Numerical Modelling of Fretting Wear and Debris Particles

Submitted by Pengyuan Lu to the University of Exeter
as a thesis for the degree of
Doctor of Philosophy in Engineering
In October 2018

I certify that all material in this thesis which is not my own work has been identified and that no material has previously been submitted and approved for the award of a degree by this or any other University.

Signature: ……..

College of Engineering, Mathematics and Physical Science

Abstract

Fretting wear has been observed for decades and can be found in structures or machines with contacting surfaces which experience oscillatory motion. Debris particles generated during fretting wear was recently known to be one critically important factor affecting the progress of fretting wear. Correctly predicting fretting wear inside structures can be very helpful to avoid structural failure and reduce maintenance cost. However, the current method to predict fretting wear relies on empirically measured coefficients and does not consider the debris particles. This PhD thesis investigates the effect of the debris particles via a hybrid Finite-Discrete Element Modelling (FDEM) method developed in this work. The hybrid method takes the advantages of Finite Element (FE) method and Discrete Element (DE) method by simulating large solid bodies/continuum material in FE while simulating discrete bodies/granular media in DE. Thus, the debris particles can be included in the simulation. Via the FDEM model, the effect of debris particles is studied via comparing the numerical results obtained from the models with and without debris particles. The introduction of debris particles is found to affect the contact between surfaces via changing the local contact force especially reducing the maximum force value. The physical mechanism of fretting wear is also explored in this PhD thesis by implementing seven different material evolution criteria including the most popular one: the Dissipated Energy method by Fouvry. It was found that the Dissipated Energy method is likely to agree with the experimental data at the cycle where the empirical coefficient is calculated but disagree before and after this point. The shear stress evolution criterion uses a damage indicator to reflect applied shear stress as a fraction of ultimate shear stress, and as such is a threshold method. Compared to the other criteria, it predicts fretting wear well vs experimental results.

Acknowledgements

Foremost, I would like to thank Professor Christopher Smith and Professor Ken Evans for their guidance, patience, and continuous support throughout this fascinating project. They gave me the opportunity to engage with the various aspects of academic research, helping me to acquire excellent technical knowledge.

I am also deeply grateful for the help and advice provided by Professor David Nowell from the Imperial College London and Professor David Hills from the University of Oxford on implementing the numerical model. Their advice offered unique insights to the project.

I extend special thanks to Neeraj Cherukunnath and Stephen Pattison from Rolls-Royce plc for their very helpful advice and discussion throughout the project. There are many others I would like to thank for their help and support during my project, but cannot be mentioned. Thank you all.

I am grateful for the financial support of Rolls-Royce plc for this research project. I would like to deeply thank my parents who have supported me through my studies and life. I also want to thank my colleague Luke Blades and Florent Vandekerkoeve for their cheerfulness, wisdom and fun. Finally, I would like to thank those people and friends who I meet and acquire during this time for their company and kind support.

Contents

List of Figures

Chapter 1

Introduction

1.1 Context and Motivation

According to a report by Jost, approximately 1.4% of the GNP of the UK could be saved if a proper method or way could be applied to avoid or limit the effect of wear [1]. Wear is a dynamic process which indicates material failure or material loss from an operating body. Such phenomenon can be found in almost all moving parts and many fixed joints. Fretting wear is a subcategory of wear caused by fretting or small reciprocating motion between two contacting surfaces [2]. Typically, 'fretting' is defined as a motion with a relative displacement of less than $100 - 300\mu m$ [3]–[5]. Small amplitude reciprocating motion between bodies, i.e. fretting, is widespread in structures or machines with contacting surfaces which experience oscillatory motion (whether tangential, normal, or rotatory) or vibration [3], [6]–[8]. As a result of this small amplitude, often reciprocating motion wear can take place – termed fretting wear. The rate of fretting wear in industrial applications can grow exponentially as the dimension of the components changes. Those joints or interfaces under threat of fretting wear can be damaged and further cause structural failure or even an industrial disaster.

A typical example of fretting wear can be found in a heat transport system used in the nuclear industry. Inside the heat transport system, vibrations caused by steam flows could induce relative motion between the components inside the system, e.g. fuel tubes and their supports. As a consequence, fretting wear can occur between those components and damage the whole system if proper maintenance is not applied. Another example of fretting wear can be found in the spline couplings inside a jet engine via which torque forces from a transmission shaft are transferred into turbofans. Because a jet engine can rotate at a very high speed, e.g. 14,950 RPM[9], extremely large torque and moment can be generated and exerted upon the spline couplings. Thus, large stress and cyclic misalignment occur between contacting surfaces inside the spline couplings, which can lead to severe fretting wear or structural failure at those components. Due to the significant importance of those components during the operation of an engine, material or structure failure can result in the reduction of operational

efficiency or even the breakdown of the engine. However, maintaining those engineering structures requires great labour. Monitoring the health of each interface is extremely low efficient and expensive because those engineering structures can be extremely complicated: e.g. a Trent 900 turbofan engine is made up of over 20,000 components and includes a huge number of interfaces. Thus, if the mechanism of fretting wear could be properly understood and predicted, then a large number of expenses in maintenance could be reduced.

The Archard equation, proposed by Archard in 1953, is one of the most popular numerical methods used nowadays to calculate sliding wear given the load applied, sliding distance and material hardness[10]. Normally, fretting wear is treated just like any other wear phenomena. Hence, it is not surprising that the Archard equation, which is generally useful for describing sliding wear, has been applied to calculate material removal rates during fretting wear, unfortunately with limited success [11]. An alternative method proposed by Fouvry et al. [12], termed Dissipated Energy method, is becoming popular recently. The Dissipated Energy method calculates the wear volume via measuring the total energy dissipated during fretting wear. Both the Archard equation and the Dissipated Energy method provide an empirical description of fretting wear without addressing the physical mechanism of material failure during fretting wear. Other researchers have tried to explain the mechanism of fretting wear using various considerations like pressure, slip displacement or rotational angle, temperature[13], oxidation[14], debris etc.[2], [15], [16]. However, until now, there is still no single widely accepted and successful methodology to describe and predict fretting wear. Compared with sliding wear, fretting wear is a much more complex procedure which involves a number of affecting factors from mechanical ones such as contact pressure, to chemical factors such as oxidation or nitridation [17]. Among those factors, the effect of debris particles which are known to be critically important during fretting wear [18], e.g. Hurricks proposed the link between the nonlinear behaviour of fretting wear and the debris particles [2]. But, the effect of debris particles has not been properly considered inside a prediction model.

1.2 Aim

The aim of this thesis is to present a novel numerical model named Finite-Discrete Element Modelling (FDEM) which is developed to include the effect of debris particles in a model. With the novel model, the behaviour of debris particles

in Hertzian contact during fretting wear can be modelled. The influence of debris particles on the contact between solid bodies is studied. Additionally, the physical mechanism of fretting wear is to be explored via applying different material evolution criteria in a numerical model. This thesis is structured as follows.

1.3 Thesis Scope

Chapter 2 will review the necessary background information of fretting wear with a state of the art in methodologies of predicting fretting wear. Several key numerical prediction methods will be discussed such as empirical analysis method, wear-mechanism map method, and numerical modelling method.

In Chapter 3, after reviewing the applications of Finite Element Analysis (FEA) and Discrete Element Modelling (DEM) in fretting wear prediction, the development of the FDEM method will be introduced. The introduction will be structured following the sequence: programme structure, model setup, and operation procedure.

Numerical modelling becomes very computational expensive with the increasing size of the model. Such problem becomes worse when modelling fretting wear as simulating a large number of fretting cycles is also required. A methodology for reducing computational expenses will be introduced in Chapter 4 after reviewing different general cycle-jumping techniques.

In Chapter 5, with the FDEM model, the effect and behaviour of debris particles in Hertzian contact will be investigated. The Dissipated Energy method will be applied in this section as the material evolution criterion. The predicted result via the Dissipated Energy Method will also be discussed.

In Chapter 6, the physical mechanism of fretting wear will be explored through applying different material evolution criteria. Different fretting wear predicted by the model will be then compared to data from experiments via which the mechanism of fretting wear will be explored.

Finally, the conclusion and recommendations for future work will be discussed in Chapter 7.

Chapter 2

Background and Literature Review

2.1 Background of Fretting Wear

Fretting wear has been observed for decades, a typical definition being “the progressive loss of material from the operating surface of a body as a result of relative motion at its surface” [19]. The material removed from one or both surfaces in contact (the ‘first bodies’) is usually observed as small particles, and these often conglomerate together to form a plaque (the ‘third body’) [20]. In practice, fretting wear occurs in almost all machinery with moving parts, and the loss of material eventually causes impaired performance, loss of function, and reducing the working life of machine [21]. Attempts to understand fretting wear involve both explorations of the mechanisms responsible for the removal of material and observations of the general effect of varying experimental parameters [22], [23]. For decades, a great number of experiments of fretting wear focusing on different parameters have been conducted by researchers in order to understand wear. However, a full knowledge about fretting wear has not be obtained because of the large number of interdependent variables included in the fretting wear phenomenon, such as normal load, sliding amplitude, sliding speed, properties of contacting materials, and atmosphere condition. To deal with the complexity of fretting wear, several classifications based on different wear variables were developed which include such as adhesive wear, abrasive wear, oxidation wear, mild and severe wear. The degree to which wear scenarios varies significantly between different studies and authors. Table 2.1 shown below is a summary of the possible classification of wear parameters. In Table 2.1, different criteria has been used in history to describe different kinds of wear observed in reality. Thus, it can be found that wear is complicated behaviour.

Table 2.1. Comparison of various wear classifications[24].

Class	Parameter					
Friction type	Rolling	Rolling–sliding	Sliding	Fretting	Impact	
Contact shape	Sphere/ sphere	Cylinder/ cylinder	Flat/ flat	Sphere/ flat	Cylinder/ flat	Punch/ Flat
Contact pressure level	Elastic		Elasto-plastic		Plastic	
Sliding speed or loading speed	Low		Medium		High	
Flash temperature	Low		Medium		High	
Mating contact material	Same	Harder	Softer	Compatible	Incompatible	
Environment	Vacuum	Gas	Liquid	Slurry		
Contact cycle	Low (single)		Medium		High	
Contact distance	Short		Medium		Long	
Phase of wear	Solid	Liquid	Gas	Atom	Ion	
Structure of wear particle	Original		Mechanically mixed		Tribochemically formed	
Freedom of wear particle	Free	Trapped	Embedded	Agglomerated		
Unit size of wear	mm scale		μm scale		nm scale	
Elemental physics and chemistry in wear	Physical adsorption, chemical adsorption, tribochemical activation and tribofilm formation, oxidation and delamination, oxidation and dissolution, oxidation and gas formation, phase transition, recrystallization, crack nucleation and propagation, adhesive transfer and retransfer					
Elemental system dynamics related to wear	Vertical vibration	Horizontal vibration	Self-excited vibration	Harmonic vibration	Stick-slip motion	
Dominant wear process	Fracture (ductile or brittle)	Plastic flow	Melt flow	Dissolution	Oxidation	Evaporation
Wear mode	Abrasive	Adhesive	Flow	Fatigue	Corrosive	Melt	Diffusive
Wear type	Mechanical		Chemical		Thermal	

The most relevant classification to this review is sliding wear and fretting wear. Such classification is made according to relation to the magnitude of the motion between the contacting bodies relative to the dimensions of the contacting zone. If the relative motion of the contacting surfaces is either unidirectional or multidirectional and reciprocating with a large amplitude, then the contact is considered to undergo sliding wear. Otherwise, if the motion of the two surfaces prior to a reversal is relatively small, then it is usually characterised as ‘fretting’, and any resulting wear as ‘fretting wear’. There is no strict distinction between reciprocating sliding wear and fretting wear, but one significant difference between those two rises from the ease with which the wear debris can escape from the contacting zone [25], [26]. The relative motion between two first bodies of interest is normally tangential to the contacting surfaces, but in some cases, it can be normal or even rotational to the surfaces as shown in Figure 2.1. In Figure 2.1, contact occurs between a sphere and a place. The F_n stands for the normal loading force and the arrow represents the direction of fretting motion. The tangential fretting motion, as the simplest one compared to the other ones, is usually the one to be studied mostly. Thus, the majority of the fretting studies found in the literature concern reciprocating motion in the plane of the contact surface, known as tangential fretting. The following section will provide an overview of the state of the art of numerical fretting prediction methods developed to predict tangential fretting wear.

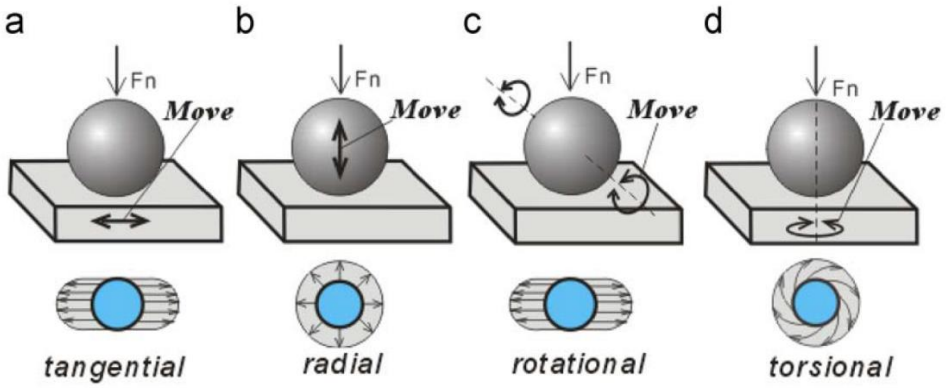


Figure 2.1. Four commonly defined modes of fretting wear, and the motions of the contacting surfaces[27].

2.2 Mechanism of Fretting Wear

Compared to the knowledge about sliding wear, limited understanding has been obtained for fretting wear. In the work by Hurricks [2], the fretting wear process is divided into three different stages from initial metal contact and transfer, to the production of debris particles, and finally to steady-state wear. In the first stage, two metals contact each other and the debris particles are about to detach from the surfaces due to material degradation. Most metals are likely covered in a thin layer of oxide which is worn away quickly and results in an initial period of wear. During the second stage, debris particles are generated continuously either from the surface or from the “virgin debris” (large metal debris particles). At the same time, part of those debris particles are then ejected out of the contact zone until a steady stage is met, i.e. the generation rate of debris is equal to the escape rate of debris. During this stage, the influence of debris particles reaches a steady stage which results in the final stage which is an equilibrium state of fretting wear. Other factors such as slip displacement, rotational angle and temperature affect the procedure of oxidation, nitridation, lamination, delamination, and ejection rate of debris particles. As a result, they influence the fretting wear in the stage of debris particles generation and of course the steady stage later. Godet et al.[28]–[31] explained the mechanism of generating debris (in the first stage) by adhesion and abrasion wear due to microstructure, i.e. asperity. Varenberg et al. proposed that the role of oxide debris particles depends on the dominant wear mechanism: debris particles act like a lubricant reducing wear rates when adhesive wear is dominant but increase it during abrasive wear [32].

In the paper by Meng et al. [11], it was pointed that the Archard equation works well for sliding wear but less so for fretting wear: the reciprocating

displacements result in wear rates do not match the predictions made using the Archard equation. The downfall of the Archard equation is that, due to reciprocating displacement, debris particles generated during fretting wear are trapped in the contact zone and then influence the subsequent wear process [33]. The function of the debris particles trapped in the contacting zone is so critical to the wear rate, that it impedes the establishment of the wear coefficient which is crucial for wear prediction via the Archard equation [33].

2.3 Fretting Wear Prediction and the Archard Equation

Since the middle of the twentieth century, attempts have been made to develop a theoretical basis to explain and predict wear [34]. In a number of papers, various methods and mathematical models have been proposed to calculate the fretting wear, e.g. wear-mechanism map and the Archard equation.

2.3.1 Wear-mechanism Map

Wear-mechanism map is a technique used to present wear data systematically according to a hierarchy of parameter so as to define a wear system[35]. Fretting wear is not a single property of a material: i.e. different mechanisms could be found such as adhesion, abrasion, fatigue and corrosion during the procedure of wear [35]. A wear-mechanism map is able to provide a multi-dimensional graphical presentation of wear data which reflects the multi-factors included during wear [36]. Thus, a wear-mechanism map can be helpful in understanding the effect of a specific physical factor [37]. The dependent variable inside a wear-mechanism map can be wear-data or wear-mechanism, while the independent variables could be material properties of the first bodies like Young's' modulus or experimental setup parameters like displacement, load and time. A wear-mechanism map can be two or three dimensional. The majority of wear-mechanism maps are two-dimensional diagrams which partition the whole graphical domain into several sections each of which represents a kind of wear mechanism. A three-dimensional map can display more information than a two-dimensional map but take much more effort to build one. Building a wear-mechanism map requires massive experimental data with various testing conditions [36]. A great effort has been made by researchers represented by Lim, Hsu, and their co-authors [36], [38]–[41] who provided wear-mechanism maps for different material under different conditions.

In the work by Vingsbo and Soderberg [42], a wear-mechanism map was used to explore the contact conditions of fretting wear under different loading conditions. Such wear map can be called running condition fretting map (i.e. RCFM) [37]. In this work, a series of fretting wear experiments with different tangential force and normal loading force were conducted [42]. An RCFM about the characteristic of contact under different combinations of normal load and tangential force was generated as presented in Figure 2.2 [42]. Inside this RCFM, based on different values of normal force (N) and tangential force (T), three different regimes were found each of which was characterised by different contact conditions in fretting wear, i.e. stick, mixed stick and slip, and gross slip [42]. With the help of RCFM, Vingsbo and Soderberg pointed out the link between contact conditions and wear mechanisms: mix stick and slip regime leads to material failure through fretting fatigue and gross slip regime through fretting wear [42]. A similar idea was also stated in the work by Fouvry et al.[43].

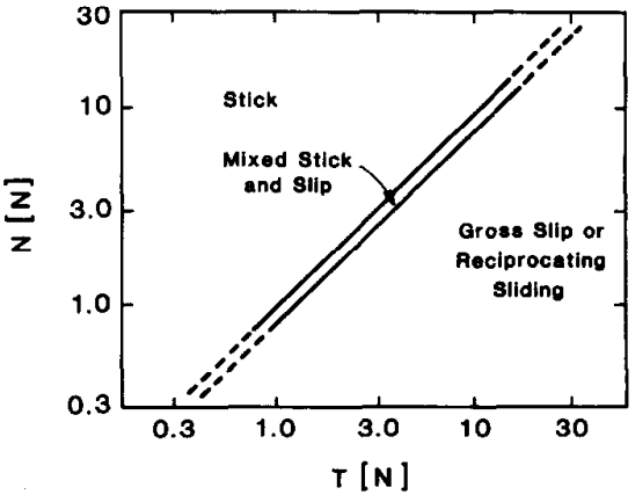


Figure 2.2. A fretting wear map of a normal force N vs. tangential force T[42].

In the work by Zhou et al., the material behaviour during fretting wear under different normal loads and slip amplitudes was investigated[37]. In this work, a material response fretting map (MRFM) was created after conducting a number of tests with different normal loads (F_n) and slip amplitudes (D) as shown in Figure 2.3: three different regimes of material response were found: no degradation (ND), cracking (C), and particle detachment (PD)[44]. In the recent work by Fouvry et al. [45], an MRFM was generated about differently shaped wear scars obtained under different fretting frequencies and force ratio as presented in Figure 2.4. The force ratio (R_P) was calcualted as the maximum normal force P_{max} over the minimum one P_{min} during the fretting wear tests with sinusoidal progression of the normal force [46]. As displayed in the MRFM, it clearly indicated the effect of

frequency (f) on fretting wear profile, i.e. a “W” or “U” shape[46]. Wear-mechanism map could also be helpful for validating the predicted results using mathematical models. In the work by Garcin et al., an MRFM fretting map was generated with results from a numerical model and compared to results from literature [45].

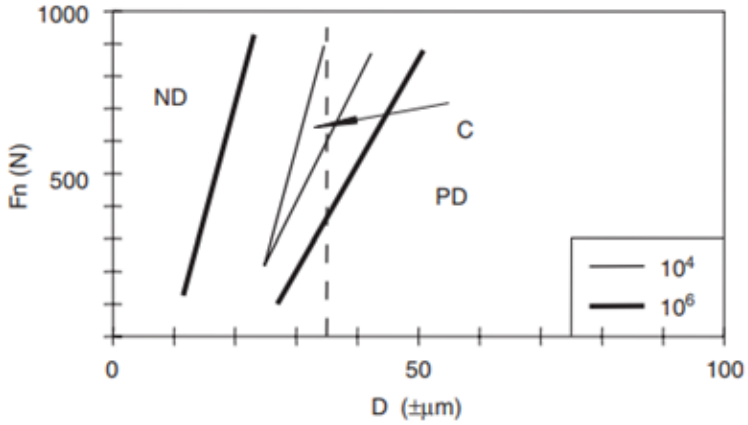


Figure 2.3. A material response fretting map including no degradation (ND) or slight degradation, cracking (C) and detachment particles (DP)[37], [44], [47]. *The horizontal axis represents the slip amplitude while the vertical axis stands for the normal load applied during fretting wear tests.*

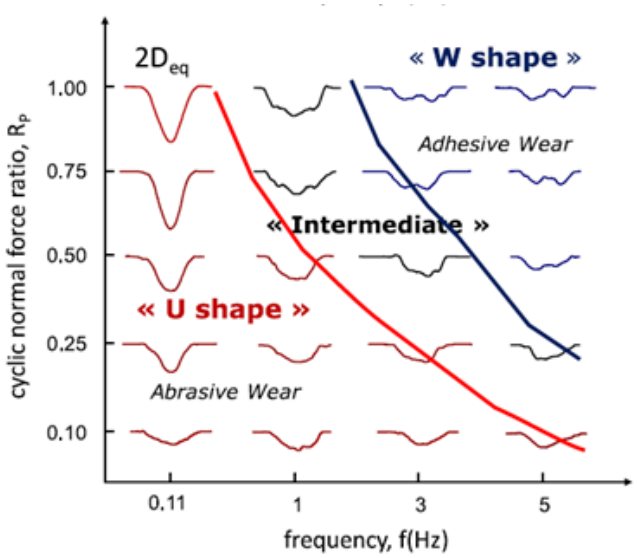


Figure 2.4. A material response fretting wear about 2D surface profiles obtained under different fretting conditions[46]. *The horizontal axis is the frequency of fretting motion while the vertical axis is the ratio of the minimum and maximum normal force during tests:* $R_P = P_{min}/P_{max}$.

In essence, wear-mechanism map could be a good choice when identifying the procedure of wear under a prior known condition [24]. It could also be very useful in roughly estimating the wear mechanism or wear data if the given operation condition falls within the region inside the wear-mechanism map [48]. One of the main limits of wear-mechanism map is that it loses its application when the operation condition is not included or described by the map. In addition, because fretting wear is not a single property of the material, any wear data could

be obtained if one of many variables of the system is changed [35]. Thus, the relationship between two factors observed inside a wear-mechanism map may not indicate a true relation in reality. Thus, although wear map could not provide a full description of fretting wear, it assists characterising of fretting wear from a certain perspective and help understanding the effect of a specific physical factor [37]. More examples of wear map about fretting wear can be found in the following papers[39], [49]–[51].

2.3.2 Some Early Mathematical Wear Models

Wear-mechanism map is useful within its known space. But in order to make a prediction outside the boundaries of known data, a mathematical description is referred especially when the accuracy of prediction is required [40]. A number of mathematical models have been proposed for wear prediction. Those mathematical wear models were usually developed for particular wear mechanisms. No equation was found to able to cover all kind of wear[11]. It is impossible to list all the wear models here but they could be categorised into two groups: 1) empirical equations which were derived from experimental results; 2) wear models which are based on material failure mechanisms [11].

An empirical equation of wear model describes a relation between two or several parameters mathematically without demonstrating a real physical mechanism. Similar to the wear-mechanism map, an empirical equation is only valid within a certain range of the test data or for a particular type of wear mechanism, but with great risks of predicting with large error beyond their ranges. Some examples of the empirical equation of wear model developed in history are given as shown below:

$$V = \frac{\beta}{\alpha}(1 - e^{-\alpha t}) \quad (2.1)$$

$$V = \alpha t \quad (2.2)$$

$$V = \beta e^{\alpha t} \quad (2.3)$$

$$\Delta W = K F^{a} V^{b} t^{c} \quad (2.4)$$

Where α, β, K, a, b, and c are experimental terms or constants. V, t, W, F represent the volume loss, time, weight loss, and applied force respectively [52], [53]. Another typical and more popular example is the Archard equation which will be further discussed in the subsequent section.

The second type of mathematical wear model can be understood as a material evolution criteria or threshold beyond which material will be removed from the surface. A good example could be the material evolution criteria proposed by Bayer et al. which emphasis on a material evolution criteria about critical shear

stress ($\tau_{critical}$) during a ball-plate fretting wear experiment [54]. It was found that the value of critical shear stress varies with the material properties, lubricants used and lifetime [54]. Another example of this kind mathematical wear model can be found in the paper by Hornbogen [55] which proposed a critical strain value.

2.3.3 Archard Equation

The Archard equation is an empirical mathematical wear model and one of the most popular models used to predict fretting wear nowadays. The theory behind the Archard equation was derived from experimental results and analytically by Archard et al. [10], [34]. A series of sensitivity tests about sliding wear contact was run by Archard for different velocities, contact surfaces, materials and so on. Eliminating the irrelevant factors, Archard identified three key variables with which wear rate varies linearly in general. An empirical relation was found between a worn volume and normal load applied, sliding distance, and material hardness. The Archard equation is given as:

$$Q = k\frac{WL}{H} \tag{2.5}$$

where W is the normal load, L stands for the sliding distance or relative motion between two contacting first bodies, and H is the hardness of the material in contact. k is known as the 'wear coefficient' and is specific to the contact conditions. However, the simplicity of the Archard equation in contrast to more physically derived theories has been criticised: the empirical theory does not provide any insight into the mechanism of wear of metals under different conditions; many of the assumptions employed in this mathematical model are unreasonable and arbitrary [56] particularly ignoring the fretting wear debris which is known to be critically important in fretting wear [18]. In spite of these criticisms, the Archard equation is the most commonly applied model when estimating the volume of material transferred in a fretting wear problem.

2.4 Finite Element Analysis with the Archard Equation

Nowadays, with the increasing calculation capability of computers, most of the engineering problems could be simulated and solved by a computational software like Finite Element Analysis (FEA). Inside FEA, engineering problems are divided into small parts or elements to be analysed at a local scale which therefore reduces the complexity of the global scale. Modelling fretting wear inside FEA is

analysed through discretising the whole fretting motion into a number of steps and both first bodies into a number of elements. Then, the fretting wear rate per step can be calculated for each element. Most studies on simulating fretting wear with FEA are based on the previous model of reciprocating sliding wear, i.e. the Archard equation[57]. It should be noted that the effect of wear debris is usually neglected in most research applying the Archard equation [58].

McColl et al. first used FEA to simulate fretting wear using a localised Archard equation to calculate wear rate[59]. The principle of the localised Archard equation can be described as:

$$H(x,t) = K \times P(x,t) \times \delta(x,t) \quad (2.6)$$

where $H(x,t)$ is the local wear depth calculated at an element at location x and time t. Similarly, $P(x,t)$ and $\delta(x,t)$ represent the loading force and relative slip displacement at certain location x and time t. The K stands for the wear coefficient which varies for different materials and operation conditions. Such calculation will be conducted at every node of the contacting zone where relative motion is detected. The above procedure can be briefly demonstrated in Figure 2.5. As shown in Figure 2.5, before launching the simulation, a fretting wear model is created by defining different properties including the dimension of the first bodies, the material properties, the loading force/pressure, the motion amplitude, the number of cycles and so on. Then, required by the FEA software, a master-slave surface between two contact surfaces is used to enforce a contact constraint between two first bodies [59]. A master surface is usually assigned to the moving surface and the other one to be slave [60]. The whole model is then discretised by creating nodes and elements, such procedure can be automatically achieved by most FEA software[60]. However, the mesh automatically generated may not be the most adequate to accurately reflect reality. Much finer elements are required at the region where the material is more sensitive and needs more detailed analysis [59]. Based on the nature of FEA, finer mesh elements produce better simulation results and also require more computational analysing capability. In order to save the computational cost, the part of interest in the model is usually meshed with fine mesh elements while leaving the rest with coarse mesh as shown in the Figure 2.6. Such setup is usually ascertained via a convergence test in which a target parameter is required to be converged with minimum calculation cost.

During the FE simulation, through measuring the penetration between elements, the FEA software calculates material responses such as nodal contact load $P(x,t)$, and nodal slip distance $\delta(x,t)$ for each node at each step based on the properties initially assigned. These data are then fed back to an inbuilt code (i.e. UMESHMOTION) to calculate the corresponding wear based on Equation 2.6. Then, the wear mechanism or material removal procedure inside FEA is achieved by moving the worn node downwards inside the model. After updating the geometry, the subsequent fretting cycle is launched unless the total number of cycles has been reached. It should be noticed that, even though the local wear coefficient k_i used in this method is assumed to be the same as global wear coefficient[59], there is an increasing number of researches has found an inequality between the local and global wear coefficient[61]–[64]. This methodology has also been extended into different applications like fretting wear between coated materials, wear in steel wires, and wear in heat exchanger due to fuel vibration in a nuclear power station [58], [64]–[67]. More detailed information about this method can be further found in the literature [58], [68]–[73].

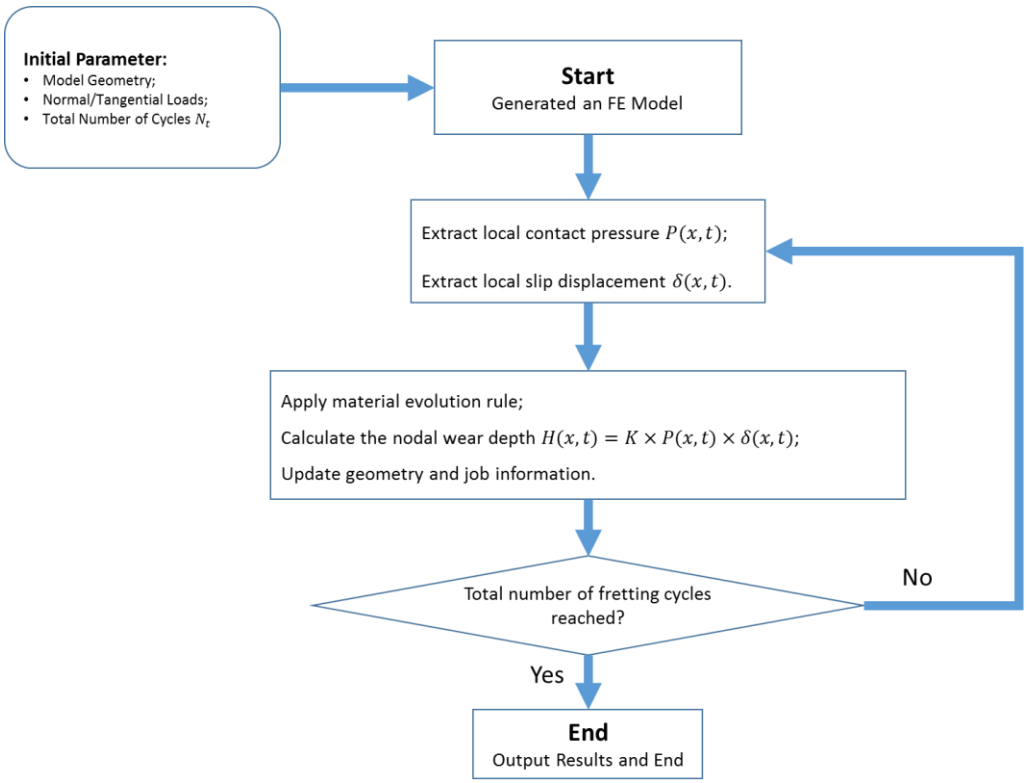


Figure 2.5. Brief flowchart of Finite Element Analysis with the Archard Equation

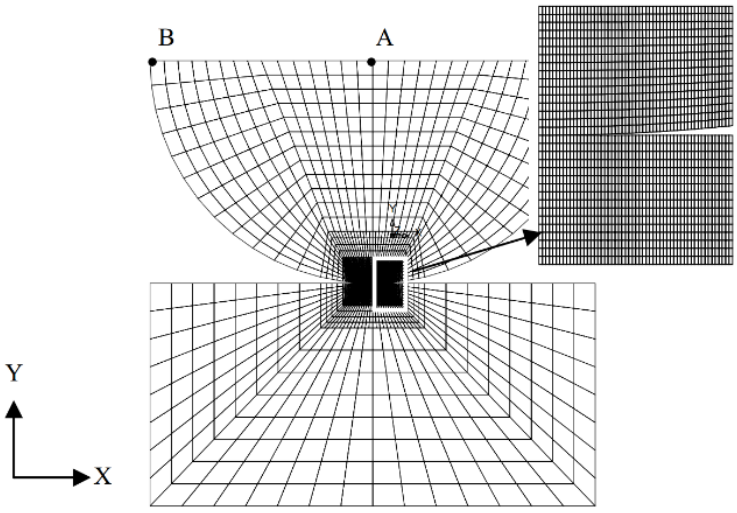


Figure 2.6. Finite Element Model for Cylinder-on-flat Configuration[59]

Following the work by McColl et al., other researchers have tried to modify the method by adding more physical features into the model especially regarding the third body as it is now acknowledged as a driving factor that cannot be neglected in fretting wear investigation. In the work by Ding et al., the third body was simulated as an oxidation-debris layer by adding an individual layer structure between two contact first bodies as shown in Figure 2.7 [18], [74]. That layer was defined with the material properties and dimensions of a debris layer measured from the experiments at the time when it is first observed [74]–[76]. The thickness of the third body or debris layer (Q_3) was increased by new debris particles forming from the first bodies (Q_1 and Q_2), and subtracted by the loss of particles escaping from the contact zone. Such procedure was controlled by an a priori wear coefficient defined from experimental data. Fouvry et al. has questioned whether assuming constant wear coefficients in the Archard equation is the cause of the model inaccuracy [61], [77]. This third body structure functioned as experimentally observed by acting as a lubricant limiting wear rate and was shown to produce a closer prediction for fretting wear, i.e. wear depth, compared to the model that did not include the third body. This difference might stem from the alteration of the contact mode from point-plate into plate-plate via the third body layer between the two first bodies. Basseville et al. simulated the fretting wear with the third body in a more aggressive way: simulated the third body explicitly using finely meshed rectangular particles evenly distributed within the contact zone [78]. Due to the high computational cost of this method, only a very small fraction of the real number of particles, i.e.12, could be simulated in the model.

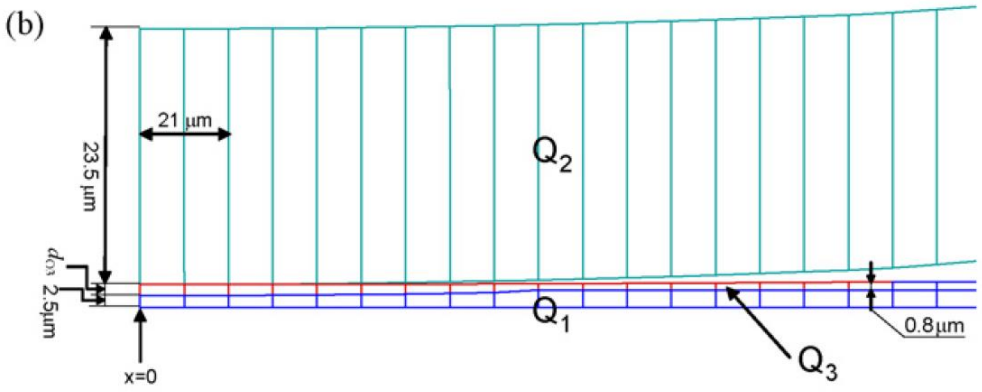


Figure 2.7. A close-up view of the debris layer (Q_3) between discretized first bodies (Q_1 and Q_2)[18].

On one hand, the FEA fretting wear model successfully analyses the interaction between first bodies. On the other hand, the FEA can only model the third body as a layered plate or a very limited number of particles due to the nature of FEA code [18], [74]–[76], [79]–[81].

2.5 Finite Element Analysis with Dissipated Energy Method

To account for the shortcomings of the previously reviewed model, Fouvry and co-workers have tried to describe the fretting wear from a different aspect. One of the main drawbacks of the Archard equation is assuming a constant wear coefficient whereas it actually changes over time which results in poor predictions [18]. Fouvry et al. proposed that the wear coefficient increased critically when the elastic "shakedown boundary" was crossed due to metal hardening [59], [61], [62], [82]. Shakedown boundary was first analysed by Johnson [83] and Kapoor[84] saying the material properties change, e.g. material strength as the internal property while the coefficient of friction as the external contacting property, due to accumulated plastic dissipation under loading condition.

In an attempt to get away from concepts of constant wear coefficients, Fouvry et al. proposed a new description of fretting wear through the energy dissipated during fretting wear in 1994 [85]. A positive linear relationship was found between the dissipated energy and material removed during fretting wear[12], [86]. The theory proposed by Fouvry et al., named Dissipated Energy method, links the wear volume and the dissipated energy by a coefficient named the energy wear volume coefficient (α_v , $\mu m^3 J^{-1}$)[12]. The energy wear volume coefficient describes the amount of material removed by a unit amount of dissipated energy[87]. The principle of Dissipated Energy method can be presented as:

$$W_v = \alpha_v \sum Ed \quad (2.7)$$

where the W_v is the total wear volume deduced from fretting wear when the amount of energy $\sum Ed$ is dissipated during the fretting wear. Like the wear coefficient K in the Archard equation, the energy wear volume coefficient α_v is not a constant value but varies based on materials and operation conditions [46]. Fouvry et al. also introduced the method to ascertain the averaged value of energy wear coefficient through experiments as displayed in Equation 2.8:

$$a_{average} = \frac{V^w}{{E_d}^{tot}} \quad (2.8)$$

where $V^w$is the total wear volume measured at the certain stage of fretting wear, ${E_d}^{tot}$ is the energy dissipated up to that stage which is calculated as the product of tangential force and fretting amplitude. Details can be found in papers [61], [88], [89]. As noted by Pearson et al.[90], there was a close agreement between the Archard and Fouvry methods, which is due to the similarity between the Archard wear coefficient (with a unit of $m^3 m^{-1} N^{-1}$) and the energy-based wear coefficient (with a unit of $m^3 J^{-1}$).

When being implemented in the FEA, the Dissipated Energy method is applied locally to every node where relative motion is detected and wear is caused. The calculation conducted inside the FEA could be described as:

$$H(x,t) = \alpha_v \times Q(x,t) \times \delta(x,t) \quad (2.9)$$

where $H(x,t)$ is the nodal wear depth at give traction force $Q(x,t)$ and relative slip displacement $\delta(x,t)$ at certain location x and time t. The schematically frame of implementing the Dissipated Energy method is displayed in Figure 2.8. As shown in Figure 2.8, the procedure is similar to the one introduced in Section 2.4 but different in the wear equation.

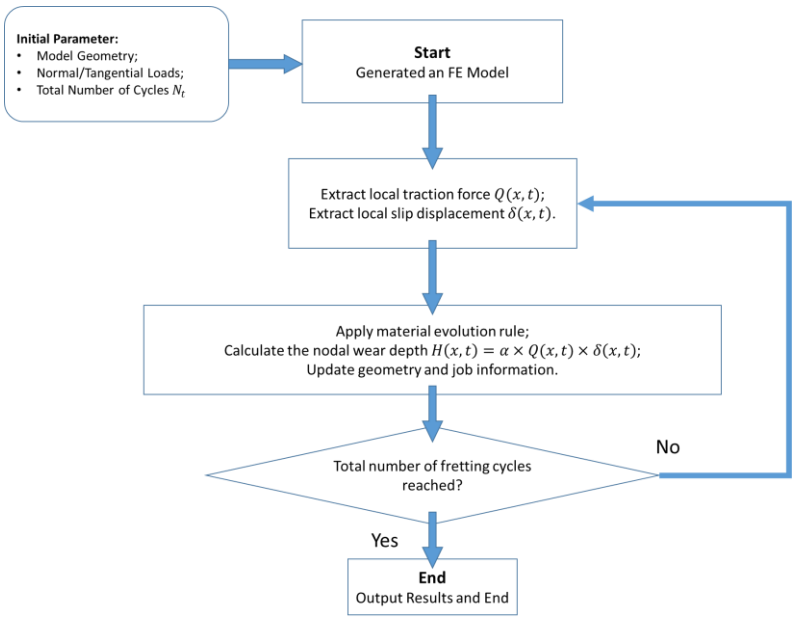


Figure 2.8. Brief flowchart of Finite Element Analysis with Dissipated Energy Method

In the later work by Fouvry et al., a tribology transformed structure (TTS) was used to describe the function of the third body in the fretting wear[61], [91]–[93]. It was proposed that a TTS generated in the early stage of fretting wear nonlinearly affects the fretting wear. The TTS can be equivalently described as a critical threshold of dissipated energy (E_{TTS}) as shown below [91]:

$$W_v = 0 \text{ if } \sum Ed < E_{TTS} \quad (2.10)$$

$$W_v = \alpha_{True} \sum (Ed - E_{TTS}) \text{ if } \sum Ed > E_{TTS} \quad (2.11)$$

According to Fouvry et al. a threshold of generating TTS (E_{TTS}) is used: below that, the total dissipated energy ($\sum Ed$) is transferred into metal plastic transformation and the wear volume W_v equals to zero; at the threshold, TTS is suddenly generated which bear the loading and protect the material beneath from getting worn; above the threshold, the excess dissipated energy starts to wear the material[91], [93]. It should be noticed that, comparing the Equation 2.9 and 2.11, it can be found that the calculated energy wear coefficient from Equation 2.9 using the method by Fouvry et al. is an averaged energy wear coefficient $a_{average}$ which is different from the true energy wear coefficient a_{True} supposed to be used in the Equation 2.11 [90].

2.6 Finite Element Analysis with Other Evolution Criteria

Apart from the Archard equation and Dissipated Energy method, other fretting wear models or material evolution criteria have been proposed and investigated. Failure theory is usually used to describe the material status at a certain stage.

The material is regarded to be failed when the material status satisfies certain conditions. Compared to an empirical failure method like the Archard Equation, the Failure theory is able to demonstrate more material physical properties. Failure theory is expressed in the various forms of failure criteria which are valid for different material (e.g. fracture for brittle materials while yielding for ductile materials) or specific condition. Usually, the failure criteria are expressed in a function of stress or strain which separates “failed” state from “unfailed” state. The failure of material could be observed in different scales, from microscopic to macroscopic. Microscopic material failure is defined as a material failure due to crack initiation and propagation. Macroscopic material failure is defined in terms of the load carrying capability or energy storing capability, like von Mises stress[94].

In the work by Ding et al. the material worn during fretting wear is predicted via simulating the plastic deformation of micro-asperity [74], [76], [95]. In the work by McCarthy et al., a material evolution criterion of microstructure-sensitive plasticity was used to calculate the material failure during fretting wear[96], [97]. According to McCarthy et al., fretting wear could be understood as a material removal where the equivalent plasticity reaches a threshold [96], [97]. In the work by McClintock and Rice et al. [97]–[102], a failure indicator termed ductile failure damage indicator (DFDI) was introduced which evaluated the amount of damage accumulated inside the material due to plastic strain as shown below:

$$D_{DFDI} = \int \frac{d\varepsilon_{pc}}{\varepsilon_f} \tag{2.12}$$

where ε_{pc} describes the equivalent plastic strain which is accumulated over loading cycles. The threshold value of failure strain limit ε_f was calculated as [103]:

$$\varepsilon_f = 1.65\varepsilon_0 \exp(-\frac{3\sigma_m}{2\sigma_{eq}}) \tag{2.13}$$

where σ_m is the average stress of three principal stresses, and σ_{eq} is the equivalent von Mises stress. ε_0 is the critical strain of material for an incipient crack which equals to the failure strain in a uni-axial tension test [104]. By definition, the DFDI ranges from 0 (undamaged) to 1 (damaged). Failure will occur when $D_{DFDI} \geq 1$. Inside a finite element model, the D_{DFDI} at every node is calculated every step. A brief demonstration about the implementation of material evolution criterion can be seen in Figure 2.9. The one difference of this method is using a different material evolution criteria in the flowchart below compared to

the two introduced above. The wear depth at the specific position is ascertained by the location where plasticity reaches failure strain limit (p_{crit}) as shown in Figure 2.10: $p(x, y, N) >? p_{crit}$. Besides, the plasticity value between two nodes is calculated using interpolation[96]. Detailed information can be found in the similar work [105], [106].

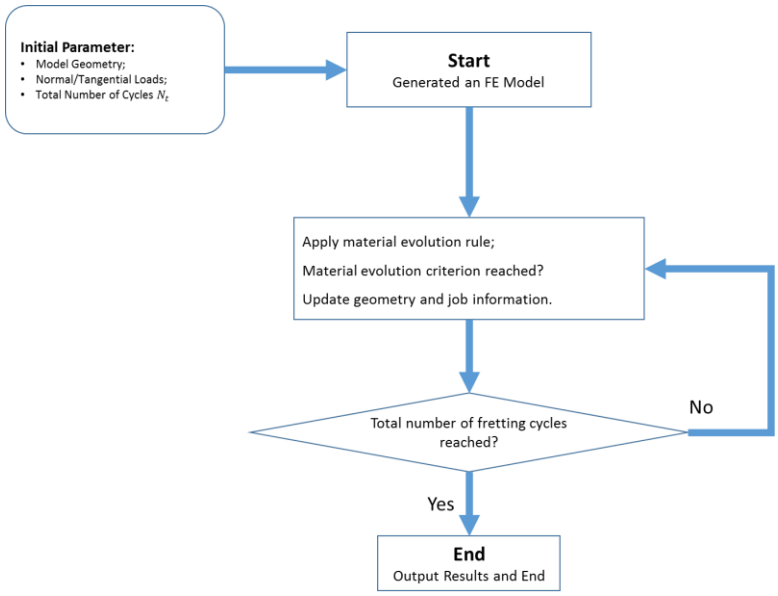


Figure 2.9. Brief flowchart of Finite Element Analysis with microstructure-sensitive plasticity failure

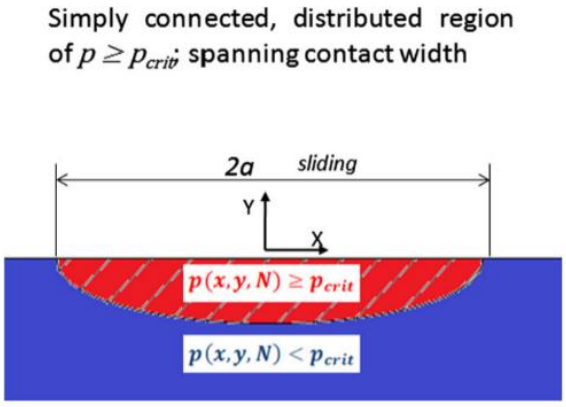


Figure 2.10. Demonstration of fretting wear failure mode with plasticity (p in the figure stands for the accumulated plastic strain at certain location)[96]

There are also other researchers trying to explain the fretting wear from different aspects. Because the procedure of fretting wear is a complex phenomenon, different mechanisms like wear, nucleation and propagation of crack (i.e. fatigue) can take place together and influence each other[107]. Ghosh et al. [108] described fretting wear as material removed due to micro-cracks generated and propagated by fatigue under fretting load. When the micro-crack joins together and reaches the surface, the material was removed and created a wear scar. In the work by Hager [109] and Fridrici with his co-authors [110], the micro-cracks or fretting cracks was observed as shown in Figure 2.11. Other

research investigating the link between fretting fatigue and wear can also be found in [45], [70], [111]–[119].

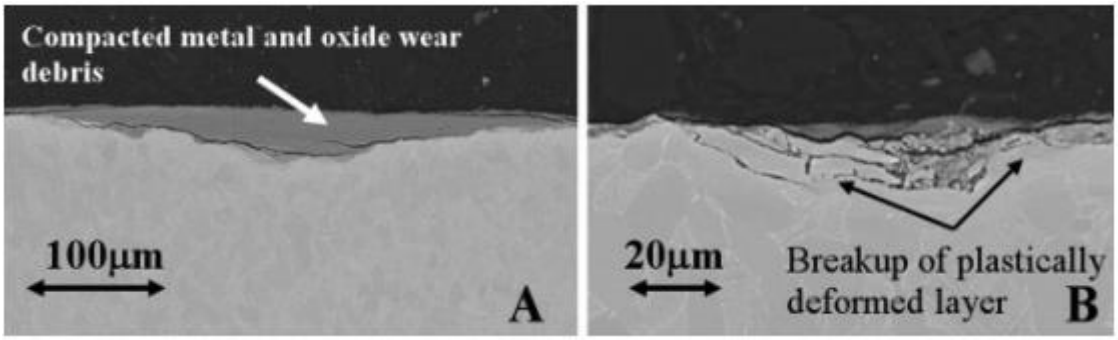


Figure 2.11. An SEM image showing the cracks observed at the centre and edge of a fretting wear scar respectively[109], [110]

When geometries are experiencing fretting wear specifically tangentially reciprocating relative motion, damage inside the material is cumulated along certain direction until the material is worn. During the procedure of material fatigue, crack initiation and crack propagation or growth are two main stages which are controlled by Mode II and Mode III fracture mechanics individually. According to the work by Plumbridge[120], [121], during multiaxial loading, the shear mechanism initiates the cracks in the slip plane on crystallographic planes. The tensile mechanism orthogonal to the shear plane then speeds up the growth of cracks. The plane on which the maximum damage is applied to the material and along which cracks initiate is called a critical plane. The Critical Plane Approach is able to evaluate the damage of material by identifying the critical plane. The damage or material is then evaluated via calculating the number of cycles to failure (N_f): material is fully damaged when the N_f is reached. The Critical-Plane Approach has been widely used in analysing material failure or mainly fatigue life prediction upon materials which experiences cyclic loading history [122]–[124]. The Critical-Plane Approach was first discussed in the work by Brown and Miller, which found a correlation between fatigue life and two factors: maximum shear strain and orthogonal tensile strain along a critical plane [125]. The Critical-Plane Approach was employed by Fatemi and Socie in 1988 to analyse the multiaxial fatigue damage where material failed along a particular plane through evaluating the stresses or strain in the location of the plane[122]. According to Fatemi [122] and McDiarmid [126], a shear stress based Critical-Plane Approach would be appropriate for fatigue analysis of mode II failure mechanism (shear mode) is dominant. The Critical Plane Approach could be expressed as follows:

$$\frac{\Delta\tau_a}{\tau_f'} + \frac{\sigma_n}{2\sigma_y} = (2N_f)^b \quad (2.14)$$

Where $\Delta\tau_a$ is the shear stress amplitude, σ_n is normal stress, τ_f' is the shear fatigue strength, σ_y is the yield stress, and b is fatigue strength exponent. The orientation of the critical plane is estimated via a Maximum Variance Method (MVM) [127]. After ascertaining the direction of the critical plane, the parameters required by Equation 2.14 like the shear stress amplitude is determined using Minimum Circumscribed Circle (MCC) method[128]. The value of shear fatigue strength and fatigue strength exponent could be estimated via the method from literature [129]–[133]. Afterwards, by applying the Miner rule, the material is removed when N_f is reached. Both the MCC method and the Miner rule will be introduced with more details in Chapter 6.

Apart from the implementation of evolution rules mentioned above, it could be interesting to study the mechanism of fretting wear via investigating some basic variables like pressure, von Mises stress, and internal stress. Such methodology has mature applications in the field of experimental study where the effect of one single variable like pressure, friction force or yield stress is studied. A typical example is the work by Archard who linked the wear failure with the load and hardness. Another example is the wide application of von Mises in identifying and predicting weak part of a structure or material failure. In the work by Mi [134], Cai [27], [135], and Liskiewicz [46], [136], [137], the important role played by the internal shear stress or interfacial shear work during fretting wear was proposed. In the work by Cai et al. [135], a “W” shaped wear scar was observed in a gross fretting wear experiment where a “U” shaped scar was expected according to the classical model. This could indicate the material failure due to maximum shear stress.

2.7 Discrete Element Analysis

Third body or debris particles have been observed since the very early stages of research about fretting wear[16], [138]. The critical role of debris particles during fretting wear has also been observed however it has not been widely considered in modelling fretting wear[139], [140]. While FEA is a commonly implemented methodology of fretting wear modelling, there is not a single widely accepted method to model the evolution of debris particles during fretting wear [141]. The methods used for modelling the third body inside FEA are also questioned. Ding

et al. mentioned that wear debris particles were discontinuously distributed between the contact surfaces [18] which indicated the inadequacy of modelling the third body as one single layer of solid elements with constant material properties. The properties of the third body are not constant with the progress of fretting wear but are affected by different factors such as contact geometry, pressure, and fretting speed [46], [138]. Although it might be possible to obtain those properties of third bodies and wear coefficient via measurement, this would require experimental testing for every new case - being largely impractical and rendering prediction unnecessary. Lying beneath all of these problems is the fact that the Archard equation does not include the mechanisms by which material is removed from the surface of a first body, and relies on an empirical and linear relationship between applied pressure and wear volume. This relationship holds true empirically for sliding wear.

Discrete Element Analysis or Discrete Element Modelling (DEM) has gradually gained wider application where granular media are involved. Proposed by Cundall in 1971, DEM was primarily implemented in the field of geography like modelling jointed rocks. DEM has mature applications in fields such as granular flows[142] and lubrication[143], [144]. Each particle in DEM can be shaped as a circle in 2-dimension or a sphere in 3-dimension and can have properties including location, velocity, mass, contact stiffness, damping, adhesion, and friction for instance. Particles in complex shape can be defined by grouping several circle or sphere elements. The movements of particles in space are calculated by applying Newtonian mechanics, resolving contacts with other particles or solid boundaries.

Fillot and co-workers abandoned the method of modelling the third body with FEA and employed DEM to handle the kinematics of large numbers of solid particles in the time domain[145]–[153]. Inside work by Fillot et al., both first bodies were constituted of spheres as shown in Figure 2.12[145]–[153]. In Figure 2.12, two first bodies are presented as the green first body is non-degradable (the bonds between green particles are non-degradable) and the yellow first body in degradable (the bonds can be broken). The purple particles are those detached from the yellow first body. Fillot's model is able to capture the evolution of debris particles including detachment from a first body surface, motion within a contact zone and finally ejection, as well as behaviours under conditions such as particle-particle adhesion. It was able to predict phenomena such as detachment of debris

from the first body or debris particle adhesion into a third body, at the scale of individual particles which prior methods could not [146], [154]. As stated by Iordanoff, even though DEM is helpful to better understand the procedure of material detachment as well as the whole evolution of the third body[154], DEM may not be accurate in calculating wear volume and wear rate.

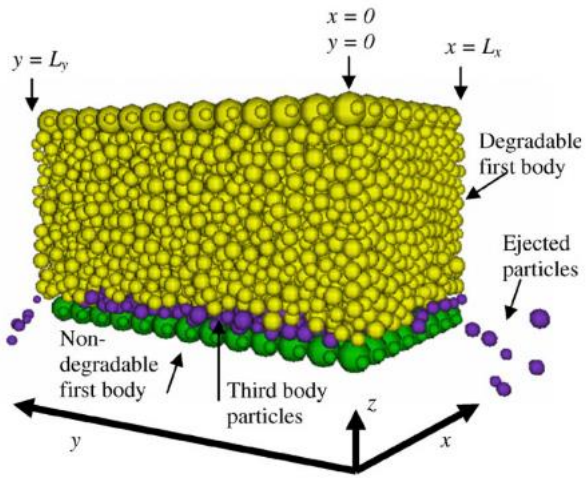


Figure 2.12. Two bulk materials modelled with discrete elements in the colour of yellow and green. The third body is detached from degradable bulk material in yellow[146].

During the simulation of DEM, geometries are defined via grouping a number of particles so as to represent the right dimension. Then, bonds or linear springs with a stiffness of k and yield tensile force F_r are formed as shown in Figure 2.13 between adjacent particles so as to form a solid body [146]. Those bonds are used to define different mutual forces including adhesion, repulsion, and shear friction forces between particles. In the work by Fillot, one first body was defined to be non-degradable whose bonds between particles have large yield strength while that in the other degradable first body would fail after reaching a comparatively lower yield strength, insuring that the former was not worn down while the latter was. When two first bodies contact each other, the interaction between first bodies is realised by calculating the interactions (δ) between particles inside two first bodies. The interaction between particles is ascertained by calculating the penetration or distance between elements as shown in Figure 2.14. A coordinate searching algorithm and space partition algorithm are used inside DEM to reduce the space around each particle within which it must perform calculations, such as detecting contact and development of forces, changed velocities and new particle locations. Closer particles lead to compression while more distant particles produce tension. When a bond reaches its yield strength, the initially bonded particles are allowed to detach, which can be expressed as:

$$F = k\bar{u} \quad if\ F\ < F_r \tag{2.15}$$

$$F = 0 \quad if\ F\ \geq F_r \tag{2.16}$$

where F is the interaction force between particles, k is the stiffness of the spring while $\bar{u}$ is a relative velocity vector between the two particles. The F_r represents the yield strength of the bond. If the force is larger than the yield strength, the bond will be broken and the corresponding element is free to move like the single purple particles shown inside the Figure 2.14. After a particle is detached from its host, its motion is fully decided by Newton's second law. The accelerations ($\ddot{x_i^t}$, $\ddot{y_i^t}$, and $\ddot{z_i^t}$), speed and displacement of particles due to the resultant force could be calculated via integration over one time-step length dt. As a result, the new location of particles at the next time step ($t + dt$) is found and updated. Some DEM software package also provides features about re-generating "bond" between elements under certain conditions so as to simulate adhesion property or re-grouping of third body particles[146], [151], [155]. Relative work can be found in [156], [157]. As the DEM method is an explicit solver, the accuracy of the simulation is greatly controlled by the value of time-step. Following the Raleigh method [158], [159], the maximum time step required by the DEM should be calculated as:

$$T = \frac{\pi R(\rho/G)^{0.5}}{0.163v + 0.8766} \tag{2.17}$$

where R, ρ, G and v are the particle radius, material density, shear modulus and Poisson's ratio respectively.

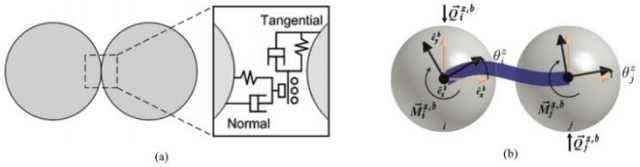


Figure 2.13. The bond between two elements in the Discrete Element Analysing: (a), the Spring and Damper Model[160]*; (b), the Cohesive Beam Model*[161].

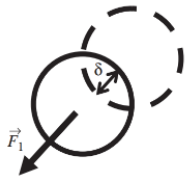


Figure 2.14.Demonstration of interaction between elements in Discrete Element Analysis

Initially, DEM was mainly run as research-oriented open-source software. Currently, more commercial DEM software is developed and more FEA software start to include a DEM algorithm in their software package. There are several names of similar methodology that can be found outside like a cellular automata

model and self-organizing method [162]. Particles inside the DEM model are undeformable which leads to its shortcoming in modelling the deformation of a solid geometry. Therefore, FEA still has its advantage in modelling a continuum material [163]. In addition, there is no force-control function inside the DE code. DE software is designed to simulate the evolution of discretised particles under the influence of solid geometry. It is assumed that the forces applied by the particles on solid are small so they are neglected. Such assumption works well for the situations where the interaction of particle upon solid is not of interest. An example is modelling the procedure of loading rock into a truck bucket where the impact of rock does not effect on the behaviour of the truck bucket. Lacking the force-control function included in the DE code is contrast to the demand of a fretting wear model. Therefore, development is required in order to implement a boundary condition of loading.

2.8 Combined Finite-Discrete Element Analysis

FEA has proven its ability in modelling continuum material or solid mechanics but has shown less appropriate in discontinuous problems like jointed and discrete rocks where DEM is more suitable [164], [165]. Taking advantages from both FEA and DEM, a hybrid method which couples the FEA and DEM termed Combined Finite-Discrete Element Modelling or FDEM was proposed by Munjiza and Owen who also pointed out the superiorities as well as deficiency of both FEA and DEM [165] [134]. This hybrid method was initially implemented in the field of geography where rock blasting and collapsing buildings in the civil industry [164]. This method allows modelling of mechanisms involving both continuum (solid deformability) and discontinuous problems (evolution of a large number of bodies) [164], for example, fracturing solids [165] and delamination analysis of composites [166].

There are various principles concerning the best ways in which to couple FE and DE methods [164], [167], [168]. There are two main categories of methods to which FDEM is coupled: I) there is no dynamic coupling between the FE and DE phases. In these cases, information such as the interaction between particles and solids is not transferred between the DE and FE phases. When FDEM is applied to such problems, the FE phase acts as a static boundary while the evolution of discrete elements is the main interest. In the work by Rojek and

Onate, a procedure of rock cutting via a mining pick was simulated via FDEM as shown in Figure 2.15 [169]. The disintegration process of cutting rock and the subsequent movement of rock debris (i.e. the orange and green dots shown in Figure 2.15) were modelled by the DE part, while the cutting machine tool and other rocks away from the surface (i.e. white triangle mesh elements) were modelled with the FE part. Because the material response of cutting machine tool was not studied in this case, interaction data was no transferred from the DE part to the FE part. As a result, the cutting machine tool was modelled to be rigid. Similar applications of this kind of FDEM can also be found in the work [170]–[172]: e.g. Cao et al. and Wang et al. studied the procedure of sliding wear via FDEM inside which a layer of degradable discrete elements (i.e. material to be worn during sliding) was attached to the surface of the FE body [171], [172]; II) the FE phase and DE phase are coupled. A typical example can be found in the work of the group at Oxford led by Petrinic. An FDEM was implemented to study the material behaviour after a treatment of shot peening [173]. Specifically, they used the DE part to calculate the velocity of metal balls, including locations of impact onto a solid section. The impact locations and velocities modelled by the DE phase were then used to generate impact forces on the solid section, and the FE phase was used to calculate the corresponding elastic/plastic response in the solid section. Inside such a method, information from DE and FE parts are coupled in one-way, which can be summarised in a flowchart shown in Figure 2.16.

Leonard et al. from the group of Sadeghi implemented FDEM into modelling the behaviour of debris particles between first bodies where the FE part was used to resolve the material response or behaviour (i.e. deformation, material remove and so on) and DE was responsible to estimate the evolution of third body particles between first bodies [57], [174], [175]. In the work by Leonard et al., the third body was modelled as a combination of rigid spheres, the volume of which was set as the volume removed from the first bodies in each physical cycle calculated via the Archard equation [174]. Leonard simulated the third body by connecting a small number of spheres, for instance, 2 to 14, in series following experimental work reporting thin platelets within the third body [174]. These were considered to be of a fixed shape and unable to break apart, extend or change. Although this work took considerable steps forward in coupling FE and DE phases as well as capturing some of the reality of fretting wear, it was not

validated against experimental data, and it was built around the Archard equation which was known to be inaccurate for fretting wear. Reported by Leonard, the debris particles located between contacting surfaces caused higher contact stresses. Therefore, applying the Archard Equation, a higher wear rate was expected which went against the experimental results where wear rate was reduced. Detailed information about the work by Leonard et al. can be accessed in [57], [63], [174]–[176].

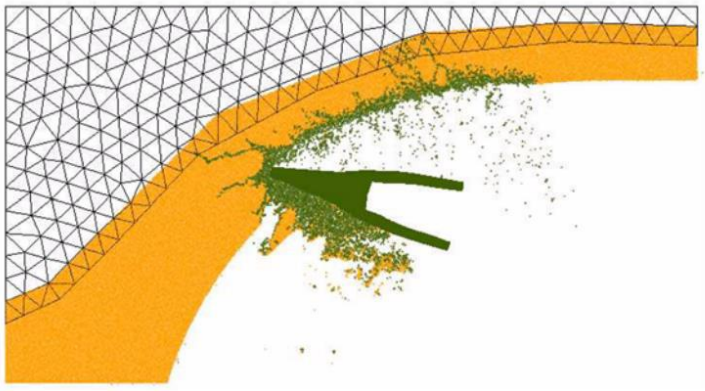


Figure 2.15. Simulation of rock cutting using Combined Finite-Discrete Element Modelling[169]*.The white triangle-meshed structure is finite elements while the orange and green dots are discrete elements.*

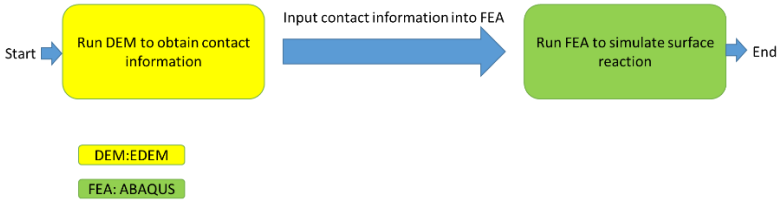


Figure 2.16. Flowchart of a 'one-way' Combined Finite-Discrete Element Modelling

The methodology about FDEM proposed by Benjamin et al. [57], [174] can be divided into 8 steps: step 1) an FE model is created according to the user-defined parameters including dimension, normal loads, fretting frequency, displacement amplitude, material properties, total number of cycles and so on; step 2) nodal properties like nodal pressure and slip distance are calculated; step 3) the Archard equation is implemented to calculate the nodal wear depth; step 4) the coordinates of the surface nodes where material is worn are updated based on the value of wear depth calculated in the previous step; step 5) a number of particles whose total volume equals to the worn volume are added and evenly distributed between the contact zone[174]; step 6) Using a map-searching algorithm by Hopkins[177], the relative location between particles are detected. The separation distances are then compared to a specified distance (e.g. 2 times the radius of the particle). If the distance is less than the threshold, a contact pair is recorded; step 7) applying Newton's second law, the interactions between

particles are calculated which are then used to estimate the new position of particles. As the time-step of simulation is set to be short, the relative particle position changes little. So the detection of the particles' location is performed only once per a user-defined number fretting cycles in order to reduce the computational expense[63].

2.9 Finite Element Analysis with Reduced Computational Cost

The nature of FE relies on the number of mesh element employed inside a model: a higher mesh density produces a result closer to reality. As the behaviour of each mesh element inside a model is analysed individually and interdependently, the computational cost can grow exponentially when the number of mesh element included in a model is increased. The same problem occurs to the DE when the number of discrete elements is increased inside a DE model. In a complex numerically intensive model such as this, simulation times can be very computational expensive as it requires to model a large number of elements (finite elements and discrete elements) and physical fretting cycles before obtaining the result. In addition, fretting wear is well known to behave nonlinearly which requires more calculation resource than linear analysis. There are three other solutions for speeding up the simulation or reducing the computational cost that are usually applied: 1) improve the calculation capacity of the computer; 2) balance accuracy and simulation speed; 3) use cycle-jumping by assuming a constantly changing rate of target variables so as to skip several calculation loops.

One of the most popular ways of implementing solution 1 is High-Performance Computing (HPC). It has become popular due to the increasing demand for complicated models. When applying HPC, a complete simulation job is divided into several groups each of which contains a small portion of job information. Each portion is then distributed to and processed by one of the processors inside a Central Processing Unit (CPU). Information is transferred between processors when such communication is required. Such procedure could also be called parallel computing. As a result, the total processing load is separated and carried out by many processors which increases the overall efficiency of calculation. Similarly, parallel computing is also conducted on Graphics Processing Unit (GPU), AKA General-Purpose Computing on Graphics Processing Units (GPGPU). Although one GPU processor has a lower operating speed, GPU

accelerator has the advantage of having much more processors with a lower price than the CPU. Therefore, GPU is usually implemented to handle massive but simpler calculations. Similar to the HPC, the whole job is split into a number of groups which are processed individually and interactively. Examples could be found as CUDA developed by Nvidia and OpenCL. Based on the description of HPC and GPGPU, it could be found that both techniques actually rely on parallel processing. Such strategy works fine in handling a large and complicated model but not in a model with a large time-span, i.e. a time-dependent model, such as modelling fretting wear for a large number of cycles. Solution 2 has been commonly applied via convergence tests included in the standard operating procedure of FEA: many trials with different mesh element sizes before ascertaining the most suitable mesh setting. Normally, convergence tests are conducted before deciding the most suitable mesh size.

Utilizing cycle-jumping, solution 3, the target variables are predicted over some cycles (i.e. jumping length) based on certain methods which therefore eliminates the need for simulating each individual cycle and significantly reduces the computational cost. There are two main groups of cycle-jumping techniques: I) predicts the target value (Y_{target}) based on an evolution theory obtained from experiments. Such evolution theory is usually expressed by a function of a target variable related to time as shown in Equation 2.18. In the work by Mumm et al., cycle jumping was applied to predict the material properties of a thermal barrier coating (TBC) via applying a function related to time [178], [179]. Such time-dependent function was obtained after a series of tests under certain conditions, e.g. averaged thickness of a thermal coating at various stages of cyclic exposure.

$$Y_{target} = f(t) \tag{2.18}$$

Such cycle-jumping technique has been largely employed in the field of simulating cyclically loaded structures or material fatigue. The material properties of structures which experience a large number of cyclic loading could be dramatically different to the initial properties. Instead of simulating the values of every material properties cycle by cycle, a cycle-jumping technique is used here for each property to estimate the corresponding value over some cycles. Different evolution theories will be implemented for different target variables. Again, a series of experiments are required which actually renders the modelling unnecessary. II) predict the value through extrapolation. It is normally assumed

that the variation of the target variable (Y) is constant over the jumping length (N). Such cycle-jumping method could be presented as below:

$$Y_{total} = N \times Y^N \quad (2.19)$$

where Y^N is the target variable predicted at the N^{th} cycle. Compared to the first prediction, this cycle jumping method does not limit its applications of problem. In the work by the Nottingham group, a constant jumping length (i.e N, 30 was used in the work by McColl et al.[59]) was used and the wear rate during the N cycles was assumed to be constant. Therefore, the total wear amount over the jumping length ($H(x,t)_{total}$) was calculated to be the product of the fretting wear rate of the N^{th} cycle ($H(x,t)^N$) predicted by the model and the number of cycles N as shown below:

$$H(x,t)_{Total} = N \times H(x,t)^N \quad (2.20)$$

Another method of extrapolation considers the linearity of the target variable: cycle-jumping is only permitted when linearity is detected and little to no cycle-jumping is allowed if non-linearity is detected as shown in an example by Cojocaru and Karlsson shown in Figure 2.17 [180]. As shown in Figure 2.7, a small N is applied in the left part of the curve, which makes the model simulate more often and improve the accuracy. A large N, a large jump, is applied in the right part of the curve in Figure 2.17.In the work by Cojocaru and Karlsson [180], an adaptive cycle jumping method was applied which is able to identify the non-linear part and only execute jumping when linear behaviour is detected. Inside that method, the ratio of slopes among three adjacent target variables was calculated as shown in Figure 2.18, cycle jumping was only executed if linearity was detected or the ratio of slopes was within certain tolerance. The linearity of the curve is checked via:

$$\frac{s_{23}(t_2) - s_{12}(t_1)}{s_{12}(t_1)} \leq ? \, q_y \quad (2.21)$$

$$s_{12}(t_1) = \frac{y_2 - y_1}{x_2 - x_1} \quad (2.22)$$

where q_y is a user-defined threshold of the difference between two adjacent slopes ($s_{23}(t_2)$ and$s_{12}(t_1)$) beneath which the curve is considered to be linear. The value of slopes ($s_{23}(t_2)$ and $s_{12}(t_1)$) are calculated between points P1 and P2 as S_{12}, and between P2 and P3 as S_{23} as shown in Figure 2.18. Of course, there are alternative methods to calculate the slopes, for instance, the value of slopes could be the gradient of fitted linear functions of two adjacent groups

$\overline{s_{23}(t_2)}$ and $\overline{s_{12}(t_1)}$ as shown in Figure 2.19. The linearity of the curve is checked as:

$$\frac{\overline{s_{23}(t_2)}-\overline{s_{12}(t_1)}}{\overline{s_{12}(t_1)}} \leq ? \, q_y \tag{2.23}$$

Such method is able to eliminate the effect of noisy data compared to the original one proposed by Cojocaru and Karlsson. After the linearity is checked, the jumping length Δt_{jump}^{M} is calculated based on the difference between the linearity and tolerance value (q_e): A smaller value of linearity (the data behaves more linearly) would generate a larger jump length, while a bigger value of linearity (the data behaves less linearly but under the tolerated limit) would generate a small jump length.

$$\Delta t_{jump}^{M} = [q_e \times s_{12}(t_1)] / \left[\frac{s_{12}(t_1)-s_{23}(t_2)}{\Delta t_{cycle}}\right] \tag{2.24}$$

It could be found that a large rate of change of slope (i.e.$\frac{[s_{12}(t_1)-s_{23}(t_2)}{\Delta t_{cycle}}$) or highly non-linearity results in small jumping length Δt_{jump}^{M}. In the work by Cojocaru and Karlsson, the outcome of this cycle-jumping method is presented in Figure 2.17.

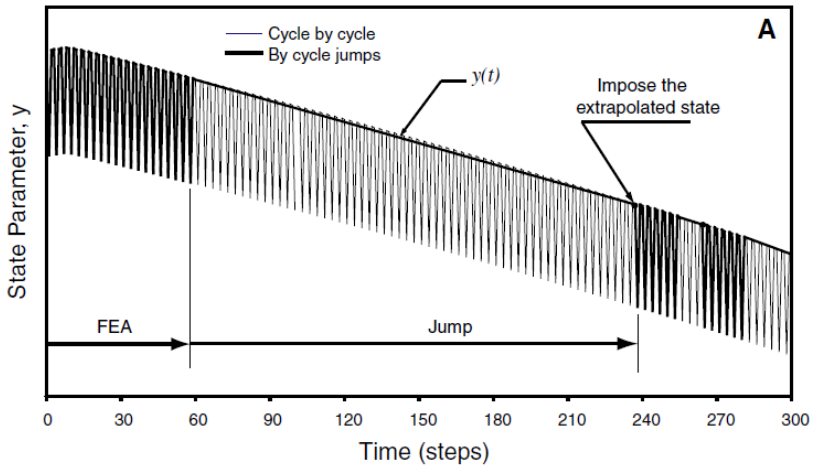


Figure 2.17. A brief demonstration of cycle-jumping by Cojocaru: cycle-jumping is allowed when conditions are satisfied as shown in the middle part of the graph[180].

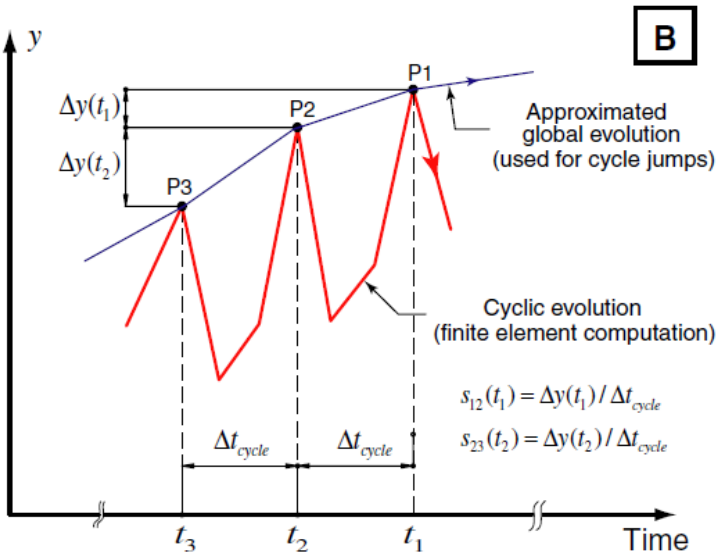


Figure 2.18. Demonstration of the cycle-jumping method by Cojocaru.

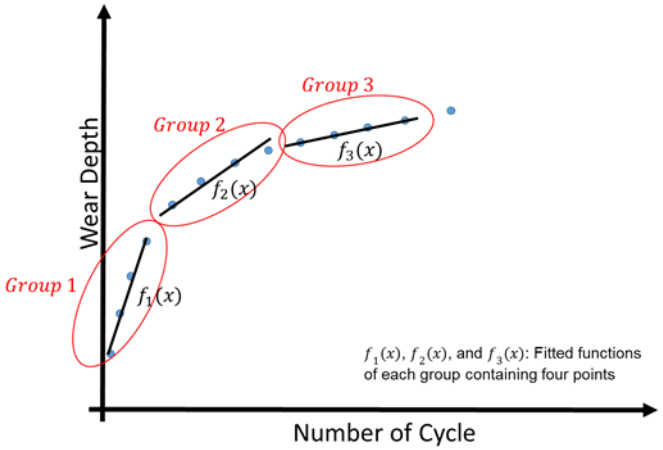


Figure 2.19. Demonstration of an alternative method to have four adjacent data points as a data group.

This cycle jumping method will constantly check the gradient of the variable of interest against simulation cycle number. A user-defined threshold (i.e. critical value of changing rate) is used over which the target variable is identified as non-linear and otherwise as linear. As a result, a small N is applied when change rate is bigger than a threshold (as shown in the left part of the curve in Figure 2.17), which makes the model simulate more often and improve the accuracy. A large N is applied when the property is identified as linear so as to save computational cost (as shown in the right part of the curve in Figure 2.17).

As discussed in the previous section, the first cycle jumping method requires a large number of experiments so as to form an empirical function of the target parameter. The second cycle-jumping technique does not require any prior knowledge. Therefore, it has a much wider range of application. However, it is a purely linear approximation in contrast to the possibly nonlinear function that might be derived from experimental results. One must be careful in selecting the

cycle jumping length as such a method could cause divergence issues if left unchecked. Therefore, a more careful processing strategy should be developed in order to apply this method in a complicated model like fretting wear.

2.10 Geometry Updating with Searching and Mapping Algorithm

Apart from the matrix processing inside the FE code, the procedure of updating mesh elements' coordinates takes most of the time spent by the whole simulation job. After wear depth is calculated at each element node, the coordinate of those nodes are updated accordingly. Such procedure is usually realized via a built-in algorithm inside the software like the code called UMESHMOTION inside ABAQUS. In the work by McColl et al. [59], UMESHMOTION was applied to update the location of mesh elements so as to reflect the wearing of material. However, this developed hybrid FDEM fretting wear model calculates the wear rate via combining the results from both FE and DE solvers. As a result, the procedure of calculation and updating nodes' coordinates has to be handled externally. On the other hand, due to the way ABAQUS handles nodes storage, identifying and searching the location of a target node might be extremely time-consuming. A typical scenario could be schematically shown in Figure 2.20: when material at node A is worn, the coordinate of node A will be updated downwards so as to represent a material loss. After that, the modification of the coordinate of the node B lying beneath A is required to maintain the aspect ratio of the mesh element. Such procedure will be applied to all the nodes beneath node A as shown in Figure 2.20.

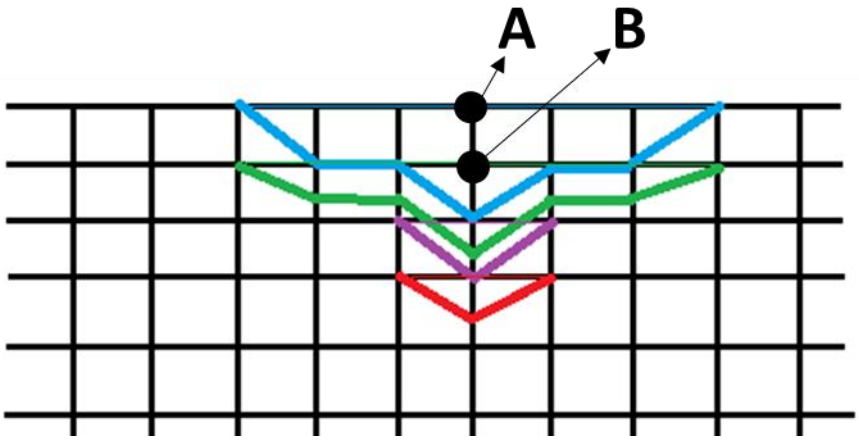


Figure 2.20. A Demonstration of mesh morphing in order to reflect material wearing.

This procedure of “identifying” a node is commonly referred to as searching and mapping in the field of computer science. Searching for the nearest node inside a space domain can be handled via different methods: 1) direct searching method, which calculates the distances (d_{min}) between the current node and all the other nodes via the Equation 2.25. Such calculation is applied all the nodes. If there are N nodes to be processed, it would take N^2 times of iterations to complete the searching procedure. In addition, the computational expense increases exponentially when the number of nodes is increased; 2) Extension of the direct searching method: this method only searches those nodes that located in the same plane with the current node. Specifically, the nodes which share the value of x or y or z are investigated. As a result, the number of nodes to be searched is reduced as well as the calculation is simplified as shown in Equation 2.26; 3) subdomain searching approach: which is developed based on an algorithm called binary-tree-search-method which has been largely applied in the field of computer science. This method subdivides the space domain into identical cells within which only several nodes are placed or mapped. The position-index of cells is then presented with a matrix made of three integers, i.e. (X_i, Y_i,Z_i) which displays the relative location of the cell inside the space domain. Those three integers are calculated via the Equation 2.27-2.29.

$$d_{min} = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2 + (z_1 - z_2)^2} \quad (2.25)$$

$$d_{min} = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2} \; or \; d_{min} = \sqrt{(x_1 - x_2)^2 + (z_1 - z_2)^2} \; or \; d_{min} = \sqrt{(y_1 - y_2)^2 + (z_1 - z_2)^2} \quad (2.26)$$

$$X_i, = 1 + Int(\frac{x_i - x_{min}}{d}) \quad (2.27)$$

$$Y_i = 1 + Int(\frac{y_i - y_{min}}{d}) \quad (2.28)$$

$$Z_i = 1 + Int(\frac{z_i - z_{min}}{d}) \quad (2.29)$$

where x_{min}, y_{min}, z_{min} are the minimum value of coordinates among all the nodes. Inside the subdomain, a direct searching method is also conducted. Such method further reduces the searching domain around the current node from the whole domain into a small cell.

Chapter 3

Combined Finite-Discrete Element Modelling

3.1 Introduction

Fretting wear can be observed to behaviour non-linearly. In several fretting wear experiments, a marked change in the wear rate was observed after some number of cycles [31], [181], [182]. It is proposed that such non-linear behaviour is caused by the debris particles trapped in between contacting surfaces[139], [140]. Most wear prediction models are made via FEA which works well in handling interaction between solid bodies but less in modelling the evolution of a large number of bodies, i.e. debris particles in this case. A typical example of modelling fretting wear in FEA can be found in the work by McColl et al. [59]. DEM is mature in simulating the kinematic for granular media but not at modelling continuum material. A typical example of DEM method for modelling fretting wear is the work by Fillot et al. [146], [151]. Since DEM is suitable to model discrete bodies [183] and FEA is good at resolving linear and nonlinear solid mechanics, FDEM is able to take advantage of both methods [169]. One hybrid fretting wear model which combines FEA and DEM was developed by the group of Sadeghi and Benjamin in Purdue, based on earlier hybrid methods developed by Potapov et al.[184]–[187] for applications such as solid fracture and granular packaging. Within this hybrid method, FE was employed to simulate the first bodies, while DE was used to model the evolution of the debris particles, i.e. the third body.

Benjamin's work successfully took a step forward in coupling FE and DE, demonstrating some new results. For example, it was found that the increment of contact load led to a reduction of wear volume during partial fretting wear, which was explained by a decreased slip distance due to the increased friction forces in a load-driven system (i.e. the stick zone between contact surfaces increased under high contact loads) [57]. It also found that the rigid debris particles caused local stress concentrations in the first bodies at the contacting points[57]. However, according to the Archard equation, that result will predict higher wear rates which contradicts much of the experimental literature demonstrating

reduced wear rates where debris plaques are present. This shows the Archard equation is not fit for fretting wear predictions. This FDEM fretting wear model still has several key problems: I) poor description of material failure during fretting wear; II) incorrect prediction of debris particles evolution, i.e. the debris particles are constrained within a single vertical plane; III) linear cycle-jumping technique which does not account for non-linear and noisy data. Besides those problems above, it is extremely difficult to implement the FDEM model developed by Benjamin by the users since it requires professional knowledge about the code. In this paper, a new formation of the FDEM fretting wear model was developed, which not only allowed easy implementation but also allowed full control of physical variables in the fretting wear model addressing the three principal problems identified previously.

Problem I: poor description of material failure during fretting wear

Both the Archard equation and the Dissipated Energy method rely on a wear coefficient which has received criticisms for its simplicity and does not reveal any wear mechanism. The model proposed in this work is sufficiently general so that it was possible to implement a range of criteria for material failure. In each case, a criterion derived from a proposed evolution mechanism was used to identify worn material. Stress, strain and other data available from the FE model phase can be used to evaluate the volume of material worn. The surface geometry is then updated and wear particles of the corresponding volume are generated in place for the DE phase.

Problem II: incorrect prediction of debris particles evolution

Complex 3D models of contacting surfaces, with significant non-linearity and thousands of wear particles, have high computational cost. This is why the FDEM fretting wear model proposed by Benjamin was built in 2-dimensional. However, modelling fretting wear in 2-dimensional limits the evolution of debris particles which could cause the load distribution inside the contact zone to be misevaluated. It is important to consider the third dimension if possible since the evolution and fate of debris particles are so significant to the prediction of fretting wear[151], [188].

Problem III: linear cycle-jumping technique

The cycle-jumping technique in the work by Benjamin used a constant jumping length and does not check linearity. This conflicts with the non-linearity observed in fretting wear experiment. An alternative cycle-jumping method, such

as that by Cojocaru et al. [180] only implements cycle-jumping when linearity is detected. Although it is a more refined approach, it was also found in this work that it poorly handled noisy data, therefore one new adaptive cycle jumping was developed based on the method by Cojocaru and implemented herein. An error correction technique was also implemented here so as to avoid error accumulated due to the non-linear behaviour of fretting wear.

Besides those problems stemming from Benjamin's FDEM fretting wear model, two main issues appeared due to the features added in this work. Firstly, the more general nature of the model produced in this work means that a more hands-on approach of the numerical solution was taken. Usually, the procedure of predicting a wear rate and updating information of the model is conducted inside the software (e.g. UMESHMOTION). However, the wear rate predicted via the FDEM model combines both calculation results from the FE and DE parts, which requires the procedure of updating information to be controlled externally. Due to the way FE code handles nodes storage, identifying and searching the location of a target node might be extremely time-consuming. Therefore, an algorithm was developed in this work to quickly identify and search target nodes. Secondly, the DEM is a displacement-controlled solver inside which a force-controlled boundary condition cannot be implemented. In order to simulate the procedure of fretting wear where a normal load is applied, a boundary moving algorithm was developed to implement a force-control feature inside the DE code.

3.2 Combined Finite-Discrete Element Fretting Wear Model

The following content is arranged as follows. Before introducing the setup of the FDEM fretting wear model, two critical algorithms used inside the model are introduced first: a searching and mapping algorithm and a boundary moving algorithm. Then, the programme structure of this FDEM fretting wear model is overviewed to help readers understand the interaction between the FE and DE solvers as well as the information flow between the two software. After that, the preparation of files is briefly introduced with details included in a user manual in the appendix. Finally, following a description of the setup inside the FE and DE phases, a road-mapping of this FDEM fretting wear model is presented step by step.

3.2.1 Searching and Mapping Algorithm

Inside the FE code, the coordinates of nodes are saved in a special format, i.e. ($Index_i$, $X-Coord_i$, $Y-Coord_i$, $Z-Coord_i$), which indicates the index of the nodes and corresponding 3-dimensional location. It should be noticed that two adjacent nodes (e.g. a node with $Index_i$ and a node with $Index_{i+1}$) may not necessarily be neighbour in space. As a result, identifying and searching the location of a target node might be extremely time consuming. An example can be identifying a surface node and searching for its closest neighbour node. A similar problem occurs when transferring nodal force statements from the DE phase to the FE phase because of their different strategies of saving node coordinates. Thus, in order to quickly process those nodes, a searching and mapping algorithm based on the idea of subdomain searching was developed in this work. A comparison is made between the developed algorithm and other three traditional methods and the best of them will be applied in the model.

After the coordinates of nodes are read into memory, all the nodes are re-arranged according to their coordinates as indicated in Figure 3.1: firstly, nodes are arranged in ascending order according to the value of their x-coordinates; for the nodes that share the same x-coordinate, they are further re-arranged in ascending order based on their z-coordinate; finally, for those nodes that share the same x and z-coordinates, they are ordered descending according to their y-coordinates to ensure the first node in each list to be a surface node. As a result, inside the re-ordered list of nodes, the first node is always the node on the corner on the top surface which could be illustrated in Figure 3.1(a). Then position-index, describing the node's relative location inside space, is assigned to the re-ordered node accordingly as shown in Figure 3.1(b). For instance, inside the example displayed in Figure 3.1, the index of the node at the top left corner is (1,1,1). The node directly beneath the node (1,1,1) is assigned to be (1,1,2) and the one besides (1,1,1) is then defined to be (1,2,1) or (2,1,1) depending on its relative position.

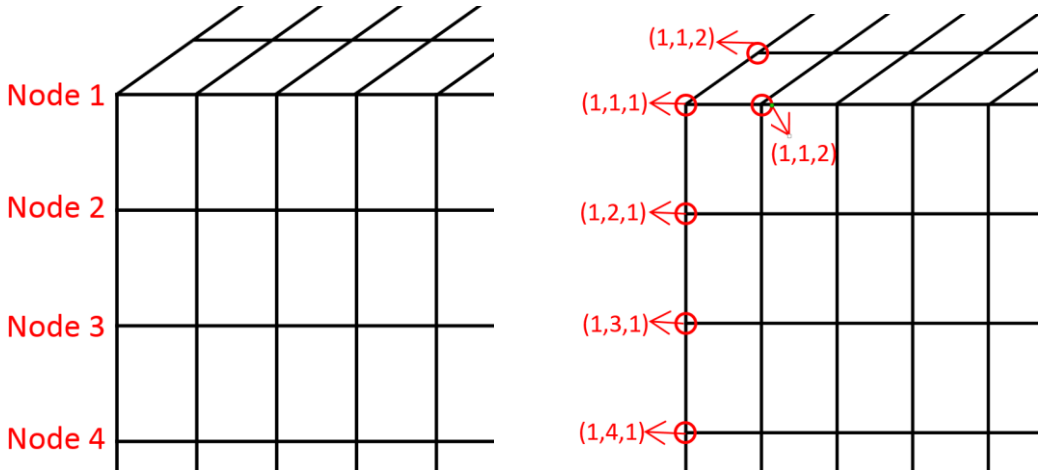


Figure 3.1. (a) A demonstration of re-ordered nodes; (b) A demonstration of re-order nodes with position index.

Taking advantage of the association between node and position index, the procedure of identifying and searching is greatly simplified. For instance, when a material evolution criterion is applied, the surface nodes can be identified (with index in form of $(o, 1, q)$) and then investigated which saves the time spent on checking the bottom nodes which could be less likely to be worn. After that, when a node with index $(o, 1, q)$ is identified to be worn or modified according to the material evolution criterion, the coordinates of all the nodes beneath are then modified in order to maintain the quality of mesh elements. By searching the position of the index $(o, p+n, q)$, the n^{th} node beneath the surface node is accessed. The same strategy was used to transfer or map force statements from the DE phase to FE phase: via searching for a matched position-index, node's information inside the DE phase is mapped into the FE phase. The procedure of mesh morphing and calculation of wear depth are detailed later.

Four searching and mapping algorithms were tested through two tasks: mesh morphing or updating the mesh geometry in a <.inp> file and mapping force statements into a <.inp> file. In Table 3.1, the performance of four different searching and mapping algorithms is presented. Because the algorithm developed in this work avoids wasting time in checking the "wrong" nodes, it is able to process the file with the highest efficiency compared to the others. As presented in Table 1, for one algorithm, the task of mesh morphing takes longer time than that of mapping force statements. That is because the latter one only works on the surface nodes while the former one needs to process the nodes beneath the surface nodes.

Table 3.1. A comparison of implementing different searching algorithm was presented.

Task One: Mesh Morphing/Geometry Updating				
Item	Method One	Method Two	Method Three	New Method
Processing Speed	11 Nodes/s	25 Nodes/s	10324 Nodes/s	11762 Nods/s
Task Two: Mapping Force Statements into a <.inp> File				
Item	Method One	Method Two	Method Three	New Method
Processing Speed	60 Nodes/s	145 Nodes/s	23534 Nodes/s	37638 Nodes/s

3.2.2 Boundary Moving Method as Force-Controlled inside DE Phase

DEM software is an explicit solver whose accuracy is decided by its time step. Thus, a displacement-control function is required to control the simulation. The kinematic of boundary elements inside the DE code is calculated as:

$$\Delta d_{x,y,z} = v_{x,y,z} \times dt \tag{3.1}$$

where $\Delta d_{x,y,z}$ is the displacement in x, y, and z direction per analytical step, dt is step length, and $v_{x,y,z}$ is a user-defined velocity which is usually constant. There is no force-control function inside the DE code which on the contrary is demanded by the fretting wear model. Inside this work, a boundary moving method was developed so as imitate a force-control function and apply a “virtual” loading force on the upper first body. Inside the boundary moving method, the moving distance per analytical step is calculated via evaluating the contacting condition: a large displacement is applied to bring one surface closer to the other one when the contact force is lower than that required, while a small or no displacement is used when contact force is reaching the target value.

During the implementation of the boundary moving algorithm, the target or “virtual” loading force F_v is first applied on the upper first body. The corresponding virtual acceleration a_v is calculated via:

$$a_v = \frac{F_v}{M} \tag{3.2}$$

where M is the mass of the first body. Then, the speed produced by the acceleration in the present analytical step is calculated as:

$$v_{new} = v_{old} + a_v \times t \tag{3.3}$$

where v_{old} is the speed of the first body in the previous analytical step, which equals to zero in this case. An initial velocity could also be included in order to

speed up the procedure. After that, the dislocation of the first body within the current time step is:

$$d_{new} = v_{new} \times t \tag{3.4}$$

The dislocation value is then passed to the DE solver on the upper first body. At the same time, the total interaction between the particles and the first body F_{int} is ascertained by the solver and fed back to the algorithm. Thus, in the subsequent analytical step, the acceleration of the upper first body is updated to be:

$$a_v = \frac{F_v - F_{int}}{M} \tag{3.5}$$

The same steps are followed to calculate the corresponding updated speed and dislocation which are then passed to the DE solver to update the location of the first body again. Virtual damping forces can be added between the particles and first bodies so as to improve the stability of the simulation. Then, the calculation of acceleration becomes:

$$a_v = \frac{F_v - F_{int} - C_c \times v}{M} \tag{3.6}$$

where C_c is the damping coefficient of the virtual dampers. This procedure above is repeated until the total loading force applied by the upper first body was completely carried by the debris particles. Following such procedure, a force-control function has been realized inside the DE code.

3.2.3 Overview of Combined Finite-Discrete Element Model

The FDEM model consisted of two main parts: the interaction between first bodies or continuum bulk geometries is modelled with FE while the debris particles or known as the third body is simulated using DE. ABAQUS/Standard was employed herein to handle the FE part for its powerful non-linear analysis capacity and considerable flexibility in pre- and post-processing. On the other hand, there are a number of options for DE codes on the market: commercial software such as EDEM from DEM Solution Ltd, PFC from ITASCA Group, and Rocky-dem from ESSS, and open-source software like LIGGGHTS, Yade, ESyS-Particle. A DE code had also been added into ABAQUS/Explicit, but it was found to be more computationally expensive compared to the other pure DEM codes[189]. It could be caused by the extra numerical cost of solving the matrix equation for finite element even when there is no or little deformation on solid bodies. The DE code from DEM Solution was used here for its rich customised

features and flexible controllability. Another advantage of this DE code is its ability to model complexly shaped particles via a multi-sphere function.

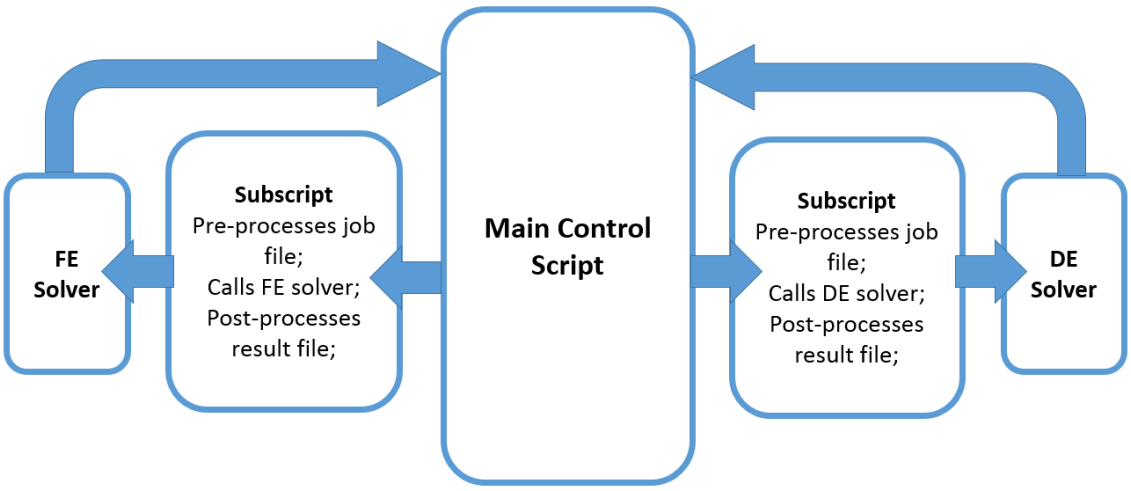


Figure 3.2. Schematic structure of the hybrid FDEM. A main control script (in Python) calls the subscripts (in C++) by sequence accordingly in order to launch the FE or the DE part of the code. After the job of FE or DE part, the main script retrieves the data to process on it.

Because there is no built-in function inside ABAQUS or EDEM to allow a direct coupling between the two solvers, a "main control script" was developed herein in order to link or couple the FE and DE phase. As the programme structure illustrated in Figure 3.2, the main control script controls the FE and DE phase via two subscripts. The two subscripts are responsible for pre- and post-processing, specifically passing job information from the "main control script", triggering simulations, extracting results from corresponding solvers when simulations were completed, and transferring the results back to the "main control script".

Both FE and DE solvers are controlled and manipulated in sequence inside the FDEM fretting wear model as presented in Figure 3.3. Because running the DE code is extremely computational expensive and computational resource is limited, the DE part of the code is turn off until reaching the number of cycle when the effect of debris particle required to be investigated. For example, in order to study the effect of debris particles at the 1000th fretting cycle, only the FE part inside the code works from 0th cycle to 999th cycle. At the 1000th cycle, DE part will use results from the FE part to calculates the interaction between third bodies as well as between particles and solids. Then, the result generated by the DE part is transferred back to FE part to be further processed. As shown in Figure 3.3, the "main control script" firstly calls an FE simulation whose results are firstly used to calculate wear volume according to the criteria selected by the user. Afterwards, that results are then used to generate data such as the number of

debris particles with their locations required by the subsequent DE phase. After completing the simulation inside the DE phase, results are then transferred back to the FE phase. In the following sections, the preparation of initial simulation files was briefed followed by an introduction of the setup of the fretting wear model inside both FE and DE phases.

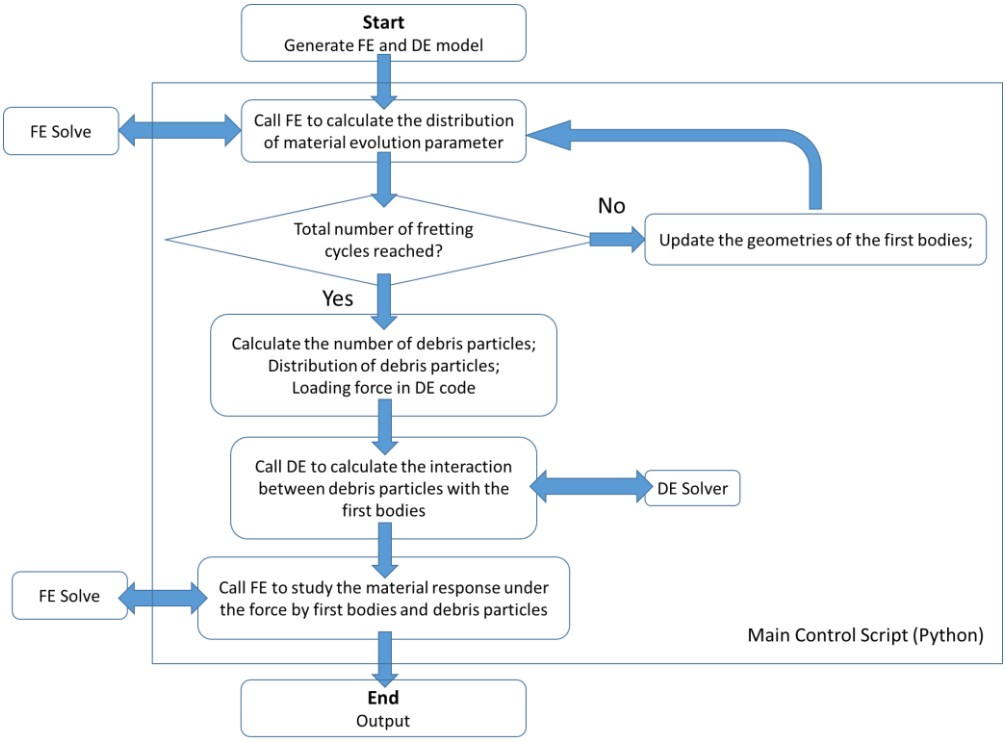


Figure 3.3. A demonstration of information flowchart of the hybrid FDEM.

3.2.4 Preparation of Finite Element Phase and Discrete Element Phase

Before launching the simulation, several files as listed in Table 3.2 should be prepared before being called.

Table 3.2. Check List of initial files required by the Hybrid FDEM model.

	FE Phase	DE Phase
Item	<.inp> job file	• <.dem>, <idx>, <.ess>, <dfg> and <.ptf> job files • <.txt> boundary condition file • <.txt> information of debris particles • <.dll> API processes information of debris particles • <.exe> API processes movement of first bodies

Because ABAQUS saves the job information in a single <.inp> file, for the FE phase, one FE job <.inp> file is required which containing information such as geometry dimension, material properties, and boundary conditions. Inside the <.inp> file, the information of geometry dimension is modified when the material evolution criterion is applied so as to reflect material worn. This <.inp> file could

be generated via ABAQUS GUI. Two geometry files, i.e. <.stl> files describing the shape of the first bodies, are required by the DE phase. The geometry files are generated from the FE phase via 'export geometry' function provided inside the FE code, which maintains the consistency between the FE and DE phases. Both geometry files are updated together with the geometry information inside <.inp> file according to the material evolution criterion implemented. The DE job files (i.e. <.dem>, <idx>, <.ess>, <dfg> and <.ptf> files) are generated via EDEM GUI which defines the material property and a library of debris particles. The boundary conditions of the DE job and initial location of debris particles are defined inside two <.txt> files individually and implemented via Application Programming Interface (API). The API (i.e. a <.exe> and a <.dll> files.) read the information from the <.txt> file and apply to the DE phase explicitly. Details information could be found in the manual included in the Appendix.

3.2.5 Setup of Finite Element Phase

The FDEM model described in this paper is a three-dimensional fretting test geometry, which contains two perpendicularly-crossed contact cylinders (of 8 mm diameter), but of course, the method is fully general and allows any geometry required. For computational efficiency, only a portion of both cylinders was considered, as shown in Figure 3.4. In exchange for further simulation efficiency, both cylinders were segmented into several parts. Fine, eight-node hexahedral mesh elements (c3d8 in Abaqus) were employed in the contact region where the material behaviour is especially of interest. Coarse, four-node tetrahedral elements (c3d4 in Abaqus) were used in the other part where the material is not as sensitive.

The size of mesh elements was ascertained through conducting a convergence test so as to capture the correct material response in the contact region as well as to have the least computational cost. During the convergence test, comparisons were made between results predicted by the well-known Hertz contact solution and FE simulation results. The result was considered to be converged when the difference was less than 3% and corresponding mesh size was $20\mu m$ applied in the contact region. A slightly big mesh height, i.e. $20.1\mu m$, was assigned to those fine mesh elements in order to avoid bad aspect ratio generated after implementing the mesh morphing algorithm (section 3.3.7). In addition, similar convergence tests were conducted to decide the dimension of the partition of fine mesh elements and the size of coarse mesh elements. Finally,

the dimension of the contact region with fine mesh was decided to be $3mm$. Tie constraints were applied to the surfaces where the mesh densities were different on two sides.

Between the two first bodies, a surface-to-surface contact algorithm inside ABAQUS was employed. Since the use of a material evolution criteria at the scale of individual elements could lead to irregular incongruent surface profiles, using surface-to-surface algorithm is able to avoid bad simulation results like over penetration which could be caused by using a node-to-surface algorithm. During the simulation, the translation scheme of the upper first body was presented in Figure 3.5 (a). A normal loading was first applied on the upper first body during step 1, followed by a linear constant rate translation motion which was then reversed in direction during step 2 and step 3. The lower first body was encastrated, i.e. translations and rotations in all directions were blocked. After finishing the setup above, an FE job file was then generated by having the CAE GUI write the job information into a <.inp> file. After the FE simulation of a single fretting cycle was completed, the result of material evolution parameter was extracted from output <.odb> files and fed back to the “main control script” to be further processed.

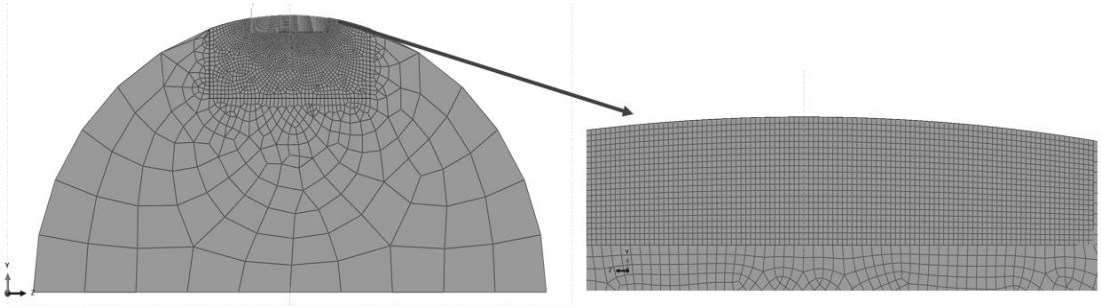


Figure 3.4. Finite element model of cylinders used in FDEM model. Part of the model is defined with fine mesh elements while the rest has coarser elements.

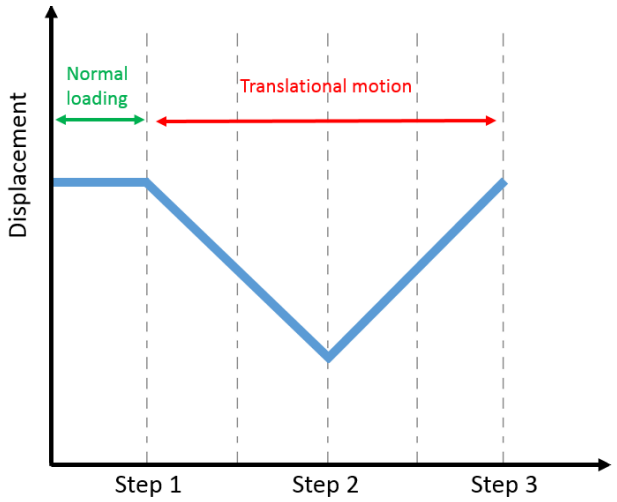


Figure 3.5. Motion scheme of fretting wear mode. A normal force is firstly applied to reach the loading force followed by a translation motion as the fretting motion.

3.2.6 Setup of Discrete Element Phase

Inside the DE phase, the two first bodies were defined via importing the <.stl> files generated from the FE code. After the first bodies were created inside the FE phase, both first bodies were exported into two “virgin” <.stl> files individually. Following the update of geometry information inside the <.inp> files, both “virgin” <.stl> geometry files were also modified according to the same material evolution criterion. Then, the two modified <.stl> files were imported into the DE solver. The material of the first bodies was defined to the same as that inside the FE phase. A library of different discrete elements was also defined based on the type of debris particles observed in fretting wear experiments [190], [191]. In this work, three differently shaped debris particles named spherical, chunk and rubbing were included in this library. Each kind of debris particle was imitated inside a three-dimensional coordinate by grouping several spheres as shown in Figure 3.6. Debris particles can be observed oxidized during a certain type of fretting wear (e.g. gross fretting wear) or a certain stage of fretting wear (e.g. steady stage). Oxidation of debris particles was not included in this model, the material properties of debris particles were defined the same as the solid material.

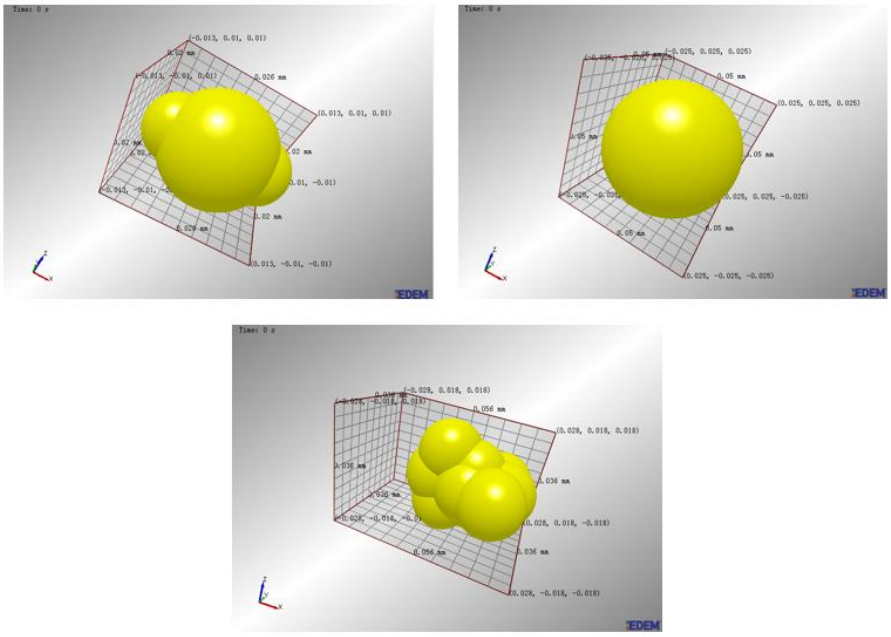


Figure 3.6. Imitation of debris particles inside DE simulation. Three different kinds of debris particles are simulated inside the DE code.

In this model, a Hertz contact model was employed here to calculate the interactive force between debris particles and that between particles and first bodies. No adhesive forces were included in this model although this could be included via implementing a JKR contact model[192]. A Coefficient of Restitution (CoR) was used to define the ratio of the final to initial relative velocity between two debris particles after they collide. According to work by Marinack et al.[193], the value of CoR between steel was set to be 0.72. The Coefficient of Static Friction and Roll Friction are set to be 0.6 and 0.001 respectively. In this work, the interaction coefficients between discrete elements were assumed to be the same as that between the discrete elements and the first bodies. After the interactions were solved, Newton's second law was then used to calculate the acceleration, speed and update the position of the particles accordingly. Following the introduction of debris particles into the contact zone as shown in Figure 3.7, a load was applied and particles were compressed and relocated as shown in Figure 3.8.

The DE phase resolved the kinematics of the particles and their interactions with each other and first body surfaces. The forces exerted on the first bodies by the particles were then fed back from the DE phase into the FE phase so that forces acting on the surface from both debris particles (and plaques) and contacting first bodies can be considered in the FE phase.

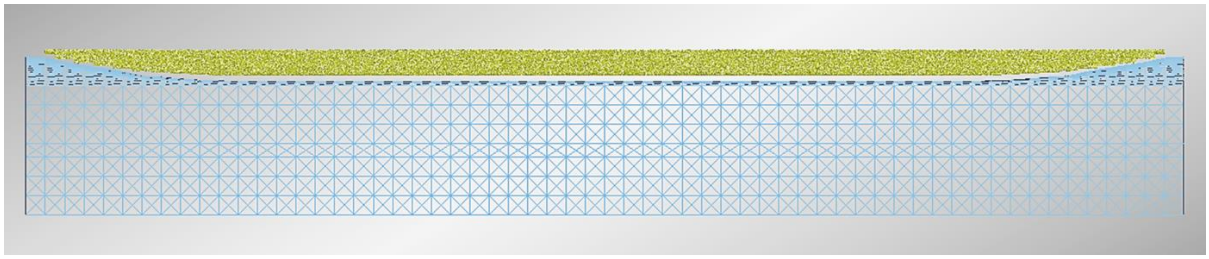


Figure 3.7. Distribution of debris particles generated in space: particles are evenly distributed across space between contact surfaces before compression loading.

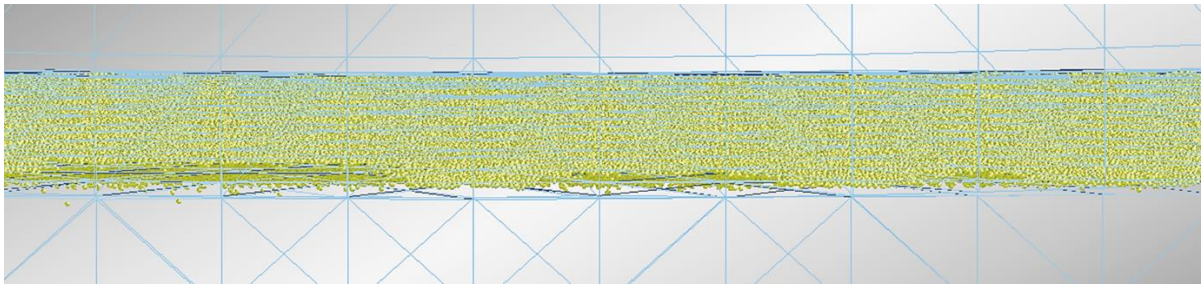


Figure 3.8. Distribution of debris particles after compression: particles are compressed, forming a single layer between contact surfaces.

3.2.7 Material Evolution, Mesh Morphing and Force Exchange

After the FE phase was completed, material evolution criteria were implemented in the main control script. The algorithm chosen here determined if volumes of material were considered to remain on the surface of the first bodies, or if they were removed from the surface to become wear debris. As material evolution is handled by the main control script, this method is general so any material evolution criteria can be used. In this case, a total of 6 criteria were implemented, specifically the i) internal vertical stress (corresponding to the Archard equation), ii) the total dissipated energy (corresponding to Fouvry's method [61]), iii) equivalent plastic strain (corresponding to McCarthy et al's method [97]), iv) i.e. the von Mises stress criterion, v) shear stress based critical plane method (corresponding to Brown et al[122], [125]):, vii) the shear stress in the plane normal to the applied contact pressure.

Table 3.3. List of material evolution criterion implemented in this work

	Material Evolution Criteria	Algebraic Form
i)	Internal vertical stress	$\sigma_{11} \geq \sigma_{yield}$
ii)	Dissipated energy	$E_d \geq \sigma_{yield}$
iii)	Equivalent plastic strain	$PEEQ \geq \sigma_{yield}$
iv)	von Mises	$\sigma_v \geq \sigma_{yield}$
v)	Critical plane method	$f(\tau_{12}, \tau_{\perp}) \geq \sigma_{yield}$
vi)	In-plane shear stress	$\tau_{12} \geq \sigma_{yield}$

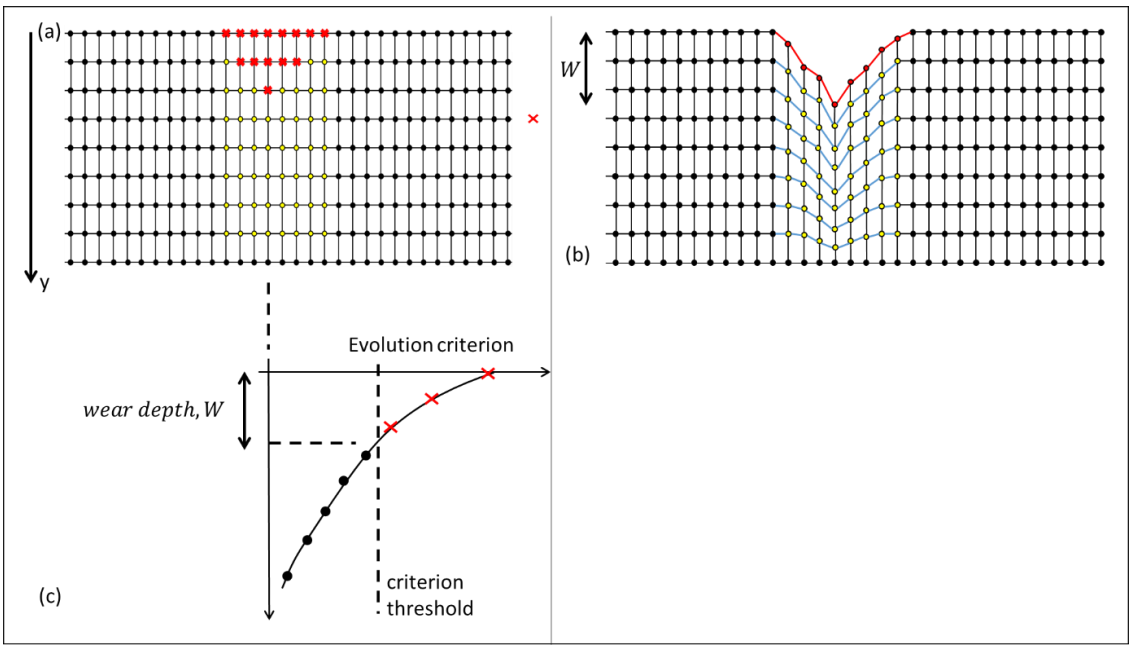


Figure 3.9. A schematic demonstration of mesh morphing. Evaluation of the mesh nodes as presented in (a). Part (c) illustrates the interpolation to calculate the morphing distance between element nodes. After morphing, the updated mesh is presented in (b)

The FE analysis solves/predicts the results of models based on the nodes inside the models. Via interpreting the boundary conditions (e.g. displacement or force distribution) applied inside the model, the corresponding material behaviours like the stress and strain at each node inside the model can then be predicted. Such solving procedure involves applying shape functions (depending on the type of mesh element used inside the model) to interpolate results between elements and checking of converging criteria especially for a non-linear material behaviour. After the desired results at each node were generated, the algorithm developed in this work read the results for material evolution.

During the process of the algorithm, any nodes in the FE model reaching the user-defined critical criterion value were considered to be worn, and the material close by as lost from the first body surface. In order to update the geometry of the first body surface and to avoid re-meshing costs, the nodes affected were moved downwards so that the revised surface lost the required volume. To determine how far the nodes must be displaced, a linear interpolation was made of the evolution criterion parameter (such as von Mises stress) from the surface node downwards beneath it to the last node in the ultra-fine model section, as shown in Figure 3.9(b). This interpolation allowed approximate determination of the depth below the surface where the criterion was exceeded at a resolution finer than the element size. The new location of the surface node (red nodes shown in Figure 3.9(c)) was therefore at the depth at which the criterion threshold

was met. So as to gradually displace subsurface nodes, a morphing coefficient m was defined as being the displacement of the worn surface node ΔW_n divided by the total depth of the ultra-fine model section H. Other dimensions could be used at this juncture for normalisation of node repositioning. This morphing coefficient was iteratively applied to each subsequent lower node until the bottom of the ultra-fine section, so as to achieve a graduated and gentle change in element shape and aspect ratio. Such procedure required searching the nearest element node around the given element node, a searching and mapping algorithm was developed so as to speed up this procedure (Section 3.7). As shown in the example in Figure 3.9(b), the morphing coefficient, m, was calculated as:

$$m = \frac{H - \Delta W_n}{H} \tag{3.7}$$

The change in position for all the nodes below the worn surface was calculated using m. The vertical position for node i after relocation was calculated as:

$$y_i = m \times (y_i - y_{bottom}) + y_{bottom} \tag{3.8}$$

The geometries inside the FE phase were modified by updating the nodes information inside the <.inp> file. The solid geometries modelled inside DE phase was updated by modifying the <.stl> files.

In order to calculate the worn volume (V) at a number of nodes transformed into new debris particles of similar volume, a polyhedron was defined around congruent worn nodes, as shown in Figure 3.10. This had one base face as the initial surface geometry, and further faces made by connecting adjacent displaced nodes. Calculation of the exact volume of such irregular and complex polyhedral was possible but possibly computationally costly.

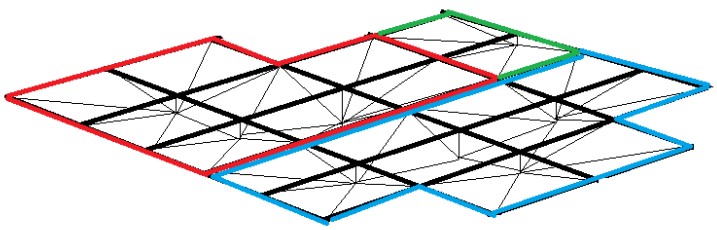


Figure 3.10. Worn volume identified inside the FE mesh geometry.The volume is usually made of a number of squares as base and wear depths as the heights.

An approximate solution for the worn volume of a complex polyhedral can be derived much faster by using the average wear depth of neighbouring nodes and

the number of congruent elements affected. Since the element size is predetermined, the worn volume can be calculated as:

$$V = A \times \Delta W_n \tag{1.9}$$

where V is the total wear volume of this set of congruent elements, A is the sum of the element in plane areas, and W_n the average wear depth in the nth fretting cycle. After statistical analysis, this approximation method would generally over-proximate the volume by around 30%. To balance the over-estimation, the value of volume used in the subsequent calculation was reduced to 77%. Then, the number of discrete elements was calculated by dividing the volume of worn material with the volume of a single sphere particle via:

$$N_d = \frac{V}{V_{sphere}} = \frac{V}{\frac{1}{6}\pi d^3} \tag{3.10}$$

Based on previous research, the size of debris particles falls within the range between $0.5\mu m$ and $1.2\mu m$ [174]. A further research about the size of debris particles during fretting wear could be conducted. In this work, discrete elements with a diameter of $0.6\mu m$ based on the work by Leonard [174] were used. It was of course possible to generate any size of the particles, or indeed distribution of sizes and shapes as preferred. Inside this work, due to the lack of exact distribution of debris particles from fretting wear tests, it was assumed that in this model each kind of discrete element occupied the same percentage of the whole particles. These were initially located randomly within the space of the congruent worn area, for instance within the coloured lines in Figure 3.10. Afterwards, the locations or centroid of debris particles are generated by producing random points within the space of the worn material. These debris particles information was then used to create debris particles in the subsequent DE phase. Additionally, it was avoided to generate points located on the edge or surface as it could cause an extremely high re-bounded force between debris particles and solid geometry.

The total normal force applied by the first bodies/solid bodies upon the debris particles in the FE model was used to set a target normal force in the DE using the moving boundary method (section 3.2.2). The DE phase required the sum of loads on particles at the sites of wear. Using forces at wear locations extracted from the FE output data, a shape-function method was developed to calculate the local loading forces. As shown in the example in Figure 3.11, the solid black points were locations of existing debris particles as identified by the DE phase, and nodes from the FE model were shown in red. For each particle, the normal force was required, yet the FE phase did not consider particles but only reported

forces at element nodes (X1,Y1) to (X4, Y4) for each mesh element. In order to determine forces at arbitrary locations over the surface, for instance at location (x,y), the shape function in Equation 3.11 to 3.15 were applied to the four nearest neighbouring nodal forces. This planar function worked well with forces from linear elements such as those used in this FE model and indeed was the only function to solve with a unique set of parameters. If quadratic type elements were used in the FE model, more complex force data would be available for each element, and thus use of polynomial or other shape functions would be required.

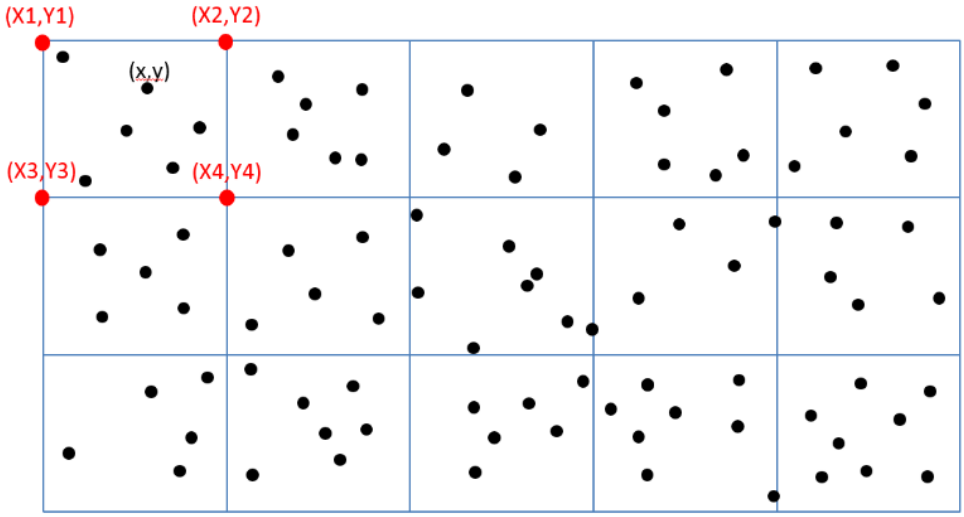


Figure 3.11. The squares are elements from the FE mesh, and the black dots represent debris particles.

$$Local\ force_1 = a_1 + a_2 * X1 + a_3 * Y1 + a_4 * X1 * Y1 \quad (3.11)$$

$$Local\ force_2 = a_1 + a_2 * X2 + a_3 * Y2 + a_4 * X2 * Y2 \quad (3.12)$$

$$Local\ force_3 = a_1 + a_2 * X3 + a_3 * Y3 + a_4 * X3 * Y3 \quad (3.13)$$

$$Local\ force_4 = a_1 + a_2 * X4 + a_3 * Y4 + a_4 * X4 * Y4 \quad (3.14)$$

This allowed the best fit of the function to the nodal forces in each element. After the coefficients a_1 to a_4 were solved, the local contacting force at debris locations (the black dots) could be calculated using that equation:

$$Local\ force_{(x,y)} = a_1 + a_2 * x + a_3 * y + a_4 * x * y \quad (3.15)$$

After ascertaining the value of the local normal force for each debris particle, the sum of these was then passed over to the DE phase as the loading force applied by the upper first body. Since there may also be a transfer of force at the direct contact between first bodies in the contact zone, the sum of forces at the debris locations (the black dots in Figure 3.11) did not necessarily equate to the sum of nodal forces, i.e. force can be either transmitted directly between first bodies or through the debris particles. By this method, the force on existing debris particles

was influenced most strongly by nearest neighbouring nodes. After the updated set of forces acting on the first body by debris particles were resolved, force statements were mapped back into the FE phase.

3.3 Summary and Conclusion

In this section, the executive procedure of the FDEM fretting wear model has been reviewed. After the simulation job of a physical fretting cycle was completed in FE phase (Section 3.2.5), the simulation results of material evolution parameters from FE were directly passed to the “main control script” to be processed (Section 3.2.7). The solid geometry was updated via the material evolution criterion through the “main control script” which also ascertained the number of debris particles and the total loading force to be simulated in the DE phase. After that, the DE phase job files were again modified, such as by updating <.stl> geometry files and adding information of those newly generated debris particles. Such process could be altered if no inherited debris particles existed in between the contact surfaces at the beginning. After the DE phase completed the simulation (Section 3.2.6), the resolved set of forces acting on the first bodies were then extracted from the DE phase and mapped into the job description of FE phase (Section 3.2.7). At the same time, the information inside the DE job description file was updated as some of the debris particles initially generated had been ejected. After that, the updated FE job was sent to the FE solver to be solved. As a result, the forces exerted on the first bodies by the particles were then fed back from the DE phase into the FE phase so that forces acting on the surface from both debris particles (and plaques) and contacting first bodies can be considered in the FE phase.

Chapter 4

Simulation Speedup and Numerical Reduction

4.1 Introduction

With the increasing calculation capability of computer analysis, more engineering problems can be simulated with computational software such as FEA software. Since one of the principal computational costs of solving FEA models is the matrix inversion process, and many FEA models are large, e.g. many thousands of degrees of freedom, they can be computationally expensive. The time taken to solve the model increases exponentially with the increasing size of the matrix [194]. Explicit solution strategies such as the proposed method require the solution of multiple steps, i.e. modelling a large number of fretting cycles before obtaining the results desired. Thus, strategies of simulation speedup or numerical reduction are often sought for large numerical models. An alternative approach to this problem is simply to apply more computing power, for example by use of High-Performance Computing (HPC). A lower financial cost approach might be the use of Graphics Processing Unit (GPU), AKA General-Purpose Computing on Graphics Processing Units (GPGPU). It is nearly always advantageous to reduce computational power requirement of a modelling method.

High-Performance Computing and GPU rely on parallel processing, which can work well for large complicated models but not so well for models with large time-spans, i.e. a time-dependent model, such as modelling fretting wear for a large number of cycles. Cycle-jumping has been widely implemented in computationally intensive as well as time-dependent models to shorten simulations. The principle of cycle-jumping is that there is no need to calculate each time step, so it may be possible to skip several cycles, saving simulation time. Normally, such strategies extrapolate results from one or a few cycles to many. Following the revision of cycle-jumping technique in Chapter Two, the performance of four different cycle-jumping methods are compared and evaluated for fretting wear prediction. It was found in this work that those four methods poorly handled noisy data, therefore a new adaptive cycle jumping was developed based on the method by Cojocaru and Karlsson [180] and

implemented herein. The new cycle-jumping strategy is then tested for suitability for the fretting wear model.

4.2 Evaluation of Four Cycle-jumping Strategies

In the following sections, four different cycle-jumping strategies are evaluated. As it is not the FDEM model but the cycle jumping methods that are tested, experimental data rather than numerical model outputs were preferred as the computational cost of modelling restricts the size of samples available for testing. Furthermore, both model predictions (Figure 4.1(a)) and observed results (Figure 4.1(b)) behave similarly, as can be seen in their comparison shown in Figure 4.1. In Figure 4.1, the wear rates predicted by the numerical model and observed in fretting wear experiments are presented.

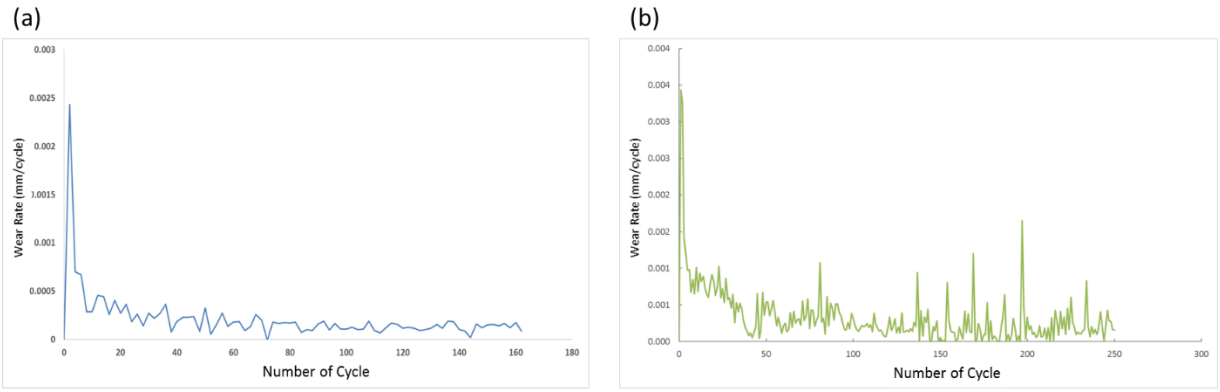


Figure 4.1. (a) The result predicted by a fretting wear model using the Dissipated Energy method which is detailed in the following chapter; (b) An example of fretting wear data obtained from experiment.

4.2.1 Method with a Constant Jumping Length

Following the cycle-jumping method utilized in work by McColl et al.[59], a constant jumping length $(\Delta N = 20)$ was used which assumed the wear rate predicted before applying cycle-jumping was constant during the following $(\Delta N - 1)$ cycles. During the implementation of this method, the fretting wear rate (i.e. wear depth per cycle) was fed into the cycle-jumping algorithm which then calculated the total wear depth over the ΔN cycles as shown in Equation 2.20. In Figure 4.2, the result predicted via such method is presented together with the result without using cycle-jumping. It was found that the result predicted by this cycle-jumping method started to diverge from the actual result after the transition phase of fretting wear. A relative difference of 50% was found at the 1000th cycle which would become worse in the subsequent predictions and lead to the loss of

accuracy. The relative difference was caused by applying cycle-jumping without checking the linearity especially at the transition phase.

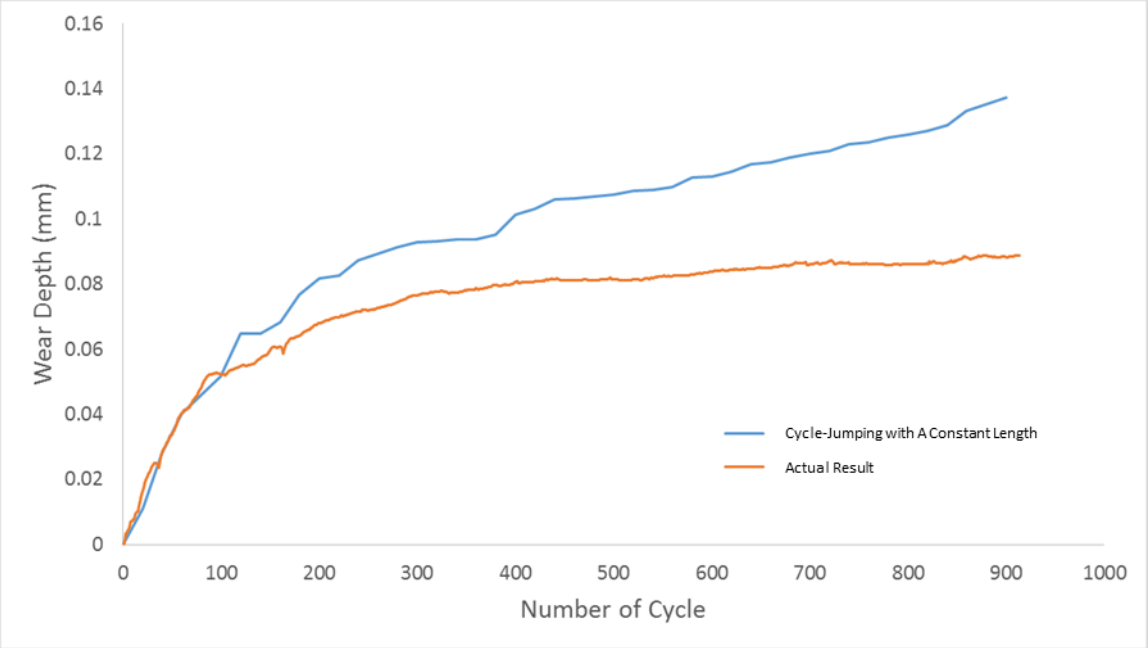


Figure 4.2. Comparison between actual result (shown in orange) and that predicted using the cycle-jumping method with a constant jumping length (shown in blue).

4.2.2 Method by Cojocaru

The cycle-jumping method applied by McColl et al. [59] failed to capture the nonlinear fretting wear because it applied linear extrapolation at the nonlinear transition phase. The adaptive cycle-jumping method used by Cojocaru is able to identify the non-linear part and only execute cycle-jumping when linear behaviour is detected[180]. Cojocaru's method includes two steps: 1) to identify the linearity of the target parameter; 2) to implement cycle-jumping if a linear behaviour is detected. During the first step, this method uses two pairs of neighbouring data, i.e. n and $n+1$ as one pair, and $n+1$ and $n+2$ as the second pair as displayed in Figure 4.3. The slope of each pair (i.e. $s_{(n)\text{ to }(n+1)}$ and $s_{(n+1)\text{ to }(n+2)}$) is calculated as shown in Figure 4.3. Afterwards, the ratio of the slope between the two pairs of data (i.e. $\frac{s_{(n+1)\text{ to }(n+2)}-s_{(n)\text{ to }(n+1)}}{s_{(n)\text{ to }(n+1)}}$) is calculated and compared against a user-defined threshold value (q_y) below which it is considered to be "linear" and cycle-jumping is permitted as shown:

$$\frac{s_{(n+1)\text{ to }(n+2)}-s_{(n)\text{ to }(n+1)}}{s_{(n)\text{ to }(n+1)}} \leq ? \, q_y \tag{4.1}$$

If the ratio of the slope is smaller than the threshold, the data is then considered to be linear. Ideally, this will detect a significant change from grossly linear behaviour. The larger the ratio of the slope, the less linear it is. On the other hand,

the smaller the ratio of the slope, the more linear it is. Before applying the second step, the jumping length Δx is then calculated via:

$$\Delta x = [q_e] \Big/ \left[\frac{s_{(n+1) \text{ to } (n+2)} - s_{(n) \text{ to } (n+1)}}{s_{(n) \text{ to } (n+1)}} \right] \quad (4.2)$$

Where q_e is a user-set value defining the maximum tolerance of slope changed within the cycles jumped Δx, i.e. how far ahead the user is willing to jump. A smaller ratio $\frac{s_{n+1 \text{ to } n+2} - s_{n \text{ to } n+1}}{s_{n \text{ to } n+1}}$, i.e. the data behaves more linearly, would generate a larger jumping length, while a bigger ratio, i.e. the data behaves less linearly under the tolerant limit, would generate a smaller jumping length.

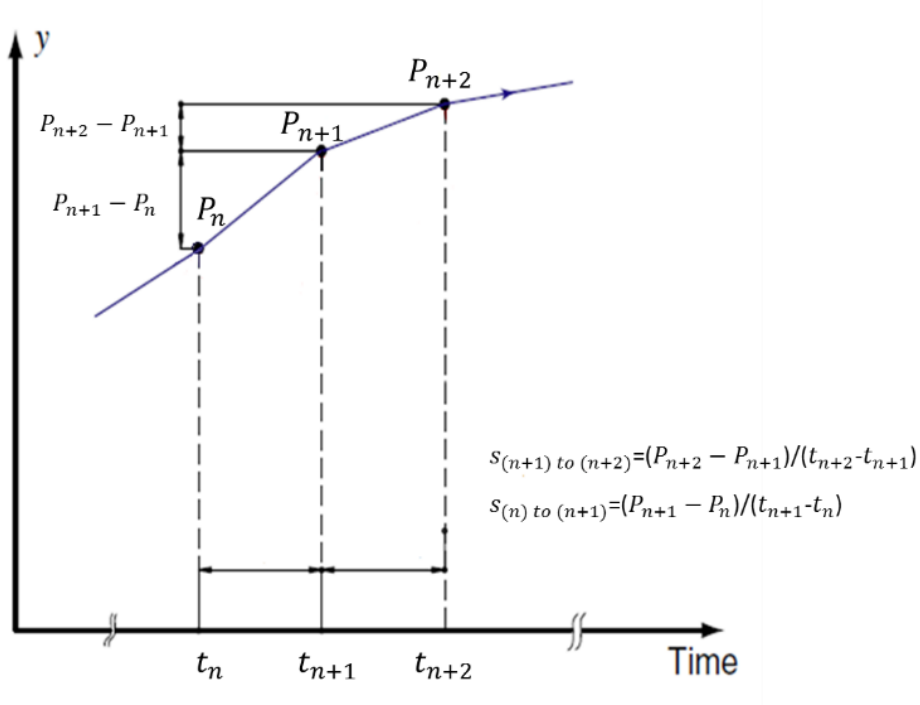


Figure 4.3. Demonstration of the adaptive cycle-jumping method by Cojocaru [2]. Two adjacent gradient ratios are calculated to be compared with each other so as to know the linearity of the curve.

After applying the cycle-jumping method by Cojocaru (with q_y equals to 0.02 and q_e equals to 0.05), it was found that both lines overlap each other with no cycle-jumping occurring during the simulation. Figure 4.4 illustrates the history in-cycle ratio of slope calculated by the algorithm. Ideally, a group of big ratio of slope was expected at the early stage of fretting wear and small ratio of slope at the later stage. As can be seen in Figure 4.4, the diagram about the ratio of slope displays a series of big value and noisy data which indicates the failure of this cycle-jumping method to identify the linear from nonlinear. As a result, this method by Cojocaru failed to apply cycle-jumping in fretting wear prediction.

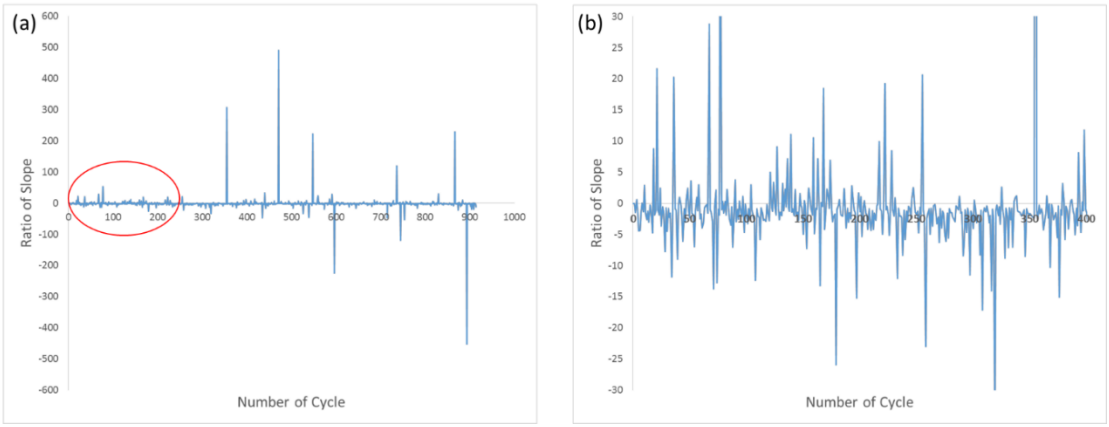


Figure 4.4. (a) The history of the in-cycle ratio of slope calculated by the cycle-jumping algorithm used by Cojocaru. (b) The enlarged view of the in-cycle ratio of slope shown inside the red cycle in (a).

4.2.3 A Moving Averages Improvement on Cojucaru's Method

As shown in the last section, the method by Cojocaru is not able to handle the wear rate during fretting wear. When the data become noisy, such a method fails to eliminate the noise and capture the behaviour of data. In this section, a modification is applied to the method by Cojocaru by expanding the size of the data sets from pairs to groups as an example shown in Figure 4.5. During the implementation of this method, linear functions were fitted to two adjacent groups of data points. The averaged gradients of two groups (i.e. $s_{(n)\ to\ (n+x)}$ and $s_{(n+x)\ to\ (n+2x)}$ for the case where x nodes are contained in each group) were then used to calculate the ratio of the slopes as shown in Equation 4.3. Similarly, a nominal threshold value (q_y) is used for linearity check below which cycle jumping is permitted. Then, the jumping length Δx was calculated via Equation 4.4.

$$\frac{s_{(n+x)\ \mathrm{to}\ (n+2x)} - s_{(n)\ \mathrm{to}\ (n+x)}}{s_{(n)\ \mathrm{to}\ (n+x)}} \leq ?\ q_y \tag{4.3}$$

$$\Delta x = [q_e] \Big/ \left[\frac{s_{(n+x)\ \mathrm{to}\ (n+2x)} - s_{(n)\ \mathrm{to}\ (n+x)}}{s_{(n)\ \mathrm{to}\ (n+x)}}\right] \tag{4.4}$$

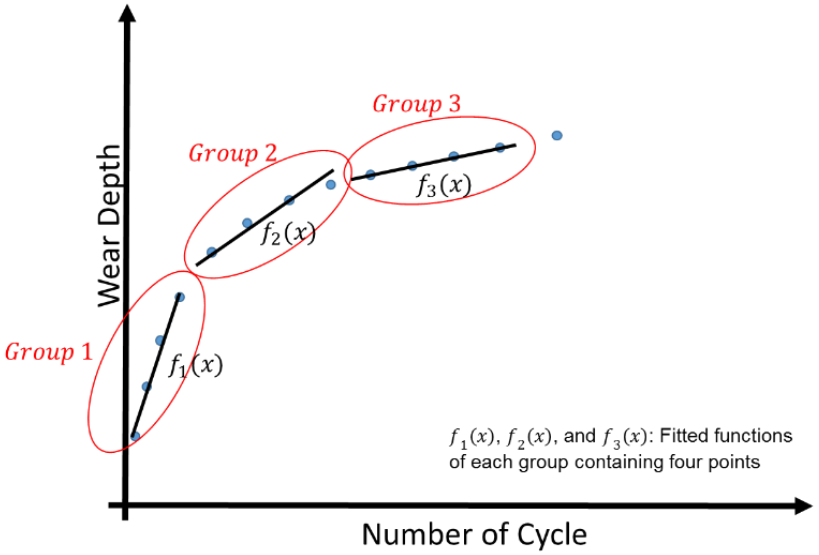


Figure 4.5. An example of a moving averages improvement on Cojucaru's method with four data points included in each group.

The key to this method is the selection of a suitable size x for the groups, a threshold q_y for linearity check and a maximum tolerance value q_e. Generally speaking, increasing the size of the groups allows better elimination of noise but also loses data features. Increasing the value of either q_y or q_e relaxes the condition of cycle-jumping which allows more jumping. In this work, three combinations of different values x, q_y and q_e were tested and compared to the result obtained without cycle-jumping. Different group sizes, i.e. 8, 10, 12, 14, 16, were tested with q_y and q_e varying from 0.5 to 1 and 0.1% to 0.2% respectively. Inside Table 4.1, the performance of this method with the three different combinations of parameters is presented. Three sections (i.e. red, green, and blue section) are included in the table, each of which stands for one combination of different parameters: a series of group sizes x, a threshold q_y for linearity check and a maximum tolerance q_e. Inside each section, two key indicators of performance are presented: the total number of step jumped and the relative difference compared to the result obtained without using cycle-jumping. As the results presented inside the red section, with the values of q_y and q_e equal to 0.5 and 0.1 respectively, the algorithm did not apply any cycle-jumping. Such result was caused by the strict condition which did not permit any cycle-jumping. With slightly bigger values of q_y or q_e, as shown in the green and blue section respectively, the algorithm was able to apply a few jumping steps at the cost of increasing relative difference. An exception was found in the cases where the sizes of the groups equalled to 12, 14 (between green and blue section) and 16

(between green and blue section): when the restriction was relaxed (i.e. either the value of q_y or q_e was increased), the total number of step jumped was not increased.

Table 4.1. A table about the indicators of the performance of the algorithm

Group Size	Linearity Check q_y: 0.5 Tolerance q_e: 0.1%		Linearity Check q_y: 1 Tolerance q_e: 0.1%		Linearity Check q_y: 0.5 Tolerance q_e: 0.2%	
	Step Jumped	Relative Difference(%)	Step Jumped	Relative Difference(%)	Step Jumped	Relative Difference(%)
8	0	0	16	1.08%	8	2.03%
10	0	0	0	0	10	1.56%
12	0	0	0	0	0	0
14	0	0	14	0.94%	0	0
16	0	0	16	2.82%	0	0

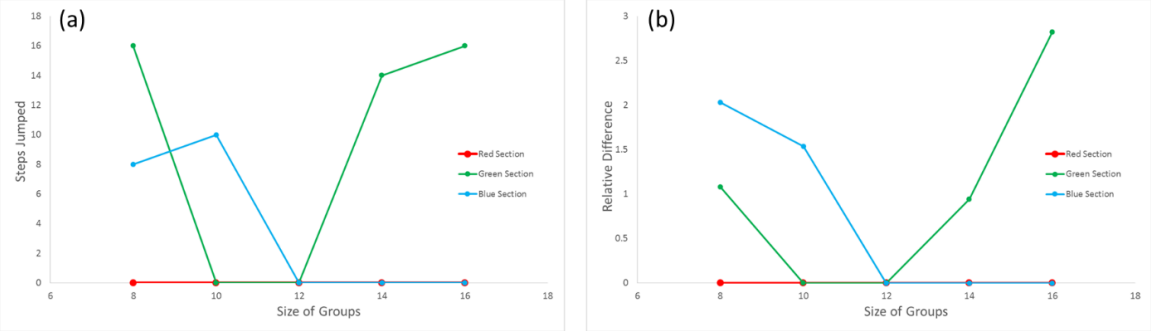


Figure 4.6. (a) The value of steps jumped by the method with different user-defined parameters. (b) The evolution of relative difference obtained using different user-defined parameters.

In Figure 4.6, the total number of cycles jumped with corresponding relative difference obtained is plotted against the size of groups which is expected to indicate the most suitable combination of parameters. As can be seen in Figure 4.6, there is no obvious relation found between the steps jumped and the size of the group: the total steps jumped was reduced to 0 when the size of groups was increased from 8 to 12 but then dramatically increased when the size continued to increase. Figure 4.7 illustrates the history of the in-cycle ratio of slopes calculated by this cycle-jumping method with different size of groups. Ideally, a group of the big ratio of the slope was expected at the early stage of fretting wear and small ratio of slope at the later stage. In Figure 4.7(a) and (b), a small peak of the ratio of slopes were found around the 250th cycle. However, the algorithm was not able to tell the difference between this peak and other ones obtained in the subsequent cycles. Thus, because there was no obvious large value of the ratio of slopes at the early stage as expected which means this method fails to identify the nonlinear of fretting wear. In addition, as shown in Figure 4.6, it is impossible to ascertain a combination of x, q_y and q_e which could be applied universally to every group of data. A particular combination of x, q_y and q_e can

work well with one group of fretting wear data offering little relative difference but also might produce a big relative difference for another group.

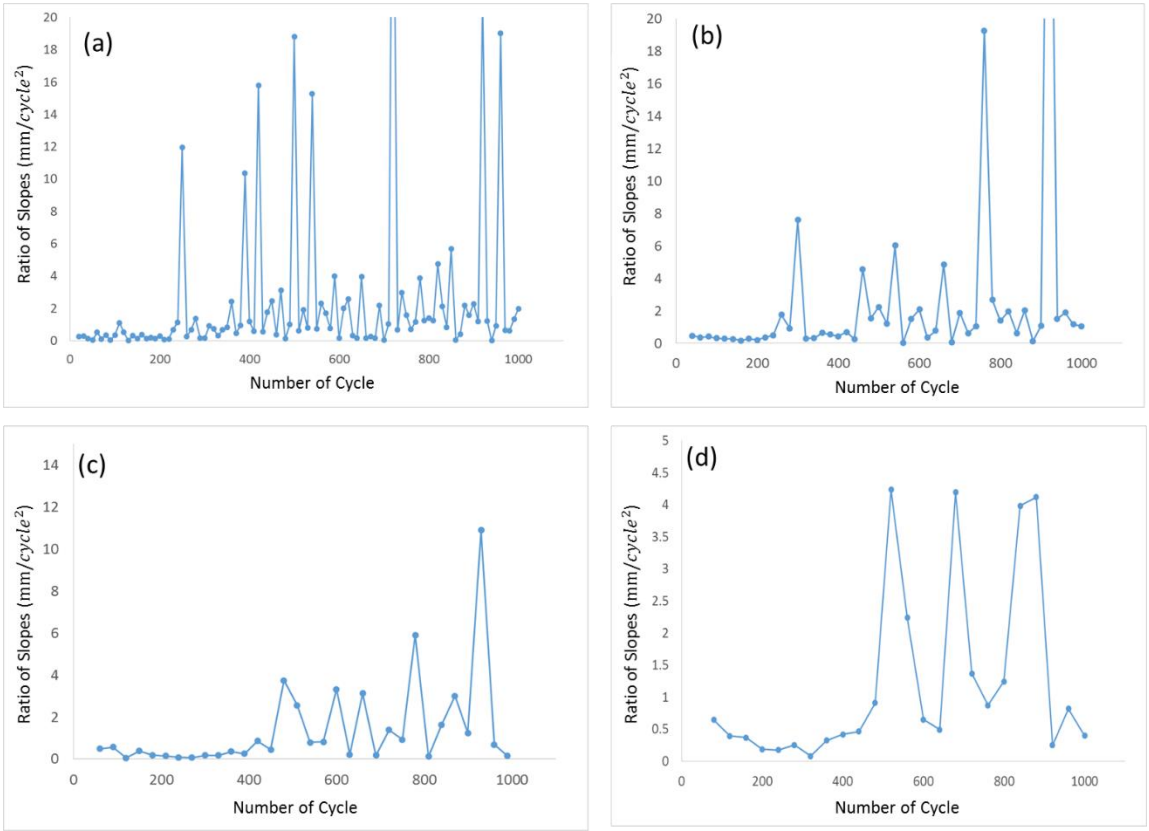


Figure 4.7. The evolution of ratio of sthe lopes calculated by the cycle-jumping method with different size of groups: (a) 10 data points were included in each group; a) 20 data points were included in each group; a) 30 data points were included in each group; a) 40 data points were included in each group.

4.2.4 An Overlapping Averages Improvements on Cojucaru's Method

The previous method succeeds to produce a peak of the ratio of slopes (as shown in Figure 4.7(a) and Figure 4.7(b)) but failed to identify it as it was mixed with other noisy data. In this section, further modification to the previous method is to have these groups abutting and exclusive or overlapping each other by half as shown in Figure 4.8. Thus, the ratio of slopes could be "smoothed" which is expected to increases the continuity between two adjacent groups and better describe the fretting wear data. The ratio of the slopes (e.g. $s_{(n) \text{ to } (n+x)}$ and $s_{(n+0.5x) \text{ to } (n+1.5x)}$ for the case which contains x nodes in each group) between two groups of data is calculated via Equation 4.5. Similarly, a nominal threshold value (q_y) is used below which cycle-jumping is permitted. Then, the jumping length Δx is calculated via Equation 4.6.

$$\frac{s_{(n+0.5x) \text{ to } (n+1.5x)} - s_{(n) \text{ to } (n+x)}}{s_{(n) \text{ to } (n+x)}} \leq ? \, q_y \tag{4.5}$$

$$\Delta x = [q_e] \Big/ \left[\frac{s_{(n+0.5x) \text{ to } (n+1.5x)} - s_{(n) \text{ to } (n+x)}}{s_{(n) \text{ to } (n+x)}} \right] \tag{4.6}$$

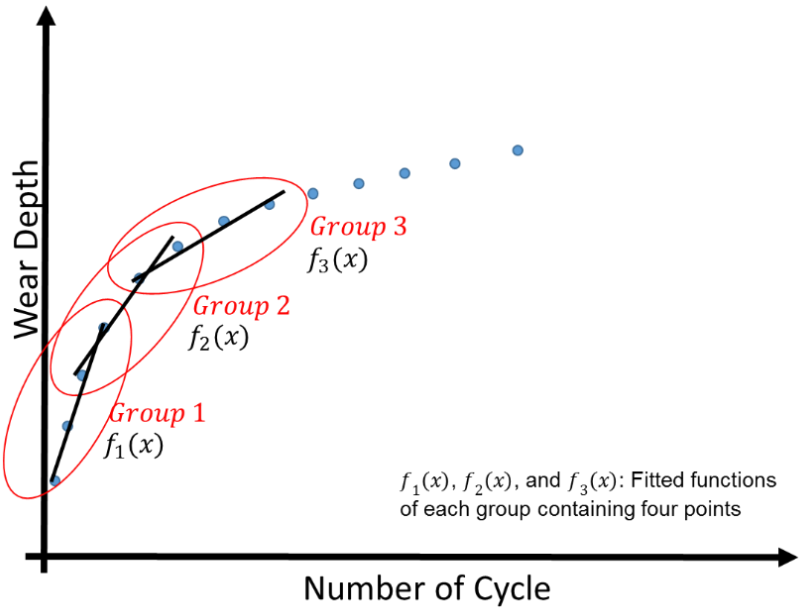


Figure 4.8. An example of an overlapping moving averages improvement on Cojucaru's method with four data points included in each group. Between two adjacent groups, two points are shared.

This method with three combinations of different values x, q_y and q_e were also tested. The size of the groups increased from 8 to 18 with different q_y and q_e varying from 0.5 to 1 and 0.1% to 0.2% respectively. Table 4.2 presents the indicators of performance with three combinations of parameters. As can be seen in Table 4.2, this method was able to apply more cycle-jumping compared to the previous method because of the better description of data as well as minimising the noisy effect. In addition, the maximum relative difference was found to be 4.05% which fall within a safe range. In Figure 4.9, the total number of cycles jumped with corresponding relative difference obtained is plotted against the size of groups which is expected to indicate the most suitable combination of parameters. Again, it is impossible to identify the best size of groups. Underlying that problem, it is impossible to ascertain a combination of x, q_y and q_e which could be applied universally to every group of data.

Table 4.2. A table about the indicators of the performance of the algorithm

Group Size	Linearity Check q_y: 0.5 Threshold q_e: 0.1%		Linearity Check q_y: 1 Threshold q_e: 0.1%		Linearity Check q_y: 0.5 Threshold q_e: 0.2%	
	Step Jumped	Relative Difference(%)	Step Jumped	Relative Difference(%)	Step Jumped	Relative Difference(%)
8	16	0.40	32	2.19	16	0.76
10	10	1.11	28	2.13	12	1.07
12	14	3.44	24	0.68	29	1.30
14	7	0.21	16	1.94	12	3.85
16	8	1.89	32	1.42	13	4.05
18	10	1.01	15	1.40	7	1.01

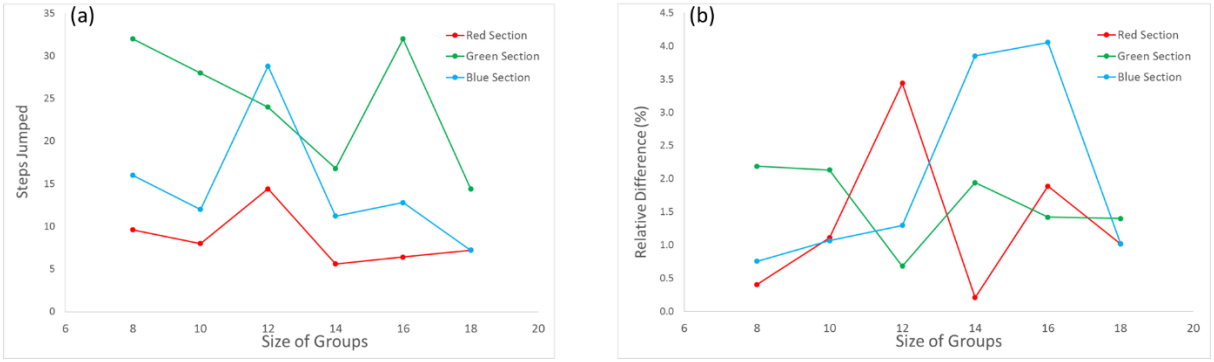


Figure 4.9. (a) The value of steps jumped by the method with different user-defined parameters. (b) The evolution of relative difference obtained using different user-defined parameters.

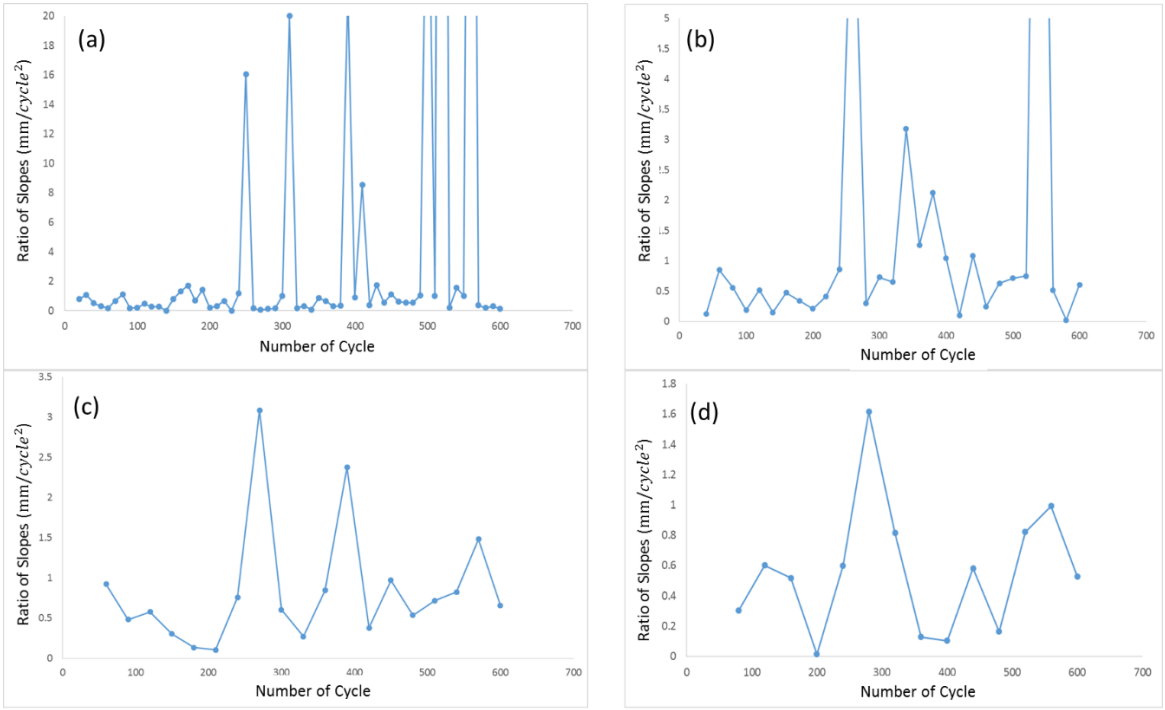


Figure 4.10. The evolution of the ratio of slopes calculated by the cycle-jumping method with different size of groups: (a) 10 data points, (b) 20 data points, (c) 30 data points, (d) 40 data points were included in each group.

Inside Figure 4.10, the in-cycle ratio of slopes generated by this method with different size of groups is presented. The Figure 4.10(c) and (d) succeed in generating a single peak around 300th cycle, i.e. the early stage of fretting wear, which indicates they can identify the non-linear transition phase where little or no

cycle-jumping was permitted. However, the value of the ratio of slopes at other cycles was not small and generated a limited number of cycle-jumping according to Equation 4.6. As a result, although a suitable size of groups can be identified inside this method, only very few of cycle-jumping was permitted.

4.3 A New Cycle-jumping Strategy

This method builds on that of Cojocaru [180]. At the start of a simulation and prior to any cycle-jumping, a user-defined number of simulation cycles are required as a foundation. Following on from this foundation a difference rule is applied which evaluates whether cycle-jumping is "safe" via a convergence test. If found "safe", a linearity tolerance check is applied to calculate the length of cycle-jumping.

The convergence test applies a user-defined function, f_i, (e.g. a 3rd order polynomial as shown in Equation 4.7) to the total in-cycle wear depth w_i and the cycle number n_i (e.g. at least 4 group of data were needed in order to derive the 3rd order polynomial function). One further simulation cycle is then performed and the same function, f_{i+1}, fitted to w_{i+1} and n_{i+1}. The relative differences of the parameter values for the two fitted functions f_i and f_{i+1} are calculated, e.g. $\mathrm{q_a}$, q_b, q_c and q_d for a third order polynomial;

$$f_i:\ w_i = a_i * {n_i}^3 + b_i * {n_i}^2 + c_i * n_i + d_i \tag{4.7}$$

$$f_{i+1}:\ w_{i+1} = a_{i+1} * {n_{i+1}}^3 + b_{i+1} * {n_{i+1}}^2 + c_{i+1} * n_{i+1} + d_{i+1} \tag{4.8}$$

$$\frac{a_{i+1}-a_i}{a_i} = q_a \tag{4.9}$$

$$\frac{b_{i+1}-b_i}{b_i} = q_b \tag{4.10}$$

$$\frac{c_{i+1}-c_i}{c_i} = q_c \tag{4.11}$$

$$\frac{d_{i+1}-d_i}{d_i} = q_d \tag{4.12}$$

where q_a, q_b, q_c and q_d are relative difference. If the value of any relative difference exceeds a threshold, cycle-jumping is considered "unsafe" and not implemented as shown in Figure 4.11(a), with only a few data for clarity. The threshold value of relative error used in this work was 2%. It is clearly possible to implement a wide variety of rules here e.g. a constant 2% threshold or a varying threshold etc. In effect, this is a check on the convergence of the functions. This is different from previous cycle-jumping procedures is not a linearity check.

If a converged function is found, i.e. it is “safe” as shown in Figure 4.11(b), the maximum allowable jumping length is calculated via a linearity tolerance check. More specifically, the second derivative of the fitted polynomial function, $f''_{(x)}$ is compared to a user-defined value q_y:

$$f''_{(x)} <? q_y \tag{4.13}$$

The fitted polynomial function is considered “linear” if the second derivative of the function is smaller than q_y. The linearity check after each new cycle continues until the degree of relative difference is reduced to a user-defined level which would permit cycle-jumping. In the work described in this chapter, the polynomial functions were fitted to data going back to the first cycle throughout the simulation run. If desired for computational cost reasons, it would be possible to use fewer cycles through a moving window for this process.

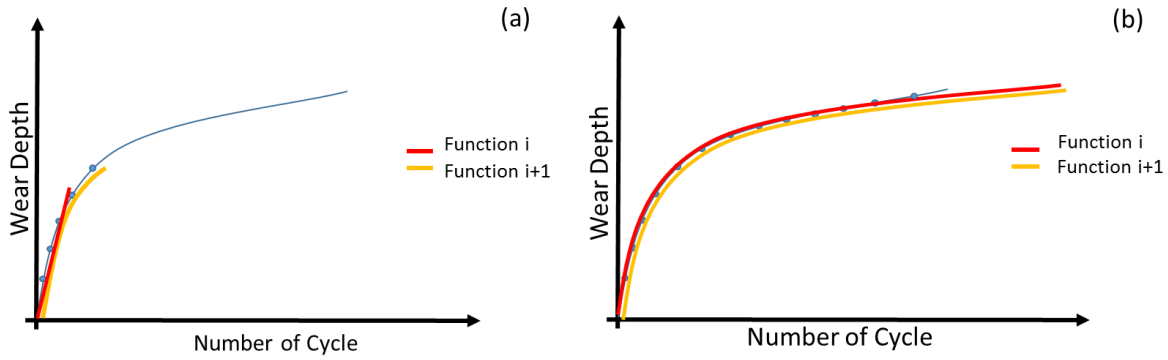


Figure 4.11. A demonstration of the adaptive cycle-jumping technique. The blue curve presents the simulated fretting wear behaviour without jumping, the red curve shows the best-fitting function fi and the yellow curve the function i+1. (a) Cycle-jumping is nor allowed as function f_i and f_{i+1} are dramatically different; (b) Cycle-jumping is permitted when the function f_i and f_{i+1}have little difference.

If the linearity check is passed, i.e. $f''_{(x)}$ is small enough, the jumping length Δx (i.e. the number of cycles which could be skipped) is calculated via:

$$\Delta x = [q_e \times f'(x_{end})] / f''_{(x)}) \tag{4.13}$$

The larger the second derivative $f''_{(x)}$, i.e. the less linear it is, the smaller the jumping length (Δx) (the number of cycles to be skipped). On the other hand, the smaller the second derivative $f''_{(x)}$, i.e. the more linear it is, the bigger the jumping length (Δx). The aim is to minimise the difference between runs with and without cycle-jumping. The cost of using a larger value of q_e, speeding up the simulation, would be potentially larger errors.

Once the jumping length is obtained, extrapolation is implemented to calculate the value after jumping $f(x + \Delta x)$ following the principle of the Taylor Expansion as shown:

$$f(x+\Delta x) = f(x) + f'(x) \times \Delta x + \frac{f''(x)}{2!} \times \Delta x^2 + \cdots + \frac{f^n(x)}{n!} \times \Delta x^n + \cdots \quad (4.14)$$

where $f^n(x)$ is the n^{th} derivative of $f(x)$. By omitting the terms of the higher orders, the following equation can be obtained:

$$f(x+\Delta x) \approx f(x) + \Delta x f'(x) \quad (4.15)$$

where $f(x+\Delta x)$ was the value after applying cycle-jumping, $f(x)$ was the value before applying cycle-jumping. Although keeping more terms inside the Taylor Expansion produces results with higher accuracy, the result obtained via current setting was found to be satisfactory.

Following the cycle-jumping, the predicted value $f(x+\Delta x)$ is calculated and used as input for a run of the full model. Following a user-defined number of these individual cycles, the jumping process of checking for linearity and calculating the jumping length is initiated again. The procedure ends upon reaching a user-defined number of cycles or if a trigger threshold value of an output is reached.

4.4 Test and Modification

In this section, the new cycle-jumping strategy developed herein was tested with preliminary results presented. By default, a third order polynomial function was used to ascertain a converged function. A constant value of difference (q_a, q_b ,q_c , and q_d) used in this work is 2% with a threshold value for linearity check q_y equalling to 0.5% and q_e equalling to 2%. It is clearly possible to implement a wide variety of rules here e.g. a constant threshold or a varying threshold etc. In Figure 4.12, the ratio of slopes calculated by the cycle-jumping code at each cycle is plotted. A single and clear peak was found around the 150th cycle which suggested that the algorithm was able to identify the non-linear transition phase from the rest linear phase. A small jumping length or no jumping is applied at the region where a large value of the ratio of slopes is found. The extremely large values of the ratio of slopes around 50th cycle was noisy data at which time the converged function had not generated.

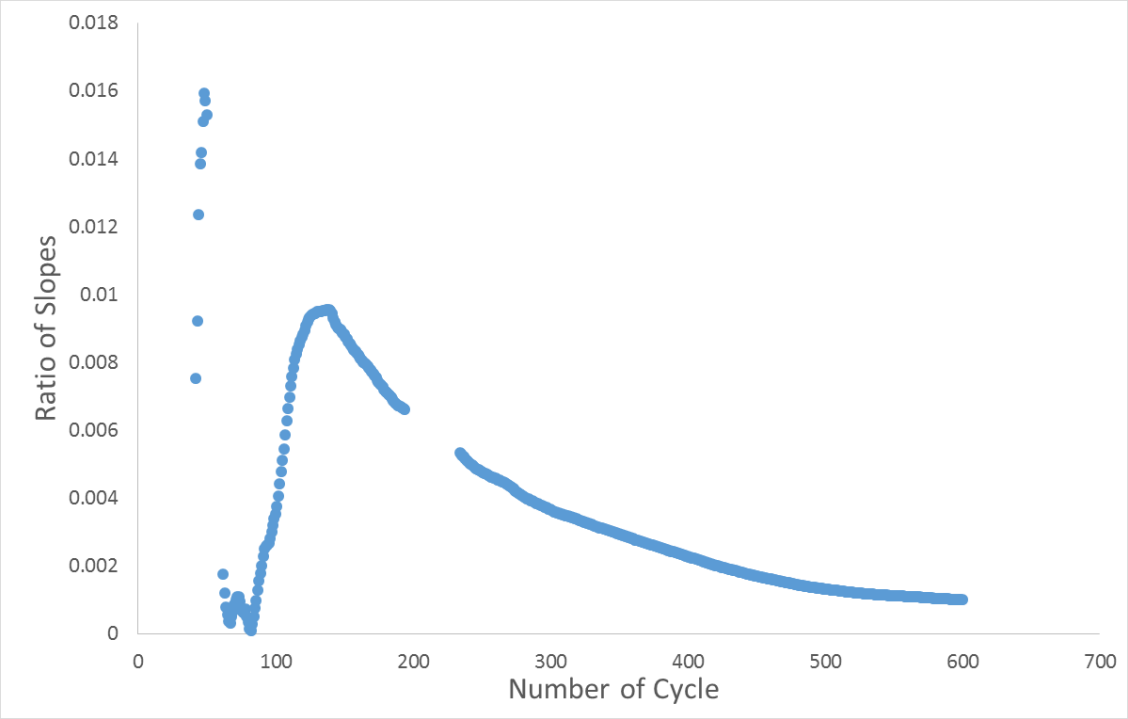


Figure 4.12. The evolution of ratio of slopes calculated by the new cycle-jumping strategy.

In Figure 4.13, the fretting wear predicted using this cycle-jumping method is displayed, inside which a huge jump is generated around the 50th cycle. This was caused by the extremely low value of $f''_{(n)}$ as displayed in the early part of Figure 4.12. Applying such a large jump missed the highly non-linear phase transition around the 100th cycles. Therefore, an additional restriction is required to limit the maximum jumping length, especially in the initial stage of fretting wear when non-linear behaviour observed.

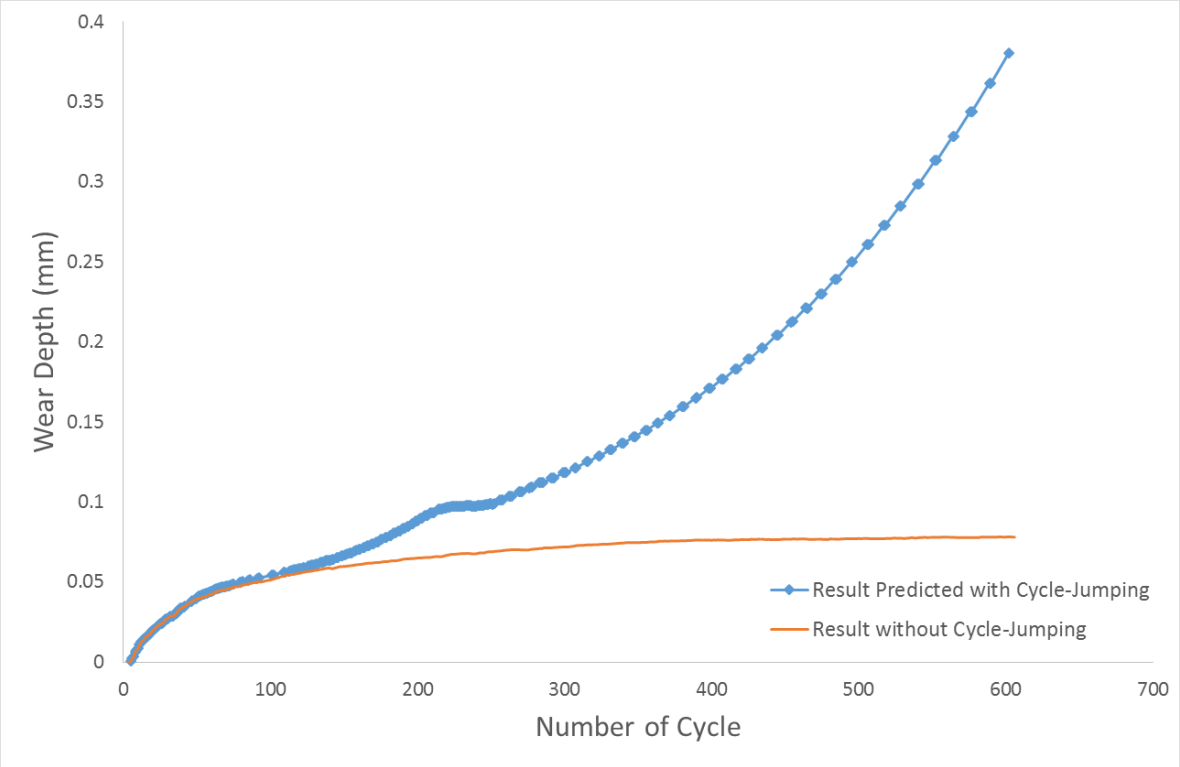


Figure 4.13. A comparison of the preliminary result produced by the new cycle-jumping strategy with default parameters (shown in blue) and result without cycle-jumping (shown in orange).

A series of frequent small jumps were observed after the 200th cycle by which the relative difference between the results with and without cycle-jumping was accumulated as displayed in Figure 4.14. It was found that those tiny and frequent jumps built up and finally lead to loss of accuracy and incorrect predictions. Thus, an additional modification is required on limiting the maximum number of consequent jumps, specifically, when the maximum number of consequent jumps is reached, the subsequent cycle-jumping is not permitted until some criteria have been reached.

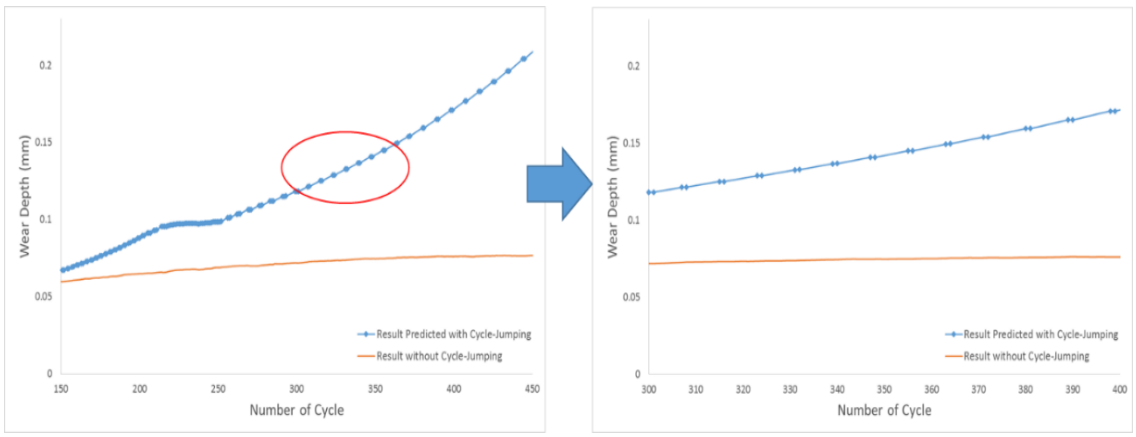


Figure 4.14. An enlarged view of the comparison in the range from the 300th to 400th cycle.

The model could also benefit from other improvements, such as: 1) applying a higher order polynomial function. Although it would allow a better fit of the data

and implementation of better cycle-jumping, it would incur additional computational cost in order to ensure convergence of the function; 2) increasing the number of simulation cycles before generating the polynomial function. With more data received as a foundation, it becomes faster to generate a converged polynomial function. For instance, before fitting a third order polynomial function, 8 simulation cycles could be simulated whose results could then be fed into the algorithm.

According to the problems identified above, three modifications were applied to the cycle-jumping algorithm: 1) applying a maximum jumping length. Specifically, a maximum jumping length was initialised to be 15 before 200 cycles, i.e. any predicted jumping length longer than 15 cycles was reduced to 15, and then gradually increased afterwards; 2) applying a decreasing q_y. When the ratio of slopes becomes small at the later stage of fretting wear, the possibility of generating large cycle-jumping is increased under the initial condition which can lead to incorrect prediction. Therefore, one more strict condition was implemented at the later stage of fretting wear: reducing the value of the threshold q_y for linear checking. As a result, it took much more data than the previous algorithm to generate a converged function; 3) applying a maximum number of frequent cycle-jumping. When a series of frequency cycle-jumping was detected, cycle-jumping was not permitted until a user-defined number of new input data was fed into the algorithm to generated a new converged function; 4) A moving window strategy was applied here. When a series of frequent jumps were detected, e.g. three big cycle-jumping was executed continuously, it was assumed that the noisy data was accumulated and the current converged function became inaccurate. Then, the algorithm would abandon the old polynomial, started to collect data from the present number of cycle so as to form a new converged polynomial function. In Figure 4.15, a maximum jumping length was defined to be 15 before the 200th cycle. This version of the algorithm was able to save 70% of the cycles with 32% relative difference compared to the data without using cycle-jumping. Figure 4.16 displayed the result predicted by this cycle-jumping strategy with a deceasing threshold q_y and a dynamic maximum jumping length, i.e. based on the number of cycle. The maximum jumping length was initialized to be 200 and gradually increased with the progress of fretting wear. A much smaller threshold (i.e. $q_y = 0.02\%$) for a linear check was applied to the later stage of fretting wear. This cycle-jumping strategy was able to save

around 60% of the cycles with 10% of relative difference compared to the data without cycle-jumping.

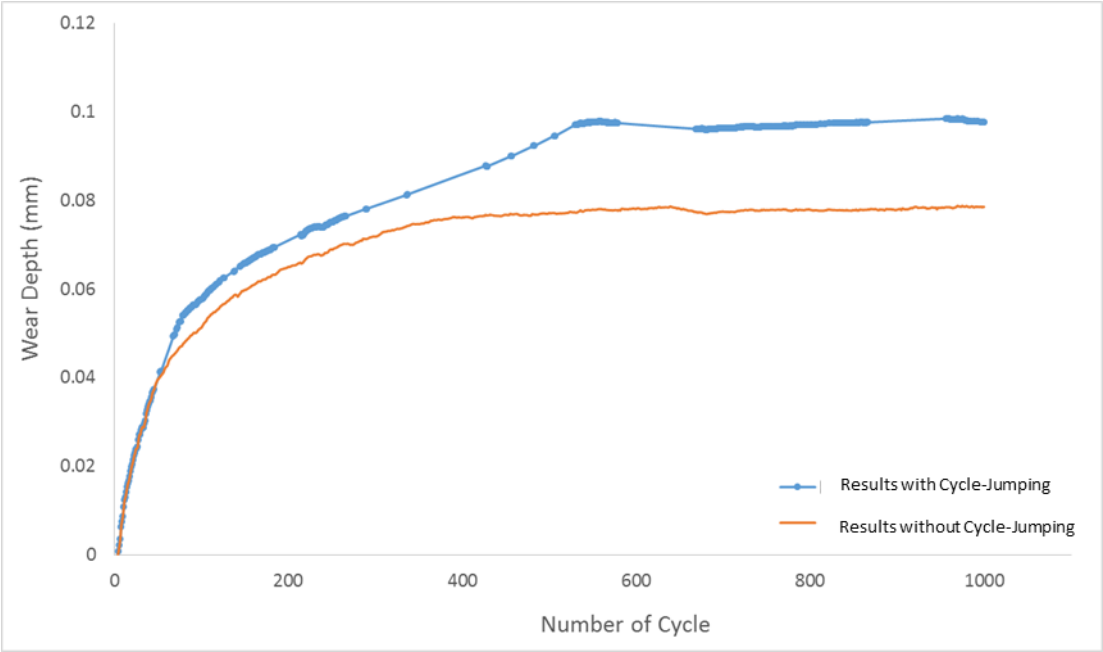


Figure 4.15. Comparison between predicted and actual wear data using the improved algorithm.

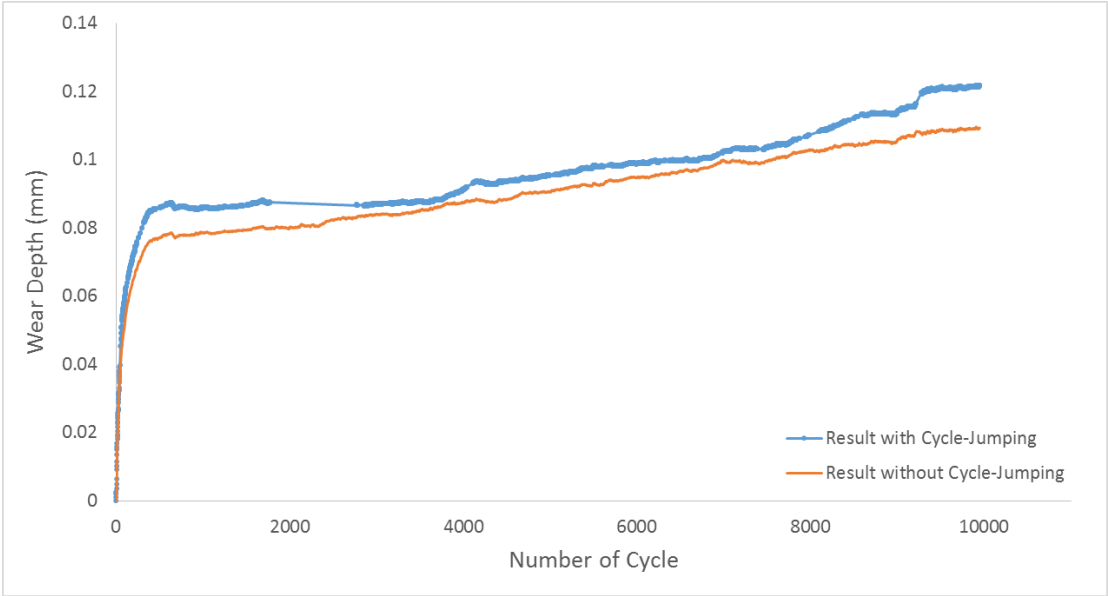


Figure 4.16. Comparison between predicted and actual wear data using the improved algorithm.

4.5 Conclusion

In this chapter, different cycle-jumping techniques were reviewed. It was found that most cycle-jumping techniques failed to capture the nonlinear of fretting wear data and were unable to produce an efficient cycle-jumping performance. The method by Cojocaru was chosen inside this work as it was able to identify the non-linear part of the data and stop cycle-jumping so that the transition phase was not missed. Similarly, an improvement was required as the method by Cojocaru was not able to handle the noisy fretting wear data. In all cases, the slope of a fitted function is used to extrapolate to a cycle number thereby missing out a predetermined number of cycles.

A cycle-jumping method handling nonlinearities and with novel noise elimination technique was developed and tested in this work. This algorithm was able to greatly eliminate or minimise the effect of noisy data without losing the original data features, specifically the non-linear phase transition of fretting wear. At the same time, further improvement was applied to the algorithm. For example, a moving window strategy was employed inside the algorithm which was able to remove the accumulation of error from previous predictions. Several other modifications such as using a cycle-number dependent threshold and maximum jumping length were made to the algorithm. In one case, the improved cycle-jumping technique was able to save or avoid 60 percentages of cycles with obtaining less than 10 percentages difference compared to the result without cycle-jumping. It should be noticed that the performance could vary between cases.

More effort can be spent on adjusting each user-defined parameter in order to further improve the performance of the algorithm. However, such effort could only make the algorithm more specific to a certain group of data and lose its general capability.

Chapter 5

Study of the Effect of Debris Particles in the Contact Zone with FDEM

5.1 Introduction

Debris particles are a commonly observed feature of most fretting wear experiments [31], [32], [195]. Lower wear rates have been linked experimentally with a higher number of debris particles [138], which implies that the debris particles between first bodies have some influence on fretting wear between first bodies [31], [32]. In this section, the behaviour and effect of debris particles in a Hertizan contact system is analysed with the FDEM model to understand more details about the impact of debris on fretting wear. A comparison is made between a model ran with debris particles and ran without debris particles. In addition, because the debris particles between two first bodies are a dynamics system which can be affected by other factors such as the shape of contact zone geometry. Simulations are run at different stages of fretting wear in which the shape of the wear scar is radically changed by wear: one from an early stage of fretting wear with a relatively small contact zone, and the second one after more wear with a wider contact zone.

The fretting wear profiles implemented in this FDEM model is predicted using the Dissipated Energy method. In both model cases the worn and irregular upper first body surface is artificially replaced with a smoothed profile of the same wear depth, so as to reduce the variables in the comparison. Four simulations are run to study the effect of debris particles at different stages of fretting wear, a) the two contact zone shapes and b) for each of those with and without the presence

of debris particles. Additionally, in order to study the effect caused by a different number of debris particles, further model refinements are run in which the numbers of debris particles and thus total worn volume are altered. That is, the number of particles predicted to be present at the cycle under consideration, plus models with more and fewer particles. In this way the effect of the debris particles at a different stage of fretting wear and number of particles is evident.

5.2 Implementation of FDEM Model

Before the FDEM fretting wear model can be launched, the set-up of the model was checked. Inside the FE phase of the FDEM model, a mesh convergence test was applied to ascertain the maximum size of an element required. It was demonstrated that the maximum size of an element within the ultra-fine model section was $20\ \mu m$ for the example case of Hertzian contact used herein. The stress distribution in the FE model was in good agreement with that predicted by the standard analytical formulations for Hertzian contact [196], and are shown in Figure 5.1. The disagreement between the FE model and the analytical formulation is worst at the edge of the contact, being less than 5% of maximum pressure, which is not likely to cause problems in the prediction of fretting wear.

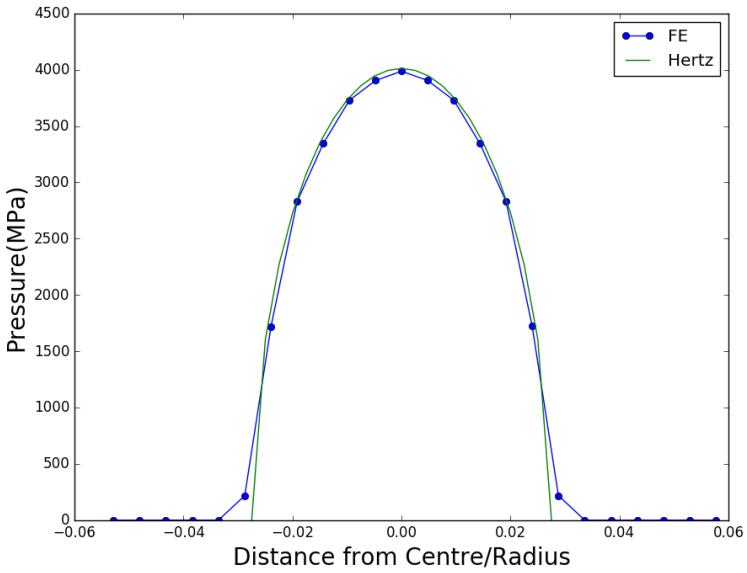


Figure 5.1. shows the predicted contact pressure from the FE model across the contact zone for a Hertzian loading case, vs the pressure from the analytical contact model by Hertz [195].

Inside the DE phase, the accuracy of the moving boundary method used to apply forces across particles in the DE phase was checked by application to the model shown in Figure 5.2(a): this application has a number of particles constrained between two rigid boundaries, the uppermost of which was displaced according to the algorithm, for the purposes of verification set at 100N total compressive force on the particles. Following the Raleigh method [158], [159], the maximum time step required for the DE phase in order for convergence was calculated to be $2.165 \times 10^{-6} ms$ (equating to $1 \times 10^{-4}\%$ of cycle time). As shown in shown in Figure 5.2(b), the output from the DE model of the sum of vertical forces on all particles in contact with the upper boundary over time reached a steady state accurate to within 5 significant figures of the target force.

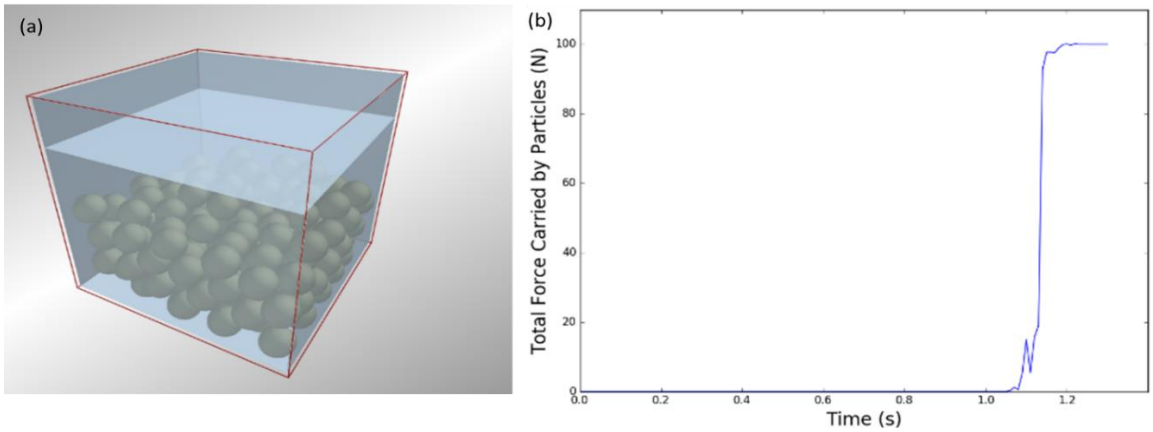


Figure 5.2. (a) Illustration of a compression model designed to test the moving boundary method; (b) Demonstration of the history of total force carried by particles when a 100N force was gradually applied by the upper flat boundary.

In the process of transferring data from FE to DE phase: using a shape function to apportion the total contact force into that between solid-solid contact and that between solid-particle contact, was found to work as intended, see Figure 5.3. Inside the Figure 5.3, the four red vertical vectors indicate nodal forces at the element nodes, which is the output from the FE phase; the green vectors indicate the local forces transmitted between solid and particles, which is an input to the DE phase. In the case where there are no particles between the contacting elements, 100% of the contact force for this element is transmitted via solid-solid interaction, as shown in Figure 5.3(a). In the case where there is a solitary particle present in the contact between elements, a small percentage of the contact force for that element is considered to be borne by the particle, as shown in 5.3(b). The shape functions ensure that as the number of particles between the contacting elements increases, the proportion of the load borne by the particles increases. For example, see Figures 5.3(c) and 5.3(d) where a significant fraction and then near 100% of the contact force is transmitted via the particles.

In the process of updating elements' coordinate: the morphing of elements in this process did not cause any distortion or aspect ratio problems within the FE phase. The detailed procedure of solving an FE model can be found in the content of Section 3.2.7. No degradation was found in the elements after being morphed

which are then successfully resolved by the FE solver. The removal of mass from the first body surfaces, resulting in new debris particles was confirmed to conserve mass between the start and end of simulations.

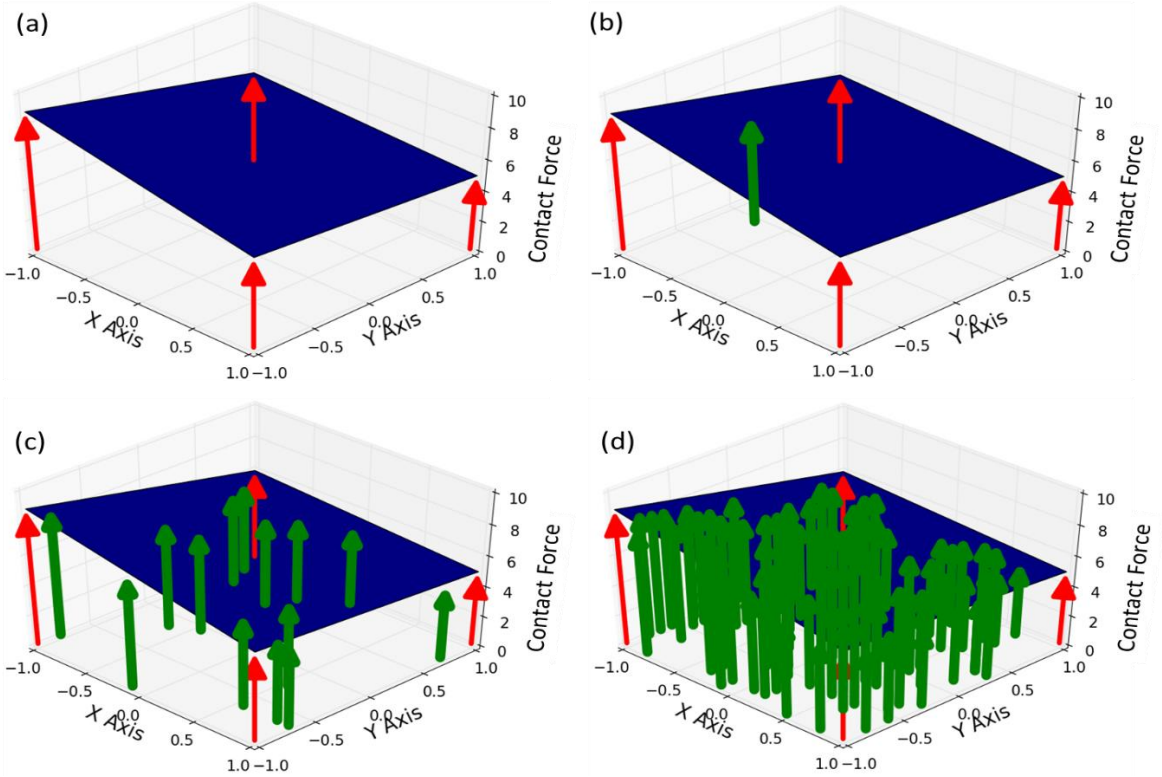


Figure 5.3. A demonstration of the algorithm, transferring data from FE to DE phase, applied to one FE element. (a) The red arrows represent the value of nodal forces at four vertices of the element. The blue surface stands for the fitted surface produced by the shape function method which shares the same value at four vertices. (b) The green arrow represents the value of force calculated at the location of the arrow (i.e. the location of debris particle in the case of this work). (c) The length of green arrow (i.e. the value of force) is mostly affected by the nearest nodal force (i.e. the red arrow). (d) With more debris particles lying upon the element, a corresponding number of green arrows are calculated.

The cycle jumping method detected non-linearity as intended, and indicated where cycle jumping ought to be minimised or avoided for the purpose of accuracy. A sample of the result obtained from the model, as shown in Figure 5.4(a), was employed to test the performance of the cycle-jumping algorithm. It can found that the sample data was nonlinear at around 30th cycle and became generally linear afterwards. The result of the algorithm, the second derivative of the sample over the fretting cycles, was presented in Figure 5.4(b). A group of the second derivative with big value was identified correctly from 20th to 60th cycle, i.e. the nonlinear or transition phase. When the value of the second derivative

was identified to be above a threshold value (in this case set at $5 \times 10^{-2} mm/cycle^2$), zero cycle jumping was allowed (data are shown in orange).

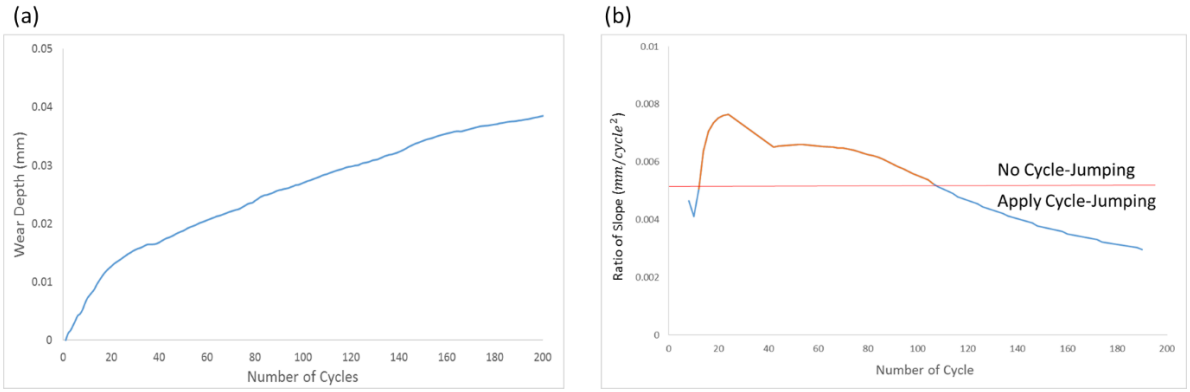


Figure 5.4. Demonstration of the test for the cycle-jumping algorithm. (a) The sample data processed by the cycle-jumping algorithm. (b) The result, produced by the algorithm, identifies the nonlinear and linear of sample data via comparing the second derivative of wear depth to a threshold 5×10^{-2}.

The value of energy wear coefficient applied in this work was obtained from experiments done by Luke Blades as part of his PhD thesis and the test results were not published at the time of writing this thesis, some of these results were used to calculate the analytical estimation, so it had to mention the data and conditions about those experiments: the fretting wear tests were set up with three perpendicularly crossed cylinders as shown in Figure 5.5(a). A bilinear material property model, which assumes a linear relationship between the stress and strain after the yield point, was applied inside the FE model here. A simple example of a bilinear material model is displayed in Figure 5.5. The properties of the material (values provided by the supplier) were presented in Table 5.1. Tests were resolved by applying a loading force on two horizontally paralleled cylinders to compress the vertical cylinder. A vertical fretting motion was exerted on the vertical cylinder via a hydraulic system. An LVDT was used to measure the wear depth. A profile scanning machine was used to estimate the total material worn volume. The total energy dissipated during fretting wear was calculated as the product of tangential friction force and displacement recorded during fretting wear experiment.

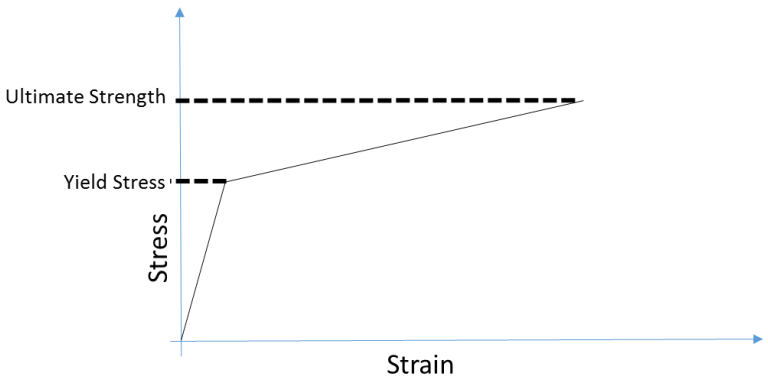


Figure 5.5. An example of bilinear material model. The first line with a higher gradient stands for the elastic deformation while the other one is the material behaviour after the yield point.

Table 5.1. The material properties used in this model [197].

Young's Modulus (MPa)	Yield Stress (MPa)	Ultimate Tensile Strength (MPa)	Hardness (HB)
240000	680	1000	302

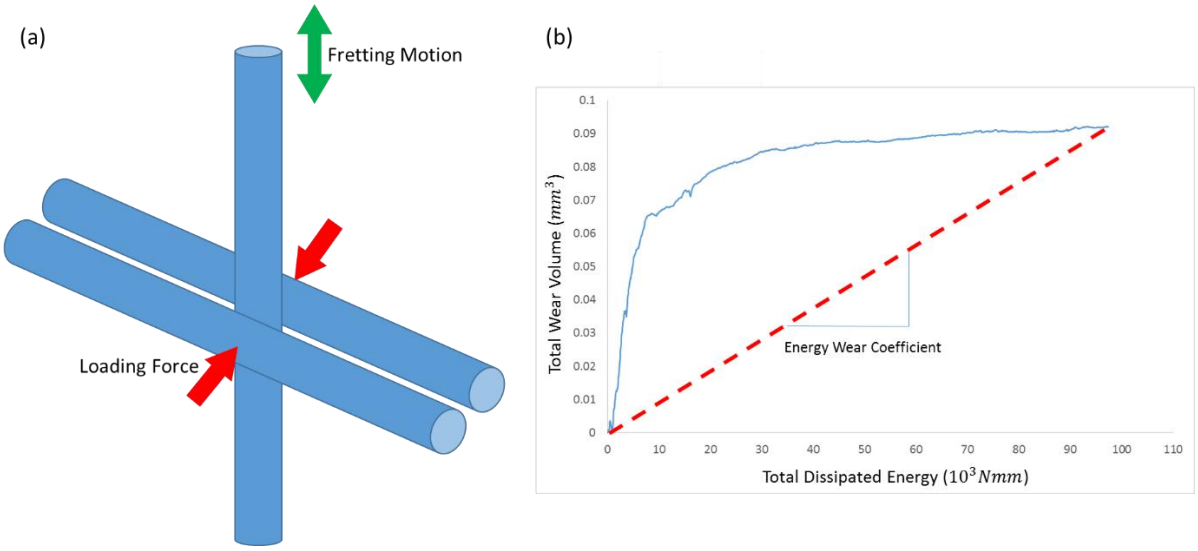


Figure 5.6. (a)Illustration of the setup of fretting wear experiment with loading force equals to 100N and fretting motion of 0.6mm. (b) The evolution of total wear volume and total dissipated energy recorded during the fretting experiment (with 1000 cycles).

Fouvry *et al.* observed a linear relationship between the material volume and total energy dissipated during fretting wear in their experiments [12]. In fact, the relationship between total wear volume and total dissipated energy using data from experiments in this thesis (Chapter 5) was strongly non-linear, see Figure 5.6(a). Such a phenomenon was also observed by Yamamoto et al.[198]: an energy wear coefficient in the range of $10^{-4} mm^3 J^{-1}$ was found during the initial

stage, with $10^{-7}mm^3J^{-1}$ in the later stabilised stage. Following the method used in the work by Fouvry, the energy wear coefficient was calculated via dividing the total worn volume by the total dissipated energy as shown in Figure 5.6(b). The value of energy wear coefficient employed in this work was ascertained to be $1.641 \times 10^{-6}mm^3/J^{-1}$ by averaging energy wear coefficients obtained from a series of fretting experiments in order to reduce human error. It should be noticed that different values of energy wear coefficient could be obtained by looking at the different stage of fretting wear as fretting wear is non-linear.

At the end of the simulation, the geometries (i.e. the first bodies) were updated according to the users' criteria whose 2-dimensional shape can be similar to that displayed in Figure 3.9 and 3-dimensional shape like a concave grid. Such shape or profile can then be used to compare with the scanning profile of the wear scar measured in fretting wear experiments. Alternatively, a comparison can be made between the wear rates calculated from the profiles predicted (details in Section 3.2.7) and measured (with LVDT). In the following sections, a history of fretting wear rate is obtained with the Dissipated Energy method, which is then compared to that measured in experiments so as to study the material evolution criteria.

5.3 Fretting Wear Predicted with the Dissipated Energy Method

In this section, the fretting wear predicted using the Dissipated Energy method is presented and compared to experimental results. The upper first body is the mobile first body as shown in Figure 5.7. The lower first body is referred as the static first body as shown in Figure 5.7.

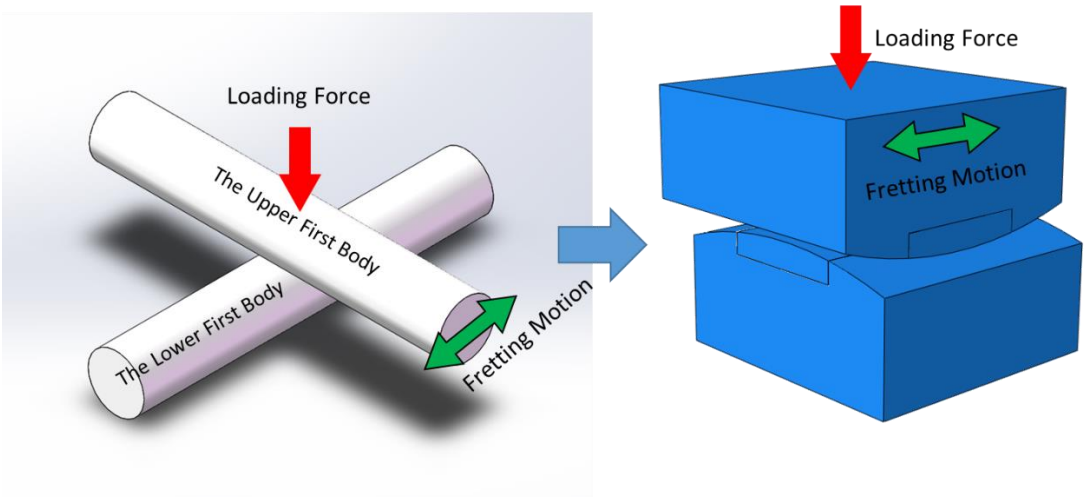


Figure 5.7. Demonstration of model simulated with a load of 100N and amplitude of 0.6mm.

5.3.1 Evolution of Contact Traction Stress and Relative Slip Displacement

The evolution of the traction stress and the relative slip displacement is of importance since their product is the dissipated energy postulated to drive fretting wear [12]. When a fretting motion was applied between first bodies as shown in Figure 5.7, the evolution of contact traction stress on both first bodies during the 1^{st} and 60^{th} fretting cycle are presented in Figure 5.8 and Figure 5.9. Because the contact traction stress is a vector, the red colour stands for the direction of the vectors to the left while the blue stands for those vectors to the right. In Figure 5.10 and 5.11, the corresponding accumulated relative slip displacement for each node is displayed. The red colour of the accumulated relative slip displacement is the displacement accumulated to the left while the blue colour stands for the displacement accumulated to the right. The left columns of Figure 5.8-5.11 show the evolution results on the lower first body and the right columns give the corresponding results on the upper first body. Both contact surfaces were predicted to be rough (see Figure 5.11 for example), i.e. contact between first bodies was occurring between points which resulted in altering the distribution of the contact traction stress and relative slip from being uniform to non-uniform as shown in Figure 5.9 and 5.11.

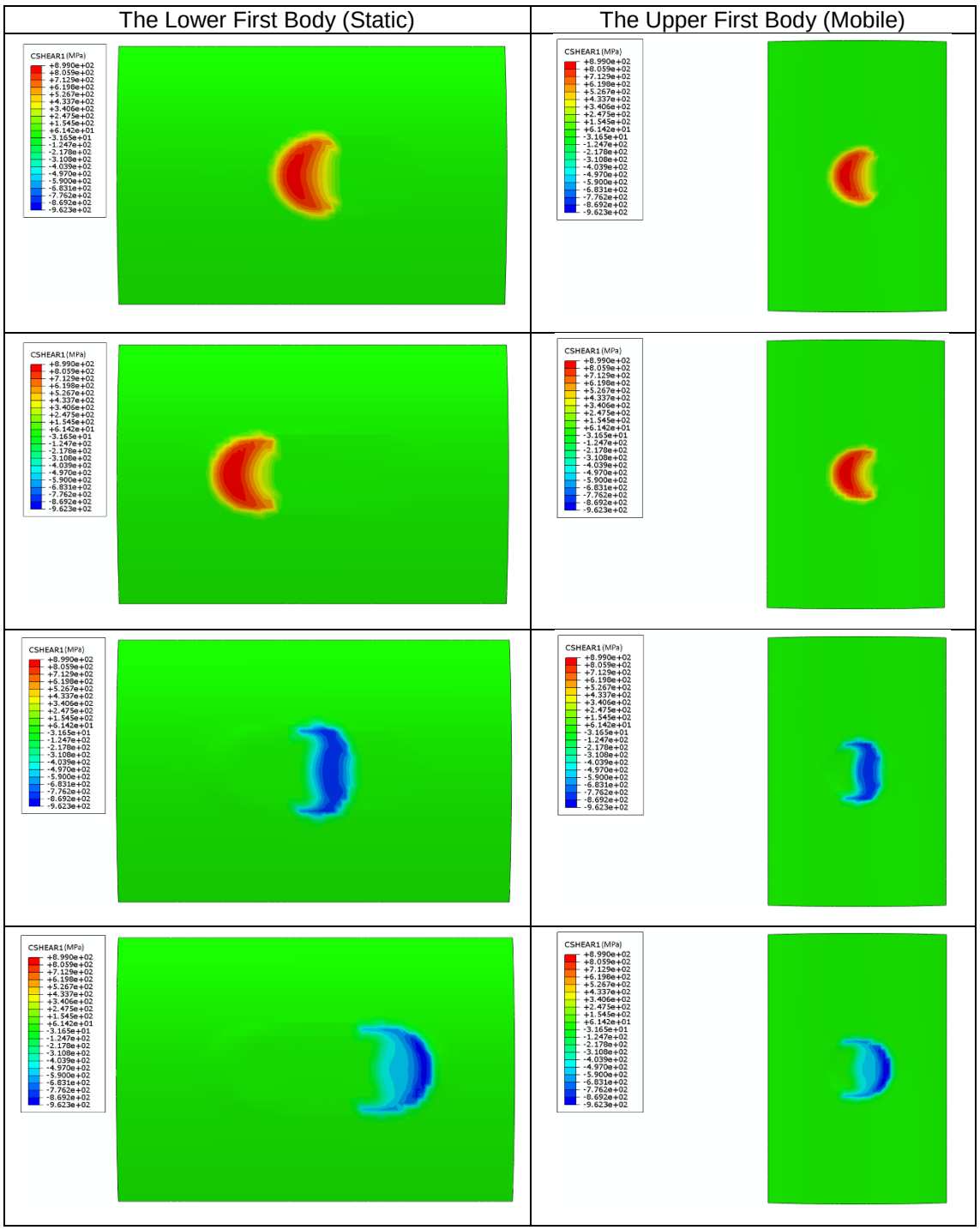


Figure 5.8. The evolution of contact traction stress on both first bodies during the 1st fretting cycle. The cyclic motion starts in the uppermost image and progresses including a reversal towards the lowest image.

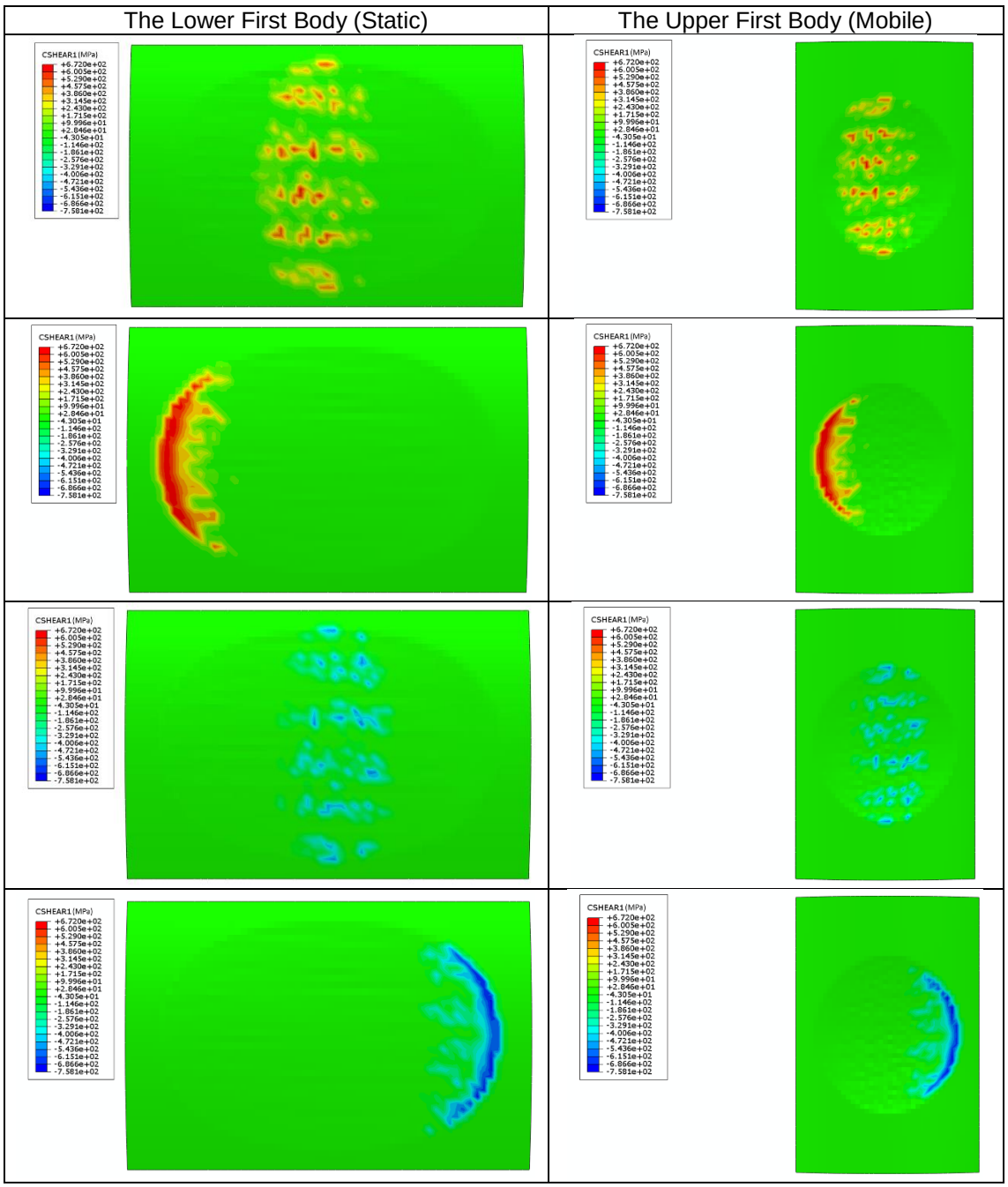


Figure 5.9. The evolution of contact traction stress on both first bodies during the 60th fretting cycle. The cyclic motion starts in the uppermost image and progresses including a reversal towards the lowest image.

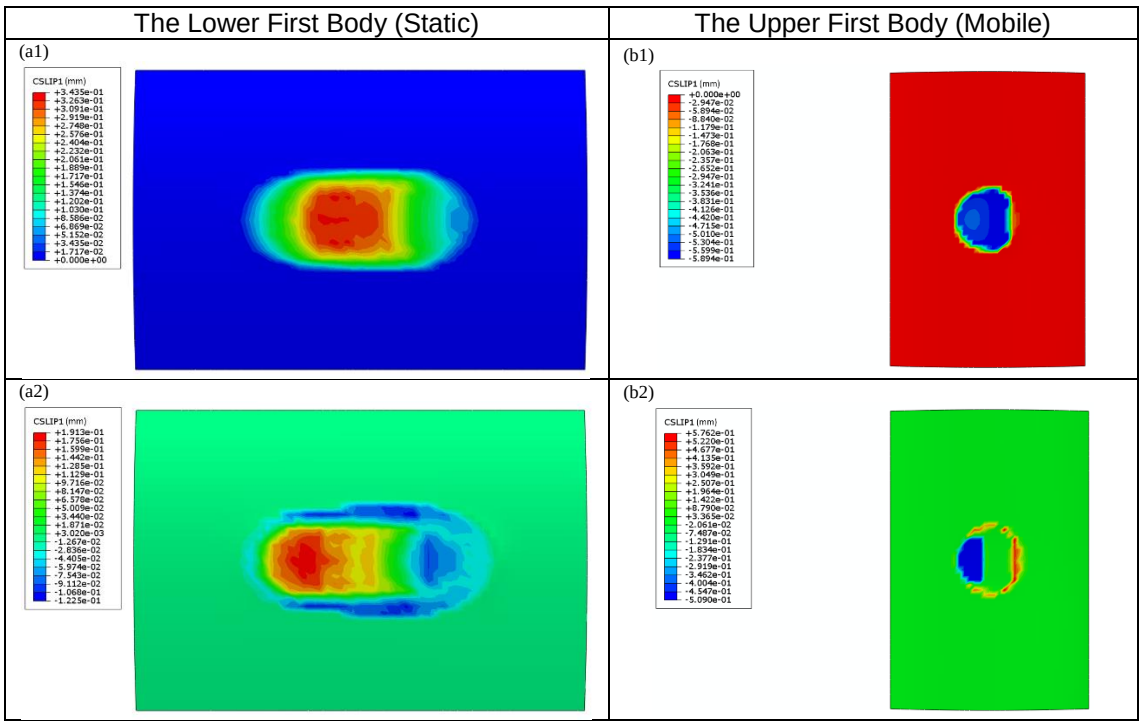


Figure 5.10. The evolution of relative displacement between two first bodies, on both first bodies during the 1st fretting cycle. The upper images (a1 and b1) are from the middle of the cyclic motion, and the lower (a2 and b2) at the end of the motion.

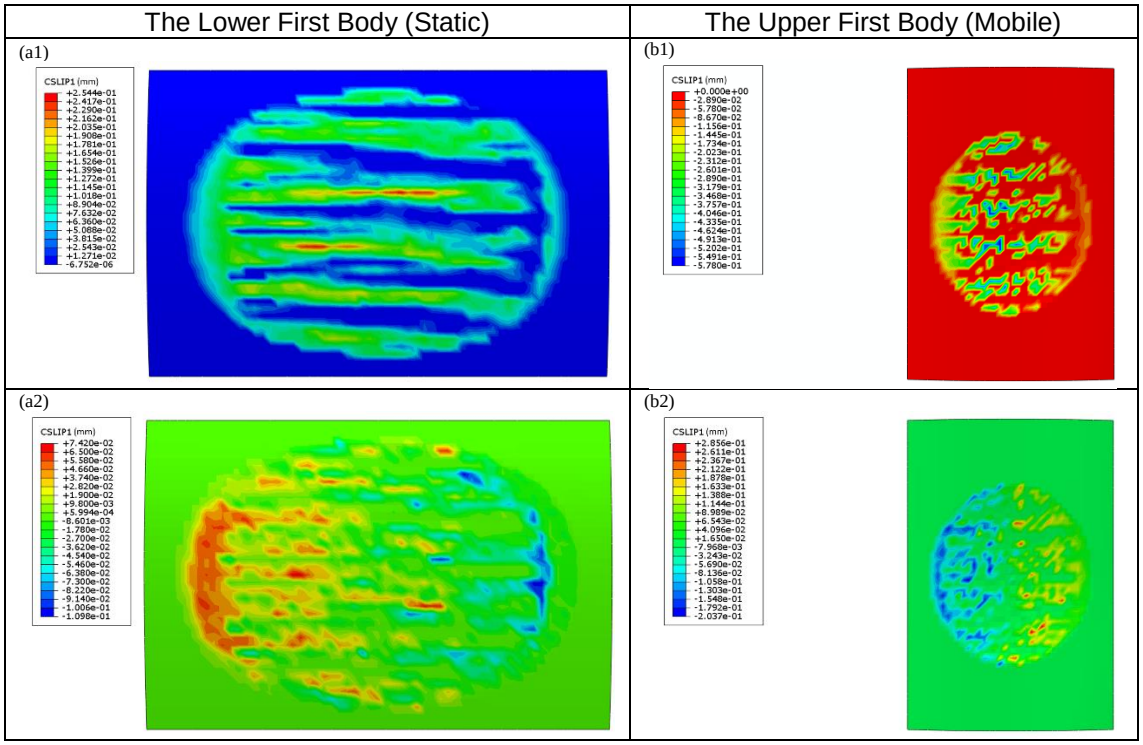


Figure 5.11. The evolution of relative displacement between two first bodies, on both first bodies during the 60th fretting cycle. The upper images (a1 and b1) are from the middle of the cyclic motion, and the lower (a2 and b2) at the end of the motion.

5.3.2 Evolution of Surface Profile

The surface profiles predicted by the model are presented in this section. The darker colour stands for larger depth. Figure 5.12(a1), 5.12(b1), 5.13(a1) and

5.13(b1) illustrate the distribution of wear depth without considering the curvature of both first bodies at the 1st and 60th fretting cycle. In Figure 5.12(a2), 5.12(b2), 5.13(a2) and 5.13(b2), the corresponding surface profiles with the curvature of both first bodies are displayed. In Figure 5.14, the evolution of the distribution of wear depth from the 1st to the 120th fretting cycle is demonstrated. The left column shows the evolution results on the lower first body and the right column gives the corresponding results on the upper first body. Figure 5.15 demonstrates the evolution of the wear scar profile at the central cross-section from the 1st to the 120th fretting cycle.

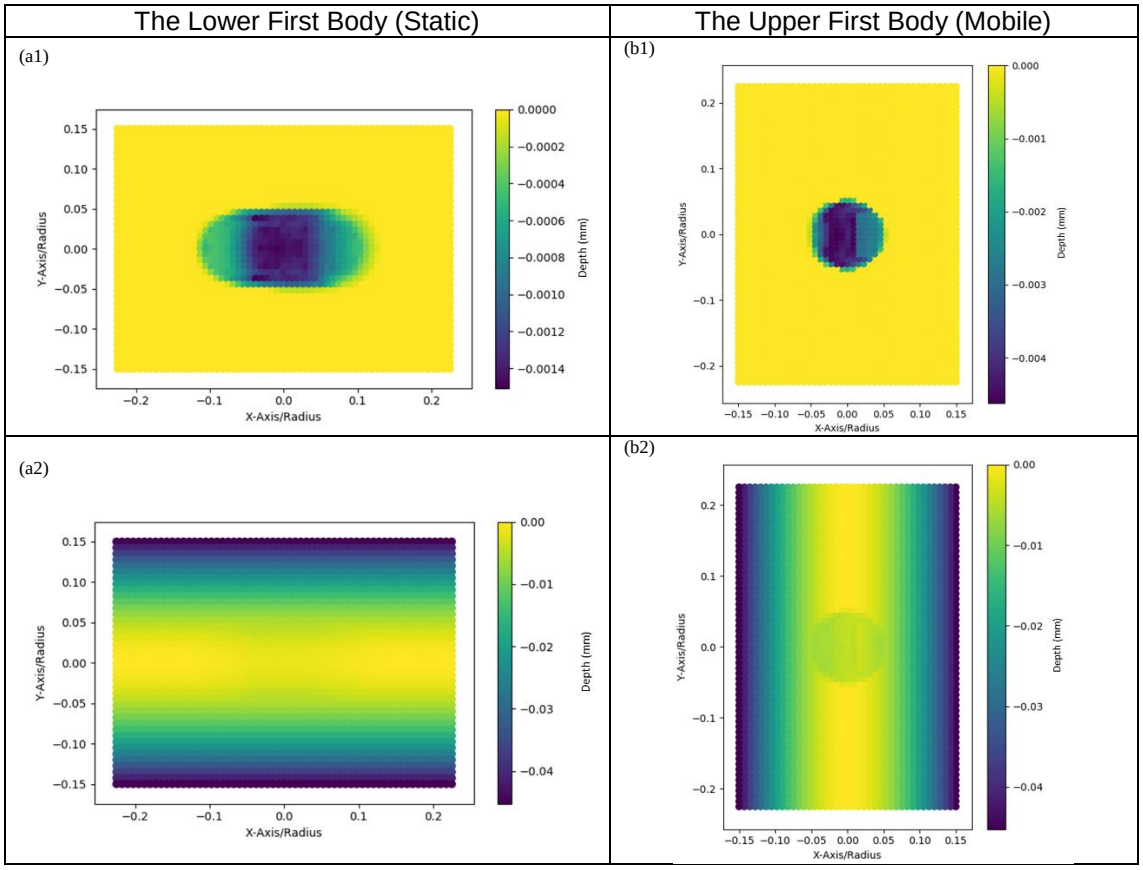


Figure 5.12. The wear scar predicted at the end of the 1st fretting cycle. The upper images (a1 and b1) present the distribution of wear without considering the curvature of the first body. The lower (a2 and b2) present the surface profile of the lower and upper first body with the curvature of the first bodies.

The Lower First Body (Static)	The Upper First Body (Mobile)
(a1)	(b1)

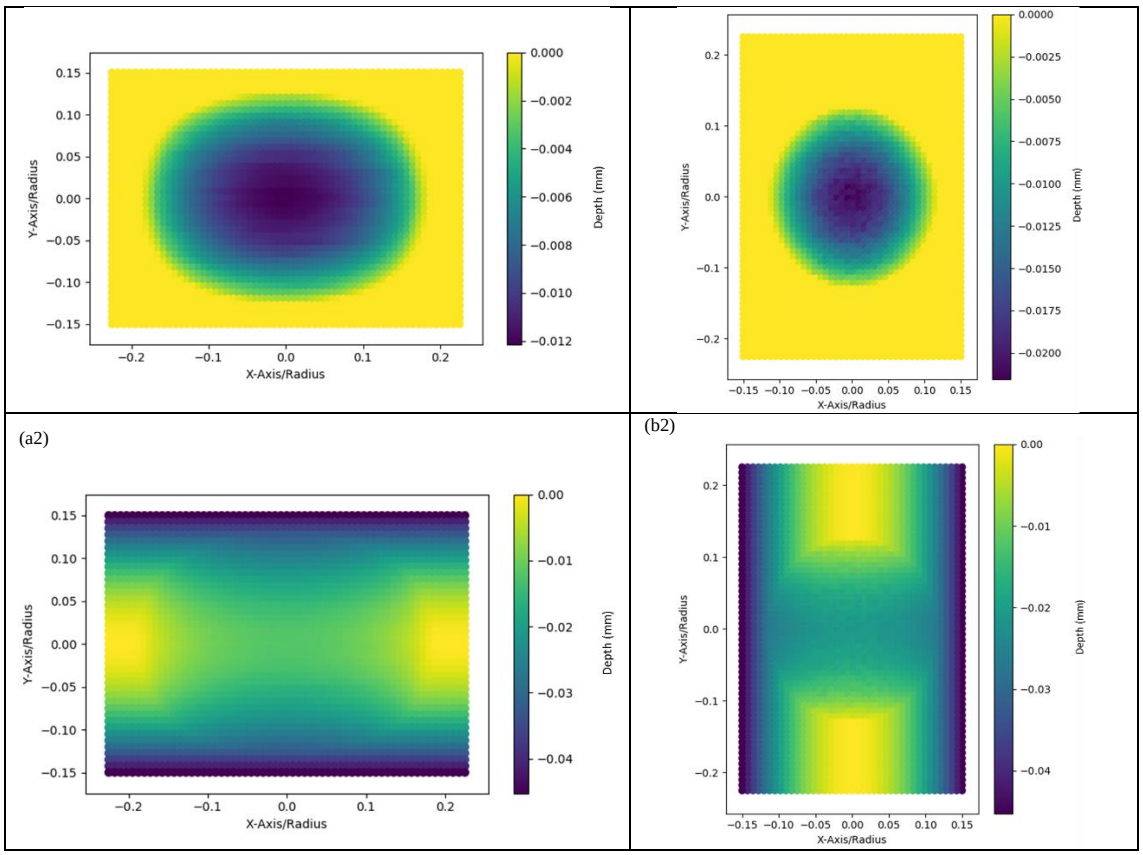
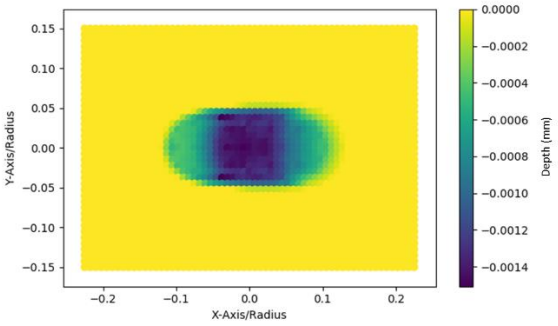
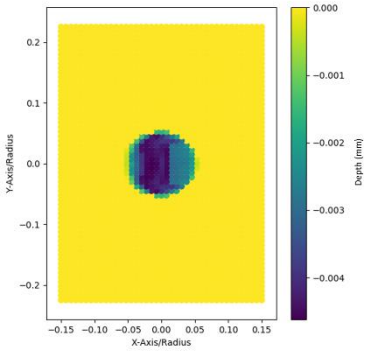
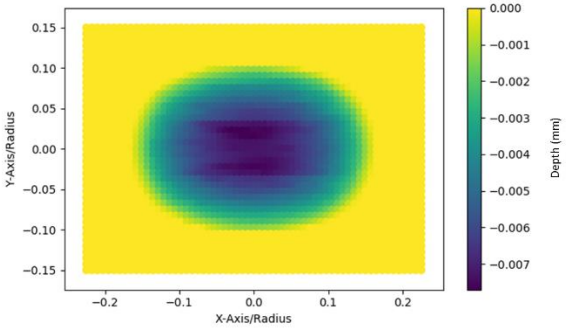
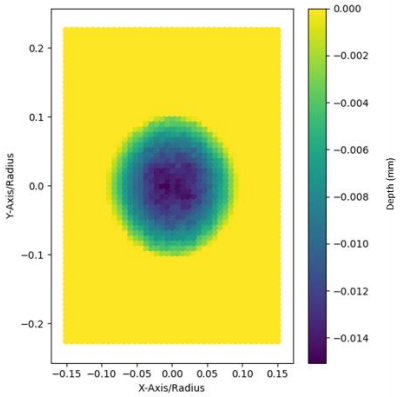
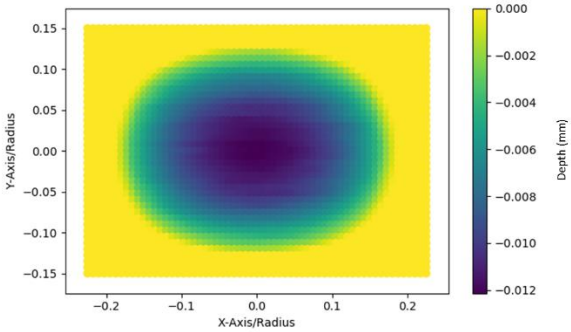
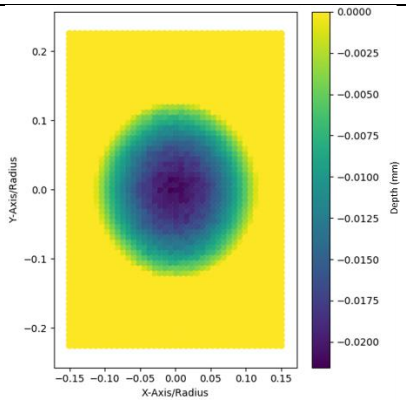
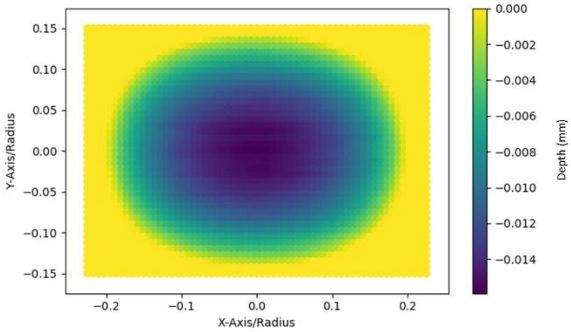
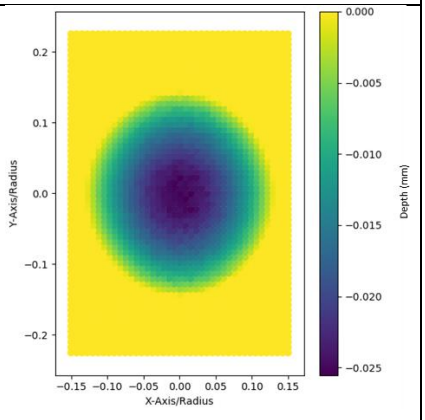


Figure 5.13. The wear scar predicted at the end of 60th fretting cycle. The upper images (a1 and b1) present the distribution of wear without considering the curvature of the first body. The lower (a2 and b2) present the surface profile of the lower and upper first body with the curvature of the first bodies.

	The Lower First Body (Static)	The Upper First Body (Mobile)
1st		
30th		
60th		
90th		

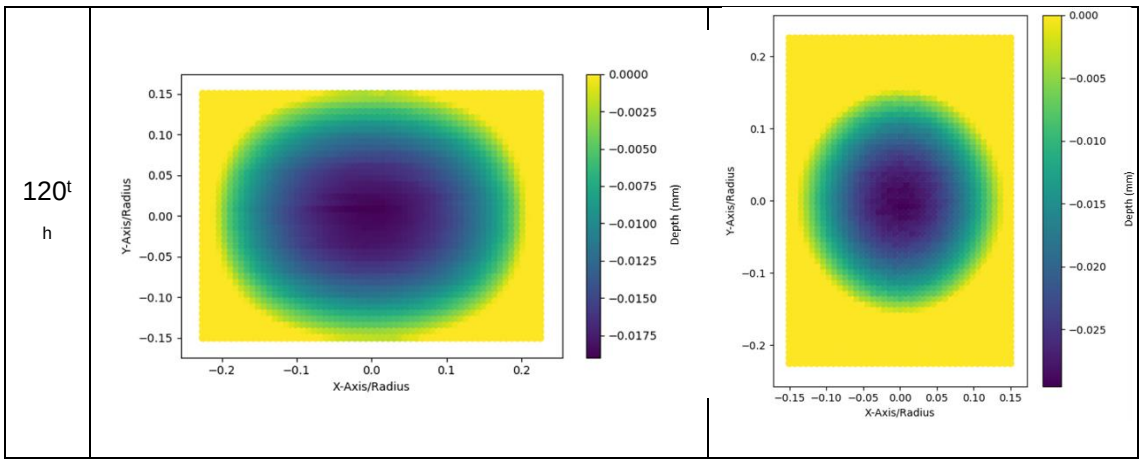
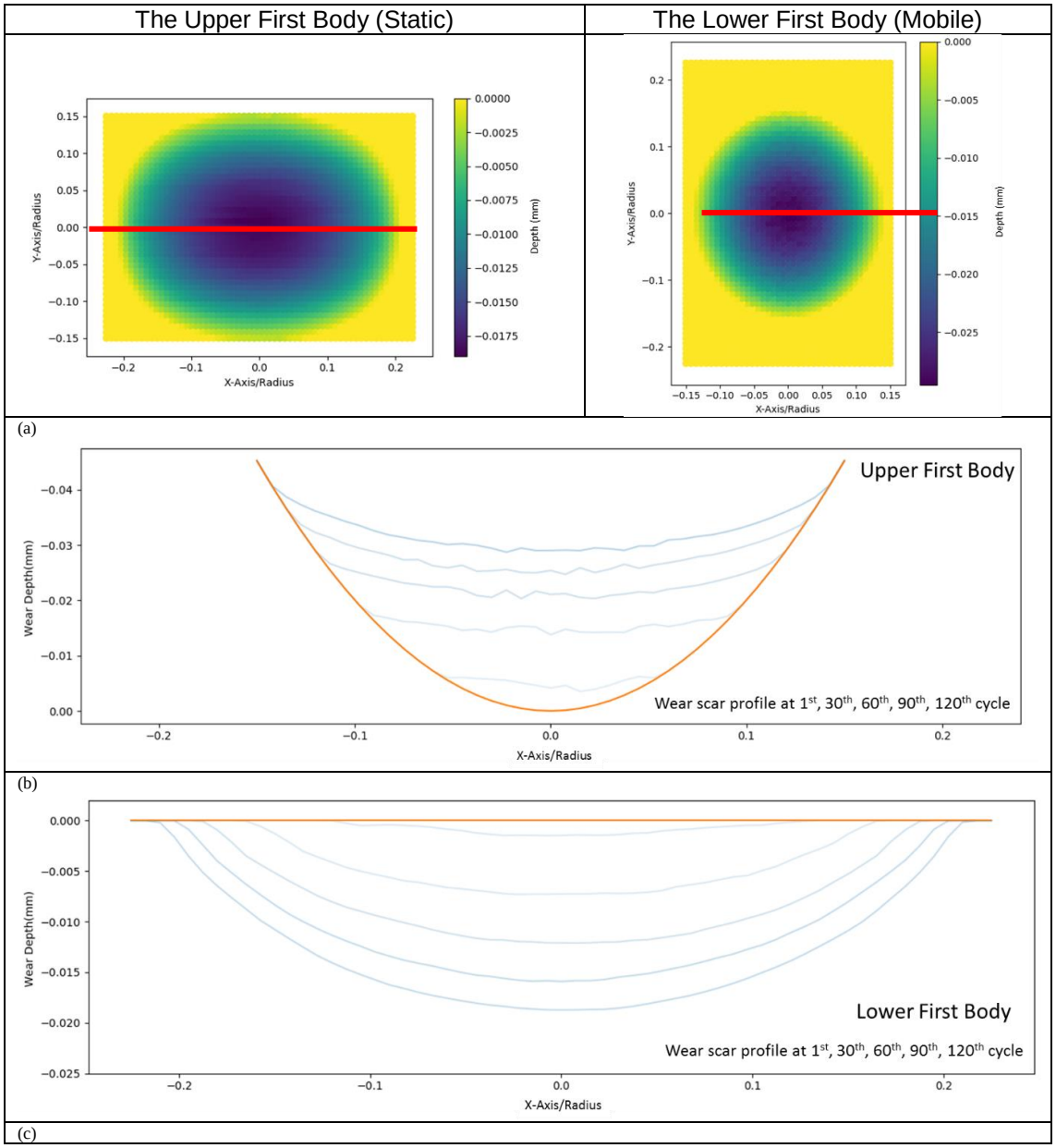


Figure 5.14. The evolution of wear scar on both first bodies without curvature at end of 1st, 20th, 30th, 60th, 90th, 120th fretting wear cycle.



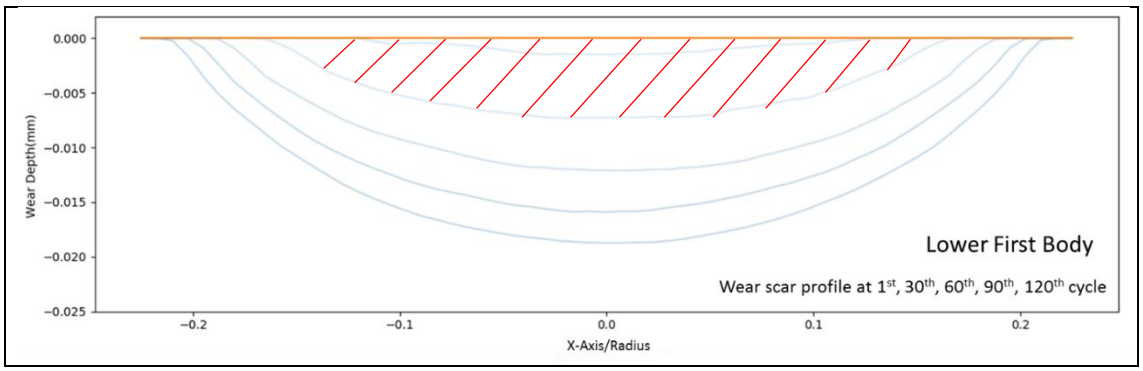


Figure 5.15. The evolution of the wear scar profile at the central cross-section (indicated by the red line) on both first bodies at end of 1st, 20th, 30th, 60th, 90th, 120th fretting wear cycle, i.e. (a) the upper first body; (b) the lower first body. The orange lines represent the initial unworn profile. The blue curves show the profile after wear predicted by the model; (c) an example of calculating wear depth indicated by the red lines. With the progress of fretting wear, the area covered by those red lines will increase.

In Figure 5.16, the evolution of the total averaged wear depth predicted by the models with and without cycle-jumping is presented. It was calculated as the sum of the averaged distances between the wear profile and the initial geometry profile of both first bodies as shown in Figure 5.15. In Figure 5.17, the wear rate per fretting cycle from the model without cycle-jumping is presented, and the marked non-linearity of wear rate in the first few cycles is clearly evident. Figure 5.16 compares the results from the two models with and without cycle-jumping and from the corresponding fretting wear experiment. It is clear the two models underestimate the wear as found in the experiment.

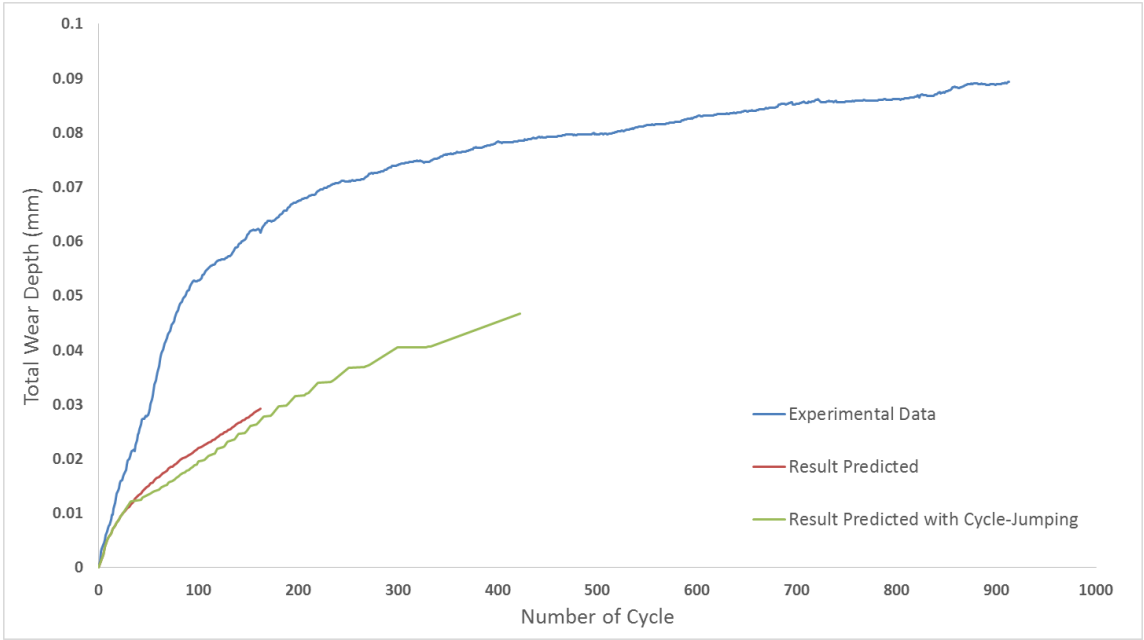


Figure 5.16. A comparison between the results predicted by the two models with and without cycle-jumping and that obtained from fretting wear experiment. An 8% difference is found between the results obtained with and without cycle-jumping.

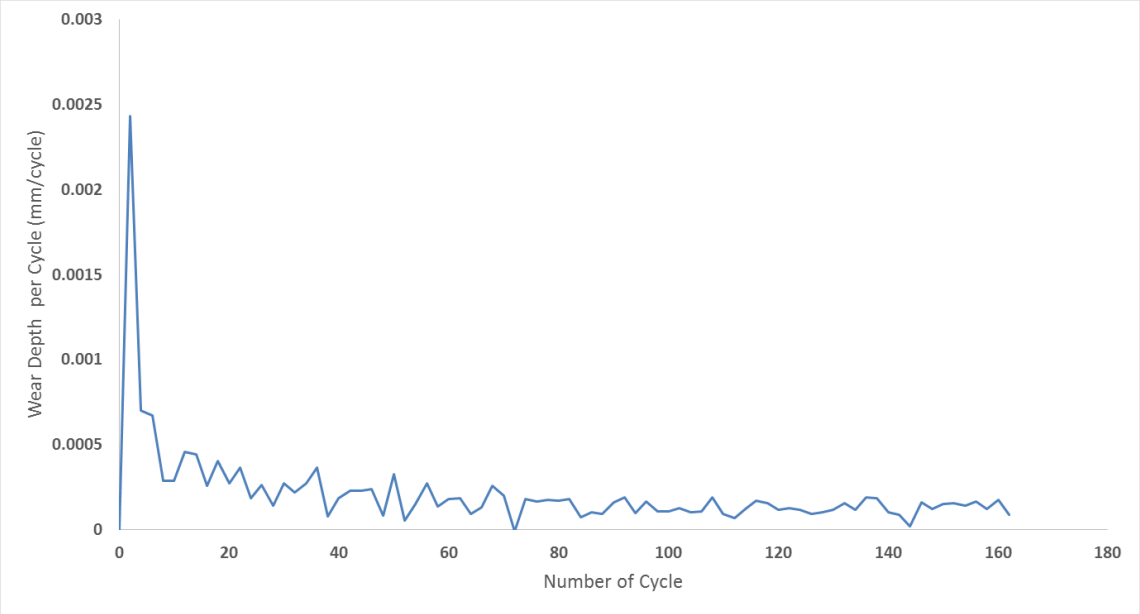


Figure 5.17. Fretting wear depth per cycle produced by the model without cycle-jumping, found as the difference between the total wear depth of two adjacent fretting cycles.

5.4 Effect of the Debris Particles in Hertzian Contact

The worn surface profiles of lower first bodies, as shown in Figure 5.18(a), were calculated at the early and late stage of fretting wear, i.e. 20th and 140th fretting cycle, via the Dissipated Energy method implemented on data from the FE phase. The profiles of the upper first bodies, are presented in Figure 5.18(b), were created as a cylinder with a flat surface which has the same averaged wear depth as the original upper first body at the 20th and 140th fretting cycle. Both first bodies are simulated as rigid bodies inside the DE phase due to the nature of the DE code. Initial debris particles, generated at points of wear, tended to be evenly distributed across space between contact surfaces as presented in Figure 5.19(a). Under the force of gravity and vertical loads applied by the upper first body, the particles were compressed and formed a single layer of debris particles as shown in Figure 5.19(b). At the end of loading step, the debris particles were relocated as being pushed away by any convex parts of the first bodies or trapped inside concave regions of the lower first body. Figure 5.20 and 5.21 present the moving distance of debris particles at the end of the loading step of the two models is

displayed. In Figure 5.22 and 5.24, the effect of debris particles to the distribution of the load applied to the lower first bodies is displayed via comparing the results from both models with and without debris particles. The corresponding results about the force transferred via debris particles are demonstrated in Figure 5.23 and 5.25. Figure 5.26 displays how the distribution of the averaged total force on the lower first body varies with the number of debris particles increases.

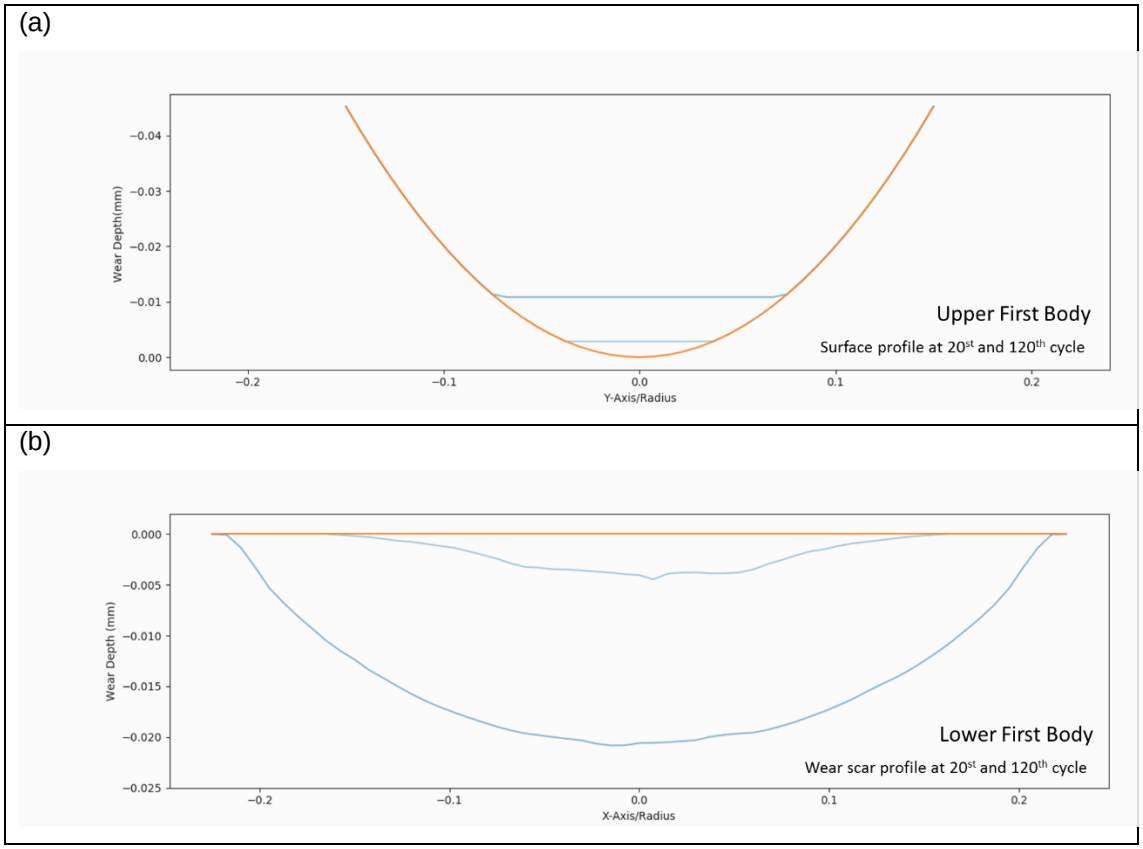


Figure 5.18. The cross-section view of the surface profile of (a) the upper first body and (b) the lower first body shown in blue with the initial surface profile shown in orange.

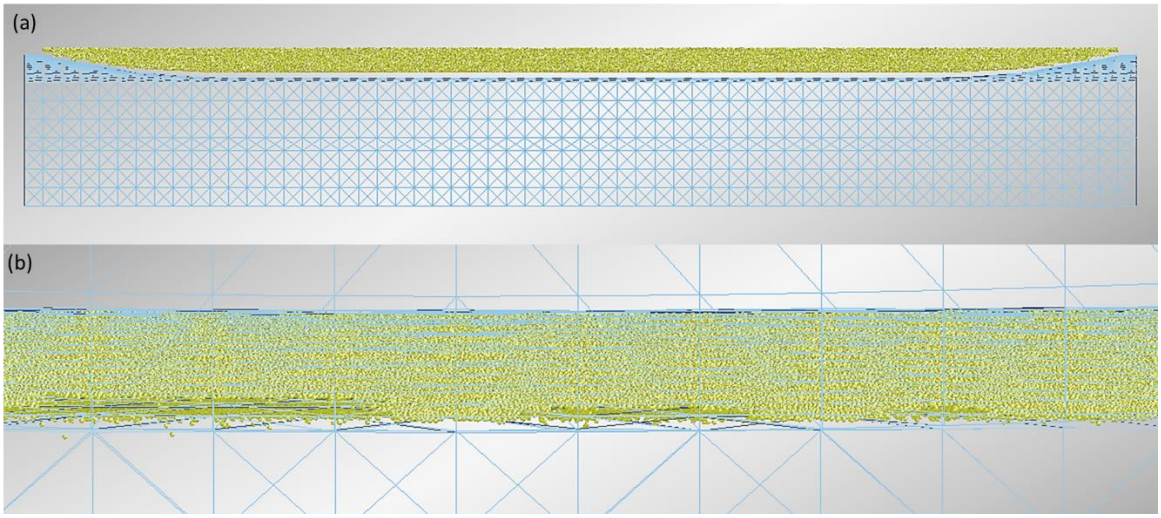


Figure 5.19. Distribution of debris particles in space: (a) particles are evenly distributed across space between contact surfaces before compression; (b) after compression, particles are compressed and form a single layer between contact surfaces.

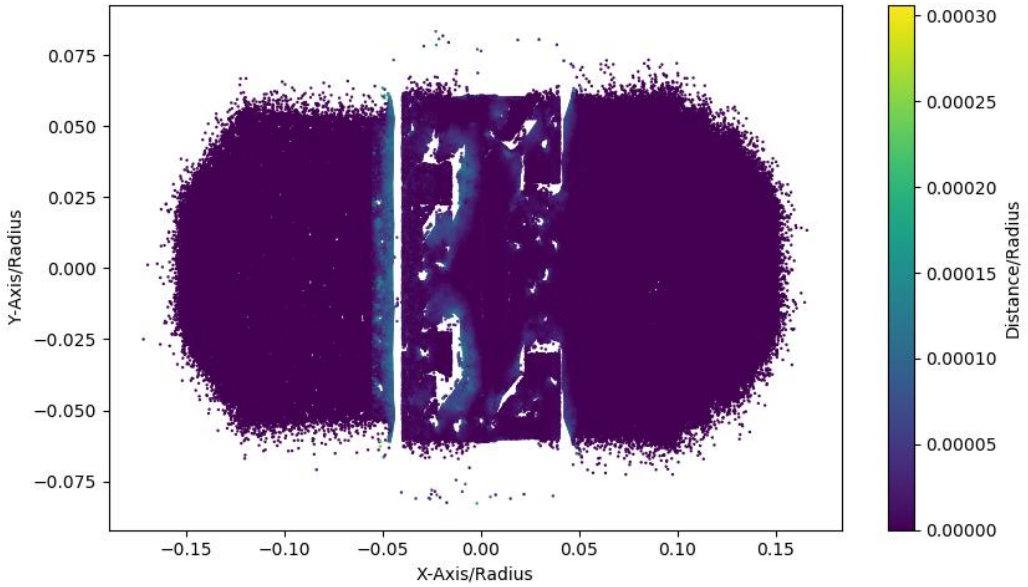


Figure 5.20. The moving distance of debris particles at the end of loading step of the model at the 20th cycle.

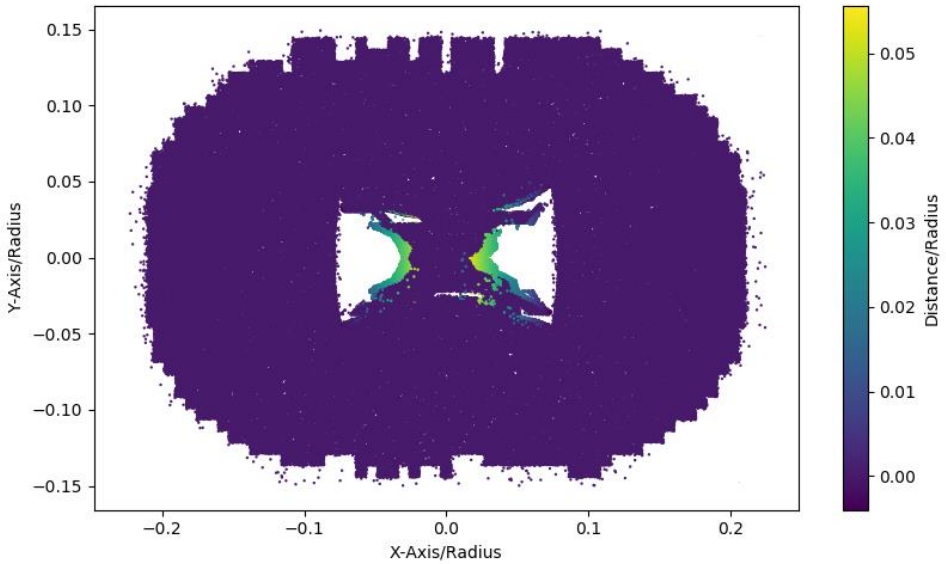


Figure 5.21. The moving distance of debris particles at the end of loading step of the model at the 160th cycle.

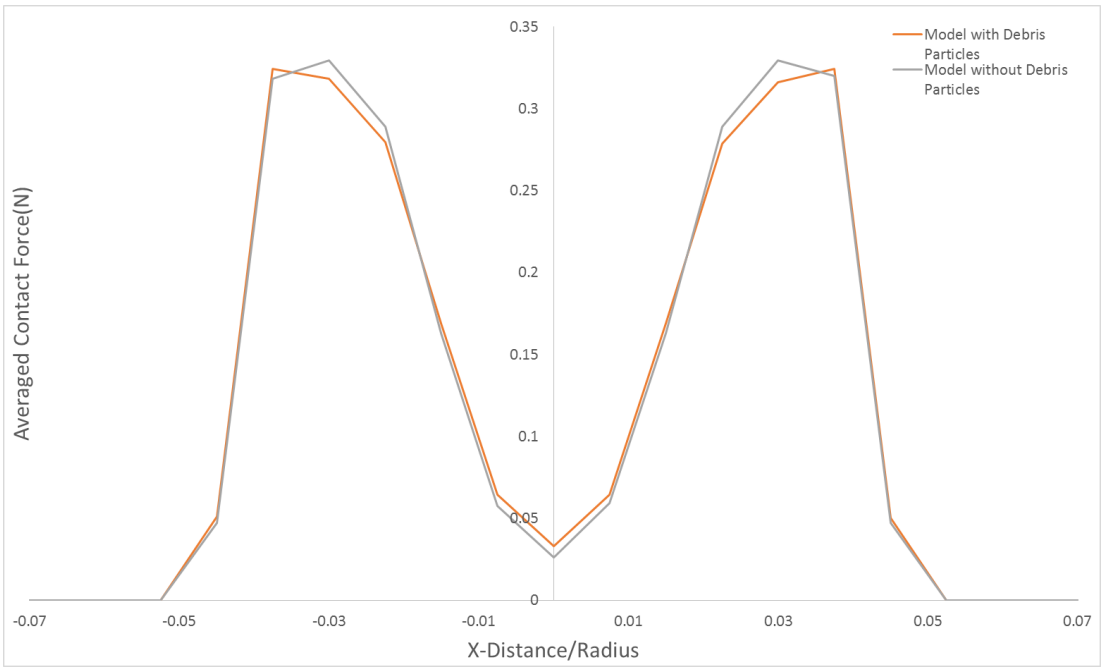


Figure 5.22. Comparison of the distribution of averaged contact force exerted on the lower first body of the models without and with debris particle at the 20th cycle.

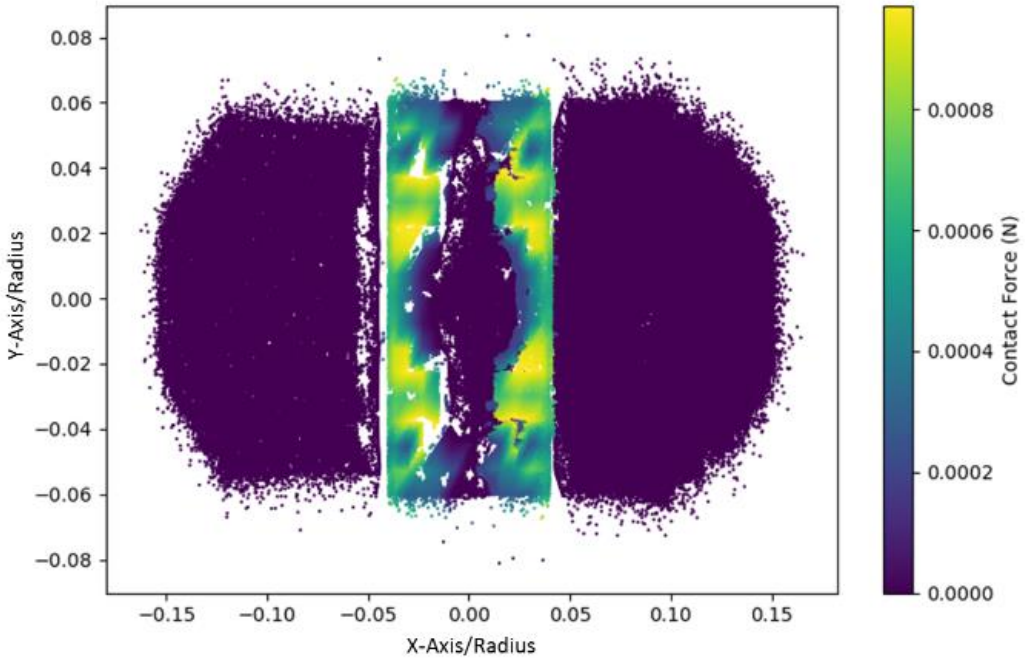


Figure 5.23. The force applied to the debris particles at the 20th cycle.

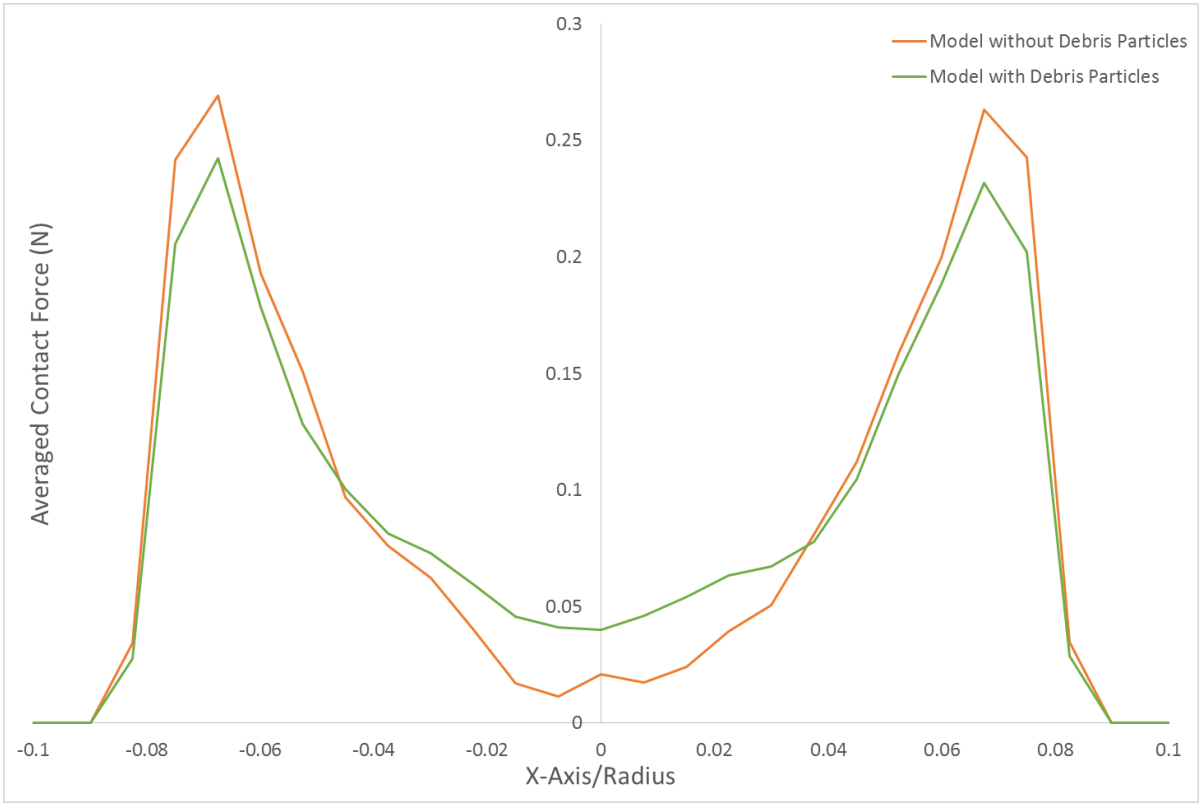


Figure 5.24. Comparison of the distribution of averaged contact force exerted on the lower first body of the models without and with debris particle at the 160th cycle.

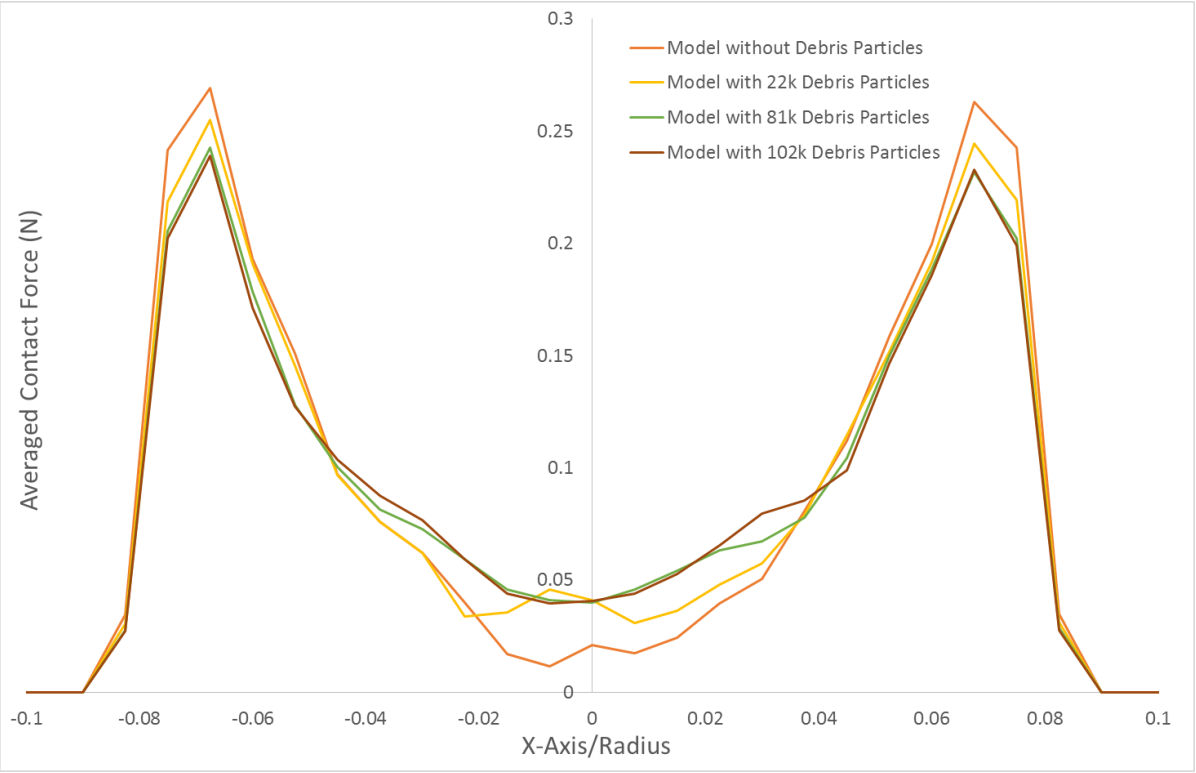


Figure 5.25. Comparison of the distribution of averaged contact force exerted on the lower first body of the models with different number of debris particle at the 160th cycle

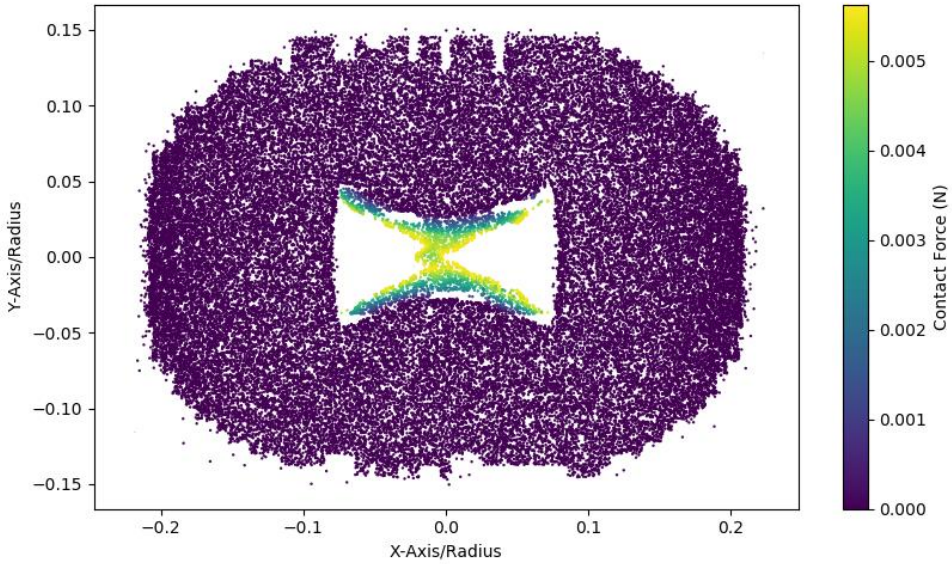


Figure 5.26. The force applied on 22k debris particles at the 160th cycle.

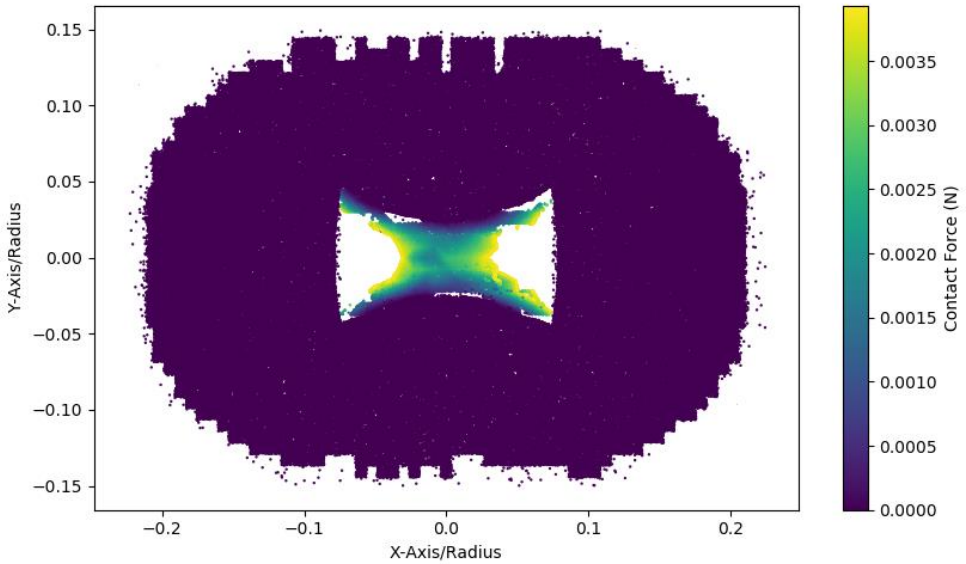


Figure 5.27. The force applied on 81k debris particles at the 160th cycle.

5.5 Discussion of the Fretting Wear Predicted by the Dissipated Energy Method

It was found that the two critical factors of the Dissipated Energy method, i.e. contact traction stress and relative slip displacement, demonstrated considerable nonlinearity in the fretting wear models: both factors decreased with progression of fretting wear. As can be seen in Figure 5.8 and Figure 5.9, the maximum contact traction stress reduced by 21.3% from 963.2 MPa to 758.1 MPa as the contact zone expanded as the material was removed. Since the contact surfaces of both first bodies become rough as wear progresses, indeed as is predicted by the model here, some parts of these surfaces remain in contact and others drop out of contact. Those parts in contact are the only parts with any relative slip displacement, which is required for dissipation of energy. Hence no energy is dissipated in parts out of contact. If as wear progresses the total area of parts in contact increases the contact stress and thus local contacting forces decrease, then both the total slip displacement and the tractive forces decrease resulting in an overall reduction of dissipated energy. This was the case in the models run

herein, with the maximum accumulated relative slip displacements at the nodes reduced significantly in the lower static body (26%) as shown in Figures 5.10(a1) and 5.11(a1), and slightly on the dynamic body (1.9%) see Figures 5.10(b1) and 5.11(b1). As a result, the fretting wear per cycle predicted by the model was nonlinear as wear progressed, as illustrated in Figure 5.16. As shown in Figure 5.15, the change of wear depth on both first body between the 1st and 30th fretting cycle was larger than that between 30th and 60th cycle, which indicates the nonlinear behaviour of fretting wear predicted by the model. A similar behaviour can also be found in the upper first body. Figure 5.17 demonstrates that the rate of wear depth was initially 10 times higher than in the later steady state. This non-linear pattern of wear correctly reflects those observed in experimental data.

As shown by the darkest colours in Figure 5.12(a1) and 5.12(b1), the position of the maximum wear depth predicted by the model, at the end of the 1st fretting cycle, was found to be at the centre of the contact zone for the lower first body but not so for the upper first body. This changed with progressing wear, for instance after 60 fretting cycles [shown in Figure 5.13(a1), 5.13(b1)] the location of maximum wear depth on both first bodies was the centre of the contact zone, as was predicted by the location of maximum dissipated energy in the work of Fouvry et al. [91]. In the work of Fouvry et al. [91], the cumulated dissipated energy was located at the centre of contact as shown in Figure 5.28, which indicates the maximum wear depth (dark colour) also at the centre because of its linear relationship with the energy dissipated. The evolution of surface profiles is explicitly presented in Figure 5.14. The wear scar on the lower first body at the end of the 1st fretting cycle was nearly invisible due to very little wear taking place as presented in Figure 5.12(a2). At the following cycles, a “U” shaped wear scar was observed at the lower first body along the direction of x-axis as displayed in

Figure 5.15(b). On the other hand, the wear scar was found to be relatively “flat” due to the curvature on the upper first body as shown in Figure 5.13(b2) and 5.16(a). Both contact surfaces were predicted to be rough, i.e. concave regions were observed on both first bodies, especially at the centre of the contact zone and early stage of fretting wear. As displayed in Figure 5.14, with the progress of fretting wear, although both surfaces became relatively less jagged because the contact zone expanded which reduced the relative size of the concave regions, the concave regions can be still found at the centre of the contact zone. This pattern of wear scare development and wear scar profile is also seen in experimental data of wear scars [12]. This provides further validation of the model, in that it not only predicts global wear rates but also the detailed patterns of wear across the scars.

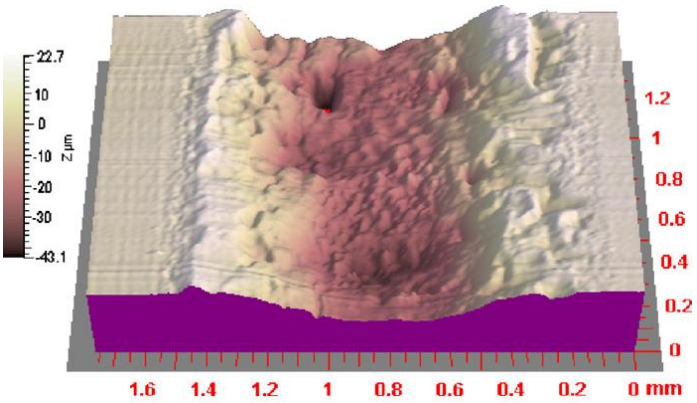


Figure 5.28. 3D surface profile illustrating the ‘‘U-wear shape’’ obtained on a non-adhesive fretting wear [61].

The method derived for cycle-jumping is effective and has minimal cost to accuracy, in this case, 8% vs no cycle-jumping. Further runs of the model would have required a much larger finely meshed section. Due to the high computational cost (i.e. two weeks’ time), only 160 fretting cycles were simulated in the model without using the cycle-jumping technique. As presented in Figure 5.16, a good agreement is found between the two results at the initial stage of fretting wear, i.e. between 0^{th} and 60^{th} cycle, where fretting wear behaved nonlinearly and little cycle-jumping was implemented. After that, fretting wear was considered linear

which allowed the implementation of cycle-jumping and led to a divergence between the two results. Comparison with experimental data in Figure 5.16 shows both of these models are underpredicting significantly. According to the result produced by the model, the fretting wear started by behaving nonlinearly and then became almost linear afterwards. The result predicted by the model is likely to agree with the experimental data at the 1000th cycle but disagree before and after this point. It is caused by the implementation of an energy wear coefficient as a secant tangent at 1000 cycles. Because of the non-linearity of fretting wear, any secant method will produce this error. Therefore, it is found that the Dissipated Energy method is successful in predicting the result at the stage from where the energy wear coefficient was calculated but is inaccurate in describing the evolution of fretting wear. As is this would limit the practical use of this method.

5.6 Discussion of the Effect of the Debris Particles in Hertzian Contact

As shown in Figure 5.22 and 5.24, the distribution of load on the lower first body is changed after introducing debris particles into the contact zone, though the total the same across the entire zone. In the model without debris particles, higher peak forces occur at the two edges of the contact zone while relatively lower forces occur at the centre of the contact zone. After debris particles are introduced into the model, the gross shape of contact force distribution remains unchanged but there are changes evident in the magnitude of peak forces, e.g. the force at the centre of the contact zone in the 160th fretting cycle (shown in Figure 5.24), and the location of peak force in the 20th fretting cycle (shown in Figure 5.22). The Figure 5.29 schematically presents the contact between first bodies without and without the presence of debris particles. In the absence of

debris particles, it might be expected that the peak force would occur at the edge of the contact with the upper flat first body, as shown in 5.29(a) which corresponds to a flat on curved plane contact geometry. With debris particles present, those located near the edge of the contact zone are under higher forces, especially near the edge. It is postulated that lateral forces on particles arising from curved contact surfaces act to push particles away from the contact zone edge, as illustrated in Figure 5.29(b). In Figure 5.23 and 5.25, the movement of debris particles from the two models are presented.

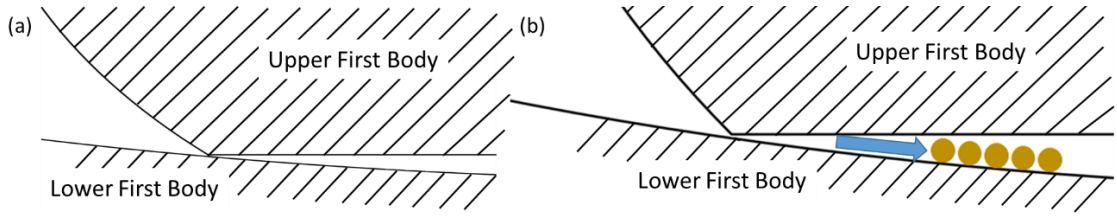


Figure 5.29. Illustrates the space between two first bodies with debris particles between contacting surfaces. The blue arrow indicates the lateral force on the debris particles to push them away from the edge of the contact zone.

In Figure 5.25, the effect of the number of debris particles is presented. Three models were run with differing particle counts, i.e. 102k, 81k, and 22k particles. The results show significant changes in contact force distribution over the contact zone surface. The force at the centre of the contact zone tended to increase and force at the edge of the contact zone to decrease, with increasing particle count. In this case, reduction in the peak contact force due to more particles was of the order of 12% for 81k particles. Since the contact load distribution is changed, according to the principle of the Dissipated Energy method, the pattern of worn material is likely to be different to that without debris particles.

These findings may have particular importance when it comes to choice of wear mechanism. Since the precise nature of the material evolution out of surfaces under fretting wear is less than completely resolved, a user wishing to simulate fretting wear must choose somewhat arbitrarily a failure mechanism. Broadly these mechanisms fall into rates and thresholds, that is some predict

material evolution at a rate proportional to some physical quantity, e.g. dissipated energy, whereas others set a threshold physical parameter beyond which material is deemed to have been removed, e.g. a yield stress. Changes in contact pressures, such as those seen above, can affect predictions using the rate and threshold criteria very markedly. For example in the Hertzian system studied herein, as wear progresses the contact pressure reduces since contact area increases. At some points it is likely this pressure will fall below a critical value for threshold criteria in which case wear ceases, but under a rate criteria continue albeit at a slower rate. At the macro scale this can mean strong divergence between criteria.

5.7 Conclusion

It was shown that the Dissipated Energy method was insufficient to predict the non-linearity of fretting wear observed in experiments with Hertzian contact, see Figure 5.18. This is likely to be caused by the way in which the energy wear coefficient was calculated, itself assuming a linear behaviour. As shown in the experimental data, a non-linear wear behaviour was observed inside which two different wear rates (and possibly mechanisms) were spotted. By applying a dynamic adjustment to the energy wear coefficient it may be possible to make this method predict more accurately. This however requires extensive experimentation prior to modelling, which negates the purpose of developing a predictive model. Though the Dissipated Energy method has been supported by part of the scientific literature, this may be because it was employed only in cases where contact is purely elastic [199]. Hence it may be the case that the Dissipated Energy method can be applied with confidence for predicting fretting wear with

elastic contact, but it is demonstrated that there is at least one case with an elastic-plastic contact where it cannot predict the non-linear wear observed.

With the FDEM model, the effect of debris particles in the contact zone between first bodies was explored. It was found that while the presence of debris particles did not change the gross shape of the contact force distribution, it did influence local details of the contact forces, in particular reducing maxima. The debris particles tended to reduce peak forces (shown in Figure 5.25). As fretting wear progresses, the debris particles tend to even out the distribution of contact pressure.

Chapter 6

Evaluation of Evolution Criteria

6.1 Introduction

There are two established approaches to predict wear, i.e. the Archard equation by J R Archard [10] and the Dissipated Energy method by S Fouvry et al. [61] both of which rely on empirically measured constant coefficients. These methods have been shown to predict the fretting wear under certain conditions [59]. However, there is good evidence showing these methods can fail or have low reliability when operating conditions and material properties change to those in effect when the empirical constant was measured, which severely limits their applicability [200]. This limitation stems from two causes, i) use of an empirical constant required for each and every material pair and set of loading conditions, and ii) because neither approach engages with the mechanism of material failure under fretting wear. The dissipated energy method begins to link the physical quantity of energy and material failure at the surface, but not specifically beyond a statistical correlation.

There is poor evidence in the literature of what mechanics and chemistry occur at the very fine size scale during fretting contact, which causes the material to flow, rupture, and evolve out from a surface to become debris. This is understandable given the challenge of making experimental measurements of rapid, complex and very small scale events taking place typically over microns between contacting surfaces. If the mechanics of evolution were well understood it may be possible to define a ‘correct’ material evolution law able to accurately predict fretting wear. However, in its absence the only route available for studying fretting wear via numerical modelling is to identify criteria which predict wear rates

well and those which don’t, and make the assumption that successful criteria do indeed capture the complex evolutionary process. In this section, the mechanism of material failure during fretting wear is studied via the implementation of six different material evolution criteria into the numerical model. The wear rate obtained using different material evolution criteria is calculated following the Section 3.2.7. The material evolution criteria give markedly different wear rate predictions. A comparison is made between the results predicted by the model and those obtained from a fretting wear experiment.

6.2 Selection of Material Evolution Criteria

One of the difficulties in the study of the mechanism of fretting wear is the great number of influencing factors, load, material properties, temperature, chemistry etc. These factors can be interdependent which further increases the difficulty of identifying those with the most influence on material evolution out from a surface during fretting wear. Therefore, material evolution criteria must be carefully selected.

After reviewing the literature, a number of factors related to the material failure in fretting wear were identified and are summarised in Table 6.1. Those factors were separated into two main categories: external and internal. External factors included parameters such as the contact load, amplitude and frequency, temperature, and gasses available for reaction. Internal factors were those parameters of the contacting bodies such as Young's modulus, hardness, and internal pre-stress. Internal factors could be further divided into two groups: pre-factors and post-factors. The pre-factors are inherent to the materials in contact and not subject to change during the process like Young's modulus. The post-factors were those changing in the process of fretting wear, such as the

distribution and magnitudes of internal stresses and strains. Because fretting wear is a complex process subject to influence from a large range of independent factors, it is unlikely that a single evolution criteria will describe fully and accurately the relationship with fretting wear. In this work only internal factors such as Young's modulus and strengths will be used to predict material evolution and thus wear, leaving the method independent of external factors and empirical fits, attempting to give the best possible general predictability.

Figure 6.1, sets out the factors, separated into groups representing the external factors, internal pre-factors and internal post-factors. The arrows between groups illustrate dependencies between them arising because of the coupled nature of fretting wear. The post factor parameters such as contact stress, shear stress, or dissipated energy, are used to calculate material evolution at points in the model, as indicated in Figure 6.1 with the red arrow. Therefore, in the following sections, material evolution criteria involving internal post-factors: von Mises, equivalent plastic, vertical internal stress, on-plane internal shear stress and critical plane approach were investigated in this work. Subsequently, the links between those internal post-factors and material failure during fretting wear were studied.

Table 6.1. A table of factors related to the material failure of fretting wear.

External Factors	**Internal Factors**	
	Pre-Factors	**Post-Factors**
• Loading force • Amplitude • Frequency • Contact area • Environment, e.g. Temperature • Debris particles	• Young's modulus • Hardness • Yield strength • Ultimate strength • Friction Coefficient	• Contact stress; • In-plane shear stress • Stress-related, i.e. von Mises, Tresca; • Strain-related, i.e. plastic strain; • Dissipated energy;

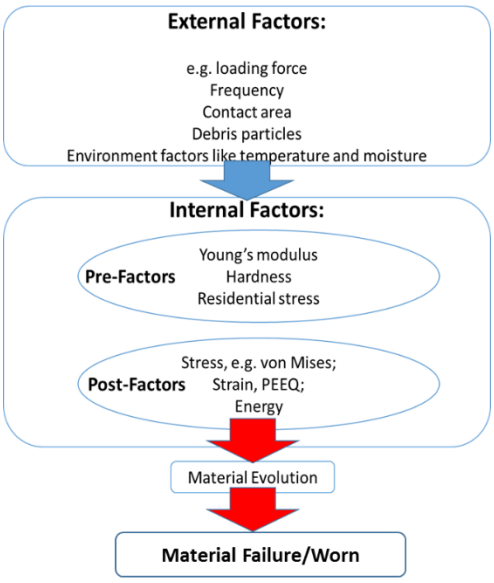


Figure 6.1. Demonstration of dependency between factors of fretting wear.

The model is simulated in the following sections, i.e. of the contact zone between two first bodies, are shown schematically in Figure 6.2. The lower first body remains static during fretting wear while the upper body translates in the longitudinal direction of the lower first body.

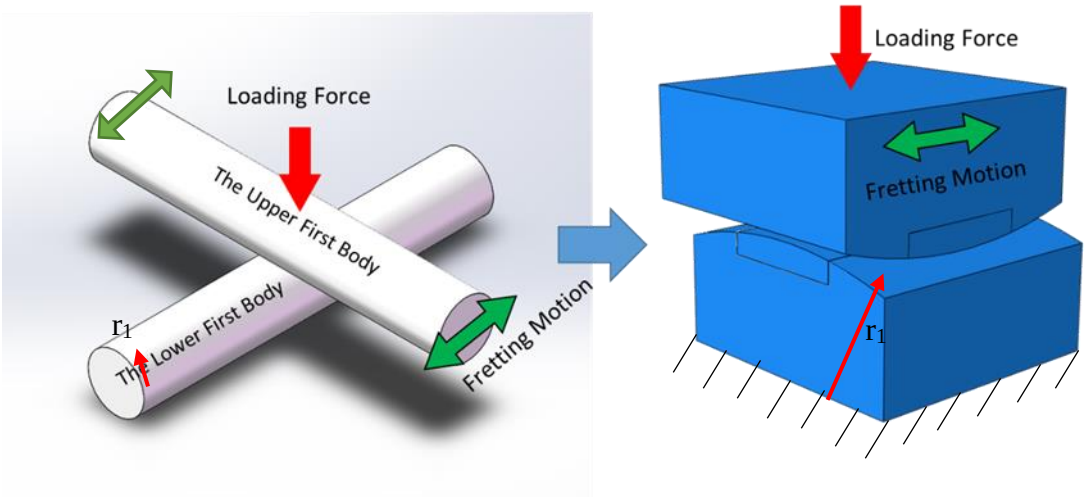


Figure 6.2. Schematically Presentation of The Contact Model with Normal Load of 100N and Stroke Amplitude of 0.6mm.

6.3 Material Evolution Criterion of von Mises

The von Mises stress or equivalent tensile stress criterion has been widely applied to describe the strength of ductile materials very well. The von Mises criterion was implemented within the numerical model, specifically at element

nodes, so that those which exceeded a user-specified yield stress (i.e. 680 MPa in this case for EN24T [197]), were removed and the surface altered accordingly to reflect the volume loss. The material model implemented in the model is described in Chapter Three.

In Figure 6.3 the distribution of maximum nodal von Mises stress on the surface of the lower first body is presented, which was used to calculate fretting wear. The dark blue stands for low value of von Mises stress while yellowish colour represents the higher value. It was found that the maximum von Mises stress in the lower first body was 1600 MPa well above yield at 680 MPa, hence there was a significant plastic deformation of the material in this case. This result suggests that, under a load of 100 N, a reasonable amount of material near to the contact zone in the first body was deforming plastically immediately upon first application of the contact load. The von Mises stress data was then subjected to the material evolution criterion to calculate material to be removed, manifesting as fretting wear. In the implementation of this material evolution criterion, the depth below the surface at which the value of von Mises stress was calculated to reach the yield stress σ_y, ascertained using the following logic argument :

while ($\sigma_{x,y+1,z} > \sigma_{x,y,z}$)
    Search for the node beneath
if ($\sigma_{x,y,z} > \sigma_y$):
    Apply failure mechanism

Where $\sigma_{x,y,z}$ is the von Mises stress of a node with position index of (x, y, z), and $\sigma_{x,y+1,z}$ is the von Mises stress of the node lying beneath. If the von Mises stress matching the yield stress happened to be located between two nodes, an interpolation was then applied to calculate the location as shown in Figure 6.4.

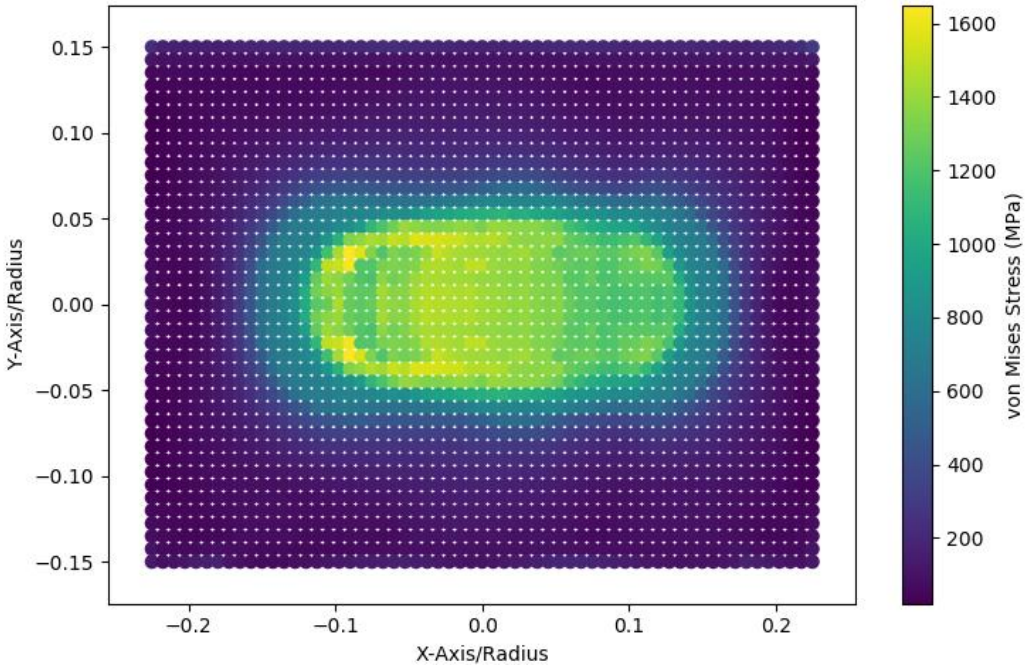


Figure 6.3. Distribution of maximum von Mises stress over a whole cycle in the lower first body. Each dot represents an element node inside the FE model.

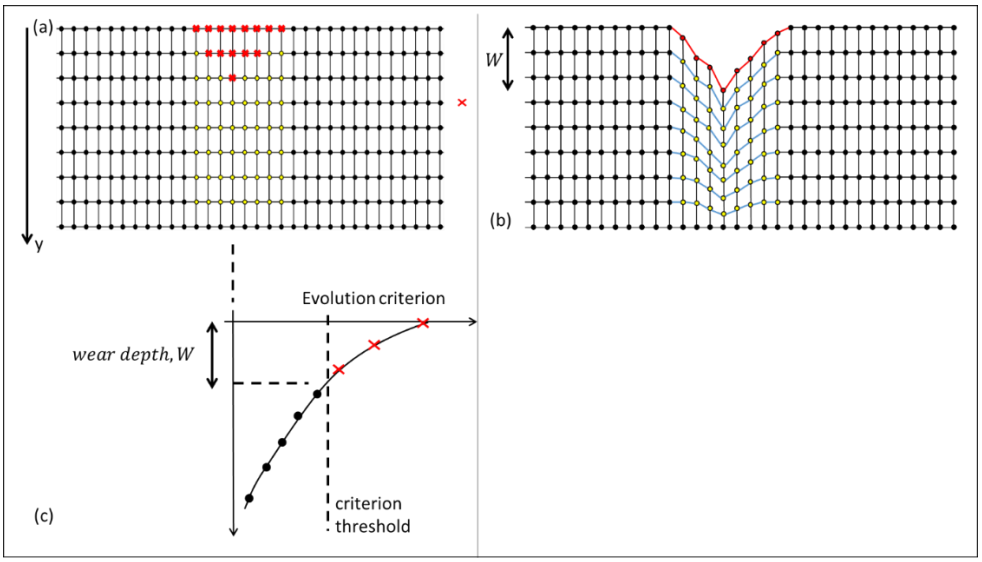


Figure 6.4. Demonstration of implementation of linear interpolation to ascertain the wear depth between nodes.

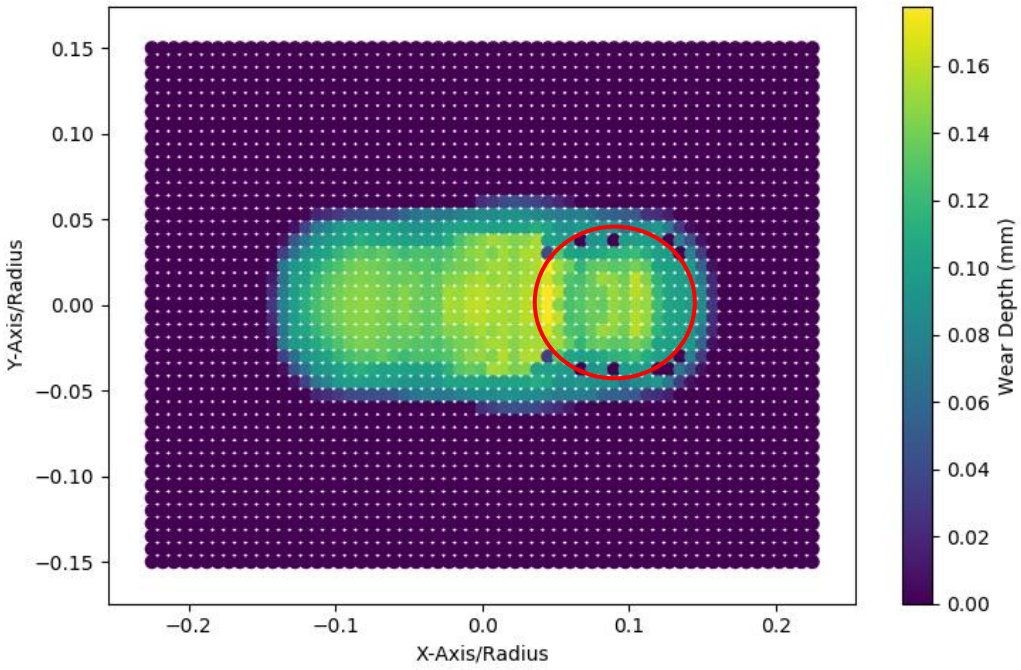


Figure 6.5. Fretting wear predicted by the material evolution criteria of von Mises stress at the lower first body. The red cycle indicates the diameter of the contact zone after load is applied.

Figure 6.5 displays the fretting wear predicted using the von Mises criterion after one fretting cycle. Similarly, the dark blue means there is little or no wear occurred while the yellowish colour stands for material removed at that location. It is clear there are nodes within the contact zone with zero predicted wear. This is caused by large plastic deformation occurring during the initial loading ramp, which removes these point from further contact with the upper first body, hence they see no lateral friction forces within this one cycle, but likely would in subsequent steps as wear progressed. The averaged wear depth was found to be 0.1066 mm over all nodes in the wear scar for this one cycle. In this case the fretting wear model became unstable due to the large distortion of mesh elements. It was found that the von Mises criterion overpredicted the wear rate compared to the experimental result (see Figure 5.15). The implication is that von Mises stress seems not to be adequate for fretting wear prediction.

It was evident from this study that yield of material under contact was significant and extensive in the first bodies prior to any fretting motion. It may be

possible that plasticity is linked to evolution, at least in ductile materials. A plasticity driven criterion cannot work in brittle materials however.

6.4 Material Evolution Criterion of Internal Stress

As indicated by the classical wear model, i.e. the Archard equation, the normal load or the pressure applied to the first bodies is the main factor for the rate of fretting wear, in general terms the higher the contact force the faster wear takes place. Although a positive correlation is observed between contact force and wear rate, it cannot without any doubt be used to explain fretting wear because other factors such as lateral forces due to friction may be drivers of wear, but which themselves cannot be easily be decoupled from the contact load. Thus, the internal stress normal to the contact surface (S22) was implemented as a material evolution criterion. Inside the numerical model, the yield stress, i.e. 680MPa, was used as the threshold beyond which material was removed.

In Figure 6.6, the distribution of maximal S22 stress proximal to both first body surfaces during a single one fretting cycle is demonstrated. The darker colour means a lower or smaller value of the parameter while a brighter colour stands for a higher value. As shown in Figure 6.6, many nodes in both first bodies were found to experience maximal S22 values well beyond the tensile yield stress, peaking at 1400MPa. Similarly, much of the contact zone would be well into yield upon application of the contact load (100 N) and in the first fretting cycle.

As previously, the criterion was applied down from the surface to the nodes at which the value of S22 reached the user-defined yield stress, using linear interpolation as appropriate. In Figure 6.7, the fretting wear predicted using the S22 yield criterion is presented at the end of one fretting cycle, indicating a substantial volume of material was predicted to be lost, i.e. area covered by the

yellowish colour. The averaged wear depth predicted on the lower and upper first body was 0.157 mm and 0.102 mm respectively, which is much bigger than that was observed in fretting wear experiments. Any realistic value of yield will result in similar over predictions. It is unlikely that this criterion adequately predicts the evolution of material during fretting wear. The success of the Archard equation, which uses contact stress as a factor, in predicting sliding wear is probably due to the wear constant which is empirically derived.

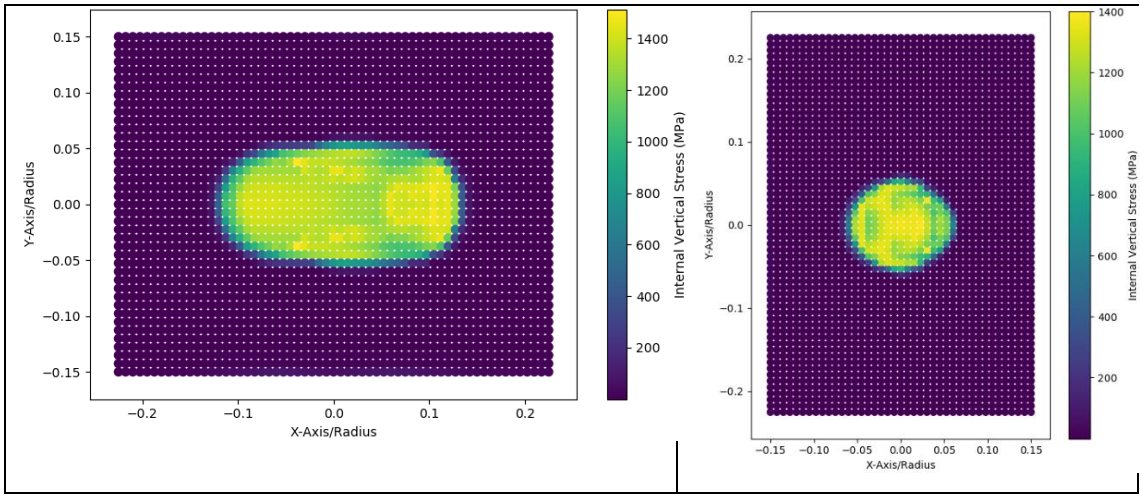


Figure 6.6. Distribution of maximum internal vertical stress inside the lower first body during a fretting cycle.

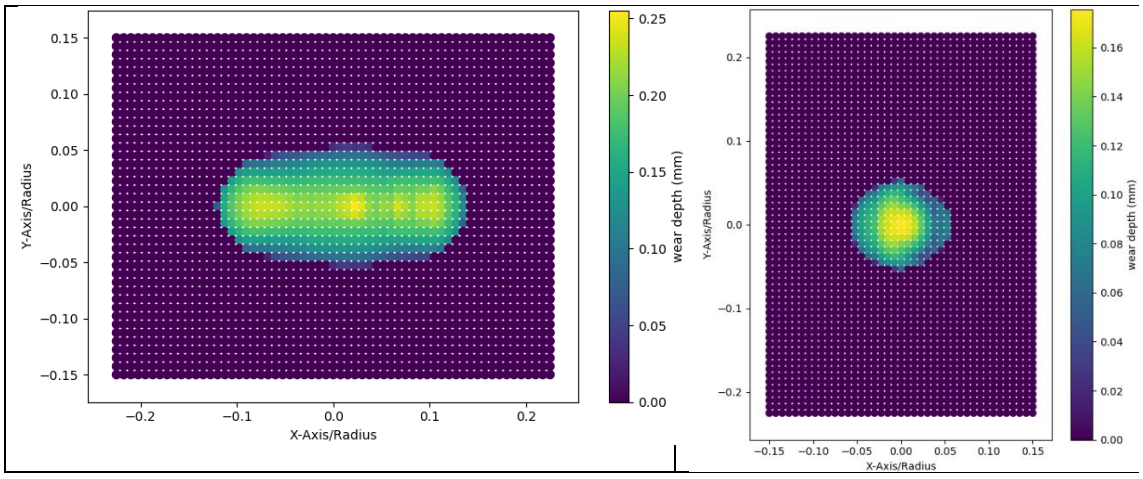


Figure 6.7. Wear depth predicted by the material evolution criterion of internal vertical stress at the lower first body.

6.5 Material Evolution Criterion of Equivalent Plastic Strain

Implementation of both previous criteria showed significant and extensive yield but both over predicted wear rates. Whilst it is clear that simply being in yield itself is not an adequate predictor of material evolution it may be possible that

another feature of plasticity post-yield onset is important, thus the evolution of plasticity of material during fretting wear was studied. A material evolution criterion about the equivalent plastic strain ε_{pc} was employed here. The equivalent plastic strain ε_{pc} is an accumulated scalar measurement of all components of plastic strain, which echoes the direction independent summed threshold concept of von Mises stress. The empirically determined ductile failure damage indicator (DFDI) [103], [104] was applied as a threshold to determine material evolution. The DFDI describes the ratio of equivalent plastic strain and failure strain limit ε_f, which indicates the level of damage at a point or within a volume of material as shown below:

$$DFDI = \int \frac{d\varepsilon_{pc}}{\varepsilon_f} \tag{6.1}$$

where ε_{pc} is the equivalent plastic strain. The threshold value of failure strain limit ε_f is calculated as [103]:

$$\varepsilon_f = 1.65\varepsilon_0 \exp(-\frac{3\sigma_m}{2\sigma_{eq}}) \tag{6.2}$$

where σ_m is the average value of three principle stress, and σ_{eq} is the equivalent von Mises stress. The ε_0 is the critical strain of a material for an incipient crack which is assumed to be equal to the failure strain obtained from a uni-axial tension test [104]. The value of DFDI ranges from 0 to 1 indicating the level of damage from undamaged to fully-damaged. The material is considered to be evolved from the surface, i.e. worn, when the value of DFDI exceeds 1 or the equivalent plastic strain $d\varepsilon_{pc}$ reaches the calculated failure strain limit ε_f.

In Figure 6.8, the distribution of equivalent plastic strain on both first bodies during one fretting cycle is presented. It is clear the maximum equivalent plastic strain was obtained at the centre of the contact zone where the maximum relative motion between two first bodies occurs. In Figure 6.9, the distribution of DFDI on

both first bodies is presented. The maximum values of DFDI on the lower first body were found to be approximately 6, while those on the upper first body was approximately 4. This confirmed significant plastic deformation in the contact zone, predicted by the previous two criteria.

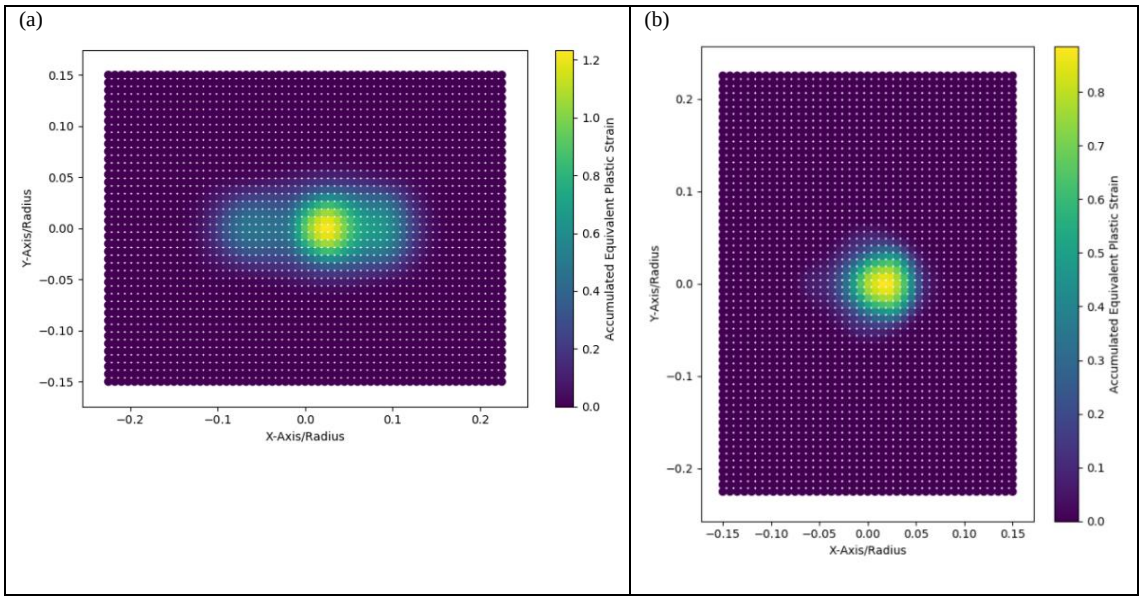


Figure 6.8. (a) Distribution of accumulated equivalent plastic strain on the surface of the lower first body during a fretting cycle; (b) Distribution of accumulated equivalent plastic strain on the surface of the upper first body during a fretting cycle.

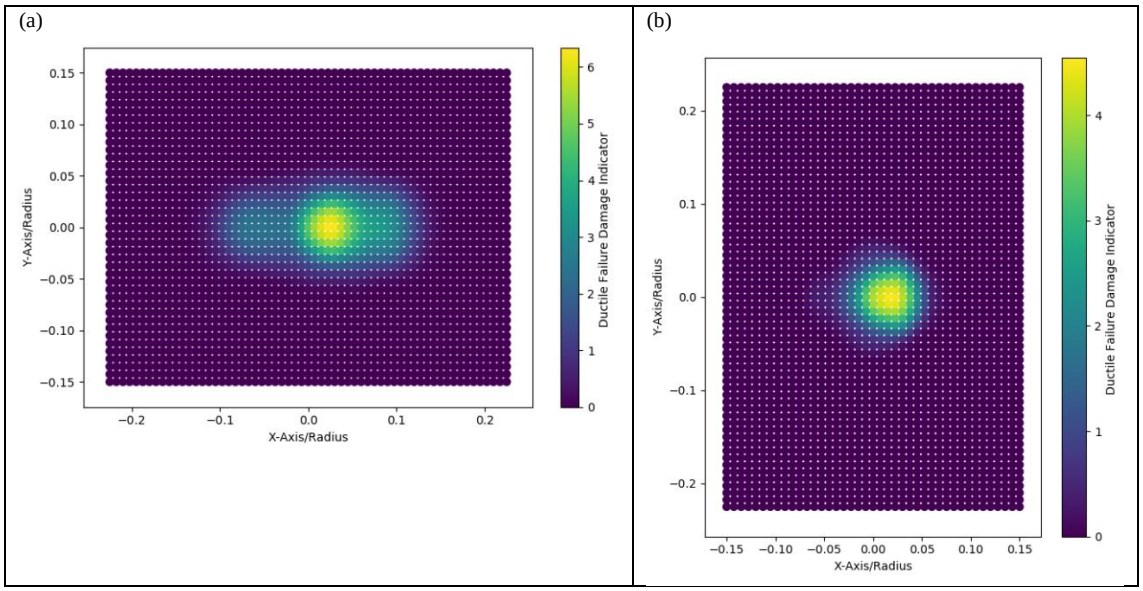


Figure 6.9. (a) Distribution of Ductile Failure Damage Indicator on the surface of the lower first body during a fretting cycle; (b) Distribution of Ductile Failure Damage Indicator on the surface of the upper first body during a fretting cycle.

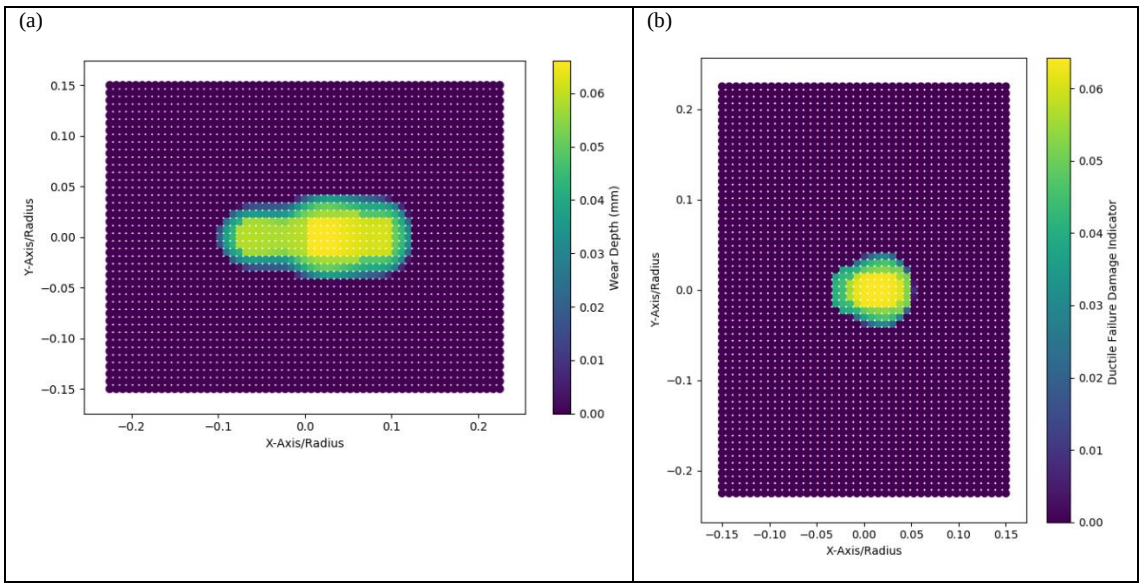


Figure 6.10. (a) Fretting wear predicted with the material evolution criterion of Ductile Failure Damage Indicator on the lower first body during a fretting cycle; (b) Fretting wear predicted with the material evolution criterion of Ductile Failure Damage Indicator on the upper first body during a fretting cycle.

In Figure 6.10, the corresponding wear predicted by the material evolution criteria of equivalent plastic strain is presented. The averaged wear depth of the lower first body was found to be 0.0456 mm while that of the upper first body was 0.0463 mm at the end of first fretting wear cycle. It was found that this material evolution criterion of equivalent plastic strain still over-predicted the fretting wear. But, compared to the results predicted by the previous two material evolution criteria, this method was able to produce a result closer to the real fretting wear obtained from experiments. This confirmed that the material continued to deform plastically until the material failure. The over-prediction of this material evolution criteria can be caused by the method by which the equivalent plastic strain is calculated, i.e. it measures the accumulated plastic strains in all directions and one of which may not actually contribute to the material failure during fretting wear. It suggested that the material failure during fretting wear (i.e. tangential fretting wear in this work) can be caused by the work or a force or a force related parameter applied in one certain direction. It can very likely be the direction of fretting motion. Such idea can be correlated to the Dissipated Energy method inside which the material worn is calculated via estimating the total energy

dissipated along the direction of fretting motion. An alternative reason can be a small value of critical strain which was measured in a tension test. A much higher critical strain can be obtained when the material is failed in shearing which can be observed during tangential fretting wear.

6.6 Material Evolution Criterion of Critical Plane Approach

When evaluating the damage parameter along one certain direction or plane, the Critical Plane Approach was implemented here so as to evaluate the material damage and predict material failure via calculating the corresponding number of cycles to failure. A critical plane is the plane along which the material experiences the most significant damage. The Critical Plane Approach is able to identify the direction of the critical plane and predict the cracking which was observed in the fretting wear experiments by Fridrici et al.[109], [110].

As mentioned in Chapter Two, there is a number of different damage parameters that have been proposed to be correlated to the material failure for different materials and different loading conditions. During fretting contact, the material inside the first bodies can experience a complicated loading history, i.e. both low-cycle fatigue (LCF) and high-cycle fatigue (HCF) conditions can occur. The Smith-Watson-Topper (SWT) fatigue damage parameter, i.e. normal energy, has been successfully employed by many researchers [4], [201]–[203] to study the evolution of fatigue inside the material during fretting wear because it combines high- and low-cycle fatigue equation as follow:

$$SWT = \frac{\sigma_{max}\Delta\varepsilon_a}{2} = \frac{(\sigma'_f)^2}{E}(2N_i)^{2b} + \sigma'_f\varepsilon'_f(2N_i)^{b+c} \quad (6.3)$$

where σ_{max} is the peak normal stress, $\Delta\varepsilon_a$ is the maximum strain range within one fatigue cycle, σ'_f is the fatigue strength coefficient, E is Young's modulus, N_i is the number of cycles to crack initiation, b is the fatigue strength exponent, ε'_f

is the fatigue ductility coefficient and c is the fatigue ductility exponent. In this work, the SWT fatigue parameter was employed here as the material evolution criterion.

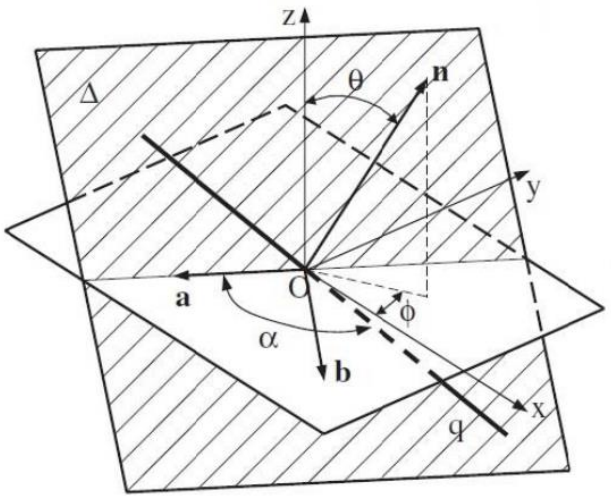


Figure 6.11. Demonstration of a candidate plane Δ with the original coordinate system[126].

During the implementation of the shear stress based Critical Plane Approach, the orientation of critical plane was identified via the Maximum Variance Method (MVM) which transferred the time history of stress for each element onto a series of candidate planes at 5^o intervals over 180^o. Each candidate plane was defined in terms of spherical angles φ and θ related to the reference coordinate system XYZ as shown in Figure 6.11. In Figure 6.11, a reference coordinate system XYZ is shown in white plane while a candidate plane is defined with values of spherical angles φ and θ. Both spherical angles varied from 0^o to 180^o respectively. Inside each candidate plane, the maximum SWT was calculated and compared to that of other candidate planes so as to ascertain the critical plane.

Within each candidate plane, a new coordinate system was built with unit axes vectors of a, b and n. The vectors a, b were perpendicular to each other and lay on the candidate plane with the vector n normal to the candidate plane. There

three unit axes vectors can be expressed in term of the original coordinate system as:

$$n = \begin{bmatrix} n_x \\ n_y \\ n_z \end{bmatrix} = \begin{bmatrix} \sin(\theta)\cos(\varphi) \\ \sin(\theta)\sin(\varphi) \\ \cos(\theta) \end{bmatrix}; \quad a = \begin{bmatrix} a_x \\ a_y \\ a_z \end{bmatrix} = \begin{bmatrix} \sin(\varphi) \\ -\cos(\varphi) \\ 0 \end{bmatrix}; \quad b = \begin{bmatrix} b_x \\ b_y \\ b_z \end{bmatrix} =$$

$$\begin{bmatrix} \cos(\theta)\cos(\varphi) \\ \cos(\theta)\sin(\varphi) \\ -\sin(\theta) \end{bmatrix} \tag{6.4}$$

Those unit axes were then employed to calculate the corresponding normal stress and normal strain inside each candidate plane. When the material on the point O is subjected to a stress tensor $\sigma(t)$ and a strain tensor $\varepsilon(t)$, the instantaneous stress normal to a candidate plane $\sigma_n(t)$ and normal strain ε_n resolved along the direction of vector q can be calculated as:

$$\sigma_n(t) = [n \quad n_y \quad n_z][\sigma(t)]\begin{bmatrix} n_x \\ n_y \\ n_z \end{bmatrix} = [n_x \quad n_y \quad n_z]\begin{bmatrix} \sigma_x(t) & \tau_{xy}(t) & \tau_{xz}(t) \\ \tau_{xy}(t) & \sigma_y(t) & \tau_{yz}(t) \\ \tau_{xz}(t) & \tau_{yz}(t) & \sigma_z(t) \end{bmatrix}\begin{bmatrix} n_x \\ n_y \\ n_z \end{bmatrix} \tag{6.5}$$

$$\varepsilon_n(t) = [n \quad n_y \quad n_z][\varepsilon(t)]\begin{bmatrix} n_x \\ n_y \\ n_z \end{bmatrix} =$$

$$[n \quad n_y \quad n_z]\begin{bmatrix} \varepsilon_x(t) & \gamma_{xy}(t)/2 & \gamma_{xz}(t)/2 \\ \gamma_{xy}(t)/2 & \varepsilon_y(t) & \gamma_{yz}(t)/2 \\ \gamma_{xz}(t)/2 & \gamma_{yz}(t)/2 & \varepsilon_z(t) \end{bmatrix}\begin{bmatrix} n_x \\ n_y \\ n_z \end{bmatrix} \tag{6.6}$$

The maximum normal stress σ_{max} with respect to time and the corresponding strain range $\Delta\varepsilon_a$ are determined for each candidate plane. These results were then employed to establish the critical plane with the maximum SWT damage parameter in each element. After that, the corresponding number of cycles to failure N_f was calculated along that critical plane via Equation 6.3. The value of fatigue strength exponent was estimated based on the Mitchell method, also known as Socie et al method, for steel with hardness below 500HB [131], [132] as shown in Equation 6.7.

$$b = -\frac{1}{6}\log(\frac{2\times\sigma_f}{\sigma_T}) \quad (6.7)$$

where σ_f is the fracture strength, σ_T is the tensile strength. According to the Mitchell method, the fatigue strength coefficient is numerically equal to the true fracture stress, and the fatigue strain coefficient can be assumed as the fracture stain [131], [132]. The fatigue strain exponent is assumed to be -0.6 [131]. Table 6.2 presents the value of constants applied inside the shear stress based Critical Plane Approach. The value of tensile strength (i.e. 1132 MPa) used here as a "true value" is obtained via transferring an "engineering data" (i.e. measured in a lab) as follow:

$$\sigma_{True} = \sigma_{Engineering} \times (1 + \varepsilon_{Enngineering})$$

Table 6.2. . A table of constants employed in the Critical Plane Approach.

Fatigue strength exponent	-0.103
True fracture strength	2349 MPa
Tensile strength	1132 MPa
fatigue ductility coefficient	1.22
fatigue ductility exponent	-0.6

Due to the fact that the material removal effect of wear changes the loading conditions from the present cycle to the subsequent, the result (i.e. number of cycles to failure) obtained from the Equation 6.3 can only provide an instantaneous measure of material damage rate at the present cycle. Thus, a damage accumulation strategy termed "Miner's Rule" [204] was applied here, which has been widely employed in the industry for calculating the accumulated material failure. According to Miner's Rule, after obtaining the number of cycles to failure at the n^{th} cycle, the corresponding damage conducted to the material is defined as $1/N_{f_n}$. Thus, the total damage D accumulated from the beginning to the n^{th}cycle can be calculated as:

$$D = \sum_{i=1}^{n} \frac{1}{N_{f_i}} \quad (6.8)$$

where D is the accumulated damage, N_{f_i} is the number of cycles to failure calculated at i^{th} cycle. When the value of D reaches 1, the material is considered to be failed or worn.

Figure 6.12 illustrates the distribution of damage parameter with the corresponding predicted number of failure on both first bodies at the end of the first fretting cycle. It can be found that the material at the surface of both first bodies is quickly removed. In the Figure 6.13, the worn surface profile predicted by the Critical Plane Approach is presented.

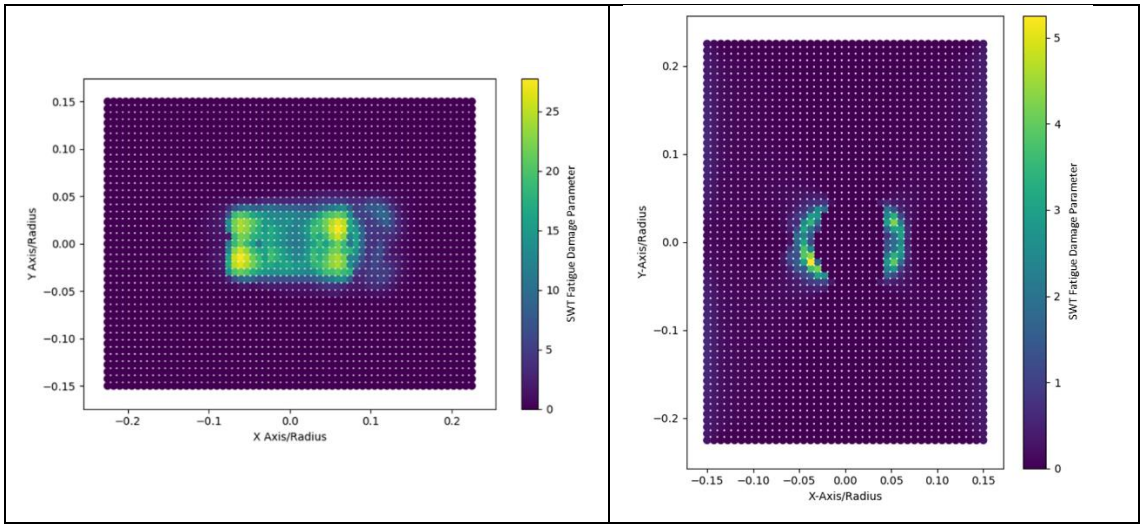


Figure 6.12. Demonstration of the SWT Damage Parameter on both first body.

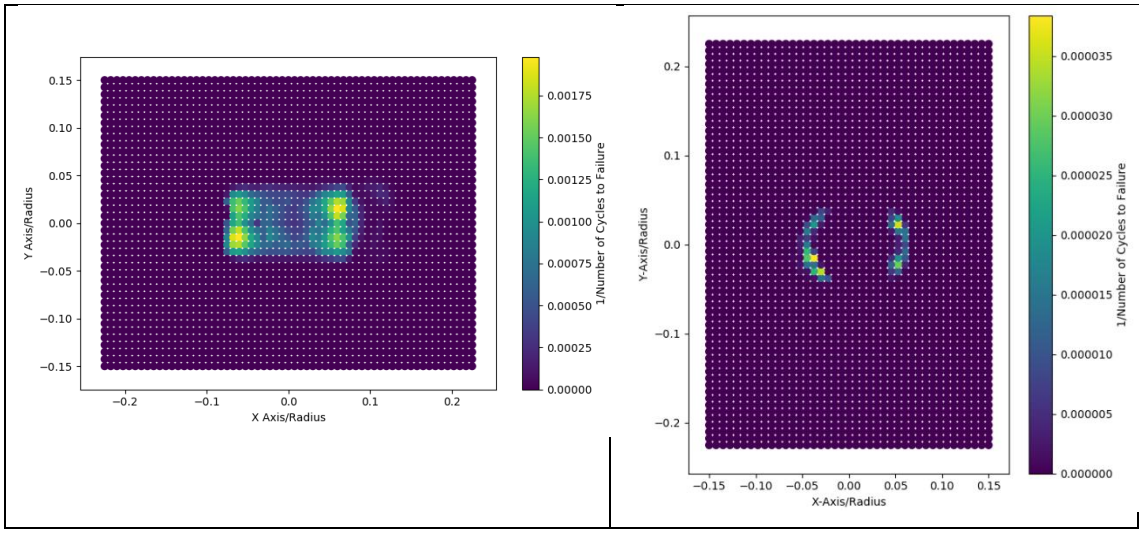


Figure 6.13. Demonstration of the $1/N_{f_n}$ predicted with the material evolution criterion of Critical Plane Approach.

As shown in Figure 6.12, it can found be found that the maximum SWT damage parameter on both first bodies are not focused on one location but occur

at the edges of the contact region, which is caused by the plastic behaviour of the material under load. Such a result is also observed in the work in Nottingham [4]. In the upper first body, the material at the centre of the contact zone obtained very small values of SWT damage parameters, which can suggest the material at that region experienced very little normal work or energy. Figure 6.13 presents the distribution of $1/N_{f_n}$ calculated via the Equation 6.3. It can be found that, according to the results of the model, a large number of fretting cycles will be required before the material is worn: i.e. $\sum_{i=1}^{n} \frac{1}{N_{f_i}}$ becomes 1. Approximately, 600 fretting cycles at least are required by the lower first body and over 200 thousands of cycles are required by the upper first body before the material on the first bodies can be considered worn according to the material evolution criterion. Compared to the result obtained from the fretting wear experiment, this material evolution criterion with Critical Plane Approach of SWT fatigue damage parameter under-predicts the fretting wear. It can be because the SWT damage parameter only calculates the normal energy. It is suitable for predicting the material failure under the tensile cracking mode but not shear cracking mode (i.e. tangential fretting wear) [205]. This result also suggests that the critical role of shear work of the material failure during fretting wear.

6.7 Material Evolution Criterion of In-plane Internal Shear Stress

When calculating the material worn during tangential fretting wear, the two numerical models often used for fretting wear prediction (Archard and Fouvry) take the tangential friction force into consideration directly or indirectly [90]. For the Dissipated Energy method (Fouvry), the tangential friction force is employed directly in calculating the total dissipated energy. For the Archard equation, the

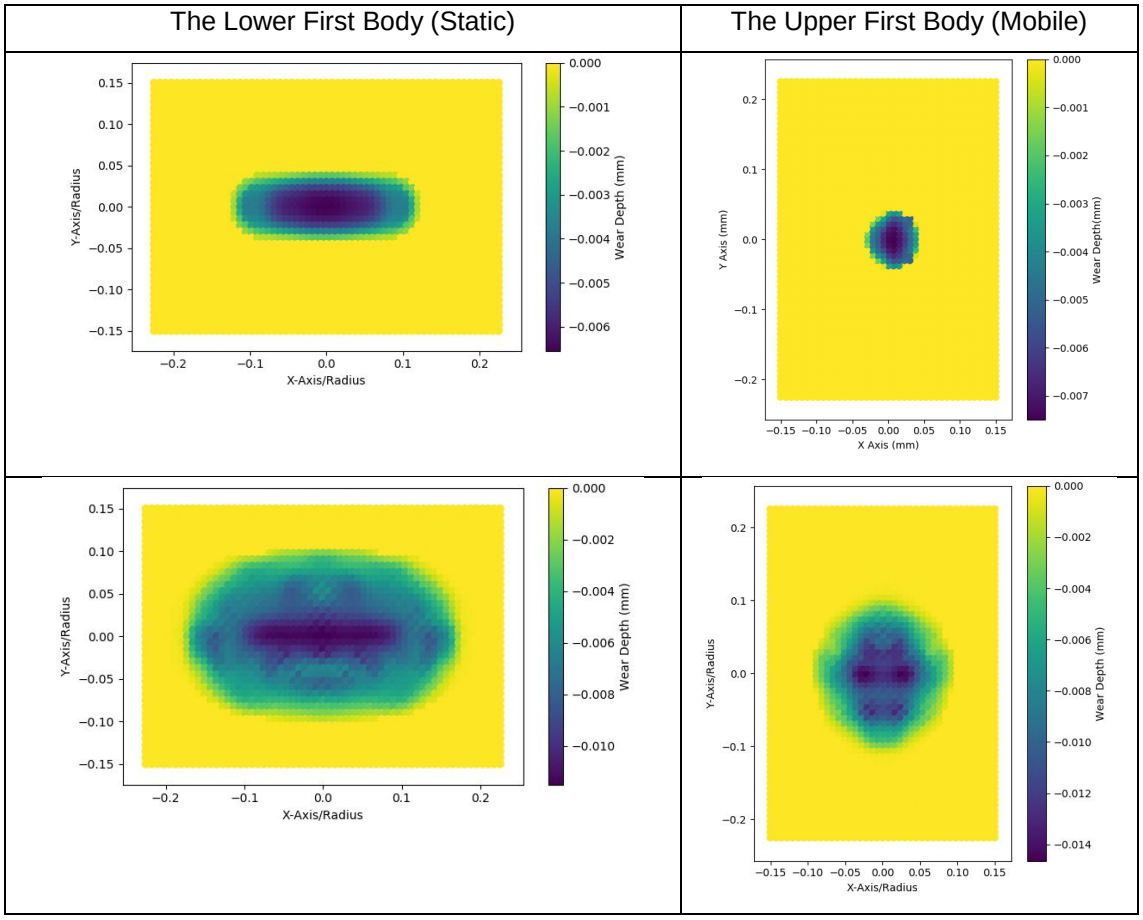
tangential force is taken into the model by multiplying the normal loading force with a friction coefficient which is actually reflected in some way by the wear coefficient [90]. As stated by Fouvry et al., the work done by shear at the interface which is dissipated through within the contact is closely related to the volume of material worn [77]. A major consideration in the work done in this interface is the frictional resistance generated at the contact, which may be described as a friction coefficient.

As mentioned in Chapter Two, several recent papers link shear stress and fretting wear [134], [135], [206], [207], e.g. Liu et al. calculate the wear rate via the shear stress with an empirically measured coefficient. Ghosh [115] calculates a material damage indicator via a function of shear stress τ_{xy} and an ultimate shear stress S_{us}. In this section, a material evolution criterion of shear stress in the plane of the contact zone is developed. Evolution of material out from a surface, i.e. fretting wear, is assume do to be due to the shear stress, and is estimated via a damage indicator D as:

$$D = \frac{\tau_{xy}}{S_{us}} \tag{6.9}$$

where the ultimate shear stress can be estimated based on the ultimate tensile strength of the material: $S_{us} = 0.8 \times S_{ut}$ [208]. The use of a 0.8 multiplier between tensile strength and ultimate shear stress is widely used in estimating the material failure under shearing and is known to work well in engineering design. The damage indicator takes values ranging from 0 (undamaged) to 1(fully damaged). At the point a volume of material reaches D = 1 it is assumed to evolve out from the surface. Within the global script, this material evolution criterion is calculated in each cycle for all elements on or near the contact zone surfaces, and the mesh distorted to reflect the missing volume accordingly.

Figure 6.14 illustrates the predicted fretting wear of both first bodies at the 2nd, 11th, 22th, and 30th fretting cycle for the same problem used in the preceding chapters. Further runs of the model would have required a much larger finely meshed section. Figure 6.15 presents wear depths at various cycles between 1st and 30th, taken along an axis in the middle of the scar parallel to the fretting motion. The wear depth is calculated as the mean depth across the wear scar normal to the fretting motion and relates to the macro scale rate of fretting wear. These patterns of wear are similar in form to those shown by Fouvry et al. [46], as reproduced in figure 6.14, and as would be expected for any similar fretting wear case. Comparison with experimental data from a matching experiment follows.



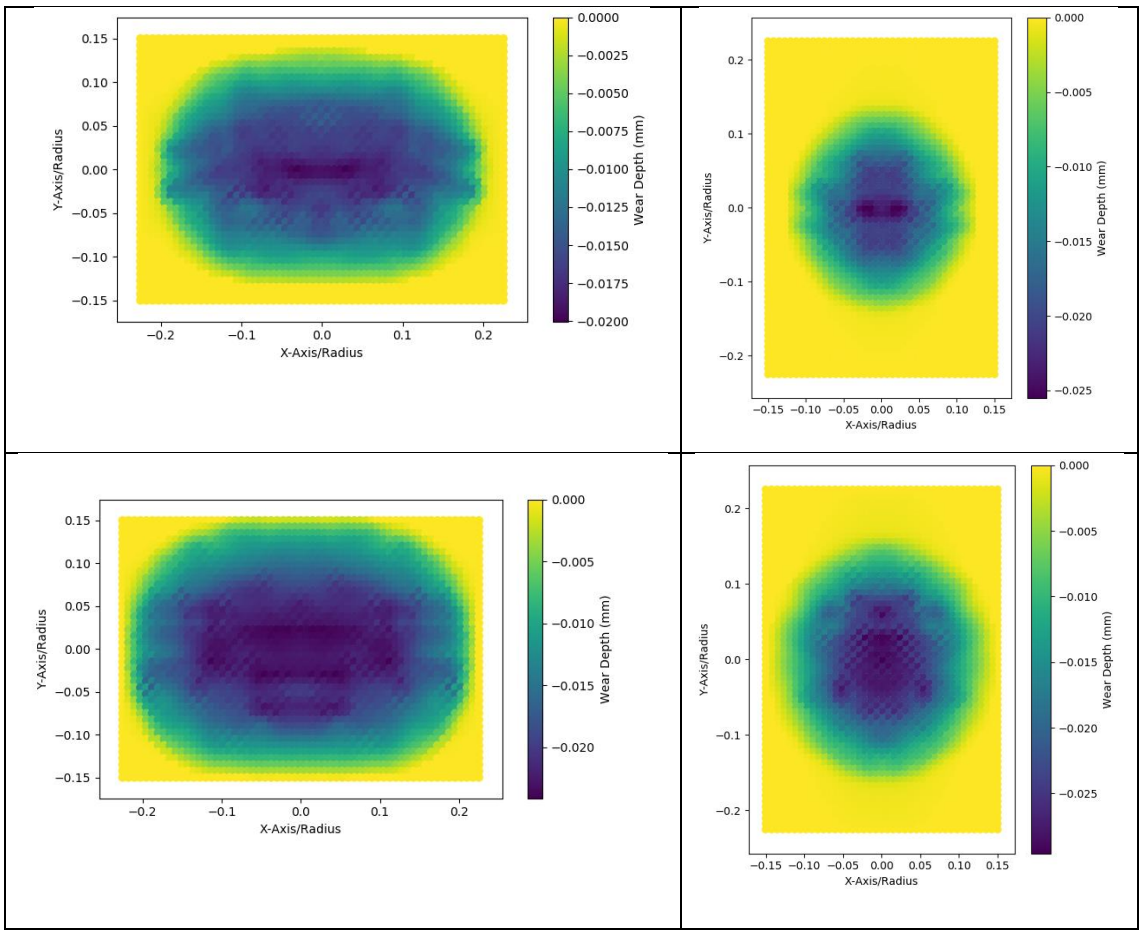


Figure 6.14. The fretting wear predicted the model at the 2nd, 11th, 22th, and the 30th fretting cycle.

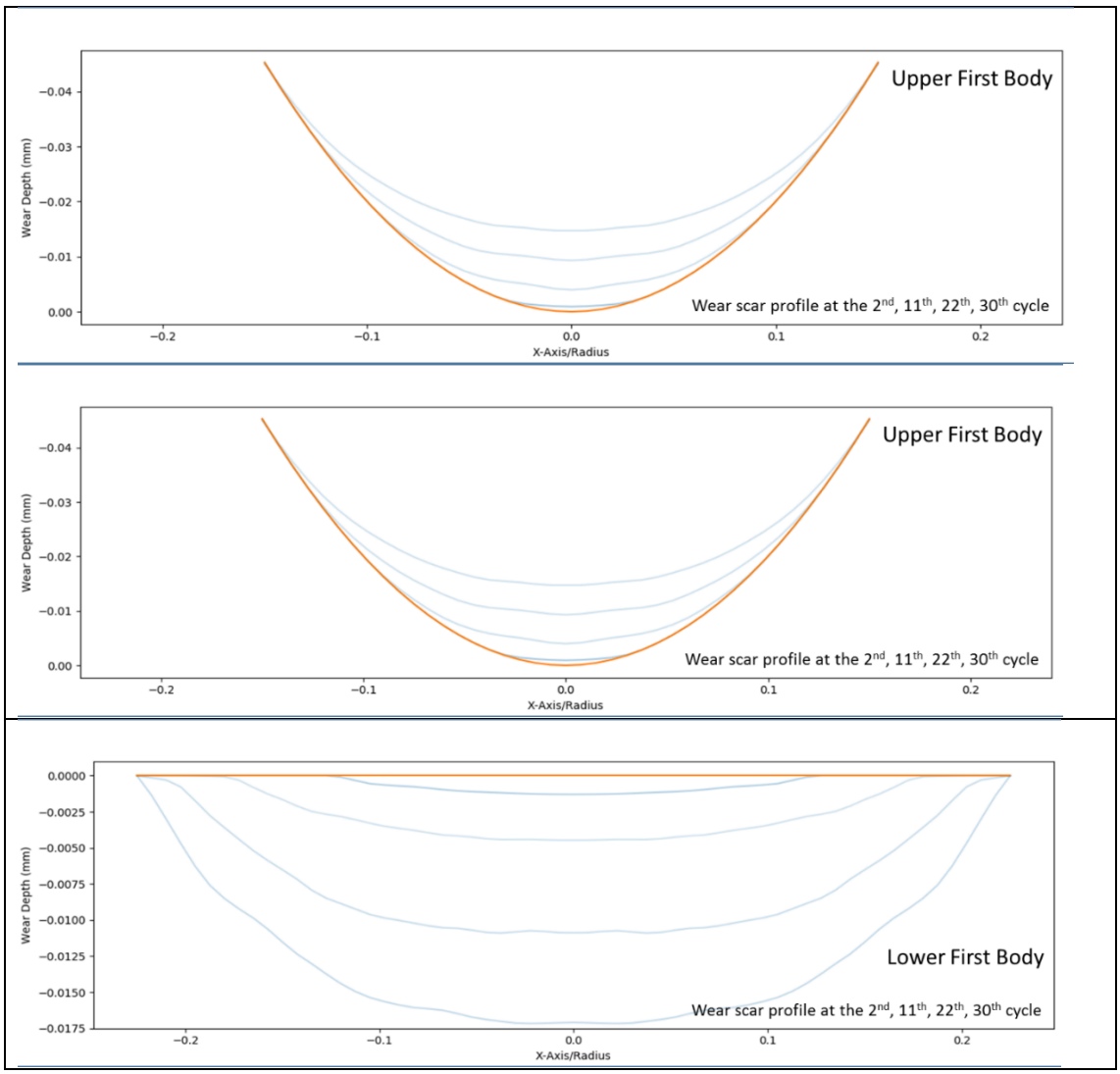


Figure 6.15. The evolution of the wear scar profile on both first bodies at end of 1st, 11th, 22th, 30th fretting wear cycle. The orange lines represent the initial unworn profile. The blue curves show the profile after wear predicted by the model.

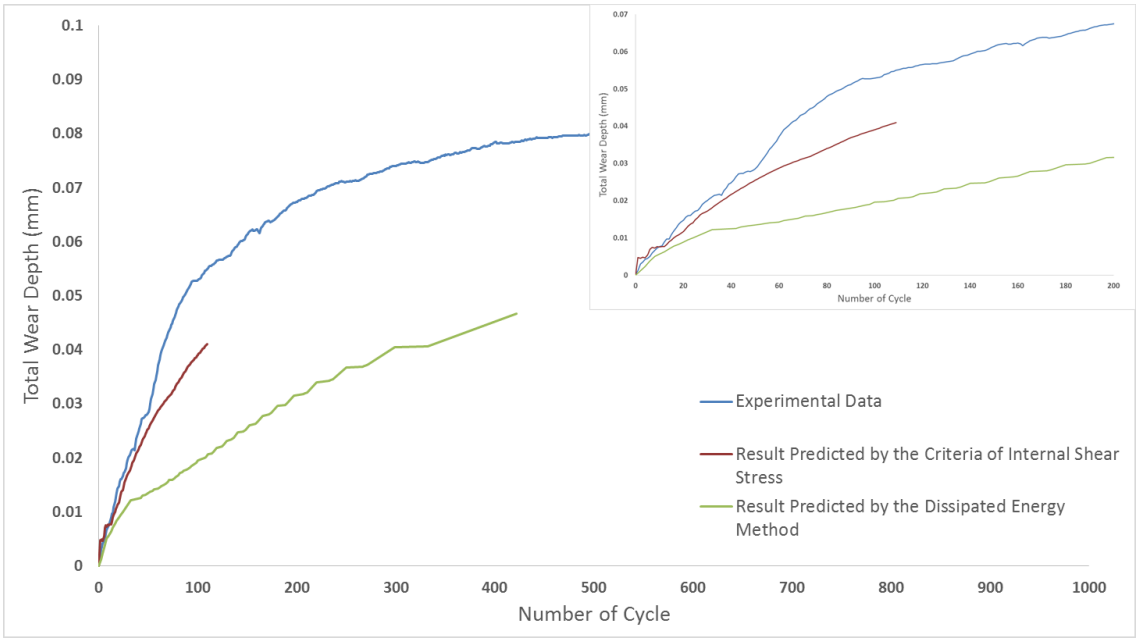


Figure 6.16. A comparison between the results predicted by the two models with and without cycle-jumping and that obtained from the matching fretting wear experiment.

In Figure 6.16, the evolution of fretting wear predicted by the model using the two criteria is compared to experimental test from Blades *et al*. The result predicted by the Dissipated Energy method does not agree with the experimental data well, seeming to be on target to agree with the data only at the start of the test and one further point at approximately 1000 cycles. The shear stress criterion gives a much better fit to the experimental data. The model was used with cycle jumping turned on for both criteria. The model predicted the wear scar would grow beyond the finely meshed contact zone after 110 cycles hence the simulation was stopped.

6.8 Discussion

It was found that the two material evolution criteria using i) von Mises threshold and ii) vertical stress threshold grossly over-predicted fretting wear compared to experiments. There is abundant evidence for the large plastic flow of material under fretting, for instance, the pile-up of material at the edge of wear scars. This would suggest, at least for the case modelled here, that the material experiences significant plastic deformation prior to evolution out from the surface. Neither criterion allows for flow before evolution.

The equivalent plastic strain criterion allows for some plastic deformation to take place in the material before being removed, and it produces much smaller fretting wear rates than von Mises or vertical stress, see section 6.5. It still, however, over-predicts vs experiments. A larger threshold for the criterion may have allowed more accurate prediction vs experiment, but the value implemented herein was based on experimental data itself [104]. Alternatively, the over-prediction could be inherent in how the equivalent plastic strain is calculated. That is, the equivalent plastic strain criterion reflects accumulated plastic strains in all directions, whereas it may be the case that plastic strains in some directions do

not contribute to the cause of failure, i.e. it is fundamentally unsuited to predict failure in this situation.

The Dissipated Energy method markedly underestimates fretting wear loss, see Chapter Five. Since the energy dissipated is spent in overcoming friction resisting relative movement of the first bodies, there is no energy expended in other directions, unlike the plastic strains. The value of the coefficient used in this criterion was taken from the best experimental data available [61]. As noted in Chapter 5, the curve for the Dissipated Energy criterion seems like it will intersect with the experimental data at approximately 1000 cycles, which is also the cycle at which wear coefficients were measured in the original work.

The critical plane approach was implemented following the method by Madge [4], which calculates fretting wear via estimating the SWT damage parameter, i.e. energy in the direction normal to the critical plane. It is found that the critical plane approach under-predicts the fretting wear, see section 6.6, which can be because the SWT damage parameter is suitable for predicting the material failure under the tensile cracking (i.e. mode I) but not shear cracking (i.e. mode II especially relevant for a tangential fretting wear case) [205].

The shear stress evolution criterion uses a damage indicator to reflect applied shear stress as a fraction of ultimate shear stress, and as such is a threshold method. Compared to the other criteria, it predicts fretting wear well vs experimental results. This may indicate that the real cause of material evolution from surfaces such as these under fretting wear is a shear stress limit in the material at the surface or very near to the surface. This result indicates that the reciprocally relative motion causes shear stress accumulated. When the shear stress on the material reaches the limit, the material starts detaching or de-bonding from its host, i.e. the first bodies and becoming debris particles.

6.9 Conclusion

In this chapter, five different material evolution criteria are implemented in the model in an attempt to explore possible mechanisms for material evolution, resulted in different surface profiles and wear rates. It was found that, in the case modelled herein, the material experiences a lot of plastic deformation before being worn or detached from surfaces. The material evolution criteria of von Mises and vertical stress, which use yield stress as the threshold, over-predict the fretting wear. There are at least two orders of magnitude difference between wear depths predicted by the different criteria and by experiments, specifically the von Mises and normal stress. In order to explore the relationship between plasticity and material failure during fretting wear, a material evolution criteria of equivalent plasticity is applied. The model with criteria of equivalent plasticity still over-predicts the fretting wear which indicates the material failure during fretting wear can be caused by one component in one direction. The critical plane approach predicts fretting wear which is much lower than the result obtained from the experiments. This is because the damage parameter, i.e. SWT, calculates the normal energy which is unstable for the shear cracking mode. Compared to the criterion above, the material evolution criteria of shear stress predicts the fretting wear which is closest to the value from the fretting wear experiment.

Chapter 7
Conclusion

7.1 General Discussion

The work undertaken in this thesis examines fretting wear via numerical modelling. Attention is focused on studying the effect of debris particles or the third body in the Hertzian contact and exploring the non-empirical mechanism of fretting wear. A novel simulation method termed Hybrid Finite-Discrete Element Modelling (FDEM) was developed in this work to investigate the behaviour of debris particles in between the contact surface. The advantage of the FDEM method is its high efficiency to couple an FE code and a DE code, see Chapter 2 and Chapter 5. Such advantage addresses the shortcoming of the other numerical model about neglecting the effect of debris particles which is believed to influent the behaviour of fretting wear significantly. In addition, via implementing the DE code inside the model developed here, a great number of debris particles can be simulated which gives better estimation of reality compared to limited number of particles in FE code. Both properties of solid and discrete material simulated via the FDEM can be controlled individually. Another benefit of the FDEM here is its ability to simulate the debris particles with more degree of freedom than the other methods, which avoids misevaluating the evolution of debris particles. Thus, with the limited computational resource available, the FDEM model developed in this work is able to better simulate the behaviour of fretting wear without losing the information of debris particles.

In this work, a novel cycle-jumping method is developed and successfully implemented to save 60 percentages of cycles with less than 10 percentages difference compared to the result without cycle-jumping. On the contrary, the

cycle-jumping methods used in other work can fail to speed up the simulation or even greatly miss-predict the results. One main challenge of developing this cycle-jumping method is that the fretting wear rate predicted with material evolution criteria is noisy. The advantage of the cycle-jumping method is its ability to handle noise via an advanced fitting algorithm and nonlinearities. In addition, a self-correcting algorithm during the fretting wear calculation is also included inside the algorithm which makes the cycle-jumping method more intelligent so as to not loss the original data features, see Figure 4.15 and 4.16 for more details. It should be noticed that this method is also generalized, and its performance could vary between cases. But that method can be applied to different scenarios with little modification.

With the help of this FDEM method developed in this work, it was found that the introduction of the debris particles is essential for fretting wear. Two identical models with and without debris particles are compared in this work. Change in the value of local contact force is evident with the presence of debris particles. The debris particles tend to reduce peak forces (shown in Figure 5.24). Via the FDEM method, the debris particles are able to be directly included in the contact with both first bodies. As fretting wear progresses or the number of debris particles increased, the debris particles tend to even out the distribution of contact pressure. According to the results found in this work, the effect of debris particles can be significantly important in wear rate prediction. When an empirical method is applied, e.g. Archard equation, the wear rate calculated can be reduced when the local contact force/pressure is reduced. Such evidence is able to support the finding of non-linear behaviour of fretting wear as the wear rate is reduced at the later stage of fretting wear or a steady stage is found during the fretting wear. Such effects can become remarkable when a threshold is used as the material

evolution criteria. One potential application of such find can assist the engineers to design the shape of contact joints. In addition, a calculated volume of particles can be placed between contacting surfaces on purpose in order to obtain a small wear rate rather than a large one between two virgin surfaces.

It was also found that the Dissipated Energy method is insufficient to predict the non-linearity of fretting wear observed in experiments with Hertzian contact. As discussed in Chapter 5, the result predicted by the Dissipated Energy method greatly underestimates the fretting wear compared to the experimental result. The predicted result behaves linearly rather than nonlinearly observed in the experiments. The biggest underestimation occurs at the transient point as shown in Figure 5.15. However, the result predicted by the Dissipated Energy method is likely to agree with the experimental data from which cycle the energy wear coefficient is calculated. It is caused by the implementation of an energy wear coefficient as a secant tangent. The result obtained using the model in this work indicates the failure of applying an empirical coefficient to predict fretting wear. A question mark should be placed on implementing the Dissipated Energy method to predict the fretting wear. According to the results found in this work, it can be seen that utilizing an constant coefficient to represent a non-linear behaviour can significantly mislead the prediction.

With the advantages of this method developed herein, the mechanism of fretting wear is explored via implementing five different material evolution rules, i.e. the von Mises, internal vertical stress, equivalent plastic strain, fatigue failure via Critical Plane Approach, and internal on-plane shear stress. The material evolution criteria of von Mises and vertical stress over-predict the wear rate compared to the result obtained from the experiment, see Section 6.3 and Section 6.4. Such result indicates, at least for the case modelled here, that the material

experiences significant plastic deformation prior to evolution out from the surface. The material evolution criterion of equivalent plastic strain gives a better prediction compared to the previous two criteria but still over-predicts the fretting wear compared to the experimental data, see Section 6.5. Such result could be caused by the inclusion of the component of plastic strain that does not actually contribute to material failure during fretting wear, which reflects its unsuitability to predict fretting wear. A Critical Plane Approach with SWT damage under-predicts the fretting wear because the SWT damage parameter calculates the normal energy which may not be suitable for shear dominate failure, see Section 6.6. Compared to the other criteria, it predicts fretting wear well vs experimental result, i.e. crossed cylinders, see Section 6.7.

The material evolution criteria of shear stress calculates the material failure based on the relation between shear stress and ultimate shear stress. This may indicate that the real cause of material evolution from surfaces such as these under fretting wear is a shear stress limit in the material at or very near to the surface. And the shear stress based material evolution criteria agrees with the principle of the dissipated energy method as both of them describe the tangential work done between the contacting surfaces. Under the reciprocating motion, i.e. fretting motion, the repeated shear stress weakens the material until the material worn occurs as indicated by Equation 6.9. Since, the shear stress can be altered by changing factors like the normal force and friction of coefficient. Thus, in order to prevent or reduce the damage caused by the fretting wear, action such like reducing the contact friction between contacting surfaces, and reducing the surface roughness can be taken to reduce the internal shear stress. Lubrication can also be applied, using liquid for example, can also reduce the fretting wear via reducing the friction between the contacting surfaces.

The modelling method developed in this work is able to explore the physical mechanism of fretting wear as well as predict the service life of a structural component subject to fretting. The study in Section Six highlights the ability of this modelling method to adapt different material evolution criterion. Thus, different results (i.e. wear scar or wear rate) predicted by the model can be compared to the data measured from experiments so as to help ascertain the mechanism of fretting wear. In addition to the wear mechanism implemented in this work, other material evolution criteria can also be applied in parallel. As a result, the criteria which is triggered firstly causes the material failure and geometry updated. Such implementation requires little modification which is quick and straight forward. With such function, this model will help engineers to identify the failure mechanism of structures as well as avoid potentially dangerous operating.

7.2 Recommendations for Future Work

According to the results obtained from the model developed in this work, it is found that the presence of debris particles changes the interaction between first bodies which influence local details of the contact forces, in particular reducing maxima. Although this FDEM model did not aim at providing a precise estimation of the interaction between one debris particle and first body surface. It could be refined to simulate the interaction between the first body and each debris particles with more fidelity. Such refinement would require implementation of a multi-scale coupling strategy: the interactions between each single debris particles with first bodies will be simulated in a micro-model whose results is then passed to and mapped into a macro-model where the global wear calculation is conducted. The advantage of this strategy is able to investigate the behaviour of each debris particles in a different region of the contacting zone. However, more

computational resource will then be required for such method. Thus, reasonable assumptions and a coupling strategy, e.g. dictionary-searching method, is required.

One deeper coupling algorithm can be developed by taking advantages of open-source FE and DE plateform. This allows more degrees of freedom in controlling the data flow and data management inside the code. A deeper understanding about the data transferring between DE particles and FE elements is required to achieve this study. With such FE and DE code, a parallel computing rather than a sequence computing can be achieved which means great reduction in simulation time.

Due to the limit of time of this project, the effect of debris particles on the different kinds of fretting wear was not investigated. For example, in the work by Fouvry et al., different wear profiles were observed under different values of fretting frequencies and loads. It can be interesting to investigate the effect of frequency and load on the evolution of debris particles and further effect on the wear behaviour. Another interesting topic can be fretting wear with water layer between contacting surface, because the water layer can influence the evolution the debris particles which may affect the fretting wear.

In this project, the properties of debris particles are referred from literature and assumed to be fixed during the progress of fretting wear. However, in reality, such assumption is not true. Thus, it can be interesting to conduction a further research on studying the properties of debris particles like the shape, material, hardness, and volumes at different stages of fretting wear. Such research can contribute to the development of the numerical model as well as understanding the evolution of debris particles during the fretting wear.

This project studied the fretting wear between steels which is most common material used in industry. The material property of the metal is uniform along each direction. However, that is not true for composite material like carbon fiber. With the development of technology, more and more composite materials gain their applications for the unique properties. The wear behaviour can be significantly different along different directions between carbon fibers. It then becomes interesting to investigate the behaviour of fretting between composite materials.

According to the result produced by the model in this work, the internal on-plane shear stress is believed to be the critical factor which affects the material failure during fretting wear. In addition to the computational model, the effect of shear stress in fretting wear can be experimentally studied. To achieve this, residential shear stress can be applied to experimental samples by applying a torsion load. A series of fretting wear experiments with different residential shear stress on samples can be conducted to further investigate the effect of shear stress on fretting wear. It is expected that varying the value of residential shear stress will influence the fretting wear rate.

In addition, the results obtained in the work herein indicates the failure of the dissipated energy method because of the implementation of the energy wear coefficient applied. Fretting wear has been observed as a non-linear behaviour while using a constant coefficient can lead to the failure of prediction. In the work of Fouvry et al., the wear coefficient is measured at one specific number of cycle without investigating the variation of the wear coefficient over different number of fretting cycles. It is very important to understand the behaviour of the energy wear coefficient because this can reveal more information about the relationship between the work done upon the surfaces and the volume of material worn from that surfaces during the fretting motion.

Material coatings have become a common method for preventing wear. In addition, failure of the coating can be not limited to wear but also delamination and sub-surface cracking. The numerical model developed in this work shall be modified in order to be applicable for modelling the material with the coating. Investigating the failure mechanism for different materials and sub-surface material failure will be critical for accurately predicting the life for materials.

Furthermore, the results in this work indicates the reduction of local contact stress because of the debris particles. A further study can focus on the relationship between the volume of debris particles and wear rate. The research strategy can be placing different volumes of debris particles into the space between the contacting surfaces and then measure the volume/mass reduced from the first bodies after a number of fretting cycles. The existence of debris particles between first bodies may help reducing the wear rate under certain conditions. Via such study, it can help engineers to avoid or reduce the wear between contacting surfaces via manually place certain amount of debris particles between them.

In the numerical investigations presented above, the value of the coefficient of friction is assumed to remain constant during the fretting wear which however, in reality, is not true. It is possible to apply a dynamic value of COF according to other variables such as contact pressure. In the work by Naboulsi et al., the value of COF inside an FE model was ascertained based on the local contact pressure and slip [209].

The numerical tool was developed for practical use in analysing real engineering components. The numerical model developed in work has shown potential for predicting fretting wear compared to the experimental results. At the time of writing, sufficient computational resource was not available for applying

this numerical model to geometries with complex shapes. With the increasing of computational speed, it will become possible to implement this numerical tool to study complex geometries and help better designing those engineering components. Different strategies can be applied like using HPC or reducing the dimension of model.

Reference

[1] H. P. Jost, “Tribology — Origin and future,” *Wear*, vol. 136, no. 1, pp. 1–17, Feb. 1990.

[2] P. L. Hurricks, “The mechanism of fretting — A review,” *Wear*, vol. 15, no. 6, pp. 389–409, Jun. 1970.

[3] R. . Waterhouse, *Fretting Corrosion*. Dordrecht: Oxford, New York, Pergamon Press, 1972.

[4] J. J. Madge, “Numerical modelling of the effect of fretting wear on fretting fatigue,” University of Nottingham, 2008.

[5] L. Kreines, G. Halperin, I. Etsion, M. Varenberg, A. Hoffman, and R. Akhvlediani, “Fretting wear of thin diamond films deposited on steel substrates,” *Diam. Relat. Mater.*, vol. 13, no. 9, pp. 1731–1739, Sep. 2004.

[6] W. ; Wallace, D. W. Hoeppner, and P. . Kandachar, “AGARD Corrosion Handbook. Volume 1. Aircraft Corrosion: Causes and Case Histories,” 1985.

[7] M. Helmi Attia, “Fretting fatigue and wear damage of structural components in nuclear power stations—Fitness for service and life management perspective,” *Tribol. Int.*, vol. 39, no. 10, pp. 1294–1304, Oct. 2006.

[8] R. Waterhouse, “Occurrence of Fretting in Practice and Its Simulation in the Laboratory,” in *Materials Evaluation Under Fretting Conditions*, 100 Barr Harbor Drive, PO Box C700, West Conshohocken, PA 19428-2959: ASTM International, pp. 3-3–14.

[9] “Pratt and Whitney PW1100G Geared Turbofan Engine,” *The Flying Engineer*, 2013. [Online]. Available: http://theflyingengineer.com/flightdeck/pw1100g-gtf/. [Accessed: 08-May-2017].

[10] J. F. Archard, “Contact and Rubbing of Flat Surfaces,” *J. Appl. Phys.*, vol. 24, no. 8, p. 981, 1953.

[11] H. C. Meng and K. C. Ludema, “Wear models and predictive equations: their form and content,” *Wear*, vol. 181–183, pp. 443–457, Mar. 1995.

[12] S. Fouvry, P. Kapsa, and L. Vincent, “Quantification of fretting damage,” *Wear*, vol. 200, no. 1–2, pp. 186–205, Dec. 1996.

[13] P. L. Hurricks and K. S. Ashford, “The effect of temperature on the fretting wear of mild steel,” *Arch. Proc. Inst. Mech. Eng. Conf. Proc. 1964-1970 (vols 178-184), Var. titles Label. Vol. A to S*, vol. 184, no. 312, pp. 165–175, Sep. 1969.

[14] I. FENG and H. UHLIG, “FRETTING CORROSION OF MILD STEEL IN AIR AND IN NITROGEN,” *J. Appl. Mech. ASME1*, vol. 21, no. 4, pp. 395–

400, 1954.
[15] H. UHLIG, “MECHANISM OF FRETTING CORROSION,” *J. Appl. Mech. ASME*, vol. 21, no. 4, pp. 401–407, Dec. 1954.
[16] B. Bethune and R. Waterhouse, “Electrochemical studies of fretting corrosion,” *Wear*, vol. 12, no. 1, pp. 27–34, Jul. 1968.
[17] J. Dobromirski, “Variables of Fretting Process: Are There 50 of Them?,” in *Standardization of Fretting Fatigue Test Methods and Equipment*, 100 Barr Harbor Drive, PO Box C700, West Conshohocken, PA 19428-2959: ASTM International, pp. 60-60–7.
[18] J. Ding, I. R. McColl, S. B. Leen, and P. H. Shipway, “A finite element based approach to simulating the effects of debris on fretting wear,” *Wear*, vol. 263, no. 1–6, pp. 481–491, Sep. 2007.
[19] J. A. Williams, “Wear and wear particles—some fundamentals,” *Tribol. Int.*, vol. 38, no. 10, pp. 863–870, Oct. 2005.
[20] Y. Berthier, C. Colombié, L. Vincent, and M. Godet, “Fretting Wear Mechanisms and Their Effects on Fretting Fatigue,” *J. Tribol.*, vol. 110, no. 3, p. 517, 1988.
[21] H.-J. Kim, S.-S. Yoo, and D.-E. Kim, “Nano-scale wear: A review,” *Int. J. Precis. Eng. Manuf.*, vol. 13, no. 9, pp. 1709–1718, Sep. 2012.
[22] M. Scherge, D. Shakhvorostov, and K. Pöhlmann, “Fundamental wear mechanism of metals,” *Wear*, vol. 255, no. 1–6, pp. 395–400, Aug. 2003.
[23] C. Subramanian, “Effects of sliding speed on the unlubricated wear behaviour of Al-12.3wt.%Si alloy,” *Wear*, vol. 151, no. 1, pp. 97–110, Nov. 1991.
[24] G. W. (Gwidon W. . Stachowiak, *Wear--materials, mechanisms and practice*. Wiley, 2005.
[25] M. Long and H. . Rack, “Titanium alloys in total joint replacement—a materials science perspective,” *Biomaterials*, vol. 19, no. 18, pp. 1621–1639, Sep. 1998.
[26] S. Söderberg, S. Nikoonezhad, K. Salama, and O. Vingsbo, “Accelerated fretting wear testing using ultrasonics,” *Ultrasonics*, vol. 24, no. 6, pp. 348–353, Nov. 1986.
[27] Z. Cai, M. Zhu, and Z. Zhou, “An experimental study torsional fretting behaviors of LZ50 steel,” *Tribol. Int.*, vol. 43, pp. 361–369, Jan. 2010.
[28] M. Godet, “The third-body approach: A mechanical view of wear,” *Wear*, vol. 100, no. 1–3, pp. 437–452, Dec. 1984.
[29] M. Godet, “Third-bodies in tribology,” *Wear*, vol. 136, no. 1, pp. 29–45, Feb. 1990.
[30] M. Godet, Y. Berthier, J. Lancaster, and L. Vincent, “Wear modelling: Using fundamental understanding or practical experience?,” *Wear*, vol. 149, no. 1–2, pp. 325–340, Sep. 1991.

[31] A. Iwabuchi, “The role of oxide particles in the fretting wear of mild steel,” *Wear*, vol. 151, no. 2, pp. 301–311, Dec. 1991.
[32] M. Varenberg, G. Halperin, and I. Etsion, “Different aspects of the role of wear debris in fretting wear,” *Wear*, vol. 252, no. 11–12, pp. 902–910, Jul. 2002.
[33] J. Warburton, “The fretting of mild steel in air,” *Wear*, vol. 131, no. 2, pp. 365–386, Jun. 1989.
[34] J. F. Archard and W. Hirst, “The Wear of Metals under Unlubricated Conditions,” *Proc. R. Soc. A Math. Phys. Eng. Sci.*, vol. 236, no. 1206, pp. 397–410, Aug. 1956.
[35] S. M. Hsu and M. C. Shen, “Wear mapping of materials,” in *Wear-Materials, Mechanisms and Practice*, Chichester, England: John Wiley & Sons Ltd, 2005, pp. 369–423.
[36] S. C. Lim, Y. B. Liu, S. H. Lee, and K. H. W. Seah, “Mapping the wear of some cutting-tool materials,” *Wear*, vol. 162–164, pp. 971–974, Apr. 1993.
[37] Z. R. Zhou, K. Nakazawa, M. H. Zhu, N. Maruyama, P. Kapsa, and L. Vincent, “Progress in fretting maps,” *Tribol. Int.*, vol. 39, no. 10, pp. 1068–1073, Oct. 2006.
[38] S. M. Hsu and M. C. Shen, “Ceramic wear maps,” *Wear*, vol. 200, no. 1–2, pp. 154–175, Dec. 1996.
[39] S. M. Hsu, Y. S. Wang, and R. G. Munro, “Quantitative wear maps as a visualization of wear mechanism transitions in ceramic materials,” *Wear*, vol. 134, no. 1, pp. 1–11, Nov. 1989.
[40] S. . Lim, “Recent developments in wear-mechanism maps,” *Tribol. Int.*, vol. 31, no. 1–3, pp. 87–97, Jan. 1998.
[41] S. C. Lim, S. H. Lee, Y. B. Liu, and K. H. W. Seah, “Wear maps for uncoated high-speed steel cutting tools,” *Wear*, vol. 170, no. 1, pp. 137–144, Nov. 1993.
[42] O. Vingsbo and S. Söderberg, “On fretting maps,” *Wear*, vol. 126, no. 2, pp. 131–147, Sep. 1988.
[43] S. Fouvry, P. Kapsa, and L. Vincent, “Fretting-Wear and Fretting-Fatigue: Relation Through a Mapping Concept,” in *Fretting Fatigue: Current Technology and Practices*, 100 Barr Harbor Drive, PO Box C700, West Conshohocken, PA 19428-2959: ASTM International, pp. 49-49–16.
[44] Z. R. Zhou, S. Fayeulle, and L. Vincent, “Cracking behaviour of various aluminium alloys during fretting wear,” *Wear*, vol. 155, no. 2, pp. 317–330, Jun. 1992.
[45] S. Garcin, S. Fouvry, and S. Heredia, “A FEM fretting map modeling: Effect of surface wear on crack nucleation,” *Wear*, vol. 330–331, pp. 145–159, May 2015.

[46] S. Fouvry, P. Arnaud, A. Mignot, and P. Neubauer, “Contact size, frequency and cyclic normal force effects on Ti–6Al–4V fretting wear processes: An approach combining friction power and contact oxygenation,” *Tribol. Int.*, vol. 113, pp. 460–473, Sep. 2017.
[47] Z. R. Zhou and L. Vincent, “Mixed fretting regime,” *Wear*, vol. 181–183, pp. 531–536, Mar. 1995.
[48] S. C. Lim and M. F. Ashby, “Overview no. 55 Wear-Mechanism maps,” *Acta Metall.*, vol. 35, no. 1, pp. 1–24, Jan. 1987.
[49] T. H. C. Childs, “The Mapping of Metallic Sliding Wear,” *Proc. Inst. Mech. Eng. Part C J. Mech. Eng. Sci.*, vol. 202, no. 6, pp. 379–395, Nov. 1988.
[50] M. F. Ashby and S. C. Lim, “Wear-mechanism maps,” *Scr. Metall. Mater.*, vol. 24, no. 5, pp. 805–810, May 1990.
[51] Y. Liu, R. Asthana, and P. Rohatgi, “A map for wear mechanisms in aluminium alloys,” *J. Mater. Sci.*, vol. 26, no. 1, pp. 99–102.
[52] F. . Barwell, “Lubrication of bearings,” *Wear*, vol. 1, no. 3, p. 257, Dec. 1957.
[53] S. K. Rhee, “Wear equation for polymers sliding against metal surfaces,” *Wear*, vol. 16, no. 6, pp. 431–445, Dec. 1970.
[54] R. G. Bayer, W. C. Clinton, C. W. Nelson, and R. A. Schumacher, “Engineering model for wear,” *Wear*, vol. 5, no. 5, pp. 378–391, Sep. 1962.
[55] E. Hornbogen, “The role of fracture toughness in the wear of metals,” *Wear*, vol. 33, no. 2, pp. 251–259, Jul. 1975.
[56] N. P. Suh, “The delamination theory of wear,” *Wear*, vol. 25, no. 1, pp. 111–124, Jul. 1973.
[57] B. D. Leonard, P. Patil, T. S. Slack, F. Sadeghi, S. Shinde, and M. Mittelbach, “Fretting Wear Modeling of Coated and Uncoated Surfaces Using the Combined Finite-Discrete Element Method,” *J. Tribol.*, vol. 133, no. 2, p. 021601, 2011.
[58] A. L. Mohd Tobi, P. H. Shipway, and S. B. Leen, “Gross slip fretting wear performance of a layered thin W-DLC coating: Damage mechanisms and life modelling,” *Wear*, vol. 271, no. 9–10, pp. 1572–1584, Jul. 2011.
[59] I. R. McColl, J. Ding, and S. B. Leen, “Finite element simulation and experimental validation of fretting wear,” *Wear*, vol. 256, no. 11–12, pp. 1114–1127, 2004.
[60] “Abaqus Analysis User’s Guide. Abaqus 6.13.”
[61] S. Fouvry, T. Liskiewicz, P. Kapsa, S. Hannel, and E. Sauger, “An energy description of wear mechanisms and its applications to oscillating sliding contacts,” *Wear*, vol. 255, no. 1–6, pp. 287–298, Aug. 2003.

[62] S. Fouvry, T. Liskiewicz, and C. Paulin, “A global–local wear approach to quantify the contact endurance under reciprocating-fretting sliding conditions,” *Wear*, vol. 263, no. 1–6, pp. 518–531, Sep. 2007.
[63] B. D. Leonard, F. Sadeghi, S. Shinde, and M. Mittelbach, “A Numerical and Experimental Investigation of Fretting Wear and a New Procedure for Fretting Wear Maps,” *Tribol. Trans.*, vol. 55, no. 3, pp. 313–324, May 2012.
[64] A. Cruzado, M. A. Urchegui, and X. Gómez, “Finite element modeling and experimental validation of fretting wear scars in thin steel wires,” *Wear*, vol. 289, pp. 26–38, Jun. 2012.
[65] L. Gallego and D. Nélias, “Modeling of Fretting Wear Under Gross Slip and Partial Slip Conditions,” *J. Tribol.*, vol. 129, no. 3, p. 528, 2007.
[66] M. Yetisir and N. J. Fisher, “Prediction of pressure tube fretting-wear damage due to fuel vibration,” *Nucl. Eng. Des.*, vol. 176, no. 3, pp. 261–271, Nov. 1997.
[67] A. V. Dimaki, A. I. Dmitriev, Y. S. Chai, and V. L. Popov, “Rapid simulation procedure for fretting wear on the basis of the method of dimensionality reduction,” *Int. J. Solids Struct.*, vol. 51, no. 25–26, pp. 4215–4220, Dec. 2014.
[68] J. Ding, S. B. Leen, E. J. Williams, and P. H. Shipway, “Finite element simulation of fretting wear-fatigue interaction in spline couplings,” *Tribol. - Mater. Surfaces Interfaces*, vol. 2, no. 1, pp. 10–24, Mar. 2008.
[69] S. B. Leen, T. H. Hyde, C. H. H. Ratsimba, E. J. Williams, and I. R. McColl, “An investigation of the fatigue and fretting performance of a representative aero-engine spline coupling,” *J. Strain Anal. Eng. Des.*, vol. 37, no. 6, pp. 565–583, Jan. 2002.
[70] A. Cruzado, S. B. Leen, M. A. Urchegui, and X. Gómez, “Finite element simulation of fretting wear and fatigue in thin steel wires,” *Int. J. Fatigue*, vol. 55, pp. 7–21, Oct. 2013.
[71] J. Ding, J. Madge, S. B. Leen, and E. J. Williams, “Towards the Modelling of Fretting Wear and Fatigue Interaction in Spline Couplings,” *Appl. Mech. Mater.*, vol. 5–6, pp. 165–172, 2006.
[72] S. B. Leen, I. J. Richardson, I. R. McColl, E. J. Williams, and T. R. Hyde, “Macroscopic fretting variables in a splined coupling under combined torque and axial load,” *J. Strain Anal. Eng. Des.*, vol. 36, no. 5, pp. 481–497, Jan. 2001.
[73] J. Ding, “The effect of slip regime on fretting wear-induced stress evolution,” *Int. J. Fatigue*, vol. 26, no. 5, pp. 521–531, May 2004.
[74] J. Ding, S. B. Leen, E. J. Williams, and P. H. Shipway, “A multi-scale model for fretting wear with oxidation-debris effects,” *Proc. Inst.*

Mech. Eng. Part J J. Eng. Tribol., vol. 223, no. 7, pp. 1019–1031, Nov. 2009.

[75] N. M. Everitt, J. Ding, G. Bandak, P. H. Shipway, S. B. Leen, and E. J. Williams, "Characterisation of fretting-induced wear debris for Ti-6Al-4 V," *Wear*, vol. 267, no. 1–4, pp. 283–291, Jun. 2009.

[76] J. Ding, S. B. Leen, E. J. Williams, and P. H. Shipway, "An Asperity-Contact Based Oxidation Model for Fretting Wear with the Presence of Debris," in *Advanced Tribology*, Berlin, Heidelberg: Springer Berlin Heidelberg, 2009, pp. 346–347.

[77] S. Fouvry and P. Kapsa, "An energy description of hard coating wear mechanisms," *Surf. Coatings Technol.*, vol. 138, no. 2–3, pp. 141–148, Apr. 2001.

[78] S. Basseville, E. Héripré, and G. Cailletaud, "Numerical simulation of the third body in fretting problems," *Wear*, vol. 270, no. 11–12, pp. 876–887, May 2011.

[79] Y. Berthier, "Experimental evidence for friction and wear modelling," *Wear*, vol. 139, no. 1, pp. 77–92, Jul. 1990.

[80] Y. Berthier, L. Vincent, and M. Godet, "Velocity accommodation in fretting," *Wear*, vol. 125, no. 1–2, pp. 25–38, Jul. 1988.

[81] M. Dragon-Louiset, "On a predictive macroscopic contact-sliding wear model based on micromechanical considerations," *Int. J. Solids Struct.*, vol. 38, no. 9, pp. 1625–1639, Feb. 2001.

[82] C. H. H. Ratsimba, I. R. McColl, E. J. Williams, S. B. Leen, and H. P. Soh, "Measurement, analysis and prediction of fretting wear damage in a representative aeroengine spline coupling," *Wear*, vol. 257, no. 11, pp. 1193–1206, Dec. 2004.

[83] K. L. Johnson, "Contact mechanics and the wear of metals," *Wear*, vol. 190, no. 2, pp. 162–170, Dec. 1995.

[84] A. Kapoor, "Wear by plastic ratchetting," *Wear*, vol. 212, no. 1, pp. 119–130, Nov. 1997.

[85] S. Fouvry, P. Kapsa, and L. Vincent, "Fretting behaviour of hard coatings under high normal load," *Surf. Coatings Technol.*, vol. 68–69, pp. 494–499, Dec. 1994.

[86] S. Fouvry, P. Kapsa, H. Zahouani, and L. Vincent, "Wear analysis in fretting of hard coatings through a dissipated energy concept," *Wear*, vol. 203–204, pp. 393–403, Mar. 1997.

[87] C. Mary and S. Fouvry, "Numerical prediction of fretting contact durability using energy wear approach: Optimisation of finite-element model," *Wear*, vol. 263, no. 1–6, pp. 444–450, Sep. 2007.

[88] C. Paulin, S. Fouvry, and S. Deyber, "Wear kinetics of Ti–6Al–4V under constant and variable fretting sliding conditions," *Wear*, vol. 259, no. 1–6, pp. 292–299, Jul. 2005.

[89] S. Fouvry and C. Paulin, “An effective friction energy density approach to predict solid lubricant friction endurance: Application to fretting wear,” *Wear*, vol. 319, no. 1–2, pp. 211–226, Nov. 2014.
[90] S. R. Pearson and P. H. Shipway, “Is the wear coefficient dependent upon slip amplitude in fretting? Vingsbo and Söderberg revisited,” *Wear*, vol. 330–331, pp. 93–102, May 2015.
[91] E. Sauger, S. Fouvry, L. Ponsonnet, P. Kapsa, J. . Martin, and L. Vincent, “Tribologically transformed structure in fretting,” *Wear*, vol. 245, no. 1–2, pp. 39–52, Oct. 2000.
[92] K. Elleuch and S. Fouvry, “Experimental and modelling aspects of abrasive wear of a A357 aluminium alloy under gross slip fretting conditions,” *Wear*, vol. 258, no. 1–4, pp. 40–49, Jan. 2005.
[93] C. Mary, S. Fouvry, J. M. Martin, and B. Bonnet, “Pressure and temperature effects on Fretting Wear damage of a Cu–Ni–In plasma coating versus Ti17 titanium alloy contact,” *Wear*, vol. 272, no. 1, pp. 18–37, Oct. 2011.
[94] Q. M. Li, “Strain energy density failure criterion,” *Int. J. Solids Struct.*, vol. 38, no. 38–39, pp. 6997–7013, Sep. 2001.
[95] A. L. Mohd Tobi, J. Ding, G. Bandak, S. B. Leen, and P. H. Shipway, “A study on the interaction between fretting wear and cyclic plasticity for Ti–6Al–4V,” *Wear*, vol. 267, no. 1–4, pp. 270–282, Jun. 2009.
[96] O. J. McCarthy, J. P. McGarry, and S. B. Leen, “Micro-mechanical modelling of fretting fatigue crack initiation and wear in Ti–6Al–4V,” *Int. J. Fatigue*, vol. 62, pp. 180–193, May 2014.
[97] O. J. McCarthy, J. P. McGarry, and S. B. Leen, “A finite element study of microstructure-sensitive plasticity and crack nucleation in fretting,” *Comput. Mater. Sci.*, vol. 50, no. 8, pp. 2439–2458, Jun. 2011.
[98] F. A. McClintock, “A Criterion for Ductile Fracture by the Growth of Holes,” *J. Appl. Mech.*, vol. 35, no. 2, p. 363, Jun. 1968.
[99] J. R. Rice and D. M. Tracey, “On the ductile enlargement of voids in triaxial stress fields∗,” *J. Mech. Phys. Solids*, vol. 17, no. 3, pp. 201–217, Jun. 1969.
[100] P. G. Kossakowski, “STRESS MODIFIED CRITICAL STRAIN CRITERION FOR S235JR STEEL AT LOW INITIAL STRESS TRIAXIALITY,” *J. Theor. Appl. Mech.*, vol. 52, no. 4, pp. 995–1006, 2014.
[101] R. W. (Robert W. Revie, *Oil and gas pipelines : integrity and safety handbook*. .
[102] E. Corona and B. Reedlunn, “A Review of Macroscopic Ductile Failure Criteria,” 2013.
[103] J. W. Hancock and A. C. Mackenzie, “On the mechanisms of ductile failure in high-strength steels subjected to multi-axial stress-states,”

J. Mech. Phys. Solids, vol. 24, no. 2–3, pp. 147–160, Jun. 1976.
[104] M. Gao, R. Krishnamurthy, S. Tandon, and U. Arumugam, "Critical Strain Based Ductile Damage Criterion and its Application to Mechanical Damage in Pipelines."
[105] C.-H. Goh, R. W. Neu, and D. L. McDowell, "Crystallographic plasticity in fretting of Ti–6AL–4V," *Int. J. Plast.*, vol. 19, no. 10, pp. 1627–1650, Oct. 2003.
[106] J. R. Mayeur, D. L. McDowell, and R. W. Neu, "Crystal plasticity simulations of fretting of Ti-6Al-4V in partial slip regime considering effects of texture," *Comput. Mater. Sci.*, vol. 41, no. 3, pp. 356–365, Jan. 2008.
[107] S. Basseville and G. Cailletaud, "An evaluation of the competition between wear and crack initiation in fretting conditions for Ti–6Al–4V alloy," *Wear*, vol. 328–329, pp. 443–455, Apr. 2015.
[108] A. Ghosh, "Analytical investigation of fretting wear with special emphasis on stress based models," *Open Access Diss.*, Mar. 2016.
[109] C. H. Hager, J. H. Sanders, and S. Sharma, "Unlubricated gross slip fretting wear of metallic plasma-sprayed coatings for Ti6Al4V surfaces," *Wear*, vol. 265, no. 3–4, pp. 439–451, Jul. 2008.
[110] V. Fridrici, S. Fouvry, and P. Kapsa, "Fretting wear behavior of a Cu–Ni–In plasma coating," *Surf. Coatings Technol.*, vol. 163–164, pp. 429–434, Jan. 2003.
[111] J. J. Madge, S. B. Leen, and P. H. Shipway, "The critical role of fretting wear in the analysis of fretting fatigue," *Wear*, vol. 263, no. 1–6, pp. 542–551, Sep. 2007.
[112] B. Zeise, R. Liebich, and M. Prölß, "Simulation of fretting wear evolution for fatigue endurance limit estimation of assemblies," *Wear*, vol. 316, no. 1–2, pp. 49–57, Aug. 2014.
[113] F. Shen, W. Hu, and Q. Meng, "A damage mechanics approach to fretting fatigue life prediction with consideration of elastic–plastic damage model and wear," *Tribol. Int.*, vol. 82, pp. 176–190, Feb. 2015.
[114] M. . Dubourg, A. Chateauminois, and B. Villechaise, "In situ analysis and modeling of crack initiation and propagation within model fretting contacts using polymer materials," *Tribol. Int.*, vol. 36, no. 2, pp. 109–119, Feb. 2003.
[115] A. Ghosh, B. Leonard, and F. Sadeghi, "A stress based damage mechanics model to simulate fretting wear of Hertzian line contact in partial slip," *Wear*, vol. 307, no. 1–2, pp. 87–99, Sep. 2013.
[116] Y. Berthier, L. Vincent, and M. Godet, "Fretting fatigue and fretting wear," *Tribol. Int.*, vol. 22, no. 4, pp. 235–242, Aug. 1989.
[117] K. Dang-Van, "Macro-Micro Approach in High-Cycle Multiaxial

Fatigue," in *Advances in Multiaxial Fatigue*, 100 Barr Harbor Drive, PO Box C700, West Conshohocken, PA 19428-2959: ASTM International, 1993, pp. 120-120–11.
[118] M. P. Szolwinski and T. N. Farris, "Mechanics of fretting fatigue crack formation," *Wear*, vol. 198, no. 1–2, pp. 93–107, Oct. 1996.
[119] R. C. Bill, "Fretting Wear and Fretting Fatigue—How Are They Related?," *J. Lubr. Technol.*, vol. 105, no. 2, p. 230, 1983.
[120] W. J. Plumbridge and D. A. Ryder, "The metallography of fatigue," *Metall. Rev.*, vol. 14, no. 1, pp. 119–142, Jan. 1969.
[121] W. J. Plumbridge, "Review: Fatigue-crack propagation in metallic and polymeric materials," *J. Mater. Sci.*, vol. 7, no. 8, pp. 939–962, Aug. 1972.
[122] A. Fatemi and D. F. Socie, "A CRITICAL PLANE APPROACH TO MULTIAXIAL FATIGUE DAMAGE INCLUDING OUT-OF-PHASE LOADING," *Fatigue Fract. Eng. Mater. Struct.*, vol. 11, no. 3, pp. 149–165, Mar. 1988.
[123] J. Park, "Evaluation of an energy-based approach and a critical plane approach for predicting constant amplitude multiaxial fatigue life," *Int. J. Fatigue*, vol. 22, no. 1, pp. 23–39, Jan. 2000.
[124] L. Susmel, "A simple and efficient numerical algorithm to determine the orientation of the critical plane in multiaxial fatigue problems," *Int. J. Fatigue*, vol. 32, no. 11, pp. 1875–1883, Nov. 2010.
[125] M. W. Brown and K. J. Miller, "A theory for fatigue failure under multiaxial stress-strain conditions," *Arch. Proc. Inst. Mech. Eng. 1847-1982 (vols 1-196)*, vol. 187, no. 1973, pp. 745–755, Jun. 1973.
[126] D. L. McDiarmid, "A SHEAR STRESS BASED CRITICAL-PLANE CRITERION OF MULTIAXIAL FATIGUE FAILURE FOR DESIGN AND LIFE PREDICTION," *Fatigue Fract. Eng. Mater. Struct.*, vol. 17, no. 12, pp. 1475–1484, Dec. 1994.
[127] Y. Wang and L. Susmel, "Critical plane approach to multiaxial variable amplitude fatigue loading," vol. 33322133, no. 10, pp. 345–35638, 2015.
[128] Z. Engin and D. Coker, "Comparison of Equivalent Stress Methods with Critical Plane Approaches for Multiaxial High Cycle Fatigue Assessment," *Procedia Struct. Integr.*, vol. 5, pp. 1229–1236, 2017.
[129] ASM International. Handbook Committee., *ASM handbook*. .
[130] M. F. Ashby, D. R. H. (David R. H. Jones, and M. F. Ashby, *Engineering materials 1 : an introduction to their properties and applications*. Butterworth-Heinemann, 1996.
[131] T. Mac, I. Fonseca Junior, and R. Magnabosco, "EVALUATION OF METHODS FOR ESTIMATING FATIGUE PROPERTIES APPLIED TO STAINLESS STEELS AND ALUMINUM ALLOYS," vol. 9, no. 4, pp. 284–

293, 2012.
[132] J. Park, “Detailed evaluation of methods for estimation of fatigue properties,” *Int. J. Fatigue*, vol. 17, no. 5, pp. 365–373, Jan. 1995.
[133] J. Clark. McMahon, “Predicting fatigue properties through hardness measurements /,” 2017.
[134] X. Mi *et al.*, “Investigation on fretting wear behavior of 690 alloy in water under various temperatures,” *Tribol. Int.*, vol. 100, pp. 400–409, Aug. 2016.
[135] Z. Cai, H. M. Meyer, C. Ma, M. Chi, H. Luo, and J. Qu, “Comparison of the tribological behavior of steel–steel and Si3N4–steel contacts in lubricants with ZDDP or ionic liquid,” *Wear*, vol. 319, no. 1–2, pp. 172–183, Nov. 2014.
[136] T. W. Liskiewicz, B. D. Beake, N. Schwarzer, and M. I. Davies, “Short note on improved integration of mechanical testing in predictive wear models,” *Surf. Coatings Technol.*, vol. 237, pp. 212–218, Dec. 2013.
[137] A. L. M. Tobi, W. Sun, and P. H. Shipway, “Evolution of plasticity-based wear damage in gross sliding fretting of a Ti-6Al-4V non-conforming contact,” *Tribol. Int.*, vol. 113, pp. 474–486, Sep. 2017.
[138] C. Colombié, Y. Berthier, A. Floquet, L. Vincent, and M. Godet, “Fretting: Load Carrying Capacity of Wear Debris,” *J. Tribol.*, vol. 106, no. 2, p. 194, Apr. 1984.
[139] J. D. Lemm, A. R. Warmuth, S. R. Pearson, and P. H. Shipway, “The influence of surface hardness on the fretting wear of steel pairs—Its role in debris retention in the contact,” *Tribol. Int.*, vol. 81, pp. 258–266, Jan. 2015.
[140] H. Ian and S. Philip, *Tribology : Friction and Wear of Engineering Materials*, Edition 2. 2017.
[141] A. Zmitrowicz and Z. Alfred, “Wear debris: a review of properties and constitutive models.,” *J. Theor. Appl. Mech.*, vol. 43, no. 1, pp. 3–35, 2005.
[142] H. G. Elrod and D. E. Brewe, “Paper V (iv) Numerical Experiments with Flows of Elongated Granules,” 1992, pp. 219–226.
[143] W. G. Sawyer and J. A. Tichy, “Lubrication With Granular Flow: Continuum Theory, Particle Simulations, Comparison With Experiment,” *J. Tribol.*, vol. 123, no. 4, p. 777, 2001.
[144] A. A. Lubrecht, C. Chan-Tien, and Y. Berthier, “A Simple Model for Granular Lubrication; Influence of Boundaries,” 1996, pp. 377–385.
[145] P. A. Cundall and O. D. L. Strack, “A discrete numerical model for granular assemblies,” *Géotechnique*, vol. 29, no. 1, pp. 47–65, Mar. 1979.
[146] N. Fillot, I. Iordanoff, and Y. Berthier, “Modelling third body flows

with a discrete element method—a tool for understanding wear with adhesive particles," *Tribol. Int.*, vol. 40, no. 6, pp. 973–981, Jun. 2007.
[147] N. Fillot, I. Iordanoff, and Y. Berthier, "Kinetics of particle detachment: contribution of a granular model," 2003, pp. 63–73.
[148] N. Fillot, I. Iordanoff, and Y. Berthier, "Simulation of Wear Through Mass Balance in a Dry Contact," *J. Tribol.*, vol. 127, no. 1, p. 230, 2005.
[149] N. Fillot, I. Iordanoff, and Y. Berthier, "A Granular Dynamic Model for the Degradation of Material," *J. Tribol.*, vol. 126, no. 3, p. 606, 2004.
[150] I. Iordanoff, N. Fillot, and Y. Berthier, "Numerical study of a thin layer of cohesive particles under plane shearing," *Powder Technol.*, vol. 159, no. 1, pp. 46–54, Nov. 2005.
[151] N. Fillot, I. Iordanoff, and Y. Berthier, "Wear modeling and the third body concept," *Wear*, vol. 262, no. 7–8, pp. 949–957, Mar. 2007.
[152] I. Iordanoff, D. Richard, M. Renouf, and Y. Berthier, "Friction Coefficient as a Macroscopic View of Local Dissipation," in *ASME/STLE 2007 International Joint Tribology Conference, Parts A and B*, 2007, pp. 1–3.
[153] M. Renouf, F. Massi, N. Fillot, and A. Saulot, "Numerical tribology of a dry contact," *Tribol. Int.*, vol. 44, no. 7–8, pp. 834–844, Jul. 2011.
[154] I. Iordanoff, Y. Berthier, S. Descartes, and H. Heshmat, "A Review of Recent Approaches for Modeling Solid Third Bodies," *J. Tribol.*, vol. 124, no. 4, p. 725, 2002.
[155] V. Linck, L. Baillet, and Y. Berthier, "Modeling the consequences of local kinematics of the first body on friction and on third body sources in wear," *Wear*, vol. 255, no. 1–6, pp. 299–308, Aug. 2003.
[156] I. Iordanoff, A. Battentier, J. Néauport, and J. L. Charles, "A discrete element model to investigate sub-surface damage due to surface polishing," *Tribol. Int.*, vol. 41, no. 11, pp. 957–964, Nov. 2008.
[157] I. Iordanoff, B. Seve, and Y. Berthier, "Solid Third Body Analysis Using a Discrete Approach: Influence of Adhesion and Particle Size on Macroscopic Properties," *J. Tribol.*, vol. 124, no. 3, p. 530, 2002.
[158] A. Jensen, K. Fraser, and G. Laird, "3 th International LS-DYNA Users Conference Improving the Precision of Discrete Element Simulations through Calibration Models 13 th International LS-DYNA Users Conference," 2014.
[159] B. Bhuvaraghan, S. M. Srinivasan, B. Maffeo, R. D. McCLain, Y. Potdar, and O. Prakash, "Shot peening simulation using discrete and finite element methods," *Adv. Eng. Softw.*, vol. 41, no. 12, pp. 1266–1276, Dec. 2010.
[160] J. L. Choi and D. T. Gethin, "A discrete finite element modelling and

measurements for powder compaction," *Model. Simul. Mater. Sci. Eng.*, vol. 17, no. 3, p. 35005, 2009.
[161] H. A. Carmona, F. K. Wittel, F. Kun, and H. J. Herrmann, "Fragmentation processes in impact of spheres," *Phys. Rev. E*, vol. 77, no. 5, p. 051302, May 2008.
[162] A. Wang, L. Hu, Y. Fu, T. Jiang, and G. Liu, "Failure Model of Fretting Wear Based on Self-Organizing Method," in *2008 Fifth International Conference on Fuzzy Systems and Knowledge Discovery*, 2008, pp. 565–569.
[163] N. RAJE, T. SLACK, and F. SADEGHI, "A discrete damage mechanics model for high cycle fatigue in polycrystalline materials subject to rolling contact," *Int. J. Fatigue*, vol. 31, no. 2, pp. 346–360, Feb. 2009.
[164] A. Munjiza, *The Combined Finite-Discrete Element Method*. Chichester, UK: John Wiley & Sons, Ltd, 2004.
[165] A. Munjiza, D. R. J. Owen, and N. Bicanic, "A combined finite-discrete element method in transient dynamics of fracturing solids," *Eng. Comput.*, vol. 12, no. 2, pp. 145–174, Feb. 1995.
[166] S. Moharnmadi, D. R. J. Owen, and D. Peric, "A combined finite/discrete element algorithm for delamination analysis of composites," *Finite Elem. Anal. Des.*, vol. 28, no. 4, pp. 321–336, Mar. 1998.
[167] D. O. Potyondy and P. A. Cundall, "A bonded-particle model for rock," *Int. J. Rock Mech. Min. Sci.*, vol. 41, no. 8, pp. 1329–1364, Dec. 2004.
[168] R. P. Young *et al.*, "An innovative 3-D numerical modelling procedure for simulating repository-scale excavations in rock-SAFETY," in *Proceedings of the Euradwaste04 Conference on Radioactive Waste Management Community Policy and Research Initiatives*, 2004.
[169] J. Rojek and E. Onate, "Multiscale analysis using a coupled discrete/finite element model," *Interact. multiscale Mech.*, vol. 1, no. 1, pp. 1–31, Mar. 2008.
[170] W. Li, Y. Huang, B. Fu, Y. Cui, and S. Dong, "Fretting damage modeling of liner-bearing interaction by combined finite element – discrete element method," *Tribol. Int.*, vol. 61, pp. 19–31, May 2013.
[171] H.-P. Cao, M. Renouf, F. Dubois, and Y. Berthier, "Coupling Continuous and Discontinuous Descriptions to Model First Body Deformation in Third Body Flows," *J. Tribol.*, vol. 133, no. 4, p. 041601, 2011.
[172] W. Wang, Y. Liu, G. Zhu, and K. Liu, "Using FEM–DEM coupling method to study three-body friction behavior," *Wear*, vol. 318, no. 1–2, pp. 114–123, Oct. 2014.

[173] K. Murugaratnam, S. Utili, and N. Petrinic, “A combined DEM–FEM numerical method for Shot Peening parameter optimisation,” *Adv. Eng. Softw.*, vol. 79, pp. 13–26, Jan. 2015.
[174] B. D. Leonard, A. Ghosh, F. Sadeghi, S. Shinde, and M. Mittelbach, “Third body modeling in fretting using the combined finite-discrete element method,” *Int. J. Solids Struct.*, vol. 51, no. 6, pp. 1375–1389, Mar. 2014.
[175] B. D. Leonard, F. Sadeghi, S. Shinde, and M. Mittelbach, “Rough surface and damage mechanics wear modeling using the combined finite-discrete element method,” *Wear*, vol. 305, no. 1–2, pp. 312–321, Jul. 2013.
[176] B. D. Leonard, F. Sadeghi, R. D. Evans, G. L. Doll, and P. J. Shiller, “Fretting of WC/a-C:H and Cr2N Coatings Under Grease-Lubricated and Unlubricated Conditions,” *Tribol. Trans.*, vol. 53, no. 1, pp. 145–153, 2010.
[177] M. Hopkins, “Numerical Simulation of Systems of Multinidious Polygonal Blocks,” 1992.
[178] D. R. Mumm, A. G. Evans, and I. T. Spitsberg, “Characterization of a cyclic displacement instability for a thermally grown oxide in a thermal barrier system,” *Acta Mater.*, vol. 49, no. 12, pp. 2329–2340, 2001.
[179] A. Rabiei and A. . Evans, “Failure mechanisms associated with the thermally grown oxide in plasma-sprayed thermal barrier coatings,” *Acta Mater.*, vol. 48, no. 15, pp. 3963–3976, 2000.
[180] D. COJOCARU and A. KARLSSON, “A simple numerical method of cycle jumps for cyclically loaded structures,” *Int. J. Fatigue*, vol. 28, no. 12, pp. 1677–1689, Dec. 2006.
[181] L. E. Samuels, E. D. Doyle, and D. M. Turley, “Sliding wear mechanisms,” in *Fundamental of Friction and Wear of Materials*, 1980, pp. 13–41.
[182] D. Aldham, J. Warburton, and R. E. Pendlebury, “The unlubricated fretting wear of mild steel in air,” *Wear*, vol. 106, no. 1–3, pp. 177–201, Nov. 1985.
[183] B. K. Cook and R. P. Jensen, “Discrete element methods: numerical modeling of discontinua,” in *Third International Conference on Discrete Element Methods*.
[184] A. V. POTAPOV and C. S. CAMPBELL, “A HYBRID FINITE-ELEMENT SIMULATION OF SOLID FRACTURE,” *Int. J. Mod. Phys. C*, vol. 07, no. 02, pp. 155–180, Apr. 1996.
[185] A. V. POTAPOV, M. A. HOPKINS, C. S. CAMPBELL, and M. A. HOPKINS, “A TWO-DIMENSIONAL DYNAMIC SIMULATION OF SOLID FRACTURE PART II: EXAMPLES,” *Int. J. Mod. Phys. C*, vol. 06, no. 03, pp. 371–

398, Jun. 1995.
[186] J. Paavilainen, J. Tuhkuri, and A. Polojärvi, “2D combined finite-discrete element method to model multi-fracture of beam structures,” *Eng. Comput.*, vol. 26, no. 6, pp. 578–598, Aug. 2009.
[187] R. Guises, J. Xiang, J.-P. Latham, and A. Munjiza, “Granular packing: numerical simulation and the characterisation of the effect of particle shape,” *Granul. Matter*, vol. 11, no. 5, pp. 281–292, Oct. 2009.
[188] N. Diomidis and S. Mischler, “Third body effects on friction and wear during fretting of steel contacts,” *Tribol. Int.*, vol. 44, no. 11, pp. 1452–1460, Oct. 2011.
[189] O. H. Christine, “Simuleon FEA Blog Using Abaqus DEM to model sand in a drop test,” *SIMULIA*, 2016. [Online]. Available: http://info.simuleon.com/blog/using-abaqus-dem-to-model-sand-in-a-drop-test. [Accessed: 10-Apr-2017].
[190] Z. Peng and T. B. Kirk, “Wear particle classification in a fuzzy grey system,” *Wear*, vol. 225–229, pp. 1238–1247, Apr. 1999.
[191] Z. Peng and T. B. Kirk, “Computer image analysis of wear particles in three-dimensions for machine condition monitoring,” *Wear*, vol. 223, no. 1–2, pp. 157–166, Dec. 1998.
[192] K. L. Johnson, K. Kendall, and A. D. Roberts, “Surface Energy and the Contact of Elastic Solids,” *Proc. R. Soc. A Math. Phys. Eng. Sci.*, vol. 324, no. 1558, pp. 301–313, Sep. 1971.
[193] M. C. Marinack, R. E. Musgrave, and C. F. Higgs, “Experimental Investigations on the Coefficient of Restitution of Single Particles,” *Tribol. Trans.*, vol. 56, no. 4, pp. 572–580, Jul. 2013.
[194] C. A. Felippa, *Introduction to Finite Element Methods*. 2003.
[195] G.-Z. Xu, J.-J. Liu, and Z.-R. Zhou, “The Effect of the Third Body on the Fretting Wear Behavior of Coatings.”
[196] K. L. Johnson, *Contact Mechanics*. Cambridge: Cambridge University Press, 1985.
[197] Steel Express, “EN24T Steel Properties.” [Online]. Available: https://www.steelexpress.co.uk/engineeringsteel/EN24T-properties.html. [Accessed: 15-Jun-2016].
[198] S. Yamamoto, M. Egashira, K. Kondoh, and C. Masuda, “Quantification of wear, strain and heat energy consumption rates for sliding steel ball against diamond-like carbon coatings,” *Tribol. - Mater. Surfaces Interfaces*, vol. 9, no. 2, pp. 71–76, Jun. 2015.
[199] C. Paulin, S. Fouvry, and C. Meunier, “Finite element modelling of fretting wear surface evolution: Application to a Ti–6A1–4V contact,” *Wear*, vol. 264, no. 1–2, pp. 26–36, Jan. 2008.
[200] E. Rabinowicz, *Friction and wear of materials*. Wiley, 1995.

[201] S. M. O'Halloran, P. H. Shipway, A. D. Connaire, S. B. Leen, and A. M. Harte, "A combined wear-fatigue design methodology for fretting in the pressure armour layer of flexible marine risers," *Tribol. Int.*, vol. 108, pp. 7–15, Apr. 2017.
[202] J. Ding, S. B. Leen, E. J. Williams, and P. H. Shipway, "Finite element simulation of fretting wear- fatigue interaction in spline couplings," *Tribology*, vol. 2, no. 1, 2008.
[203] T. Zhang, "Fretting wear-fatigue study for tribologically-induced damage in simple and complex geometries," Feb. 2013.
[204] M. A. Miner, "Cumulative Damage in Fatigue," *J. Appl. Mech.*, vol. 67, 1945.
[205] Z.-Y. Yu, S.-P. Zhu, Q. Liu, and Y. Liu, "A New Energy-Critical Plane Damage Parameter for Multiaxial Fatigue Life Prediction of Turbine Blades," *Materials (Basel).*, vol. 10, no. 5, p. 513, May 2017.
[206] Y. Liu, J. Q. Xu, and Y. Mutoh, "Modeling of Fretting Wear Based on Cumulative Plastic Strain and Saturated Shear Stress," *Key Eng. Mater.*, vol. 340–341, pp. 421–428, Jun. 2007.
[207] A. Ghosh, N. Paulson, and F. Sadeghi, "A fracture mechanics approach to simulate sub-surface initiated fretting wear," *Int. J. Solids Struct.*, vol. 58, pp. 335–352, Apr. 2015.
[208] R. L. Norton, *Machine design : an integrated approach*. .
[209] S. Naboulsi and T. Nicholas, "Limitations of the Coulomb friction assumption in fretting fatigue analysis," *Int. J. Solids Struct.*, vol. 40, no. 23, pp. 6497–6512, Nov. 2003.

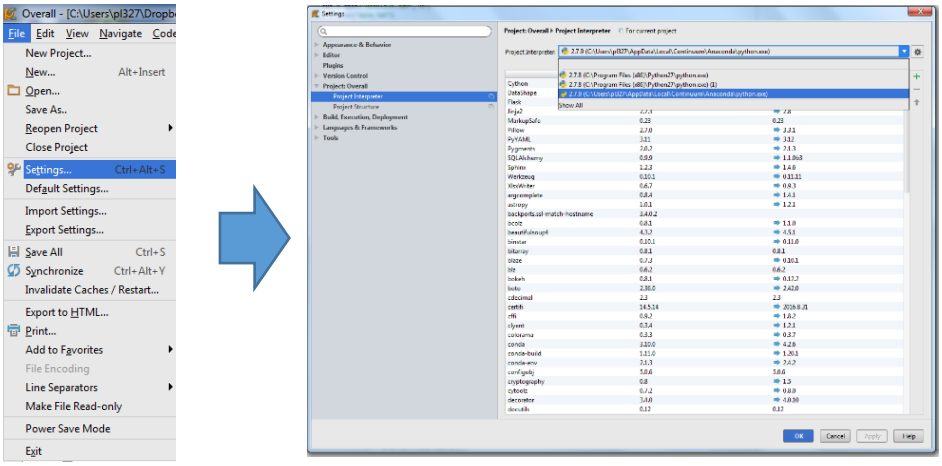
Appendix I

Manual of the FDEM model

Software Required and Installation

1. FEA solver: Abaqus 6.14 or higher (license required)
2. DEM solver: EDEM 2016 or higher (license required)
3. Python package: Anaconda
4. Python IDE: PyCharm

After installation, open PyCharm and select Anaconda as the default interpreter as shown below:



Programme/Script Files

Following are scripts used in the whole programme. For the purpose of easy reading, the codes are scripted into different files based on their functions.

The programme starts with a main script named “Overall4.py”, which controls the simulation of Abaqus, EDEM and processing the data from both two software solvers. “Overall4.py” also makes sure the information is correctly transferred between two software packages in each simulation. The other script files stores functions called from the main script.

Main script:

Overall4.py # main code

Script files of functions called:

FunCalled.py # main functions called for Abaqus
FunCalled.pyc
GetParticleLocation.py # main functions called for EDEM
GetParticleLocation.pyc
AbaRun-.py
AbaRun+.py
ResultExp.py

Input/Initial Files

Following are initial files need to be prepared before running the programme. For the purpose of easy reading, files are divided into two groups depending their functions.

Abaqus:

Default ABAQUS file: <reg.inp>
Upper and bottom geometry files: <1-GS.stl>, and <2-GP.stl>

EDEM:

Default EDEM files: <.dem>, <.dfg>, <.ess>, <.idx>, and <.ptf> file
API programme: <Test.exe>
API library file: <ParticleIntake.dll>
EDEM reference files: <reference-.txt> and <reference+.txt>

Result File:

Result Folder: stores simulation results
Target results: >ResultAnalysis.csv> saves information includes number of cycles, number of debris particles, and wear depth

<reg.inp> and EDEM default files are the main simulation files. Initial upper and bottom geometry files: <1-GS.stl> and <2-GP.stl> are to be updated regularly during simulation based on simulation results and evolution rule. <Test.exe> is used to assist EDEM. <ParticleIntable.dll> defines the path of the library about generating particles during simulation in EDEM.

Working Directories

Following are the default working directories set in the given script.

All script files and initial files are under default directory:

C:\AbaqusWorkingDir or C:\Temp

Note: which will decide the reading and exporting command latter.

Abaqus:

Please refer to: Line 11 in AbaRun-.py and AbaRun+.py.

EDEM:

Please refer to setting in EDEM default files: EDEM – Tools – Options – File Locations – Factories:

Line 44, 45, 46 in the main script (i.e. Overall4.py)

NOTE: It is highly suggested to run part of the programme before running the whole programme.

Parameters

Following is a list of parameters included in programme that can be customised based on demand.

- Dimension of geometry: e.g. shape of geometry (i.e. Height, Width, and Length), partition of geometry with fine mesh (i.e. Height, Width, Length);
- Material properties of geometry: e.g. Young's Modulus, Poisson's ratio, Material plasticity (i.e. True Stress, Plastic strain);
- Interaction properties of geometry: e.g. tangential behaviour (i.e. Coefficient of Friction), Normal Behaviour (i.e. Hard contact);
- Material properties of particle: e.g. material density, Young's Modulus, Poisson's ratio;
- Particle shapes: e.g. customized shape in STEP file, or built with surface of multi-spheres (i.e. Radius of particles, relative location of spheres' surface, as shown below);
- Particle shapes vary over time: particles sizes/shapes/properties over time;
- Particle location when generated: particles shape distribution within generated volume (as shown below);
- Interaction properties of particles: e.g. adhesion (i.e. normal stiffness per area of bond, shear stiffness per area of bond, radius of bond; or linear spring bond, i.e. energy density of bond)
- Mesh density/distribution of geometry: e.g. mesh size (i.e. Height, Width, and Length), mesh gradient (i.e. Size gradient along edge);
- Analysis domain in DEM: grid density (i.e. mesh size related to particle's radius);
- Load/force: e.g. force value (i.e. loading force or pressure)
- Displacement/amplitude of motion: e.g. displacement value;
- Boundary conditions;
- Number of processors used in simulation: default 8 processors in Abaqus and 2 processor in EDEM;

Methods of Setting a Simulation Job

There are two methods of creating a simulation job: **Method 1**: Create a completely new job from GUI; **Method 2:** Modify the parameters in the existing default files. Method 1 enables users to have much more freedom in customising a job compared to Method 2. While, Method 2 is able to provide a quicker solution. Noted it is highly suggested to test the programme step by step in Method 1 in order to ensure the files' name and directories are correct.

Method 1 – Create a new model:

ABAQUS:

!! Be careful about the names of simulation work and exported files, which should be the same as those in the script file. Either changing the names in your model or changing the names in the all script files works.

Generating <.inp> from Abaqus/CAE: *geometry, mesh, material, interaction properties, and motion*

Generating <.stl> from Abaqus/CAE by: *Module: Part – Plug-ins – Tools – STL Export – Name with type of ASCII*

Change the Name of simulation results file in AbaRun-.py and AbaRun+.py in Line 10, 16, 19, 25, 28, and 35

Make sure the part-name in in AbaRun-.py and AbaRun+.py is the same in Line 20 as in <.inp> file

Change vector value in AbaRun-.py and AbaRun+.py in Line 20 is based on amplitude of motion.

EDEM:

Generating default EDEM files through EDEM GUI

Make sure the gravity is set in right direction: *Creator – Globals – Gravity*

Set the material properties: *Creator – Globals – Materials*

Set particle properties: *Creator – Particles – Name* and *Surfaces (complex shapes can be created by multi-balls)*

Make sure the name of particle is the same as in the script in FunCalled.py: Read_Generate_STL_INP_TXT1 and Read_Generate_STL_INP_TXT2

Set particle-particle and particle interaction property: *Creator – Globals – Physics*

Import the particle factory library (i.e. <dll> file): *Creator – Factories – Select Factory – Transfer*

Name the result file "Export2.csv" by: *Analyst – File – Results Data -- Filename*

Make sure the reference-.txt and reference+.txt file are modified based on the amplitude set.

Make sure the properties are selected in the "to be exported result file":

Q01: Geometry -- Section 1 (1) ID
Q02: Geometry -- Section 1 (1) Node 1 X
Q03: Geometry -- Section 1 (1) Node 1 Y
Q04: Geometry -- Section 1 (1) Node 1 Z
Q05: Geometry -- Section 1 (1) Node 2 X
Q06: Geometry -- Section 1 (1) Node 2 Y
Q07: Geometry -- Section 1 (1) Node 2 Z
Q08: Geometry -- Section 1 (1) Node 3 X
Q09: Geometry -- Section 1 (1) Node 3 Y
Q10: Geometry -- Section 1 (1) Node 3 Z
Q11: Geometry -- Section 1 (1) Total Force X
Q12: Geometry -- Section 1 (1) Total Force Y
Q13: Geometry -- Section 1 (1) Total Force Z
Q14: Particle – Type rock_particle Position X
Q15: Particle – Type rock_particle Position Y
Q16: Particle – Type rock_particle Position Z
Q17: Particle – Type rock_particle Orientation XX
Q18: Particle – Type rock_particle Orientation XY
Q19: Particle – Type rock_particle Orientation XZ
Q20: Particle – Type rock_particle Orientation YX
Q21: Particle – Type rock_particle Orientation YY
Q22: Particle – Type rock_particle Orientation YZ
Q23: Particle – Type rock_particle Orientation ZX
Q24: Particle – Type rock_particle Orientation ZY

Q25: Particle – Type rock_particle Orientation ZZ

Method 2 – Modify the existing model:

ABAQUS:

Change material properties in the default files:

Material Properties: from Line 43880 in <.inp> file

Friction and Slip tolerance: from Line 43889 in <.inp> file

Load value: in Line 43989 in <.inp> file

Amplitude value: in Line 44027 in <.inp> file

EDEM:

Material properties in the default files

Particles shapes in the default files

Interaction properties in the default files

Amplitude value in the <reference+.txt> and <reference-.txt>

Particle shape describe in <preference_file.txt>

Output/Results Files

Following are the results will be saved during programme. Results obtained in one simulation cycle from FEA and DEM will be saved together.

Total:

1. ResultAnalysis.csv: number of cycles vs. number of particles vs. wear depth
2. Results files from Abaqus;
3. Results files from EDEM;

Abaqus result files include:

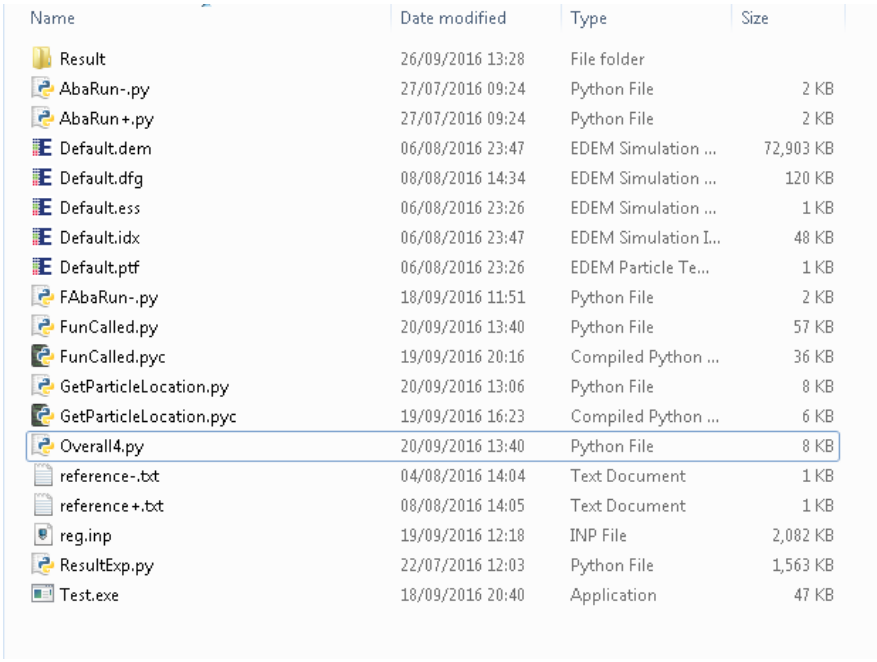
1. 1.stl: the new Bottom geometry file, which will be used for next simulation
2. 2.stl: the new Upper geometry file, which will be used for next simulation
3. 12.stl: the geometry removed, which is used to generating debris particles
4. 1-GS.stl: the Upper geometry simulated in this simulation
5. 2-GP.stl: the Bottom geometry simulated in this simulation
6. reg.inp: the <.inp> file used in this simulation
7. regNew.inp: the new updated <.inp> file, which will be used in the next simulation
8. reference_file.txt: including the location of debris particles, which will then be used in EDEM
9. reg+wear_depth.odb: result file of Abaqus
10. abaqus.rpt: result file of selected parameters
11. preference_file: information of the newly generated particles

EDEM:

1. 2.stl: Upper geometry file used in this simulation
2. EDEM files: <.dem>, <.dfg>, <.ess>, <.idx>, and <.ptf> file
3. Export2.csv: results file of selected parameters
4. GetParticleLocation.txt: the updated location of particles includes those left from previous and generated in current simulation.

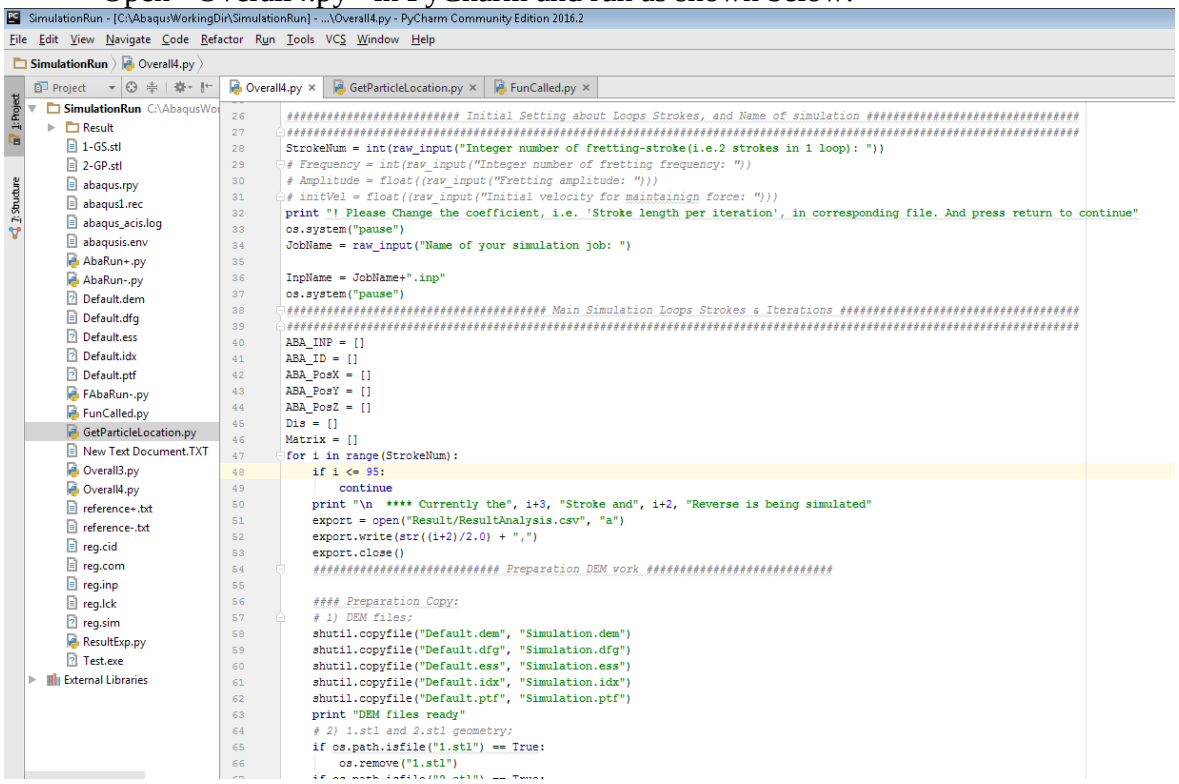
Example of How to Run the Programme

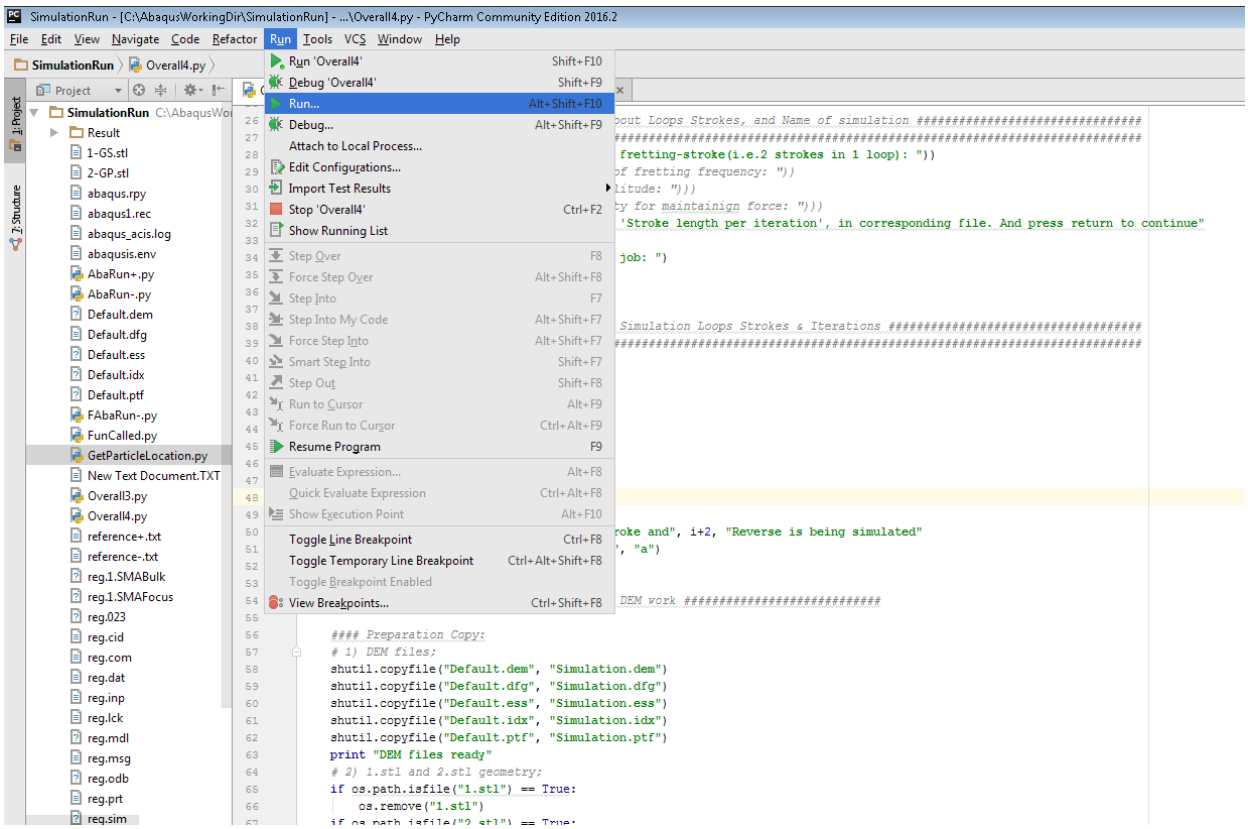
- Default setting as shown below:



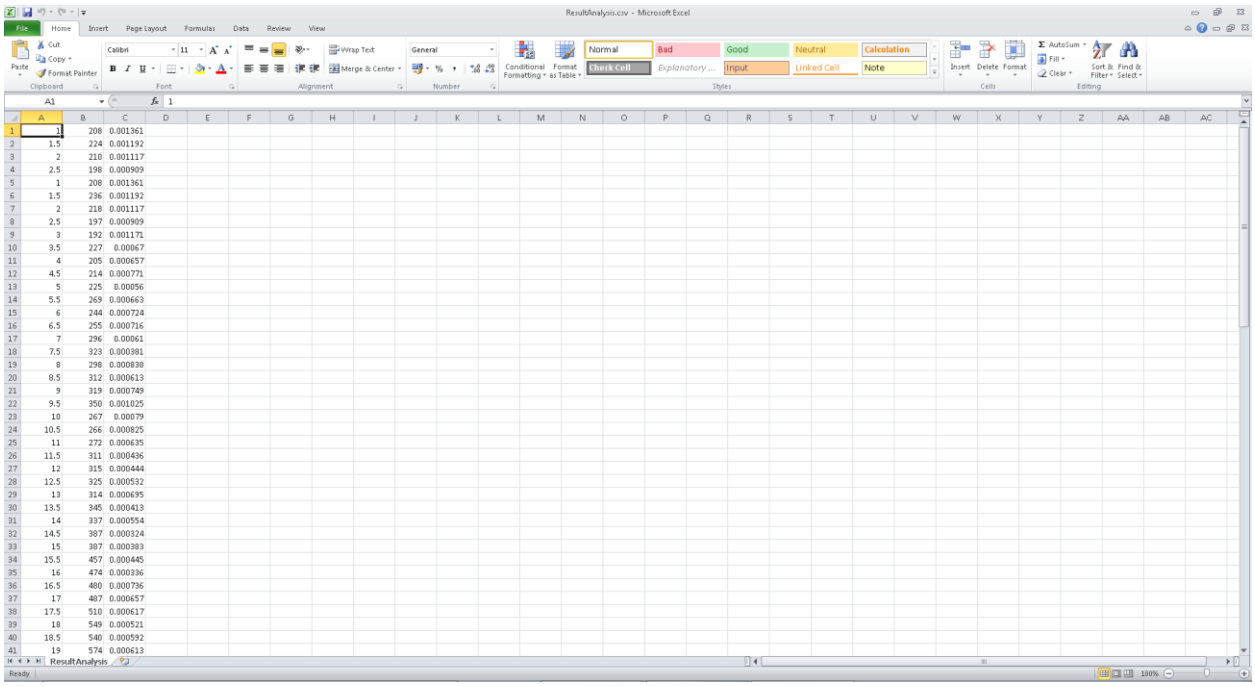
Name	Date modified	Type	Size
Result	26/09/2016 13:28	File folder	
AbaRun-.py	27/07/2016 09:24	Python File	2 KB
AbaRun+.py	27/07/2016 09:24	Python File	2 KB
Default.dem	06/08/2016 23:47	EDEM Simulation ...	72,903 KB
Default.dfg	08/08/2016 14:34	EDEM Simulation ...	120 KB
Default.ess	06/08/2016 23:26	EDEM Simulation ...	1 KB
Default.idx	06/08/2016 23:47	EDEM Simulation I...	48 KB
Default.ptf	06/08/2016 23:26	EDEM Particle Te...	1 KB
FAbaRun-.py	18/09/2016 11:51	Python File	2 KB
FunCalled.py	20/09/2016 13:40	Python File	57 KB
FunCalled.pyc	19/09/2016 20:16	Compiled Python ...	36 KB
GetParticleLocation.py	20/09/2016 13:06	Python File	8 KB
GetParticleLocation.pyc	19/09/2016 16:23	Compiled Python ...	6 KB
Overall4.py	20/09/2016 13:40	Python File	8 KB
reference-.txt	04/08/2016 14:04	Text Document	1 KB
reference+.txt	08/08/2016 14:05	Text Document	1 KB
reg.inp	19/09/2016 12:18	INP File	2,082 KB
ResultExp.py	22/07/2016 12:03	Python File	1,563 KB
Test.exe	18/09/2016 20:40	Application	47 KB

- Open <Overall4.py> in PyCharm and run as shown below:





- Input cycles you want to run;
- Input the name of your Abaqus <.inp> files: i.e. “reg” (by default);
- Open <ResultAnalysis.csv> file to check if the results are saved correctly as shown below:



The first column is cycle number; the second column saves the number of particles in the contact zone; the third column saves the wear depth

Appendix II

The script of the FDEM model

```
from itertools import islice
import subprocess
import time
from datetime import datetime
import os
import shutil
## clean the duplicate contents due to multi-Translate-To function
import math
import numpy
import random

##############################################################################
#####################################
##############################################################################
#####################################
def Re_Num(item): # function that get rig of very tiny number, e.g. XXXe-12
    if "e" in str(item):
        return 0
    else:
        ############!! important. the round number depends on the different
mesh seize.
        return round(item,2)

def main(Number):
    name = str(Number)
    print name

    Infor1 = ['Node']
    Infor2 = ['Element']
    Infor4 = ['name=Up-1']
    Infor5 = [' Instance']

    ABA_ID = [] # array stores node-ID of each node
    ABA_PosX = [] # array stores X-coordinate of each node
    ABA_PosY = [] # array stores Y-coordinate of each node
    ABA_PosZ = [] # array stores Z-coordinate of each node
    Dis = []
    Up = []

    with open (name + "th.inp") as myfile:
        Flag_Cut1 = 0
        f = 0
        for line in myfile:
            if any(s in line for s in Infor1):
                Flag_Cut1 = 1
                f = f + 1 # Signal for Part2
                continue
```

```
        if any(s in line for s in Infor2):
            Flag_Cut1 = 0
            continue
        if any(s in line for s in Infor4):
            Flag_Cut1 = 2
            continue
        if any(s in line for s in Infor5):
            Flag_Cut1 = 0
            continue
        if "CYLINDER" in line:
            Flag_Cut1 = 3
            continue
        if Flag_Cut1 == 3 and "Rigid Body" in line:
            Flag_Cut1 = 0
            continue
        if Flag_Cut1 == 1:
            if f == 1:
                ABA_ID.append(eval(line.split(",")[0]))
                ABA_PosX.append(eval(line.split(",")[1]))
                ABA_PosY.append(eval(line.split(",")[2]))
                ABA_PosZ.append(eval(line.split(",")[3]))
                continue
            else:
                continue
        if Flag_Cut1 == 2:
            Dis.append(eval(line.split(",")[0]))
            Dis.append(eval(line.split(",")[1]))
            Dis.append(eval(line.split(",")[2]))
            Flag_Cut1 = 0
            continue
        if Flag_Cut1 ==3:
            Up.append((line.split(",")[1], line.split(",")[2]))
            continue
        if Flag_Cut1 == 0:
            continue

for i in range(len(Up)):
    Up[i] = ((eval(Up[i][0]), eval(Up[i][1])))

######### Import MISES information ##########



IndexRaw = []
IndexTemp = []
Index = []
CShearMax = []
```

```
    CShearMin = []
    CShear = []
    ABA_Cri1 = [] # array stores Von-Mises value of each node in Part2 # !!!!!
save for future modification
    with open (name+"thCShear1.rpt") as myfile:
        text = list(islice(myfile, 3, None))
        IndexRaw = text[0].split()[0:None]
        for line in text:
            if "MAX" in line:
                CShearMax = line.split()[2:None]
            elif "MIN" in line:
                CShearMin = line.split()[2:None]
            else:
                continue
    for i in range(len(CShearMax)):
    ##    CShear.append(abs(abs(eval(CShearMin[i]))-abs(eval(CShearMax[i]))))
        if abs(eval(CShearMax[i])) <= abs(eval(CShearMin[i])):
            CShear.append(abs(eval(CShearMin[i])))
        else:
            CShear.append(abs(eval(CShearMax[i])))

    for i in range(len(IndexRaw)):
        if "N" in IndexRaw[i]:
            IndexTemp.append(eval(IndexRaw[i][2:None]))

    for i in range(len(IndexTemp)):
        ABA_Cri1.append((IndexTemp[i],CShear[i]))
    ABA_Cri1 = list(set(ABA_Cri1))
    ABA_Cri1 = sorted(ABA_Cri1,key=lambda k: (-k[0]), reverse=True)

    IndexRaw = []
    IndexTemp = []
    Index = []
    CSlipR =[]
    CSlipL = []
    ABA_Cri2 = []
    with open (name+"thCSlipL1.dat") as myfile:
        text = list(islice(myfile, 1, 2502))
        for line in text:
            CSlipL.append(line.split()[1])
            IndexTemp.append(eval(line.split()[0]))
    with open (name+"thCSlipR1.dat") as myfile:
        text = list(islice(myfile, 1, 2502))
        for line in text:
            CSlipR.append(line.split()[1])
    CSlip = []
    for i in range(len(CSlipR)):
```

```
        CSlip.append(abs(eval(CSlipR[i]))+abs(eval(CSlipR[i])-
eval(CSlipL[i])))
    for i in range(len(IndexTemp)):
        ABA_Cri2.append((IndexTemp[i],CSlip[i]))



    CPressMaxRaw = []
    IndexRaw = []
    Index = []
    with open("CPRESS.rpt") as myfile:
       text = list(islice(myfile,3,None))
       IndexRaw = text[0].split()[0:None]
       for line in text:
           if "0.150" in line:
               CPressMaxRaw = line.split()[1:None]
           else:
               continue
    for i in range(len(IndexRaw)):
       if "N" in IndexRaw[i]:
           Index.append(eval(IndexRaw[i][2:None]))

    CPressMax = []
    for i in range(len(Index)):
       CPressMax.append((Index[i],eval(CPressMaxRaw[i])))
    CPressMax = list(set(CPressMax))
    CPressMax = sorted(CPressMax,key=lambda k: (-k[0]), reverse=True)

    PressureCalculation = []
    cc = 0
    for i in range(len(ABA_ID)):
       if cc != len(CPressMax) and ABA_ID[i] == CPressMax[cc][0]:

PressureCalculation.append((ABA_ID[i],ABA_PosX[i],ABA_PosY[i],ABA_PosZ[i],CPre
ssMax[cc][1]))
           cc = cc +1



    ########## 1st re-arrange and 3D-Matrix of input file Generating
############
    print "3D-Matrix-Inp"
    # print str(datetime.now())



    ABA_INP = [] # array stores node-ID, X-coordinate, Y-coordinate, Z-
coordinate of each node
    ABA_INP0 = [] # array stores original node-ID, X-coordinate, Y-coordinate,
```

```
Z-coordinate of each node
    cc = 0
    for i in range(len(ABA_ID)):
        if cc != len(ABA_Cri1) and ABA_ID[i] == ABA_Cri1[cc][0]:

ABA_INP.append((ABA_ID[i],round(ABA_PosX[i],2),round(ABA_PosY[i],2),round(ABA_
PosZ[i],2),ABA_Cri2[cc][1]*ABA_Cri1[cc][1]))

ABA_INP0.append((ABA_ID[i],ABA_PosX[i],ABA_PosY[i],ABA_PosZ[i],ABA_Cri2[cc][1]
*ABA_Cri1[cc][1]))
            cc = cc+1
        else:

ABA_INP.append((ABA_ID[i],round(ABA_PosX[i],2),round(ABA_PosY[i],2),round(ABA_
PosZ[i],2),0))
            ABA_INP0.append((ABA_ID[i],ABA_PosX[i],ABA_PosY[i],ABA_PosZ[i],0))



    # !!!!!!!!!!!!!!!!!!!!!!!!!!!! #  * important * the node will be re-
arranged by order: x: from small to bigger; y: from small to bigger; z: from
big to smaller
    # !!!!!!!!!!!!!!!!!!!!!!!!!!!! #  * important * therefore, the node on
the most LHS and front will be the first and the most RHS and backward will be
the last
    ABA_INP = sorted(ABA_INP,key=lambda k: (-k[1], k[3], k[2]), reverse=True)
#Sort list of tuples by first Ascending and second element Descending



    Matrix = [(1,1,1)] # tuples stores 3D-Matrix of each node, which present
its corresponding location in the module's shaped space
    x = y = z = 1 # initial numbers; as long as there is an extra node along x
or y or z axis, one will be added to corresponding x, y, or z
    for i in range(len(ABA_INP)):
        if i == 0:
            continue # as there is already (1,1,1) in the Matrix
        else:
            if ABA_INP[i][1] == ABA_INP[i-1][1] and ABA_INP[i][2] !=
ABA_INP[i-1][2] and ABA_INP[i][3] == ABA_INP[i-1][3]:
                y = y + 1
                Matrix.append((x,y,z))
            else:
                if ABA_INP[i][1] == ABA_INP[i-1][1] and ABA_INP[i][2] !=
ABA_INP[i-1][2] and ABA_INP[i][3] != ABA_INP[i-1][3]:
                    y = 1
                    z = z + 1
                    Matrix.append((x,y,z))
```

```
                else:
                    y = z = 1
                    x = x + 1
                    Matrix.append((x,y,z))
    lastX = x
    lastZ = z
    last = y #### !!! Important !!!! this will be used in the mesh morphing


    ######### Geometry Import ############
    print "<.stl> Geometry input"
    # print str(datetime.now())

    Tri_ID = [] # array stores Tri-element-ID, each Tri-element has three
nodes and each node has three vertex or x,y,z coordinates
    Tri_Vertex1X = [] # array stores the x-coordinate of first node
    Tri_Vertex1Y = [] # array stores the y-coordinate of first node
    Tri_Vertex1Z = [] # array stores the z-coordinate of first node
    Tri_Vertex2X = [] # array stores the x-coordinate of second node
    Tri_Vertex2Y = [] # array stores the y-coordinate of second node
    Tri_Vertex2Z = [] # array stores the z-coordinate of second node
    Tri_Vertex3X = [] # array stores the x-coordinate of third node
    Tri_Vertex3Y = [] # array stores the y-coordinate of third node
    Tri_Vertex3Z = [] # array stores the z-coordinate of third node

    with open ("F2.stl") as myfile:
        myfile.next()
        Flag_Cut3 = 0
        Cont = 0
        for line in myfile:
            Flag_Cut3 = Flag_Cut3 + 1
            if Flag_Cut3 == 1 or Flag_Cut3 == 2 or Flag_Cut3 == 6:
                continue
            if Flag_Cut3 == 3:
                Cont = Cont + 1
                Tri_ID.append(Cont)
                Tri_Vertex1X.append(eval(line.split()[1]))
                Tri_Vertex1Y.append(eval(line.split()[2]))
                Tri_Vertex1Z.append(eval(line.split()[3]))
                continue
            if Flag_Cut3 == 4:
                Tri_Vertex2X.append(eval(line.split()[1]))
                Tri_Vertex2Y.append(eval(line.split()[2]))
                Tri_Vertex2Z.append(eval(line.split()[3]))
                continue
            if Flag_Cut3 == 5:
                Tri_Vertex3X.append(eval(line.split()[1]))
```

```
                Tri_Vertex3Y.append(eval(line.split()[2]))
                Tri_Vertex3Z.append(eval(line.split()[3]))
                continue
            if Flag_Cut3 == 7:
                Flag_Cut3 = 0


    ######## 3D-Matrix for Stl generation #########
    print "3D-Matrix-Stl"
    # print str(datetime.now())


    Tri_STL = []
    Tri_STL2 = []
    for i in range(len(Tri_ID)):

Tri_STL.append((Tri_ID[i],Re_Num(Tri_Vertex1X[i]),Re_Num(Tri_Vertex1Y[i]),Re_N
um(Tri_Vertex1Z[i])))

Tri_STL.append((Tri_ID[i],Re_Num(Tri_Vertex2X[i]),Re_Num(Tri_Vertex2Y[i]),Re_N
um(Tri_Vertex2Z[i])))

Tri_STL.append((Tri_ID[i],Re_Num(Tri_Vertex3X[i]),Re_Num(Tri_Vertex3Y[i]),Re_N
um(Tri_Vertex3Z[i])))

Tri_STL2.append((Tri_ID[i],Tri_Vertex1X[i],Tri_Vertex1Y[i],Tri_Vertex1Z[i]))

Tri_STL2.append((Tri_ID[i],Tri_Vertex2X[i],Tri_Vertex2Y[i],Tri_Vertex2Z[i]))

Tri_STL2.append((Tri_ID[i],Tri_Vertex3X[i],Tri_Vertex3Y[i],Tri_Vertex3Z[i]))

    Tri_STL=sorted(Tri_STL,key=lambda k: (-k[1], k[3], k[2]),
reverse=True)#Sort list of tuples by first Ascending and second element
Descending



    Tri_Matrix = [(1,1,1)]
    x = y = z = 1
    for i in range(len(Tri_STL)):
        if i != 0:# as there is already (1,1,1) in the Matrix
            if Tri_STL[i][1] == Tri_STL[i-1][1] and Tri_STL[i][2] ==
Tri_STL[i-1][2] and Tri_STL[i][3] == Tri_STL[i-1][3]:
                Tri_Matrix.append(Tri_Matrix[-1])
            else:
                if Tri_STL[i][1] == Tri_STL[i-1][1] and Tri_STL[i][2] !=
Tri_STL[i-1][2] and Tri_STL[i][3] == Tri_STL[i-1][3]:
```

```
                    y = y + 1
                    Tri_Matrix.append((x,y,z))
                else:
                    if Tri_STL[i][1] == Tri_STL[i-1][1] and Tri_STL[i][2] !=
Tri_STL[i-1][2] and Tri_STL[i][3] != Tri_STL[i-1][3]:
                        y = 1
                        z = z + 1
                        Tri_Matrix.append((x,y,z))
                    else:
                        y = z = 1
                        x = x + 1
                        Tri_Matrix.append((x,y,z))
    CancelLoca = []
    print "Match-Von Mises and Updating"
    # print str(datetime.now())

    import sys

    # the following if statement just check if the first element of both 3D-
Matrix matched each other. theoretically, they shall have the same first
element.
    if Tri_STL[0][1] == ABA_INP[0][1] and Tri_STL[0][2] == ABA_INP[0][2] and
Tri_STL[0][3] == ABA_INP[0][3]:
        print "checked!"
    else: # otherwise, there must be some error in the previous steps. if so,
then this script will be ended.
        sys.exit()


    Tri_Part2 = [] # storing the Tri-element-ID of those Tri-element that was
updated here. the updated Tri-element will and written with the original Tri-
element to forming the Part2/cutted part
    x=0
    temp2 = temp4 = 0
    coefficient = 0.0
    ABA_INP3 = [] # store the morphed mesh nodeID
    ABA_INP4 = [] # store the mesh structure where debris particles are
generated
    Depth = [0] #store all the wear depth for each node
    Location = [(1,1,1)]
    for i in range(len(Tri_Matrix)):
        if i != 0 and Tri_Matrix[i][1] == 1: # the first node not need to be
morphed; only surfaces node, i.e. y =1, need morphing
            if Tri_Matrix[i] == Tri_Matrix[i-1]:
                Tri_STL[i] = (Tri_STL[i][0],Tri_STL[i-1][1],Tri_STL[i-
1][2],Tri_STL[i-1][3])
                ############ store the cut part
```

```
                if temp2 == 1:
                    Tri_Part2.append(Tri_STL[i][0])
                else:
                    continue
            else:
                x = Matrix.index(Tri_Matrix[i])
                bottomIndex = Matrix.index((Matrix[x][0],last,Matrix[x][2]))
                Depth.append(ABA_INP[x][-1]*0.00000164*2)
                Location.append(Tri_Matrix[i])
                coefficient = Depth[-1]/(ABA_INP0[ABA_INP[x][0]-1][2]-
ABA_INP0[ABA_INP[bottomIndex][0]-1][2])
                if coefficient != 0:
                    ABA_INP4.append((ABA_INP0[ABA_INP[x][0]-1][1],
ABA_INP0[ABA_INP[x][0]-1][2], ABA_INP0[ABA_INP[x][0]-1][3]))
                    ABA_INP4.append((ABA_INP0[ABA_INP[x][0]-1][1],
ABA_INP0[ABA_INP[x][0]-1][2]-Depth[-1], ABA_INP0[ABA_INP[x][0]-1][3]))
                    for element in range(last):
                        index =
Matrix.index((Matrix[x][0],element+1,Matrix[x][2]))
                        ABA_INP3.append((ABA_INP0[ABA_INP[index][0]-
1][0],ABA_INP0[ABA_INP[index][0]-1][1],(1-
coefficient)*ABA_INP0[ABA_INP[index][0]-
1][2]+coefficient*ABA_INP0[ABA_INP[bottomIndex][0]-1][2],
ABA_INP0[ABA_INP[index][0]-1][3]))


    record = sum(Depth)/sum(1 if element != 0 else 0 for element in Depth)

    k=1
    for i in range(len(Tri_Matrix)):
        if i != 0 and Tri_Matrix[i][1] == 1 and Tri_Matrix[i] not in
CancelLoca:
            if Tri_Matrix[i] == Tri_Matrix[i-1]:
                Tri_STL[i] = (Tri_STL[i][0],Tri_STL[i-1][1],Tri_STL[i-
1][2],Tri_STL[i-1][3])
            else:
                x = Matrix.index(Tri_Matrix[i])
                Tri_STL[i] =
(Tri_STL[i][0],Tri_STL[i][1],ABA_INP0[ABA_INP[x][0]-1][2]-
Depth[k],Tri_STL[i][3])
                k = k+1


    ########## 2nd back-sorted and New Stl generation ############
    print "Re-sort and New-Stl"
    # print str(datetime.now())
```

```
    Tri_STL = sorted(Tri_STL,key=lambda k: k[0]) # re-arranged by having Tri-
element, therefore they can be grouped and written into a STL format file of
Part2

    out = open(name+"thF1.stl","w")
    out.write("solid" + "\n")
    x = 1
    for i in range(len(Tri_STL)):
        if Tri_STL[i][0] in Tri_Part2:
            ABA_INP4.append((Tri_STL[i][1],Tri_STL[i][2],Tri_STL[i][3]))
            if x == 1:
                out.write("facet normal 0.0 0.0 0.0" + "\n" + "outer loop" +
"\n")
            out.write("vertex " + str(Tri_STL[i][1]) + " " +
str(Tri_STL[i][2]) + " " + str(Tri_STL[i][3]) + "\n")
            if x == 3:
                out.write("endloop" + "\n" + "endfacet" + "\n")
                x = 1
                continue
            x = x + 1
        else:
            if x == 1:
                out.write("facet normal 0.0 0.0 0.0" + "\n" + "outer loop" +
"\n")
            out.write("vertex " + str(Tri_STL[i][1]) + " " +
str(Tri_STL[i][2]) + " " + str(Tri_STL[i][3]) + "\n")
            if x == 3:
                out.write("endloop" + "\n" + "endfacet" + "\n")
                x = 1
                continue
            x = x + 1
    out.write("endsolid" + "\n")
    out.close()


    ########## Particle Generation ############
    print "Particle Pos Generation"
    Ini_ABA_ID = [] # array stores node-ID of each node
    Ini_ABA_PosX = [] # array stores X-coordinate of each node
    Ini_ABA_PosY = [] # array stores Y-coordinate of each node
    Ini_ABA_PosZ = [] # array stores Z-coordinate of each node
    Ini_Dis = []
    Ini_Up = []

    with open ("1th.inp") as myfile:
        Flag_Cut1 = 0
```

```
        f = 0
        for line in myfile:
            if any(s in line for s in Infor1):
                Flag_Cut1 = 1
                f = f + 1 # Signal for Part2
                continue
            if any(s in line for s in Infor2):
                Flag_Cut1 = 0
                continue
            if any(s in line for s in Infor4):
                Flag_Cut1 = 2
                continue
            if any(s in line for s in Infor5):
                Flag_Cut1 = 0
                continue
            if "CYLINDER" in line:
                Flag_Cut1 = 3
                continue
            if Flag_Cut1 == 3 and "Rigid Body" in line:
                Flag_Cut1 = 0
                continue
            if Flag_Cut1 == 1:
                if f == 1:
                    Ini_ABA_ID.append(eval(line.split(",")[0]))
                    Ini_ABA_PosX.append(eval(line.split(",")[1]))
                    Ini_ABA_PosY.append(eval(line.split(",")[2]))
                    Ini_ABA_PosZ.append(eval(line.split(",")[3]))
                    continue
                else:
                    continue
            if Flag_Cut1 == 2:
                Ini_Dis.append(eval(line.split(",")[0]))
                Ini_Dis.append(eval(line.split(",")[1]))
                Ini_Dis.append(eval(line.split(",")[2]))
                Flag_Cut1 = 0
                continue
            if Flag_Cut1 ==3:
                Ini_Up.append((line.split(",")[1], line.split(",")[2]))
                continue
            if Flag_Cut1 == 0:
                continue
    Ini_ABA_INP = [] # array stores node-ID, X-coordinate, Y-coordinate, Z-
coordinate of each node
    cc = 0
    for i in range(len(ABA_ID)):

Ini_ABA_INP.append((Ini_ABA_ID[i],Ini_ABA_PosX[i],Ini_ABA_PosY[i],Ini_ABA_PosZ
```

```
[i]))


    Distance = []
    for i in range(len(Matrix)):
        if Matrix[i][1] == 1:
            Distance.append(Ini_ABA_INP[ABA_INP[i][0]-1][2] -
ABA_INP0[ABA_INP[i][0]-1][2])

    for i in range(len(Distance)):
        Depth[i] = Depth[i] + Distance[i]
    record = sum(Depth)/sum(1 if element != 0 else 0 for element in Depth)
    print "Till Zero:", record

    Xvalue = []
    Yvalue = []
    Zvalue = []
    #print str(datetime.now())
    if len(ABA_INP4) != 0:
        list1, list2, list3 = zip(*ABA_INP4)
    else:
        0
    Xmax = sorted(list(set(list1)))[-2]
    Xmin = sorted(list(set(list1)))[1]


    out = open("preference_file.txt","w")
    out.write('#Unit("m"_or_"mm")' + "\nmm\n\n")
    out.write("#Number_of_loading" + "\n1\n\n")
    out.write("#Time_of_loading_&_Center_of_mass_(in_mm)"+ "\n0.00001 0 0
0\n\n")
    out.write("#Number_of_particle_type"+"\n1\n")
    out.write("#Particle_type_and_mass\n" + "rock_particle 1.0472e-15\n\n")
    out.write("#particle(X,Y,Z)position(in_mm)_mass(kg)_and_Orientation\n")
    for i in range(int(0.25*(0.003*0.003*sum(1 if element != 0 else 0 for
element in Depth)*record/3)*0.9/2.68083e-10)):
        sample = numpy.random.uniform(Xmin, Xmax)
        # while sample < Xmin or sample > Xmax:
        #     sample = numpy.random.normal((Xmin+Xmax)/2.0, 0.5)
        Xvalue.append(sample)
        X=min(sorted(list1), key=lambda x:abs(x-Xvalue[-1]))

        Zlist0 = [item for item in ABA_INP4 if (item[0] == X)]
        Zlist = set(list(zip(*Zlist0)[2]))
        Z1=sorted(Zlist)[1]
        Z2=sorted(Zlist)[-2]
        sample = numpy.random.uniform(Z1, Z2)
```

```
        # while sample < (Z1+0.003) or sample > (Z2-0.003):
        #     sample = numpy.random.normal((Z1+Z2)/2.0,0.5)
        Zvalue.append(sample)

        Z=min(sorted(Zlist), key=lambda x:abs(x-Zvalue[-1]))
        Ylist0 = [item for item in Zlist0 if (item[2] == Z)]
        Ylist = set(list(zip(*Ylist0)[1]))
        Y1=sorted(Ylist)[0]
        Y2=sorted(Ylist)[-1]
        # if Y1 == Y2:
        #     Yvalue.append(Y1)
        # else:
        sample = numpy.random.uniform(Y1+0.0005, Y2+0.0005)
            # while sample < (Y1) or sample > (Y2):
            #     sample = numpy.random.normal((Y1+Y2)/2.0,0.5)
        Yvalue.append(sample+0.01)
        # print "X Y Z"
        # print Xvalue[-1]
        # print Yvalue[-1]
        # print Zvalue[-1]
        out.write("rock_particle " + str(Xvalue[-1]) + " " + str(Yvalue[-1]) +
" " + str(Zvalue[-1]) + " 5.0472e-15 ")

        Xangle = random.uniform(0,math.pi)
        Yangle = random.uniform(0,math.pi)
        Zangle = random.uniform(0,math.pi)

        out.write(str(round(math.cos(Yangle)*math.cos(Zangle),6))+" "+
str(round(math.cos(Yangle)*math.sin(Zangle),6))+" "+str(round(-
math.sin(Yangle),6))+" ")
            ## particle orientation in XX, XY, XZ
        out.write(str(round(-
math.cos(Xangle)*math.sin(Zangle)+math.sin(Xangle)*math.sin(Yangle)*math.cos(Z
angle),6))+" ")

out.write(str(round(math.cos(Xangle)*math.cos(Zangle)+math.sin(Xangle)*math.si
n(Yangle)*math.sin(Zangle),6))+" ")
        out.write(str(round(math.sin(Xangle)*math.cos(Yangle),6))+" ")
            ## particle orientation in YX, YY, YZ

out.write(str(round(math.sin(Xangle)*math.sin(Zangle)+math.cos(Xangle)*math.si
n(Yangle)*math.cos(Zangle),6))+" ")
        out.write(str(round(-
math.sin(Xangle)*math.cos(Zangle)+math.cos(Xangle)*math.sin(Yangle)*math.sin(Z
angle),6))+" ")
        out.write(str(round(math.cos(Xangle)*math.cos(Yangle),6))+"\n")
    out.close()
```

```
    print "CPress calculation"
    CpressMaxMin = [item[1] for item in PressureCalculation if item[-1]!=0]
    CPMax = max(CpressMaxMin)
    CPMin = min(CpressMaxMin)

    IndexCpress = []
    XCpress = []
    YCpress = []
    ZCpress = []
    ValueCpress = []
    IndexCpress, XCpress, YCpress, ZCpress, ValueCpress =
zip(*PressureCalculation)
    TotalForce = 0
    for i in range(len(Xvalue)):
        X = min(sorted(XCpress), key=lambda x: abs(x - Xvalue[i]))
        if X >= CPMin and X <= CPMax:
            Z = min(sorted(ZCpress), key=lambda x: abs(x - Zvalue[i]))
            match =  [x for x in PressureCalculation if x[1] == X and x[3] ==
Z]
            TotalForce = TotalForce + match[0][4]*3.14*(8E-4)*(8E-4)
    out = open("ForceApply.txt","w")
    out.write("#Geometry Name & mass(kg) & Initial Velocity(m/s)\n")
    out.write("0.0002  1\n")
    out.write("#Force(N)\n")
    out.write(str(TotalForce))
    out.close()
    return TotalForce


def FileChange(filename, number):
    out = open("temp.py","w")
    with open(filename) as myfile:
        for line in myfile:
            if "target = " in line:
                out.write("target = '" + str(number) + "'\n")
            else:
                out.write(line)
    out.close()
    os.remove(filename)
    os.rename("temp.py",filename)

def ReadLocation():
    Location = []
    start = 0
    PositionX = []
```

```
    PositionY = []
    PositionZ = []
    Mass = []
    Orientation = []
    ID = []
    with open("ParticleLocation.csv") as myfile:
        for line in myfile:
            if "EXTRACTED DATA" in line:
                start = 1
                continue
            if start == 1:
                if "Q01" in line:
                    PositionX = line.split(",")[1:None]
                    PositionX = [float(i) for i in PositionX]
                if "Q02" in line:
                    PositionY = line.split(",")[1:None]
                    PositionY = [float(i) for i in PositionY]
                if "Q03" in line:
                    PositionZ = line.split(",")[1:None]
                    PositionZ = [eval(i) for i in PositionZ]
                if "Q04" in line:
                    Mass = line.split(",")[1:None]
                    Mass = [float(i) for i in Mass]
                if "Q05" in line:
                    Orientation = line.split(",")[1:None]
                if "Q06" in line:
                    ID = line.split(",")[1:None]
                    ID = [int(i) for i in ID]
    for i in range(len(ID)):

Location.append((ID[i],PositionX[i],PositionY[i],PositionZ[i],Mass[i]))
    Location = sorted(Location,key=lambda k: (-k[0]), reverse=True)

    out = open("ParticleLoc.txt", "w")
    out.write('#Unit("m"_or_"mm")\nmm\n\n')
    out.write("#Number_of_loading\n1\n\n")
    out.write("#Time_of_loading_&_Center_of_mass_(in_mm)\n0.00001 0 0 0\n\n")

out.write("#Number_of_particle_type\n1\n#Particle_type_and_mass\nrock_particle
1.0472e-15\n\n")
    out.write("#particle(X,Y,Z)position(in_mm)_mass(kg)_and_Orientation\n")
    for i in range(len(Mass)):
        out.write("rock_particle " + str(PositionX[i] * 1000) + " " +
str(PositionY[i] * 1000) + " " + str(PositionZ[i] * 1000) +
                  " " + str(3.2302e-15) + " " + Orientation[i].split("
")[1][3:None] + " " + Orientation[i].split(" ")[2][3:None] + " " +
                  Orientation[i].split(" ")[3][3:None]
```

```
                    + " " + Orientation[i].split(" ")[4][3:None] + " " +
Orientation[i].split(" ")[5][3:None] + " " +
                    Orientation[i].split(" ")[6][3:None]
                    + " " + Orientation[i].split(" ")[7][3:None] + " " +
Orientation[i].split(" ")[8][3:None] + " " +
                    Orientation[i].split(" ")[9][3:None]+ "\n")
    out.close()

    return Location

def ForceRead(Name, Force):
    Infor1 = ['Node']
    Infor2 = ['Element']
    Infor4 = ['name=Up-1']
    Infor5 = [' Instance']

    ABA_ID = []  # array stores node-ID of each node
    ABA_PosX = []  # array stores X-coordinate of each node
    ABA_PosY = []  # array stores Y-coordinate of each node
    ABA_PosZ = []  # array stores Z-coordinate of each node
    Dis = []

    with open(str(Name) +"th.inp") as myfile:
        Flag_Cut1 = 0
        f = 0
        for line in myfile:
            if any(s in line for s in Infor1):
                Flag_Cut1 = 1
                f = f + 1  # Signal for Part2
                continue
            if any(s in line for s in Infor2):
                Flag_Cut1 = 0
                continue
            if any(s in line for s in Infor4):
                Flag_Cut1 = 2
                continue
            if any(s in line for s in Infor5):
                Flag_Cut1 = 0
                continue
            if Flag_Cut1 == 1:
                if f == 1:
                    ABA_ID.append(eval(line.split(",")[0]))
                    ABA_PosX.append(eval(line.split(",")[1]))
                    ABA_PosY.append(eval(line.split(",")[2]))
                    ABA_PosZ.append(eval(line.split(",")[3]))
                    continue
                else:
```

```
                    continue
            if Flag_Cut1 == 2:
                Dis.append(eval(line.split(",")[0]))
                Dis.append(eval(line.split(",")[1]))
                Dis.append(eval(line.split(",")[2]))
                Flag_Cut1 = 0
                continue
            if Flag_Cut1 == 0:
                continue

    IndexRaw = []
    IndexTemp = []
    Index = []
    CShearMax = []
    CShearMin = []
    CShear = []
    ABA_Cri1 = []  # array stores Von-Mises value of each node in Part2
# !!!!! save for future modification
    with open(str(Name)+"thCShear1.rpt") as myfile:
        text = list(islice(myfile, 3, None))
        IndexRaw = text[0].split()[0:None]
        for line in text:
            if "MAX" in line:
                CShearMax = line.split()[2:None]
            elif "MIN" in line:
                CShearMin = line.split()[2:None]
            else:
                continue
    for i in range(len(CShearMax)):
        ##    CShear.append(abs(abs(eval(CShearMin[i]))-
abs(eval(CShearMax[i]))))
        if abs(eval(CShearMax[i])) <= abs(eval(CShearMin[i])):
            CShear.append(abs(eval(CShearMin[i])))
        else:
            CShear.append(abs(eval(CShearMax[i])))

    for i in range(len(IndexRaw)):
        if "N" in IndexRaw[i]:
            IndexTemp.append(eval(IndexRaw[i][2:None]))

    for i in range(len(IndexTemp)):
        ABA_Cri1.append((IndexTemp[i], CShear[i]))
    ABA_Cri1 = list(set(ABA_Cri1))
    ABA_Cri1 = sorted(ABA_Cri1, key=lambda k: (-k[0]), reverse=True)

    IndexRaw = []
    IndexTemp = []
```

```
    Index = []
    CSlipR = []
    CSlipL = []
    ABA_Cri2 = []
    with open(str(Name)+"thCSlipL1.dat") as myfile:
        text = list(islice(myfile, 1, 2502))
        for line in text:
            CSlipL.append(line.split()[1])
            IndexTemp.append(eval(line.split()[0]))
    with open(str(Name)+"thCSlipR1.dat") as myfile:
        text = list(islice(myfile, 1, 2502))
        for line in text:
            CSlipR.append(line.split()[1])
    CSlip = []
    for i in range(len(CSlipR)):
        CSlip.append(abs(eval(CSlipR[i])) + abs(eval(CSlipR[i]) -
eval(CSlipL[i])))


    for i in range(len(IndexTemp)):
        Index.append((IndexTemp[i], CSlip[i]))
    ABA_Cri2 = list(set(Index))
    ABA_Cri2 = sorted(ABA_Cri2, key=lambda k: (-k[0]), reverse=True)

    CPressMaxRaw = []
    IndexRaw = []
    Index = []
    with open("CPRESS.rpt") as myfile:
        text = list(islice(myfile, 3, None))
        IndexRaw = text[0].split()[0:None]
        for line in text:
            if "0.150" in line:
                CPressMaxRaw = line.split()[1:None]
            else:
                continue
    for i in range(len(IndexRaw)):
        if "N" in IndexRaw[i]:
            Index.append(eval(IndexRaw[i][2:None]))


    CPressMax = []
    for i in range(len(Index)):
        CPressMax.append((Index[i], eval(CPressMaxRaw[i])))
    CPressMax = list(set(CPressMax))
    CPressMax = sorted(CPressMax, key=lambda k: (-k[0]), reverse=True)
```

```
    PressureCalculation = []
    cc = 0
    for i in range(len(ABA_ID)):
        if cc != len(CPressMax) and ABA_ID[i] == CPressMax[cc][0]:
            PressureCalculation.append((ABA_ID[i], ABA_PosX[i], ABA_PosY[i],
ABA_PosZ[i], CPressMax[cc][1]))
            cc = cc + 1

    ABA_INP = []  # array stores node-ID, X-coordinate, Y-coordinate, Z-
coordinate of each node
    ABA_INP0 = []  # array stores original node-ID, X-coordinate, Y-
coordinate, Z-coordinate of each node
    cc = 0
    for i in range(len(ABA_ID)):
        if cc != len(ABA_Cri1) and ABA_ID[i] == ABA_Cri1[cc][0]:
            ABA_INP.append((ABA_ID[i], round(ABA_PosX[i], 2),
round(ABA_PosY[i], 2), round(ABA_PosZ[i], 2),
                            ABA_Cri2[cc][1] * ABA_Cri1[cc][1]))
            ABA_INP0.append((ABA_ID[i], ABA_PosX[i], ABA_PosY[i], ABA_PosZ[i],
ABA_Cri2[cc][1] * ABA_Cri1[cc][1]))
            cc = cc + 1
        else:
            ABA_INP.append((ABA_ID[i], round(ABA_PosX[i], 2),
round(ABA_PosY[i], 2), round(ABA_PosZ[i], 2), 0))
            ABA_INP0.append((ABA_ID[i], ABA_PosX[i], ABA_PosY[i], ABA_PosZ[i],
0))

    ABA_INP = sorted(ABA_INP, key=lambda k: (-k[1], k[3], k[2]),
                     reverse=True)  # Sort list of tuples by first Ascending
and second element Descending

    Matrix = [(1, 1, 1)]  # tuples stores 3D-Matrix of each node, which
present its corresponding location in the module's shaped space
    x = y = z = 1  # initial numbers; as long as there is an extra node along
x or y or z axis, one will be added to corresponding x, y, or z
    for i in range(len(ABA_INP)):
        if i == 0:
            continue  # as there is already (1,1,1) in the Matrix
        else:
            if ABA_INP[i][1] == ABA_INP[i - 1][1] and ABA_INP[i][2] !=
ABA_INP[i - 1][2] and ABA_INP[i][3] == \
                    ABA_INP[i - 1][3]:
                y = y + 1
                Matrix.append((x, y, z))
            else:
                if ABA_INP[i][1] == ABA_INP[i - 1][1] and ABA_INP[i][2] !=
ABA_INP[i - 1][2] and ABA_INP[i][3] != \
```

```
                    ABA_INP[i - 1][3]:
                y = 1
                z = z + 1
                Matrix.append((x, y, z))
            else:
                y = z = 1
                x = x + 1
                Matrix.append((x, y, z))


start = 0
ForceID = []
Force1X = []
Force1Y = []
Force1Z = []
Force2X = []
Force2Y = []
Force2Z = []
Force3X = []
Force3Y = []
Force3Z = []
ForceF1 = []
ForceF2 = []
with open("ForceExport_Loc+CForce.csv") as myfile:
    for line in myfile:
        if "EXTRACTED DATA" in line:
            start = 1
        if start == 1:
            if "Q01" in line:
                ForceID = line.split(",")[1:None]
                ForceID = [int(i) for i in ForceID]
                continue
            if "Q02" in line:
                Force1X = line.split(",")[1:None]
                Force1X = [float(i) for i in Force1X]
                continue
            if "Q03" in line:
                Force1Y = line.split(",")[1:None]
                Force1Y = [float(i) for i in Force1Y]
                continue
            if "Q04" in line:
                Force1Z = line.split(",")[1:None]
                Force1Z = [float(i) for i in Force1Z]
                continue
            if "Q05" in line:
                Force2X = line.split(",")[1:None]
                Force2X = [float(i) for i in Force2X]
```

```
                    continue
                if "Q06" in line:
                    Force2Y = line.split(",")[1:None]
                    Force2Y = [float(i) for i in Force2Y]
                    continue
                if "Q07" in line:
                    Force2Z = line.split(",")[1:None]
                    Force2Z = [float(i) for i in Force2Z]
                    continue
                if "Q08" in line:
                    Force3X = line.split(",")[1:None]
                    Force3X = [float(i) for i in Force3X]
                    continue
                if "Q09" in line:
                    Force3Y = line.split(",")[1:None]
                    Force3Y = [float(i) for i in Force3Y]
                    continue
                if "Q10" in line:
                    Force3Z = line.split(",")[1:None]
                    Force3Z = [float(i) for i in Force3Z]
                    continue
                if "Q11" in line:
                    ForceF1 = line.split(",")[1:None]
                    ForceF1 = [float(i) for i in ForceF1]
                    continue
                if "Q12" in line:
                    ForceF2 = line.split(",")[1:None]
                    ForceF2 = [float(i) for i in ForceF2]
                    continue
    ForceFN = []
    ForceFT = []
    for i in range(len(ForceF1)):
        ForceFN.append(ForceF2[i])
        ForceFT.append(ForceF1[i])

    Tri_STL = []
    Tri_STL2 = []
    for i in range(len(ForceID)):
        Tri_STL.append((ForceID[i], round(Force1X[i], 5), round(Force1Y[i],
5), round(Force1Z[i], 5), ForceFN[i] / 3,
                        ForceFT[i] / 3))
        Tri_STL.append((ForceID[i], round(Force2X[i], 5), round(Force2Y[i],
5), round(Force2Z[i], 5), ForceFN[i] / 3,
                        ForceFT[i] / 3))
        Tri_STL.append((ForceID[i], round(Force3X[i], 5), round(Force3Y[i],
5), round(Force3Z[i], 5), ForceFN[i] / 3,
                        ForceFT[i] / 3))
```

```
        Tri_STL2.append((ForceID[i], Force1X[i], Force1Y[i], Force1Z[i],
ForceFN[i] / 3, ForceFT[i] / 3))
        Tri_STL2.append((ForceID[i], Force2X[i], Force2Y[i], Force2Z[i],
ForceFN[i] / 3, ForceFT[i] / 3))
        Tri_STL2.append((ForceID[i], Force3X[i], Force3Y[i], Force3Z[i],
ForceFN[i] / 3, ForceFT[i] / 3))

    Tri_STL = sorted(Tri_STL, key=lambda k: (-k[1], k[3], k[2]),
                     reverse=True)  # Sort list of tuples by first Ascending
and second element Descending

    Tri_Matrix = [(1, 1, 1)]
    x = y = z = 1
    for i in range(len(Tri_STL)):
        if i != 0:  # as there is already (1,1,1) in the Matrix
            if Tri_STL[i][1] == Tri_STL[i - 1][1] and Tri_STL[i][2] ==
Tri_STL[i - 1][2] and Tri_STL[i][3] == \
                    Tri_STL[i - 1][3]:
                Tri_Matrix.append(Tri_Matrix[-1])
            else:
                if Tri_STL[i][1] == Tri_STL[i - 1][1] and Tri_STL[i][2] !=
Tri_STL[i - 1][2] and Tri_STL[i][3] == \
                        Tri_STL[i - 1][3]:
                    y = y + 1
                    Tri_Matrix.append((x, y, z))
                else:
                    if Tri_STL[i][1] == Tri_STL[i - 1][1] and Tri_STL[i][2] !=
Tri_STL[i - 1][2] and Tri_STL[i][3] != \
                            Tri_STL[i - 1][3]:
                        y = 1
                        z = z + 1
                        Tri_Matrix.append((x, y, z))
                    else:
                        y = z = 1
                        x = x + 1
                        Tri_Matrix.append((x, y, z))

    total = [(Tri_STL[0][1], Tri_STL[0][2], Tri_STL[0][3], Tri_STL[0][4],
Tri_Matrix[0])]
    for i in range(len(Tri_Matrix)):
        if Tri_Matrix[i][1] == 1:
            if Tri_STL[i][1] == total[-1][0] and Tri_STL[i][3] == total[-
1][2]:
                total[-1] = (total[-1][0], total[-1][1], total[-1][2], total[-
1][3] + Tri_STL[i][4], total[-1][4])
            else:
                total.append((Tri_STL[i][1], Tri_STL[i][2], Tri_STL[i][3],
```

```
Tri_STL[i][4], Tri_Matrix[i]))

    xdirection = [total[0][0]]
    height = [total[0][3]]
    for i in range(len(total)):
        if total[i][0] == xdirection[-1]:
            height[-1] = height[-1] + total[i][-2]
        else:
            xdirection.append(total[i][0])
            height.append(total[i][3])
    out = open("demon.txt", "w")
    out1 = open("demon2.txt", "w")
    x, y, z, c, m = zip(*total)
    ii = 2
    for i in range(len(m)):
        if c[i] > 1e-3:
            ind = Matrix.index(m[i])
            out.write(
                "*Nset,nset=TheLoad" + str(ii) + ",internal,instance=BOTTOM-
1\n" + str(ABA_INP[ind][0]) + ",,,\n")
            out1.write("TheLoad" + str(ii) + ",2,-" + str(c[i]) + "\n")
            ii = ii + 1

    out = open("Location_Force.txt","w")
    for i in range(len(total)):
        if total[i][-1][1] == 1:
            out.write(str(total[i][-1])+"," + str(total[i][-2])+"\n")
    out.close()

def Unstable(Location1,Location2):
    Dis =[]
    for i in range(len(Location1)):
        DisX = (Location1[i][1]-Location2[i][1])*1000
        DisY = (Location1[i][2]-Location2[i][2])*1000
        DisZ = (Location1[i][3]-Location2[i][3])*1000
        Dis.append(math.sqrt(DisX**2 + DisY**2 + DisZ**2))
    if max(Dis) > 0.00001:
        return True
    else:
        return False

def Geo_Modi():
    out = open("22.stl", "w")
    with open("2.stl") as myfile:
        for line in myfile:
            if "vertex" in line:
                data = line.split()
```

```
                out.write("vertex " + str(eval(data[3]) - 1.5) + " " + str(-
eval(data[2]) + 0.45) + " " + str(
                    eval(data[1]) + 1.5) + "\n")
            else:
                out.write(line)
    out.close()
    os.remove("2.stl")
    os.rename("22.stl", "2.stl")

if __name__ == "__main__":
    Added = 140
    Force = 0
    FileChange("CPRESS.py",Added)
    os.system("abaqus cae nogui=CPRESS.py")
    os.system("abaqus cae nogui=GeoExtract.py")
    Geo_Modi()
    Force = main(140)
    if
os.path.exists("C:\Users\pl327\Documents\EDEM_2.7.3\Factories\preference_file.
txt") == True:

os.remove("C:\Users\pl327\Documents\EDEM_2.7.3\Factories\preference_file.txt")
        shutil.copyfile("preference_file.txt",
"C:\Users\pl327\Documents\EDEM_2.7.3\Factories\preference_file.txt")
    else:
        shutil.copyfile("preference_file.txt",
"C:\Users\pl327\Documents\EDEM_2.7.3\Factories\preference_file.txt")

    Location1 =[]
    baby_Stable = subprocess.Popen("edem.exe -console -i ForStable.dem -p 4 -
rewind -cs -e ForStable.dfg")
    time.sleep(10)
    if baby_Stable.poll() is None:
        baby_Control1 = Popen("Run_Stable.exe")
    while baby_Control1.poll() is None:
        time.sleep(10)
    Location1 = ReadLocation()
    ForStableCount = 5
    while ForStableCount > 0:
        print "stable", ForStableCount, "th"
        if
os.path.exists("C:\Users\pl327\Documents\EDEM_2.7.3\Factories\preference_file.
txt") == True:

os.remove("C:\Users\pl327\Documents\EDEM_2.7.3\Factories\preference_file.txt")
            shutil.copyfile("ParticleLoc.txt",
```

```
"C:\Users\pl327\Documents\EDEM_2.7.3\Factories\preference_file.txt")
        else:
            shutil.copyfile("ParticleLoc.txt",

"C:\Users\pl327\Documents\EDEM_2.7.3\Factories\preference_file.txt")
        baby_Stable = subprocess.Popen("edem.exe -console -i ForStable.dem -p
4 -rewind -cs -e ForStable.dfg")
        time.sleep(10)
        if baby_Stable.poll() is None:
            baby_Control1 = Popen("Run_Stable.exe")
        while baby_Control1.poll() is None:
            time.sleep(10)
        Location1 = ReadLocation()
        ForStableCount -= 1
    print "complete stable procedure"
    baby_Gravity = subprocess.Popen("edem.exe -console -i ForGravity.dem -p 4
-rewind -e ForGravity.dfg")
    time.sleep(10)
    if baby_Gravity.poll() is None:
        baby_Control2 = Popen("Run_Gravity.exe")
    while baby_Control2.poll() is None:
        time.sleep(10)
    ReadLocation()
    if
os.path.exists("C:\Users\pl327\Documents\EDEM_2.7.3\Factories\preference_file.
txt") == True:

os.remove("C:\Users\pl327\Documents\EDEM_2.7.3\Factories\preference_file.txt")
        shutil.copyfile("ParticleLoc.txt",
"C:\Users\pl327\Documents\EDEM_2.7.3\Factories\preference_file.txt")
    else:
        shutil.copyfile("ParticleLoc.txt",
"C:\Users\pl327\Documents\EDEM_2.7.3\Factories\preference_file.txt")
    baby1 = subprocess.Popen('edem.exe -console -i ForRun.dem -p 4 -rewind -cs
-e ForRun.dfg')
    time.sleep(10)
    if baby1.poll() is None:
        baby = Popen("FinalVersion0929.exe")
    while baby1.poll() is None:
        time.sleep(10)
    ForceRead(Added,Force)
```

Appendix III

The script of the cycle-jumping algorithm

```
Z1 = [] # coefficient of 3 order item in fitting function
Z2 = [] # coefficient of 2 order item in fitting function
Z3 = [] # coefficient of 1 order item in fitting function
Z4 = [] # coefficient of constant item in fitting function
Jump_List = []
Flag = 0

x = [0] # readin x value
y = [0] # readin y value
out = open("Result_Predictor.csv","w")
out2 = open("Result_Indicator.csv","w")
with open("Result.csv") as myfile:
  for line in myfile:
      if Flag != 0:
          Flag  = Flag - 1
          continue

      x.append(eval(line.split(",")[0]))
      y.append(eval(line.split(",")[1])+y[-1])
      out.write(str(x[-1])+","+str(y[-1])+"\n")
      maximum_jump_times = 15
      maximum_diff = 0.05
      if len(y) >= 6:
          if x[-1] < 300:
              z = np.polyfit(x, y, 3)
              f = np.poly1d(z)
              Z1.append(z[0])
              Z2.append(z[1])
              Z3.append(z[2])
              Z4.append(z[3])
              if len(Z1) > 1:
                  out2.write(str(x[-
1])+","+str(z[0])+","+str(z[1])+","+str(z[2])+","+str(z[3])+",")

                  if x[-1] > 300:
                      maximum_diff = maximum_diff*2

                  if max(abs((Z1[-1]-Z1[-2])/Z1[-2]),abs((Z2[-1]-Z2[-2])/Z2[-
2]),abs((Z3[-1]-Z3[-2])/Z3[-2]),
                         abs((Z4[-1]-Z4[-2])/Z4[-2])) < maximum_diff:
                      out2.write("CorrectFitted,")

                      ### set different maximum jumping times limitatio
                      # if x[-1] < 200:
                      #     maximum_jump_times = int(maximum_jump_times/1.2)
                      # elif x[-1] < 500:
                      #     maximum_jump_times = int(maximum_jump_times/2.4)
```

```
                    # elif x[-1] < 10000:
                    maximum_jump_times = int(maximum_jump_times/3.6)

                    if np.prod(Jump_List[-maximum_jump_times:]) != 0:
                        out2.write("ContinuseJumpNotAllowed\n")
                        x = x[-1:]
                        y = y[-1:]
                        Jump_List.append(0)
                        continue
                    else:
                        ratio = Z1[-1]*6*x[-1]+Z2[-1]*2
                        slop = Z1[-1]*3*x[-1]*x[-1]+Z2[-1]*2*x[-1]+Z3[-1]

                        ### set different maximum jumpinng length limitation

Jump_List.append(min(int(abs(maximum_diff*slop/ratio)),60))
                        # if x[-1] < 200:
                        #
Jump_List.append(min(int(abs(maximum_diff*slop/ratio)),30))
                        # elif x[-1] < 300:
                        #
Jump_List.append(min(int(abs(maximum_diff*slop/ratio)),60))

                        out2.write("ALLOWED!,"+str(Jump_List[-1])+"\n")
                        if Jump_List[-1] != 0:
                            Flag = Jump_List[-1]
                            X_new=x[-1]+Jump_List[-1]
                            # Y_new=f(X_new)
                            Y_new=y[-1]+0.5*(slop+Jump_List[-
1]*ratio+slop)*Jump_List[-1]
                            x.append(X_new)
                            y.append(Y_new)
                            out.write(str(x[-1])+","+str(y[-1])+"\n")
                    else:
                        Jump_List.append(0)
                        out2.write("\n")
                else:
                    out2.write(str(x[-
1])+","+str(z[0])+","+str(z[1])+","+str(z[2])+","+str(z[3])+"\n")
            else:
                z = np.polyfit(x, y, 1)
                f = np.poly1d(z)
                Z1.append(z[0])
                Z2.append(z[1])
                if len(Z1) > 1:
                    out2.write(str(x[-1])+","+str(z[0])+","+str(z[1])+",")
```

```
                    ### set different maximum allowable difference
                    if x[-1] > 300:
                        maximum_diff = maximum_diff/20

                    if max(abs((Z1[-1]-Z1[-2])/Z1[-2]),abs((Z2[-1]-Z2[-2])/Z2[-
2])) < maximum_diff:
                        out2.write("CorrectFitted,")

                        ### set different maximum jumping times limitation
                        # if x[-1] < 200:
                        #     maximum_jump_times = int(maximum_jump_times/1.2)
                        maximum_jump_times = int(maximum_jump_times/3.6)


                        if np.prod(Jump_List[-maximum_jump_times:]) != 0:
                            out2.write("ContinuseJumpNotAllowed\n")
                            x = x[-1:]
                            y = y[-1:]
                            Jump_List.append(0)
                            continue
                        else:
                            ratio = (Z1[-1]-Z1[-2])
                            slop = Z1[-1]

                            ### set different maximum jumpinng length limitation

Jump_List.append(min(int(abs(maximum_diff*slop/ratio)),8000))

                            out2.write("ALLOWED!,"+str(Jump_List[-1])+"\n")
                            if Jump_List[-1] != 0:
                                Flag = Jump_List[-1]
                                X_new=x[-1]+Jump_List[-1]
                                # Y_new=f(X_new)
                                Y_new=y[-1]+0.49*(slop+Jump_List[-
1]*ratio+slop)*Jump_List[-1]
                                x.append(X_new)
                                y.append(Y_new)
                                out.write(str(x[-1])+","+str(y[-1])+"\n")
                    else:
                        Jump_List.append(0)
                        out2.write("\n")
                else:
                    out2.write(str(x[-1])+","+str(z[0])+","+str(z[1])+"\n")
            else:
                continue
```

Appendix IV

The script of different material evolution criteria

von Mises & S22:

```
with open ("MaxS22U.rpt") as myfile:
    text = list(islice(myfile, 4, None))
    for line in text:
        if "MAX" in line:
            S12Max = line.split()[2:None]
        if "MIN" in line:
            S12Min = line.split()[2:None]
        else:
            continue
for i in range(len(S12Max)):
    ABA_Cri1.append(abs(eval(S12Min[i])))
```

DFDI:

```
ABA_Cri1 = [] # array stores Von-Mises value of each node in Part2 # !!!!!
save for future modification

with open ("MaxDFDIPart.rpt") as myfile:
    text = list(islice(myfile, 3, None))
    for line in text:
        if "MAX" in line:
            PEEQMax = line.split()[2:None]
        else:
            continue
with open ("vonMisesPart.rpt") as myfile:
    text = list(islice(myfile, 3, None))
    for line in text:
        if "MAX" in line:
            vMisesMax = line.split()[2:None]
        if "MIN" in line:
            vMisesMin = line.split()[2:None]
        else:
            continue
with open ("PrinMaxPart.rpt") as myfile:
    text = list(islice(myfile, 3, None))
    for line in text:
        if "MAX" in line:
            PrinMaxMax = line.split()[2:None]
        if "MIN" in line:
            PrinMaxMin = line.split()[2:None]
        else:
```

```
            continue
with open ("PrinMidPart.rpt") as myfile:
    text = list(islice(myfile, 3, None))
    for line in text:
        if "MAX" in line:
            PrinMidMax = line.split()[2:None]
        if "MIN" in line:
            PrinMidMin = line.split()[2:None]
        else:
            continue
with open ("PrinMinPart.rpt") as myfile:
    text = list(islice(myfile, 3, None))
    for line in text:
        if "MAX" in line:
            PrinSmaMax = line.split()[2:None]
        if "MIN" in line:
            PrinSmaMin = line.split()[2:None]
        else:
            continue

PEEQMax = [float(i) for i in PEEQMax]
vMisesMax = [float(i) for i in vMisesMax]
vMisesMin = [float(i) for i in vMisesMin]
PrinMaxMax = [float(i) for i in PrinMaxMax]
PrinMaxMin = [float(i) for i in PrinMaxMin]
PrinMidMax = [float(i) for i in PrinMidMax]
PrinMidMin = [float(i) for i in PrinMidMin]
PrinSmaMax = [float(i) for i in PrinSmaMax]
PrinSmaMin = [float(i) for i in PrinSmaMin]


print "MaxPEEQMax: ", max(PEEQMax)
vMises = []
for i in range(len(vMisesMax)):
    vMises.append(vMisesMax[i]-vMisesMin[i])
print "vMises: ", max(vMises)
PrinMax = []
for i in range(len(PrinMaxMax)):
    PrinMax.append(PrinMaxMax[i]-PrinMaxMin[i])
print "PrinMax: ", max(PrinMax)
PrinMid = []
for i in range(len(PrinMidMax)):
    PrinMid.append(PrinMidMax[i]-PrinMidMin[i])
print "PrinMid: ", max(PrinMid)
PrinSma = []
for i in range(len(PrinSmaMax)):
    PrinSma.append(PrinSmaMax[i]-PrinSmaMin[i])
```

```
print "PrinSma: ", max(PrinSma)

PrinAve = []
for i in range(len(PrinMax)):
    PrinAve.append((PrinMax[i]+PrinMid[i]+PrinSma[i])/3.0)
print "PrinAve: ", max(PrinAve)

FailE = []
SSum = sum(vMises)
for i in range(len(PrinAve)):
    FailE.append(1.65*0.118*(math.exp(-1.5*PrinAve[i]/(SSum))))
print "FailE: ", max(FailE)
xx = max(FailE)

for i in range(len(FailE)):
    ABA_Cri1.append(PEEQMax[i]/FailE[i])
print "ABA_Cri1: ", max(ABA_Cri1), min(ABA_Cri1)
```

Critical Plane Approach:

```
NormalR11 = []
NormalR22 = []
NormalR33 = []
NormalL11 = []
NormalL22 = []
NormalL33 = []
ShearR12 = []
ShearR13 = []
ShearR23 = []
ShearL12 = []
ShearL13 = []
ShearL23 = []
with open("SS11_22_33R.rpt") as myfile:
    for line in myfile:
        if "125" in line:
            content = line.split()[1:None]
            NormalR11 = content[0:20008]
            NormalR22 = content[20008:40016]
            NormalR33 = content[40016:None]

with open("SS11_22_33L.rpt") as myfile:
    for line in myfile:
        if "150" in line:
            content = line.split()[1:None]
            NormalL11 = content[0:20008]
            NormalL22 = content[20008:40016]
            NormalL33 = content[40016:None]
```

```
with open("SS12_13_23R.rpt") as myfile:
    for line in myfile:
        if "125" in line:
            content = line.split()[1:None]
            ShearR12 = content[0:20008]
            ShearR13 = content[20008:40016]
            ShearR23 = content[40016:None]

with open("SS12_13_23L.rpt") as myfile:
    for line in myfile:
        if "150" in line:
            content = line.split()[1:None]
            ShearL12 = content[0:20008]
            ShearL13 = content[20008:40016]
            ShearL23 = content[40016:None]

print "start find orientation"
import numpy as np
import math

Damage = 0
angleX = 0
angleY = 0
print len(ShearR12)
for ele in range(len(ShearR12)):
    if ele%1000 == 0:
        print ele
    x =
np.matrix([[eval(NormalR11[ele]),eval(ShearR12[ele]),eval(ShearR13[ele])],

[eval(ShearR12[ele]),eval(NormalR22[ele]),eval(ShearR23[ele])],

[eval(ShearR13[ele]),eval(ShearR23[ele]),eval(NormalR33[ele])]])
    y =
np.matrix([[eval(NormalL11[ele]),eval(ShearL12[ele]),eval(ShearL13[ele])],

[eval(ShearL12[ele]),eval(NormalL22[ele]),eval(ShearL23[ele])],

[eval(ShearL13[ele]),eval(ShearL23[ele]),eval(NormalL33[ele])]])
    for i in range(26, 34, 2):
        for j in range(0, 30, 10):
            n =
np.matrix([[math.sin(i)*math.cos(j)],[math.sin(i)*math.sin(j)],[math.cos(i)]])
            a = np.matrix([[math.sin(j)],[-math.cos(j)],[0]])
            b =
np.matrix([[math.cos(i)*math.cos(j)],[math.cos(i)*math.sin(j)],[-
math.sin(i)]])
```

```
            xR = np.dot(n.transpose(), np.dot(x,n))
            xL = np.dot(n.transpose(), np.dot(y,n))

            yR = 0
            yRangle = 0
            yL = 0
            yLangle = 0
            for k in range(0, 180, 5):
                v1 = np.squeeze(np.asarray(a))
                v2 = np.squeeze(np.asarray(b))
                q =
np.array([v1[0]*math.cos(k)+v2[0]*math.sin(k),v1[1]*math.cos(k)+v2[1]*math.sin
(k),math.sin(k)*v2[2]])
                temp = np.dot(q,np.dot(x,n))
                if temp > yR:
                    yR = temp
                    yRangle = k

                temp = np.dot(q,np.dot(y,n))
                if temp > yL:
                    yL = temp
                    yLangle = k

            yAmp = 0.5*math.sqrt(yR*yR+yL*yL-2*yR*yL*math.cos(abs(yRangle-
yLangle)))
            xAmp = max(np.squeeze(np.asarray(xR)),np.squeeze(np.asarray(xL)))
            if yAmp/1218 + xAmp/2264 > Damage:
                Damage = yAmp/1272 + xAmp/2264
                angleX = i
                angleY = j
print Damage, angleX, angleY
os.system("pause")
angleX = 30
angleY = 0

with open("SS11_22_33_MMFirstSingle.rpt") as myfile:
    for line in myfile:
        if "MAX" in line:
            content = line.split()[2:None]
            NormalR11 = content[0:20008]
            NormalR22 = content[20008:40016]
            NormalR33 = content[40016:None]
        if "MINI" in line:
            content = line.split()[2:None]
            NormalL11 = content[0:20008]
            NormalL22 = content[20008:40016]
            NormalL33 = content[40016:None]
```

```
with open("SS12_13_23_MMFirstSingle.rpt") as myfile:
    for line in myfile:
        if "MAX" in line:
            content = line.split()[2:None]
            ShearR12 = content[0:20008]
            ShearR13 = content[20008:40016]
            ShearR23 = content[40016:None]
        if "MINI" in line:
            content = line.split()[2:None]
            ShearL12 = content[0:20008]
            ShearL13 = content[20008:40016]
            ShearL23 = content[40016:None]

print "start processing"
ABA_Cri1 = []
ABA_Cri2 = []
for i in range(len(ShearR12)):
    x = np.matrix([[eval(NormalR11[i]), eval(ShearR12[i]), eval(ShearR13[i])],
                   [eval(ShearR12[i]), eval(NormalR22[i]), eval(ShearR23[i])],
                   [eval(ShearR13[i]), eval(ShearR23[i]),
eval(NormalR33[i])]])
    y = np.matrix([[eval(NormalL11[i]), eval(ShearL12[i]), eval(ShearL13[i])],
                   [eval(ShearL12[i]), eval(NormalL22[i]), eval(ShearL23[i])],
                   [eval(ShearL13[i]), eval(ShearL23[i]),
eval(NormalL33[i])]])
    n = np.matrix([[math.sin(angleX) * math.cos(angleY)], [math.sin(angleX) *
math.sin(angleY)], [math.cos(angleX)]])
    a = np.matrix([[math.sin(angleY)], [-math.cos(angleY)], [angleY]])
    b = np.matrix([[math.cos(angleX) * math.cos(angleY)], [math.cos(angleX) *
math.sin(angleY)], [-math.sin(angleX)]])

    xR = np.dot(n.transpose(), np.dot(x, n))
    xL = np.dot(n.transpose(), np.dot(y, n))

    yR = 0
    yRangle = 0
    yL = 0
    yLangle = 0
    for k in range(0, 180, 10):
        v1 = np.squeeze(np.asarray(a))
        v2 = np.squeeze(np.asarray(b))
        q = np.array([v1[0] * math.cos(k) + v2[0] * math.sin(k), v1[1] *
math.cos(k) + v2[1] * math.sin(k),
                      math.sin(k) * v2[2]])
        temp = np.dot(q, np.dot(x, n))
        if temp > yR:
```

```
            yR = temp
            yRangle = k

        temp = np.dot(q, np.dot(y, n))
        if temp > yL:
            yL = temp
            yLangle = k

    yAmp = 0.5 * math.sqrt(yR * yR + yL * yL - 2 * yR * yL *
math.cos(abs(yRangle - yLangle)))
    xAmp = max(np.squeeze(np.asarray(xR)), np.squeeze(np.asarray(xL)))
    ABA_Cri1.append(yAmp / 1218 + xAmp / 2264)
    para = ABA_Cri1[-1]
    if para <= 0:
        ABA_Cri2.append(0)
    else:
        ABA_Cri2.append(1 / (0.5 * math.pow(10, (-(math.log10(para)) /
0.103))))
```

In-plane Shear Stress:

```
content = []
S12Max = []
S12Min = []
with open(name + "thS12" + str(item) + "MaxMin.rpt") as myfile:
    text = list(islice(myfile, 4, None))
    for line in text:
        if "MAX" in line:
            S12Max = line.split()[2:None]
        if "MIN" in line:
            S12Min = line.split()[2:None]

out = open("S12Saturated" + str(item) + ".txt","w")
TotalShear = []
for i in range(len(S12Max)):
    TotalShear.append((eval(S12Max[i])-eval(S12Min[i]))/ ShearLimit)
    out.write(str(TotalShear[-1])+" ")
out.close()
ABA_Cri1 = TotalShear
```