i))^(k_vel+1)));
                    end
                else %M
                   if (cos_t_nfx_nfz_ext_1(k,i)<=0) %Apply mask visibility for M
                   F_ext_nfx_nfz_1_1st(k,i)=0;
                   else
                   F_ext_nfx_nfz_1_1st(k,i)=real((C_M/rho_s_M).*Eros_pow_ext(k,i).*((cos_t_nfx_nfz_ext_1(k,i)).^n_1_M).*((1+H_vic_M*(1-
cos_t_nfx_nfz_ext_1(k,i))).^n_2_M));
                   end
                end

                %Note: cos_theta_D = 0 if no 2nd strk.
                if (xz_surf(I_min(k,i),2)>=(h*sin(alfa))) %T
                   F_ext_nfx_nfz_1(k,i)=F_ext_nfx_nfz_1_1st(k,i)+real((C/rho_s)*Eros_pow_ext_2nd(k,i).*((cos_t_nfx_nfz_D(I_min(k,i)))^(k_vel+1)));
                else %M

F_ext_nfx_nfz_1(k,i)=F_ext_nfx_nfz_1_1st(k,i)+real((C_M/rho_s_M).*Eros_pow_ext_2nd(k,i).*((cos_t_nfx_nfz_D(I_min(k,i))).^n_1_M).*((1+H_vic_M*(1-
cos_t_nfx_nfz_D(I_min(k,i)))).^n_2_M));
                end

                if (xz_surf(I_min(k,i),2)>=(h*sin(alfa))) %T
                    if (cos_t_star_ext_1(k,i)<=0)
                   F_ext_star_1_1st(k,i)=0;
                   else
                   F_ext_star_1_1st(k,i)=real((C/rho_s)*Eros_pow_ext(k,i).*((cos_t_star_ext_1(k,i))^(k_vel+1)));
                    end
                else %M
                   if (cos_t_star_ext_1(k,i)<=0) %Apply mask visibility for M
                   F_ext_star_1_1st(k,i)=0;
                   else
                   F_ext_star_1_1st(k,i)=real((C_M/rho_s_M).*Eros_pow_ext(k,i).*((cos_t_star_ext_1(k,i)).^n_1_M).*((1+H_vic_M*(1-
cos_t_star_ext_1(k,i))).^n_2_M));
                   end
                end

                %Note: cos_theta_D = 0 if no 2nd strk.
                if (xz_surf(I_min(k,i),2)>=(h*sin(alfa))) %T
                   F_ext_star_1(k,i)=F_ext_star_1_1st(k,i)+real((C/rho_s)*Eros_pow_ext_2nd(k,i).*((cos_t_star_D(I_min(k,i)))^(k_vel+1)));
                else %M
                   F_ext_star_1(k,i)=F_ext_star_1_1st(k,i)+real((C_M/rho_s_M).*Eros_pow_ext_2nd(k,i).*((cos_t_star_D(I_min(k,i))).^n_1_M).*((1+H_vic_M*(1-
cos_t_star_D(I_min(k,i)))).^n_2_M));
                end

                if (xz_surf(I_min(k,i),2)>=(h*sin(alfa))) %T
                    if (cos_t_cen_ext_1(k,i)<=0)
                   F_ext_cen_1_1st(k,i)=0;
                   else
                   F_ext_cen_1_1st(k,i)=real((C/rho_s)*Eros_pow_ext(k,i).*((cos_t_cen_ext_1(k,i))^(k_vel+1)));
                    end
                else %M
                   if (cos_t_cen_ext_1(k,i)<=0) %Apply mask visibility for M
                   F_ext_cen_1_1st(k,i)=0;
                   else
                   F_ext_cen_1_1st(k,i)=real((C_M/rho_s_M).*Eros_pow_ext(k,i).*((cos_t_cen_ext_1(k,i)).^n_1_M).*((1+H_vic_M*(1-
cos_t_cen_ext_1(k,i))).^n_2_M));
                   end
                end

                %Note: cos_theta_D = 0 if no 2nd strk.
                if (xz_surf(I_min(k,i),2)>=(h*sin(alfa))) %T
                   F_ext_cen_1(k,i)=F_ext_cen_1_1st(k,i)+real((C/rho_s)*Eros_pow_ext_2nd(k,i).*((cos_t_cen_D(I_min(k,i)))^(k_vel+1)));
                else %M
                   F_ext_cen_1(k,i)=F_ext_cen_1_1st(k,i)+real((C_M/rho_s_M).*Eros_pow_ext_2nd(k,i).*((cos_t_cen_D(I_min(k,i))).^n_1_M).*((1+H_vic_M*(1-
cos_t_cen_D(I_min(k,i)))).^n_2_M));
                end


        else %we are on surface and grid node = surface node (and phi=0)
        %Repeat above algorithm but using grid nodes
%____________________________________________________________________________
        %Calculation of dphi/dx,dphi/dz; Define BC's
        if i==i_max
          phi_x_pos_ext(k,i)=0;
        else
          phi_x_pos_ext(k,i)=(phi_1(k,i+1)-phi_1(k,i))./dx;
        end

        if i==1
          phi_x_neg_ext(k,i)=0;
        else
          phi_x_neg_ext(k,i)=(phi_1(k,i)-phi_1(k,i-1))./dx;
        end

        if (i==i_max)||(i==1)
          phi_x_cen_ext(k,i)=0;
        else
          phi_x_cen_ext(k,i)=(phi_1(k,i+1)-phi_1(k,i-1))./(2*dx);
        end

        if k==1
          phi_z_pos_ext(k,i)=0;
        else
          phi_z_pos_ext(k,i)=(phi_1(k-1,i)-phi_1(k,i))./dz;
        end

        if k==k_max
          phi_z_neg_ext(k,i)=0;
        else
          phi_z_neg_ext(k,i)=(phi_1(k,i)-phi_1(k+1,i))./dz;
        end

        if (k==k_max)||(k==1)
          phi_z_cen_ext(k,i)=0;
        else
          phi_z_cen_ext(k,i)=(phi_1(k-1,i)-phi_1(k+1,i))./(2*dz);
        end

        %Calculate phi_stars
        phi_x_star_ext(k,i)=(phi_x_pos_ext(k,i)+phi_x_neg_ext(k,i))/2;
        phi_z_star_ext(k,i)=(phi_z_pos_ext(k,i)+phi_z_neg_ext(k,i))/2;
%____________________________________________________________________________
        %Masking function for T, M
        x_prime_ext(k,i)=x_cord_local(i).*sin(alfa)-z_cord(k).*cos(alfa); %Rotated local x
        z_prime_ext(k,i)=x_cord_local(i).*cos(alfa)+z_cord(k).*sin(alfa); %Rotated local z

            if (z_cord(k)>=(h*sin(alfa))) %T

            if (z_cord(k)>=(h*sin(alfa)+dz*MT_pt_dist))

            if (max_x_prime_surf_LM<0) %Case (a) and (c)
              if (x_prime_ext(k,i)<0)
                L_mask_ext(k,i)=z_prime_ext(k,i).*tan_fi_min;
              else %i.e., when x_prime>=0
                L_mask_ext(k,i)=z_prime_ext(k,i).*tan_fi_max;
              end
            else %i.e., when x_m>=W_m/2 Case (b)
              x_lim_ext(k,i)=z_prime_ext(k,i).*tan_fi_min;
              if (x_prime_ext(k,i)<x_lim_ext(k,i))
                L_mask_ext(k,i)=0;
              else %i.e., when x_prime>=x_lim
                L_mask_ext(k,i)=z_prime_ext(k,i).*tan_fi_max-x_lim_ext(k,i);
              end
            end

            %Define proportion of mass of particle that pass through mask opening having a
            %specific particle size (of radius r) distribution
            if (abs(x_prime_ext(k,i))>=L_mask_ext(k,i))
              M_r_x_prime_ext(k,i)=0;
            else
              Int_P_r_x_prime_ext(k,i)=real(P_1-P_1*erf(P_2*log(L_mask_ext(k,i)-abs(x_prime_ext(k,i)))-P_3));
              Int_P_r_L_mask_ext(k,i)=real(P_1-P_1*erf(P_2*log(L_mask_ext(k,i))-P_3));
              M_r_x_prime_ext(k,i)=Int_P_r_x_prime_ext(k,i)./Int_P_r_L_mask_ext(k,i);
            end

            else %T w/in zTadj
              if ((time*v_scan/(2*r_s))<=No_p_MrxM_ON)
                  M_r_x_prime_ext(k,i)=1;
              else
              if ((x_cord(i)>=x_min_grid)&&(x_cord(i)<=(x_min_grid+leng_M_L)))%TL
                if (x_cord(i)<(x_lim_LM-dx))
                  M_r_x_prime_ext(k,i)=1;
                else
                 M_r_x_prime_ext(k,i)=M_r_x_prime_LM;
                end
              elseif ((x_cord(i)>=(x_max_grid-leng_M_R))&&(x_cord(i)<=x_max_grid)) %TR
                if (x_cord(i)>(x_lim_RM+dx))
                  M_r_x_prime_ext(k,i)=1;
                else
                  M_r_x_prime_ext(k,i)=M_r_x_prime_RM;
                end

              else
                M_r_x_prime_ext(k,i)=1;
              end

              end
            end

            else  %M
              if ((time*v_scan/(2*r_s))<=No_p_MrxM_ON)
                   M_r_x_prime_ext(k,i)=1;
              else

              if ((x_cord(i)>=x_min_grid)&&(x_cord(i)<=(x_min_grid+leng_M_L)))%ML
                if (x_cord(i)<(x_lim_LM-dx))
                  M_r_x_prime_ext(k,i)=1;
                else
                 M_r_x_prime_ext(k,i)=M_r_x_prime_LM;
                end
              elseif ((x_cord(i)>=(x_max_grid-leng_M_R))&&(x_cord(i)<=x_max_grid)) %MR
                if (x_cord(i)>(x_lim_RM+dx))
                  M_r_x_prime_ext(k,i)=1;
                else
                  M_r_x_prime_ext(k,i)=M_r_x_prime_RM;
                end

              else
                M_r_x_prime_ext(k,i)=1;
              end

              end

            end

%___________________________________________________________________________
            %Define velocity v(x,z) at each grid node
              v_ext(k,i)=v_o*(1-H_slp*((x_cord_local(i)*sin(alfa)-z_cord(k)*cos(alfa)).^2+(y_mean).^2).^0.5./(x_cord_local(i)*cos(alfa)+z_cord(k)*sin(alfa)));
            if (v_ext(k,i)<0)
              v_ext(k,i)=0;
            end

            %Define particle mass flux(x,z) at each grid node
              flux_ext(k,i)=(MFR/pi)*(beta./(x_cord_local(i)*cos(alfa)+z_cord(k)*sin(alfa))).^2....
              *exp(-(beta^2.*((x_cord_local(i)*sin(alfa)-z_cord(k)*cos(alfa)).^2+(y_mean).^2)./(x_cord_local(i)*cos(alfa)+z_cord(k)*sin(alfa)).^2));

            %Define Erosive Power Eros_pow(k,i) at each grid
            %node (1st Strike)
            if (z_cord(k)>=(h*sin(alfa))) %T
              Eros_pow_ext(k,i)=M_r_x_prime_ext(k,i).*v_ext(k,i).^k_vel.*flux_ext(k,i);
            else %M
              Eros_pow_ext(k,i)=M_r_x_prime_ext(k,i).*v_ext(k,i).^k_vel_M.*flux_ext(k,i);
            end

            %Define Erosive Power for 2nd strike
            %NOTE: No Mask here
            if (z_cord(k)>=(h*sin(alfa))) %T
              if (NaN_Chk_theta_D(I_min(k,i))==0)
                Eros_pow_ext_2nd(k,i)=(f_v_AR_fin(I_min(k,i)).*v_AR(I_min(k,i))).^k_vel.*flux_AR(I_min(k,i));
              else
                Eros_pow_ext_2nd(k,i)=0;
              end
            else %M
              if (NaN_Chk_theta_D(I_min(k,i))==0)
                Eros_pow_ext_2nd(k,i)=(f_v_AR_fin(I_min(k,i)).*v_AR(I_min(k,i))).^k_vel_M.*flux_AR(I_min(k,i));
              else
                Eros_pow_ext_2nd(k,i)=0;
              end
```

```
           end

%_____________________________________________________________________________________
      %Calculate F_extensions

        cos_t_pfx_pfz_ext_1(k,i)=(x_cord_local(i).*(phi_x_pos_ext(k,i))+z_cord(k).*(phi_z_pos_ext(k,i)))./sqrt(x_cord_local(i).^2+z_cord(k).^2);
        cos_t_pfx_nfz_ext_1(k,i)=(x_cord_local(i).*(phi_x_pos_ext(k,i))+z_cord(k).*(phi_z_neg_ext(k,i)))./sqrt(x_cord_local(i).^2+z_cord(k).^2);
        cos_t_nfx_pfz_ext_1(k,i)=(x_cord_local(i).*(phi_x_neg_ext(k,i))+z_cord(k).*(phi_z_pos_ext(k,i)))./sqrt(x_cord_local(i).^2+z_cord(k).^2);
        cos_t_nfx_nfz_ext_1(k,i)=(x_cord_local(i).*(phi_x_neg_ext(k,i))+z_cord(k).*(phi_z_neg_ext(k,i)))./sqrt(x_cord_local(i).^2+z_cord(k).^2);
        cos_t_star_ext_1(k,i)=(x_cord_local(i).*(phi_x_star_ext(k,i))+z_cord(k).*(phi_z_star_ext(k,i)))./sqrt(x_cord_local(i).^2+z_cord(k).^2);
        cos_t_cen_ext_1(k,i)=(x_cord_local(i).*(phi_x_cen_ext(k,i))+z_cord(k).*(phi_z_cen_ext(k,i)))./sqrt(x_cord_local(i).^2+z_cord(k).^2);
        if (cos_t_pfx_pfz_ext_1(k,i)>1)
           cos_t_pfx_pfz_ext_1(k,i)=1;
        end
        if (cos_t_pfx_nfz_ext_1(k,i)>1)
           cos_t_pfx_nfz_ext_1(k,i)=1;
        end
        if (cos_t_nfx_pfz_ext_1(k,i)>1)
           cos_t_nfx_pfz_ext_1(k,i)=1;
        end
        if (cos_t_nfx_nfz_ext_1(k,i)>1)
           cos_t_nfx_nfz_ext_1(k,i)=1;
        end
        if (cos_t_star_ext_1(k,i)>1)
           cos_t_star_ext_1(k,i)=1;
        end
        if (cos_t_cen_ext_1(k,i)>1)
           cos_t_cen_ext_1(k,i)=1;
        end

        if (z_cord(k)>=(h*sin(alfa))) %T
           if (cos_t_pfx_pfz_ext_1(k,i)<=0)
          F_ext_pfx_pfz_1_1st(k,i)=0;
          else
          F_ext_pfx_pfz_1_1st(k,i)=real((C/rho_s)*Eros_pow_ext(k,i).*((cos_t_pfx_pfz_ext_1(k,i))^(k_vel+1)));
           end
        else %M
          if (cos_t_pfx_pfz_ext_1(k,i)<=0) %Apply mask visibility for M
          F_ext_pfx_pfz_1_1st(k,i)=0;
          else
          F_ext_pfx_pfz_1_1st(k,i)=real((C_M/rho_s_M).*Eros_pow_ext(k,i).*((cos_t_pfx_pfz_ext_1(k,i)).^n_1_M).*((1+H_vic_M*(1-
cos_t_pfx_pfz_ext_1(k,i))).^n_2_M));
           end
        end

          %Note: cos_theta_D = 0 if no 2nd strk.
        if (z_cord(k)>=(h*sin(alfa))) %T
          F_ext_pfx_pfz_1(k,i)=F_ext_pfx_pfz_1_1st(k,i)+real((C/rho_s)*Eros_pow_ext_2nd(k,i).*((cos_t_pfx_pfz_D(I_min(k,i)))^(k_vel+1)));
        else %M

F_ext_pfx_pfz_1(k,i)=F_ext_pfx_pfz_1_1st(k,i)+real((C_M/rho_s_M).*Eros_pow_ext_2nd(k,i).*((cos_t_pfx_pfz_D(I_min(k,i))).^n_1_M).*((1+H_vic_M*(1-
cos_t_pfx_pfz_D(I_min(k,i)))).^n_2_M));
        end

        if (z_cord(k)>=(h*sin(alfa))) %T
           if (cos_t_pfx_nfz_ext_1(k,i)<=0)
          F_ext_pfx_nfz_1_1st(k,i)=0;
          else
          F_ext_pfx_nfz_1_1st(k,i)=real((C/rho_s)*Eros_pow_ext(k,i).*((cos_t_pfx_nfz_ext_1(k,i))^(k_vel+1)));
           end
        else %M
          if (cos_t_pfx_nfz_ext_1(k,i)<=0) %Apply mask visibility for M
          F_ext_pfx_nfz_1_1st(k,i)=0;
          else
          F_ext_pfx_nfz_1_1st(k,i)=real((C_M/rho_s_M).*Eros_pow_ext(k,i).*((cos_t_pfx_nfz_ext_1(k,i)).^n_1_M).*((1+H_vic_M*(1-
cos_t_pfx_nfz_ext_1(k,i))).^n_2_M));
           end
        end

          %Note: cos_theta_D = 0 if no 2nd strk.
        if (z_cord(k)>=(h*sin(alfa))) %T
          F_ext_pfx_nfz_1(k,i)=F_ext_pfx_nfz_1_1st(k,i)+real((C/rho_s)*Eros_pow_ext_2nd(k,i).*((cos_t_pfx_nfz_D(I_min(k,i)))^(k_vel+1)));
        else %M

F_ext_pfx_nfz_1(k,i)=F_ext_pfx_nfz_1_1st(k,i)+real((C_M/rho_s_M).*Eros_pow_ext_2nd(k,i).*((cos_t_pfx_nfz_D(I_min(k,i))).^n_1_M).*((1+H_vic_M*(1-
cos_t_pfx_nfz_D(I_min(k,i)))).^n_2_M));
        end

        if (z_cord(k)>=(h*sin(alfa))) %T
           if (cos_t_nfx_pfz_ext_1(k,i)<=0)
```

```
                  F_ext_nfx_pfz_1_1st(k,i)=0;
                  else
                  F_ext_nfx_pfz_1_1st(k,i)=real((C/rho_s)*Eros_pow_ext(k,i).*((cos_t_nfx_pfz_ext_1(k,i))^(k_vel+1)));
                   end
                else %M
                  if (cos_t_nfx_pfz_ext_1(k,i)<=0) %Apply mask visibility for M
                  F_ext_nfx_pfz_1_1st(k,i)=0;
                  else
                  F_ext_nfx_pfz_1_1st(k,i)=real((C_M/rho_s_M).*Eros_pow_ext(k,i).*((cos_t_nfx_pfz_ext_1(k,i)).^n_1_M).*((1+H_vic_M*(1-
cos_t_nfx_pfz_ext_1(k,i))).^n_2_M));
                  end
                end

                %Note: cos_theta_D = 0 if no 2nd strk.
                if (z_cord(k)>=(h*sin(alfa))) %T
                  F_ext_nfx_pfz_1(k,i)=F_ext_nfx_pfz_1_1st(k,i)+real((C/rho_s)*Eros_pow_ext_2nd(k,i).*((cos_t_nfx_pfz_D(I_min(k,i)))^(k_vel+1)));
                else %M

F_ext_nfx_pfz_1(k,i)=F_ext_nfx_pfz_1_1st(k,i)+real((C_M/rho_s_M).*Eros_pow_ext_2nd(k,i).*((cos_t_nfx_pfz_D(I_min(k,i))).^n_1_M).*((1+H_vic_M*(1-
cos_t_nfx_pfz_D(I_min(k,i)))).^n_2_M));
                end

                if (z_cord(k)>=(h*sin(alfa))) %T
                   if (cos_t_nfx_nfz_ext_1(k,i)<=0)
                  F_ext_nfx_nfz_1_1st(k,i)=0;
                  else
                  F_ext_nfx_nfz_1_1st(k,i)=real((C/rho_s)*Eros_pow_ext(k,i).*((cos_t_nfx_nfz_ext_1(k,i))^(k_vel+1)));
                   end
                else %M
                  if (cos_t_nfx_nfz_ext_1(k,i)<=0) %Apply mask visibility for M
                  F_ext_nfx_nfz_1_1st(k,i)=0;
                  else
                  F_ext_nfx_nfz_1_1st(k,i)=real((C_M/rho_s_M).*Eros_pow_ext(k,i).*((cos_t_nfx_nfz_ext_1(k,i)).^n_1_M).*((1+H_vic_M*(1-
cos_t_nfx_nfz_ext_1(k,i))).^n_2_M));
                  end
                end

                %Note: cos_theta_D = 0 if no 2nd strk.
                if (z_cord(k)>=(h*sin(alfa))) %T
                  F_ext_nfx_nfz_1(k,i)=F_ext_nfx_nfz_1_1st(k,i)+real((C/rho_s)*Eros_pow_ext_2nd(k,i).*((cos_t_nfx_nfz_D(I_min(k,i)))^(k_vel+1)));
                else %M

F_ext_nfx_nfz_1(k,i)=F_ext_nfx_nfz_1_1st(k,i)+real((C_M/rho_s_M).*Eros_pow_ext_2nd(k,i).*((cos_t_nfx_nfz_D(I_min(k,i))).^n_1_M).*((1+H_vic_M*(1-
cos_t_nfx_nfz_D(I_min(k,i)))).^n_2_M));
                end

                if (z_cord(k)>=(h*sin(alfa))) %T
                   if (cos_t_star_ext_1(k,i)<=0)
                  F_ext_star_1_1st(k,i)=0;
                  else
                  F_ext_star_1_1st(k,i)=real((C/rho_s)*Eros_pow_ext(k,i).*((cos_t_star_ext_1(k,i))^(k_vel+1)));
                   end
                else %M
                  if (cos_t_star_ext_1(k,i)<=0) %Apply mask visibility for M
                  F_ext_star_1_1st(k,i)=0;
                  else
                  F_ext_star_1_1st(k,i)=real((C_M/rho_s_M).*Eros_pow_ext(k,i).*((cos_t_star_ext_1(k,i)).^n_1_M).*((1+H_vic_M*(1-
cos_t_star_ext_1(k,i))).^n_2_M));
                  end
                end

                %Note: cos_theta_D = 0 if no 2nd strk.
                if (z_cord(k)>=(h*sin(alfa))) %T
                  F_ext_star_1(k,i)=F_ext_star_1_1st(k,i)+real((C/rho_s)*Eros_pow_ext_2nd(k,i).*((cos_t_star_D(I_min(k,i)))^(k_vel+1)));
                else %M
                  F_ext_star_1(k,i)=F_ext_star_1_1st(k,i)+real((C_M/rho_s_M).*Eros_pow_ext_2nd(k,i).*((cos_t_star_D(I_min(k,i))).^n_1_M).*((1+H_vic_M*(1-
cos_t_star_D(I_min(k,i)))).^n_2_M));
                end

                if (z_cord(k)>=(h*sin(alfa))) %T
                   if (cos_t_cen_ext_1(k,i)<=0)
                  F_ext_cen_1_1st(k,i)=0;
                  else
                  F_ext_cen_1_1st(k,i)=real((C/rho_s)*Eros_pow_ext(k,i).*((cos_t_cen_ext_1(k,i))^(k_vel+1)));
                   end
                else %M
                  if (cos_t_cen_ext_1(k,i)<=0) %Apply mask visibility for M
                  F_ext_cen_1_1st(k,i)=0;
                  else
                  F_ext_cen_1_1st(k,i)=real((C_M/rho_s_M).*Eros_pow_ext(k,i).*((cos_t_cen_ext_1(k,i)).^n_1_M).*((1+H_vic_M*(1-
cos_t_cen_ext_1(k,i))).^n_2_M));
                  end
                end

                %Note: cos_theta_D = 0 if no 2nd strk.
                if (z_cord(k)>=(h*sin(alfa))) %T
                  F_ext_cen_1(k,i)=F_ext_cen_1_1st(k,i)+real((C/rho_s)*Eros_pow_ext_2nd(k,i).*((cos_t_cen_D(I_min(k,i)))^(k_vel+1)));
                else %M
                  F_ext_cen_1(k,i)=F_ext_cen_1_1st(k,i)+real((C_M/rho_s_M).*Eros_pow_ext_2nd(k,i).*((cos_t_cen_D(I_min(k,i))).^n_1_M).*((1+H_vic_M*(1-
cos_t_cen_D(I_min(k,i)))).^n_2_M));
                end

        end


        end  %#####*****#####$$$$$#####*****#####
      end
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
                   %END OF SDF AND F_EXT ALGORITHM
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%__________________________________________________________________________________
%Update F_ext for next iteration
F_ext_pfx_pfz=F_ext_pfx_pfz_1;
F_ext_pfx_nfz=F_ext_pfx_nfz_1;
F_ext_nfx_pfz=F_ext_nfx_pfz_1;
F_ext_nfx_nfz=F_ext_nfx_nfz_1;
F_ext_star=F_ext_star_1;
F_ext_cen=F_ext_cen_1;

%__________________________________________________________________________________
%Time counter
time=time+dt;

%Iteration counter
counter=counter+1;

%__________________________________________________________________________________
%XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
%XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
%Save data to Data Files

N_chk=v_scan/(2*r_s); %Extract data when specified pass reached
if ((time*N_chk>=2)&&(time*N_chk<(2+dt*N_chk)))||((time*N_chk>=4)&&(time*N_chk<(4+dt*N_chk)))...
  ||((time*N_chk>=6)&&(time*N_chk<(6+dt*N_chk)))||((time*N_chk>=10)&&(time*N_chk<(10+dt*N_chk)))

xz_surf_final=zeros(b_max,2); %Transformed surface points for plotting
for b=1:1:b_max
        if (xz_surf(b,1)==0)&&(xz_surf(b,2)==0)
        xz_surf_final(b,1)=NaN; %Not a Number (no cell entry in Excel)
        xz_surf_final(b,2)=NaN;
        else
        xz_surf_final(b,1)=xz_surf(b,1)-x_min_grid; %Transform x
        xz_surf_final(b,2)=-(xz_surf(b,2)-(h*sin(alfa)-H_m)); %Transform z
        end
end

%Input Data
inputs_xls={'time(s)=',time;'MFR(kg/s)=',MFR;'C()=',C;'H_slp()=',H_slp;'beta()=',beta;'v_o(m/s)=',v_o;'v_scan(m/s)=',v_scan;'rho_s(kg/m3)=',rho_s;...
   'k_vel()=',k_vel;'alfa(rad)=',alfa;'epsilon()=',epsilon;'h()=',h;'i_max()=',i_max;'k_max()=',k_max;'dx(m)=',dx;'dz(m)=',dz;'dt(m)=',dt;...
'z_in(m)=',z_in;'z_air(m)=',z_air;'W_m(m)=',W_m;'H_m(m)=',H_m;'mu_l()=',mu_l;'sigma_l()=',sigma_l;'C_M()=',C_M*(adj_r_s/adj_r_s_M);'rho_s_M(kg/m3)=',rho
_s_M;...
   'k_vel_M()=',k_vel_M;'H_vic_M(GPa)=',H_vic_M;'n_1_M()=',n_1_M;'n_2_M()=',n_2_M;'leng_M_L(m)=',leng_M_L;'leng_M_R(m)=',leng_M_R};

%Write data to Excel
if ((time*N_chk>=2)&&(time*N_chk<(2+dt*N_chk)))
headings_xls={'2passes',NaN,'x_surf_fin','z_surf_fin'};
xlswrite(fil_name,headings_xls,'Sheet1','A1');
xlswrite(fil_name,inputs_xls,'Sheet1','A2');
xlswrite(fil_name,xz_surf_final,'Sheet1','C2');
end
if ((time*N_chk>=4)&&(time*N_chk<(4+dt*N_chk)))
headings_xls={'4passes',NaN,'x_surf_fin','z_surf_fin'};
xlswrite(fil_name,headings_xls,'Sheet1','E1');
xlswrite(fil_name,inputs_xls,'Sheet1','E2');
xlswrite(fil_name,xz_surf_final,'Sheet1','G2');
end
if ((time*N_chk>=6)&&(time*N_chk<(6+dt*N_chk)))
```

```
headings_xls={'6passes',NaN,'x_surf_fin','z_surf_fin'};
xlswrite(fil_name,headings_xls,'Sheet1','I1');
xlswrite(fil_name,inputs_xls,'Sheet1','I2');
xlswrite(fil_name,xz_surf_final,'Sheet1','K2');
end
if ((time*N_chk>=10)&&(time*N_chk<(10+dt*N_chk)))
headings_xls={'10passes',NaN,'x_surf_fin','z_surf_fin'};
xlswrite(fil_name,headings_xls,'Sheet1','M1');
xlswrite(fil_name,inputs_xls,'Sheet1','M2');
xlswrite(fil_name,xz_surf_final,'Sheet1','O2');
end


end



end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
                        %END OF Main Loop
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%___________________________________________________________________________
%Number of iterations
counter;

toc; %Stop timer
Exec_time=toc; %Execution time
%___________________________________________________________________________
%Write other data to Excel at end of execution
other_data_xls={'execution time(s)=',Exec_time;'Number of Iterations=',counter;'r_s(m)=',(1/adj_r_s)*r_s;'Visibility=',Visibility;...
'BS_L=',BS_L;'BS_U=',BS_U;'Num_iter_RE=',Num_iter_RE;'prop_Num_iter_RE=',prop_Num_iter_RE;'Crit_D=',Crit_D;'No_RE=',No_RE;'adj_r_s=',adj_r_s;'adj_r_s_M=',adj_r_s_M;...
'T_pass_T=',T_pass_T;'T_pass_M=',T_pass_M;'dt_alpha=',dt_alpha;'prs=',(y_mean*adj_r_s)/r_s;'MT_pt_dist=',MT_pt_dist;'No_p_MrxM_ON=',No_p_MrxM_ON;'r_UCt=',r_UCt;...
    'f_alfa_AR_T=',f_alfa_AR_T;'f_v_AR_T=',f_v_AR_T;'f_alfa_AR_M=',f_alfa_AR_M;'f_v_AR_M=',f_v_AR_M;'ds_crit=',ds_crit;'No_ds_cr',No_ds_cr};
xlswrite(fil_name,other_data_xls,'Sheet1','Y1');
```

References

[1] J.H.M. ten Thije Boonkkamp, J.K.M. Jansen, An analytical solution for mechanical etching of glass by powder blasting, *J. Eng. Math.* 43 (2002) 385–399.

[2] H. Getu, A. Ghobeity, J.K. Spelt, M. Papini, Abrasive jet micromachining of polymethylmethacrylate, *Wear* 263 (2007) 1008–1015.

[3] Y.H. Kim, H.W. Lee, K.W. Lee, Y.S. Kim, Formation of barrier ribs for plasma display panels via a powder-blasting process: Part I. Effects of binder content and baking treatment, *J. Am. Ceram.* Soc. 87 (2004) 342–347.

[4] E. Belloy, A. Sayah, M.A.M. Gijs, Micromachining of glass inertial sensors, *J. Microelectromech. Syst.* 11 (1) (2002) 85–90.

[5] S. Schelautmann, H. Wensink, R. Schasfoort, M. Elwenspoek, A.V.D. Berg, Powder-blasting technology as an alternative tool for microfabrication of capillary electrophoresis chips with integrated conductivity sensors, *J. Micromech. Microeng.* 11 (2001) 386–389.

[6] E. Belloy, A.G. Pawlowski, A. Sayah, M.A.M. Gijs, Microfabrication of high-aspect ratio and complex monolithic structures in glass, *J. Microelectromech. Syst.* 11 (5) (2002) 521–526.

[7] Q.S. Yan, Z.Q. Zhang, Present situation and developing trend of abrasive air jet micromachining, *Key Eng. Mater.* 259-260 (2004) 648-652.

[8] H. Getu, J.K. Spelt, M. Papini, Cryogenically Assisted Abrasive Jet Micromachining of Polymers, *J. Micromech. Microeng.* 18 (2008) 115010.

[9] C. Liu, J. Chen, J. Engel, J. Zou, X. Wang, Z. Fan, K. Ryu, K. Shaikh, D. Bullen, Polymer micromachining and applications in sensors, microfluidics, and nanotechnology, *The 226th National Meeting of the American Chemical Society (ACS)*, New York, NY, 11–17 September, 2003.

[10] D.S. Park, M.W. Cho, T.I. Seo, Mechanical etching of micro pockets by powder blasting, *International Journal of Advanced Manufacturing Technology* 25 (11-12) (2005) 1098–1104.

[11] D. Belder, F. Kohler, M. Ludwig, K. Tolba, N. Piehl, Coating of powder-blasted channels for high-performance microchip electrophoresis, *Electrophoresis* 27 (2006) 3277–3283.

[12] A.G. Pawlowski, E. Belloy, A. Sayah, M.A.M. Gijs, Powder blasting patterning technology for microfabrication of complex suspended structures in glass, *Microelectron. Eng.* 67–68 (2003) 557–565.

[13] P. J. Slikkerveer and F. H. in‘t Veld, Model for patterned erosion, *Wear* 233-235 (1999) 377–386.

[14] A. Ghobeity, T. Krajac, T. Burzynski, M. Papini, J.K. Spelt, Surface evolution models in abrasive jet micromachining, *Wear* 264 (2008) 185–198.

[15] J.G.A. Bitter, A study of erosion phenomena - Part II, *Wear* 6 (1963) 169–190.

[16] J.A. Sethian, Level Set and Fast Marching Methods: Evolving interfaces in computational geometry, fluid mechanics, computer vision, and material science, second ed., *Cambridge University Press*, Cambridge, 1999.

[17] A. Ghobeity, H Getu, M Papini, J K Spelt, Surface evolution models for abrasive jet micromachining of holes in glass and polymethylmethacrylate (PMMA), *J. Micromech. Microeng*. 17 (2007) 2175–2185.

[18] Z. Moktadir, H. Wensink, M. Kraft, Analytical model of micromachining of brittle materials with sharp particles, *Microelectronics Journal* 36 (2005) 608–611.

[19] A. Ghobeity, D. Ciampini, M. Papini, An analytical model of the effect of particle size distribution on the surface profile evolution in abrasive jet micromachining, *J. Mater. Process. Technol.* 209 (20) (2009) 6067-6077.

[20] H. Wensink , J.W. Berenschot, H.V. Jansen, M. C. Elwenspoek, High resolution powder blast micromachining, *Proc.* 13th *Int. Workshop on Micro Electro Mechanical Systems, MEMS 2000,* Miyazaki, Japan (2000) 769–774.

[21] H. Wensink, M.C. Elwenspoek, Reduction of sidewall inclination and blast lag of powder blasted channels, *Sensors and Actuators A* 102 (2002) 157–164.

[22] A. Ghobeity, M. Papini, J.K. Spelt, Computer simulation of particle interference in abrasive jet micromachining, *Wear* 263 (1-6) (2007) 265–269.

[23] D. Ciampini, M. Papini, Cellular-automata and particle-tracking simulation of abrasive jet micromachining considering particle spatial hindering and second strikes, *J. Micromech. Microeng.* 20 (2010) 045025.

[24] D.S. Park, M.W. Cho, H. Lee, Effects of the impact angle variations on the erosion rate of glass in powder blasting process, *Int. J. Adv. Manuf. Technol.* 23 (2004) 444–450.

[25] H. Wensink, H.V. Jansen, J.W. Berenschot, M.C. Elwenspoek, Mask materials for powder blasting, *J. Micromech. Microeng*. 10 (2000) 175–180.

[26] M. Achtsnick, J. Drabbe, A.M. Hoogstrate, B. Karpuschewski, Erosion behaviour and pattern transfer accuracy of protecting masks for micro-abrasive blasting, *Journal of Materials Processing Technology* 149 (2004) 43–49.

[27] A.G. Pawlowski, A. Sayah, M.A.M Gijs, Accurate masking technology for high-resolution powder blasting, *J. Micromech. Microeng*. 15 (2005) S60–S64.

[28] A. Sayah, V.K. Parashar, A.G. Pawlowski, M.A.M Gijs, Elastomer mask for powder blasting microfabrication, *Sensors and Actuators A* 125 (2005) 84–90.

[29] P.J. Slikkerveer, P.C.P. Bouten, F.C.M. de Haas, High quality mechanical etching of brittle materials by powder blasting, *Sensors and Actuators A* 85 (2000) 296–303.

[30] E. Belloy, A. Sayah, M.A.M. Gijs, Oblique powder blasting for three-dimensional micromachining of brittle matrials, *Sensors and Actuators A* 92 (2001) 358–363.

[31] H. Getu, A. Ghobeity, J.K. Spelt, and M. Papini, Abrasive jet micromachining of acrylic and polycarbonate polymers at oblique angles of attack, *Wear* 265 (5-6) (2008) 888-901.

[32] H. Yagyu, O. Tabata, Three-dimensional simulation of powder blasting with a polymer mask using a cellular automaton, *J. Micromech. Microeng.*, 18 (2008) 1–9.

[33] T. Burzynski, M. Papini, Analytical models of the interference between incident and rebounding articles within an abrasive jet: Comparison with computer simulation, *Wear* 263 (7-12) (2007) 1593-1601.

[34] T. Burzynski, M. Papini, Analytical model of particle interference effects in divergent erosive jets, *Tribol. Int.* 43 (3) (2010) 554-567.

[35] D. Ciampini, J.K. Spelt, M. Papini, Simulation of interference effects in particle streams following impact with a flat surface, Part I: Theory and analysis, *Wear* 254 (2003) 237–249.

[36] N. Shafiei, H. Getu, A. Sadeghian, M. Papini, Computer simulation of developing abrasive jet machined profiles including particle interference, *J. Mater. Process. Technol.* 209 (9) (2009) 4366-4378.

[37] S. Osher, R.P. Fedkiw, Level Set Methods and Dynamic Implicit Surfaces, first ed., *Springer,* Verlag New York, 2002.

[38] Wikipedia: The free encyclopedia, File: Level set method.jpg (2004, November 25). Retrieved September 12, 2011, from http://en.wikipedia.org/wiki/File:Level_set_method.jpg.

[39] T. Burzynski, M. Papini, Level set methods for the modelling of surface evolution in the abrasive jet micromachining of features used in MEMS and microfluidic devices, *J. Micromech. Microeng.* 20 (2010) 085004.

[40] Y.I. Oka, H. Ohnogi, T. Hosokawa, M. Matsumura, The impact angle dependence of erosion damage caused by solid particle impact, *Wear* 203-204 (1997) 573-579.

[41] P.H. Shipway, I.M. Hutchings, Influence of nozzle roughness on conditions in gas-blast erosion rig, *Wear* 162-164 (1993) 148–158.

[42] T. Burzynski, M. Papini, Measurement of the particle spatial and velocity distributions in micro-abrasive jets, *Measurement Science and Technology* 22 (2011) 025104.

[43] A. Ghobeity, H. Getu, T. Krajac, J.K. Spelt, M. Papini, Process repeatability in abrasive jet micro-machining, *J. Mater. Process. Technol.* 190 (1-3) (2007) 51-60.

[44] A. Ghobeity, Beta and velocity for 0.3 mm x 3.8 mm rectangular nozzle [e-mail], Personal communication, 15 April, 2010.

[45] W.E. Schiesser, G.W. Griffiths, A Compendium of Partial Differential Equation Models: Method of Lines Analysis with Matlab, first ed., *Cambridge University Press*, New York, 2009.

[46] T. Burzynski, M. Papini, A level set methodology for predicting the surface evolution of inclined masked micro-channels resulting from abrasive jet micro-machining at oblique incidence, *International Journal of Machine Tools and Manufacture* 51 (2011) 628–641.

[47] T.S.J. Lammerink, V.L. Spiering, M. Elwenspoek, J.H.J. Fluitman, A. Berg van den, Modular concept for fluid handling systems: a demonstrator micro analysis system, *The Ninth Annual International Workshop on Micro Electro Mechanical Systems, MEMS, 'An Investigation of Micro Structures, Sensors, Actuators, Machines and Systems'*, San Diego, CA, 11-15 February, 1996.

[48] R.E. Oosterbroek, T.S.J. Lammerink, J.W. Berenschot, G.J.M. Krijnen, M.C. Elwenspoek, A. Berg van den, A micromachined pressure/flow-sensor, *Sens. Actuators A Phys.*77 (3) (1999) 167–177.

[49] D. Adalsteinsson, J.A. Sethian, A level set approach to a unified model for etching, deposition, and lithography I: Algorithms and two-dimensional simulations, *Journal of Computational Physics* 120 (1995) 128-144.

[50] T. Burzynski, M. Papini, A level set methodology for predicting the effect of mask wear on surface evolution of features in abrasive jet micro-machining, *J. Micromech. Microeng.*, submitted July 2011 (under review).

[51] T. Burzynski, M. Papini, Modelling of surface evolution in the abrasive jet micro-machining including particle second strikes: A level set methodology, *J. Mater. Process. Technol.*, accepted January 2012.