POLITECNICO DI MILANO

Facoltà di Ingegneria Industriale

Corso di Laurea in
Ingegneria Meccanica

Forces On Bearings: Variational Approach

Relatore: Prof. Paolo PENNACCHI

Tesi di Laurea di:

Filippo VISCHI Matr. 739744

Anno Accademico 2010 - 2011

Capitolo 1

Introduzione

Il capitolo introduttivo spiega in poche pagine i motivi per cui è necessaria una corretta lubrificazione degli organi rotanti, i danni che possono derivare da un'errata alimentazione del cuscinetto stesso, e i costi che questa comporta.

La trattazione spazia poi sui vari tipi di lubrificanti utilizzati nelle normali applicazioni, evidenziandone le varie caratteristiche e i limiti di utilizzo. Il sottocapitolo conclusivo illustra 3 diversi tipi di cuscinetti a lubrificazione a film d'olio, soffermandosi sulle caratteristiche costruttive di ognuno.

In particolar modo, vengono descritte le caratteristiche dei cuscinetti a lubrificazione idrodinamica, dove l'elemento rotante galleggia su un sottile film d'olio di una definita viscosità e ad una specifica temperatura. La lubrificazione idrodinamica, sul piano puramente tecnologico, implica una maggiore complessità d'impianto: la presenza di olio in pressione, infatti, implica l'installazione di pompe, filtri e sistemi di raffreddamento necessari al corretto funzionamento del cuscinetto stesso. Inoltre, ci sono dei temi di natura ambientale legati allo smaltimento dell'olio esausto.

Per assicurare alte performance del cuscinetto, oltre all'aspetto ingegneristico, è necessario conoscere a fondo il comportamento dinamico del film d'olio stesso. All'interno del film, la rotazione dell'albero genera un campo di forze in grado di sostentare l'albero stesso: questo campo di forza presenta caratteristiche di rigidezza e smorzamento, e va quindi ad influire sul comportamento dinamico del rotore.
Chiaramente, nel caso in cui l'albero non giri ad una velocità sufficiente, il campo di forze non è in grado di supportare l'albero stesso e si genera attrito molto dannoso per i componenti meccanici. La creazione di questo campo di forze è descritta analiticamente dall'equazione di Reynolds, la cui integrazione sarà oggetto di studio nei capitoli successivi.

Capitolo 2

Idrodinamica: teoria di Reynolds

L'oggetto del secondo capitolo della tesi è la teoria classica di Reynolds dell'idrodinamica. Introdotte le ipotesi necessarie allo studio idrodinamico (forze di volume trascurabili, fluido Newtoniano, flusso laminare, inerzia del fluido trascurabile, densità del fluido costante, viscosità costante attraverso il film d'olio), il comportamento del film d'olio viene studiato a partire dalle equazioni di equilibrio di un elemento infinitesimo di fluido stesso. Sostituendo nell'equazione di equilibrio le equazioni corrispondenti alle ipotesi prima citate, si arriva alle espressioni analitiche delle velocità della particella di fluido nelle due direzioni x e y.

La seconda equazione necessaria ad ottenere l'equazione di Reynolds si ricava dall'equazione di continuità di flusso in una colonna. Anche in questo caso, si sostituiscono nell'equazione di continuità le ipotesi iniziali al fine di ricavare l'espressione completa e definitiva dell'equazione di Reynolds:

$$\frac{\partial}{\partial x}\left(\frac{h^3}{\eta}\frac{\partial p}{\partial x}\right)+\frac{\partial}{\partial y}\left(\frac{h^3}{\eta}\frac{\partial p}{\partial y}\right)=6\left(U\frac{dh}{dx}+V\frac{dh}{dy}\right)+12(w_h-w_0)$$

A causa della complessità dell'equazione, per scopi ingegneristici si tende a semplificare l'espressione stessa imponendo che la velocità sia unidirezionale.
In questo modo si ottiene l'espressione semplificata dell'equazione di Reynolds:

$$\frac{\partial}{\partial x}\left(\frac{h^3}{\eta}\frac{\partial p}{\partial x}\right)+\frac{\partial}{\partial y}\left(\frac{h^3}{\eta}\frac{\partial p}{\partial y}\right)=6U\frac{dh}{dx}+12(w_h-w_0)$$

utile allo studio delle applicazioni trattate.

Un ulteriore adeguamento al sistema oggetto dello studio è la trasformazione dell'equazione di Reynolds in coordinate polari:

$$\frac{\partial}{\partial x}\left(\frac{\rho h^3}{\mu}\frac{\partial p}{\partial x}\right)+\frac{\partial}{\partial z}\left(\frac{\rho h^3}{\mu}\frac{\partial p}{\partial z}\right)=6U\frac{\partial(\rho h)}{\partial x}+12\rho V_0$$

Capitolo 3

Variational Approach

A partire dall'equazione di Reynolds in coordinate polari, viene studiata una soluzione alternativa al classico metodo ad elementi finiti, in grado di ridurre di almeno una decina di volte i tempi computazionali.

Il metodo proposto è il Variational Approach: siccome lo scopo dello studio non è ottenere un'accurata distribuzione di pressione all'interno del cuscinetto ma l'effetto generale del film d'olio con una sufficiente precisione, il metodo variazionale è una soluzione efficace a questo tipo di problemi. Nel caso in cui si abbia rottura del film d'oliò, però, il metodo variazionale non è più adeguato allo studio del sistema dinamico.

Il metodo variazionale qui proposto inoltre, può essere esteso anche ad applicazioni con cuscinetti di geometria ellittica. I pad di tipo ellittico, infatti, sono trattati come pad circolari ma con un centro equivalente situato in una posizione differente rispetto al centro reale del cuscinetto. Per ottenere la soluzione, vengono geometricamente calcolati i parametri equivalenti del cuscinetto (posizione del centro, attitude angle, etc..) e si applica il procedimento risolutivo descritto per i cuscinetti circolari.

Il metodo è valido anche in caso di sistema non stabile, e quindi con componenti di velocità non rotazionale differenti da zero. In questa situazione, il tempo totale computazionale risulterà chiaramente più elevato, ma comunque notevolmente ridotto rispetto ad un tradizionale metodo ad elementi finiti.

L'algoritmo risolutivo parte dall'imposizione delle condizioni al contorno (ovvero che la pressione sia nulla ai bordi del pad) e dall'imposizione della condizione di positività del vettore pressione in ogni punto della mesh.

Applicando rigorosamente il metodo variazionale, si arriva ad una equazione matriciale del tipo

$$Kp=f$$

Dove p è il vettore delle pressioni ed è l'incognita. L'equazione può velocemente essere risolta con una fattorizzazione simile ad una fattorizzazione di tipo LU. La peculiarità dell'algoritmo risolutivo risiede nel "chasing process" utilizzato per ricavare il vettore delle pressioni, simile al tradizionale processo risolutivo di un sistema di tipo LU ma con la differenza che l'algoritmo impone alla soluzione di essere sempre positiva in tutti i punti della mesh. In questo modo si ottiene una distribuzione di pressione lievemente più elevata della pressione reale, ma che poi nel calcolo delle forze non va ad inficiare l'errore percentuale sul risultato finale.

Una volta risolto il sistema LU ed ottenuto il vettore delle pressioni, si passa al calcolo delle forze con delle semplici moltiplicazioni del vettore delle pressioni. Il punto di forza dell'algoritmo risiede proprio in questo passaggio: mentre un metodo agli elementi finiti procede con l'integrazione punto per punto del campo delle pressioni, il metodo variazionale semplicemente moltiplica fra di loro dei vettori precedentemente calcolati, velocizzando enormemente il processo.

Capitolo 4

Risoluzione al calcolatore dell'equazione di Reynolds

Il quarto capitolo dell'elaborato riporta il codice implementato in MatLab per la risoluzione dell'equazione di Reynolds attraverso il metodo variazionale. Sono illustrate tutte le subfunction utilizzate, col ciclo iterativo utilizzato per convergere al risultato, e le parti di codice utilizzate per le interfacce con l'utente. La subfunction più importante, che include i passaggi dell'algoritmo, contiene al suo interno altre subfunction necessarie al calcolo di alcuni parametri fondamentali per ottenere la soluzione. Al termine dell'integrazione, una volta ricavata la distribuzione di pressione, la subfunction forces.m calcola rapidamente le forze scaricate dal cuscinetto sull'albero.

Il programma è strutturato in modo da presentare un'interfaccia iniziale, che permette all'utente di scegliere il tipo di cuscinetto e le condizioni che si andranno a studiare.

Una volta scelto il cuscinetto e le condizioni di moto, stabili o instabili, l'utente setta i parametri geometrici del cuscinetto (estensione assiale, estensione radiale del pad, attitude angle…) e i parametri matematici necessari per l'integrazione dell'equazione di Reynolds (numero di nodi, relaxation factor, parametro m iniziale…)

Il programma fornisce in output i grafici delle distribuzioni delle pressioni sia sul pad inferiore che sul superiore (nonostante uno dei due, in regime stabile, sia scarico) e le forze lungo le due direzioni x e y. La forza totale è naturalmente la somma vettoriale delle due componenti.

Capitolo 5

Risultati

Nell'ultimo capitolo si espongono i risultati del nuovo metodo, comparati coi risultati ottenuti con un metodo ad elementi finiti.

Come si può vedere nei grafici allegati, i risultati in termini di forze adimensionali sono pressoché identici a quelli ottenuti col metodo tradizionale, con un risparmio computazionale molto elevato anche per mesh non eccessivamente fitte.

L'errore che si ottiene fra il metodo variazionale e il metodo ad elementi finiti tende a 0 per eccentricità molto elevate, con risultati ottenuti in un tempo circa 10 volte minore.

Contents

Figures and Charts

Chapter 1

Introduction

1.1 Lubrication.

The following thesis aims to a fast way to evaluate forces on bearings during the transient of a rotor. This topic needs a short introduction about what we usually refer to using the term “lubrication”. In everyday applications which involve two surfaces in relative motion, thin low shear strength layers of gas, liquid and solid are usually interposed between these two surfaces in order to improve the smoothness of motion and reduce the risks of damage of the surfaces. These layers are generally very thin and difficult to observe. Thickness of these films may vary in a range of 1-100 μm, although thinner and thicker films can be found in some applications. Knowledge related to diagnosing these layers (in order to prevent serious damages of the surfaces) and improving the performances of these films is commonly known as “lubrication”.

Although there are no restrictions on the type of materials used, different kind of materials used to form the lubrication film between the surfaces may extremely change the results of the study and influence the limits of the theory application. The branch of lubrication object of this study is “hydrodynamic lubrication”, which pertains to the detailed analysis of gaseous or liquid films. Lubrication by solid interfaces between the surfaces is generally termed “solid lubrication”. An alternative kind of lubrication, which does not count on the presence of any films, is based on putting an external force field between the opposing surfaces who maintains the right distance between the two bodies. The force field may be, for example, a magnetic force field type which applies magnetic forces in order to ensure the correct functioning of the machine. However, magnetic levitation is still at the experimental stage. This work will study that particular kind of lubrication termed “hydrodynamic lubrication”.

From a technological point of view, liquid lubrication requires more devices than other types of lubrication. Appearance of liquid film implies pumps, cooling systems, filters in order to maintain the performances of the film over a period of time. Plus, there are environmental issues associated to the disposal of used lubricants. Lubrication based on an oil film has got limits, principally associated to the loss of load carrying capacity at high temperature and degradation during the service. Performances of oil strictly depend on its chemical composition, and its physical and mechanical characteristics. So, to ensure high performances of the bearings, beneath the practical engineering aspects, it is extremely important to predict film characteristics and properties. Prediction methods involve, at a basic level, hydrodynamic, hydrostatic and elastohydrodynamic lubrication. More advanced level of study involves computational methods, since there is still no analytical method for the resolution of the equations associated.

1.2 Wear

Wear damages.

Prediction methods are pretty important for a correct functioning of the lubrication film, in order to avoid film rupture and consequent wear damages of the moving surfaces. Film failure leads to severe damages and, obviously, bad working of the machine plus extremely high costs of reparation. In these circumstances, wear is caused by adhesion of contacting bodies. This phenomenon is termed “adhesive wear”. If milder forms of wear occur by repetitive stresses and consequent fatigue processes, these milder forms of wear are termed “fatigue wear”. Two associated forms of wear are “erosive wear” (due to impacting particles versus the walls of the bearing) and “cavitation wear”, caused by fast flowing liquids. But wear can also occur due to chemical reasons: sometimes, film material may be formed by chemical attack of either contacting body. So, if this provides the effective lubrication, it involves also a massive wear damage of the bearing. This kind of damage is termed as “corrosive wear”. When the corroding agent is oxygen, the correct term is “oxidative wear”. Last of the most important types of wear is “fretting wear”. It occurs if the available space between the two moving surfaces is restricted in a few micrometers. In this particular condition, oil film keeps trapped between the surfaces and may become destructive. Wear involved in solid lubrications will not be examined, as the main topic of the work is hydrodynamic lubrication.

Wear costs.

The main problem in wear damage phenomenon is obviously the economic side of the argument. Wear damage is equal to large amount of energy and material losses, so this implies massive costs of maintenance and repairing devices. In order to avoid these losses, tribology is getting more and more importance such as the best and the only way to reduce costs and improve the performances of every machine. Basically, the following simple equation summarizes and estimates the costs involved in existing tribological practice:

$$Total\ Tribological\ Cost = Sum\ of\ Individual\ Machine\ Cost \cdot Number\ of\ Machines$$

Actually, if only we could evaluate the number of nowadays operating machines, this equation should give us an idea of savings which a good lubrication could imply. However, such as in every engineering application, every single case requires a detailed investment plan in order to evaluate costs of tribology versus possible savings. Anyway, economics of tribology assume huge proportions, as we can see from the following data: in 1966, Peter Jost evaluated that by application of the basic principles of tribology, economy of U.K. could save approximately 515 £ per year (at 1965 values) [1]. A similar report published in Germany, in 1976, shows that economic losses by friction and wear cost about 10 million DM per year (at 1975 values) [2]. An U.S.A. report estimated than about 11% of whole annual energy can be saved in the four major areas of transportation, turbo machinery, power generation and industrial processes by practical use of tribological improvements [3]. Problems of tribology economics thus become such important to an engineer. Concerning for example pneumatic transportation of material through pipes, the erosive wear at bends can be up to 50 times more than wear in straight sections [4].

As soon as tribology comes so important and expensive, engineers and researchers aim to new ways to get more efficient machines, especially studying new materials and lubricants. Research nowadays brought some radical improvements, such radical to wholly change the technology and the economics of some products. A classic example of tribology improvements is the adiabatic engine. Development behind adiabatic engine aims to use a dry, high temperature self lubricant instead of traditional oils and lubricants as so to have a chance to turn heat (formerly absorbed by obsolete radiator) in mechanical work. This kind of technique should improve the performances of traditional engines, reducing maintenance and operating costs and increasing enginès fuel efficiency.

Generally, and above all easier, wear damages can be controlled by a correct choice of components and materials properties, depending on conditions of use of the machine.

1.3 Lubricants and their composition.

A lubricant (sometimes referred to as "lube") is a substance (often a liquid) introduced between two moving surfaces to reduce the friction between them, improving efficiency and reducing wear. They may also have the function of dissolving or transporting foreign particles and of distributing heat. The following sub-chapter aims to explain how different chemical composition of the lubricant may vary the performance of the machine, in order to choose the best kind of lubricant for every single application. Physical properties of the lubricant are strictly connected to his chemical composition. Thus, lubricants are used to be classified essentially by their origin: first grade of the stair is distinction between biological and non-biological oils. Non-biological oils class provides a huge array of hydrocarbon compounds: these substances are usually present as complex mixtures and so they can be used for many more purposes, like control of wear and friction. Nowadays, technological processes require a high grade of performances from the lubricants. That is the reason why selection of correct oil for the purposes of lubrication become such an important step during the projecting phase of the operating machine. For example, most of the common used natural oils contain substances which can compromise lubrication properties, but they also contain substances essential to satisfy application requirements. The balance between purity and impurities is critical concerning oxidation stability of the oil and it may vary depending on the application of the lubricant. Problem of oxidation stability is solved by deliberately adding to the natural formula of the oil plenty of substances called “additives”, able to radically change the behaviour of the lubricant. Additives percentage also influences specific characteristics of the lubricant, such as corrosion tendency, foaming, clotting, oxidation, wear, friction and other properties.

Lubricant performances are rated essentially by two different aspects: achieving of required level of friction and wear rates, and maintaining these standards in spite of continuous degradation of the lubricant. Additives present in the oil also deteriorate during operation due to the contact with metallic parts of the machine and the environment. Degradation is inevitable, so needs to be postponed to the predicted lifetime of the machine.

Plain mineral oil is defined as base stock. A typical lubricating oil is composed of 95% base stock and 5% additives. Usually, base stock is chemically inert, as so to avoid unrequested chemical reactions between the lubricant and the outside environment. Actually, base stocks are rated by their source: therefore, we usually divide base stocks in biological, mineral and synthetic. The choice of the correct kind of base stock obviously depends on application considered: biological oils are used to be applied in application where risk of contamination must be reduced to a minimum (like food or pharmaceutical industry). Mineral oils are commonly used in almost all industry spheres that require moderate operating temperatures (gears, bearings, engines...). Finally, synthetic oils are developed to substitute mineral oils in those particular applications in which mineral oils can’t guarantee high performances due to the inevitable deterioration. Synthetic oils provide a big array of possible applications, both at low and high operating temperatures.

Further lubricants do exist, such as greases. Greases are not essentially different from oils: they consist of mineral or synthetic oil, but the oil is trapped in minute pockets formed by soap fibres which constitute the internal structure of the grease. Greases have

been developed principally in order to provide semi-permanent lubrication, since the oil trapped in the fibrous structure is unable to flow away from the containing surfaces.

Mineral oils.

Mineral oils are the most commonly used lubricants. That's due to the low cost of mining the crude oil which is the base from which mineral oils are manufactured. Actually, minerals oils are supposed to derive from decomposition of animal and plant matter in salt water. That's usually named as "fossil theory" [5]. According to this theory, mineral oils directly come from a transformation process driven by high temperatures and pressures, leading the organic matter to the complex hydrocarbon molecules which are the basic constituents of crude oil. By following the theory path, since 60 % of world oil resources are in the Middle East concentrated in 25 giant fields, it could seem that the Persian Gulf was a large sink for plant and animal life for few million of years. This involves that this environment did not change by the time stream, remaining the same undisturbed environment despite of earthquakes, storms, faulting, etc for millions of years. This leads to some doubts about the validity of fossil theory as the only source for mineral oils. Plus, it is hard to believe that in ancient times most of the plant and animal life on Earth was concentrated in the Persian Gulf region.
Further hypothesis about mineral oils source lie beneath the suggestions of Gold [6]. It's commonly known that many hydrocarbons are present on meteorites and they cannot possibly originate from decomposition process of any plant or animal life. Moreover, some planets in our solar system got a huge presence of hydrocarbons in their atmospheres. Thus, theory suggests that hydrocarbons on Earth generated from non-biological sources in the same way as on the other planets. New theory supposes that large reservoirs of gas and oil are still buried in the deep of our planet, waiting to be discovered. We will need an efficient deep drilling technology in order to discover these reservoirs and use them.
The structure of common mineral oils is quite complex. It is not possible to predict the exact composition of a mineral oil, due to the large array of elements composing it. Actually, mineral oils are also impure: this implies that their structure is made more and more complex and difficult to study by the presence of exterior elements. Therefore, easiest way to classify mineral oils is divide them into categories depending on the source of crude oil and refining process. Fundamental differences between mineral oils are chemical forms, sulphur content and viscosity. Chemical forms are divided themselves into paraffinic forms, naphthenic forms and aromatic forms. Sulphur content depends on the source of the crude oil and his refining process. If good lubrication and oxidation properties are required, small amount of sulphur in the oil is desirable. Basically, high amounts of sulphur often lead to an early decay of machine performances. Last classifying property is viscosity, which strictly depends on the degree of refining.

Synthetic oils.

Synthetic oil development initially originates by those countries lacking a reliable supply of mineral oils. These lubricants were expensive, so they took a bit of time to be applied by a large scale of users. Synthetic oils basically supply oxidation problems and viscosity loss at high temperatures of mineral oils, and combustion or explosion in the presence of strong oxidative agents. Essentially, synthetic lubricants and mineral oils cover the same plant needing, with differences located in costs and properties at extreme conditions of usage. Synthetic lubricants can usually be divided into 2 groups: fluids aiming to provide superior lubrication at ambient or elevated temperatures, and fluids for extremes temperature or chemical attack. Some particular kinds of synthetic lubricants provide very high performances of the machine, reflected in the cost of these oils. In 1987, the price of a halogen based synthetic lubricant was $ 450/kg, almost as silver's price.

Emulsions and aqueous lubricants.

Water too is a good base stock for lubricants: cheap, with good heat transfer properties and non flammability. Obviously, water by itself is a very poor lubricant but if mixed with oils (as so to form emulsions) it represents a good way to solve some lubrication problems. These lubricants can be used such as coolants too, thanks to good heat transfer properties of water. The strictest limitation of these lubricants lies in the temperature range in which they can be successfully used. Temperature obviously cannot cross the melting point of ice and boiling point of water, so these problem excludes aqueous lubricants from such a big part of applications (for example as engine oils).
Basically, emulsions are usually classified into W/O emulsion, which are water droplets suspended in an oil base, and the opposite O/W emulsions, which obviously are oil droplets suspended in water. Biggest emulsions deal concerns viscosity of the lubricant. It has been discovered that viscosity declines with increasing shear stress. Most interesting characteristic of emulsions lies in their behaviour in concentrated contacts operating in elastohydrodynamic lubrication regime (often named as EHL). The size of an EHL contact is comparable to the droplet size, so this suggests that the elastohydrodynamic films generated would be unstable. This, however, has not yet been confirmed experimentally: in fact, emulsions with a low grade of stability keep on giving the best lubrication
Emulsions and aqueous solutions are used especially in manufacturing sphere, used as cutting fluids in the metal working industry.

Greases.

The term “grease” usually refers to a number of solid lubricants possessing a higher viscosity than oil. A true grease is made by a liquid lubricant mixed with another lubricant, usually termed as a “soap” (in the chemical sense, so referring to a fatty acid or a metallic salt), as so to form a solid lubricant. Such as in case of emulsions, greases viscosity declines with increasing of shear stress. As soon as shear reaches a critical value, viscosity drastically drops by: this kind of behaviour makes greases considered as plastic fluids. Since the oil is unable to flow away from the lubricant chemical structure, greases usually provide semi-lubrication. This reason leads to certain strict limitations on use of greases as lubricants. Usually, most massive use of greases is in those applications in which machines can only be lubricated unfrequently and in which oil would not stay in position. In these situations, grease also covers the function of protecting the machine from water and dust. Greases are great lubricants in case of required semi-permanent lubrication, because they can be packed into the bearing and left there operating for months, before they start decaying their performances. Essentially, greases lubricating behaviour is not too different from mineral oils: the main difference is that greases structure needs to remain as a solid mass in spite of high service temperatures. If grease gets liquid form it can flow away leading to a failure.
Greases are also applied in plants rotating at unusually high (10.000 rpm) or low (5-10 rpm) speeds. In these cases, lubricants may contain additives [7]. Additives used in greases are basically the same used in mineral oils. In some cases, additive may change the basic structure of the soap in order to increase and improve the performances of the lubricant. Commonly, additives are also used to prevent oxidation.

Lubricant additives.

Lubricant additives are organic or organometallic substances mixed in a few percent with a base lubricant in order to improve the performance of the lubricant itself. Recently, additives development satisfied lot of purposes, apart from pure wear improvement: nowadays, additives are commonly applied as so to improve oxidation resistance, control of corrosion, control of contamination by reaction products, reducing excessive decrease of lubricant viscosity at high temperatures, inhibiting the generation of foam. Additives structures are subject of massive research by companies involved in this industry, because good use of additives may improve of a few percent the performances of the lubricant. For this reason, companies keep secrecy on the details of additives composition. This secrecy means that additives studies are more such an art generated by experience than an exact scientific study referred to the chemical composition of these substances.

1.4 Literature Survey.

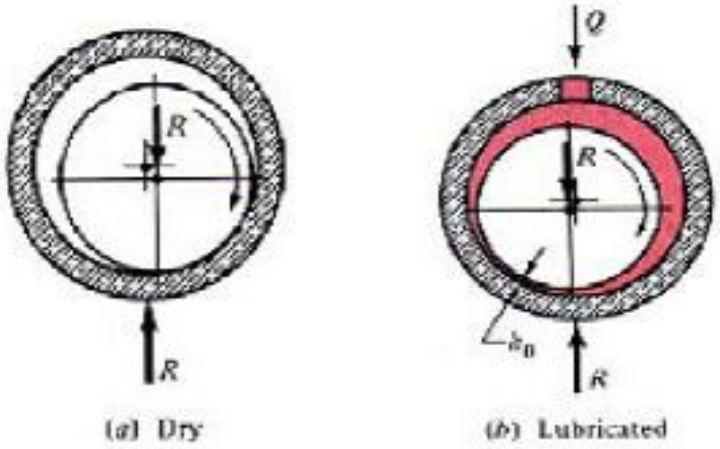
To complete the goals of this thesis, an extensive literature search was carried out to examine what have been done on hydrodynamics bearing characteristics particularly plain bearings, elliptical and off-set journal bearings. The basic concept of topic lies beneath the theory of lubrication and related to solving the Reynolds equation.

Journal bearings.

A journal bearing is the most common used kind of bearing. With this term, we refer to a bearing made by a shaft (also known as "journal") moving relatively to the bearing, separated by a thin film of oil or grease. Both the shaft and the main bearing body are usually simple cylinders with lubricant filling the gap within the two bodies. Lubricant film is thick enough to ensure that shaft and bearing never get in contact with each other. Oil is generally fed into the bearing through a hole in the cylinder body under pressure. The casing containing the whole bearing is usually termed as journal box. Liquid journal bearings can be hydrodynamically lubricated or hydrostatically lubricated. The difference between hydrostatic and hydrodynamic forces is in the way the pressure that supports the bearing is initially and subsequently maintained. Main characteristic of this kind of bearing is that the oil film, once in movement, is able to carry by itself the load applied on the bearing, by generating a gradient of pressure and a consequent force system. This is permitted by fluid-dynamics laws, especially Reynolds equations, which will be object of study in the following chapters.

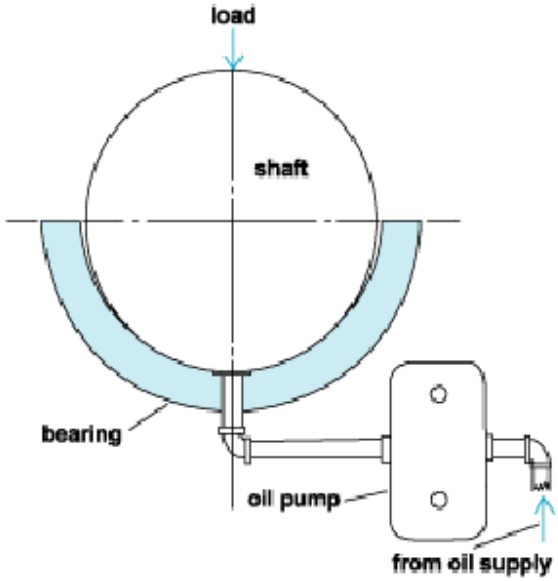
Fluid film bearings.

Fluid film also known as hydrodynamic bearing has the rotating element floating on a thin film of lubrication. It is lubricated and cooled by a continuous supply of a filtered liquid, typically oil, of a specified viscosity and at a specified temperature. The rotation of the bearing generates a hydrodynamic film with a high pressure gradient that creates a net force which supports the load. It possesses stiffness and damping properties which have an effect on the dynamic behaviour of the rotor bearing system.
Fluid film bearings have been widely used in rotating machinery. The characteristics of the bearings are very much influenced by the stiffness and damping coefficient of the bearings, which in turn are dependent on the fluid film forces at the bearings.
Hydrodynamic bearings are much more prone to initial wear because lubrication does not occur until there is rotation of the shaft. At low rotational speeds the lubrication may not attain complete separation between shaft and bush. They require much greater care in design and operation than hydrostatic bearings.



Hydrostatic Bearings.

In a hydrostatic bearing, the pressure is always present at a value that is desirable and is achieved by an external pump which forces lubricant into the system, as you can see from the diagram. The pump provides a magnitude of pressure that aims to supplement the pressure which is created by the bearings rotation.



Detailed theory and application of hydrodynamic journal bearing is explained by Fuller [8], where focuses more on practical designs and problem solving including friction, power losses and material of the bearing, analysis from origin of hydrodynamic theory and its uses.

Pinkus [9] explained the concept of lubrication and cavitations; he also covers great detail of fundamental concept of Reynolds equation in general and introduction to hydrodynamic journal bearing.

N.S Feng and E.J Hahn [10] developed software of computing the characteristics of elliptical and tilting pad bearing which is written such that the output is able to used as input data for various analysis programs. Furthermore, the computer codes were developed in such a way that the calculated bearing characteristics are readily able to be edited into tabulated data files for input to other vibration analysis programs.

L.P. Wang [11] did a PhD thesis on dynamic modelling and behaviour of tilting-pad bearings, the thesis gave 5 propositions based on the variational inequality theory of hydrodynamic lubrication. The thesis also present a weighted finite element algorithm to calculate the oil-film forces of journal bearings and using two-dimensional solution that could be interpolated by one-dimensional elements with high accuracy. The solution also proposed amendatory direct-method to solve the equation whose coefficient matrices is in diagonal form. Based on the interior characteristics, the solution of oil-forces is united with the solution of Jacobian matrices and in this

process; the operations are solved within the positive pressure region. Therefore many redundant computations involves are avoided.

H.L. Xi [12] did a thesis based on the non-linear dynamical behaviour of a finite journal bearing-rotor system. In the thesis, the analytical formula of the unsteady oil-film force of the finite journal bearing is cited. Furthermore, the methods for the motion characteristics of the system are introduced. The dynamical model of the finite journal bearing rotor system is established and the motion differential equation of the system is solved using Runge-kutta method. Motion characteristics of the finite journal bearing-rotor are investigated under the consideration of oil-film force. Poincare maps, spectrum analysis about dimensionless rotating speed, dimensionless mass eccentricity and synthetically parameter are obtained.
Various paper articles in the book "Journal Bearings for Reciprocating & Turbo Machinery" [13] gave a clear concept on the hydro-dynamic theory of dynamically loaded bearings. It also explains and investigates into the performance of dynamically loaded journal bearings. Furthermore, it includes the hydrodynamic theory of finite journal bearings using bipolar co-ordinate system.

Chapter 2

Hydrodynamics: the Reynolds' theory of lubrication.

This chapter will give an idea of the behaviour of the film oil from an analytical view point, discussing the basic principles of hydrodynamic lubrication. The first who provided an analytical proof that a viscous liquid can separate two sliding surfaces by creating a field of pressure, resulting in low friction and theoretically zero wear, was Osborne Reynolds in 1886 in his 'Proceedings of the Royal Society by Reynolds' [14]. This theory was successfully applied to bearings in 20^{th} century, and the results are bearings able to carry load of several tons at sliding speed of 10-50 m/s (in common applications such as hydroelectric power stations). In this plants, operating surfaces are totally separated by a lubricating film in order to maintain the friction coefficient at a very low level. Hydrodynamic (Full Film) Lubrication is obtained when two mating surfaces are completely separated by a cohesive film of lubricant. Usually, every kind of hydrodynamic lubrication can be explained by one mathematical equation, originally discovered by Reynolds and simply called "Reynolds equation". Although there are many ways to derive this equation, as it's a direct simplification of Navier-Stokes momentum and continuity equation, it can be derived from this basis. The necessary conditions to use the Reynolds theory are that the surfaces must move relatively to each other with sufficient velocity for a load-carrying lubricating film to be generated and that surfaces must be inclined at some angle to each other. In case of surfaces are parallel the pressure field will not form the lubrication film able to carry the required load (even if there are cases in which the second condition is not so necessary). The way the lubricant generate itself a pressure field able to carry the required load is pretty simple, as it is described in Figure 2.1: assuming that the bottom surface is running and the top one is staying quiet (and assuming they're inclined at a certain angle relatively to each other), the moving surface drag the lubricant with his movement. The pressure gradient generated simply avoids that the entrant lubricant exceeds the exiting. The pressure gradient also does so that the velocity profile bends inwards at the entrance of the wedge and bends outwards at the exit. This difference create a pressure able to carry a certain load, which is the real purpose of the hydrodynamic lubrication. In case of a bearing, the wedge is curved (or wrapped around a shaft) as so to create a journal bearing, object of this work. In case of wedge remains planar, then a pad bearing is the result. This whole process of lubrication and load-carrying by an oil film can be totally described by mathematical rules and equations which require a large amount of computational work to solve. Modern engineering processes aim to a fast way to solve this equations, with a good accuracy of the results.

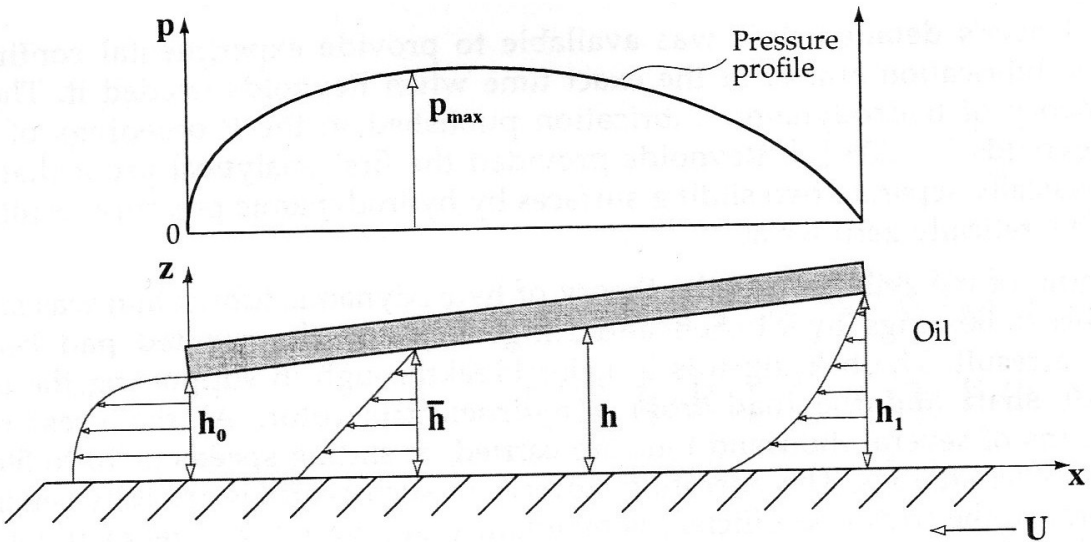


FIGURE 2.1. Principle of hydrodynamic pressure generation between non-parallel surfaces.

2.1 Reynolds Equation.

Simplifying assumptions.

As in every engineering applications, some simplifying assumptions have to be made in order to get a less complicated and easier solving problem. Real processes involved in hydrodynamic lubrications are too complicated to be studied considering every detail, implying an unacceptable load of operations in order to solve the equations. The factors involved in a real process sometimes make the exact description of the phenomenon such difficult, if not impossible. These simplifying assumptions are introduced with the purpose to get the mathematical description of the process, even though a good grade of accuracy is obviously pretty required.
First assumption is that the body forces are neglected. This is always a valid assumption, and it means that there are no exterior fields of forces acting on the bodies subject of the study. So, the only force acting in our study is hydrodynamic force, doing an exception in some cases referring to magnetohydrodynamic fluids and their applications (which won't be object of this study). Second assumption, always valid as well, pressure is constant through the film. This is because usually hydrodynamic films thickness is in the range of several micrometers, so we can totally neglect the pressure gradient in the oil film. Third assumption, always valid, no slip at the boundaries. The velocity of oil layer adjacent to the boundary is always as that of the boundaries. The remaining assumptions now concern the physic behaviour of the fluid used as a lubricant: first one is that the fluid has to be a Newtonian fluid. Exceptions can be made in case of polymeric oils, but these cases won't be discussed in this work. Flow has also to be laminar, except for large bearings. Plus, fluid inertia has to be neglected. This is valid if bearing is rotating at a low speed or carrying a massive load. If an elevate grade of accuracy is required, this assumption has to be considered false. Last two assumptions concern density of the fluid and his viscosity: they both have to be considered as constants, but with a difference. Fluid viscosity is constant only throughout the generated film, while density is considered constant all over, apart from the generated film. Obviously, density only can be considered constant in case of fluids not subject to great thermal expansion: this assumption is totally wrong in case of gases. Regarding the viscosity, this is generally a wrong assumption but necessary to simplify the load of calculations. In real processes, viscosity is definitely not constant.
As so to summarize, these are the simplifying assumptions made to get the mathematical description of the process of hydrodynamic lubrication:

- Body forces are neglected.
- Pressure constant through the film.
- No slip at the boundaries.
- Lubricant is a Newtonian fluid.
- Flow is laminar.
- Fluid inertia is neglected.
- Fluid density is constant.
- Viscosity is constant throughout the generated fluid film.

Equilibrium of an element.

The whole mathematical description (and the path to derive Reynolds equation) starts from the equilibrium of an element of fluid.

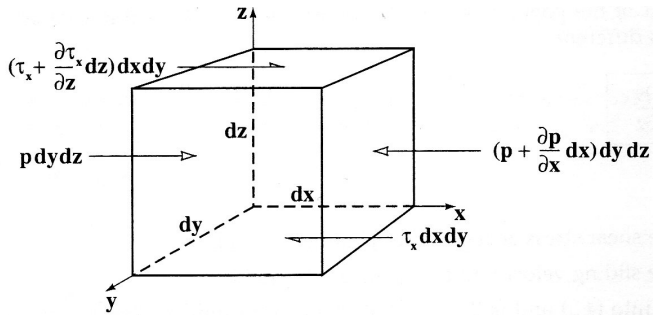


FIGURE 2.2 Equilibrium of an element

Considering a small element of fluid, from a hydrodynamic film (as shown in Figure 2.2), assuming for simplicity that forces initially act only in the x direction, forces acting to the left must be balance forces acting to the right. So,

$$pdydz + \left(\tau_x + \frac{\partial \tau_x}{\partial z} dz\right) dxdy = \left(p + \frac{\partial p}{\partial x} dx\right) dydz + \tau_x dxdy \qquad (2.1)$$

Which, with simplifying, becomes

$$\frac{\partial \tau_x}{\partial z} dxdydz = \frac{\partial p}{\partial x} dxdydz \qquad (2.2)$$

$$\frac{\partial \tau_x}{\partial z} = \frac{\partial p}{\partial x} \qquad (2.3)$$

Equation (2.3) strictly derives from the assumption that $dxdydz \neq 0$ (non zero volume) as so to divide both member of equation (2.2) by this value. Same exercise can be made in the y direction, obtaining basically the same equation written in other variables:

$$\frac{\partial \tau_y}{\partial z} = \frac{\partial p}{\partial y} \qquad (2.4)$$

Studying z directions, since according to the Assumption number 2 pressure gradient is zero through the oil film, we obtain

$$\frac{\partial p}{\partial z} = 0 \qquad (2.5)$$

By substituting formula of dynamic viscosity in the equation of the shear stress τ , we get an equation for shear stress depending on dynamic viscosity and shear rates

$$\tau_x = \eta \frac{u}{h} = \eta \frac{\partial u}{\partial z} \qquad (2.6)$$

Where τ_x is the shear stress acting in the x direction, measured in [Pa]. u is the velocity along the x axis. This substitution is possible even along y direction, obtaining the similar formula

$$\tau_y = \eta \frac{v}{h} = \eta \frac{\partial v}{\partial z} \qquad (2.7)$$

Where v is the velocity along y axis.
These equation lead us to equilibrium conditions for the forces acting in the x and y directions, simply by substituting equations (2.6) and (2.7) in original equation (2.4):

$$\frac{\partial p}{\partial x} = \frac{\partial}{\partial z}\left(\eta \frac{\partial u}{\partial z}\right) \qquad (2.8)$$

$$\frac{\partial p}{\partial y} = \frac{\partial}{\partial z}\left(\eta \frac{\partial v}{\partial z}\right) \qquad (2.9)$$

Integrating the above equations, using the Assumption number 8 in order to get an easier solution, is now possible for example by separation of variables. Assumption number 8 assures that viscosity is constant, so it is not a function of z , so integration is simple:

$$\frac{\partial p}{\partial x} \partial z = \partial\left(\eta \frac{\partial u}{\partial z}\right)$$

$$\frac{\partial p}{\partial x} z + C_1 = \eta \frac{\partial u}{\partial z}$$

$$\left(\frac{\partial p}{\partial x} z + C_1\right) \partial z = \eta \partial u$$

$$\frac{\partial p}{\partial x} \frac{z^2}{2} + C_1 z + C_2 = \eta u \qquad (2.10)$$

Using Assumption number 3, as there is no slip or velocity discontinuity between solid and liquid at the boundaries of the wedge, boundary conditions are set as

$u = U_2 \quad at \quad z = 0$
$u = U_1 \quad at \quad z = h$

We can evaluate the integration constants C_1 and C_2 simply by substituting the boundary conditions in equation (2.10)

$$C_1 = (U_1 - U_2)\frac{\eta}{h} - \frac{\partial p}{\partial x}\frac{h}{2}$$

$$C_2 = \eta U_2$$

Finally, substituting these constants and dividing and simplifying we get the expression for velocity along x axis:

$$\frac{\partial p}{\partial x}\frac{h^2}{2} + (U_1 - U_2)\frac{\eta z}{h} - \frac{\partial p}{\partial x}\frac{hz}{2} + \eta U_2 = \eta u$$

$$u = \left(\frac{z^2 - zh}{2\eta}\right)\frac{\partial p}{\partial x} + (U_1 - U_2)\frac{z}{h} + U_2 \qquad (2.11)$$

Velocity along y direction is obtained in a similar way:

$$v = \left(\frac{z^2 - zh}{2\eta}\right)\frac{\partial p}{\partial y} + (V_1 - V_2)\frac{z}{h} + V_2 \qquad (2.12)$$

Continuity of a flow in a column.

We now write the second necessary equation, in order to get the final Reynolds equation: the continuity of a flow in a column.

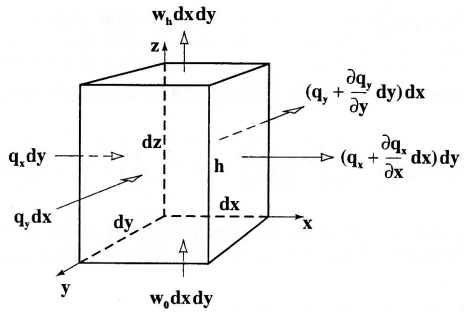


FIGURE 2.3. Continuity of a flow in a column

As shown in Figure 2.3, lubricant flows into the column horizontally at rates q_x and q_y and out of the column at rates of $\left(q_x + \frac{\partial q_x}{\partial x}dx\right)$ and $\left(q_y + \frac{\partial q_y}{\partial y}dy\right)$ per unit length and width. Concerning the vertical direction, lubricant flows at the rate of $w_0 dxdy$ and $w_h dxdy$ respectively into the column and out of the column. w_0 is the velocity of the bottom of the column, while w_h is the velocity at which the top of the column moves up. Remembering Assumption number 7, and so using a constant density of the lubricant, we

can simply write the equation of continuity by satisfying the principle of continuity itself: influx must be equal to efflux from a control volume under steady conditions.

$$q_x dy + q_y dx + w_0 dxdy = \left(q_x + \frac{\partial q_x}{\partial x} dx\right) dy + \left(q_y + \frac{\partial q_y}{\partial y} dy\right) dx + w_h dxdy \qquad (2.13)$$

Which can be simplified in:

$$\frac{\partial q_x}{\partial x} dxdy + \frac{\partial q_y}{\partial y} dxdy + (w_h - w_0) dxdy = 0 \qquad (2.14)$$

And, as $dxdy \neq 0$, we get

$$\frac{\partial q_x}{\partial x} + \frac{\partial q_y}{\partial y} + (w_h - w_0) = 0 \qquad (2.15)$$

Last equation obtained (2.15) is the equation of continuity of a flow in a column.
The two variables q_x and q_y, which are the flow rates per unit of length, can be evaluated simply by integrating the lubricant profile velocity over the film thickness, so

$$q_x = \int_0^h u dz \qquad (2.16)$$

$$q_y = \int_0^h v dz \qquad (2.17)$$

Substituting the two profiles u and v previously obtained, we get

$$\left(\frac{z^3}{3} - \frac{z^2 h}{2}\right) \frac{\partial p}{2\eta \partial x} + (U_1 - U_2) \frac{z^2}{2h} + U_2 z \Big|_0^h$$

$$q_x = -\frac{h^3}{12\eta} \frac{\partial p}{\partial x} + (U_1 + U_2) \frac{h}{2} \qquad (2.18)$$

Similarly, the flow rate in y direction is obtained by substituting v in equation (2.17) from equation (2.12):

$$q_y = -\frac{h^3}{12\eta} \frac{\partial p}{\partial y} + (V_1 + V_2) \frac{h}{2} \qquad (2.19)$$

Substituting equations (2.18) and (2.19) into the continuity of the flow equation

$$\frac{\partial}{\partial x}\left[-\frac{h^3}{12\eta} \frac{\partial p}{\partial x} + (U_1 + U_2) \frac{h}{2}\right] + \frac{\partial}{\partial y}\left[-\frac{h^3}{12\eta} \frac{\partial p}{\partial y} + (V_1 + V_2) \frac{h}{2}\right] + (w_h - w_0) = 0 \qquad (2.20)$$

Defining $U = U_1 + U_2$ and $V = V_1 + V_2$ and assuming that there are no local variation in surface velocity in the x and y directions, we get

$$\frac{\partial}{\partial x}\left(\frac{h^3}{\eta}\frac{\partial p}{\partial x}\right) + \frac{\partial}{\partial y}\left(\frac{h^3}{\eta}\frac{\partial p}{\partial y}\right) = 6\left(U\frac{dh}{dx} + V\frac{dh}{dy}\right) + 12(w_h - w_0) \tag{2.21}$$

Simplifications to the Reynolds equation.

Since Reynolds equation we just get is too complex for practical and engineering applications, it need to be simplified before it can be easily used. Since the subject of this work will be a journal bearing, the most important regarding this study is the *Unidirectional Velocity Approximation.*

As it's always possible to choose axes as so that one of the velocities is equal to zero, we can assume $V = 0$. This is exactly the case of a journal bearing, in which the bearing itself slides along a rotating shaft.

Assuming $V = 0$, equation (2.21) can be rewritten as follows:

$$\frac{\partial}{\partial x}\left(\frac{h^3}{\eta}\frac{\partial p}{\partial x}\right) + \frac{\partial}{\partial y}\left(\frac{h^3}{\eta}\frac{\partial p}{\partial y}\right) = 6U\frac{dh}{dx} + 12(w_h - w_0) \tag{2.22}$$

This equation, rewritten in dimensionless parameters and in a polar set of coordinates, will be the object of the whole following study.

Chapter 3

Variational Approach.

The generalized partial differential equation form of Reynolds equation for journal bearings is usually written as:

$$\frac{\partial}{\partial x}\left(\frac{\rho h^3}{\mu}\frac{\partial p}{\partial x}\right)+\frac{\partial}{\partial z}\left(\frac{\rho h^3}{\mu}\frac{\partial p}{\partial z}\right)=6U\frac{\partial(\rho h)}{\partial x}+12\rho V_0$$

3.1 Dimensionless Reynolds Equation

Basic equation of Reynolds equation for journal bearing film thickness can be written as:

$$\frac{1}{R^2}\frac{\partial}{\partial\theta}\left(\frac{H^3}{12\mu}\frac{\partial P}{\partial\theta}\right)+\frac{\partial}{\partial Z}\left(\frac{H^3}{12\mu}\frac{\partial P}{\partial Z}\right)=-\frac{\omega}{2}\frac{\partial H}{\partial\theta}-\frac{\partial H}{\partial T} \qquad (3.1)$$

Where film thickness is

$$H = C_r - E\cos(\theta-\beta) = C_r - X\cos\theta - Y\sin\theta \qquad (3.2)$$

Assumptions made to establish the Reynolds equation are:

i. Film thickness over its length is very small: therefore, the direction of film pressure along the film thickness can be considered the same.
ii. Oil flow is laminar flow, Reynolds number is small, and vortex and turbulence do not appear in the film.
iii. Oil is isotropic; viscosity in the direction of film is constant.
iv. No slip between oil and journal bearing surface.
v. Lubricants follow Newton's law of viscosity, namely, shear stress and shear rate are proportional.
vi. Excluding oil inertia.

Considering dimensionless parameters:

$$p = \frac{P}{\frac{6\mu\omega R_j^2}{C_r^2}}$$

$$\zeta = \frac{Z}{R_j}$$

$$h = \frac{H}{C_r}$$

$$\tau = \omega t$$

With Reynolds boundary conditions, we get dimensionless Reynolds equation:

$L(p) = f$ Within Ω^+

$p = 0, \frac{\partial p}{\partial n} = 0$ On Γ^0, where n is the normal direction of Γ^0 (3.3)

$p = 0$ In Ω^0 and on Γ

Where

$$L(p) = \frac{\partial}{\partial\theta}\left(h^3\frac{\partial p}{\partial\theta}\right) + \frac{\partial}{\partial\zeta}\left(h^3\frac{\partial}{\partial\zeta}\right) \quad (3.4)$$

$$f = -\frac{\partial h}{\partial\theta} - 2\frac{\partial h}{\partial t} = (2\hat{x} + y)\cos\theta + (2\hat{y} - x)\sin\theta \quad (3.5)$$

$$h = 1 - x\cos\theta - y\sin\theta \quad (3.6)$$

Reynolds equation (3.3) is a binary variable second-order non-homogeneous partial differential equation. Usually, finite difference or finite element method together with setting zero algorithm are used to directly get the solution. This way of solving leads to relatively accurate results, not justified by the massive computational need of the machine. Besides, in terms of non-linear rotor-bearing system, analysis has major limitations. For a non-linear system, solving the dynamic response of the system needs to repeat each step of the calculation of non-linear oil film force. If we use large scale numerical algorithm, the system dynamic response calculation is almost the amount of calculation occupied by oil-film force. Even if computer's speed has been greatly improved, the demand of this calculation is difficult to tolerate. Essentially, these are the reasons why variational approach is preferable to a normal approach (FEM).

3.2 Theoretical foundation of variational approach.

Non-linear dynamic analysis of rotating machinery required integration of oil film pressure for the distribution of bearing oil film force. When we are not demanding the accuracy of the pressure distribution point by point, but the overall effect of oil-film force with sufficient accuracy, variational method is exactly an effective way to solve this problem. However, if the oil film rupture occurs, the traditional variational principle is not suitable so need to be amended. Rohde [15] as early as 1975, gave the traditional variation principle to amend the formula to accommodate the Reynolds boundary conditions at the film rupture [16], but the proof of equivalence given was only for one-dimensional Reynolds equation. Kinderlehrer had given a full film pressure to satisfy the variational inequality equation, Zheng and Hasebe [17] directly applied this variational inequality equation to the actual bearings without proves, and used complementary finite element method to solve the elliptical bearing non linear oil film force. Subsequently, Zheng Tie-Sheng [18] to make a systematic classification and certification used free boundary value theory that non-steady state Reynolds equation in Reynolds boundary conditions are equivalent to the variational principle in order to provide a theoretical basis for non-steady approximate analytic solution of oil film forces.

From the quoted literature, solution of Reynolds equation under the Reynolds boundary conditions (3.3) is equivalent to solving the following propositions:

Proposition 1. [4]

Find, boundary at $\Gamma = 0$, fully smooth pressure function p , to fulfil

$$\begin{cases} p \geq 0, & in\ \Omega \\ p.(L(p) - f) = 0 \end{cases} \qquad (3.7)$$

Proposition 2. [4]

Seeking convex external problem on the functional

$$J(p) = \min_{q \in K} J(q) \qquad (3.8)$$

Proposition 3. ***[4]***

Variation inequality

$$p \in K, we\ get\ a(p, q-p) \geq f(q-p), \forall q \in K \tag{3.9}$$

Proposition 4. ***[4]***

Variation inequality

$$p \in K, we\ get\ a(p, q) \geq f(q) \forall q \in K \tag{3.10}$$

Proposition 5. ***[4]***

Variation equality

$$p \in K, we\ get\ a(p, p) = f(p) \tag{3.11}$$

Where,

$$J(q) = \frac{1}{2} a(q, q) - f(q) \tag{3.12}$$

$$a(p, q) = \iint_{\Omega} h^3 \left(\frac{\partial p}{\partial \theta} \frac{\partial q}{\partial \theta} + \frac{\partial p}{\partial \zeta} \frac{\partial q}{\partial \zeta} \right) d\theta d\zeta \tag{3.13}$$

$$f(q) = \iint_{\Omega} (f . q) d\theta d\zeta \tag{3.14}$$

$$K = \{ q \in H_0^1(\Omega) | q \geq 0\ in\ \Omega \} \tag{3.15}$$

H_0^1 - On fluid film Ω of Sobolev space, satisfying the zero boundary condition and continuous first derivative

K - In $H_0^1(\Omega)$ as non-negative function, clearly K is a convex set of $H_0^1(\Omega)$, and it's close to the origin

q - In $H_0^1(\Omega)$, pressure distribution of a test function

Obviously, from equation (3.13) and (3.14), we can see $a(p,q)$ is symmetric, bilinear and positive definite; $f(q)$ is linear.

Looking into finite element method based on variation inequalities, the propositions below are the proof of the equivalence.

1°: Reynolds equation (3.3) and proposition 1 equivalent.

Equivalence between the two equations is obvious. Because from the second formula of equation (3.7), the valve problem from the free boundary conditions stated that: when $p > 0$ (withinΩ^+), $L(p) - f = 0$, in the first formula in equation (3.3); instead, equation (3.3) also implied formula 2 from equation (3.7). Also, Ω^0 is non-empty, by the smooth pressure function p, it only can be within Ω^+ that come within the smooth transition zero value within the Ω^0, therefore there must be $p = 0$ and $\frac{\partial p}{\partial n} = 0$ on the border between the two Γ^0.

2°: Proposition 2-5 equivalents.

Wèll follow the path from proposition 2 to 3 to 4 to 5 to 3 to 2.

Assume p is the solution based on proposition 2, $p \in K$ take an arbitrary $q \in K$, for any $t \in [0,1]$, because K is a convex set, of course there are $p + t(q-p) \in K$. Also by minimum of $J(p)$, and taking note of $a(p,q)$ symmetry, bilinear and linearity of $f(q)$, we have

$$J(p) = \frac{1}{2}a(p,p) - f(p) \leq J(p + t(q-p))$$

$$= \frac{1}{2}a(p + t(q-p), p + t(q-p)) - f(p + t(q-p))$$

$$= \frac{1}{2}a(p,p) + \frac{t^2}{2}a(q-p, q-p) + ta(p, q-p) - f(p) - tf(q-p)$$

Finally we got

$$\frac{t}{2}a(q-p, q-p) + a(p, q-p) - f(q-p) \geq 0$$

Using $t \to 0$ in proposition 3.

And because K is the vertex at the origin of the closed convex cone, $p, q \in K \Rightarrow p + q \in K$, replacing q with $p + q$ in proposition 3 gives us proposition 4.

In order to prove proposition 5, we only have to make $0 = q \in K$ in proposition 3, we get $a(p,p) \geq -f(p)$, that is

$$a(p,p) \leq f(p) \qquad (3.16)$$

Again in proposition 4, make $q = p$, we get

$$a(p,p) \geq f(p) \qquad (3.17)$$

This two equations implied (3.11), which is proposition 5. Instead, when propositions 4, 5 set, then (3.10) minus (3.11), we get (3.9), namely proposition 3.

Lastly, if proposition 3 is valid, we note that

$$\frac{1}{2}[a(q,q) - a(p,p)] = \frac{1}{2}[a(p+q-p, p+q-p) - a(p,p)]$$

$$= a(p, q-p) + \frac{1}{2}a(q-p, q-p) \geq a(p, q-p)$$

The above formula used the positive definite of $a(\cdot,\cdot)$. Putting (3.9) into the above formula, we have

$$\frac{1}{2}[a(q,q) - a(p,p)] \geq f(q-p) = f(q) - f(p)$$

Namely

$$\frac{1}{2}a(q,q) - f(q) \geq \frac{1}{2}a(p,p) - f(p)$$

$$J(q) \geq J(p)$$

Therefore, p is a functional solution to equation (3.8), namely proposition 2.

3°: Proposition 4,5 with proposition 1.

Set $p, q \in K \subset H_0^1(\Omega)$, according to Green theorem, we get

$$\iint_\Omega q \cdot L(p) d\theta d\zeta = -\iint_\Omega q \cdot \left[\frac{\partial}{\partial\theta}\left(h^3\frac{\partial p}{\partial\theta}\right) + \frac{\partial}{\partial\zeta}\left(h^3\frac{\partial p}{\partial\zeta}\right)\right] d\theta d\zeta$$

$$= \iint_\Omega h^3\left(\frac{\partial p}{\partial\theta}\frac{\partial q}{\partial\theta} + \frac{\partial p}{\partial\zeta}\frac{\partial q}{\partial\zeta}\right) d\theta d\zeta - \int_\Gamma h^3 q\frac{\partial p}{\partial n} d\theta d\zeta$$

Note that at $\Gamma, q = 0$, the last term is equal to zero. By the definition of $a(p, q)$ in (3.13), the above formula become

$$a(p, q) = \iint_\Omega q \cdot L(p) d\theta d \qquad , \forall q \in K \tag{3.18}$$

Minus $f(q)$ (3.14) from the last equation, we have

$$a(p, q) - f(q) = \iint_\Omega q \cdot (L(p) - f) d\theta d\zeta \quad , \forall q \in K \tag{3.19}$$

In particular, when replacing $q = p$, we get

$$a(p, p) - f(p) = \iint_\Omega p \cdot (L(p) - f) d\theta d\zeta \tag{3.20}$$

Thus if p is the solution of the variational inequality (3.10) then equation (3.19) will gives

$$\iint_{\Omega} q \cdot (L(p) - f) d\theta d\zeta \geq 0 \quad , \forall q \in K \tag{3.21}$$

Also because in the above equation $0 \leq q \in K$ is arbitrary, thus

$$L(p) - f \geq 0 \tag{3.22}$$

Equations (3.11) and (3.20) give us

$$\iint_{\Omega} p \cdot (L(p) - f) d\theta d\zeta = 0 \tag{3.23}$$

Then p is solution of variational inequality (3.10), $0 \leq p \in K$, thus we have

$$p \cdot (L(p) - f) = 0 \tag{3.24}$$

Namely the free boundary value problem (3.7) in terms of complementary.

However, if p is the solution of free boundary problem (3.7), then the right hand side of equation (3.20) is zero, so there is

$a(p, p) - f(p) = 0, \qquad p \in K \subset H_0^1(\Omega)$

p is the variational equation solution of proposition 5.

End of the proof.

3.3 Complement of one-dimensional variational inequality and the Finite Elements Method.

The variational inequality approach of this subchapter is from the proposition 4 of the previous section which is re-described below:

solving the Reynolds equation (3.3) under Reynolds boundary conditions is equivalent to solving the following two-dimensional variational inequality:

$$seek\ p \in K\ to\ get\ a(p,q) \geq f(q)\ ,\forall q \in K \tag{3.25}$$

In which

$$K = \{q \in H_0^1(\Omega) | q \geq 0\ in\ \Omega\} \tag{3.26}$$

$$a(p,q) = \iint_\Omega h^3 \left(\frac{\partial p}{\partial \theta}\frac{\partial q}{\partial \theta} + \frac{\partial p}{\partial \zeta}\frac{\partial q}{\partial \zeta}\right) d\theta d\zeta \tag{3.27}$$

$$f(q) = \iint_\Omega (f \cdot q) d\theta d\zeta \tag{3.28}$$

$H_0^1(\Omega)$ is in Sobolev space Ω , K is the $H_0^1(\Omega)$ function in a non-negative set of closed convex lubrication domain $\Omega = \theta \times \zeta = [0,\gamma] \times [\lambda,\lambda]$. Oil film thickness h and the variables expression f can be written as

$$h = 1 - x\cos\theta - y\sin\theta = 1 - e\cos(\theta - \beta) \tag{3.29}$$

$$f = b_c\cos\theta + b_s\sin\theta = A\sin(\theta - \alpha) \tag{3.30}$$

In which, $b_c = 2\dot{x} + y, b_s = 2\dot{y} - x, A = \sqrt{b_c^2 + b_s^2}, \alpha = -sgn(b_s)\arccos\left(\frac{b_c}{A}\right)$

The variational inequality is equivalent to the free boundary problem:

$$\begin{cases} p \geq 0 \ in \ \Omega; p = 0 \ in \ \partial\Omega \\ \frac{\partial}{\partial\theta}\left(h^3\frac{\partial p}{\partial\theta}\right) + \frac{\partial}{\partial\zeta}\left(h^3\frac{\partial p}{\partial\zeta}\right) \end{cases} \tag{3.31}$$

That p only in the Ω sub-domain Ω^+ (corresponding to $p > 0$ region) fulfills Reynolds equation, in the sub-domain $\Omega^0 = \Omega - \Omega^+$ cavitations occurred, at this point $p = 0$. Free boundary Γ is at the Ω^+ and Ω^0 boundary, that is the edge cavitations, not known in advance, solved from the variational inequality (3.25) with p at the same time. This is the non-linear geometric characteristic problem.

The finite element method calculation of the bearing oil film domain Ω for the finite function and pressure test function is written as:

$$p = \sum p_i\phi_i(\theta,\zeta); \quad q = \sum q_j\phi_j(\theta,\zeta) \qquad (1, j = 1,2,3,\dots,n) \tag{3.32}$$

In which, $\phi(\theta,\zeta)$ is the overall interpolation function, p_i for the node pressure, q_j for the pressure test function, n is the total number of nodes. Variational inequality (3.25) discrete into:

$$Seek\ p \geq 0, we\ get\ q^T Ap \geq q^T f, \qquad \forall q \in K_n \tag{3.33}$$

In which, $K_n = \{q \in R^n; q_i \geq 0\ (i = 1,2,\dots,n)\}$, that K_n is the n-dimensional set of non-negative quantum, therefore $q \in K_n$ is abbreviated as $q \geq 0$; matrices A and the load quantity f element were

$$a_{ij} = a(\phi_i, \phi_j), \qquad f_i = f(\phi_i) \quad (i = 1,2,3,\dots,n)$$

Functional $a(\cdot,\cdot), f(\cdot)$ are defined from the variational inequality (3.25).

In order to solve the inequality (3.33), we need to convert it into a linear complementary problem: seek $p, e \geq 0$ to fulfill

$$Ap - f = e \tag{3.34}$$

$$e^T p = 0 \tag{3.35}$$

Try to prove as follows:

Because $p \geq 0$, from the discrete equation (3.33), there is

$$p^T A p \geq p^T f \tag{3.36}$$

By proposition 3 of section 3.2, that equation (3.9) can get another form of discrete inequality:

$$Seek\ p \geq 0\ to\ get\ (q-p)^T A p \geq (q-p)^T f, \qquad \forall q \geq 0 \tag{3.37}$$

Putting $q = 0$ into the above equation, we have

$$p^T A p \geq -p^T f \Rightarrow p^T A p \leq p^T f \tag{3.38}$$

Combining equations (3.36) and (3.38), the two equations can become equation (3.34) and (3.35). In order to prove the $e \geq 0$ in the complementary problem, we only have to take note of the matrix $E_{n \times n}$ where each column vector $E_i \geq 0$, thus by substituting into equation (3.35), we get:

$$e^T E_i = e_i \geq 0 \quad (i = 1,2,3,\dots,n) \tag{3.39}$$

That $e \geq 0$, in which $e = [e_1, e_2, e_3, \dots, e_n]^T$. Incidentally, from $p \geq 0, e \geq 0$ and $e^T p = 0$ we can see that, with $i = 1,2,3,\dots,n, p_i > 0$ and $e_i > 0$ cannot exist simultaneously, that is

1. If $p_i > 0$, then $e_i = 0, f_i > 0$
2. If $p_i = 0$, then $e_i \geq 0, f_i \leq 0 \ (i = 1,2,3,\dots,n)$ (3.40)

End of the proof.

Literature [19] proposed, from the constraints in equation (3.35), can be partitioned to

$$\begin{bmatrix} A_{aa} & A_{ab} \\ A_{ba} & A_{bb} \end{bmatrix} \begin{Bmatrix} P_a \\ 0 \end{Bmatrix} - \begin{Bmatrix} f_a \\ f_b \end{Bmatrix} = \begin{Bmatrix} 0 \\ e_b \end{Bmatrix} \tag{3.41}$$

In which $p_a > 0$ and $e_b \geq 0$, iteratively adjusting this sub-block forms can lead to the solution of the complementary problem. This solution is accurate when the iteration converge.

3.4 Complementary Finite Elements Method.

From the two-dimensional finite method, lubricating oil film is approximately a rectangular field Ω. Although one of the film rupture boundary curves is slightly bent, but its derivative on the pressure are zero. It straight error is caused by the

$$p = (\theta, \zeta) = w(\zeta, \theta) p^T l(\theta), \qquad q(\theta, \zeta) = w(\zeta, \theta) q^T l(\theta) \tag{3.42}$$

Where $w(\zeta, \theta)$ is weight function, whenever

$$w(\zeta, \theta) = h^{-m} g(\zeta) \tag{3.43}$$

$$g(\zeta) = \cosh(k\lambda) - \cosh(k\zeta) \tag{3.44}$$

Weighting function to take the axial pressure distribution for the hyperbolic cosine function (parameter k is given back), because the smaller the film thickness, the higher the pressure, weighting function has to take the pressure distribution and film thickness is inversely proportional to the m-power, this has accelerated the interpolation. The examples later will be devoted to the results of m .

$l(\theta) = \{l_1(\theta), l_2(\theta), l_3(\theta), \ldots, l_n(\theta)\}^T$ is the overall interpolation function vector, from the $n - \theta$ direction of sub-one-dimensional linear interpolation function of the composition of $p = \{p_1, p_2, p_3, \ldots, p_n\}^T$ and $q = \{q_1, q_2, q_3, \ldots, q_n\}^T$ for the undetermined coefficient vector.

The following are given as straight line edge cavitations axial pressure distributions for the hyperbolic cosine function that making the edge cavitations straight, therefore the distribution of film pressure test functions can be separated from variables q

$$q(\theta,\zeta) = g(\zeta)r(\theta) \tag{3.45}$$

The distribution will be on a trial-type function of q in the previous section substituted into the functional extreme problem in proposition 2 of equation (3.12).

We get

$$J(q) = \frac{1}{2}(c_1 d_1 + c_2 d_2) - c_3 d_3 \tag{3.46}$$

Which

$$\begin{cases} c_1 = \int_{-\lambda}^{\lambda} g^2(\zeta)d\zeta \\ c_2 = \int_{-\lambda}^{\lambda} g'^2(\zeta)d\zeta \\ c_3 = \int_{-\lambda}^{\lambda} g(\zeta)d\zeta \end{cases} \tag{3.47}$$

$$\begin{cases} d_1 = \int_0^{\gamma} h^3(\theta)r'^2(\theta)d\theta \\ d_2 = \int_0^{\gamma} h^3(\theta)r^2(\theta)d\theta \\ d_3 = \int_0^{\gamma} h^3(\theta)r(\theta)d\theta \end{cases} \tag{3.48}$$

The function $f(\theta)$ is the function f in equation (3.30).

As we can see from the above equation, if we assume that function $r(\theta)$ is known, and d_1, d_2, d_3 is only decided by $r(\theta)$, therefore they can be considered as known parameters. At this point, solving the equation (3.8) become solving the Euler-Lagrange equation

$$\begin{cases} d_2 g''(\zeta) - d_1 g(\zeta) = -d_3 \\ g(-\lambda) = g(\lambda) = 0 \end{cases} \tag{3.49}$$

Its solution is

$$g(\zeta) = \cosh(k\lambda) - \cosh(k\zeta) \tag{3.50}$$

In the iteration parameter

$$k = \sqrt{\frac{d_1}{d_2}} \tag{3.51}$$

From the assumption for the film of straight line at the side shown, to take the hyperbolic cosine function for the axial pressure is accurate.

Because of the weighting function $w(\zeta, \theta) > 0$, when using the linear interpolation $l(\theta) > 0$, so when $p, q > 0$, (3.50) distribution function of pressure p and test functions q are non-negative. Substituting equation (3.42) into the variational inequality (3.25), that was after the one-dimensional discrete variational inequality.

$$Seek\ p \geq 0, we\ get\ q^T K p \geq q^T f, \qquad \forall q \geq 0 \tag{3.52}$$

And the complimentary problem

$$Seek\ p \geq 0, e \geq 0, we\ get\ Kp - f = e, at\ e^T p = 0 \tag{3.53}$$

Now the coefficient matrix K and the vector f are a weighted one-dimensional finite element results.

$$K = c_1 K_1 + c_2 K_2 \tag{3.54}$$

$$f = c_3 (b_c f_c + b_s f_s) \tag{3.55}$$

$$\begin{cases} K_1 = \int_0^\gamma h^{3-2m} \left[l'(\theta) - \frac{m h'(\theta)}{h(\theta) l(\theta)} \right] \left[l'(\theta) - \frac{m h'(\theta)}{h(\theta) l(\theta)} \right]^T d\theta \\ K_2 = \int_0^\gamma h^{3-2m} l(\theta) l^T(\theta) d\theta \end{cases} \tag{3.56}$$

$$\begin{cases} f_c = \int_0^\gamma h^{-m} l(\theta) \cos\theta \, d\theta \\ f_s = \int_0^\gamma h^{-m} l(\theta) \sin\theta \, d\theta \end{cases} \tag{3.57}$$

$$\begin{cases} c_1 = \int_{-\lambda}^{\lambda} g^2(\zeta) d\zeta = 2\lambda + \lambda \cosh(2k\lambda) - \frac{3}{2k} \sinh(2k\lambda) \\ c_2 = \int_{-\lambda}^{\lambda} g'^2(\zeta) d\zeta = \frac{k}{2} [\sinh(2k\lambda) - 2k\lambda] \\ c_3 = \int_{-\lambda}^{\lambda} g(\zeta) d\zeta = \frac{2}{k} [k\lambda \cosh(k\lambda) - \sinh(k\lambda)] \end{cases} \tag{3.58}$$

In which k is the iteration parameter

$$k = \sqrt{\frac{d_1}{d_2}} \tag{3.59}$$

Now,

$$d_1 = \int_0^\gamma h^3 \left[h^{-m}(\theta) p^T l(\theta)\right]'^2 d\theta = p^T K_1 p$$

$$d_2 = \int_0^\gamma h^3 \left[h^{-m}(\theta) p^T l(\theta)\right]^2 d\theta = p^T K_2 p$$

We can see that, given the initial iteration k ,calculated c_1, c_2, c_3, and then substitute into the equation (3.54) and (3.55) can range from one-dimensional variational inequality (3.52) to get the solution p, q , then by using equation (3.59) to get a new value for k . Repeat the iteration until satisfactory accuracy is achieved, and thus the pressure distribution function p is obtained. Then, from the following equations, the oil film force components of the pad under the local coordinate system are calculated.

$$f_x = -\iint_\Omega p(\theta, \zeta) \cos\theta \, d\theta \, d\zeta = -\int_{-\lambda}^{\lambda} g(\zeta) d\zeta \int_0^\gamma h^{-m} r(\theta) \cos\theta \, d\theta$$

$$f_y = -\iint_\Omega p(\theta, \zeta) \sin\theta \, d\theta \, d\zeta = -\int_{-\lambda}^{\lambda} g(\zeta) d\zeta \int_0^\gamma h^{-m} r(\theta) \sin\theta \, d\theta \tag{3.60}$$

3.5 Complementary problem solving.

Now we will solve the previous section ranging from one-dimensional variational equation (3.52) and the complementary problem (3.53). We will solve the following sub-block.

$$\begin{bmatrix} K_{aa} & K_{ab} \\ K_{ba} & K_{bb} \end{bmatrix} \begin{Bmatrix} P_a \\ 0 \end{Bmatrix} - \begin{Bmatrix} f_a \\ f_b \end{Bmatrix} = \begin{Bmatrix} 0 \\ e_b \end{Bmatrix} \tag{3.61}$$

Based on the K as symmetric, definite, the characteristics of the tridiagonal matrix, adopted by the amendatory law to solve the equation, concrete processional operations as it follows:

First of all, remember vector $f = (f_1, f_2, f_3, \ldots, f_n)^T$, tridiagonal symmetric matrix K of the main diagonal elements $(k_1^a, k_2^a, k_3^a, \ldots, k_n^a)$, sub-diagonal elements $(k_1^b, k_2^b, k_3^b, \ldots, k_n^b)$. Matrix decomposition of K for the next two pairs of diagonal matrices L and U, namely

$$K = LU$$

Which

$$L = \begin{bmatrix} r_1 & & & & \\ k_1^b & r_2 & & & \\ & \ddots & \ddots & & \\ & & k_{n-2}^b & r_{n-1} & \\ & & & k_n^b & r_n \end{bmatrix} \quad U = \begin{bmatrix} 1 & s_1 & & & \\ & 1 & s_2 & & \\ & & \ddots & \ddots & \\ & & & 1 & s_{n-1} \\ & & & & 1 \end{bmatrix} \tag{3.63}$$

The relationship between $r_i, s_i,$ and k_i^a, k_i^b is

$$r_1 = k_1^a$$

$$r_i = k_i^a - k_{i-1}^b s_{i-1}, s_i = \frac{k_i^b}{r_i} \qquad (i = 1,2,3,\ldots,n) \tag{3.64}$$

Then, the introduction of intermediate variable $y = Up = \{y_1, y_2, y_3, \ldots, y_n\}^T$. Amendatory Act concept steps are as follows

Step 1: for $i = 1,2,3, \ldots, n$ the following amendment to the "chase" process

(1) $r_i = k_i^a - k_{i-1}^b s_{i-1}, s_i = \frac{k_i^b}{r_i}, e_i = k_i^a f_{i-1} y_{i-1} - f_i$ (if subscript is 0, calculation is not needed);

(2) When $e_i < 0$, $y_i = -\frac{e_i}{r_i}$, otherwise $l = i - 1$ complete the "chase" process, followed by step 2.

Step 2: $p_1 = y_1$, putting $j = l - 1, l - 2, \ldots, 1$ to "chase" the process: $p_j = y_j - s_j p_{j+1}$.

Which, the former l equation is solved using the "chasing" process

$$K_{aa} = L_a U_a, L_a y_a = f_a, U_a p_a = y_a \qquad (3.65)$$

Amendatory Act to ensure that

$$y_a = \{y_1, y_2, y_3, \ldots, y_l\}^T \geq 0 \qquad (3.66)$$

Also, because of the continuous airway pressure zone unity, there must be

$$p_a = \{p_1, p_2, p_3, \ldots, p_l\}^T \geq 0 \qquad (3.67)$$

After the $n - 1$ equation become

$$e_b = \{e_1, e_2, e_3, \ldots, e_n\}^T = \{k_l^b p_l - f_{l+1}, -f_{l+2}, -f_{l+3}, \ldots, -f_n\}^T \qquad (3.68)$$

Fixing the chasing process ensure that $e_{l+1} = k_l^6 f_l y_l - f_{l+1} > 0$, from equation (3.40), we can see a necessary condition for film rupture zone is $f_l \leq 0$, then

$$\begin{Bmatrix} 0 \\ e_b \end{Bmatrix} \geq 0 \qquad (3.69)$$

In summary, the amendatory law does satisfy the complementary problems in the solution of equation (3.53).

The essence of the Fixed Amendatory Act is, in the process of the "chase", once we get $y_l < 0\ (e_l > 0)$ the "chase" process stops.

Starting from the process $l = l - 1$. This eliminates the need for iterative and automatically determines the boundary conditions of the Reynolds (which is equivalent to the complementarily problem), and just have to "chase" the pressure zone, all matrix operation are carried out only in positive-pressure zone. Therefore, the more the cavitations at front side, the lesser the computation, this is where the advantage of this algorithm lies.

3.6 Oil film forces and Jacobian matrix of the joint solution.

As we can see from the above process, amendatory law is essential for equation (3.63) where sub-blocks forms the upper portion of the operation, in solving the following algebraic equation

$$K_{aa}p_a = f_a \qquad (3.70)$$

Film rupture point m only needs to be dynamically determine the solution process, to fulfil the Reynolds boundary conditions.

To facilitate the writing, we omit the subscript; the above equation will be abbreviated as

$$Kp = f \qquad (3.71)$$

After the vector p is obtained from the "chasing" process, taking the iteration parameter from the equation (3.59)

$$k = \sqrt{\frac{d_1}{d_2}} = \sqrt{\frac{p^T K_1 p}{p^T K_2 p}} \qquad (3.72)$$

Substitute using equation (2.57) and the third formula in equation (2.58) into (2.60), noting that we only calculate the oil-film force in the positive zone, local coordinate system was under the transient oil-film force

$$f_x = -c_3 f_c^T p_a;\ f_y = -c_3 f_s^T p_a \qquad (3.73)$$

From the above equation, under the oil-film local coordinate system of the Jacobian matrix of f_x, f_y can be expressed as

$$\begin{cases} J_1 = \begin{bmatrix} \frac{\partial f_x}{\partial x} & \frac{\partial f_x}{\partial y} \\ \frac{\partial f_y}{\partial x} & \frac{\partial f_y}{\partial y} \end{bmatrix} = -c_s \begin{bmatrix} f_c^T \frac{\partial n}{\partial x} & f_c^T \frac{\partial n}{\partial y} \\ f_s^T \frac{\partial n}{\partial x} & f_s^T \frac{\partial n}{\partial y} \end{bmatrix} \\ J_2 = \begin{bmatrix} \frac{\partial f_x}{\partial \dot{x}} & \frac{\partial f_x}{\partial \dot{y}} \\ \frac{\partial f_y}{\partial \dot{x}} & \frac{\partial f_y}{\partial \dot{y}} \end{bmatrix} = -c_s \begin{bmatrix} f_c^T \frac{\partial n}{\partial \dot{x}} & f_c^T \frac{\partial n}{\partial \dot{y}} \\ f_s^T \frac{\partial n}{\partial \dot{x}} & f_s^T \frac{\partial n}{\partial \dot{y}} \end{bmatrix} \end{cases} \quad (3.74)$$

Of equation (3.71), taking the partial derivative of x, y and $\dot{x}, \dot{y}$ respectively, taking into account the definition of vector f in equation (3.55), there is

$$\begin{cases} K\frac{\partial p}{\partial x} + \frac{\partial K}{\partial x}p = -c_s f_s \\ K\frac{\partial p}{\partial y} + \frac{\partial K}{\partial y}p = c_s f_c \\ K\frac{\partial p}{\partial \dot{x}} = 2c_s f_c \\ K\frac{\partial p}{\partial \dot{y}} = 2c_s f_s \end{cases} \quad (3.75)$$

From the equation (3.54) and (3.56) earlier, we know that k is related to x, y , therefore $\frac{\partial K}{\partial x}, \frac{\partial K}{\partial x}$ from the above equation are referred as regular matrix, similarly having the tridiagonal form.

In this case, the two equations in the above formula, the left side of the second equation can be shifted to the right into the right hand side vector, with the latter 2 equations can be used using the same “chasing” process method. So, we discovered that in the equation (3.75), the vector p on the x, y and $\dot{x}, \dot{y}$ of the partial derivative is also on the matrix K by solving the linear equation, the equation (3.71) is exactly the same coefficient matrix. Therefore the equation (3.75) with equation (3.71) is the joint solution in the following equation

$$\begin{cases} Kp = f \\ K\dfrac{\partial p}{\partial x} = -c_a f_s - \dfrac{\partial K}{\partial x}p \\ K\dfrac{\partial p}{\partial y} = c_a f_c - \dfrac{\partial K}{\partial y}p \\ K\dfrac{\partial p}{\partial \dot{x}} = 2c_a f_c \\ K\dfrac{\partial p}{\partial \dot{y}} = 2c_a f_s \end{cases} \tag{3.76}$$

As stated earlier on, K is the positive symmetric definite tridiagonal matrix, the first formula in equations (3.76) is using amended "chasing" method to solve for vector p, followed by the remaining 4 types of right hand side vectors are known, its coefficient matrix is tridiagonal matrix K, therefore, "chasing" method can be used simultaneously to solve the unknown quantity (vector p of the partial derivative). Important to note that for equation (3.76), the four formulas need to be solved using the general "chasing" method. Because vector p on x, y and $\dot{x}, \dot{y}$ of the partial derivative had nothing but derivative limit.

Obviously, oil-film forces and Jacobian matrix of solving got two advantages, which are:

One, operation in the correlation matrix and vector operation are in the film positive pressure zone, and no iterative solution, can save a lot of invalid operation; secondly, the equation has the same coefficient matrix, combine the solution for K, the treatment on the matrix is only once, and it does not have to repeat the operation and thus saves a lot of computing. In this case, obtain the vector p and the partial derivative form of x, y and $\dot{x}, \dot{y}$, oil-film forces and Jacobi matrix can be obtained from the equations (3.73) and (3.74). Dimensional pressure can be obtained by dividing Jacobi matrix (3.74), J_1 by $\dfrac{6\mu\omega R_j^2}{C_r^3}$ and J_2 by $\dfrac{6\mu R_j^2}{C_r^3}$.

Finally, the local coordinate system of the oil film force (2.73) and the Jacobi matrix (2.74) is the conversion to the general coordinate system, accumulate the entire bush, eventually need to get our bearing in general and the Jacobian matrix of oil film force.

Chapter 4

Computational solving of Reynolds equation.

The following chapter is the transposition of the studies conducted in the former chapters in a numerical way to solve it. In the following pages the whole solving code will be shown, as it has been written in MatLab (Version 7). The code presents a main script, in which the user is requested to put input data (such as axial thickness of the bearing, eccentricity of the centre of the bearing, velocity of the centre of the bearing, phase shift of the centre, number of nodes and tolerance or maximum number of iterations). There are some restrictions about some input variables, such as for example the number of nodes (due to some "computational" reasons) which must be and odd number. After the input phase, the code provides to do the first step of iterations, by evaluating the pressure vector using a pre-set value of the iteration parameter k. "Pressure" sub-function gives matrices necessary to evaluate the vector by a simple loop. Then, "iteration" sub-function provides to start the loop of iterations in order to get an accurate result for the vector p. The grade of accuracy has been formerly set by the user, so the user has the faculty to get a quicker or a slower result, depending on his necessities. Once the iteration loop has finished, the code interpolate every single component of the vector of pressure, creating a double dimension mesh involving also the axial coordinate. Supposing that the pressure distribution along axial direction is an hyperbolic cosine function, sub-function "Plot2D3D" just builds a 3D graph of the pressure distribution along the whole bearing and a 2D graph for pressure distribution only along radial direction. Last simple sub-function, named "Forces", just provides to the user the value of the components of the force along the x axis and along the y axis.

mainwindow.m

```
% --------------------------------------------------------------------
%          Forces on Bearings during the transient of a rotor
%                    User Interface
%                    Main window Recall
%
%
% --------------------------------------------------------------------

% Set up program window
top     = 0.35;
left    = 0.05;
right   = 0.75;
bottom  = 0.05;
labelHt = 0.05;
spacing = 0.003;
% --------------------------------------------------------------------
% Information for all buttons
labelColor = [0.2 0.2 0.2];
top        = 0.95;
left       = 0.78;
btnWid     = 0.18;
btnHt      = 0.06;
spacing    = 0.01;
% --------------------------------------------------------------------
% The STEADY button
btnNumber   = 1;
yPos        = top-(btnNumber-1)*(btnHt+spacing);
labelStr    = 'STEADY CONDITIONS';
callbackStr = 'tribology("tribobutton")';
btnPos      = [left yPos-btnHt btnWid btnHt];
tbv         = uicontrol('Stylè,'pushbutton', ...
                    'Units','normalized', ...
                    'Position',btnPos, ...
                    'BackgroundColor',[0.5 0.5 0.5], ...
                    'ForegroundColor',[1 1 1], ...
                    'FontWeight','bold', ...
                    'String',labelStr, ...
                    'Callback', str2mat('pressurefieldsteady'));
 % --------------------------------------------------------------------
```

```
% The UNSTEADY button
btnNumber   = 2;
yPos        = top-(btnNumber-1)*(btnHt+spacing);
labelStr    = 'UNSTEADY CONDITIONS';
callbackStr = 'tribology(''tribobutton'')';
btnPos      = [left yPos-btnHt btnWid btnHt];
tbv         = uicontrol('Stylè,'pushbutton', ...
                    'Units','normalized', ...
                    'Position',btnPos, ...
                    'BackgroundColor',[0.5 0.5 0.5], ...
                    'ForegroundColor',[1 1 1], ...
                    'FontWeight','bold', ...
                    'String',labelStr, ...
                    'Callback', str2mat('pressurefieldunsteady'));
% ---------------------------------------------------------------------
% The INFO button
tbinfo = uicontrol('Stylè,'pushbutton', ...
                'Units','normalized', ...
                'Position',...
                [left bottom+1.5*btnHt+spacing btnWid 1.5*btnHt],...
                'BackgroundColor',[0.2 0.2 0.2], ...
                'ForegroundColor',[1 1 1], ...
                'FontWeight','bold', ...
                'String','Map of the Program', ...
                'Callback',str2mat('mapofprogram'));
% ---------------------------------------------------------------------
% The CLOSE button
tbclose = uicontrol('Stylè,'pushbutton', ...
                 'Units','normalized', ...
                 'Position',[left bottom btnWid 1.5*btnHt], ...
                 'BackgroundColor',[0.2 0.2 0.2], ...
                 'ForegroundColor',[1 1 1], ...
                 'FontWeight','bold', ...
                 'String','Closè, ...
                 'Callback','close(gcf)');
```

pressurefieldsteady.m

```
% ------------------------------------------------------------------
%          Forces on Bearings during the transient of a rotor
%                    User Interface - Menu Pressure
%                    Steady conditions
%                    Circular or Elliptical
%
% ------------------------------------------------------------------

% Set up program window
top     = 0.35;
left    = 0.05;
right   = 0.75;
bottom  = 0.05;
labelHt = 0.05;
spacing = 0.005;
% ------------------------------------------------------------------
% Information for all buttons
labelColor = [0.2 0.2 0.2];
top        = 0.95;
left       = 0.78;
btnWid     = 0.18;
btnHt      = 0.06;
spacing    = 0.01;
% ------------------------------------------------------------------
% The PRESSURE FIELD STEADY CIRCULAR button
btnNumber   = 1;
yPos        = top-(btnNumber-1)*(btnHt+spacing);
labelStr    = 'CIRCULAR BEARING';
callbackStr = 'tribology("tribobutton")';
btnPos      = [left yPos-btnHt btnWid btnHt];
tbv         = uicontrol('Stylè,'pushbutton', ...
                    'Units','normalized', ...
                    'Position',btnPos, ...
                    'BackgroundColor',[0.5 0.5 0.5], ...
                    'ForegroundColor',[1 1 1], ...
                    'FontWeight','bold', ...
                    'String',labelStr, ...
                    'Callback', str2mat('pressurefieldsteadycircular'));
% ------------------------------------------------------------------
```

```
% The PRESSURE FIELD STEADY ELLIPTICAL button
btnNumber   = 2;
yPos        = top-(btnNumber-1)*(btnHt+spacing);
labelStr    = 'ELLIPTICAL BEARING';
callbackStr = 'tribology(''tribobutton'')';
btnPos      = [left yPos-btnHt btnWid btnHt];
tbv         = uicontrol('Stylè,'pushbutton', ...
                    'Units','normalized', ...
                    'Position',btnPos, ...
                    'BackgroundColor',[0.5 0.5 0.5], ...
                    'ForegroundColor',[1 1 1], ...
                    'FontWeight','bold', ...
                    'String',labelStr, ...
                    'Callback', str2mat('pressurefieldsteadyelliptical'));
% ---------------------------------------------------------------------
% The MAIN button
tbinfo = uicontrol('Stylè,'pushbutton', ...
                    'Units','normalized', ...
                    'Position',...
                    [left bottom+1.5*btnHt+spacing btnWid 1.5*btnHt],...
                    'BackgroundColor',[0.2 0.2 0.2], ...
                    'ForegroundColor',[1 1 1], ...
                    'FontWeight','bold', ...
                    'String','Main Window', ...
                    'Callback',str2mat('mainwindow'));
```

pressurefieldunsteady.m

```
% ------------------------------------------------------------------------
%          Forces on Bearings during the transient of a rotor
%                    User Interface - Menu Pressure
%                    Unsteady conditions
%                    Circular or Elliptical
%
% ------------------------------------------------------------------------

% Set up program window
top     = 0.35;
left    = 0.05;
right   = 0.75;
bottom  = 0.05;
labelHt = 0.05;
spacing = 0.005;
% ------------------------------------------------------------------------
% Information for all buttons
labelColor = [0.2 0.2 0.2];
top        = 0.95;
left       = 0.78;
btnWid     = 0.18;
btnHt      = 0.06;
spacing    = 0.01;
% ------------------------------------------------------------------------
% The PRESSURE FIELD UNSTEADY CIRCULAR button
btnNumber   = 1;
yPos        = top-(btnNumber-1)*(btnHt+spacing);
labelStr    = 'CIRCULAR BEARING';
callbackStr = 'tribology("tribobutton")';
btnPos      = [left yPos-btnHt btnWid btnHt];
tbv         = uicontrol('Stylè,'pushbutton', ...
                    'Units','normalized', ...
                    'Position',btnPos, ...
                    'BackgroundColor',[0.5 0.5 0.5], ...
                    'ForegroundColor',[1 1 1], ...
                    'FontWeight','bold', ...
                    'String',labelStr, ...
                    'Callback', str2mat('pressurefieldunsteadycircular'));
% ------------------------------------------------------------------------
```

```
% The PRESSURE FIELD UNSTEADY ELLIPTICAL button
btnNumber   = 2;
yPos        = top-(btnNumber-1)*(btnHt+spacing);
labelStr    = 'ELLIPTICAL BEARING';
callbackStr = 'tribology(''tribobutton'')';
btnPos      = [left yPos-btnHt btnWid btnHt];
tbv         = uicontrol('Stylè,'pushbutton', ...
                    'Units','normalized', ...
                    'Position',btnPos, ...
                    'BackgroundColor',[0.5 0.5 0.5], ...
                    'ForegroundColor',[1 1 1], ...
                    'FontWeight','bold', ...
                    'String',labelStr, ...
                    'Callback', str2mat('pressurefieldunsteadyelliptical'));
% ---------------------------------------------------------------------
% The INFO button
tbinfo = uicontrol('Stylè,'pushbutton', ...
                    'Units','normalized', ...
                    'Position',...
                    [left bottom+1.5*btnHt+spacing btnWid 1.5*btnHt],...
                    'BackgroundColor',[0.2 0.2 0.2], ...
                    'ForegroundColor',[1 1 1], ...
                    'FontWeight','bold', ...
                    'String','Main Window', ...
                    'Callback',str2mat('mainwindow'));
```

mapofprogram.m

```
figure

% Information for all buttons ----------------------------------------

bottom  = 0.003;
left    = 0.78;
btnWid  = 0.15;
btnHt   = 0.04;

% The CLOSE button ----------------------------------------------------

tbclose = uicontrol('Stylè,'pushbutton', ...
                'Units','normalized', ...
                'Position',[left bottom btnWid 1.5*btnHt], ...
                'BackgroundColor',[0.2 0.2 0.2], ...
                'ForegroundColor',[1 1 1], ...
                'FontWeight','bold', ...
                'String','Close Map', ...
                'Callback','close(gcf)');

rectangle('Position',[0 0.25 3 1.5],'curvaturè,[0.2 0.2])
text(0.5,1.3,'Main')
text(0.5,1,'Window')
text(7.3,3.3,'Steady')
text(7.3,3,'Conditions')
text(7.3,-0.7,'Unsteady')
text(7.3,-1,'Conditions')
text(14.3,4.3,'Circular')
text(14.3,4,'Bearing')
text(14.3,2.3,'Elliptical')
text(14.3,2,'Bearing')
text(14.3,0.3,'Circular')
text(14.3,0,'Bearing')
text(14.3,-1.7,'Elliptical')
text(14.3,-2,'Bearing')
line([3 5],[1 1],'color',[0 0 0])
line([5 5],[-1 3],'color',[0 0 0])
line([5 7],[3 3],'color',[0 0 0])
line([5 7],[-1 -1],'color',[0 0 0])
```

```
rectangle('Position',[7 2.25 3 1.5],'curvaturè,[0.2 0.2])
rectangle('Position',[7 -1.75 3 1.5],'curvaturè,[0.2 0.2])
line([10 12],[3 3],'color',[0 0 0])
line([10 12],[-1 -1],'color',[0 0 0])
line([12 12],[2 4],'color',[0 0 0])
line([12 12],[-2 0],'color',[0 0 0])
line([12 14],[4 4],'color',[0 0 0])
line([12 14],[2 2],'color',[0 0 0])
line([12 14],[0 0],'color',[0 0 0])
line([12 14],[-2 -2],'color',[0 0 0])
rectangle('Position',[14 3.25 3 1.5],'curvaturè,[0.2 0.2])
rectangle('Position',[14 1.25 3 1.5],'curvaturè,[0.2 0.2])
rectangle('Position',[14 -0.75 3 1.5],'curvaturè,[0.2 0.2])
rectangle('Position',[14 -2.75 3 1.5],'curvaturè,[0.2 0.2])
axis([-1 18 -3 5])
```

pressurefieldsteadycircular.m

```
%  ------------------------------------------------------------------
%           Forces on Bearings during the transient of a rotor
%              program: Pressure Field Steady Conditions
%              CIRCULAR BEARING ~ COMPLETE BEARING
%                   BOTTOM PAD + TOP PAD
%
%  ------------------------------------------------------------------

clear all;
disp('------ Pressure Field Steady Conditions ~ Circular bearing ------');
cla reset; echo off;
global tbv tbs tbp tbt tbd tbg tba tbinfo tbclose;
set([tbv tbs tbp tbt tbd tbg tba tbinfo tbclose],'Enablè,'off');

% BEGIN OF INPUT DATA ----------------------------------------------------
% ------------------------------------------------------------------------

prompt = {'L/D ratio:',
        'Angular Extension of the bottom pad:',
        'Eccentricity ratio:',
        'Attitude angle (in degrees):'};
figTitle  = 'GEOMETRY OF THE BEARING - Bottom Pad';
lineno    = 1;
def       = {'1','170','0.8','122'};
answer    = inputdlg(prompt,figTitle,lineno,def);
if size(answer)==0, %program terminated
   set(tbp,'Valuè,get(tbp,'Min'));
   set([tbv tbs tbp tbt tbd tbg tba tbinfo tbclose],'Enablè,'On');
   break;
end;
[lambda,gammalow,epsilon,beta] = deal(answer{:});
lambda   = str2num(lambda);
gammalow = str2num(gammalow);
epsilon  = str2num(epsilon);
beta     = str2num(beta);
beta     = beta*pi/180;
gamma1   = (180-gammalow)/2;
gamma2   = gamma1+gammalow;
gammalow = gammalow*pi/180;
```

```
gamma1   = gamma1*pi/180;
gamma2   = gamma2*pi/180;

prompt = {'Number of nodes along radial coordinate:',
        'Number of nodes along axial coordinate (odd number):',
        'Initial value of iteration parameter k:',
        'm parameter:',
        'Terminating value of residual for iter. to find pressure:',
        'Maximum number of cycles during iter. to find pressure:'};
figTitle  = 'CALCULATIONS PARAMETERS - Bottom Pad';
lineno    = 1;
def       = {'31','11','3','2','0.00000001','100'};
answer    = inputdlg(prompt,figTitle,lineno,def);
if size(answer)==0, %program terminated
   set(tbp,'Valuè,get(tbp,'Min'));
   set([tbv tbs tbp tbt tbd tbg tba tbinfo tbclose],'Enablè,'On');
   break;
end;
[inode,jnode,k,m,tolerance,number_of_iterationslim] = deal(answer{:});
inode                = str2num(inode); inode2 = inode; %command rows for
jnode                = str2num(jnode); jnode2 = jnode; %Comparation.m
m                    = str2num(m);
tolerance            = str2num(tolerance);
k                    = str2num(k);
number_of_iterationslim = str2num(number_of_iterationslim);

starthour = fix(clock);
disp ('Starting date/hour:'); disp(fix(clock));

subplot(1,1,1);
text('units','normalized','position',[0.27 0.55],'FontWeight','bold',...
    'color',[0 0.3 0.6],'string','CALCULATIONS IN PROGRESS');

% BEGIN OF CALCULATIONS FOR THE BOTTOM PAD ------------------------------
% -----------------------------------------------------------------------

% CALCULATIONS FOR K1,K2 MATRICES AND fc,fs VECTORS ---------------------
-

teta = sym('tetà,'real','d');
h    = 1-epsilon*cos(teta-beta);
```

```
bs   = -epsilon*cos(beta);
bc   = epsilon*sin(beta);
k1   = zeros(inode);   k2 = zeros(inode);
fc   = zeros(inode,1); fs = zeros(inode,1);

for i=1:inode-1
   tetai        = (pi-gammalow)/2+(gamma2-gamma1)/(inode-1)*(i-1);
   tetai1       = (pi-gammalow)/2+(gamma2-gamma1)/(inode-1)*i;
   dteta        = (tetai1-tetai)/(501-1);
   L1(1,1)      = 1-(teta-tetai)/(tetai1-tetai);
   L1(2,1)      = 1-L1(1,1);
   TETA(1:501,1) = tetai:(tetai1-tetai)/(501-1):tetai1;

   Z   = h^(3-2*m)*[diff(L1)-(m*(diff(h))/h)*(L1)]*...
       [diff(L1)-(m*(diff(h))/h)*(L1)]';
   k1a = zeros(2);
   for counter1=1:2,
      for counter2=1:2,
         W = subs(Z(counter1,counter2),teta,TETA);
         for counter3=1:501-1,
           c                  = (W(counter3)+W(counter3+1))*dteta/2;
           k1a(counter1,counter2) = k1a(counter1,counter2)+c;
         end;
      end;
   end;
   k1(i,i)     = k1(i,i)+k1a(1,1);
   k1(i+1,i)   = k1(i+1,i)+k1a(2,1);
   k1(i,i+1)   = k1(i,i+1)+k1a(1,2);
   k1(i+1,i+1) = k1(i+1,i+1)+k1a(2,2);

   Z   = h^(3-2*m)*L1*L1';
   k2a = zeros(2);
   for counter1=1:2,
      for counter2=1:2,
         W = subs(Z(counter1,counter2),teta,TETA);
         for counter3=1:501-1,
           c                  = (W(counter3)+W(counter3+1))*dteta/2;
           k2a(counter1,counter2) = k2a(counter1,counter2)+c;
         end;
      end;
   end;
```

```
   k2(i,i)     = k2(i,i)+k2a(1,1);
   k2(i+1,i)   = k2(i+1,i)+k2a(2,1);
   k2(i,i+1)   = k2(i,i+1)+k2a(1,2);
   k2(i+1,i+1) = k2(i+1,i+1)+k2a(2,2);

   Z   = h^(-m)*L1*cos(teta);
   fca = zeros(2,1);
   for counter1=1:2,
      W = subs(Z(counter1),teta,TETA);
      for counter2=1:501-1,
         c           = (W(counter2)+W(counter2+1))*dteta/2;
         fca(counter1) = fca(counter1)+c;
      end;
   end;
   fc(i,1)   = fc(i,1)+fca(1,1);
   fc(i+1,1) = fc(i+1,1)+fca(2,1);

   Z   = h^(-m)*L1*sin(teta);
   fsa = zeros(2,1);
   for counter1=1:2,
      W = subs(Z(counter1),teta,TETA);
      for counter2=1:501-1,
         c           = (W(counter2)+W(counter2+1))*dteta/2;
         fsa(counter1) = fsa(counter1)+c;
      end;
   end;
   fs(i,1)   = fs(i,1)+fsa(1,1);
   fs(i+1,1) = fs(i+1,1)+fsa(2,1);
end

% CALCULATIONS FOR K MATRIX, f VECTOR AND L,U MATRICES ----------
-----

[K,f,L,U,c3] = pressure(lambda,k,k1,k2,bc,fc,bs,fs,inode);

% CALCULATIONS FOR p VECTOR ------------------------------------------

Y  = zeros(inode,1);
E  = zeros(inode,1);
pl = zeros(inode,1);
l  = inode;
```

```
E(1) = -f(1);
for i=2:inode,
   E(i) = L(i,i-1)*Y(i-1)-f(i);
   if E(i)<0
      Y(i) = -E(i)/L(i,i);
   else
      l = i-1;
      break
   end
end
pl(l) = Y(l);
for j=l-1:-1:1
   pl(j) = Y(j)-U(j,j+1)*pl(j+1);
end
w            = h^(-m)*(cosh(k*lambda)-1);
THETA(1:inode) = gamma1:(gamma2-gamma1)/(inode-1):gamma2;
W            = subs(w,teta,THETA);
pl           = pl.*W';

% ITERATION PROCESS ----------------------------------------------------

if sum(pl)~=0
   [pl,number_of_iterations,k,K,L,U,l,fxMolt,fyMolt] = iteration(k2,...
                                 k1,pl,lambda,...
                                 h,m,gamma1,gamma2,k,bc,fc,...
                                 bs,fs,inode,tolerance,c3,...
                                 number_of_iterationslim);
   [PL,ax,ang]                   = buildP(pl,lambda,inode,...
                                 jnode,gamma1,gamma2,k);

   % OIL FILM FORCES ----------------------------------------------------

   [fx,fy] = forces(PL,fc,fs,c3,l,inode,jnode,gamma1,gamma2,lambda,pl);

   % 3D-2D PLOTS ----------------------------------------------------

   close figure 1
   plot2D3D(PL,ax,ang,pl);
   figure(1),
   title('PRESSURE FIELD 3D BOTTOM PAD','units','normalized',...
```

```
        'FontWeight','bold','color',[0 0.3 0.6]);
    figure(2),
    title('PRESSURE FIELD 2D BOTTOM PAD','units','normalized',...
        'FontWeight','bold','color',[0 0.3 0.6]);
elseif sum(pl)==0
    close figure 1
    ang(1,1:inode) = gamma1*180/pi:(gamma2-gamma1)/(inode-1)*180/pi:...
                gamma2*180/pi;
    plot(ang,pl,'b.:')
    title('PRESSURE FIELD 2D BOTTOM PAD','units','normalized',...
        'FontWeight','bold','color',[0 0.3 0.6]);
    text(15,0.3,'The bottom pad is unloaded');
    xlabel('Theta (Degrees)','units','normalized','color',[0 0.3 0.6]);
    ylabel('Pressurè,'units','normalized','color',[0 0.3 0.6]);
    grid on;
end;
disp('INPUT DATA:');
fprintf('Eccentricity = %g\n',epsilon);
fprintf('L/D ratio = %g\n',lambda);
fprintf('Attitude angle = %g\n',beta*180/pi);
fprintf('Bottom pad angular extension = %g\n',gammalow*180/pi);

% BEGIN OF CALCULATIONS FOR THE TOP PAD ----------------------------------
% ------------------------------------------------------------------------

pausehour = fix(clock);

prompt    = {'Angular Extension of the top pad:'};
figTitle  = 'GEOMETRY OF THE BEARING - Top Pad';
lineno    = 1;
def       = {'170'};
answer    = inputdlg(prompt,figTitle,lineno,def);
if size(answer)==0, %program terminated
    set(tbp,'Valuè,get(tbp,'Min'));
    set([tbv tbs tbp tbt tbd tbg tba tbinfo tbclose],'Enablè,'On');
    break;
end;
[gammaup] = deal(answer{:});
gammaup   = str2num(gammaup);
gamma1    = 180+(180-gammaup)/2;
gamma2    = gamma1+gammaup;
```

```
gammaup   = gammaup*pi/180;
gamma1    = gamma1*pi/180;
gamma2    = gamma2*pi/180;

prompt = {'Number of nodes along radial coordinate:',
        'Number of nodes along axial coordinate (odd number):',
        'Initial value of iteration parameter k:',
        'm parameter:',
        'Terminating value of residual for iter. to find pressure:',
        'Maximum number of cycles during iter. to find pressure:'};
figTitle  = 'CALCULATIONS PARAMETERS - Top Pad';
lineno    = 1;
def       = {'31','11','3','2','0.00000001','100'};
answer    = inputdlg(prompt,figTitle,lineno,def);
if size(answer)==0, %program terminated
   set(tbp,'Valuè,get(tbp,'Min'));
   set([tbv tbs tbp tbt tbd tbg tba tbinfo tbclose],'Enablè,'On');
   break;
end;
[inode,jnode,k,m,tolerance,number_of_iterationslim] = deal(answer{:});
inode                 = str2num(inode);
jnode                 = str2num(jnode);
m                     = str2num(m);
tolerance             = str2num(tolerance);
k                     = str2num(k);
number_of_iterationslim = str2num(number_of_iterationslim);

restarthour = fix(clock);

k1   = zeros(inode);   k2 = zeros(inode);
fc   = zeros(inode,1); fs = zeros(inode,1);

for i=1:inode-1
   tetai        = pi+(pi-gammalow)/2+(gamma2-gamma1)/(inode-1)*(i-1);
   tetai1       = pi+(pi-gammalow)/2+(gamma2-gamma1)/(inode-1)*i;
   dteta        = (tetai1-tetai)/(501-1);
   L1(1,1)      = 1-(teta-tetai)/(tetai1-tetai);
   L1(2,1)      = 1-L1(1,1);
   TETA(1:501,1) = tetai:(tetai1-tetai)/(501-1):tetai1;

   Z   = h^(3-2*m)*[diff(L1)-(m*(diff(h))/h)*(L1)]*...
```

```
    [diff(L1)-(m*(diff(h))/h)*(L1)]';
k1a = zeros(2);
for counter1=1:2,
   for counter2=1:2,
      W = subs(Z(counter1,counter2),teta,TETA);
      for counter3=1:501-1,
        c                  = (W(counter3)+W(counter3+1))*dteta/2;
        k1a(counter1,counter2) = k1a(counter1,counter2)+c;
      end;
   end;
end;
k1(i,i)     = k1(i,i)+k1a(1,1);
k1(i+1,i)   = k1(i+1,i)+k1a(2,1);
k1(i,i+1)   = k1(i,i+1)+k1a(1,2);
k1(i+1,i+1) = k1(i+1,i+1)+k1a(2,2);

Z   = h^(3-2*m)*L1*L1';
k2a = zeros(2);
for counter1=1:2,
   for counter2=1:2,
      W = subs(Z(counter1,counter2),teta,TETA);
      for counter3=1:501-1,
        c                  = (W(counter3)+W(counter3+1))*dteta/2;
        k2a(counter1,counter2) = k2a(counter1,counter2)+c;
      end;
   end;
end;
k2(i,i)     = k2(i,i)+k2a(1,1);
k2(i+1,i)   = k2(i+1,i)+k2a(2,1);
k2(i,i+1)   = k2(i,i+1)+k2a(1,2);
k2(i+1,i+1) = k2(i+1,i+1)+k2a(2,2);

Z   = h^(-m)*L1*cos(teta);
fca = zeros(2,1);
for counter1=1:2,
   W = subs(Z(counter1),teta,TETA);
   for counter2=1:501-1,
      c           = (W(counter2)+W(counter2+1))*dteta/2;
      fca(counter1) = fca(counter1)+c;
   end;
end;
```

```
   fc(i,1)   = fc(i,1)+fca(1,1);
   fc(i+1,1) = fc(i+1,1)+fca(2,1);

   Z   = h^(-m)*L1*sin(teta);
   fsa = zeros(2,1);
   for counter1=1:2,
      W = subs(Z(counter1),teta,TETA);
      for counter2=1:501-1,
         c           = (W(counter2)+W(counter2+1))*dteta/2;
         fsa(counter1) = fsa(counter1)+c;
      end;
   end;
   fs(i,1)   = fs(i,1)+fsa(1,1);
   fs(i+1,1) = fs(i+1,1)+fsa(2,1);
end

% CALCULATIONS FOR K MATRIX, f VECTOR AND L,U MATRICES ----------
-----

[K,f,L,U,c3] = pressure(lambda,k,k1,k2,bc,fc,bs,fs,inode);

% CALCULATIONS FOR p VECTOR ------------------------------------------

Y  = zeros(inode,1);
E  = zeros(inode,1);
pu = zeros(inode,1);
l  = inode;

E(1) = -f(1);
for i=2:inode,
   E(i) = L(i,i-1)*Y(i-1)-f(i);
   if E(i)<0
      Y(i) = -E(i)/L(i,i);
   else
      l = i-1;
      break
   end
end
pu(l) = Y(l);
for j=l-1:-1:1
   pu(j) = Y(j)-U(j,j+1)*pu(j+1);
```

```
end
w           = h^(-m)*(cosh(k*lambda)-1);
THETA       = zeros(1,inode);
THETA(1:inode) = gamma1:(gamma2-gamma1)/(inode-1):gamma2;
W           = subs(w,teta,THETA);
pu          = pu.*W';

% ITERATION PROCESS ----------------------------------------------------

if sum(pu)~=0
   [pu,number_of_iterations,k,K,L,U,l,fxMolt,fyMolt] = iteration(k2,...
                                k1,pl,lambda,...
                                h,m,gamma1,gamma2,k,bc,fc,...
                                bs,fs,inode,tolerance,c3,...
                                number_of_iterationslim);
   [PU,ax,ang]                  = buildP(pu,lambda,inode,...
                                jnode,gamma1,gamma2,k);

   % OIL FILM FORCES ----------------------------------------------------

   [fx,fy] = forces(PU,fc,fs,c3,l,inode,jnode,gamma1,gamma2,lambda,pu);

   % 3D-2D PLOTS ----------------------------------------------------------

   plot2D3D(PU,ax,ang,pu);
   figure(gcf-1),
   title('PRESSURE FIELD 3D TOP PAD','units','normalized',...
       'FontWeight','bold','color',[0 0.3 0.6]);
   figure(gcf+1),
   title('PRESSURE FIELD 2D TOP PAD','units','normalized',...
       'FontWeight','bold','color',[0 0.3 0.6]);
elseif sum(pu)==0
   figure
   ang          = zeros(1,inode);
   ang(1,1:inode) = gamma1*180/pi:(gamma2-gamma1)/(inode-1)*180/pi:...
               gamma2*180/pi;
   plot(ang,pu,'b.:')
   title('PRESSURE FIELD 2D TOP PAD','units','normalized',...
       'FontWeight','bold','color',[0 0.3 0.6]);
   text(205,0.3,'The top pad is unloaded');
   xlabel('Theta (Degrees)','units','normalized','color',[0 0.3 0.6]);
```

```
   ylabel('Pressurè,'units','normalized','color',[0 0.3 0.6]);
   grid on;
end;

finishhour    = fix(clock);
totalcomptime = pausehour-starthour+(finishhour-restarthour);

fprintf('Top pad angular extension = %g\n\n',gammaup*180/pi);

disp('OUTPUT DATA:');
fprintf('fx = %g\n\n',fx)
fprintf('fy = %g\n\n',fy)
disp ('Finishing date/hour:');     disp(fix(clock));
disp('Total computational time:'), disp(totalcomptime);

% The RESTART button ----------------------------------------------------

tbinfo = uicontrol('Stylè,'pushbutton', ...
                   'Units','normalized', ...
                   'Position',...
                   [0.001 0.001 0.1 1.5*0.06],...
                   'BackgroundColor',[0.2 0.2 0.2], ...
                   'ForegroundColor',[1 1 1], ...
                   'FontWeight','bold', ...
                   'String','RESTART', ...
                   'Callback',str2mat('tribology_userinterfacè));
```

pressurefieldsteadyelliptical.m

```
%  ------------------------------------------------------------------
%           Forces on Bearings during the transient of a rotor
%           program: Pressure Field Steady Conditions
%             ELLIPTICAL BEARING ~ COMPLETE BEARING
%                   BOTTOM PAD + TOP PAD
%
%  ------------------------------------------------------------------

clear all;
disp('------ Pressure Field Steady Conditions ~ Elliptical bearing ------')
cla reset; echo off;
global tbv tbs tbp tbt tbd tbg tba tbinfo tbclose;
set([tbv tbs tbp tbt tbd tbg tba tbinfo tbclose],'Enablè,'off');

% BEGIN OF INPUT DATA ----------------------------------------------------
% ------------------------------------------------------------------------

prompt = {'L/D ratio:',
        'Angular Extension of the bottom pad:',
        'Eccentricity ratio:',
        'Ellipticity:',
        'First guess of the attitude angle (in degrees):',
        'Load inclination angle (in degrees):'};
figTitle  = 'GEOMETRY OF THE BEARING - Bottom Pad';
lineno    = 1;
def       = {'0.5','160','0.4','0.5','5','0'};
answer    = inputdlg(prompt,figTitle,lineno,def);
if size(answer)==0, %program terminated
   set(tbp,'Valuè,get(tbp,'Min'));
   set([tbv tbs tbp tbt tbd tbg tba tbinfo tbclose],'Enablè,'On');
   break;
end;
[lambda,gammalow,epsilon,delta,beta,fi] = deal(answer{:});
lambda   = str2num(lambda);
gammalow = str2num(gammalow);
epsilon  = str2num(epsilon);
delta    = str2num(delta);
beta     = str2num(beta);
fi       = str2num(fi);
```

```
beta     = (beta+fi)*pi/180;
gamma1   = (180-gammalow)/2;
gamma2   = gamma1+gammalow;
gammalow = gammalow*pi/180;
gamma1   = gamma1*pi/180;
gamma2   = gamma2*pi/180;

prompt = {'Number of nodes along radial coordinate:',
        'Number of nodes along axial coordinate (odd number):',
        'Initial value of iteration parameter k:',
        'm parameter:',
        'Relaxation Factor to find the attitude angle:',
        'Terminating value of residual for iter. to find pressure:',
        'Maximum number of cycles during iter. to find pressure:'};
figTitle  = 'CALCULATIONS PARAMETERS - Bottom Pad';
lineno    = 1;
def       = {'31','11','3','2','0.0001','0.00000001','100'};
answer    = inputdlg(prompt,figTitle,lineno,def);
if size(answer)==0, %program terminated
   set(tbp,'Valuè,get(tbp,'Min'));
   set([tbv tbs tbp tbt tbd tbg tba tbinfo tbclose],'Enablè,'On');
   break;
end;
[inode,jnode,k,m,relax,tolerance,number_of_iterationslim] = deal(answer{:});
inode                = str2num(inode);
jnode                = str2num(jnode);
m                    = str2num(m);
tolerance            = str2num(tolerance);
relax                = str2num(relax);
k                    = str2num(k);
number_of_iterationslim = str2num(number_of_iterationslim);

starthour = fix(clock);
disp ('Starting date/hour:'); disp(fix(clock));

subplot(1,1,1);
text('units','normalized','position',[0.27 0.55],'FontWeight','bold',...
    'color',[0 0.3 0.6],'string','CALCULATIONS IN PROGRESS');

gBeta = 1;
```

```
% BEGIN OF CALCULATIONS FOR THE BOTTOM PAD ------------------------------
% ----------------------------------------------------------------------

% CALCULATIONS FOR K1,K2 MATRICES AND fc,fs VECTORS ----------------------
-

teta     = sym('tetà,'real','d');
epsilonb = sqrt(epsilon^2+delta^2+2*epsilon*delta*cos(beta));
sinbetab = epsilon/epsilonb*sin(beta);
cosbetab = (delta^2+epsilonb^2-epsilon^2)/(2*delta*epsilonb);
betab    = atan2(sinbetab,cosbetab);
betab    = betab+pi/2;
h        = 1-epsilonb*cos(teta-betab);
bs       = -epsilonb*cos(betab);
bc       = epsilonb*sin(betab);
k1       = zeros(inode);   k2 = zeros(inode);
fc       = zeros(inode,1); fs = zeros(inode,1);

for i=1:inode-1
   tetai         = (pi-gammalow)/2+(gamma2-gamma1)/(inode-1)*(i-1);
   tetai1        = (pi-gammalow)/2+(gamma2-gamma1)/(inode-1)*i;
   dteta         = (tetai1-tetai)/(501-1);
   L1(1,1)       = 1-(teta-tetai)/(tetai1-tetai);
   L1(2,1)       = 1-L1(1,1);
   TETA(1:501,1) = tetai:(tetai1-tetai)/(501-1):tetai1;

   Z   = h^(3-2*m)*[diff(L1)-(m*(diff(h))/h)*(L1)]*...
       [diff(L1)-(m*(diff(h))/h)*(L1)]';
   k1a = zeros(2);
   for counter1=1:2,
      for counter2=1:2,
         W = subs(Z(counter1,counter2),teta,TETA);
         for counter3=1:501-1,
           c                  = (W(counter3)+W(counter3+1))*dteta/2;
           k1a(counter1,counter2) = k1a(counter1,counter2)+c;
         end;
      end;
   end;
   k1(i,i)     = k1(i,i)+k1a(1,1);
   k1(i+1,i)   = k1(i+1,i)+k1a(2,1);
   k1(i,i+1)   = k1(i,i+1)+k1a(1,2);
```

```
k1(i+1,i+1) = k1(i+1,i+1)+k1a(2,2);

Z   = h^(3-2*m)*L1*L1';
k2a = zeros(2);
for counter1=1:2,
   for counter2=1:2,
      W = subs(Z(counter1,counter2),teta,TETA);
      for counter3=1:501-1,
        c                  = (W(counter3)+W(counter3+1))*dteta/2;
        k2a(counter1,counter2) = k2a(counter1,counter2)+c;
      end;
   end;
end;
k2(i,i)     = k2(i,i)+k2a(1,1);
k2(i+1,i)   = k2(i+1,i)+k2a(2,1);
k2(i,i+1)   = k2(i,i+1)+k2a(1,2);
k2(i+1,i+1) = k2(i+1,i+1)+k2a(2,2);

Z   = h^(-m)*L1*cos(teta);
fca = zeros(2,1);
for counter1=1:2,
   W = subs(Z(counter1),teta,TETA);
   for counter2=1:501-1,
      c           = (W(counter2)+W(counter2+1))*dteta/2;
      fca(counter1) = fca(counter1)+c;
   end;
end;
fc(i,1)   = fc(i,1)+fca(1,1);
fc(i+1,1) = fc(i+1,1)+fca(2,1);

Z   = h^(-m)*L1*sin(teta);
fsa = zeros(2,1);
for counter1=1:2,
   W = subs(Z(counter1),teta,TETA);
   for counter2=1:501-1,
      c           = (W(counter2)+W(counter2+1))*dteta/2;
      fsa(counter1) = fsa(counter1)+c;
   end;
end;
fs(i,1)   = fs(i,1)+fsa(1,1);
fs(i+1,1) = fs(i+1,1)+fsa(2,1);
```

```
end

% CALCULATIONS FOR K MATRIX, f VECTOR AND L,U MATRICES ----------
-----

[K,f,L,U,c3] = pressure(lambda,k,k1,k2,bc,fc,bs,fs,inode);

% CALCULATIONS FOR p VECTOR ------------------------------------------

Y  = zeros(inode,1);
E  = zeros(inode,1);
pl = zeros(inode,1);
l  = inode;

E(1) = -f(1);
for i=2:inode,
   E(i) = L(i,i-1)*Y(i-1)-f(i);
   if E(i)<0
      Y(i) = -E(i)/L(i,i);
   else
      l = i-1;
      break
   end
end
pl(l) = Y(l);
for j=l-1:-1:1
   pl(j) = Y(j)-U(j,j+1)*pl(j+1);
end
w            = h^(-m)*(cosh(k*lambda)-1);
THETA(1:inode) = gamma1:(gamma2-gamma1)/(inode-1):gamma2;
W            = subs(w,teta,THETA);
pl           = pl.*W';

% ITERATION PROCESS ---------------------------------------------------

if sum(pl)~=0
   [pl,number_of_iterations,k,K,L,U,l,fxMoltB,fyMoltB] = iteration(k2,...
                                    k1,pl,lambda,...
                                    h,m,gamma1,gamma2,k,bc,fc,...
                                    bs,fs,inode,tolerance,c3,...
                                    number_of_iterationslim);
```

```
   [PL,ax,ang]                    = buildP(pl,lambda,inode,...
                                  jnode,gamma1,gamma2,k);

   % OIL FILM FORCES ----------------------------------------------------

   [fxB,fyB] = forces(PL,fc,fs,c3,l,inode,jnode,gamma1,gamma2,lambda,pl);

   % 3D-2D PLOTS ----------------------------------------------------------

   close figure 1
   plot2D3D(PL,ax,ang,pl);
   figure(1),
   title('PRESSURE FIELD 3D BOTTOM PAD','units','normalized',...
       'FontWeight','bold','color',[0 0.3 0.6]);
   figure(2),
   title('PRESSURE FIELD 2D BOTTOM PAD','units','normalized',...
       'FontWeight','bold','color',[0 0.3 0.6]);
elseif sum(pl)==0
   close figure 1
   fxMoltB       = 0; fxB = 0;
   fyMoltB       = 0; fyB = 0;
   ang(1,1:inode) = gamma1*180/pi:(gamma2-gamma1)/(inode-1)*180/pi:...
                gamma2*180/pi;
   plot(ang,pl,'b.:')
   title('PRESSURE FIELD 2D BOTTOM PAD','units','normalized',...
       'FontWeight','bold','color',[0 0.3 0.6]);
   text(15,0.3,'The bottom pad is unloaded');
   xlabel('Theta (Degrees)','units','normalized','color',[0 0.3 0.6]);
   ylabel('Pressurè,'units','normalized','color',[0 0.3 0.6]);
   grid on;
end;
disp('INPUT DATA:');
fprintf('Eccentricity = %g\n',epsilon);
fprintf('L/D ratio = %g\n',lambda);
fprintf('Ellipticity = %g\n',delta);
fprintf('Attitude angle = %g\n',beta*180/pi);
fprintf('Attitude angle for the bottom pad = %g\n',(betab-pi/2)*180/pi);
fprintf('Bottom pad angular extension = %g\n',gammalow*180/pi);

% BEGIN OF CALCULATIONS FOR THE TOP PAD ----------------------------------
% ------------------------------------------------------------------------
```

```
pausehour = fix(clock);

prompt    = {'Angular Extension of the top pad:'};
figTitle  = 'GEOMETRY OF THE BEARING - Top Pad';
lineno    = 1;
def       = {'160'};
answer    = inputdlg(prompt,figTitle,lineno,def);
if size(answer)==0, %program terminated
   set(tbp,'Valuè,get(tbp,'Min'));
   set([tbv tbs tbp tbt tbd tbg tba tbinfo tbclose],'Enablè,'On');
   break;
end;
[gammaup] = deal(answer{:});
gammaup   = str2num(gammaup);
gamma1    = 180+(180-gammaup)/2;
gamma2    = gamma1+gammaup;
gammaup   = gammaup*pi/180;
gamma1    = gamma1*pi/180;
gamma2    = gamma2*pi/180;

prompt = {'Number of nodes along radial coordinate:',
        'Number of nodes along axial coordinate (odd number):',
        'Initial value of iteration parameter k:',
        'm parameter:',
        'Terminating value of residual for iter. to find pressure:',
        'Maximum number of cycles during iter. to find pressure:'};
figTitle  = 'CALCULATIONS PARAMETERS - Top Pad';
lineno    = 1;
def       = {'31','11','3','2','0.00001','100'};
answer    = inputdlg(prompt,figTitle,lineno,def);
if size(answer)==0, %program terminated
   set(tbp,'Valuè,get(tbp,'Min'));
   set([tbv tbs tbp tbt tbd tbg tba tbinfo tbclose],'Enablè,'On');
   break;
end;
[inode,jnode,k,m,tolerance,number_of_iterationslim] = deal(answer{:});
inode                = str2num(inode);
jnode                = str2num(jnode);
m                   = str2num(m);
tolerance            = str2num(tolerance);
```

```
k                    = str2num(k);
number_of_iterationslim = str2num(number_of_iterationslim);

restarthour = fix(clock);

epsilont = sqrt(epsilon^2+delta^2-2*epsilon*delta*cos(beta));
sinbetat = epsilon/epsilont*sin(beta);
cosbetat = (-delta^2-epsilont^2+epsilon^2)/(2*delta*epsilont);
betat    = atan2(sinbetat,cosbetat);
betat    = betat+pi/2;
h        = 1-epsilont*cos(teta-betat);
bs       = -epsilont*cos(betat);
bc       = epsilont*sin(betat);
k1       = zeros(inode);   k2 = zeros(inode);
fc       = zeros(inode,1); fs = zeros(inode,1);

for i=1:inode-1
   tetai        = pi+(pi-gammalow)/2+(gamma2-gamma1)/(inode-1)*(i-1);
   tetai1       = pi+(pi-gammalow)/2+(gamma2-gamma1)/(inode-1)*i;
   dteta        = (tetai1-tetai)/(501-1);
   L1(1,1)      = 1-(teta-tetai)/(tetai1-tetai);
   L1(2,1)      = 1-L1(1,1);
   TETA(1:501,1) = tetai:(tetai1-tetai)/(501-1):tetai1;

   Z   = h^(3-2*m)*[diff(L1)-(m*(diff(h))/h)*(L1)]*...
       [diff(L1)-(m*(diff(h))/h)*(L1)]';
   k1a = zeros(2);
   for counter1=1:2,
      for counter2=1:2,
         W = subs(Z(counter1,counter2),teta,TETA);
         for counter3=1:501-1,
           c                  = (W(counter3)+W(counter3+1))*dteta/2;
           k1a(counter1,counter2) = k1a(counter1,counter2)+c;
         end;
      end;
   end;
   k1(i,i)     = k1(i,i)+k1a(1,1);
   k1(i+1,i)   = k1(i+1,i)+k1a(2,1);
   k1(i,i+1)   = k1(i,i+1)+k1a(1,2);
   k1(i+1,i+1) = k1(i+1,i+1)+k1a(2,2);
```

```
   Z   = h^(3-2*m)*L1*L1';
   k2a = zeros(2);
   for counter1=1:2,
      for counter2=1:2,
         W = subs(Z(counter1,counter2),teta,TETA);
         for counter3=1:501-1,
           c                = (W(counter3)+W(counter3+1))*dteta/2;
           k2a(counter1,counter2) = k2a(counter1,counter2)+c;
         end;
      end;
   end;
   k2(i,i)     = k2(i,i)+k2a(1,1);
   k2(i+1,i)   = k2(i+1,i)+k2a(2,1);
   k2(i,i+1)   = k2(i,i+1)+k2a(1,2);
   k2(i+1,i+1) = k2(i+1,i+1)+k2a(2,2);

   Z   = h^(-m)*L1*cos(teta);
   fca = zeros(2,1);
   for counter1=1:2,
      W = subs(Z(counter1),teta,TETA);
      for counter2=1:501-1,
         c           = (W(counter2)+W(counter2+1))*dteta/2;
         fca(counter1) = fca(counter1)+c;
      end;
   end;
   fc(i,1)   = fc(i,1)+fca(1,1);
   fc(i+1,1) = fc(i+1,1)+fca(2,1);

   Z   = h^(-m)*L1*sin(teta);
   fsa = zeros(2,1);
   for counter1=1:2,
      W = subs(Z(counter1),teta,TETA);
      for counter2=1:501-1,
         c           = (W(counter2)+W(counter2+1))*dteta/2;
         fsa(counter1) = fsa(counter1)+c;
      end;
   end;
   fs(i,1)   = fs(i,1)+fsa(1,1);
   fs(i+1,1) = fs(i+1,1)+fsa(2,1);
end
```

```
% CALCULATIONS FOR K MATRIX, f VECTOR AND L,U MATRICES ----------
-----

[K,f,L,U,c3] = pressure(lambda,k,k1,k2,bc,fc,bs,fs,inode);

% CALCULATIONS FOR p VECTOR ------------------------------------------

Y  = zeros(inode,1);
E  = zeros(inode,1);
pu = zeros(inode,1);
l  = inode;

E(1) = -f(1);
for i=2:inode,
   E(i) = L(i,i-1)*Y(i-1)-f(i);
   if E(i)<0
      Y(i) = -E(i)/L(i,i);
   else
      l = i-1;
      break
   end
end
pu(l) = Y(l);
for j=l-1:-1:1
   pu(j) = Y(j)-U(j,j+1)*pu(j+1);
end
w           = h^(-m)*(cosh(k*lambda)-1);
THETA       = zeros(1,inode);
THETA(1:inode) = gamma1:(gamma2-gamma1)/(inode-1):gamma2;
W           = subs(w,teta,THETA);
pu          = pu.*W';

% ITERATION PROCESS --------------------------------------------------

if sum(pu)~=0
   [pu,number_of_iterations,k,K,L,U,l,fxMoltT,fyMoltT] = iteration(k2,...
                                k1,pu,lambda,...
                                h,m,gamma1,gamma2,k,bc,fc,...
                                bs,fs,inode,tolerance,c3,...
                                number_of_iterationslim);
   [PU,ax,ang]                  = buildP(pu,lambda,inode,...
```

```
                                jnode,gamma1,gamma2,k);

   % OIL FILM FORCES ----------------------------------------------------

   [fxT,fyT] = forces(PU,fc,fs,c3,l,inode,jnode,gamma1,gamma2,lambda,pu);

   % 3D-2D PLOTS ---------------------------------------------------------

   plot2D3D(PU,ax,ang,pu);
   figure(gcf-1),
   title('PRESSURE FIELD 3D TOP PAD','units','normalized',...
       'FontWeight','bold','color',[0 0.3 0.6]);
   figure(gcf+1),
   title('PRESSURE FIELD 2D TOP PAD','units','normalized',...
       'FontWeight','bold','color',[0 0.3 0.6]);
elseif sum(pu)==0
   fxMoltT       = 0; fxT = 0;
   fyMoltT       = 0; fyT = 0;
   figure
   ang           = zeros(1,inode);
   ang(1,1:inode) = gamma1*180/pi:(gamma2-gamma1)/(inode-1)*180/pi:...
                gamma2*180/pi;
   plot(ang,pu,'b.:')
   title('PRESSURE FIELD 2D TOP PAD','units','normalized',...
       'FontWeight','bold','color',[0 0.3 0.6]);
   text(205,0.3,'The top pad is unloaded');
   xlabel('Theta (Degrees)','units','normalized','color',[0 0.3 0.6]);
   ylabel('Pressurè,'units','normalized','color',[0 0.3 0.6]);
   grid on;
end;

fx = fxMoltT+fxMoltB;
fy = (fyMoltT+fyMoltB)*10;

finishhour    = fix(clock);
totalcomptime = pausehour-starthour+(finishhour-restarthour);

fprintf('Attitude angle for the top pad = %g\n',(betat-pi/2)*180/pi);
fprintf('Top pad angular extension = %g\n\n',gammaup*180/pi);

disp('OUTPUT DATA:');
```

```
fprintf('fxMolt = %g\n',fx)
fprintf('fyMolt = %g\n',fy)
fprintf('fxInt = %g\n',fxT+fxB)
fprintf('fyInt = %g\n\n',fyT+fyB)

disp ('Finishing date/hour:');     disp(fix(clock));
disp('Total computational time:'), disp(totalcomptime);

% The RESTART button ----------------------------------------------------

tbinfo = uicontrol('Stylè,'pushbutton', ...
                   'Units','normalized', ...
                   'Position',...
                   [0.001 0.001 0.1 1.5*0.06],...
                   'BackgroundColor',[0.2 0.2 0.2], ...
                   'ForegroundColor',[1 1 1], ...
                   'FontWeight','bold', ...
                   'String','RESTART', ...
                   'Callback',str2mat('tribology_userinterfacè));
```

pressurefieldunsteadycircular.m

```
% ------------------------------------------------------------------
%          Forces on Bearings during the transient of a rotor
%             program: Pressure Field Unsteady Conditions
%             CIRCULAR BEARING ~ COMPLETE BEARING
%                  BOTTOM PAD + TOP PAD
%
% ------------------------------------------------------------------

clear all;
disp('------ Pressure Field Unsteady Conditions ~ Circular bearing ------')
cla reset; echo off;
global tbv tbs tbp tbt tbd tbg tba tbinfo tbclose;
set([tbv tbs tbp tbt tbd tbg tba tbinfo tbclose],'Enablè,'off');

% BEGIN OF INPUT DATA ---------------------------------------------------
% ----------------------------------------------------------------------

prompt = {'L/D ratio:',
        'Angular Extension of the bottom pad:',
        'Eccentricity ratio:',
        'Attitude angle (in degrees):',
        'Radial Velocity:',
        'Tangential Velocity:'};
figTitle  = 'GEOMETRY OF THE BEARING - Bottom Pad';
lineno    = 1;
def       = {'1','170','0.8','30','1','1'};
answer    = inputdlg(prompt,figTitle,lineno,def);
if size(answer)==0, %program terminated
   set(tbp,'Valuè,get(tbp,'Min'));
   set([tbv tbs tbp tbt tbd tbg tba tbinfo tbclose],'Enablè,'On');
   break;
end;
[lambda,gammalow,epsilon,beta,edot,epsidot] = deal(answer{:});
```

```
lambda   = str2num(lambda);
gammalow = str2num(gammalow);
epsilon  = str2num(epsilon);
beta     = str2num(beta);
edot     = str2num(edot);
epsidot  = str2num(epsidot);
beta     = beta*pi/180;
gamma1   = (180-gammalow)/2;
gamma2   = gamma1+gammalow;
gammalow = gammalow*pi/180;
gamma1   = gamma1*pi/180;
gamma2   = gamma2*pi/180;

prompt = {'Number of nodes along radial coordinate:',
        'Number of nodes along axial coordinate (odd number):',
        'm parameter:',
        'Initial value of iteration parameter k:',
        'Terminating value of residual for iter. to find pressure:',
        'Maximum number of cycles during iter. to find pressure:'};
figTitle  = 'CALCULATIONS PARAMETERS - Bottom Pad';
lineno    = 1;
def       = {'31','11','2','3','0.00001','1000'};
answer    = inputdlg(prompt,figTitle,lineno,def);
if size(answer)==0, %program terminated
   set(tbp,'Valuè,get(tbp,'Min'));
   set([tbv tbs tbp tbt tbd tbg tba tbinfo tbclose],'Enablè,'On');
   break;
end;
[inode,jnode,m,k,tolerance,number_of_iterationslim] = deal(answer{:});
inode                = str2num(inode);
jnode                = str2num(jnode);
tolerance            = str2num(tolerance);
m                    = str2num(m);
k                    = str2num(k);
number_of_iterationslim = str2num(number_of_iterationslim);
```

```
starthour = fix(clock);
disp ('Starting date/hour:'); disp(fix(clock));

subplot(1,1,1);
text('units','normalized','position',[0.27 0.55],'FontWeight','bold',...
    'color',[0 0.3 0.6],'string','CALCULATIONS IN PROGRESS');

% BEGIN OF CALCULATIONS FOR THE BOTTOM PAD ------------------------------
% ----------------------------------------------------------------------

% CALCULATIONS FOR K1,K2 MATRICES AND fc,fs VECTORS ----------------------
-

teta = sym('tetà,'real','d');
h    = 1-epsilon*cos(teta-beta);
x    = epsilon*cos(beta);
y    = epsilon*sin(beta);
xdot = edot*cos(beta)-epsidot*sin(beta);
ydot = edot*sin(beta)+epsidot*cos(beta);
bc   = 2*xdot+y;
bs   = 2*ydot-x;
k1   = zeros(inode);   k2 = zeros(inode);
fc   = zeros(inode,1); fs = zeros(inode,1);

for i=1:inode-1
   tetai        = (pi-gammalow)/2+(gamma2-gamma1)/(inode-1)*(i-1);
   tetai1       = (pi-gammalow)/2+(gamma2-gamma1)/(inode-1)*i;
   dteta        = (tetai1-tetai)/(501-1);
   L1(1,1)      = 1-(teta-tetai)/(tetai1-tetai);
   L1(2,1)      = 1-L1(1,1);
   TETA(1:501,1) = tetai:(tetai1-tetai)/(501-1):tetai1;

   Z   = h^(3-2*m)*[diff(L1)-(m*(diff(h))/h)*(L1)]*...
       [diff(L1)-(m*(diff(h))/h)*(L1)]';
```

```
k1a = zeros(2);
for counter1=1:2,
   for counter2=1:2,
      W = subs(Z(counter1,counter2),teta,TETA);
      for counter3=1:501-1,
        c                  = (W(counter3)+W(counter3+1))*dteta/2;
        k1a(counter1,counter2) = k1a(counter1,counter2)+c;
      end;
   end;
end;
k1(i,i)     = k1(i,i)+k1a(1,1);
k1(i+1,i)   = k1(i+1,i)+k1a(2,1);
k1(i,i+1)   = k1(i,i+1)+k1a(1,2);
k1(i+1,i+1) = k1(i+1,i+1)+k1a(2,2);

Z   = h^(3-2*m)*L1*L1';
k2a = zeros(2);
for counter1=1:2,
   for counter2=1:2,
      W = subs(Z(counter1,counter2),teta,TETA);
      for counter3=1:501-1,
        c                  = (W(counter3)+W(counter3+1))*dteta/2;
        k2a(counter1,counter2) = k2a(counter1,counter2)+c;
      end;
   end;
end;
k2(i,i)     = k2(i,i)+k2a(1,1);
k2(i+1,i)   = k2(i+1,i)+k2a(2,1);
k2(i,i+1)   = k2(i,i+1)+k2a(1,2);
k2(i+1,i+1) = k2(i+1,i+1)+k2a(2,2);

Z   = h^(-m)*L1*cos(teta);
fca = zeros(2,1);
for counter1=1:2,
   W = subs(Z(counter1),teta,TETA);
```

```
      for counter2=1:501-1,
         c          = (W(counter2)+W(counter2+1))*dteta/2;
         fca(counter1) = fca(counter1)+c;
      end;
   end;
   fc(i,1)   = fc(i,1)+fca(1,1);
   fc(i+1,1) = fc(i+1,1)+fca(2,1);

   Z   = h^(-m)*L1*sin(teta);
   fsa = zeros(2,1);
   for counter1=1:2,
      W = subs(Z(counter1),teta,TETA);
      for counter2=1:501-1,
         c          = (W(counter2)+W(counter2+1))*dteta/2;
         fsa(counter1) = fsa(counter1)+c;
      end;
   end;
   fs(i,1)   = fs(i,1)+fsa(1,1);
   fs(i+1,1) = fs(i+1,1)+fsa(2,1);
end

% CALCULATIONS FOR K MATRIX, f VECTOR AND L,U MATRICES ----------
-----

[K,f,L,U,c3] = pressure(lambda,k,k1,k2,bc,fc,bs,fs,inode);

% CALCULATIONS FOR p VECTOR ------------------------------------------

Y  = zeros(inode,1);
E  = zeros(inode,1);
pl = zeros(inode,1);
l  = inode;

E(1) = -f(1);
for i=2:inode,
```

```
   E(i) = L(i,i-1)*Y(i-1)-f(i);
   if E(i)<0
      Y(i) = -E(i)/L(i,i);
   else
      l = i-1;
      break
   end
end
pl(l) = Y(l);
for j=l-1:-1:1
   pl(j) = Y(j)-U(j,j+1)*pl(j+1);
end
w            = h^(-m)*(cosh(k*lambda)-1);
THETA(1:inode) = gamma1:(gamma2-gamma1)/(inode-1):gamma2;
W            = subs(w,teta,THETA);
pl           = pl.*W';

% ITERATION PROCESS ----------------------------------------------------

if sum(pl)~=0
   [pl,number_of_iterations,k,K,L,U,l,fxMolt,fyMolt] = iteration(k2,...
                                 k1,pl,lambda,...
                                 h,m,gamma1,gamma2,k,bc,fc,...
                                 bs,fs,inode,tolerance,c3,...
                                 number_of_iterationslim);
   [PL,ax,ang]                   = buildP(pl,lambda,inode,...
                                 jnode,gamma1,gamma2,k);

   % OIL FILM FORCES ----------------------------------------------------

%     [fx,fy] = forces(PL,fc,fs,c3,l,inode,jnode,gamma1,gamma2,lambda,pl);

   % 3D-2D PLOTS ----------------------------------------------------

   close figure 1
```

```
   plot2D3D(PL,ax,ang,pl);
   figure(1),
   title('PRESSURE FIELD 3D BOTTOM PAD','units','normalized',...
       'FontWeight','bold','color',[0 0.3 0.6]);
   figure(2),
   title('PRESSURE FIELD 2D BOTTOM PAD','units','normalized',...
       'FontWeight','bold','color',[0 0.3 0.6]);
elseif sum(pl)==0
   close figure 1
   ang(1,1:inode) = gamma1*180/pi:(gamma2-gamma1)/(inode-
1)*180/pi:gamma2*180/pi;
   plot(ang,pl,'b.:')
   title('PRESSURE FIELD 2D BOTTOM PAD','units','normalized',...
       'FontWeight','bold','color',[0 0.3 0.6]);
   text(15,0.3,'The bottom pad is unloaded');
   xlabel('Theta (Degrees)','units','normalized','color',[0 0.3 0.6]);
   ylabel('Pressurè,'units','normalized','color',[0 0.3 0.6]);
   grid on;
end;
disp('INPUT DATA:');
fprintf('Eccentricity = %g\n',epsilon);
fprintf('L/D ratio = %g\n',lambda);
fprintf('Attitude angle = %g\n',beta*180/pi);
fprintf('Bottom pad angular extension = %g\n',gammalow*180/pi);

% BEGIN OF CALCULATIONS FOR THE TOP PAD ----------------------------------
% ------------------------------------------------------------------------

pausehour = fix(clock);

prompt    = {'Angular Extension of the top pad:'};
figTitle  = 'GEOMETRY OF THE BEARING - Top Pad';
lineno    = 1;
def       = {'170'};
answer    = inputdlg(prompt,figTitle,lineno,def);
```

```
if size(answer)==0, %program terminated
   set(tbp,'Valuè,get(tbp,'Min'));
   set([tbv tbs tbp tbt tbd tbg tba tbinfo tbclose],'Enablè,'On');
   break;
end;
[gammaup] = deal(answer{:});
gammaup   = str2num(gammaup);
gamma1    = 180+(180-gammaup)/2;
gamma2    = gamma1+gammaup;
gammaup   = gammaup*pi/180;
gamma1    = gamma1*pi/180;
gamma2    = gamma2*pi/180;

prompt = {'Number of nodes along radial coordinate:',
        'Number of nodes along axial coordinate (odd number):',
        'Initial value of iteration parameter k:',
        'm parameter:',
        'Terminating value of residual for iter. to find pressure:',
        'Maximum number of cycles during iter. to find pressure:'};
figTitle  = 'CALCULATIONS PARAMETERS - Top Pad';
lineno    = 1;
def       = {'31','11','3','2','0.00000001','100'};
answer    = inputdlg(prompt,figTitle,lineno,def);
if size(answer)==0, %program terminated
   set(tbp,'Valuè,get(tbp,'Min'));
   set([tbv tbs tbp tbt tbd tbg tba tbinfo tbclose],'Enablè,'On');
   break;
end;
[inode,jnode,k,m,tolerance,number_of_iterationslim] = deal(answer{:});
inode                = str2num(inode);
jnode                = str2num(jnode);
m                    = str2num(m);
tolerance            = str2num(tolerance);
k                    = str2num(k);
number_of_iterationslim = str2num(number_of_iterationslim);
```

```
restarthour = fix(clock);

k1   = zeros(inode);   k2 = zeros(inode);
fc   = zeros(inode,1); fs = zeros(inode,1);

for i=1:inode-1
   tetai        = pi+(pi-gammalow)/2+(gamma2-gamma1)/(inode-1)*(i-1);
   tetai1       = pi+(pi-gammalow)/2+(gamma2-gamma1)/(inode-1)*i;
   dteta        = (tetai1-tetai)/(501-1);
   L1(1,1)      = 1-(teta-tetai)/(tetai1-tetai);
   L1(2,1)      = 1-L1(1,1);
   TETA(1:501,1) = tetai:(tetai1-tetai)/(501-1):tetai1;

   Z   = h^(3-2*m)*[diff(L1)-(m*(diff(h))/h)*(L1)]*...
       [diff(L1)-(m*(diff(h))/h)*(L1)]';
   k1a = zeros(2);
   for counter1=1:2,
      for counter2=1:2,
         W = subs(Z(counter1,counter2),teta,TETA);
         for counter3=1:501-1,
           c                  = (W(counter3)+W(counter3+1))*dteta/2;
           k1a(counter1,counter2) = k1a(counter1,counter2)+c;
         end;
      end;
   end;
   k1(i,i)     = k1(i,i)+k1a(1,1);
   k1(i+1,i)   = k1(i+1,i)+k1a(2,1);
   k1(i,i+1)   = k1(i,i+1)+k1a(1,2);
   k1(i+1,i+1) = k1(i+1,i+1)+k1a(2,2);

   Z   = h^(3-2*m)*L1*L1';
   k2a = zeros(2);
   for counter1=1:2,
      for counter2=1:2,
```

```
      W = subs(Z(counter1,counter2),teta,TETA);
      for counter3=1:501-1,
        c                  = (W(counter3)+W(counter3+1))*dteta/2;
        k2a(counter1,counter2) = k2a(counter1,counter2)+c;
      end;
   end;
end;
k2(i,i)     = k2(i,i)+k2a(1,1);
k2(i+1,i)   = k2(i+1,i)+k2a(2,1);
k2(i,i+1)   = k2(i,i+1)+k2a(1,2);
k2(i+1,i+1) = k2(i+1,i+1)+k2a(2,2);

Z   = h^(-m)*L1*cos(teta);
fca = zeros(2,1);
for counter1=1:2,
   W = subs(Z(counter1),teta,TETA);
   for counter2=1:501-1,
      c           = (W(counter2)+W(counter2+1))*dteta/2;
      fca(counter1) = fca(counter1)+c;
   end;
end;
fc(i,1)   = fc(i,1)+fca(1,1);
fc(i+1,1) = fc(i+1,1)+fca(2,1);

Z   = h^(-m)*L1*sin(teta);
fsa = zeros(2,1);
for counter1=1:2,
   W = subs(Z(counter1),teta,TETA);
   for counter2=1:501-1,
      c           = (W(counter2)+W(counter2+1))*dteta/2;
      fsa(counter1) = fsa(counter1)+c;
   end;
end;
fs(i,1)   = fs(i,1)+fsa(1,1);
fs(i+1,1) = fs(i+1,1)+fsa(2,1);
```

```
end

% CALCULATIONS FOR K MATRIX, f VECTOR AND L,U MATRICES ----------
-----

[K,f,L,U,c3] = pressure(lambda,k,k1,k2,bc,fc,bs,fs,inode);

% CALCULATIONS FOR p VECTOR ------------------------------------------

Y  = zeros(inode,1);
E  = zeros(inode,1);
pu = zeros(inode,1);
l  = inode;

E(1) = -f(1);
for i=2:inode,
   E(i) = L(i,i-1)*Y(i-1)-f(i);
   if E(i)<0
      Y(i) = -E(i)/L(i,i);
   else
      l = i-1;
      break
   end
end
pu(l) = Y(l);
for j=l-1:-1:1
   pu(j) = Y(j)-U(j,j+1)*pu(j+1);
end
w           = h^(-m)*(cosh(k*lambda)-1);
THETA       = zeros(1,inode);
THETA(1:inode) = gamma1:(gamma2-gamma1)/(inode-1):gamma2;
W           = subs(w,teta,THETA);
pu          = pu.*W';

% ITERATION PROCESS ---------------------------------------------------
```

```
if sum(pu)~=0
   [pu,number_of_iterations,k,K,L,U,l,fxMolt,fyMolt] = iteration(k2,...
                                 k1,pl,lambda,...
                                 h,m,gamma1,gamma2,k,bc,fc,...
                                 bs,fs,inode,tolerance,c3,...
                                 number_of_iterationslim);
   [PU,ax,ang]                    = buildP(pu,lambda,inode,...
                                 jnode,gamma1,gamma2,k);

   % OIL FILM FORCES ----------------------------------------------------

%     [fx,fy] = forces(PU,fc,fs,c3,l,inode,jnode,gamma1,gamma2,lambda,pu);

   % 3D-2D PLOTS --------------------------------------------------------

   plot2D3D(PU,ax,ang,pu);
   figure(gcf-1),
   title('PRESSURE FIELD 3D TOP PAD','units','normalized',...
        'FontWeight','bold','color',[0 0.3 0.6]);
   figure(gcf+1),
   title('PRESSURE FIELD 2D TOP PAD','units','normalized',...
        'FontWeight','bold','color',[0 0.3 0.6]);
elseif sum(pu)==0
   figure
   ang           = zeros(1,inode);
   ang(1,1:inode) = gamma1*180/pi:(gamma2-gamma1)/(inode-1)*180/pi:...
                gamma2*180/pi;
   plot(ang,pu,'b.:')
   title('PRESSURE FIELD 2D TOP PAD','units','normalized',...
        'FontWeight','bold','color',[0 0.3 0.6]);
   text(205,0.3,'The top pad is unloaded');
   xlabel('Theta (Degrees)','units','normalized','color',[0 0.3 0.6]);
   ylabel('Pressurè,'units','normalized','color',[0 0.3 0.6]);
   grid on;
```

```
end;

finishhour    = fix(clock);
totalcomptime = pausehour-starthour+(finishhour-restarthour);

fprintf('Top pad angular extension = %g\n',gammaup*180/pi);
fprintf('Radial velocity = %g\n',edot);
fprintf('Tangential velocity = %g\n\n',epsidot);

disp('OUTPUT DATA:');
fprintf('fx = %g\n\n',fxMolt)
fprintf('fy = %g\n\n',fyMolt)

disp ('Finishing date/hour:'); disp(fix(clock));
disp('Total computational time:'), disp(totalcomptime);

% The RESTART button ----------------------------------------------------

tbinfo = uicontrol('Stylè,'pushbutton', ...
                   'Units','normalized', ...
                   'Position',...
                   [0.001 0.001 0.1 1.5*0.06],...
                   'BackgroundColor',[0.2 0.2 0.2], ...
                   'ForegroundColor',[1 1 1], ...
                   'FontWeight','bold', ...
                   'String','RESTART', ...
                   'Callback',str2mat('tribology_userinterfacè));
```

pressurefieldunsteadyelliptical.m

```
% ------------------------------------------------------------------
%          Forces on Bearings during the transient of a rotor
%          program: Pressure Field Unsteady Conditions
%            ELLIPTICAL BEARING ~ COMPLETE BEARING
%                  BOTTOM PAD + TOP PAD
%
% ------------------------------------------------------------------

clear all;
disp...
  ('------ Pressure Field Unsteady Conditions ~ Elliptical bearing ------')
cla reset; echo off;
global tbv tbs tbp tbt tbd tbg tba tbinfo tbclose;
set([tbv tbs tbp tbt tbd tbg tba tbinfo tbclose],'Enablè,'off');

% BEGIN OF INPUT DATA ---------------------------------------------------
% ------------------------------------------------------------------

prompt = {'L/D ratio:',
        'Angular Extension of the bottom pad:',
        'Eccentricity ratio:',
        'Ellipticity:',
        'Attitude angle (in degrees):',
        'Load inclination angle (in degrees):'
        'Radial Velocity:',
        'Tangential Velocity:'};
figTitle  = 'GEOMETRY OF THE BEARING - Bottom Pad';
lineno    = 1;
def       = {'1','170','0.8','0.5','30','0','5','5'};
answer    = inputdlg(prompt,figTitle,lineno,def);
if size(answer)==0, %program terminated
   set(tbp,'Valuè,get(tbp,'Min'));
   set([tbv tbs tbp tbt tbd tbg tba tbinfo tbclose],'Enablè,'On');
```

```
   break;
end;
[lambda,gammalow,epsilon,delta,beta,fi,edot,epsidot] = deal(answer{:});
lambda   = str2num(lambda);
gammalow = str2num(gammalow);
epsilon  = str2num(epsilon);
delta    = str2num(delta);
beta     = str2num(beta);
fi       = str2num(fi);
edot     = str2num(edot);
epsidot  = str2num(epsidot);
beta     = (beta+fi)*pi/180;
gamma1   = (180-gammalow)/2;
gamma2   = gamma1+gammalow;
gammalow = gammalow*pi/180;
gamma1   = gamma1*pi/180;
gamma2   = gamma2*pi/180;

prompt = {'Number of nodes along radial coordinate:',
        'Number of nodes along axial coordinate (odd number):',
        'Initial value of iteration parameter k:',
        'm parameter:',
        'Terminating value of residual for iter. to find pressure:',
        'Maximum number of cycles during iter. to find pressure:'};
figTitle  = 'CALCULATIONS PARAMETERS - Bottom Pad';
lineno    = 1;
def       = {'31','11','3','2','0.00000001','100'};
answer    = inputdlg(prompt,figTitle,lineno,def);
if size(answer)==0, %program terminated
   set(tbp,'Valuè,get(tbp,'Min'));
   set([tbv tbs tbp tbt tbd tbg tba tbinfo tbclose],'Enablè,'On');
   break;
end;
[inode,jnode,k,m,tolerance,number_of_iterationslim] = deal(answer{:});
inode                = str2num(inode);
```

```
jnode                 = str2num(jnode);
m                     = str2num(m);
tolerance             = str2num(tolerance);
k                     = str2num(k);
number_of_iterationslim = str2num(number_of_iterationslim);

starthour = fix(clock);
disp ('Starting date/hour:'); disp(fix(clock));

subplot(1,1,1);
text('units','normalized','position',[0.27 0.55],'FontWeight','bold',...
    'color',[0 0.3 0.6],'string','CALCULATIONS IN PROGRESS');

% BEGIN OF CALCULATIONS FOR THE BOTTOM PAD ------------------------------
% ----------------------------------------------------------------------

% CALCULATIONS FOR K1,K2 MATRICES AND fc,fs VECTORS ----------------------
-

teta     = sym('tetà,'real','d');
epsilonb = sqrt(epsilon^2+delta^2+2*epsilon*delta*cos(beta-pi/2));
betab    = asin((epsilon/epsilonb)*sin(beta-pi/2))+pi/2;
h        = 1-epsilonb*cos(teta-betab);
x        = epsilonb*cos(betab);
y        = epsilonb*sin(betab);
xdot     = edot*cos(betab)-epsidot*sin(betab);
ydot     = edot*sin(betab)+epsidot*cos(betab);
bc       = 2*xdot+y;
bs       = 2*ydot-x;
k1       = zeros(inode);   k2 = zeros(inode);
fc       = zeros(inode,1); fs = zeros(inode,1);

for i=1:inode-1
   tetai         = (pi-gammalow)/2+(gamma2-gamma1)/(inode-1)*(i-1);
   tetai1        = (pi-gammalow)/2+(gamma2-gamma1)/(inode-1)*i;
```

```
dteta        = (tetai1-tetai)/(501-1);
L1(1,1)      = 1-(teta-tetai)/(tetai1-tetai);
L1(2,1)      = 1-L1(1,1);
TETA(1:501,1) = tetai:(tetai1-tetai)/(501-1):tetai1;

Z   = h^(3-2*m)*[diff(L1)-(m*(diff(h))/h)*(L1)]*...
    [diff(L1)-(m*(diff(h))/h)*(L1)]';
k1a = zeros(2);
for counter1=1:2,
   for counter2=1:2,
      W = subs(Z(counter1,counter2),teta,TETA);
      for counter3=1:501-1,
        c                  = (W(counter3)+W(counter3+1))*dteta/2;
        k1a(counter1,counter2) = k1a(counter1,counter2)+c;
      end;
   end;
end;
k1(i,i)     = k1(i,i)+k1a(1,1);
k1(i+1,i)   = k1(i+1,i)+k1a(2,1);
k1(i,i+1)   = k1(i,i+1)+k1a(1,2);
k1(i+1,i+1) = k1(i+1,i+1)+k1a(2,2);

Z   = h^(3-2*m)*L1*L1';
k2a = zeros(2);
for counter1=1:2,
   for counter2=1:2,
      W = subs(Z(counter1,counter2),teta,TETA);
      for counter3=1:501-1,
        c                  = (W(counter3)+W(counter3+1))*dteta/2;
        k2a(counter1,counter2) = k2a(counter1,counter2)+c;
      end;
   end;
end;
k2(i,i)     = k2(i,i)+k2a(1,1);
k2(i+1,i)   = k2(i+1,i)+k2a(2,1);
```

```
   k2(i,i+1)   = k2(i,i+1)+k2a(1,2);
   k2(i+1,i+1) = k2(i+1,i+1)+k2a(2,2);

   Z   = h^(-m)*L1*cos(teta);
   fca = zeros(2,1);
   for counter1=1:2,
      W = subs(Z(counter1),teta,TETA);
      for counter2=1:501-1,
         c           = (W(counter2)+W(counter2+1))*dteta/2;
         fca(counter1) = fca(counter1)+c;
      end;
   end;
   fc(i,1)   = fc(i,1)+fca(1,1);
   fc(i+1,1) = fc(i+1,1)+fca(2,1);

   Z   = h^(-m)*L1*sin(teta);
   fsa = zeros(2,1);
   for counter1=1:2,
      W = subs(Z(counter1),teta,TETA);
      for counter2=1:501-1,
         c           = (W(counter2)+W(counter2+1))*dteta/2;
         fsa(counter1) = fsa(counter1)+c;
      end;
   end;
   fs(i,1)   = fs(i,1)+fsa(1,1);
   fs(i+1,1) = fs(i+1,1)+fsa(2,1);
end

% CALCULATIONS FOR K MATRIX, f VECTOR AND L,U MATRICES ----------
-----

[K,f,L,U,c3] = pressure(lambda,k,k1,k2,bc,fc,bs,fs,inode);

% CALCULATIONS FOR p VECTOR ------------------------------------------
```

```
Y  = zeros(inode,1);
E  = zeros(inode,1);
pl = zeros(inode,1);
l  = inode;

E(1) = -f(1);
for i=2:inode,
   E(i) = L(i,i-1)*Y(i-1)-f(i);
   if E(i)<0
      Y(i) = -E(i)/L(i,i);
   else
      l = i-1;
      break
   end
end
pl(l) = Y(l);
for j=l-1:-1:1
   pl(j) = Y(j)-U(j,j+1)*pl(j+1);
end
w            = h^(-m)*(cosh(k*lambda)-1);
THETA(1:inode) = gamma1:(gamma2-gamma1)/(inode-1):gamma2;
W            = subs(w,teta,THETA);
pl           = pl.*W';

% ITERATION PROCESS ----------------------------------------------------

if sum(pl)~=0
   [pl,number_of_iterations,k,K,L,U,l,fxMoltB,fyMoltB] = iteration(k2,...
                                 k1,pl,lambda,...
                                 h,m,gamma1,gamma2,k,bc,fc,...
                                 bs,fs,inode,tolerance,c3,...
                                 number_of_iterationslim);
   [PL,ax,ang]                   = buildP(pl,lambda,inode,...
                                 jnode,gamma1,gamma2,k);
```

```
   % OIL FILM FORCES ----------------------------------------------------

%     [fx,fy] = forces(PL,fc,fs,c3,l,inode,jnode,gamma1,gamma2,lambda,pl);

   % 3D-2D PLOTS ----------------------------------------------------------

   close figure 1
   plot2D3D(PL,ax,ang,pl);
   figure(1),
   title('PRESSURE FIELD 3D BOTTOM PAD','units','normalized',...
        'FontWeight','bold','color',[0 0.3 0.6]);
   figure(2),
   title('PRESSURE FIELD 2D BOTTOM PAD','units','normalized',...
        'FontWeight','bold','color',[0 0.3 0.6]);
elseif sum(pl)==0
   close figure 1
   ang(1,1:inode) = gamma1*180/pi:(gamma2-gamma1)/(inode-1)*180/pi:...
                gamma2*180/pi;
   plot(ang,pl,'b.:')
   title('PRESSURE FIELD 2D BOTTOM PAD','units','normalized',...
        'FontWeight','bold','color',[0 0.3 0.6]);
   text(15,0.3,'The bottom pad is unloaded');
   xlabel('Theta (Degrees)','units','normalized','color',[0 0.3 0.6]);
   ylabel('Pressurè,'units','normalized','color',[0 0.3 0.6]);
   grid on;
end;
disp('INPUT DATA:');
fprintf('Eccentricity = %g\n',epsilon);
fprintf('L/D ratio = %g\n',lambda);
fprintf('Ellipticity = %g\n',delta);
fprintf('Attitude angle = %g\n',beta*180/pi);
fprintf('Attitude angle for the bottom pad = %g\n',betab*180/pi);
fprintf('Bottom pad angular extension = %g\n',gammalow*180/pi);

% BEGIN OF CALCULATIONS FOR THE TOP PAD ----------------------------------
```

```
% ------------------------------------------------------------------------

pausehour = fix(clock);

prompt    = {'Angular Extension of the top pad:'};
figTitle  = 'GEOMETRY OF THE BEARING - Top Pad';
lineno    = 1;
def       = {'170'};
answer    = inputdlg(prompt,figTitle,lineno,def);
if size(answer)==0, %program terminated
   set(tbp,'Valuè,get(tbp,'Min'));
   set([tbv tbs tbp tbt tbd tbg tba tbinfo tbclose],'Enablè,'On');
   break;
end;
[gammaup] = deal(answer{:});
gammaup   = str2num(gammaup);
gamma1    = 180+(180-gammaup)/2;
gamma2    = gamma1+gammaup;
gammaup   = gammaup*pi/180;
gamma1    = gamma1*pi/180;
gamma2    = gamma2*pi/180;

prompt = {'Number of nodes along radial coordinate:',
        'Number of nodes along axial coordinate (odd number):',
        'Initial value of iteration parameter k:',
        'm parameter:',
        'Terminating value of residual for iter. to find pressure:',
        'Maximum number of cycles during iter. to find pressure:'};
figTitle  = 'CALCULATIONS PARAMETERS - Top Pad';
lineno    = 1;
def       = {'31','11','3','2','0.00000001','100'};
answer    = inputdlg(prompt,figTitle,lineno,def);
if size(answer)==0, %program terminated
   set(tbp,'Valuè,get(tbp,'Min'));
   set([tbv tbs tbp tbt tbd tbg tba tbinfo tbclose],'Enablè,'On');
```

```
   break;
end;
[inode,jnode,k,m,tolerance,number_of_iterationslim] = deal(answer{:});
inode                = str2num(inode);
jnode                = str2num(jnode);
m                    = str2num(m);
tolerance            = str2num(tolerance);
k                    = str2num(k);
number_of_iterationslim = str2num(number_of_iterationslim);

restarthour = fix(clock);

epsilont = sqrt(epsilon^2+delta^2-2*epsilon*delta*cos(beta-pi/2));
betat    = acos((epsilon/epsilont)*sin(beta-pi/2))+pi;
h        = 1-epsilont*cos(teta-betat);
x        = epsilont*cos(betat);
y        = epsilont*sin(betat);
xdot     = edot*cos(betat)-epsidot*sin(betat);
ydot     = edot*sin(betat)+epsidot*cos(betat);
bc       = 2*xdot+y;
bs       = 2*ydot-x;
k1       = zeros(inode);   k2 = zeros(inode);
fc       = zeros(inode,1); fs = zeros(inode,1);

k1   = zeros(inode);   k2 = zeros(inode);
fc   = zeros(inode,1); fs = zeros(inode,1);

for i=1:inode-1
   tetai        = pi+(pi-gammalow)/2+(gamma2-gamma1)/(inode-1)*(i-1);
   tetai1       = pi+(pi-gammalow)/2+(gamma2-gamma1)/(inode-1)*i;
   dteta        = (tetai1-tetai)/(501-1);
   L1(1,1)      = 1-(teta-tetai)/(tetai1-tetai);
   L1(2,1)      = 1-L1(1,1);
   TETA(1:501,1) = tetai:(tetai1-tetai)/(501-1):tetai1;
```

```
Z   = h^(3-2*m)*[diff(L1)-(m*(diff(h))/h)*(L1)]*...
    [diff(L1)-(m*(diff(h))/h)*(L1)]';
k1a = zeros(2);
for counter1=1:2,
   for counter2=1:2,
      W = subs(Z(counter1,counter2),teta,TETA);
      for counter3=1:501-1,
        c                  = (W(counter3)+W(counter3+1))*dteta/2;
        k1a(counter1,counter2) = k1a(counter1,counter2)+c;
      end;
   end;
end;
k1(i,i)     = k1(i,i)+k1a(1,1);
k1(i+1,i)   = k1(i+1,i)+k1a(2,1);
k1(i,i+1)   = k1(i,i+1)+k1a(1,2);
k1(i+1,i+1) = k1(i+1,i+1)+k1a(2,2);

Z   = h^(3-2*m)*L1*L1';
k2a = zeros(2);
for counter1=1:2,
   for counter2=1:2,
      W = subs(Z(counter1,counter2),teta,TETA);
      for counter3=1:501-1,
        c                  = (W(counter3)+W(counter3+1))*dteta/2;
        k2a(counter1,counter2) = k2a(counter1,counter2)+c;
      end;
   end;
end;
k2(i,i)     = k2(i,i)+k2a(1,1);
k2(i+1,i)   = k2(i+1,i)+k2a(2,1);
k2(i,i+1)   = k2(i,i+1)+k2a(1,2);
k2(i+1,i+1) = k2(i+1,i+1)+k2a(2,2);

Z   = h^(-m)*L1*cos(teta);
fca = zeros(2,1);
```

```
   for counter1=1:2,
      W = subs(Z(counter1),teta,TETA);
      for counter2=1:501-1,
         c           = (W(counter2)+W(counter2+1))*dteta/2;
         fca(counter1) = fca(counter1)+c;
      end;
   end;
   fc(i,1)   = fc(i,1)+fca(1,1);
   fc(i+1,1) = fc(i+1,1)+fca(2,1);

   Z   = h^(-m)*L1*sin(teta);
   fsa = zeros(2,1);
   for counter1=1:2,
      W = subs(Z(counter1),teta,TETA);
      for counter2=1:501-1,
         c           = (W(counter2)+W(counter2+1))*dteta/2;
         fsa(counter1) = fsa(counter1)+c;
      end;
   end;
   fs(i,1)   = fs(i,1)+fsa(1,1);
   fs(i+1,1) = fs(i+1,1)+fsa(2,1);
end

% CALCULATIONS FOR K MATRIX, f VECTOR AND L,U MATRICES ----------
-----

[K,f,L,U,c3] = pressure(lambda,k,k1,k2,bc,fc,bs,fs,inode);

% CALCULATIONS FOR p VECTOR ------------------------------------------

Y  = zeros(inode,1);
E  = zeros(inode,1);
pu = zeros(inode,1);
l  = inode;
```

```
E(1) = -f(1);
for i=2:inode,
   E(i) = L(i,i-1)*Y(i-1)-f(i);
   if E(i)<0
      Y(i) = -E(i)/L(i,i);
   else
      l = i-1;
      break
   end
end
pu(l) = Y(l);
for j=l-1:-1:1
   pu(j) = Y(j)-U(j,j+1)*pu(j+1);
end
w            = h^(-m)*(cosh(k*lambda)-1);
THETA        = zeros(1,inode);
THETA(1:inode) = gamma1:(gamma2-gamma1)/(inode-1):gamma2;
W            = subs(w,teta,THETA);
pu           = pu.*W';

% ITERATION PROCESS ----------------------------------------------------

if sum(pu)~=0
   [pu,number_of_iterations,k,K,L,U,l,fxMoltT,fyMoltT] = iteration(k2,...
                                 k1,pl,lambda,...
                                 h,m,gamma1,gamma2,k,bc,fc,...
                                 bs,fs,inode,tolerance,c3,...
                                 number_of_iterationslim);
   [PU,ax,ang]                   = buildP(pu,lambda,inode,...
                                 jnode,gamma1,gamma2,k);

   % OIL FILM FORCES ----------------------------------------------------

%     [fx,fy] = forces(PU,fc,fs,c3,l,inode,jnode,gamma1,gamma2,lambda,pu);
```

```
   % 3D-2D PLOTS ----------------------------------------------------------

   plot2D3D(PU,ax,ang,pu);
   figure(gcf-1),
   title('PRESSURE FIELD 3D TOP PAD','units','normalized',...
        'FontWeight','bold','color',[0 0.3 0.6]);
   figure(gcf+1),
   title('PRESSURE FIELD 2D TOP PAD','units','normalized',...
        'FontWeight','bold','color',[0 0.3 0.6]);
elseif sum(pu)==0
   figure
   ang          = zeros(1,inode);
   ang(1,1:inode) = gamma1*180/pi:(gamma2-gamma1)/(inode-1)*180/pi:...
                gamma2*180/pi;
   plot(ang,pu,'b.:')
   title('PRESSURE FIELD 2D TOP PAD','units','normalized',...
        'FontWeight','bold','color',[0 0.3 0.6]);
   text(205,0.3,'The top pad is unloaded');
   xlabel('Theta (Degrees)','units','normalized','color',[0 0.3 0.6]);
   ylabel('Pressurè,'units','normalized','color',[0 0.3 0.6]);
   grid on;
end;

finishhour    = fix(clock);
totalcomptime = pausehour-starthour+(finishhour-restarthour);

fprintf('Attitude angle for the top pad = %g\n',betat*180/pi);
fprintf('Top pad angular extension = %g\n',gammaup*180/pi);
fprintf('Radial velocity = %g\n',edot);
fprintf('Tangential velocity = %g\n\n',epsidot);

disp('OUTPUT DATA:');
fprintf('fx = %g\n',fxMoltT+fxMoltB)
fprintf('fy = %g\n\n',fyMoltT+fyMoltB)
```

```
disp ('Finishing date/hour:');     disp(fix(clock));
disp('Total computational time:'), disp(totalcomptime);

% The RESTART button ----------------------------------------------------

tbinfo = uicontrol('Stylè,'pushbutton', ...
                'Units','normalized', ...
                'Position',...
                [0.001 0.001 0.1 1.5*0.06],...
                'BackgroundColor',[0.2 0.2 0.2], ...
                'ForegroundColor',[1 1 1], ...
                'FontWeight','bold', ...
                'String','RESTART', ...
                'Callback',str2mat('tribology_userinterfacè));
```

pressure.m

```
% ------------------------------------------------------------------
%          Forces on Bearings during the transient of a rotor
%             program: Pressure
%             Output: K,f,L,U,c3
%
%
% ------------------------------------------------------------------

function [K,f,L,U,c3]=pressure(lambda,k,k1,k2,bc,fc,bs,fs,inode)

c1 = 2*lambda+lambda*cosh(2*k*lambda)-3/(2*k)*sinh(2*k*lambda);
c2 = (k/2)*[sinh(2*k*lambda)-2*k*lambda];
c3 = (2/k)*[k*lambda*cosh(k*lambda)-sinh(k*lambda)];

K = zeros(inode);
f = zeros(inode,1);

K = c1*k1+c2*k2;
f = c3*(bc*fc+bs*fs);
L = zeros(inode);
U = zeros(inode);
y = zeros(inode,1);

L(1,1) = K(1,1);

for i=2:inode
   L(i,i-1) = K(i,i-1);
end

for i=1:inode
   U(i,i) = 1;
end

for i=2:inode
   U(i-1,i) = K(i-1,i)/L(i-1,i-1);
   L(i,i)   = K(i,i)-U(i-1,i)*L(i,i-1);
end
```

iteration.m

```
% ------------------------------------------------------------------
%          Forces on Bearings during the transient of a rotor
%             program: Iteration
%             Output: p vector, number of cycles done
%
%
% ------------------------------------------------------------------

function [p,number_of_iterations,k,K,L,U,l,fxMolt,fyMolt]=iteration(k2,k1,p,lambda,...
        h,m,gamma1,gamma2,k,bc,fc,bs,fs,inode,tolerance,c3,...
        number_of_iterationslim)

residual            = 1;
number_of_iterations = 0;
teta                = sym('tetà);
while (residual>tolerance) & (number_of_iterations<=number_of_iterationslim)
   d1 = p'*k1*p;
   d2 = p'*k2*p;
   k  = sqrt(d1/d2);

   [K,f,L,U,c3] = pressure(lambda,k,k1,k2,bc,fc,bs,fs,inode);

   Y  = zeros(inode,1);
   E  = zeros(inode,1);
   p2 = zeros(inode,1);
   l  = inode;
   E(1) = -f(1);
   for i=2:inode,
      E(i) = L(i,i-1)*Y(i-1)-f(i);
      if E(i)<0
         Y(i) = -E(i)/L(i,i);
      else
         l = i-1;
         break
      end
   end
   p2(l) = Y(l);
   for j=l-1:-1:1
      p2(j) = Y(j)-U(j,j+1)*p2(j+1);
```

```
   end
   residual            = (sum(p-p2)/sum(p));
   p                   = p2;
   number_of_iterations = number_of_iterations+1;
end;
fxMolt = (c3*fc'*p);
fyMolt = (c3*fs'*p);

w             = h^(-m)*(cosh(k*lambda)-1);
THETA(1:inode) = gamma1:(gamma2-gamma1)/(inode-1):gamma2;
W             = subs(w,teta,THETA);
p             = p.*W';
```

plot2D3D.m

```
function plot2D3D(P,ax,ang,p)
% subplot (2,1,1)
figure()
mesh(ax,ang,P);
% title('PRESSURE FIELD 3D','units','normalized','FontWeight','bold','color',[0 0.3 0.6]);
xlabel('Psi','units','normalized','color',[0 0.3 0.6]);
ylabel('Theta (Degrees)','units','normalized','color',[0 0.3 0.6]);
zlabel('Pressurè,'units','normalized','color',[0 0.3 0.6]);
figure()
% subplot (2,1,2);
plot(ang,p,'b.:');
% title('PRESSURE FIELD 2D','units','normalized','FontWeight','bold','color',[0 0.3 0.6]);
xlabel('Theta (Degrees)','units','normalized','color',[0 0.3 0.6]);
ylabel('Pressurè,'units','normalized','color',[0 0.3 0.6]);
grid on
```

forces.m

```
function [fx,fy]=forces(P,fc,fs,c3,l,inode,jnode,gamma1,gamma2,lambda,p)

gamma=gamma2-gamma1;
deltateta = gamma/(inode-1);
deltapsi  = 1/(jnode-1);
for i=1:inode,
      SUMY(i) = 0;
      for j=2:jnode,
         SUMY(i) = SUMY(i)+P(i,j)+P(i,j-1);
      end;
      SUMY(i) = SUMY(i)*0.5*deltapsi;
end;
fx = 0;
fy = 0;
   for i=2:inode
      x  = (i-1)*deltateta+pi-0.5*gamma;
      x2 = (i-2)*deltateta+pi-0.5*gamma;
      fy = fy-cos(x)*SUMY(i)-cos(x2)*SUMY(i-1);
      fx = fx+sin(x)*SUMY(i)+sin(x2)*SUMY(i-1);
   end;
fx = fx*deltateta*0.5;
fy = fy*deltateta*0.5;
```

Chapter 5

Results

The following charts and table refer to a circular bearing, in steady conditions.

It has been used a relatively thick mesh (341 x 101 nodes) in order to show the differences between a Finite Elements method and the variational approach, especially in terms of computational timing. Other minor parameters regarding the geometry of the bearing (such as angular extension, axial extension, etc) have not been modified in order to get easier comparation between the results.

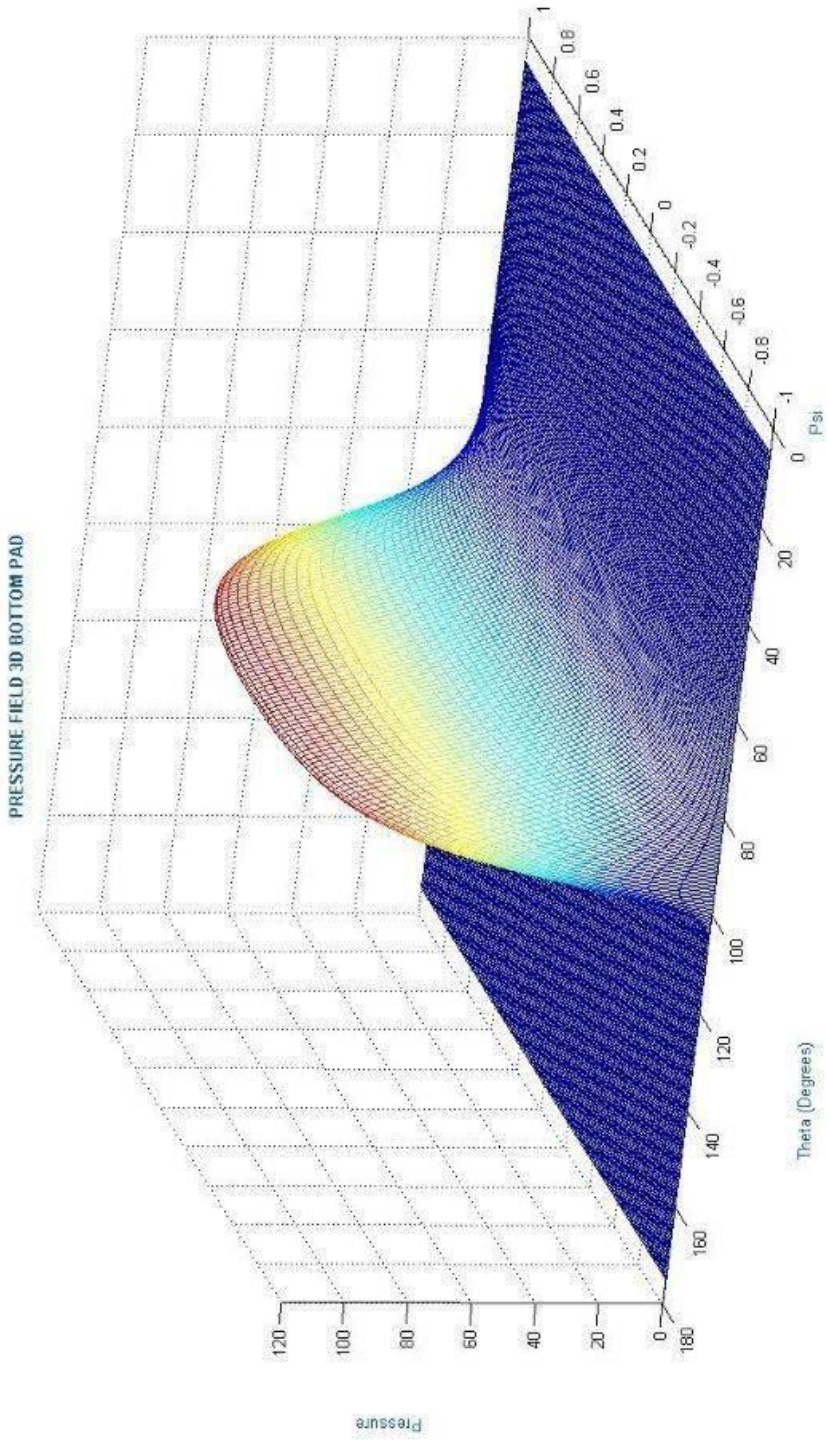
Graph 5.1 is a 3D representation of the pressure field along the radial coordinate and the axial coordinate.
As formerly defined, axial distribution is a hyperbolic cosine interpolation. This approximation gaves good results (comparing it with a 3D representation obtained with a FEM algorithm) and does not affect the final results in terms of forces.

Regarding the pressure distribution, shown in Graph 5.2, both the two curves nearly collapse in one, especially for extremely high values of eccentricity. The fact that the variational approach curve peak is slightly higher than the FEM curve peak is due to the chasing process described in Chapter 3.5, which increases a bit the results. Even in this case, this does not affect the final results in terms of forces.
The fact that the two curves are almost as one naturally leads to a very reduced percentage error in terms of force.

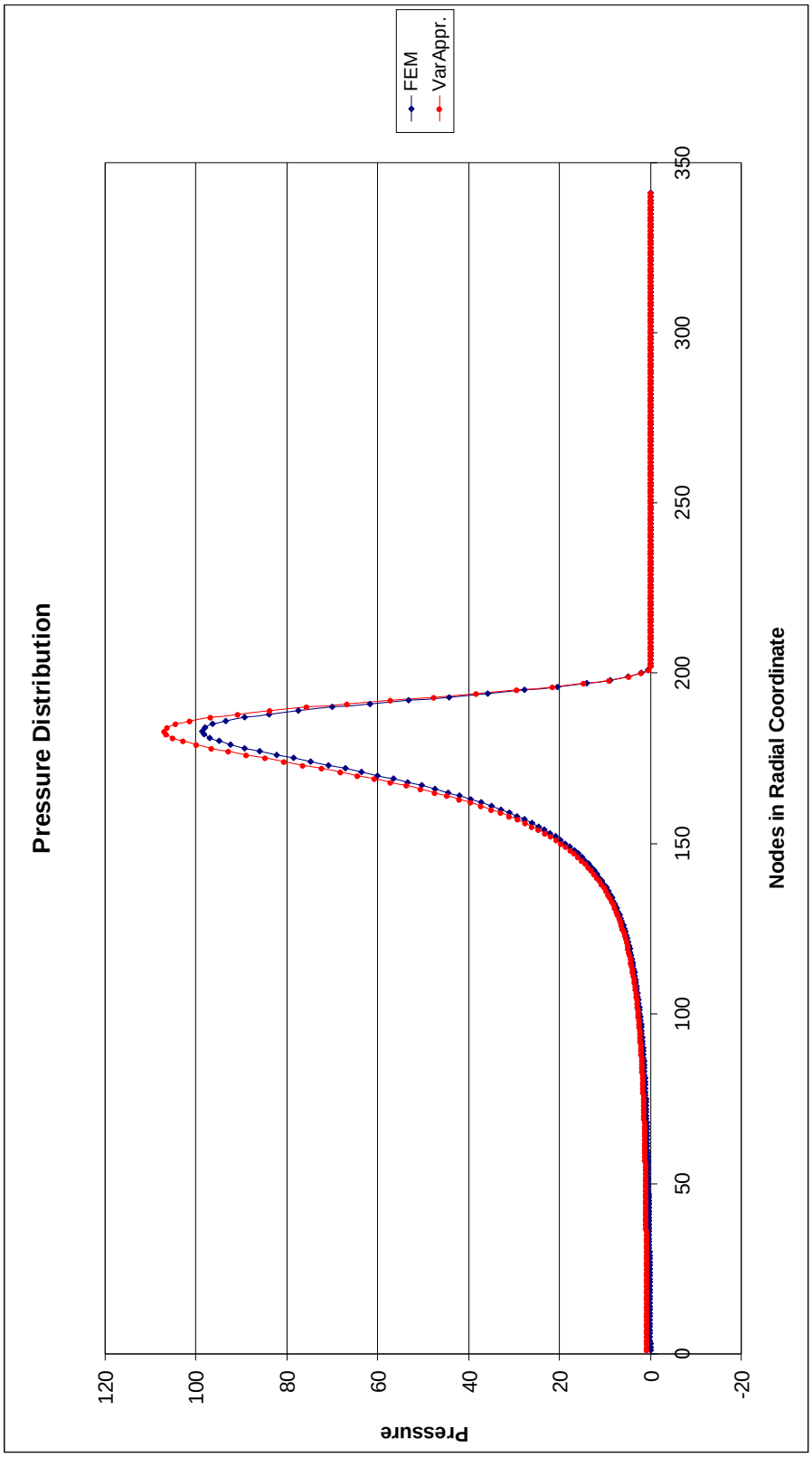
As clearly shown in Graph 5.3, the force distribution obtained with the new algorithm almost coincide with the one obtained with a traditional Finite Elements method. The two lines are slightly different for lower values of eccentricity. Infact, variational approach gets really accurate when eccentricity is very high – almost 1.

Main difference lies in the computational time (last 3 columns of table 5.1), which is almost 10 times reduced.

Table 5.1 also shows how the percentage error decreases by augmenting the eccentricity. Percentage error has been calculated doing the ratio between the value of F (square root of Fx square plus Fy square) evaluated with the variational approach and the value of F evaluated with Finite Elements method as written. As the eccentricity parameter goes down toward lower values, percentage error rises: this is clear in Graph 5.4, where the curve reachs a minimum in E = 0.985.
For this value of the eccentricity parameter, the percentage error is almost 0 % (0.35%) and time reduction is 938%.



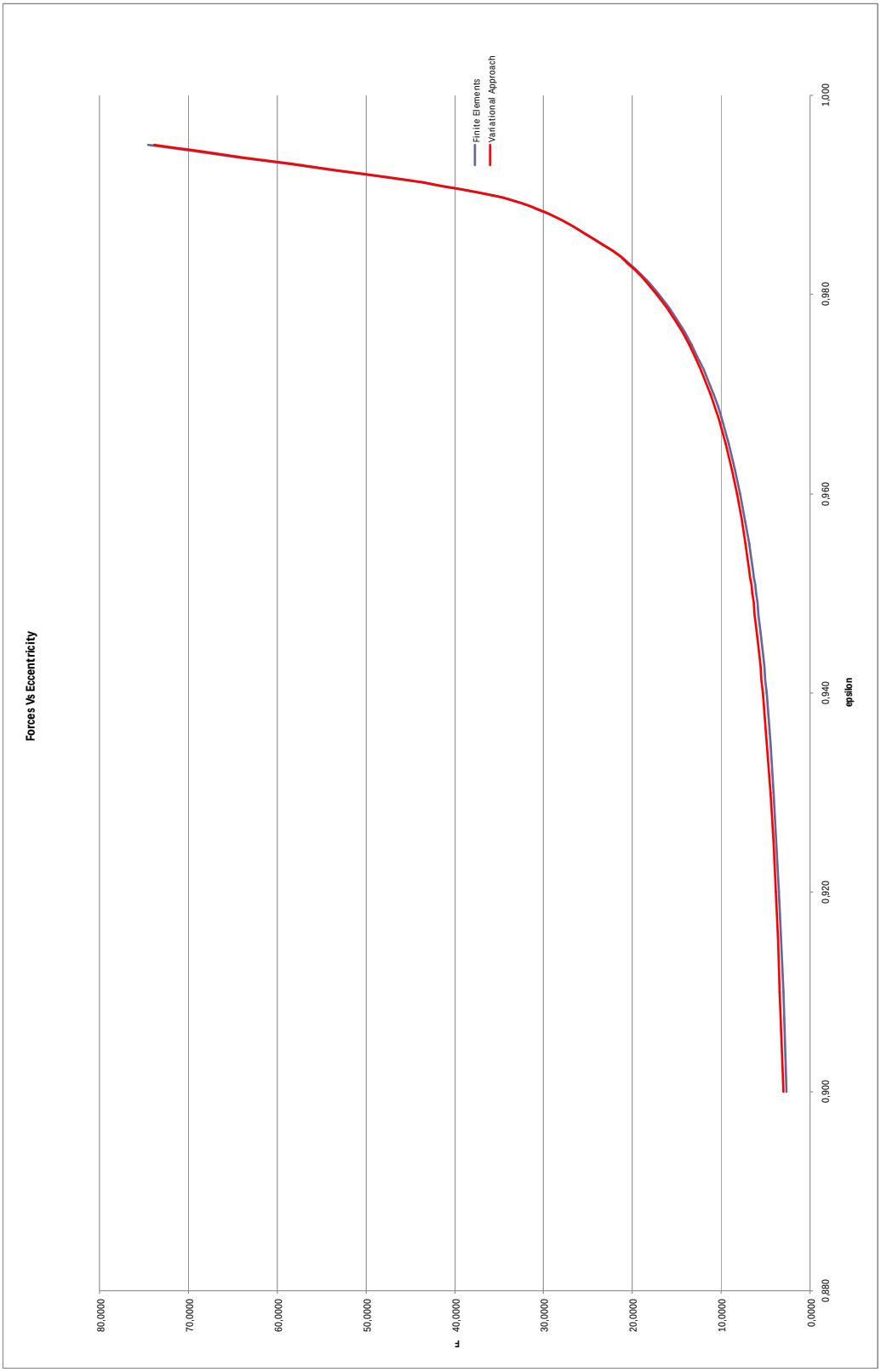
Graph 5.1 – Pressure Distribution 3D. E=0.995, L/D=1, γ=170°, β=96,6129°



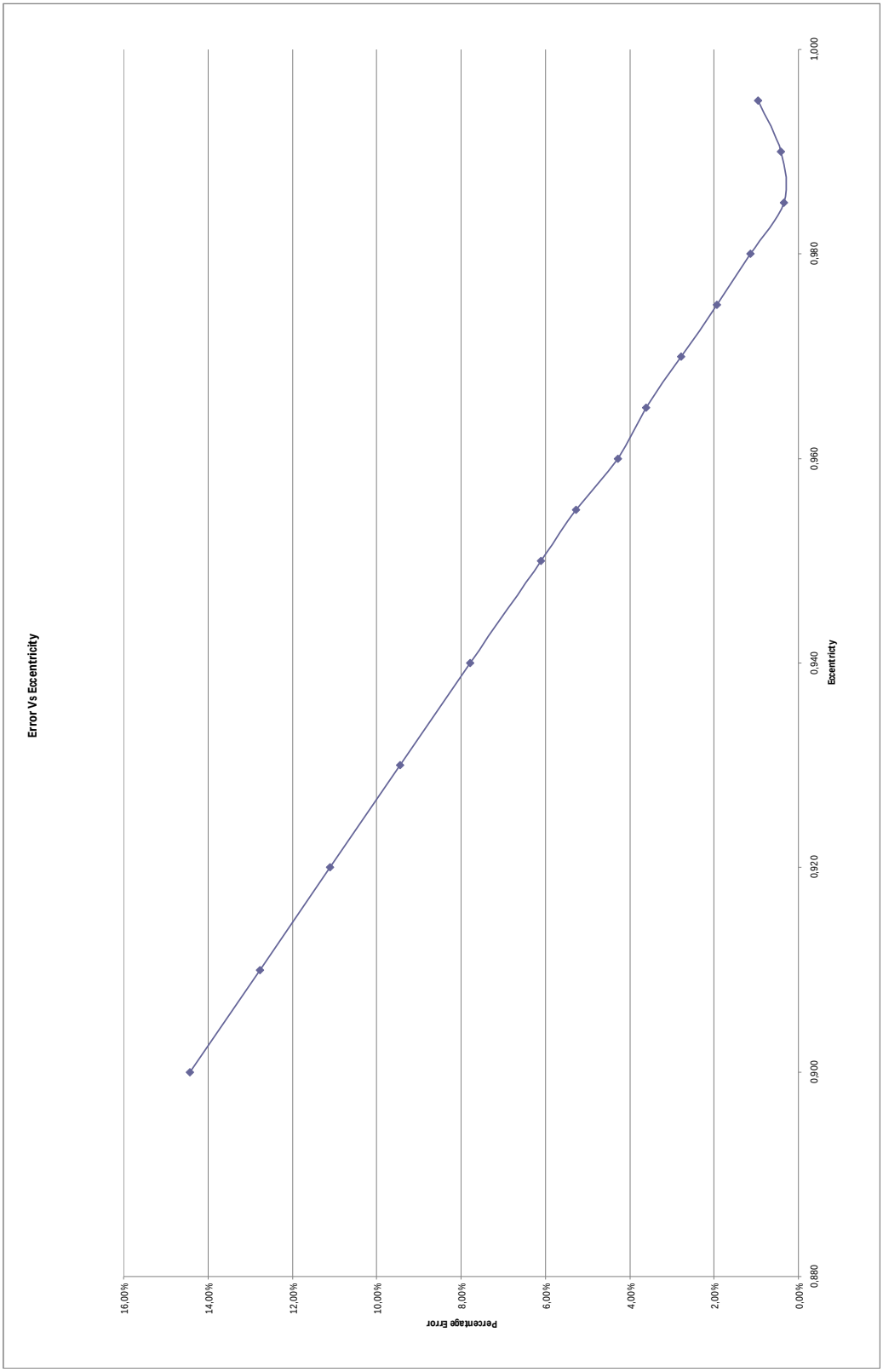
Graph 5.2 – Pressure Distribution 2D – middle line. E=0.995, L/D=1, γ=170°, β=96,6129°

Eps	Att. Ang.	Fx_FEM	Fy_FEM	F_FEM	Fx_Var.Appr.	Fy_Var.Appr.	F_Var.Appr.	Err.	T_FEM	T_Var.Appr.	Delta T
0,900	113,4760	1,30E-03	2,6219	2,6219	0,267052	2,98834	3,0002	14,43%	330	32	931%
0,910	112,4977	1,40E-03	2,9975	2,9975	0,282952	3,36847	3,3803	12,77%	331	33	903%
0,920	111,4545	1,50E-03	3,4718	3,4718	0,300326	3,84592	3,8576	11,11%	329	32	928%
0,930	110,3293	1,60E-03	4,088	4,0880	0,319592	4,46267	4,4741	9,44%	326	32	919%
0,940	109,1020	1,70E-03	4,9183	4,9183	0,340631	5,28999	5,3009	7,78%	336	32	950%
0,950	107,7398	1,80E-03	6,0941	6,0941	0,3637	6,45567	6,4659	6,10%	330	32	931%
0,955	106,9947	1,90E-03	6,8847	6,8847	0,375269	7,23861	7,2483	5,28%	330	32	931%
0,960	106,9532	1,90E-03	7,8803	7,8803	0,273024	8,21367	8,2182	4,29%	327	33	891%
0,965	105,3266	2,00E-03	9,169	9,1690	0,39912	9,49119	9,4996	3,61%	330	31	965%
0,970	104,3777	2,00E-03	10,9007	10,9007	0,409727	11,1957	11,2032	2,78%	331	32	934%
0,975	103,3226	2,10E-03	13,3461	13,3461	0,418429	13,5982	13,6046	1,94%	348	32	988%
0,980	102,1252	2,10E-03	17,0495	17,0495	0,421456	17,2388	17,2440	1,14%	331	31	968%
0,985	100,7225	2,10E-03	23,2913	23,2913	0,41554	23,3693	23,3730	0,35%	332	32	938%
0,990	98,9938	2,10E-03	35,9352	35,9352	-0,2458	35,7827	35,7835	0,42%	337	32	953%
0,995	96,6129	2,00E-03	74,5322	74,5322	0,301062	73,8125	73,8131	0,96%	333	32	941%

Table 5.1 – Forces and computational time. L/D=1, γ=170°



Graph 5.3 – Forces Vs Eccentricity. Circular bearing, L/D=1, γ=170°



Graph 5.4 – Error Vs Eccentricity

Conclusion

The whole work of code implementation gave good results in terms of computational time gaining and percentage error between FEM forces and Variational Approach forces.

As it has been shown in graphs and tables, computational time decreases quite a lot comparing it with a traditional FEM code. Infact, variational approach code is almost 10 times quicker than a Finite Elements program. As the user defines a thicker mesh for the study of the pressure field inside the bearing, the difference in terms of time between the variational approach and the Finite Elements program get higher. When using a mesh with a low number of nodes, the time spread between the two methods is not as good as in case of high number of nodes.
Naturally, is up to the user to define the thickness of the mesh (hence the computational time) depending on design needings.

On the other hand, speaking in terms of accuracy of the numbers obtained, the new code is not such as exact as the old one. This basically means that this new method can easily be applied at the beginning of the bearing design process, in order to obtain a quick and fair enough solution of the Reynolds equation.
Accuracy gets better and better as the eccentricity of the shaft rises. Infact, with high values of E (e.g. values included between 0.9 and 0.98), percentage error is near to 0 %. In these cases, variational approach is a lot better than the traditional method: the program combines high computational speed with high accuracy of the numerical results.

A natural enhancement for this work could be the study of the dynamic behaviour of a bearing in unsteady conditions. Infact, the variational approach method can be easily used for a quick evaluation of the forces in different conditions of eccentricity and shaft center speed, in order to make a further linearization and study the dynamic of the shaft inside the bearing.

References.

G.W. Stachowiak, A.W. Batchelor, Engineering Tribology, Butterworth Heinemann

Liping Wang, Study on Dynamic modeling and Behaviours of Tilting-Pad Bearings, PhD. Thesis 04/2007, Fudan University, Shanghai, China.

[1] Lubrication (Tribology) – Education and Research. A Report on the Present Position and Industry Needs (Jost report), Department of Education and Science, HM Stationary Office, London, 1966

[2] Research Report (T76-38) Tribologie (Code BMFT-FBT76-38), Bundesministerium Fur Forschung und Technologie (Federal Ministry for Research and Technology), West Germany, 1976

[3] Strategy for Energy Conservation Through Tribology, ASME, New York, November, 1977

[4] M.H. Jones and D. Scott (editors), Industrial Tribology, The practical Aspects of friction, Lubrication and Wear, Elsevier, Amsterdam, 1983

[5] G.P.K. Linkhammer and C.e. Lambert, Preservation of Organic Matter During Salinity Excursions, *Nature*, Vol. 339, 1989, pp. 271-274

[6] T. Gold, Terrestrial Sources of Carbon and Earthquake Outgassing, *Journal of Petroleum Geology*, Vol. I, 1979, pp. 3-19

[7] Barks publications, Inc. - http://www.barks.com/2002/02-04feat.html

[8] D. Fuller, Theory and Practice of Lubrication for Engineers, John Wiley & Sons, Inc, 1966.

[9] Pinkus, O and Sternlicht, B, Theory of Hydrodynamic Lubrication, McGraw-Hill Book company, Inc. 1961.

[10] N S Feng and E J Hahn, Computation of Bearing Characteristics: Elliptic Bearings and Tilting Pad Bearings, UNSW, 1993

[11] L.P. Wang, Study on Dynamic Modelling and Behaviours of Tilting-Pad Bearings, Department of Mechanics and Engineering Science, Fudan University, PhD Thesis, 2007

[12] H.L. Xi, Research on Some Problem of Non-linear Dynamical Behaviors of A Finite Journal Bearing-Rotor System, Sufen University, 2005

[13] The Institution of Mechanical Engineers, Journal Bearings for Reciprocating & Turbo Machinery, volume 181, part 3B, 1966

[14] O. Reynolds, On the Theory of Lubrication and its Application to Mr. Beauchamp Tower's Experiments Including an Experimental Determination of the Viscosity of Olive Oil, Phil. Trans., Roy. Soc. London, Vol. 177 (i), 1886, pp. 157-324.

[15] Rhode, S. M., and McAllister, G. T. A Variational Formulation for a Class of Free Boundary Problems Arising in Hydrodynamic Lubrication [J]. International J. of Engineering Science, 1975, 13: 841:850

[16] Kinderleher, D., and Stampacchia, G. An Introduction to Variational Inequalities and their Applications [M]. New York, Academic Press, 1980

[17] Zheng, T., and Hasebe, N. Calculation of Equilibrium Position and Dynamic Coefficients of a Journal Bearing Using Free Boundary Theory [J]. ASME J. of Tribology, 2000, 122(3): 616-621

[18] Zheng T., Zhang W., Ma J. and Yang S., Iteration Method for the Discretised Elliptical Calculus Variation Inequality, Applied Mathematics and Mechanics, 1995, 16 (4) pp. 329-335

[19] Zheng T., Li L. and Xu Q., Theory and Application of Non-linear Oil Film Forces of Sliding Journal Bearings,