FUNDAMENTAL STUDY ON A PARALLEL NUMERICAL ANALYSIS FOR PERFORMANCE PREDICTION OF AGRICULTURAL TIRE BY 3D FE-DE METHOD

Final Report on
Research Project (Number: 12660231)
under Grant-in-Aid for Scientific Research (C)
for 2000 to 2002
from
Japan Society for the Promotion of Science

Hiroshi NAKASHIMA
Associate Professor, Dr.
Division of Environmental Science & Technology
Graduate School of Agriculture
Kyoto University

March 2003

FUNDAMENTAL STUDY
ON
A PARALLEL NUMERICAL ANALYSIS
FOR PERFORMANCE PREDICTION
OF AGRICULTURAL TIRE
BY 3D FE-DE METHOD

Final Report on
Research Project (Number: 12660231)
under Grant-in-Aid for Scientific Research (C)
for 2000 to 2002
from
Japan Society for the Promotion of Science

Hiroshi NAKASHIMA
Associate Professor, Dr.
Division of Environmental Science & Technology
Graduate School of Agriculture
Kyoto University

March 2003

Contents

Chapter 1

Introduction

1.1 Remarks on Current Project

In terramechanics, there have been fundamental interaction problems between running devices and soil. Among others, the problem of wheel performance on a given terrain condition belongs to enthusiastic research subjects even now. However, the interaction is a typical contact problem whose mechanism is quite non-linear in nature. This non-linearity consists of two aspects; one is unknown contact boundary shape beforehand, and the other is the fact that the contact stress can only be decided after contact. In this sense, some assumption for linearization should be introduced to analyze the interaction problems.

With recent developments in computer technology, it gradually becomes possible to apply computational method to contact problems which can often be seen in applied mechanics such as terramechanics. This new approach is now classified as computational mechanics. Popular numerical method applied to contact problem is the Finite Element Method (FEM) whose formulation is based on the virtual work principle and whose existance of solution is strictly or mathematically garanteed. But the application of FEM in terramechanics has been limited to the contact analysis between two objects with smooth surfaces which are totally different from the contact situation of traction-type tire and soil. On the other hand, the Discrete (or Distinct) Element Method (DEM) becomes popular for an analysis of assembly of particles such as soils and powders. For tire-soil interactions, the lug rut formation by wheel lug has already and successfully been analyzed for the first time recently[18], which cannot be obtained by FEM. Nowadays, it is possible to apply the high performance PC system as a tool to analyze interaction problems by FEM or by DEM which have been the typical job for supercomputer at University Data Processing Center a decade ago.

The final goal of this study is to develop and to prepare a practical and portable computer simulation tool for soil-tire interaction analysis where DEM and FEM are coupled together. The introduction of parallel processing of analysis is also our final target. The objectives of this study are, firstly, to clarify the possibility of application of dynamic Finite Element-Discrete Element Method (FE-DEM) to the tire-soil interaction problems. Secondly, the application of parallel processing method to dynamic FE-DEM is also investigated. It is noted that the 3D analysis in this study is approximately done by calculating the effect of thickness in 2D analysis.

1.2 Grant-in-Aid Data

1.2.1 Investigator and institution

Hiroshi NAKASHIMA
Associate Professor, Dr.
Agricultural Systems Engineering Laboratory
Division of Environmental Science & Technology
Graduate School of Agriculture
Kyoto University, Kyoto, JAPAN
Phone: +81-75-753-6164
Fax: +81-75-753-6165
E-mail: hiron@elgin.kais.kyoto-u.ac.jp

1.2.2 Annual budgets

Fiscal Year	Amount (Thousand Yen)
2000	1,700
2001	900
2002	900
Total	3,500

1.2.3 Publications on this research

1. H. Nakashima and A. Oida: Algorithm and implementation of soil-tire contact analysis code based on dynamic FE-DE method, Proceedings of the 14th International Conference of ISTVS, Vicksburg, 2002. (CD-ROM)

2. H. Fujii, A. Oida, H. Nakashima, J. Miyasaka, M. Momozu, H. Kanamori and T. Yokoyama: Analysis of interaction between lunar terrain and treaded wheel by Distinct Element Method, Proceedings of the 14th International Conference of ISTVS, Vicksburg, 2002. (CD-ROM)

3. H. Nakashima, A. Oida: Simulation of soil-tire interaction by a coupled Distinct Element-Finite Element Method, Proceedings of the 6th Asia-Pacific ISTVS Conference, Bangkok, 59-63, 2001.

4. M. Momozu, A. Oida, H. Nakashima: Simulation of shear box test by the Distinct Element Method, Proceedings of the 6th Asia-Pacific ISTVS Conference, Bangkok, 181-188, 2001.

5. H. Nakashima Numerical analysis by FEM and DEM, Proceedings of the 1st Organized Terramechanics Workshop in Sendai, Japanese Society for Terramechanics, 22-27, 2001.

1.2.4 Key words

Terramechanics, Computational Mechanics, FEM, DEM, contact problem, tire, mesh preparation, FE-DE method, explicit solution, parallel processing.

Chapter 2

Finite Element Mesh Preparation for Agricultural Tires

2.1 Introduction

FEM nowadays becomes a powerful numerical tool which can be applied not only to structural mechanics problems but also to non-structural fluid dynamics problems. In FEM analysis, pre-processing of mesh discretization cannot be avoided. In terms of the prediction of traction performance by FEM, we have to prepare FE mesh configuration which is sufficient for the accurate analysis. However, the tread pattern for agricultural tractor tires is, in general, traction-type, i.e. the existance of tire-lug cannot be ignored in the precise prediction of tire performance. Therefore, the preparation of tire geometry with not only smooth (or no) tread pattern but also traction-type lug is inevitable in FE mesh generation. In this chapter, the tool for 3D tire mesh generation is investigated and developed.

2.2 Mesh Generation for Smooth Tires

2.2.1 Data preparation

As a simple example of mesh generation, we focus on the smooth, or rib-type treaded, tire. In this case, the data on section cutout of a tire is the only important geometric properties. Then, we rotate this data with respect to tire rotation axis so that we can construct the whole 3D tire mesh. The schematic flow of this procedure is shown in Fig.2-1 which is similar in principle to a formerly developed method used in TRAC/G[13].

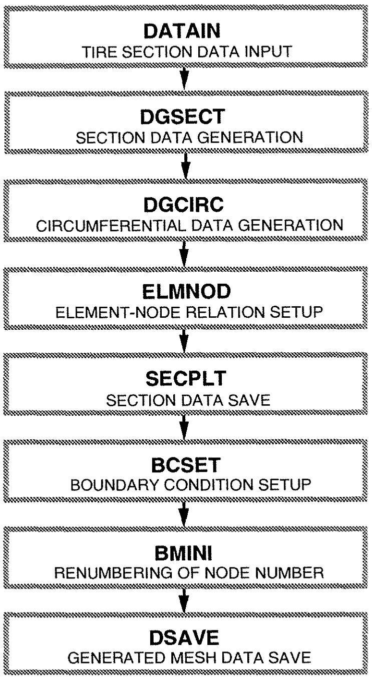

Figure 2-1: Mesh Generation for Smooth Tires

2.2.2 Example of mesh generation

As an example of mesh generation for smooth tires, the rib-patterned tire FSR 400-12 data were collected based on the information supplied by Bridgestone Corporation. It is clear that the cutout section-based smooth tire generation is quite simple and effective.

2.3 Mesh Generation for Traction Tires

2.3.1 Data preparation

The mesh generation for traction tires should consider the existance of traction-lug. We can simply divide the tire components into two parts; (i)tire bottom part and (ii)lug part. Then, the section data of tire bottom is prepared based on the formerly stated procedure for smooth tires, whose result of mesh configuration is called MESH A. Next, we check and prepare the geometry data of one lug shape and this data is used to generate the lugs elements for whole tire as MESH B. Finally, the bottom of lug elements MESH B and the surface of tire bottom elements MESH A are adjusted and connected together. Modified mesh generation

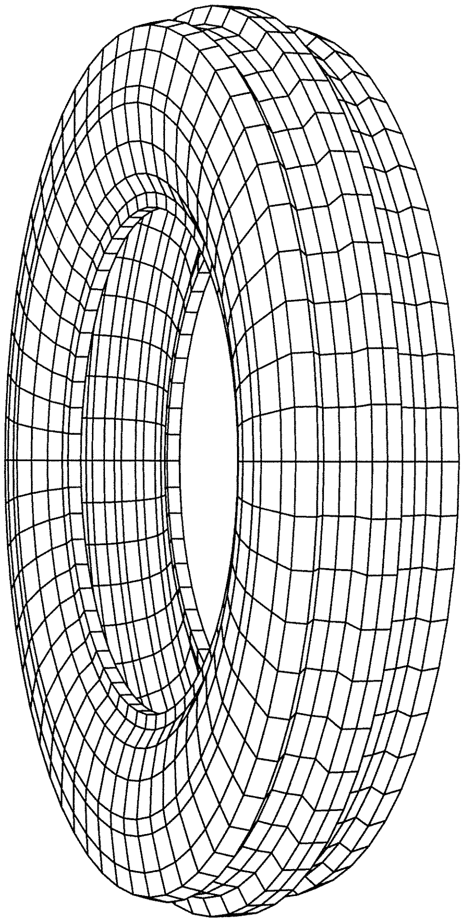

Figure 2-2: FE Mesh for Rib Tire

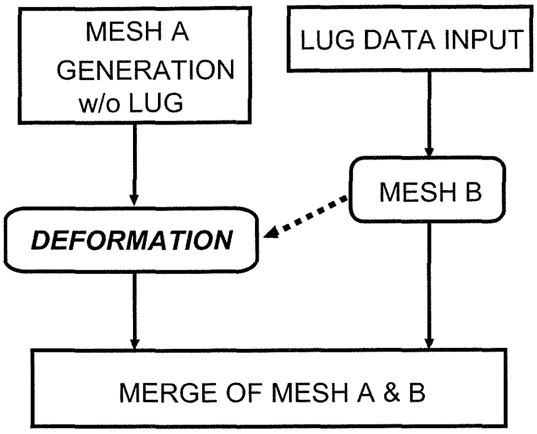

Figure 2-3: Mesh Generation for Traction Tires

procedure is schematically shown in Fig.2-3. Thus, we can develop the tool of FE mesh generation for traction tires.

2.3.2 Example of mesh generation

We demonstrate our proposed mesh generation scheme by using AGS 600-12 U10L tire data for walking-type tiller which is supplied by Bridgestone Corporation.

MESH A is firstly prepared which is based on the sectional data of tire bottom as in Fig. 2-4. Moreover, MESH B for tire lug is generated based on a measured data for one traction lug (Fig. 2-5). Then, the bottom of tire lug shape is used for the deformation of tire bottom surface data as in Fig. 2-6. By connecting the tire bottom elements and lug elements, we can generate a FE mesh for traction tire as shown in Fig. 2-7.

2.4 Concluding Remarks

- We have developed two 3D FE mesh generation programs; one for smoothed tires, and the other for traction tires.
- Generation of FE mesh for traction tire can successfully demonstrated by using a typical tire data measured and supplied by a tire manufacturer.

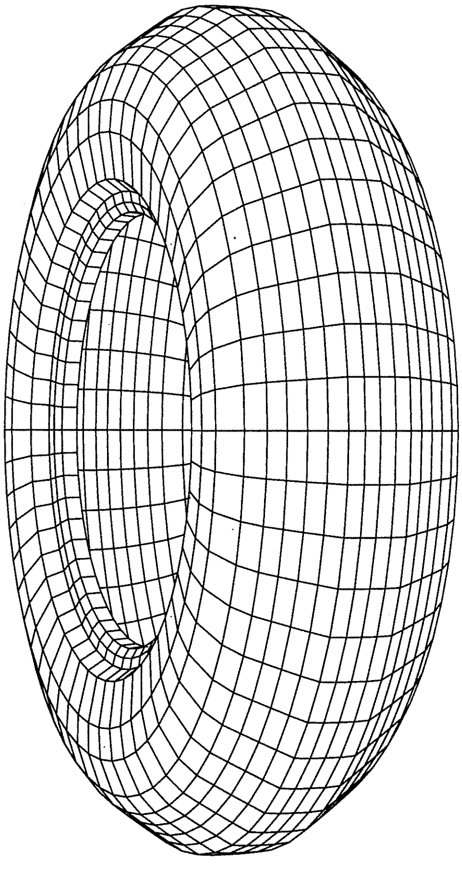

Figure 2-4: Mesh A without Connection Preparation

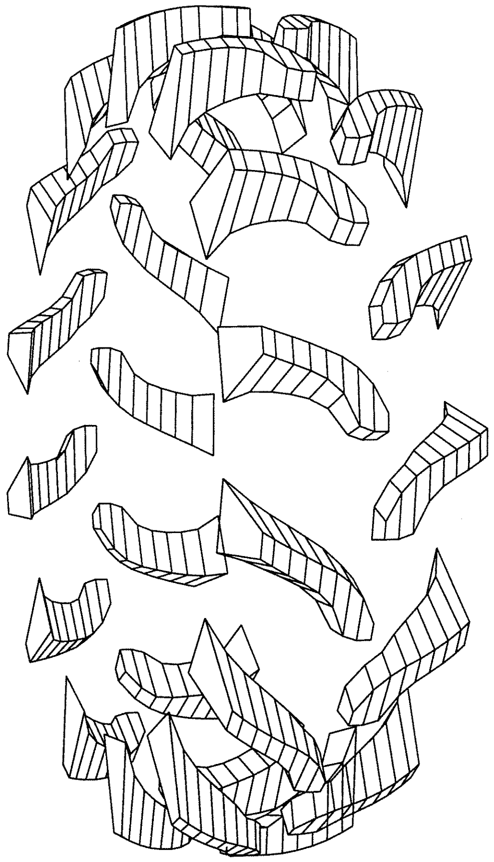

Figure 2-5: Generated Mesh B

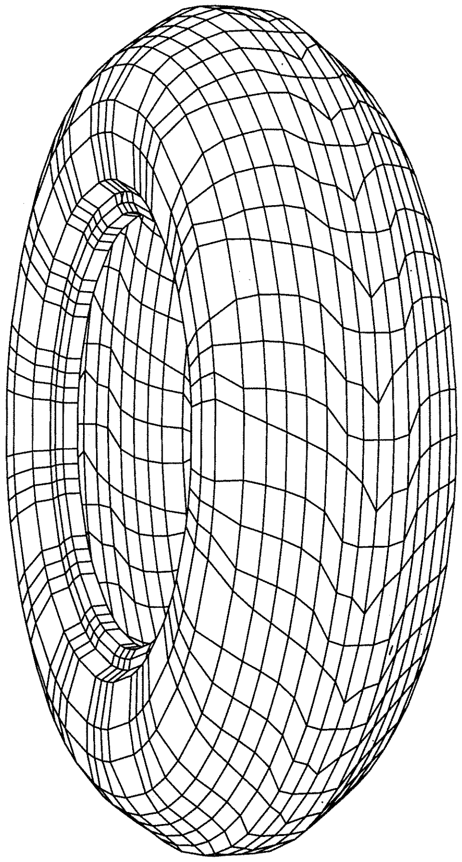

Figure 2-6: Mesh A with Connection Preparation

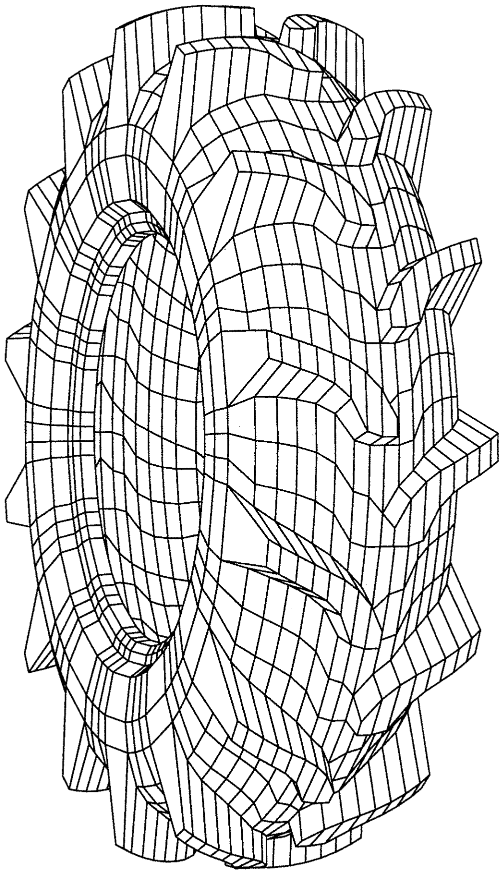

Figure 2-7: FE Mesh for Traction Tire

Chapter 3

Discrete Element Method

3.1 Introduction

Soil-tire system interaction has been one of the fundamental research subjects in terramechanics. Recent developments in information technology have been increasing the possibility of detailed numerical simulation that is applied in interaction problems. The Discrete Element Method (DEM) was originally proposed by Cundall[2] and has been applied not only to soil or rock mechanical problems but also to simple tillage, to plane shear test[12] and to wheel-soil interaction problems[17]. DEM consists of the assembly of discrete granular elements and is simple in the implementation of computer program. Oida *et al.* [18] firstly demonstrated the applicability of DEM to wheel-soil contact problem, where various wheel lugs were considered. It should be noted that the wheel rut that is usually observed in outdoor experiments can similarly be obtained by DEM[19].

3.2 Principle of DEM

In DEM, the local contact mechanics between two elements are a unit of formulation.

3.2.1 Relative displacement between DEM elements

As shown in Fig.3-1, the local axis is defined at the contact surface of two DEM elements i and j. Then, if let the angle θ_{ij} be measured between the global coodinate system of x-y and local coodinate system of n-t w.r.t. the element i, we can have relative normal and tangential displacements, u_n and u_t respectively, expressed on

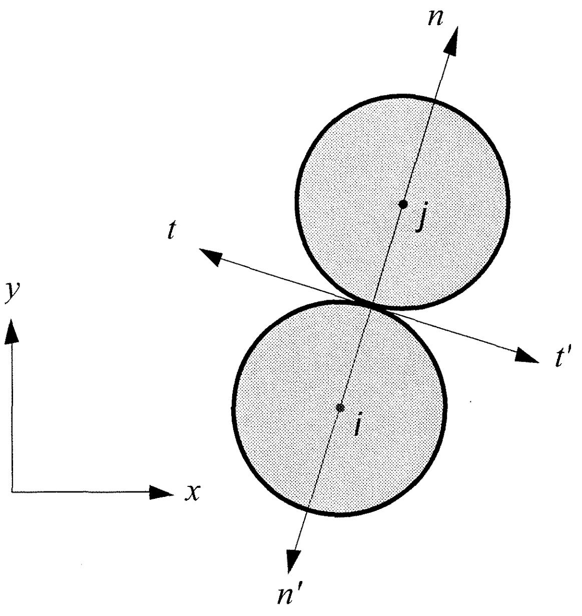

Figure 3-1: DEM-DEM Local Axis

local coodinate system as;

$$\Delta u_n = (\Delta u_i - \Delta u_j)\cos\theta_{ij} + (\Delta v_i - \Delta v_j)\sin\theta_{ij} \tag{3.1}$$

$$\begin{aligned}\Delta u_t &= -(\Delta u_i - \Delta u_j)\sin\theta_{ij} + (\Delta v_i - \Delta v_j)\cos\theta_{ij} \\ &+ (r_i\Delta\phi_i + r_j\Delta\phi_j)\end{aligned} \tag{3.2}$$

where $\Delta(*)$: incremental expression of variable in a given time step of Δt; r_k: radius of DEM element k; $\Delta\phi_k$: incremental rotational angle of DEM element k.

3.2.2 Contact mechanics

Normal force and tangential force are in general transmitted by Voigt model as in Fig. 3-2. Based on this model, we can calculate the normal contact reaction Δf_n and the normal viscous reaction Δd_n if relative displacement Δu_n and relative velocity $\Delta\dot{u}_n$ are known as follows;

$$\Delta f_n = K_n\,\Delta u_n \tag{3.3}$$

$$\Delta d_n = C_n\,\Delta\dot{u}_n \tag{3.4}$$

where K_n is the normal spring constant and C_n implies normal damping coefficient, when the distance L_{ij} of c.g. of two DEM element i-j becomes less than $r_i + r_j$, i.e. $L_{ij} < r_i + r_j$. We can obtain the current normal reaction at time t as in

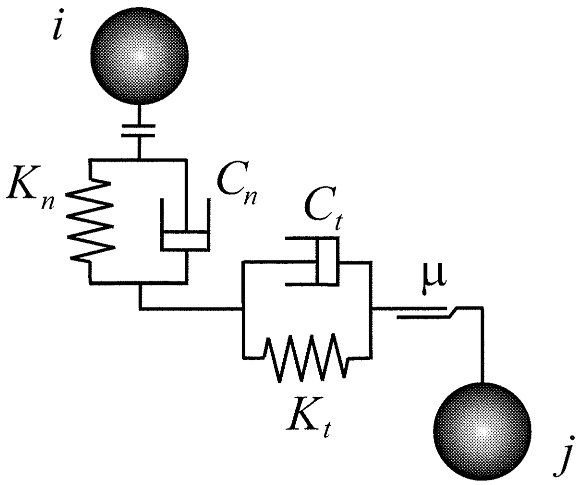

Figure 3-2: DEM model

Eqns (3.5)(3.6) by assuming no significant change of contact point between time t and time $t-1$;

$$ {}^{t}f_n = {}^{t-1}f_n + \Delta f_n \tag{3.5} $$
$$ {}^{t}d_n = \Delta d_n \tag{3.6} $$

Therefore, the normal contact reaction between DEM elements i-j at time t can be calculated as;

$$ {}^{t}F_n = {}^{t}f_n + {}^{t}d_n \tag{3.7} $$

where ${}^{t}(*)$ denotes any property at time t. In Eqn (3.7), it is noted that the additional damping reaction is included in the second term on RHS and this effect is called as local damping[2].

As for tangential contact reaction, it can be expressed as follows;

$$ \Delta f_t = K_t\, \Delta u_t \tag{3.8} $$
$$ \Delta d_t = C_t\, \Delta \dot{u}_t \tag{3.9} $$

where K_t is tangential spring constant and C_t is tangential damping coefficient.

We can obtain the current tangential reaction at time t as in Eqn (5);

$$ {}^{t}f_t = {}^{t-1}f_t + \Delta f_t \tag{3.10} $$
$$ {}^{t}d_t = \Delta d_t \tag{3.11} $$

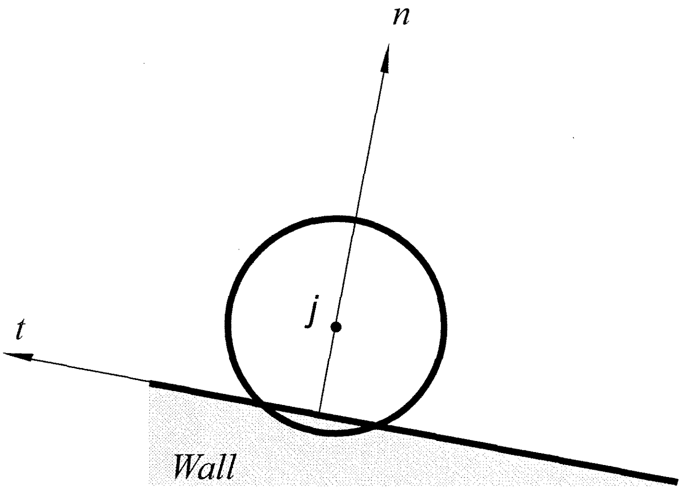

Figure 3-3: DEM-Wall Contact

Coulomb friction criterion may be applied for the lower bound and the upper bound of friction component as follows;

$$
\begin{aligned}
{}^t f_t &= {}^t d_t = 0 \quad ({}^t f_n \leq 0) && (3.12)\\
{}^t f_t &= \mu \, {}^t f_n \, ({}^t f_t / |{}^t f_t|); \; {}^t d_t = 0 \quad ({}^t f_t \geq \mu \, {}^t f_n) && (3.13)
\end{aligned}
$$

Therefore, the tangential contact reaction between DEM elements i-j at time t can be calculated as;

$$
{}^t F_t = {}^t f_t + {}^t d_t \qquad (3.14)
$$

Typical DEM element-wall contact is shown in Fig. 3-3. The procedure of wall contact is quite similar to the DEM-DEM contact procedure. Let a DEM element i be contacting to the selected wall segment which is regarded as target element j and we may substitute $\Delta u_j = \Delta v_j = r_j = 0$ into Eqns (3.1) and (3.2) to obtain local relative displacements Δu_n and Δu_t. Further calculation steps are similar to DEM-DEM case.

3.2.3 Assembly of contact equations

Obtained contact reaction equations are then summed into the total contact reaction on element i. Thus, we obtain the elemental equation of motion as follows;

$$
\begin{aligned}
{}^t X_i &= M_i \, {}^t a_i^x && (3.15)\\
{}^t Y_i &= M_i \, {}^t a_i^y && (3.16)\\
{}^t N_i &= I_i \, {}^t \dot{\phi}_i && (3.17)
\end{aligned}
$$

where X_i is x-component of the sum of contact reaction on element i; Y_i is y-component of the sum of contact reaction on element i; N_i is the sum of moment on element i; M_i is the mass of element i; I_i is the moment of inertia for element i; a_i^x is x component of acceralation on element i; and $\dot{\phi}$ is the angular acceleration of element i respectively.

From Eqns (3.15)(3.16)(3.17), we can calculate acceleration for element i. We then apply numerical integration to obtain velocity and displacement increment between time step t and $t+1$. Since the explicit time integration is applied, the solution is conditionally stable. The time step size Δt may be decided based on the critical damping of vibration and be controlled as $\Delta t \leq 2\sqrt{m/K_n}$[22]. In general, trial-and-error check of time step size should be done beforehand.

3.2.4 Parameter setup

One of the difficult tasks in DEM analysis is to fix virtual material constants such as K_n, K_t and C_n, C_t beforehand. The unique-valued spring constants can be obtained from try-and-error preliminary calculation. It is also possible to use varied-value spring constants based not only on Hertz contact theory for normal component but also on Mindlin theory for tangential component[22]. Center approach δ of 2D circular disk elements i and j can be calculated as [8];

$$\delta = \frac{2P'L}{\pi}\left(\frac{1-\nu_i^2}{E_i}\left\{\ln\frac{4R_i}{a}-\frac{1}{2}\right\}+\frac{1-\nu_j^2}{E_j}\left\{\ln\frac{4R_j}{a}-\frac{1}{2}\right\}\right) \tag{3.18}$$

$$a = \sqrt{\frac{4}{\pi}\left(\frac{1-\nu_i^2}{E_i}+\frac{1-\nu_j^2}{E_j}\right)\left(\frac{R_iR_j}{R_i+R_j}\right)P'L} \tag{3.19}$$

where P': normal load per unit length; L: contact length; E_k: Young's Modulus for two contacting elements k (k=i,j); ν_k: Poisson's Ratio for two contacting elements k (k=i,j); R_k: Radius of element k (k=i,j); and a: semi-contact width. Therefore, from Eqn (3.18), the normal component K_n becomes;

$$K_n = \frac{P'L}{\delta} = \pi/\left[2\left(\frac{1-\nu_i^2}{E_i}\left\{\ln\frac{4R_i}{a}-\frac{1}{2}\right\}+\frac{1-\nu_j^2}{E_j}\left\{\ln\frac{4R_j}{a}-\frac{1}{2}\right\}\right)\right] \tag{3.20}$$

3.3 DEM Application

As an example of DEM application, the simulated result of rigid wheel with lug running over lunar soil (regorith) is shown[4]. In this analysis, circular DEM element, whose radius is randomly distributed among 1.4, 1.6, 1.8 and 2.0 mm, is

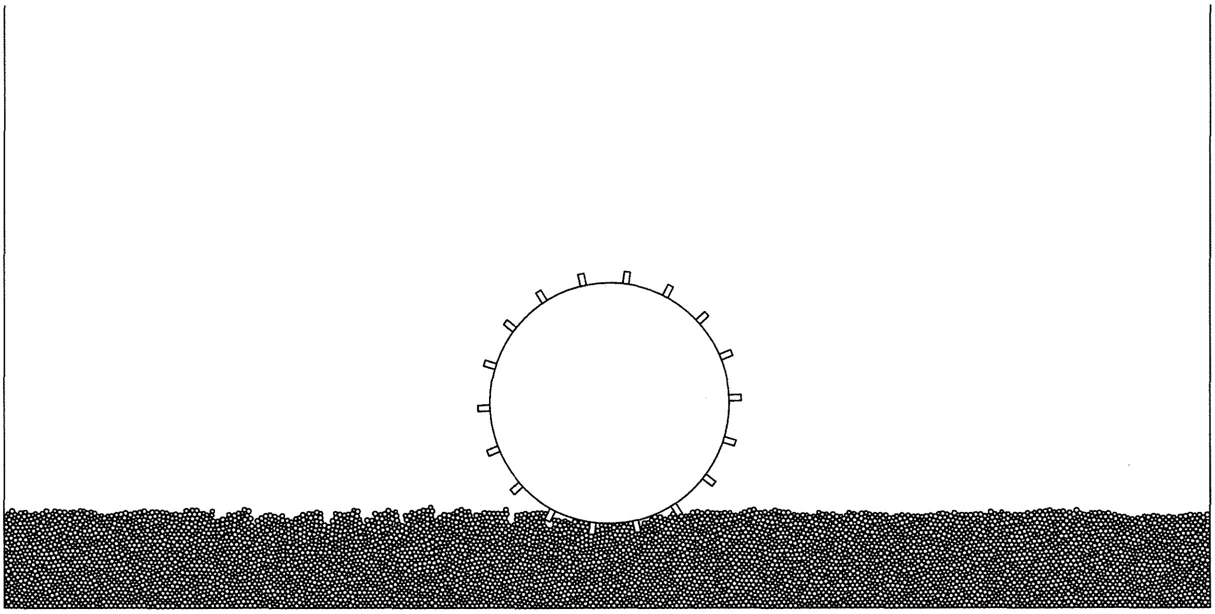

Figure 3-4: Example of DEM application

used. The total number of DEM elements for lunar soil is 6986. It is noted that the clear existance of lug rut is again regenerated on soil surface after the passage of lugged wheel. As for the calculation time for this DEM analysis, it took about 12 hours by using a PC with Alpha CPU(21264/600MHz).

3.4 Concluding Remarks

- DEM could simulate the lug rut formation after lugged wheel travel, which could not be realized by usual FEM analysis.
- The computational time for DEM tends to increase in propotion to the total number of DEM elements.

Chapter 4

Finite Element-Discrete Element Method

4.1 Introduction

In terms of numerical analysis application, soil-tire system interaction has traditionally been analyzed by using Finite Element Method (FEM) with simplified and approximate geometric boundary condition [24, 25]. The soil models introduced in previous studies were not only elastic, but also elasto-plastic models. Ueno *et al.*[23] applied an elasto-plastic soil model, called Subloading Surface Model, and contact algorithm for the analysis of 2D soil-wheel system. Hiroma *et al.*[6] also analyzed the rigid wheel-viscoelastic soil interaction by FEM with contact algorithm where the wheel surface was assumed to be smooth. Recent developments could be seen in an application of critical state soil mechanics to tire-soil problems[10].

With the further development and refinement in FEM, in-depth analysis of contact problems was formulated as well, which was summarized by Zhong[26]. Since the interaction problems in terramechanics belong to the typical contact mechanics in nature, the application of the achievements from applied mechanics becomes beneficial in the computational terramechanics for soil-tire systems. Aubel[1] has successfully analyzed the 2D interaction between soft soil and elastic rolling smooth tire. Furthermore, Fervers[3] extended and demonstrated the treaded tire-soil interaction analysis by FEM. Based on an achievement in 3D-FEM tire model[13], we also developed a 3D-FEM tire contact analysis program for static sinkage on elastic soil [14, 15].

Since there is a demerit of large computation time consumed for both contact check and integration time stepping in DEM and for solving simultaneous equations in static FEM, it is quite natural to utilize not only FEM but also DEM to compensate for the demerit of each method. If coupled together, FE-DE method will become useful in two aspects; that is, one is the reduction of calculation time and the other is the easiness in treatment of tire lug. Pan and Reed [20] applied a coupled FE-DE method to the rock mechanics problems. Flow problems in silo were also analyzed by FE-DE method[11]. Horner *et al.*[7] demonstrated the capability of precise and detailed application of massive scale DEM for soil elements and FEM for bulldozing blade. Moreover, an outline of the attempt in our laboratory for FE-DE method in soil-tire system can also be found[16].

In this chapter, an algorithm for a coupled FE-DE method is proposed in terms of easy implementation. Furthermore, a simple example of tire sinkage problem is analyzed in order to check the validity of the algorithm.

4.2 FE-DE Contact Analysis

Calculation of contact reaction force is firstly summarized from the literature survey. Then, an algorithm for FE-DE method is investigated.

4.2.1 Finite element contact analysis

There are various methods which have been proposed for FEM contact analysis[26]. Among others, the simplest one is penalty method and it has been widely applied to various problems[9]. In the penalty method for 2D contact analysis, the contact of a node 3 of contactor on the target line element 1-2 for an arbitrary time increment can be expressed by allowing slight overlap of Δu_n as in Fig. 4-1. Normal component $\Delta^n R_3$ of contact reaction ΔR_3 can be defined by Eqn (4.1), where $\alpha(\gg 1)$ means a penalty number whose physical meaning is spring constant.

$$\Delta R_3^n = \alpha \Delta u_n \tag{4.1}$$

By using ΔR_3^n, we can obtain $\Delta R_{3'}^n = -\Delta R_3^n$ from action-reaction relationship with the negative sign. Thus, normal reaction component ΔF_1^n and ΔF_2^n for target segment nodes 1 and 2 respectively can be expressed with the following relationship;

$$\Delta F_j^n = N_j \Delta R_{3'}^n = -N_j \Delta R_3^n \quad (j = 1, 2) \tag{4.2}$$

where N_j means the shape function of a line element 1-2.

Similar calculation applies for tangential component ΔF_j^s of contact reaction, namely;

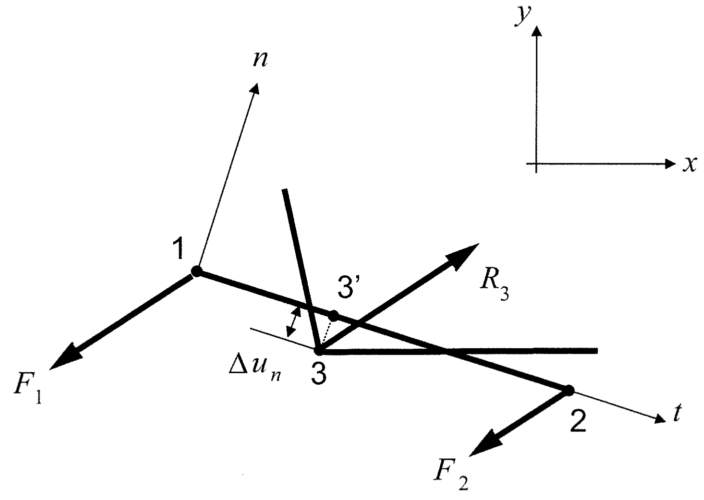

Figure 4-1: FEM Contact

$$\Delta F_j^s = N_j \Delta R_{3'}^s = -N_j \Delta R_3^s \quad (j = 1, 2) \tag{4.3}$$

where $(*)^s$ means tangential component of corresponding reaction.

If the dynamic contact problem is analyzed, Classical Coulomb friction may be introduced in the tangential reaction ${}^tR_3^s$ at a time t, after updating the tangential reaction ${}^tR_3^s = {}^{t-1}R_3^s + \Delta R_3^s$ and comparing it with the corresponding condition of $\mu\, {}^tR_3^n$ where ${}^tR_3^s = \mu {}^tR_3^n$ if ${}^tR_s^3 > \mu {}^tR_3^n$ and ${}^tR_3^n = {}^{t-1}R_3^n + \Delta R_3^n$.

4.2.2 Discrete element contact analysis

As stated in Chapter 3, the contact reaction calculation in DEM is quite simple. In general, Voigt model is assumed for locally contacting elements. The calculation of contact reaction is the important part of DEM analysis.

4.2.3 Finite element-discrete element contact

Let a particular element 3 of DEM be in contact on a line segment 1-2 of 2D FEM element as in Fig. 4-2.

Then, it is noted that the process of calculation for contact reaction is the same as in the case for wall contact in DEM analysis. However, the line segment of finite element may be translated according to contact reaction in the case of FEM and FEM-DEM analysis. The calculation of relative displacement is done, as in FEM contact case, between the c.g. point 3 of DEM and the projected point 3' of 3 on the line segment 1-2. If there exists an overlap Δu_n, then contact reaction

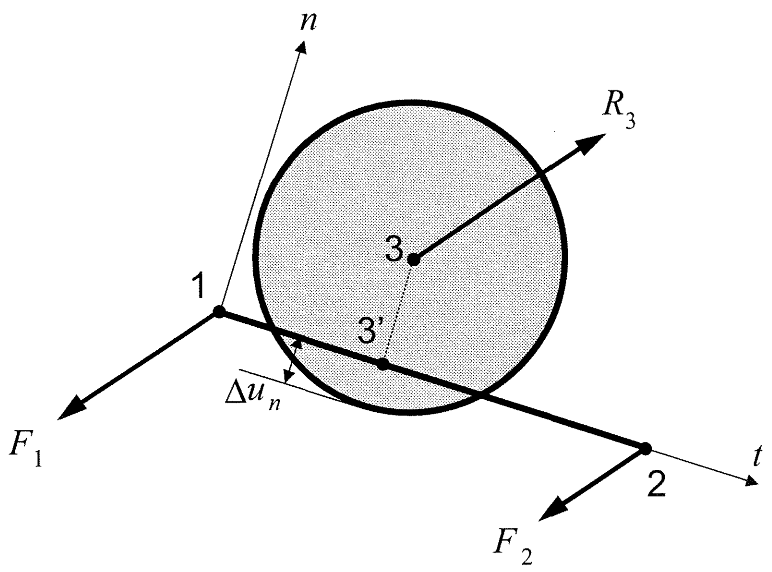

Figure 4-2: FEM-DEM Contact

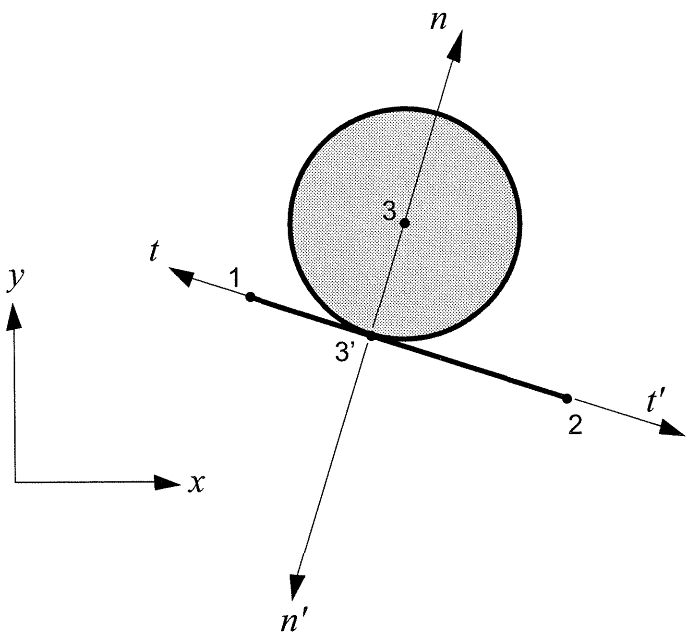

Figure 4-3: FEM-DEM Local Axis

acts on the point 3 as ΔR_3. In calculation of ΔR_3^n, either a penalty method as in Eqn (4.1) or a normal spring model in DEM-DEM contact as in Eqn (3.3) may be applied. The counterpart reaction on nodes 1 and 2 can be calculated by way of shape function as in Eqn (4.2). The tangential reaction may follow the Coulomb friction law if tangential frictional effect is included.

4.3 Dynamic explicit analysis

In 2D DEM or dynamic FEM, we must solve the following equations of motion;

$$F_k = F_c^k + F_v^k + F_b^k = M\frac{d^2u^k}{dt^2} \quad \text{(for FEM and DEM)}\ (k = x, y) \tag{4.4}$$

$$N = N_c + N_b = I\frac{d\phi}{dt} \quad \text{(for DEM only)} \tag{4.5}$$

where F_c : contact reaction, F_v : body force, F_b : boundary load, M : mass, u : displacement, N_c : moment by f_c , N_b : outer moment, I : moment of inertia, and ϕ : angular velocity.

Contrary to the static analysis where we have to solve linear systems of equation, dynamic analysis is quite simple and, among others, no need to solve linear simultaneous equations if we adopt an explicit solution method with lumped mass matrix for FEM. It should be noted that the obtained displacement result is more accurate in an implicit solution method than in an explicit method.

As for an explicit method, the central diference method is widely applied where the solution is conditionally stable, *i.e.* bounding value is used for the time increment Δt. In case of FEM, the maximum time step increment is defined by, so-called, Courant condition as $\Delta t_{FEM} \leq L_e/c$, where L_e : effective length of an element, and c : velocity of elastic wave. As stated in Chapter 3, the time step of DEM may be decided by $\Delta t_{DEM} \leq 2\sqrt{m/K_n}$. It is noted that we must choose an optimum time step by trial-and-error method beforehand so that the required calculation can be done successfully. In present FE-DE analysis, it is clear that the time step depends on DEM parameters in terms of the used element radius although the total region of DEM model can be reduced in FE-DE method.

4.4 Contact Algorithm and Coding

4.4.1 Algorithm of FE-DE contact analysis

If an explicit method is used, we can simplify an algorithm for an given time increment Δt, which can be summarized as below;

1. Select a candidate FE line segment 1-2.
2. Check which DEM element 3 belongs to the selected line segment 1-2.
3. After DEM 3 is specified, check whether or not an overlap exists between DEM 3 and a FE line segment 1-2.
4. If exists, calculate normal and tangential component of relative displacements Δu_n and Δu_t between 3 and 3'. Calculate also relative displacement velocity $\Delta \dot{u}_n$ and $\Delta \dot{u}_t$.
5. Calculate normal reaction Δf_n.
6. Calculate current total normal reaction ${}^t f_n = {}^{t-1} f_n + \Delta f_n$ as in Eqn (3.5) for a contacting pair of DEM-FEM by using spring constants based on Hertz contact theory.
7. Calculate the current total tangential reaction ${}^t f_t = {}^{t-1} f_t + \Delta f_t$ by using Eqn (3.10) and check the Coulomb friction condition. Use Eqn (3.12) or (3.13) based on the magnitude of ${}^t f_t$ and $\mu {}^t f_n$.
8. If other line segments should be calculated, go to 1. Otherwise, go to DEM calculation.

If the surface of a FEM model which will contact with DEM is not smooth as in the case of an agricultural tractor tire lug or a truck shoe, some additional strategy of contact check is required[19].

4.4.2 Program flow

Schematic program flow is shown in Fig. 4-4. Program was coded by Fortran77, compiled by Intel Fortran Compiler, and executed on a PC (Linux System), whose CPU is Pentium4 1.6A GHz with 1GB RIMM memory. Program for DEM calculation is partly used from the reference book[22]. Main part of program is listed in Appendix A1. Subroutines are based on Fortran FEM program[21] as shown in Appendix A2.

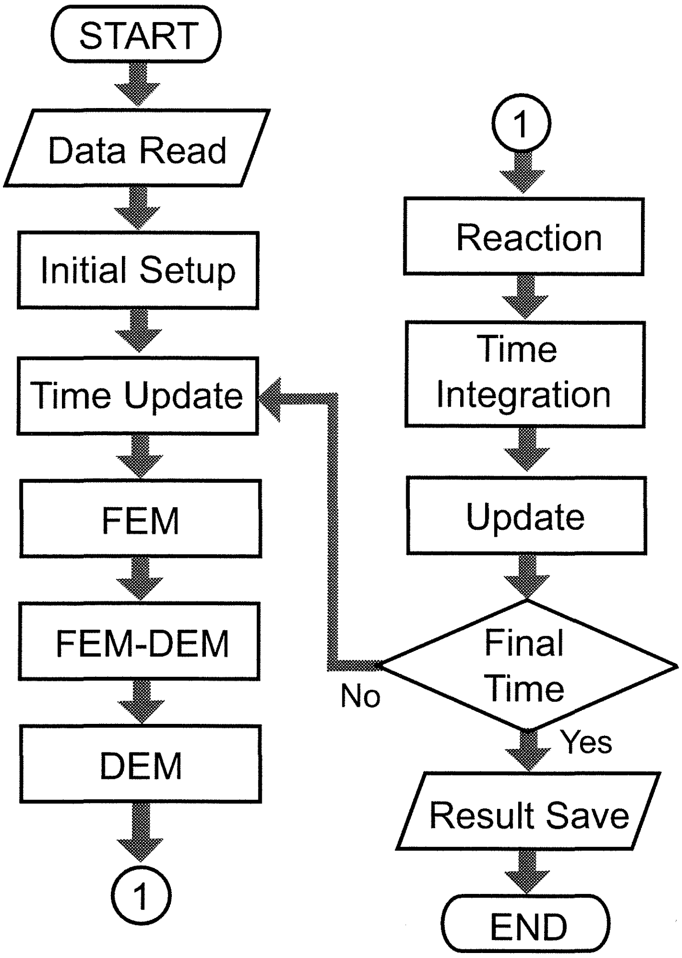

Figure 4-4: Schematic program flow of FE-DE analysis

Brief flow of analysis can be summarized as; acceleration, velocity and current displacement of an node (FEM) or an element (DEM) is obtained from known forces as in Eqns (3.15) and (3.16) by applying simple time integration to Newton's 2nd law, and obtained displacement is used for the calculation of forces in next time-step acting in a target system.

Note that the contact of FE-DE check loop is included after FEM calculation and before DEM calculation in Fig. 4-4, which means the contact reaction is treated as an acting boundary forces for DEM calculation.

Table 4-1: Used parameters for FE-DE calculation

Model	Element	E (MPa)	ν	ρ_e (kg/m^3)	μ_{i-s}
FEM	Rubber	20	0.46	50000	0.3
	Rim	206000	0.3	50000	NA
	Lower Soil	100	0.3	20000	0.3
DEM	Top Soil	32	0.33	20000	0.3
	Wall	30	0.3	NA	0.3

4.5 Numerical Experiment

4.5.1 Problem statement

Simple vertical tire sinkage problem is solved in order to verify our proposed algorithm of FE-DE analysis. The diameter of tire is assumed to be 110 cm and the FE mesh are prepared for the tire and rim and the bottom layer of soil. The upper layer of soil, where a tire contacts, is modeled by DEM. Used parameters are summarized in Table 4-1, where E: Young's Modulus, ν: Poisson's Ratio, ρ_e: Elemental density, μ_{i-s}: Coefficient of Friction for interfaces between material i and soil. Elemental density data used in the analysis are modified so that large time step as well as the stability of calculation can be realized.

We calculated FE-DE analysis in the following cases; (i)Case A, where radius of DEM element $r_{DEM} = 2.0$ cm and total number of DEM $n_{DEM} = 248$; (ii)Case B ($r_{DEM} = 1.0$ cm; $n_{DEM} = 896$); and (iii)Case C ($r_{DEM} = 1.0$ cm; $n_{DEM} = 2243$).

4.5.2 Case A result

Fig.4-5 shows the initial configuration for Case A. Fig.4-6 depicts the result of analysis after the simulated time of 0.08191 (sec), where the time step increment of 1.0×10^{-5} sec is used. The tire sinkage was found to be 3.29 cm and the total calculation time was 57 sec. The computation is stopped when the final vertical contact reaction exceeds the total weight of tire and rim. The final vertical contact reaction in Case A is found to be 4650.7 N, whereas the reaction for Case B is 4652.3 N.

The loading condition of tire is set by self-weight of tire part (tire and wheel rim), and the corresponding average falling velocity of tire part becomes about 40.2 cm/s. As shown in Fig. 4-6, the soil region just below tire contact indicates

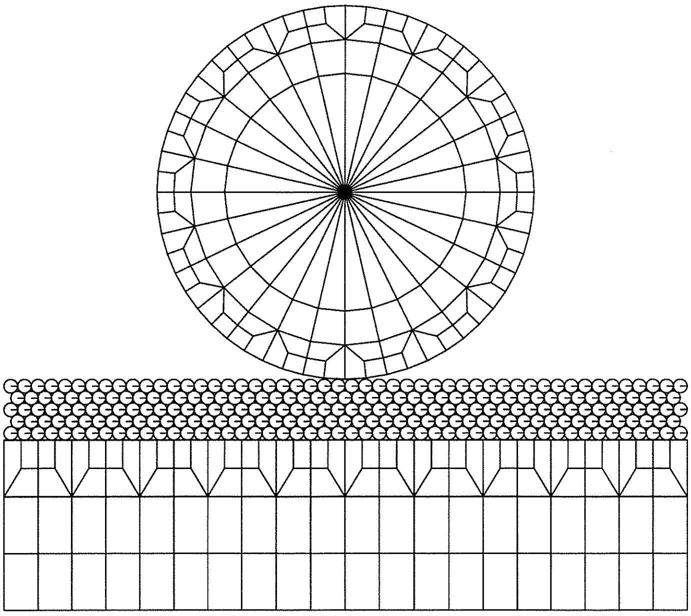

Figure 4-5: Initial Mesh Configuration of Case A

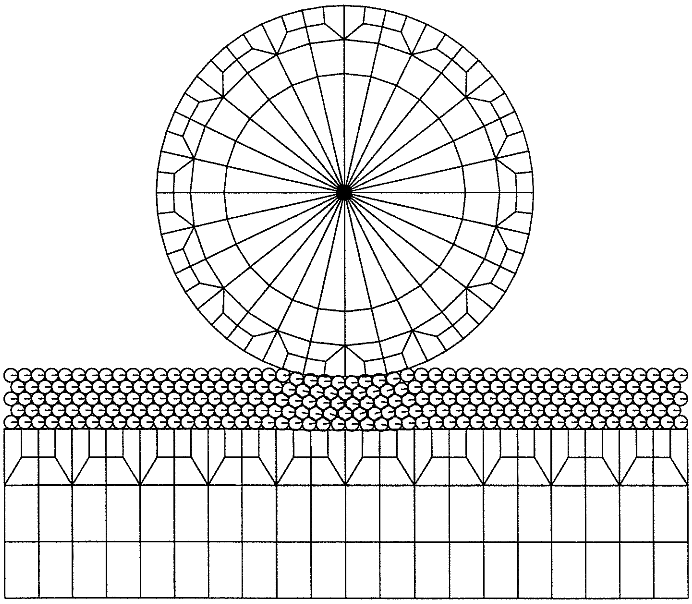

Figure 4-6: Deformation after 8191 time steps

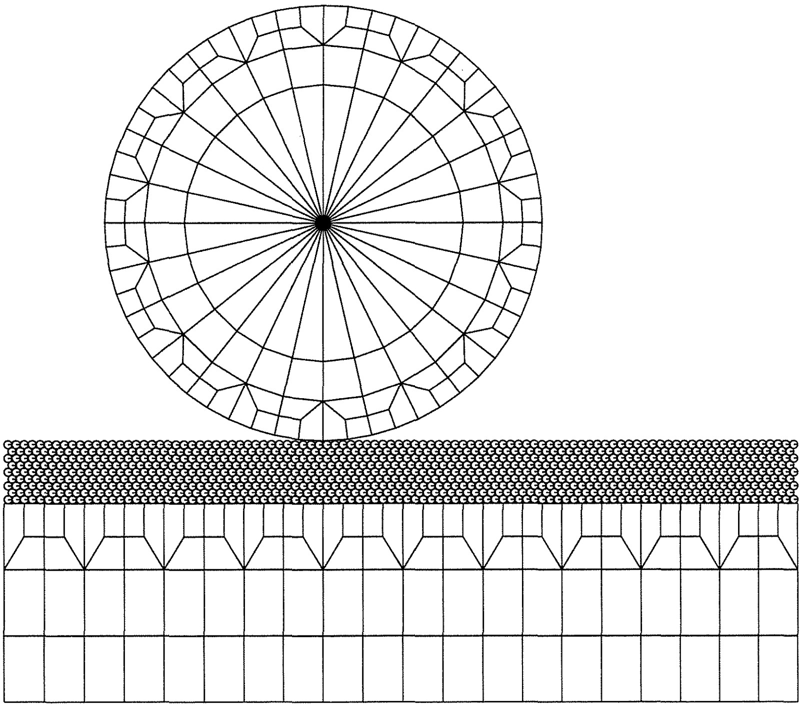

Figure 4-7: Initial Mesh Configuration of Case B

the growth of gap between DEM elements which implies the generation of internal shear lines within the soil.

4.5.3 Case B result

Fig.4-7 shows the initial configuration for Case B. The result of analysis after the simulated time of 0.08482 sec, where the calculated vertical load became larger than the total tire weight of 4650 N, is shown in Fig.4-8. The tire sinkage reached 3.53 cm. Total calculation time for Case B was 109 sec. Average falling velocity of tire in this case becomes 41.6 cm/s.

4.5.4 Case C result

The purpose of this case is to observe the effect of DEM region height on the solution of FE-DE analysis. Note that the bottom layer of soil part becomes longer, and this implies the total number of FEM element for soil bottom also increases. In this case, the total time steps becomes 9828 (0.09828 sec) when the calculated vertical load exceeds 4650 N. Sinkage of tire was found to be 4.74 cm, thus the average falling velocity of tire was 48.2 cm/s. The total calculation time was 190

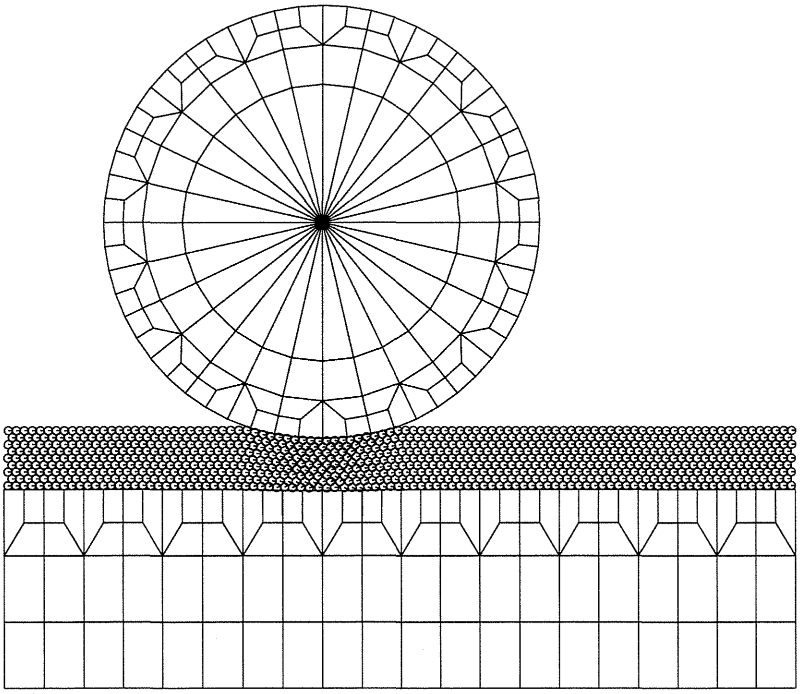

Figure 4-8: Deformation after 8482 time steps

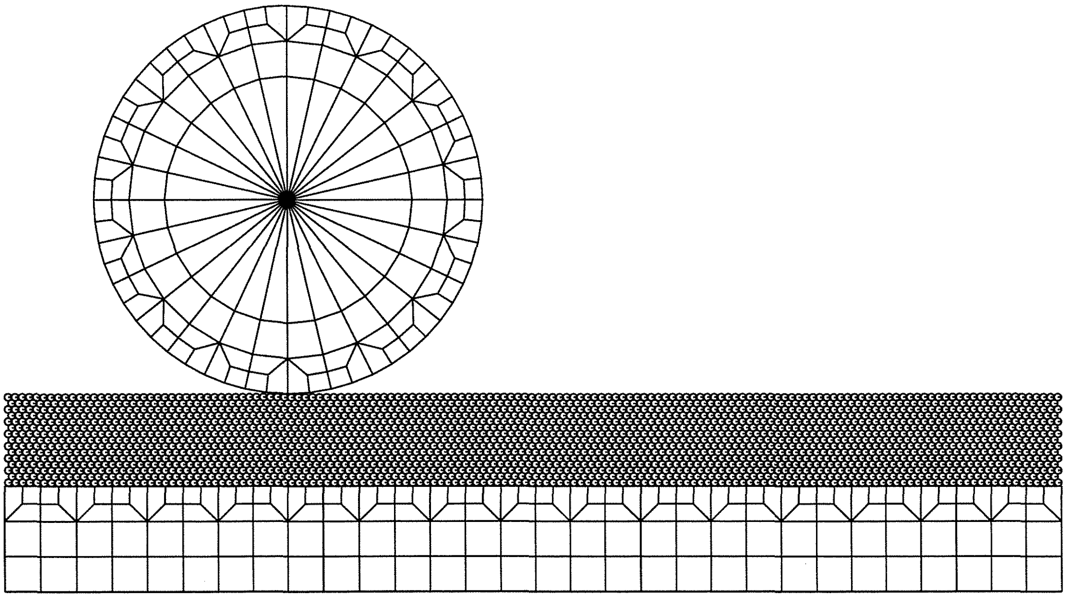

Figure 4-9: Initial Mesh Configuration of Case C

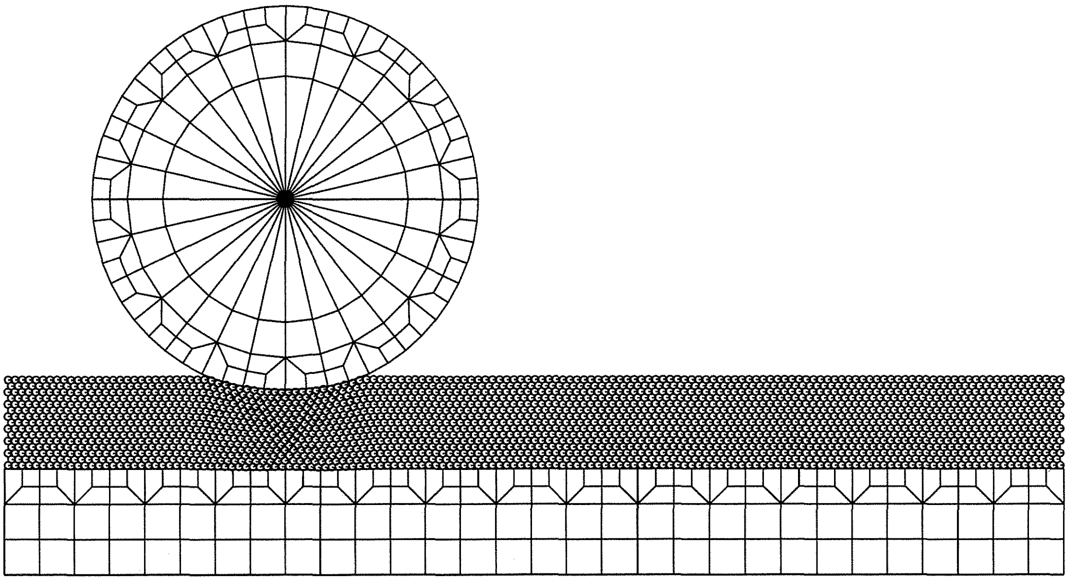

Figure 4-10: Deformation after 9828 time steps

sec.

4.6 Result and Discussion

From Figures 4-6, 4-8, 4-10, we may judge that our developed program can satisfactorily solve this simple problem in terms of deformation field and we understand that the simple algorithm of FE-DE contact works well.

The relationship of vertical contact reaction and tire sinkage is shown in Fig. 4-11, where the data with marker ■ is for Case A while the marker ● is used for Case B, and the line stands for Case C. From this Figure, it is clear that the Case A and Case B results in the similar sinkage and reaction in spite of the difference in element radius for DEM. By comparing the result of Case B and Case C, we can know the slight softening behavior of DEM at larger tire sinkage. It is interesting to note that the calculated vertical loads in all cases do not exhibit the significant difference. As for futher investigation, we need to check the relationship of vibratory results in Case A and Case B. The main cause of this behavior may be estimated as the combined effect of insufficient damping effect in the calculation and the cause of mechanics in DEM element size.

Fig. 4-12 shows the relationship between the total calculation time and total number of DEM elements. From this Figure, it is noted that the increase in calculation time can be approximated linear equation of the form: $Y = 0.065677X +$

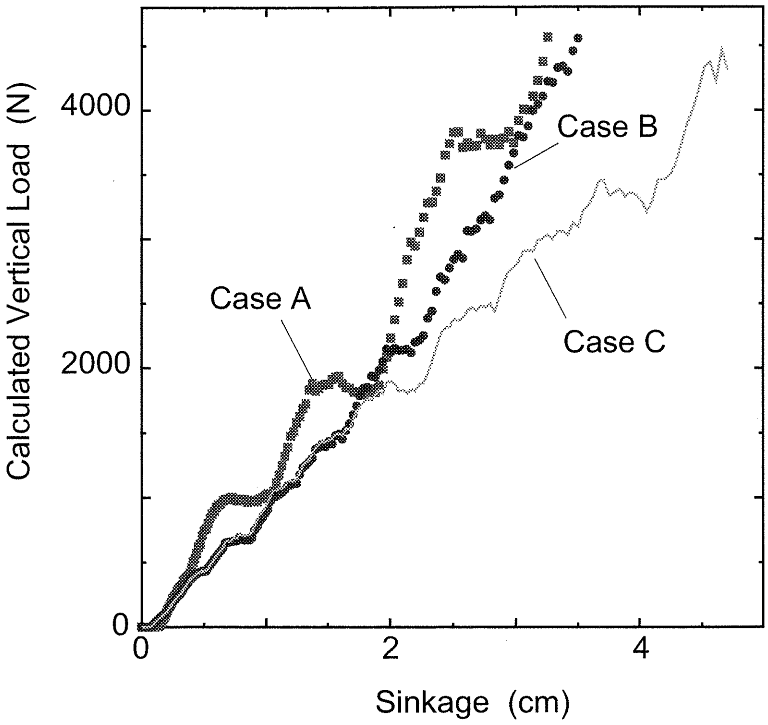

Figure 4-11: Load-sinkage result

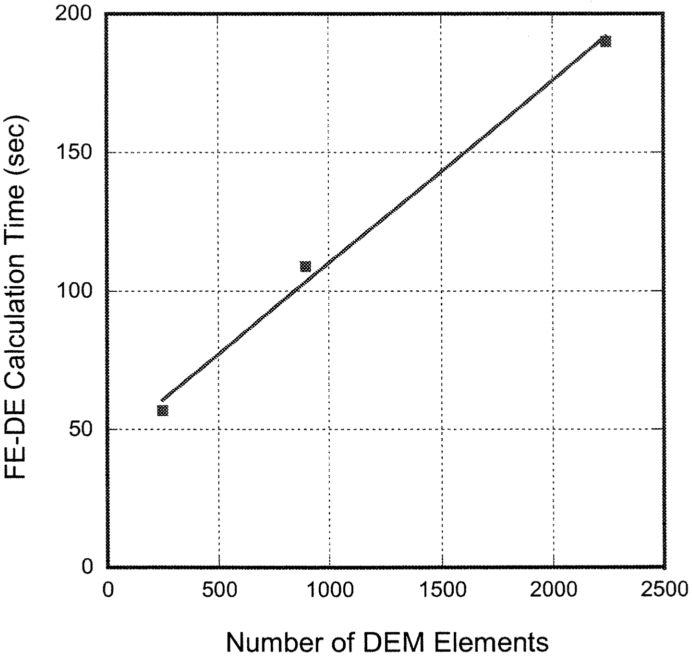

Figure 4-12: Calculation time in terms of DEM elements

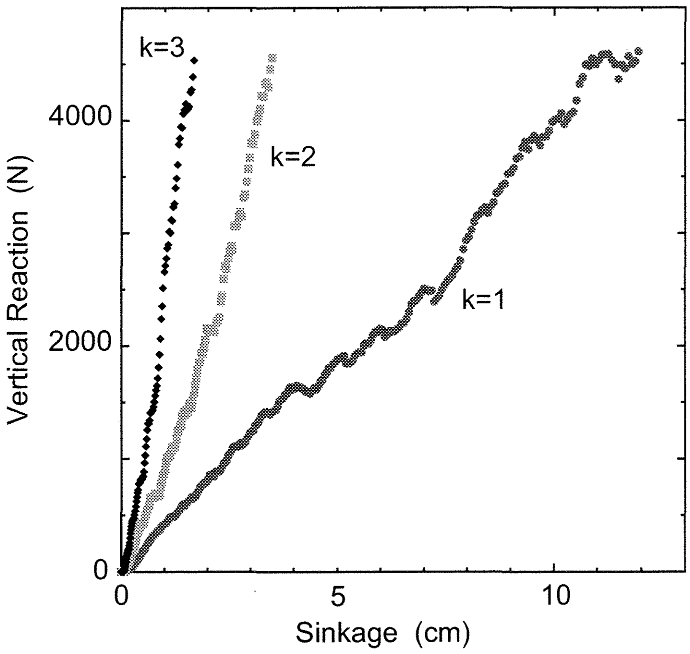

Figure 4-13: Effect of DEM parameter on calculated load

$44.518(R^2 = 0.99724)$ though there exists the increase in FEM element in Case C. Therefore, it is estimated that the larger the total number of DEM elements used, the more total calculation time consumed.

At the contact region of FE-DE boundary, it sometimes needs an adjustment of the value for spring constants so that the severe overlap at this boundary should not exist beforehand, which often occurs for penalty method. Unified method of how to fix this kind of parameter for FE-DE method should be done.

The numerical simulation for treaded tire-soil contact and the use of randomly arranged DEM element radius are our next step of development of program.

Fig. 4-13 shows the result of effect of DEM parameter on calculated vertical reaction. Selected parameter is Young's Modulus E for DEM, whose values are (i)3.2 MPa(k=1); (ii)32 MPa(k=2); (iii)320 MPa(k=3), because of the fact that the current contact reaction depends on the value of E as shown in Eqn(3.20). From the Figure, it is clear that the calculated load becomes large when E value increases. Thus, it is noted that the proper adjustment of E value should be done when DEM calculation becomes precise.

The deformation result for E=3.2 MPa case is shown in Fig.4-14. The sinkage when the calculated vertical load exceeds 4650 N was found to be 11.93 cm with the vertical reaction of 4609 N. With the smaller value of Youngs' Modulus, the overlaps at contact surface of DEM elements became obvious as in Fig.4-14.

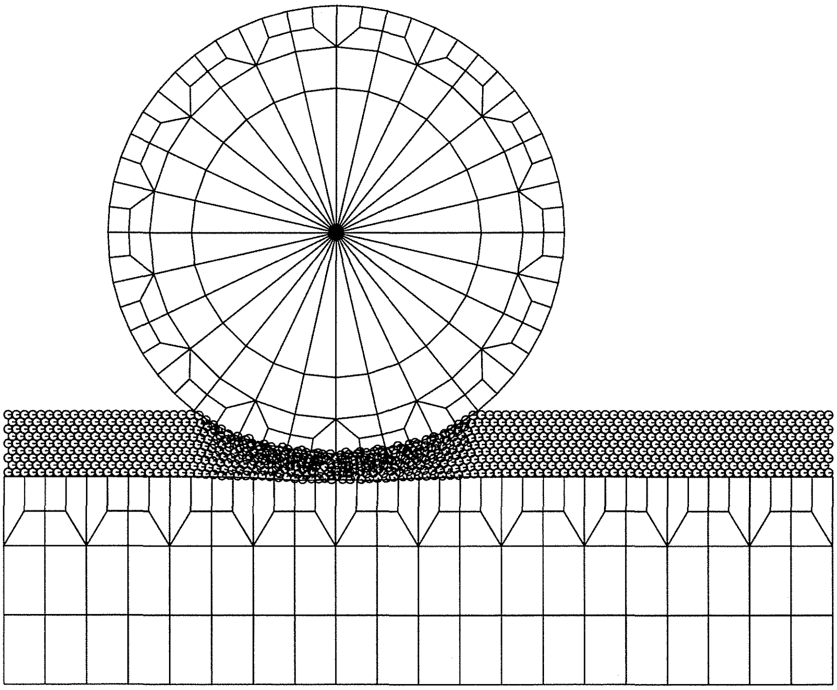

Figure 4-14: Deformation result when E=3.2 MPa

4.7 Extension to 2D Tire Rolling Problem

4.7.1 Strategy for tire rotation

As stated in former section, currently developed program deals with the vertical free sinkage of tire by self-weight and applied load of tire, W_{total}. Thus, if the calculated vertical reaction W_r of tire firstly exceeds the total tire weight, then we may understand that the vertical equilibrium of reaction becomes satisfied. Then, the translational condition of tire, V_w, and the rotation condition Ω_w w.r.t. tire center node is applied and, as the result, tire rotates with the forced wheel slip of $s = (V_w - V_r)/V_w$ where $V_w = R\Omega_w$ and R is the corresponding tire radius.

Prior test calculation resulted in the difficulty of continuous simulation of tire rotation, we decided the new parameters for elemental density on FEM as shown in Table 4-2.

4.7.2 Result of deformation field

Fig.4-15 shows the obtained result of tire-soil deformation after tire rotates with the forced slip of 0.1, where the tire moves to right. At the sinkage of 1.19 cm, the tire vertical reaction W_r firstly exceeds the current total tire load of 465 N

Table 4-2: Used parameters for tire rotation simulation

Model	Element	E (MPa)	ν	ρ_e (kg/m^3)	μ_{i-s}
FEM	Rubber	20	0.46	5000	0.3
	Rim	206000	0.3	5000	NA
	Lower Soil	100	0.3	2000	0.3

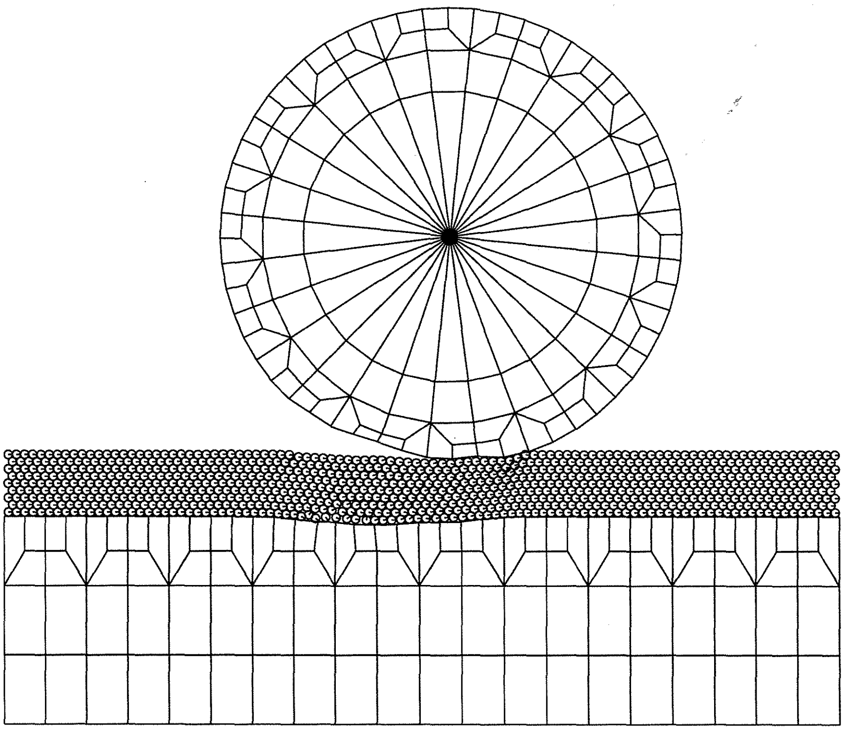

Figure 4-15: Result of tire rotation at 0.1 sec

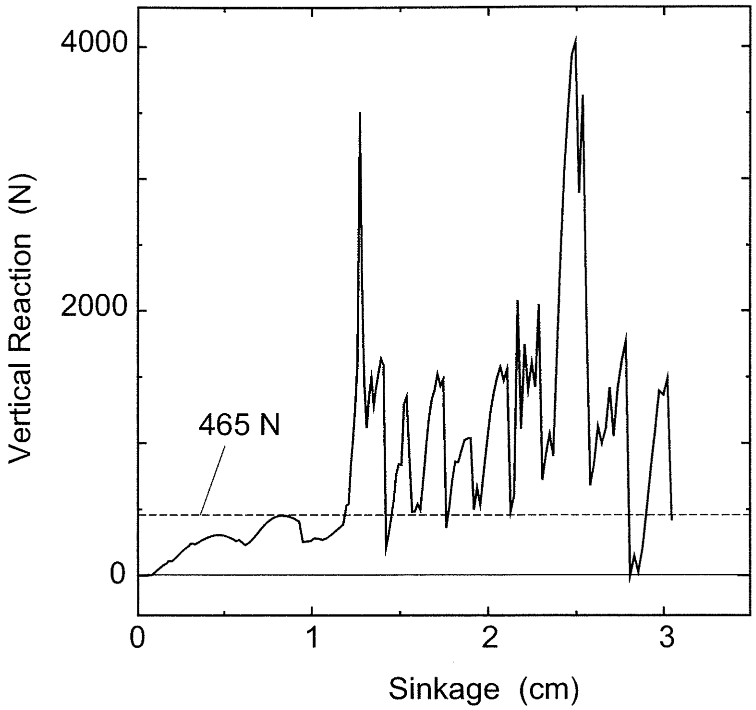

Figure 4-16: Result of total vertical reaction

(because of reduced density) at 0.0493 sec. In this Figure, it is noted that there exists an overlap of DEM elements just below the right side of contact region of tire and soil. This means the rotation of tire clearly affects the adjacent region of soil surface. After the total time steps of 10000 (0.1 sec), the deformation of tire and soil elements continued and some elements of FEM suffured severe local deformation. This implies the function to express the rebound of FEM might be insufficient. Further development and refinement of computer program should be added.

4.7.3 Result of vertical tire reaction

As for vertical reaction, the large vibratory trend of curve, as shown in Fig. 4-16, was observed. The result of vertical load-sinkage relationship is quite moderate when the total vertical load arrives 465 N when the tire begins to rotate and move. After this sinkage stage, the curve began to vibrate largely. It is possible that this type of vibration might be a result of forced constant wheel rotation and translation. Therefore, the calculation of load needs further detailed investigation.

4.8 Possible Extension to 3D Analysis

It is quite easy and straightforward to extend the 2D analysis program of FE-DEM to include 3D problems. Current analysis assumes the existance of thickness in 2D analysis, so that the approximate 3D analysis can be executed in nature.

In terms of full 3D analysis for FE-DEM, we have already prepared the FE tire mesh as stated in Chapter 2 and DEM elements are ready to use in the full 3D analysis. However, the contact detection in 3D becomes too complicated and, at this moment, the development of full 3D FE-DE analysis is still under preparation.

4.9 Concluding Remarks

We developed a combined FE-DEM code to analyze an agricultural tire-soil interaction problem. Simple contact algorithm is adopted in terms of computational speed. Small example problem of vertical tire sinkage on to a deformable soil was analyzed and we found that the sufficient accuracy of analysis for displacement field could be obtained.

As for tire rolling problem, it was found that the FE-DEM code could include the function of tire rotation and translation. The result of the basic analysis showed that the deformation could be analyzed qualitatively, but the total vertical reaction exhibited the heavy vibratory result.

Further investigation on the precise loading-unloading analysis of tire and soil contact region, the objective method of how to fix the optimum parameter for DEM, and the full 3D contact analysis by dynamic FE-DEM code should be done.

Chapter 5

Parallel Processing Method

5.1 Introduction

As stated in former Chapters, the possibility of increase in calculation time will be expected when we will apply DEM to tire-soil contact problems. One possibility of realization of DEM analysis is to limit the total number of DEM elements as in FE-DEM, proposed in Chapter 4. In terms of full 3D analysis of FE-DEM, we should prepare for the introduction of parallel processing which has become popular in massive computational mechanics. In this chapter, we focus on the fundamental problem of FE-DEM and discuss the possibility of application of parallel processing method to the FE-DEM.

5.2 MPI and PVM

With the recent development of fast internet communications, we can utilize distributed PC systems which are in general networked via ethernet. There are two groups of communication-based, or message passing, parallel processing, one is MPI (Message Passing Interface) and the other is PVM (Parallel Virtual Machine)[5].

PVM has been develped in order to create a "virtual" PC system, which consists of a wide variety of computer hardware, from PC to super computers. PVM consists of two units, one is daemon which must be running on each computer in PVM network, and the other is routine libraries which should be linked to user application program. System with PVM is schematically shown in Fig.5-1.

MPI is the standardized set of message passing libraries for API (Application Programming Interface) which was announced in early '90s. Nowadays, MPI becomes the standard for message passing parallel processing for distributed memory

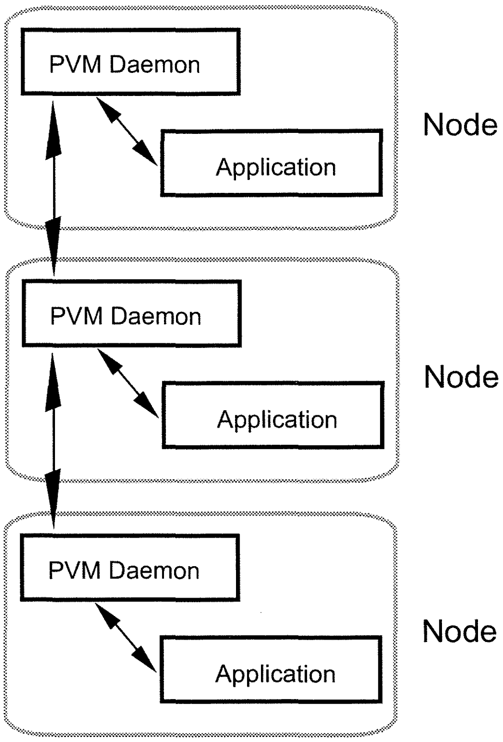

Figure 5-1: PVM model

parallel computers. Fig.5-2 shows the idea of MPI model. In MPI environment, the alloted number of processes are generated when the application is executed. The communication in MPI is two types; one is point-to-point, and the other is collective.

5.3 Analysis of Dynamic FE-DEM Program

5.3.1 Current performance

Firstly, the most significant part of calculation in the developed FE-DE method was checked by gprof using the vertical sinkage problem. Obtained result is summrarized in Table 5-1. This table shows that the subroutines pcontx and actf1 which are used for contact check and contact reaction calculation in DEM are two most time consuming parts within the program. Moreover, the time for mvmult and matmux are also consumed and these parts are mainly called in FEM part. Therefore, the efforts should be added to parallelize the DEM calculation in FE-DEM. At the same time, FEM programs need to be tuned up so that the effect on total computational time can be reduced.

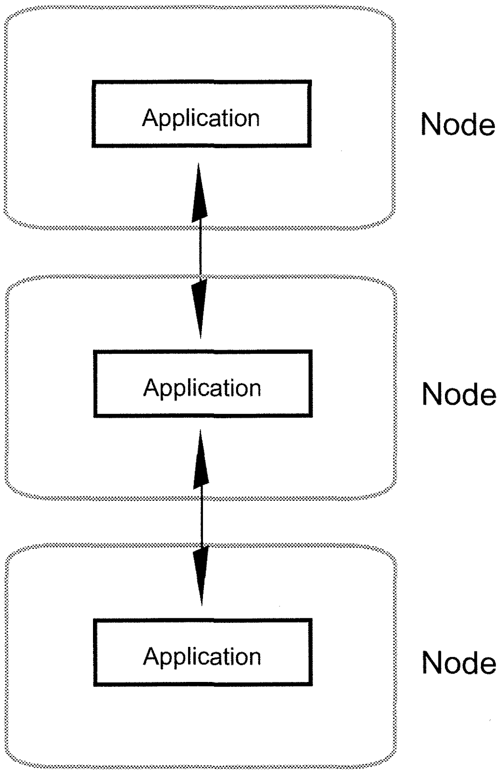

Figure 5-2: MPI model

Table 5-1: Result of gprof on FE-DEM program

% time	cumulative seconds	self seconds	calls	self ns/call	total ns/call	name
23.72	15.79	15.79	7168000	2202.15	3266.41	pcontx_
16.07	26.48	10.70				main
11.56	34.17	7.69	10024042	767.16	767.16	actf1_
9.11	40.23	6.07	15360000	394.86	394.86	mvmult_
8.36	45.80	5.56	15360000	361.98	361.98	matmux_
7.78	50.98	5.18	7680000	674.48	674.48	matran_
3.81	53.51	2.54	8000	316875.00	316875.00	tinteg_
3.80	56.04	2.53	11520000	219.62	219.62	null_
2.39	57.63	1.59	7680000	207.03	207.03	formb_
1.62	58.71	1.08				OutFormat
1.51	59.72	1.00	7680000	130.86	130.86	twoby2_
1.10	60.45	0.73	869783	839.29	839.29	actdfx_

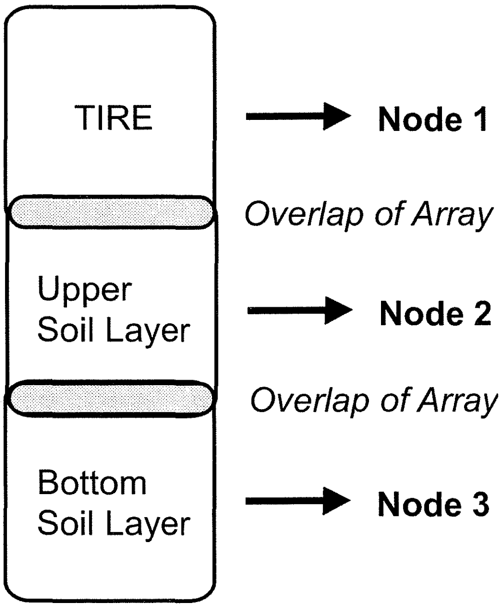

Figure 5-3: Schematic strategy for FE-DEM program

5.3.2 Possible parallel strategy for proposed FE-DEM

As shown in Table 5-1, DEM contact check and contact reaction calculation will become bottleneck when 3D extension of FE-DEM is considered. The possible clue to reduce of this problem and to apply the parallel processing to FE-DEM is summarized as follows;

- FEM and DEM are devided and processed by each CPU in parallel system separately.
- Contact check and reaction calculation part is devided into multiple processors in parallel system.
- Further modification of program, such as the development of fast algorithms, may be introduced.

With these discussions stated above, the following strategy is proposed for dynamic FE-DEM; i.e. one node is for tire (FEM), node two processes upper soil (DEM), the third node is for bottom soil (FEM), and the fourth node is used for the control of all program and input/output of data, if we use 4-node parallel processing system. The schematic figure is shown in Fig.5-3. As for the overlapped part which means the contact region, the array data on force and displacement are to be arranged so that they can be communicated between corresponding nodes.

5.4 Concluding Remarks

As the one of possible and useful tool, we investigated the possible application of message passing parallel processing for FE-DEM. The followings are clarified.

- In the developed FE-DEM program, contact check and contact reaction subroutines in DEM are the two most time-consuming parts.
- If the parallel processing system is introduced, FEM and DEM in FE-DEM program should be allocated to two nodes and there may be possible to devide the contact check loop in DEM into more than two parallel processes.

Chapter 6

Conclusion

Firstly, the preparation of FE mesh for tire was done. We developed two 3D FE mesh generation programs; one for smoothed tires, and the other for traction tires. Generation of FE mesh for traction tire could successfully demonstrated by using a typical tire data.

The capability of DEM to include the effect of wheel lug was also verified by the recent development of 2D DEM program in our Agricultural Systems Engineering Laboratory. The problem of large computational time is expected when 3D DEM program is developed.

The possibility of dynamic FE-DE method was investigated in terms of the application in soil-tire system. We assumed that a tire and deep soil layer could be modeled as FEM and soil surface layer as DEM. We proposed a simple algorithm of FE-DE coupled method and sample program was developed that could solve some basic terramechanics problems in order to verify our idea.

We developed a combined FE-DEM code to analyze an agricultural tire-soil interaction problem. Simple contact algorithm is adopted in terms of computational speed. Small example problem of vertical tire sinkage on to a deformable soil was analyzed and we found that the sufficient accuracy of analysis for displacement field could be obtained.

As for tire rolling problem, it was found that the FE-DEM code could include the function of tire rotation and translation. The result of the basic analysis showed that the deformation could be analyzed qualitatively, but the total vertical reaction exhibited the heavy vibratory result.

Further investigation on the precise loading-unloading analysis of tire and soil contact region, the objective method of how to fix the optimum parameter for DEM, and the full 3D contact analysis by dynamic FE-DEM code should be done.

As the one of possible and useful tool, we investigated the possible application of message passing parallel processing for FE-DEM. The followings are clarified.

- In the developed FE-DEM program, contact check and contact reaction subroutines in DEM are the two most time-consuming parts.
- If the parallel processing system is introduced, FEM and DEM in FE-DEM program should be allocated to two nodes and there may be possible to devide the contact check loop in DEM into more than two parallel processes.

As summarized above, we can conclude that the dynamic FE-DE Method will become a tool for computational soil-tire interaction problems.

Throuout of this project, the assistance given by former Master Program Student, Mr. Taku IBUKI, and by current Master Program Student, Mr. Hayato FUJII, of Agricultural Systems Engineering Laboratory, Division of Environmental Science & Technology, Kyoto University, are fully acknowledged. The author also expresses his sincere gratitude to current Doctoral Program Students, Messrs. Masatoshi MOMOZU and Yoshiyuki KAWASE, for their cooperation in computer programming. Tire data preparation and various discussions given by The Tire Research Department of Bridgestone Corporation are also appreciated.

Bibliography

[1] Aubel, Th. FEM-Simulation der Wechselwirkung zwischen Reifen und nachgiebiger Fahrbahn. *Automobiltechnische Zeitschrift* 1992: 94: 662-667.

[2] Cundall, P. A. and O. D. L. Strack. Discrete numerical model for granular assemblies. *Géotechnique* 1979: 29(1): 47-65.

[3] Fervers, C. W. Tyre profile effects on wheel-soil interaction. Proc. 12th Int. Conf. ISTVS, Beijing, PRC, 1996: 203-214.

[4] Fujii, H. An analysis of lugged wheel performance by DEM for the development of wheel for a lunar rover. Unpublished Master Thesis, Division of Environmental Science & Technology, Graduate School of Agriculture, Kyoto University, Kyoto, 2003. (in Japanese)

[5] Geist, A., Bequelin, A., Dongarra, J. Jiang, W.C., Manchek, R. and Sunderam, V. PVM: Parallel Virtual Machine–A User's Guide and TUtorial for Networked Parallel Computing–, The MIT Press, Cambridge, MS, USA, 1994.

[6] Hiroma, T., Y. Ohta, and T. Kataoka. Analysis of the soil deformation beneath a wheel by finite element method (Part 3). *J. Japanese Society of Agricultural Machinery* 1994: 56(6): 3-10. (in Japanese)

[7] Horner, D. A., J. F. Peters, and A. Carrillo. Large scale discrete element modeling of vehicle-soil interaction. *J. Engineering Mechanics* 2001: 127(10): 1027-1032.

[8] Johnson, K. L. Contact mechanics. Cambridge Univ. Press, Cambridge, UK, 1985.

[9] Kanto, Y. and G. Yagawa. A dynamic contact buckling analysis by the penalty finite element method. *Int. J. Numerical Methods in Engineering* 1990: 29: 755-774.

[10] Liu, C. H. and J. Y. Wong. Numerical simulations of tire-soil interaction based on critical state soil mechanics. *J. Terramechanics* 1996: 33(5): 209-221.

[11] Lu, Z., S.C. Negi and J.C. Jofreit. A numerical model for flow of granular materials in silos, Part 1: Model development. *J. Agricultural Engineering Research* 1997: 68: 223-229.

[12] Momozu, M., A. Oida and H. Nakashima. Simulation of shear box test by the Distinct Element Method, Proceedings of the 6th Asia-Pacific ISTVS Conference, Bangkok, 2001: 181-188.

[13] Nakashima, H. and J. Y. Wong, A three-dimensional tire model by the finite element method. *J. Terramechanics* 1993: 30(1): 21-34.

[14] Nakashima, H. Numerical simulation of agricultural tire-soil contact problem by 3D finite element method. Final Report on Research Project (Number: 03660260) under Grant-in-aid for Scientific Research (C) for 1991 to 1994 from The Ministry of Education, Science and Culture of Japan, 1994.

[15] Nakashima, H. and M. Yamazaki. 3D FEM procedure for static sinkage problems of tire on soil surface. Proc. ARBIP95, Japanese Society of Agricultural Machinery, Kobe, 1995: 245-252.

[16] Nakashima, H. and A. Oida. Simulation of soil-tire interaction by a coupled Distinct Element-Finite Element Method. Proc. 6th AP-ISTVS Conf., Bangkok, 2001: 59-63.

[17] Oida, A., S. Ohkubo, and H. Schwanghart. Effect of tire lug cross section on tire performance simulated by distinct element method. Proc. 13th Int. Conf. ISTVS, Munich, Germany, 1999: 345-352.

[18] Oida, A. and S. Ohkubo. Application of DEM to simulate interaction between soil and tire lug. *Agricultural & Biosystems Engineering* 2000: 1(1): 1-6.

[19] Oida, A. Application of distinct element method to analyze machine-soil interaction, Final Report on Research Project (Number: 09660273) under Grant-in-aid for Scientific Research (C) for 1997 to 2000 from The Ministry of Education, Culture, Sports, Science and Technology of Japan, 2001.

[20] Pan, X.D. and M.B. Reed. A coupled distinct element-finite element method for large deformation analysis of rock masses. *Int. J. Rock Mech. Min. Sci. & Geomech. Abstr.* 1991: 28(1): 93-99.

[21] Smith, I.M. and D.V. Griffiths. Programming the finite element method. 2nd Edition, John Wiley & Sons, UK, 1988.

[22] The Society of Powder Technology, Japan (Ed.). Introduction of computer simulation for particulate media. Sangyo-tosho, Tokyo, 1998.

[23] Ueno, M., K. Hashiguchi, Y. Nose, F. Koyama, K. Uchiyama, and H. Izumi. Analysis of soil-wheel interaction by elastoplastic finite element method. Proc. 10th Int. Conf. of ISTVS, Kobe, 1990: 253-264.

[24] Yong, R. N. and E. A. Fattah. Prediction of wheel-soil interaction and performance using the Finite Element Method. *J. Terramechanics* 1976: 13(4): 227-240.

[25] Yong, R. N., E. A. Fattah, and P. Boonsinsuk. Analysis and prediction of tyre-soil interaction and performance using finite elements. *J. Terramechanics* 1978: 15(1): 43-63.

[26] Zhong, Zhi-Hua. Finite element procedures for contact-impact problems. Oxford Univ. Press, Oxford, UK, 1993.

Appendix A

Program List

A.1 Main Program

```
C=======================================================
C
C Program: fdem99.f
C
C    2D Elasto-plastic & Dynamic FEM+DEM Analysis Code
C    Explicit Time Integration & Lumped Mass
C
C    by Hiroshi NAKASHIMA
C    originally for VT-Alpha300/433AXP on Linux/Alpha
C
C    Master  Program: fdem99.f
C    Library Program: exsub.f
C
C    Compile Option:
C    (Intel): ifc -O2 fdem99.f exsub.f -lm -o fdem99
C
C              Ver. 3.1  May 8, 2003
C=======================================================
      IMPLICIT REAL(8) (A-H,O-Z)
C
      PARAMETER(NPMAX=1000,NEMAX=1000,INO=700,NFMAX=500)
      parameter(ngmax=npmax*2)
      parameter(ni=2000,nc=20000)
      PARAMETER(MAXBN=100)
c      PARAMETER (NGX=250,NGY=400)
C
      CHARACTER(20) FILE1

      common /elempm/np,ne,np0,ne0,np1,ne1,npd
      common /convuu/r2d,d2r
      common /demscl/zmax,wmax,zmin
      common /conm/  dt,fri,frw,ev,ew,po,pow,so,g,de,pi
      common /posi/  x0(ni),z0(ni),qq(ni)
c      common /posp/xp(ni),zp(ni),qp(ni)
      common /velo/  u0(ni),v0(ni),f0(ni)
      common /forl/  xf(ni),zf(ni),of(ni)
      common /lforc/ en(ni,14),es(ni,14)
      common /wepr/  rr(ni),wei(ni),pmi(ni)
      common /celx/  n,idx,idzi,ipz,w,c,ncl(nc),nncl(ni)
      common /accel/ du0(ni),dv0(ni),df0(ni)
      common /demcc/ je(ni,14)
      common /dpmm/  u(ni+3),v(ni+3),f(ni+3)
      common /tpara1/ef1,pof1
      common /tpara2/ef2,pof2
      common /spara1/ef11,pos11
      common /presdd/dispre
      common /tpara3/rhof1,rhof2
      common /spara2/rhos1
      common /femer/ lnode(NEMAX,5)
      common /sscal/ zfmin
      common /itest/ izin,ixin
      common /cchek/ ifcont(npmax)
      common /nodsf/ isnode(npmax)
c      common /bcstat/nbc(npmax,2)
C
      REAL*8 DEE(3,3),COORD(4,2),VSAMP(4,2)
      REAL*8 XJAC(2,2),XJAC1(2,2)
      REAL*8 DER(2,4),DERIV(2,4),BEE(3,8),ELD(8)

      REAL*8 EPS(4),SIGMA(4),BT(8,3),ELOAD(8),BLOAD(8)
      REAL*8 SX(NEMAX,4),SY(NEMAX,4),TXY(NEMAX,4),SZ(NEMAX,4)
      REAL*8 EX(NEMAX,4),EY(NEMAX,4),GXY(NEMAX,4),EZ(NEMAX,4)
      real*8 stress(4),btdb(8,8),dbee(3,8),ekm(8,8)
c      real*8 fun(8)
      real*8 alax(ngmax,ngmax),blax(ngmax),vw(ngmax)
      real*8 wload(ngmax),sinks(ngmax)
      DIMENSION isp(2,NEMAX),itype(NEMAX),knf(nemax,8)
      DIMENSION LG(8)
      DIMENSION IDEM(NPMAX)
      DIMENSION IBC(MAXBN),nbc(npmax,2)
c      intrinsic dtime
      real tarray(2),tresult
C
      DATA IDEE,IBEE,idbee,IH/4*3/
      DATA IJAC,IJAC1,IDER,IDERIV,IT/5*2/
      DATA ICOORD,NOD/2*4/,IBT,IDOF,ibtdb,ikm/4*8/,NGP/2/

c      call etime(tarray,result)
      PI =4.0d0*ATAN(1.0d0)
      D2R=PI/180.0d0
      R2D=180.0d0/PI
      G  =9.80665d2

      write(*,*) ' FDEM step number='
      read(*,*) istep
c

      igaus=ngp*ngp
      do j=1,igaus
         do i=1,nemax
            sx  (i,j)=0.0d0
            sy  (i,j)=0.0d0
            txy(i,j)=0.0d0
            sz  (i,j)=0.0d0
            ex  (i,j)=0.0d0
            ey  (i,j)=0.0d0
            gxy(i,j)=0.0d0
            ez  (i,j)=0.0d0
         end do
      end do
C
C          Initial value for DEM
C
      do i=1,npmax
         nbc(i,1)=0
         nbc(i,2)=0
      end do

C              IDEM: element index of FEM or DEM
C
      CALL NULVEI(IDEM,NEMAX)
C=======================================================
C     FEM  Data Load
C=======================================================
      call ALLDAT(NBC,NB,IBC)
c
      do i=1,NPMAX
         u(i)   =0.0d0
         v(i)   =0.0d0
         f(i)   =0.0d0
         u0(i)  =0.0d0
         v0(i)  =0.0d0
         f0(i)  =0.0d0
         du0(i)=0.0d0
         dv0(i)=0.0d0
         df0(i)=0.0d0
      end do

c
c     KNF: DOF indicator for Node Num in an element
c
      do i=1,ne
         jq=0
         do j=1,4
            jq=jq+1
            k=2*lnode(i,j)-1
```

```
            knf(i,jq)=k
            jq=jq+1
            k=2*lnode(i,j)
            knf(i,jq)=k
         end do
      end do
C
C     NBC: location of free movement in serial number
C
c      CALL SETBC2(NBC,NPMAX,NBV,NO,IBF,NP,NB,NF)
      CALL SURFFE(isp,itype,nsf)
C=======================================================
C    DEM Data Load
C=======================================================
      NUMELM=npd
      N=npd
      call DEDATA
c
      do i=NE+1,NUMELM
         do j=1,14
            en(i,j)=0.0d0
            es(i,j)=0.0d0
            je(i,j)=0
         end do
      end do
c
C     Index for FEM(idem(*)=0) or DEM(idem(*)=element #)
C
      NEDF=NE+NUMELM
      NPDF=NP+NUMELM
c      write(*,*) 'NP,NE,NPDF,NEDF',np,ne,npdf,nedf
c      stop
      do I=1,NUMELM
         IDEM(NE+I)=NP+I
      end do
      call GAUSSV(VSAMP,igaus)
C======================================================
C     Explicit Integration Loop
C======================================================
      TI=0.0d0
C
      zini=z0(155)
      k4=0
c
      DO 10 JJ=1,ISTEP
         TI=TI+DT
         write(*,*)
         write(*,9103) jj,TI
 9103    format('STEP[',i6,']   TIME=',e12.3,'[sec]')
C
C                             Time increment: dt
C                               Current Time: ti
C
C  ** Initialize of Force Vector for each time step
C
         do i=1,NPDF
            xf(i) =0.0d0
            zf(i) =0.0d0
            of(i) =0.0d0
            wei(i)=0.0d0
            pmi(i)=0.0d0
         end do
c
c  ** Setup of DEM grid
c
         call fposir(rmax)
         call inmat
         call ncel
C------------------------------------------------------
C     Elemental Stress-Strain Relationship
C------------------------------------------------------
         if(jj.eq.1) allw=0.0d0
C***********************************NE: All FE Elem Num
         DO 30 IE=1,NE
C***********************************
C                                            **  For FEM
C
            ste =0.0d0
            AREA=0.0d0
            call nulvec(BLOAD,idof)
            call nulvec(eps,4)
            CALL GCOUNT(LG,IE,LNODE,NBC,NEMAX,NPMAX)
            DO I=1,4
               ILL=LNODE(IE,I)
               COORD(I,1)=x0(ILL)
               COORD(I,2)=z0(ILL)
               if(nbc(ill,1).ne.1) then
                  ELD(2*I-1)=u(ILL)
               else
                  eld(2*I-1)=0.0d0
               end if
               if(nbc(ill,2).ne.1) then
                  ELD(2*I   )=v(ILL)
               else
                  eld(2*I   )=0.0d0
               end if
            END DO
            CALL NULL(DEE,IDEE,IH,IH)
c
c      TIRE: 1--NE0; SOIL: NE0+1--
c
            if(IE.gt.NE0) then
               EE1=ef11
               VV1=pos11
               rho=rhos1
            end if
            if(IE.le.NE0) then
               if(lnode(ie,5).le.10) then
                  EE1=ef1
                  VV1=pof1
                  rho=rhof1
               else
                  EE1=ef2
                  VV1=pof2
                  rho=rhof2
               end if
            end if

            CALL FMDEPS(DEE,IDEE,EE1,VV1)
            call null(ekm,idof,idof,idof)
c            ig=0
C---------------------------------------------
C                         Gauss Integration Loop
C---------------------------------------------
            DO 50 IG=1,ngp*ngp
c               write(*,*) 'ie,ig=',ie,ig
               CALL FORMLV(DER,IDER,VSAMP,4,IG)
               CALL MAT-
MUX(DER,IDER,COORD,ICOORD,XJAC,IJAC,
     &               IT,NOD,IT)
               CALL TWOBY2(XJAC,IJAC,XJAC1,IJAC1,DET)
               CALL MAT-
MUX(XJAC1,IJAC1,DER,IDER,DERIV,IDERIV,
     &                  IT,IT,NOD)
               CALL NULL(BEE,IBEE,IH,IDOF)
               CALL FORMB(BEE,IBEE,DERIV,IDERIV,NOD)
               QUOT=DET
               if(det.le.0.0d0) then
                  write(*,*) 'FEM ER-
ROR in Elem ',ie,'* DET=',det
                  stop
               end if
c                                    area = pho
               AREA=AREA+QUOT*RHO
C
C                                sigma = D epsilon
C
               CALL MVMULT(DEE,IDEE,EPS,IH,IH,SIGMA)

               CALL MATRAN(BT,IBT,BEE,IBEE,IH,IDOF)
               CALL MVMULT(BT,IBT,SIGMA,IDOF,IH,ELOAD)

C                              BLOAD = int BT*SIGMA dV
               DO K=1,idof
                  BLOAD(K)=BLOAD(K)+ELOAD(K)*QUOT
               END DO
C-----------------------------------------------------
C                         End of Gauss Integration Loop
C-----------------------------------------------------
 50         CONTINUE
c
c        global mass & inertia term for FEM
c
            do k=1,4
               ill=lnode(ie,k)
               wei(ill)=wei(ill)+area*0.25d0
               pmi(ill)=1.0d0
            end do
c
c        Weight of Tire Calculation
c
            if((jj.eq.1).and.(ie.le.ne0)) allw=allw+area*g/100.0d0
c
c
c
c                  subtract BT*sigma from FORCE vector
c
               J1=0
               DO J=1,4
                  LI=LNODE(ie,J)
                  DO K=1,2
                     J1=J1+1
                     if(k.eq.1) then
                        xf(li)=xf(li)-bload(j1)
                     else
                        zf(li)=zf(li)-bload(j1)
```

```
                     end if
                  END DO
               END DO
C------------------------------------------------------
C                                 End of FE Element Loop
C------------------------------------------------------
 30      CONTINUE
C======================================================
C                            Contact of FEM-DEM Check
C======================================================
c         write(*,*) 'FE-DE Contact in ',jj
c
         elimi=1.0d-3
c         jcont=0
         do i=1,npdf
            ifcont(i)=0
         end do
C------------------------------------------------------
C                         Loop of FEM Surface Element
C
         DO 44 I=1,NSF
C                             L1--LNODE(*,4);L2--
LNODE(*,3)
C                             L3--Location of Surface
C                             L3=1 for tire; L3=2 for soil
            L3  =itype(I)
            L1  =ISP(1,I)
            L2  =ISP(2,I)
            XL1 =X0(L1)
            XL2 =X0(L2)
            YL1 =Z0(L1)
            YL2 =Z0(L2)
            delx=XL2-XL1
            dely=YL2-YL1
C
C                           ZLO: Boundary element length
C
            ZLO=dsqrt(delx*delx+dely*dely)
C*****************************************************
C                                 ---  DEM ELEMENT LOOP
C*****************************************************
            do 46 L=NE+1,NEDF
C                                 Select surface DEM
C
               if(lnode(L,5).eq.0) goto 46
               if((L3.eq.1).and.(lnode(L,5).eq.100 )) goto 46
               if((L3.eq.2).and.(lnode(L,5).eq.99  )) goto 46
c
C                                  lk: DEM Elem number
               lk=idem(L)
               if(ifcont(lk).ne.0) goto 46
c
c                   Coord of DEM elem
c
               UDEX=X0(LK)
               UDEZ=Z0(LK)
               if(UDEX.GT.(XL2+0.001)) goto 46
               if(UDEX.LT.(XL1-0.001)) goto 46
C
C                     contact check based on up-
dated coordinate
C
               CALL LNORML
     &         (UDEX,UDEZ,UPXL,UPZL,sin1,cos1,XL1,YL1,XL2,YL2,L3)
c
               GZAI=-1.0d0+2.0d0*UPXL/ZLO
C
C       If DEM elem is out of Boundary Line Element:
C
               IF(gzai.Lt.-1.001d0) GOTO 46
               IF(gzai.Gt. 1.001d0) GOTO 46
C
C       RR(j) = Radius of DEM ; REGI = Length of nor-
mal foot
C
               gap=UPZL-RR(LK)
               if(gap.gt.0.0d0) then
                  if(ifcont(lk).ne.0) then
                     ifcont(lk)=0
                     call detach(LK,14,L3)
                  end if
                  goto 46
               end if
               if(lk.ge.500) then
                  write(*,9008) LK,gap
               end if
 9008          format('---
contact at ',i4, '  ** gap =',f12.8)
```

```
C
               as=sin1
               ac=cos1
c
               call LFUNC2(GZAI,SS1,SS2)
               uj=u(L1)*SS1+u(L2)*SS2
               vj=v(L1)*SS1+v(L2)*SS2
               jk=14
               call actdfx(LK,L3,jk,as,ac,gap,rxx,rzz,uj,vj,L3)
c
               xf(L1)=xf(L1)-rxx*SS1
               xf(L2)=xf(L2)-rxx*SS2
               zf(L1)=zf(L1)-rzz*SS1
               zf(L2)=zf(L2)-rzz*SS2
C                               >>>> End of DEM Element
 46         continue
C                              End of Boundary Elements
 44      continue
c
         tvert=0.0d0
         do L=NE+1,NEDF
C                                ---> lk: node number
            lk=idem(L)
            if(ifcont(lk).eq.7) then
               write(*,*) 'dem num=',lk
               tvert=tvert+dabs(zf(lk))
            end if
         end do
C
C       End of FEM-DEM Calculation
C
C*********************************************************
C       DEM-DEM Calculation
C*********************************************************
         do L=NE+1,NEDF
            iel=idem(L)
c            write(*,*) 'DEM ',iel
c
            call wcont(iel)
            call pcontx(iel,rmax)
         end do
C
C       End of DEM-DEM
c
C=====================================================
C    Time Integration                  NPDF: All Nodes
C=====================================================
c
c            coordinate update
c
         call tinteg(npdf,np0,np,g,dt,nbc)
c
               write(*,8099)
c
         do i=1,npdf
            x0(i)=x0(i)+u(i)
            z0(i)=z0(i)+v(i)
            qq(i)=qq(i)+f(i)
            if((i.eq.1).or.(i.eq.221)) then
               write(*,8088) i,x0(i),z0(i),u(i),v(i),xf(i),zf(i)
 8099          format('i  --x,z,u,v,xf,zf---')
 8088          format(i5,6f12.3)
            end if
         end do
c
         if(mod(jj,50).eq.0) then
            k4=k4+1
            sinks(k4)=-(z0(155)-zini)
            wload(k4)=tvert
         end if
         write(*,*) 'Current Tire Sink  [cm] =',
     *     z0(155)-zini,v(1)
         write(*,*) 'Current Tire Load   [N] =',tvert
         write(*,*) 'Total Tire Load     [N] =',allw,ti,jj
         do i=1,NE
            if(lnode(i,5).ne.1) goto 4033
            ix1=lnode(i,1)
            if(i.eq.ne) then
               ix1=lnode(i,2)
            end if
            if(i.eq.ne0+1) then
               ix=lnode(i,2)
c               write(*,7098) ix,xf(ix),zf(ix)
               ix1=lnode(i,1)
            end if
 7098       format(i4,2f12.7)
 4033       continue
```

```
         end do

         if(allw.le.tvert) goto 868

C========================================================
 850     continue
C========================================================
C               END OF TIME INTEGRATION LOOP
C========================================================
 10   CONTINUE
C
 1200 format(i5,2x,5F12.5)
 1201 format(i5,2x,7F12.4)
 1205 format(i5,2x,3F12.4)
 1206 format(6i5)
 1207 format(2i4,1x,F12.5)
 1000 FORMAT(i5,2x,4F10.5)
C
C      Result File Save
C
 868  continue

      FILE1='fdemtestx.res'
      OPEN(14,FILE=FILE1)
      write(14,*) NP0,NE0,NP1,NE1,NPD
      do i=1,NPDF
         write(14,1205) i,x0(i),z0(i),qq(i)
      end do
      do i=1,NEDF
         write(14,1206) i,(lnode(i,k),k=1,5)
      end do
      close(14)
C
      OPEN(16,FILE='fdemtestx.stp')
      write(16,1206) istep
      close(16)
      sbuf=0.0d0
      open(14,FILE='fdemloadx.res')
      write(14,*) k4+1
      write(14,*) sbuf,sbuf
      do i=1,k4
         write(14,*) sinks(i),wload(i)
      end do
      close(14)
      STOP
      END
C==============================
      subroutine tinteg(nxx,np0,np,g,dt,nbc)
      implicit real*8(A-H,O-Z)
      PARAMETER(NPMAX=1000)
      parameter (ni=2000)
      common /posi/x0(ni),z0(ni),qq(ni)
      common /velo/u0(ni),v0(ni),f0(ni)
      common /for1/xf(ni),zf(ni),of(ni)
      common /wepr/rr(ni),wei(ni),pmi(ni)
      common /accel/du0(ni),dv0(ni),df0(ni)
      common /dpmm/u(ni+3),v(ni+3),f(ni+3)
      dimension nbc(npmax,2)

      do i=1,nxx
         ww=wei(i)*1.0d0
         if(nbc(i,2).ne.1) then
            if(wei(i).eq.0.0d0) then
               write(*,*) 'ERROR: wei 2=0  at ',i
               stop
            end if
            if(i.gt.np0) then
               accl=zf(i)/ww
            else
               accl=(zf(i)-wei(i)*g)/ww
            end if
            v0(i)=v0(i)+(dv0(i)+accl)*.5d0*dt
            dv0(i)=accl
         else
            dv0(i)=0.0d0
            v0 (i)=0.0d0
         end if

         if(nbc(i,1).ne.1) then
            if(wei(i).eq.0.0d0) then
               write(*,*) 'ERROR: wei 1=0  at ',i
               stop
            end if
            accl=xf(i)/ww
            u0(i)=u0(i)+(du0(i)+accl)*.5d0*dt
            du0(i)=accl
         else
            du0(i)=0.0d0
            u0(i) =0.0d0
         end if

         if(pmi(i).ne.0.0d0) then
            accl=of(i)/pmi(i)
            f0(i)=f0(i)+(df0(i)+accl)*.5d0*dt
            df0(i)=accl
         else
            df0(i)=0.0d0
            f0(i) =0.0d0
         end if

         if(nbc(i,2).ne.1) then
            vold=v(i)
            v(i)=(v0(i)*dt+v(i))/2.0d0
            vratio=v(i)/vold
         else
            v(i)=0.0d0
         end if

         if(nbc(i,1).ne.1) then
            uold=u(i)
            u(i)=(u0(i)*dt+u(i))/2.0d0
            uratio=u(i)/uold
         else
            u(i)=0.0d0
         end if

         f(i)=(f0(i)*dt+f(i))/2.0d0
         if((i.eq.155).or.(i.eq.221)) then
            write(*,9008) i,xf(i),du0(i),u0(i),u(i)
            write(*,9009) i,zf(i),dv0(i),v0(i),v(i)
         end if
 9008    format('Node[',i4,']:xf,du0,u0,u=',4f14.4)
 9009    format('Node[',i4,']:zf,dv0,v0,v=',4f14.4)
      end do
      return
      end
C=========================================
      SUBROUTINE LNORML(X,Y,XL,YL,sin1,cos1,X1,Y1,X2,Y2,itype)
      IMPLICIT REAL*8 (A-H,O-Z)
C
C       .3(X,Y)
C  2------------------1
C
C  XL,YL: Calculated Local Coordinate of 3 w.r.t. 2(Tire)
C  XL,YL: Calculated Local Coordinate of 3 w.r.t. 1(Soil)
C  THETA: Calculated angle of 1-2 axis to horizontal axis
C
C

      DX=X2-X1
      DY=Y2-Y1
      DL=dsqrt(dx*dx+dy*dy)
c
      sin1=dy/dl
      cos1=dx/dl
C
      if(itype.eq.1) then
         XL= (X2-X)*COS1+(Y2-Y)*SIN1
         YL=-(X2-X)*SIN1+(Y2-Y)*COS1
      end if
      if(itype.eq.2) then
         XL= (X-X1)*COS1+(Y-Y1)*SIN1
         YL=-(X-X1)*SIN1+(Y-Y1)*COS1
      end if
      RETURN
      END
C
C=========================================
      SUBROUTINE LCORD2(GX,GX0,GXN,Y0,YN)
C
C   Find GZAI value on Local Coodinate on Line Element
C
C     Y0: Before Contact; YN: After Con-
tact w/o modification
C     GX: Found GZAI-value on modified contact
C     GX0: Before Contact GZAI; GXN: After Con-
tact GZAI on YN
C     If GX>=1 or <=0, outside of line element
C
      IMPLICIT REAL*8 (A-H,O-Z)
      if(y0.eq.yn) then
         gx=gx0
      else
         GX=(Y0*GXN-YN*GX0)/(Y0-YN)
      end if
      RETURN
      END
C=========================================
      SUBROUTINE LFUNC2(GX,SS1,SS2)
C
C   Local GZAI --> Shape Function of N_1, N_2
C
      IMPLICIT REAL*8 (A-H,O-Z)
      SS1=0.5d0*(1.0d0-GX)
      SS2=0.5d0*(1.0d0+GX)
      RETURN
```

```
      END
C========================================
      SUBROUTINE ALLDAT(NBC,NB,IBC)
C
      IMPLICIT REAL*8 (A-H,O-Z)
      PARAMETER(NPMAX=1000,NEMAX=1000)
      PARAMETER (MAXBN=100)
      parameter (ni=2000)
      common /elempm/np,ne,np0,ne0,np1,ne1,npd
      common /demscl/zmax,wmax,zmin
      common /posi/x0(ni),z0(ni),qq(ni)
      common /wepr/rr(ni),wei(ni),pmi(ni)
      common /femer/lnode(NEMAX,5)
      common /tpara1/ef1,pof1
      common /tpara2/ef2,pof2
      common /spara1/ef11,pos11
      common /presdd/dispre
      common /tpara3/rhof1,rhof2
      common /spara2/rhos1
      common /sscal/zfmin
C      REAL*8 FX(*),FY(*)
      DIMENSION NBC(NPMAX,*),IBC(MAXBN)
      CHARACTER*20 DNAME
C
C     Input data file should be linked to FORT.5
C
      WRITE(*,*) ' ==> Mesh Data File Name ?'
395   READ(*,'(1A14)') DNAME
      IF(DNAME.EQ.'') THEN
        WRITE(*,*) ' ... Invalid File Name. Re-
enter please!.'
GOTO 395
      ENDIF
      OPEN(5,FILE=DNAME,STATUS='OLD')
C
      READ(5,*) NP0,NE0,NP1,NE1,NPD
c      write(*,*) NP0,NE0,NP1,NE1,NPD
C
      NP=NP0+NP1
      NE=NE0+NE1
      NP99=NP+NPD
      NE99=NE+NPD
C
      zmax= 10000.0
      wmax=-10000.0
      zfmin=10000.0
      DO I=1,NP99
         READ(5,*) IDUMMY,X0(I),Z0(I),RR(I)
         if((i.gt.np0).and.(i.le.np)) then
            if(wmax.lt.dabs(x0(i))) wmax=dabs(x0(i))
         end if
         if(i.gt.np) then
            if(zmax.lt.dabs(z0(i))) zmax=dabs(z0(i))
         end if
         if(i.le.np) then
            if(zfmin.gt.z0(i)) zfmin=z0(i)
         end if
      END DO
C
      DO I=1,NE99
         READ(5,*) IDUMMY,(LNODE(I,K),K=1,5)
      END DO
C
      CLOSE(5)
C---------------------------------------------------------
------
C     Input data file should be linked to FORT.5
C
C      WRITE(*,*) ' ==> FE B.C and Parameter File Name ?'
C 396  READ(*,*) DNAME1
C      IF(DNAME.EQ.'') THEN
C        WRITE(*,*) ' ... Invalid File Name. Re-
enter please!.'
C GOTO 396
C      ENDIF
      OPEN(15,FILE='allmesh2.prm',STATUS='OLD')
C
      READ(15,*) NB
C
      DO K=1,3
         DO I=1,NP
            NBC(I,K)=0
         END DO
      END DO
      DO I=1,NB
         READ(15,*) IBC(I),NBC(IBC(I),1),NBC(IBC(I),2)
c         write(*,*) IBC(I),NBC(IBC(I),1),NBC(IBC(I),2)
      END DO
C
c      tire: young's mod, posson's ratio, density
c      rim : young's mod, posson's ratio, density
c      soil: young's mod, posson's ratio, density
c
      READ(15,*) ef1 ,pof1 ,rhof1
      READ(15,*) ef2 ,pof2 ,rhof2
      READ(15,*) ef11,pos11,rhos1
C
C        Prescribed Displacement Condition
C            ---> load condition
      read(15,*) dispre
c
      close(15)

      RETURN
      END
C========================================
      SUBROUTINE DEDATA
C
C     READ INITIAL ELEMENT POSITION AND VELOCITY  --- DEM
C
      IMPLICIT REAL*8 (A-H,O-Z)
      common /conm/dt,fri,frw,ev,ew,po,pow,so,g,de,pi

      OPEN(12,FILE='sdemx.dat',STATUS='OLD')
C
C       de  : density of DEM element
C       dt  : time increment
C
      READ(12,*) de, dt
C
C        ev : Young's Modulus for DEM element
C        ew : Young's Modulus for FE wall
C        po : Poisson's Ratio for DEM element
C        pow: Poisson's Ratio for FE wall
C        fri: Coulomb Friction Coef for DEM
C        frw: COulomb Friction Coef for FE wall
C
      READ(12,*) ev,ew,po,pow,fri,frw
c      write(*,*) ev,ew,po,pow,fri,frw
c
      CLOSE(12)
      RETURN
      END
C========================================
      SUBROUTINE SURFFE(isp,itype,nsf)
C
C     DEFINE FEM SURFACE W.R.T. DEM
C
      IMPLICIT REAL*8 (A-H,O-Z)
      PARAMETER(NEMAX=1000)
      common /elempm/np,ne,np0,ne0,np1,ne1,npd
      common /femer/lnode(NEMAX,5)
ccc      common /nodsf/isnode(npmax),NSU
      integer ISP(2,nemax), itype(nemax)

      INS=0
      DO I=1,NE
         if(lnode(i,5).ne.1) goto 100

         J =LNODE(I,2)
         J1=LNODE(I,1)
         INS=INS+1
c         ISEE(INS)=I
c
c                itype=1 for tire; itype=2 for soil
C
         if(I.le.ne0) then
            itype(ins)=1
            ISP(1,INS)=J1
            ISP(2,INS)=J
         else
            itype(ins)=2
            ISP(1,INS)=J
            ISP(2,INS)=J1
c             write(*,*) ' Soil top=',I
         end if
 100     continue
      END DO
      NSF=INS
      RETURN
      END
C========================================
      SUBROUTINE EXTFOR(NP0,NPX,dispre)
C
C     External Force Calc.
C
      parameter (ni=2000,nc=20000)
      IMPLICIT REAL*8 (A-H,O-Z)
      common /for1/xf(ni),zf(ni),of(ni)

      DO I=1,NP0
         if(NPX.eq.i) then
            xf(I)=0.0d0
c             zf(I)=zf(i)-dispre
            zf(I)=-dispre
            of(I)=0.0d0
```

```
            write(*,*) 'gravity force=',i,zf(i)
         end if
      END DO
      RETURN
      END
C=======================================
      SUBROUTINE EXTDIS(dispre)
C
C     External Force Calc.
C
      IMPLICIT REAL*8 (A-H,O-Z)
      PARAMETER(NEMAX=1000)
      parameter (ni=2000)
      common /elempm/np,ne,np0,ne0,np1,ne1,npd
      common /dpmm/u(ni+3),v(ni+3),f(ni+3)
      common /femer/lnode(NEMAX,5)
      dimension irim(ni)
      do i=1,ni
         irim(i)=0
      end do

      do i=1,ne0
c        if(lnode(i,5).eq.11) then
c           do k=1,3
            do k=1,4
               j=lnode(i,k)
               if(irim(j).eq.0) then
                  u(j)= u(j)
                  v(j)= v(j)-dispre
                  f(j)= f(j)
                  irim(j)=1
                  if(j.eq.155) then
                  write(*,*) 'j,u,v=',j,u(j),v(j)
                  end if
               end if
            end do
c        end if
      end do
      RETURN
      END
C=======================================
      SUBROUTINE FSPARM(ALAX,IV,EKM,IKM,IE,NEMAX,KNF)
C
C      This subroutine assembles the element stiffness
C      matrix into the gloval matrix stored as ALAX
C
      IMPLICIT REAL*8 (A-H,O-Z)
      real*8 ALAX(IV,IV),EKM(IKM,IKM)
      INTEGER KNF(NEMAX,*)
      DO I=1,8
         J=KNF(IE,I)
         DO L=1,8
            J1=KNF(IE,L)
            IF(J.NE.0.AND.J1.NE.0) THEN
               ALAX(J ,J1)=ALAX(J ,J1)+EKM(I,L)
            END IF
         end do
      end do
      RETURN
      END
C=======================================
      subroutine fposir(rmax)
      implicit real*8 (a-h,o-z)
      parameter (ni=2000,nc=20000)
      common /elempm/np,ne,np0,ne0,np1,ne1,npd
      common /conm/dt,fri,frw,ev,ew,po,pow,so,g,de,pi
      common /wepr/rr(ni),wei(ni),pmi(ni)
      common /celx/n,idx,idzi,ipz,w,c,ncl(nc),nncl(ni)
      common /posi/x0(ni),z0(ni),qq(ni)
      common /demscl/zmax,wmax,zmin
      common /itest/izin,ixin
c
c      w=wmax
c
c      n0=np
      rmax=rr(np+1)
      rmin=rr(np+1)
      zmax=z0(np+1)
      zmin=z0(np+1)
C
      do i=2,npd
         k=np+i
         if(rr(k).gt.rmax) rmax=rr(k)
         if(rr(k).lt.rmin) rmin=rr(k)
         if(z0(k).gt.zmax) zmax=z0(k)
         if(z0(k).lt.zmin) zmin=z0(k)
      end do
      zz=zmax-zmin
      c =rmin*1.35d0
      idx =idint(wmax/c)+1
      idzi=idint(  zz/c)+1
      ixin=idx
      izin=idzi
      return
      end
C=======================================
      subroutine ncel
      implicit real*8 (a-h,o-z)
      parameter (ni=2000,nc=20000)
      common /elempm/np,ne,np0,ne0,np1,ne1,npd
      common /celx/n,idx,idzi,ipz,w,c,ncl(nc),nncl(ni)
      common /posi/x0(ni),z0(ni),qq(ni)
      common /lforc/en(ni,14),es(ni,14)
      common /demcc/je(ni,14)
      common /dpmm/u(ni+3),v(ni+3),f(ni+3)
      common /sscal/zfmin
c
      do ib=1,(idx*idzi)
         ncl(ib)=0
      end do
      do i=1,n
         k=np+i
         zst=z0(k)-zfmin
         ib=idint(zst/c)*idx+idint(x0(k)/c)+1
         ncl(ib)=k
         nncl(k)=ib
      end do
      return
      end
C=======================================
      subroutine inmat
      implicit real*8 (a-h,o-z)
      parameter (ni=2000,nc=20000)

      common /conm/dt,fri,frw,ev,ew,po,pow,so,g,de,pi
      common /wepr/rr(ni),wei(ni),pmi(ni)
      common /celx/n,idx,idzi,ipz,w,c,ncl(nc),nncl(ni)
      common /elempm/np,ne,np0,ne0,np1,ne1,npd
c
      so=1.0d0/2.0d0/(1.0d0+po)
      do i=1,npd
         k=np+i
c                Mass: wei(k)
c                Moment of Inertia: wei(k)*r^2/2
c
         wei(k)=pi*rr(k)*rr(k)*de
         pmi(k)=wei(k)*rr(k)*rr(k)*0.5d0
      end do
      return
      end
C=======================================
      subroutine wcont(i)
      implicit real*8 (a-h,o-z)
c
      parameter (ni=2000,nc=20000)
      common /demscl/zmax,wmax,zmin
      common /wepr/rr(ni),wei(ni),pmi(ni)
      common /celx/n,idx,idzi,ipz,w,c,ncl(nc),nncl(ni)
      common /dpmm/u(ni+3),v(ni+3),f(ni+3)
      common /posi/x0(ni),z0(ni),qq(ni)
      common /velo/u0(ni),v0(ni),f0(ni)
      common /forl/xf(ni),zf(ni),of(ni)
      common /lforc/en(ni,14),es(ni,14)
      common /elempm/np,ne,np0,ne0,np1,ne1,npd
      common /demcc/je(ni,14)
c
      xwi=x0(i)
      rwi=rr(i)
c
c   --- left wall (left wall is on x=0)
c
      jk=11
      j=np+npd+1
      if(xwi.le.rwi) then
         as= 0.0d0
         ac=-1.0d0
         gap=dabs(xwi)
         je(i,jk)=j
         call actf1(i,j,jk,as,ac,gap)
      else
         en(i,jk)=0.0d0
         es(i,jk)=0.0d0
         je(i,jk)=0
      end if
c
c   --- under wall
c
c       jk=12
c       j=n+2
c       if(zi.lt.ri) then
c          as=-1.0d0
c          ac= 0.0d0
c          gap=dabs(zi)
c          je(i,jk)=n+2
c          call actf(i,j,jk,as,ac,gap)
c       else
```

```fortran
c         en(i,jk)=0.0d0
c         es(i,jk)=0.0d0
c         je(i,jk)=0
c      end if
c
c   --- right wall  (wmax: width of soil bin)
c
      jk=13
      j=np+npd+3
      if((xwi+rwi).ge.wmax) then
         as= 0.0d0
         ac= 1.0d0
         gap=dabs(xwi-wmax)
         je(i,jk)=j
         call actf1(i,j,jk,as,ac,gap)
      else
         en(i,jk)=0.0d0
         es(i,jk)=0.0d0
         je(i,jk)=0
      end if
      return
      end
C=======================================
      subroutine pcontx(i,rmax)
      implicit real*8 (a-h,o-z)
      parameter (ni=2000,nc=20000)
      common /elempm/np,ne,np0,ne0,np1,ne1,npd
      common /wepr/rr(ni),wei(ni),pmi(ni)
      common /celx/n,idx,idzi,ipz,w,c,ncl(nc),nncl(ni)
      common /posi/x0(ni),z0(ni),qq(ni)
      common /forl/xf(ni),zf(ni),of(ni)
      common /lforc/en(ni,14),es(ni,14)
      common /demcc/je(ni,14)
      common /demscl/zmax,wmax,zmin
      common /sscal/zfmin
c
      xi=x0(i)
      zi=z0(i)-zfmin
      ri=rr(i)

      lup=idint((zi +2.d0*rmax)/c)
      lun=idint((zi -2.d0*rmax)/c)
      llf=idint((xi -2.d0*rmax)/c)
      lrg=idint((xi +2.d0*rmax)/c)

      if(lun.lt.0   ) lun=0
      if(llf.lt.0   ) llf=0
      if(lrg.ge.idx ) lrg=idx-1
      do 90 lz=lun,lup
         do 80 lx=llf,lrg
            ib=lz*idx+lx+1
            j=ncl(ib)
            if ((j.eq.0).or.(j.eq.i)) goto 80
            do 11 jj=1,10
               if (je(i,jj).eq.j) then
                  jk=jj
                  goto 70
               end if
 11         continue

            do 12 jj=1,10
               if (je(i,jj).eq.0) then
                  jk=jj
                  je(i,jj)=j
                  goto 70
               end if
 12         continue
 70         xj=x0(j)
            zj=z0(j)-zfmin
            rj=rr(j)
            gap=dsqrt((xi-xj)*(xi-xj)+(zi-zj)*(zi-zj))
            if (gap.lt.(ri+rj)) then
               if (i.gt.j) then
                  ac=(xj-xi)/gap
                  as=(zj-zi)/gap
                  j0=0
                  do 555 jj=1,10
                     if (je(j,jj).eq.i) then
                        j0=jj
                        goto 554
                     endif
 555              continue
 554              call actf1(i,j,jk,as,ac,gap)
                  en(j,j0)=en(i,jk)
                  es(j,j0)=es(i,jk)
                  j0=0
               endif
            else
 85            en(i,jk)=0.d0
               es(i,jk)=0.d0
               je(i,jk)=0
            endif
 80      continue
 90   continue
      return
      end
C=======================================
      subroutine actf1b(i,j,jk,as,ac,gap)
      implicit real*8 (a-h,o-z)
      parameter (ni=2000,nc=20000)
      common /elempm/np,ne,np0,ne0,np1,ne1,npd
      common /conm/dt,fri,frw,ev,ew,po,pow,so,g,de,pi
      common /wepr/rr(ni),wei(ni),pmi(ni)
      common /celx/n,idx,idzi,ipz,w,c,ncl(nc),nncl(ni)
      common /posi/x0(ni),z0(ni),qq(ni)
      common /velo/u0(ni),v0(ni),f0(ni)
      common /forl/xf(ni),zf(ni),of(ni)
      common /lforc/en(ni,14),es(ni,14)
      common /demcc/je(ni,14)
      common /dpmm/u(ni+3),v(ni+3),f(ni+3)
c
      nn=np+npd
      ri=rr(i)
      alpha=1.0d0
      beta=1.0d0
      if((j-nn).le.0) then
         rj=rr(j)
         dis=ri+rj-gap
         wei3=2.d0*wei(i)*wei(j)/(wei(i)+wei(j))
      else
         rj=0.0d0
         dis=ri-gap
         wei3=wei(i)
      end if
      enn=en(i,jk)
      if(enn.le.0.0d0) enn=1.0d0
      pois2 =1.0d0-po*po
      pois21=1.0d0-pow*pow
      if((j-nn).le.0) then
         exx1=pois2/ev
         exx2=pois2/ev
         requi=ri*rj/(ri+rj)
         ba1=dsqrt(4.0d0*(exx1+exx2)*requi*enn/pi)
         elx1=dlog(4.0d0*ri/ba1)-0.5d0
         elx2=dlog(4.0d0*rj/ba1)-0.5d0
         eknn=pi/(2.0d0*(exx1*elx1+exx2*elx2))
         ekss=eknn*so
         vnn=beta*dsqrt(4.d0*wei3*eknn)
         vss=so*vnn
      else
         exx1=pois2/ev
         exx2=pois21/ew
         requi=ri*rj/(ri+rj)
         if(rj.eq.0.0d0) requi=ri
         ba1=dsqrt(4.0d0*(exx1+exx2)*requi*enn/pi)
         elx1=dlog(4.0d0*ri/ba1)-0.5d0
         if(rj.eq.0.0d0) then
            elx2=0.0d0
         else
            elx2=dlog(4.0d0*rj/ba1)-0.5d0
         end if
         eknn=alpha*pi/(2.0d0*(exx1*elx1+exx2*elx2))
         ekss=eknn*so
         vnn=beta*dsqrt(4.d0*wei3*eknn)
         vss=so*vnn
      end if

      if(eknn.ne.0.0d0) then
         ddt=1.0d-1*dsqrt(wei3/eknn)
         if(ddt.lt.dt) then
            write(6,*) 'dt > ddt=',ddt,i,j,jk,eknn,wei(i)
            stop
         end if
      end if
C
      ui=u(i)
      uj=u(j)
      vi=v(i)
      vj=v(j)
      fi=f(i)
      fj=f(j)
c
      un= (ui-uj)*ac+(vi-vj)*as
      us=-(ui-uj)*as+(vi-vj)*ac+(ri*fi+rj*fj)
c
      if(en(i,jk).eq.0.d0) then
         if(un.ne.0.d0) us=us*dis/un
         un=dis
      end if
c
      en(i,jk)=en(i,jk)+eknn*un
      es(i,jk)=es(i,jk)+ekss*us
c
      dn=vnn*un/dt
      ds=vss*us/dt
```

```
      if(en(i,jk).lt.0.0d0) then
         en(i,jk)=0.0d0
         es(i,jk)=0.0d0
         dn=0.0d0
         ds=0.0d0
         je(i,jk)=0
         return
      else if ((j-nn).le.0) then
         frc=fri
      else
         frc=frw
      end if
      hn=en(i,jk)+dn
      hs=es(i,jk)+ds

      if(dabs(hs-frc*hn).gt.0.0d0) then
         hs=frc*dsign(hn,hs)
         ds=0.0d0
      end if
      xf(i)=-hn*ac+hs*as+xf(i)
      zf(i)=-hn*as-hs*ac+zf(i)
      of(i)= of(i)-ri*hs
c      write(*,*) '  * DEM-DEM: Cont Elem at ',i

      if(jk.le.10) then
         xf(j)=hn*ac-hs*as+xf(j)
         zf(j)=hn*as+hs*ac+zf(j)
         of(j)=of(j)-rj*hs
c         if((j.eq.315).or.(j.eq.314)) then
c            write(*,9000) i,j,zf(j)
c         end if
c         if((j.eq.363).or.(j.eq.364)) then
c            write(*,9000) i,j,zf(j)
c         end if
      end if
 9000 format('  * DEM-
DEM: Cont Elem[',i4,'][',i4,'] zf=',e15.7)
      return
      end
C=========================================
      subroutine actd-
fxb(i,j,jk,as,ac,gap,rxx,rzz,uj,vj,itt)
      implicit real*8 (a-h,o-z)
      PARAMETER(NPMAX=1000)
      parameter (ni=2000,nc=20000)
      common /elempm/np,ne,np0,ne0,np1,ne1,npd
      common /conm/dt,fri,frw,ev,ew,po,pow,so,g,de,pi
      common /wepr/rr(ni),wei(ni),pmi(ni)
      common /celx/n,idx,idzi,ipz,w,c,ncl(nc),nncl(ni)
      common /posi/x0(ni),z0(ni),qq(ni)
      common /velo/u0(ni),v0(ni),f0(ni)
      common /forl/xf(ni),zf(ni),of(ni)
      common /lforc/en(ni,14),es(ni,14)
      common /demcc/je(ni,14)
      common /dpmm/u(ni+3),v(ni+3),f(ni+3)
      common /tpara1/ef1,pof1
      common /tpara2/ef2,pof2
      common /spara1/ef11,pos11
      common /spara2/rhos1
      common /cchek/ifcont(npmax)
c
c     i: DEM,  j: FEM
c
      beta=1.0d0
      alpha=1.0d0
      if(itt.eq.1) then
         frc=fri
      else
         frc=frw
      end if
      pow =pos11
      ew  =ef11
c
      ri=rr(i)
      rj=0.0d0
      dis=gap

      wei3=wei(i)
c
      enn=en(i,jk)
      if(enn.le.0.0d0) enn=1.0d0
      pois2 =1.0d0-po*po
      pois21=1.0d0-pow*pow

      exx1=pois2/ev
      exx2=pois21/ew
      requi=ri*rj/(ri+rj)
      if(rj.eq.0.0d0) then
         requi=ri
      end if
      ba1=dsqrt(4.0d0*(exx1+exx2)*requi*enn/pi)
      elx1=dlog(4.0d0*ri/ba1)-0.5d0
      if(rj.eq.0.0d0) then
         elx2=0.0d0
      else
         elx2=dlog(4.0d0*rj/ba1)-0.5d0
      end if
      eknn=pi/(2.0d0*(exx1*elx1+exx2*elx2))
      vnn=beta*dsqrt(4.d0*wei3*eknn)
      vss=so*vnn
      eknn=alpha*eknn
      ekss=eknn*so

      ui=u(i)
      vi=v(i)
      fi=f(i)
      fj=0.0d0
c
c   normal & tangential relative displacement
c
      if(itt.eq.1) then
         us=-(ui-uj)*ac-(vi-vj)*as+(ri*fi+rj*fj)
         un= (ui-uj)*as-(vi-vj)*ac
      end if
      if(itt.eq.2) then
         us= (ui-uj)*ac+(vi-vj)*as+(ri*fi+rj*fj)
         un=-(ui-uj)*as+(vi-vj)*ac
      end if
c      if((i.eq.318).or.(i.eq.316)) then
c         write(*,9003) i,ui-uj,vi-vj
 9003    format('i,dui,dvi=',i5,2f12.7)
c      write(*,9001) un,us,vnn,vss,fi
c      end if
 9001 format('un,us,vnn,vss,fi=',5f12.7)
c
      if(itt.eq.1) ifcont(I)=7
      if(itt.eq.2) ifcont(I)=1

      if(en(i,jk).eq.0.0d0) then
         if(un.ne.0.0d0) us=us*dis/un
         un=gap
      end if
c
c   Total normal force=en; Total tangential force=es
c
      en(i,jk)=-eknn*un+en(i,jk)
      es(i,jk)=-ekss*us+es(i,jk)

      dn=vnn*un/dt
      ds=vss*us/dt
c
c   Tentative
C   Total normal reaction hn; Total tangential reaction hs
c
      hn =en(i,jk)
      hss=es(i,jk)
c
c   Coulomb Friction check
c
cccc      hnab=dabs(hn)
      if(dabs(hss-frc*hn).gt.0.0d0) then
         hss=frc*dsign(hn,hss)
         ds=0.0d0
      end if
c
c   Final
c   Total normal reaction hn; Total tangential reaction hs
c
      hn =hn   +dn
      hs =hss  +ds

      if(itt.eq.1) then
         rxx= hn*as-hs*ac
         rzz=-hn*ac-hs*as
      else
         rxx=-hn*as+hs*ac
         rzz= hn*ac+hs*as
      end if

      xf(i)=xf(i)+rxx
      zf(i)=zf(i)+rzz
      of(i)=of(i)+ri*hs

      if(i.ge.500) then
         write(*,9900) i,xf(i),zf(i)
      end if
      if((i.ge.310).and.(i.le.319)) then
         write(*,9900) i,xf(i),zf(i)
      end if
9900  format('FE-
DE cont at DEM',i4,'  xf=',e12.4,' zf=',e12.4)
      return
      end
C=====================================
      subroutine detach(i,jk,itt)
      implicit real*8 (a-h,o-z)
```

```
      PARAMETER(NPMAX=1000)
      parameter (ni=2000,nc=20000)
      common /elempm/np,ne,np0,ne0,np1,ne1,npd
      common /conm/dt,fri,frw,ev,ew,po,pow,so,g,de,pi
      common /wepr/rr(ni),wei(ni),pmi(ni)
      common /celx/n,idx,idzi,ipz,w,c,ncl(nc),nncl(ni)
      common /posi/x0(ni),z0(ni),qq(ni)
      common /velo/u0(ni),v0(ni),f0(ni)
      common /for1/xf(ni),zf(ni),of(ni)
      common /lforc/en(ni,14),es(ni,14)
      common /demcc/je(ni,14)
      common /dpmm/u(ni+3),v(ni+3),f(ni+3)
      common /tpara1/ef1,pof1
      common /tpara2/ef2,pof2
      common /spara1/ef11,pos11
      common /spara2/rhos1
      common /cchek/ifcont(npmax)

      if(en(i,jk).le.0.0d0) then
         en(i,jk)=0.0d0
         es(i,jk)=0.0d0
         ifcont(I)=0
      end if
      xf(i)=xf(i)
      zf(i)=zf(i)
      of(i)=of(i)
      if(i.ge.500) then
         write(*,9904) i
      end if
      if((i.ge.310).and.(i.le.319)) then
         write(*,9904) i
      end if
 9904 format(' >>> FE-DE cont deleted at DEM',i4)
      return
      end
C======================================
      subroutine actf1(i,j,jk,as,ac,gap)
      implicit real*8 (a-h,o-z)
      parameter (ni=2000,nc=20000)
      common /elempm/np,ne,np0,ne0,np1,ne1,npd
      common /conm/dt,fri,frw,ev,ew,po,pow,so,g,de,pi
      common /wepr/rr(ni),wei(ni),pmi(ni)
      common /celx/n,idx,idzi,ipz,w,c,ncl(nc),nncl(ni)
      common /posi/x0(ni),z0(ni),qq(ni)
      common /velo/u0(ni),v0(ni),f0(ni)
      common /for1/xf(ni),zf(ni),of(ni)
      common /lforc/en(ni,14),es(ni,14)
      common /demcc/je(ni,14)
      common /dpmm/u(ni+3),v(ni+3),f(ni+3)
c       write(*,*) 'n,idx,idzi,po,pow=',n,idx,idzi,po,pow
c
      nn=np+npd
      ri=rr(i)
      beta=1.0d0
      if((j-nn).le.0) then
         rj=rr(j)
         dis=ri+rj-gap
         wei3=2.d0*wei(i)*wei(j)/(wei(i)+wei(j))
      else
         rj=0.0d0
         dis=ri-gap
         wei3=wei(i)
      end if
      enn=en(i,jk)
      if(enn.le.0.0d0) enn=1.0d0
      if((j-nn).le.0) then
         b1=(3.d0/2.d0/ev*ri*rj/(ri+rj)*(1.d0-po*po)
     &           *enn)**(1.d0/3.d0)
         eknn=2.d0/3.d0*b1*ev/(1.d0-po*po)
         ekss=eknn*so
         vnn=beta*dsqrt(4.d0*wei3*eknn)
         vss=so*vnn
      else
         b1=((3.d0/4.d0*ri*((1.d0-po*po)/ev+(1.d0-
pow*pow)/ew))
     &           *enn)**(1.d0/3.d0)
         eknn=4.d0/3.d0*b1*ev*ew/((1.d0-po*po)*ew+(1.d0-
pow*pow)*ev)
         ekss=eknn*so
         vnn=beta*dsqrt(4.d0*wei3*eknn)
         vss=so*vnn
      end if
      if(eknn.ne.0.0d0) then
         ddt=1.0d-1*dsqrt(wei3/eknn)
         if(ddt.lt.dt) then
            write(6,*) 'dt > ddt=',ddt,i,j,jk,eknn,wei(i)
            stop
         end if
      end if
C
      ui=u(i)
      uj=u(j)
      vi=v(i)
      vj=v(j)
      fi=f(i)
      fj=f(j)

      un= (ui-uj)*ac+(vi-vj)*as
      us=-(ui-uj)*as+(vi-vj)*ac+(ri*fi+rj*fj)
c
      if(en(i,jk).eq.0.d0) then
         if(un.ne.0.d0) us=us*dis/un
         un=dis
      end if
c
      en(i,jk)=en(i,jk)+eknn*un
      es(i,jk)=es(i,jk)+ekss*us
c
      dn=vnn*un/dt
      ds=vss*us/dt

      if(en(i,jk).lt.0.0d0) then
         en(i,jk)=0.0d0
         es(i,jk)=0.0d0
         dn=0.0d0
         ds=0.0d0
         je(i,jk)=0
         return
      else if ((j-nn).le.0) then
         frc=fri
      else
         frc=frw
      end if
      hn=en(i,jk)+dn
      hs=es(i,jk)+ds

      if(dabs(hs-frc*hn).gt.0.0d0) then
         hs=frc*dsign(hn,hs)
         ds=0.0d0
      end if

      xf(i)=-hn*ac+hs*as+xf(i)
      zf(i)=-hn*as-hs*ac+zf(i)
      of(i)= of(i)-ri*hs
c      write(*,*) '   * DEM-DEM: Cont Elem at ',i

      if(jk.le.10) then
         xf(j)=hn*ac-hs*as+xf(j)
         zf(j)=hn*as+hs*ac+zf(j)
         of(j)=of(j)-rj*hs
c         if((j.eq.315).or.(j.eq.314)) then
c            write(*,9000) i,j,zf(j)
c         end if
c         if((j.eq.363).or.(j.eq.364)) then
c            write(*,9000) i,j,zf(j)
c         end if
      end if
 9000 format('  * DEM-
DEM: Cont Elem[',i4,'][',i4,'] zf=',e15.7)
      return
      end
C======================================
      subroutine actdfx(i,j,jk,as,ac,gap,rxx,rzz,uj,vj,itt)
      implicit real*8 (a-h,o-z)
      PARAMETER(NPMAX=1000)
      parameter (ni=2000,nc=20000)
      common /elempm/np,ne,np0,ne0,np1,ne1,npd
      common /conm/dt,fri,frw,ev,ew,po,pow,so,g,de,pi
      common /wepr/rr(ni),wei(ni),pmi(ni)
      common /celx/n,idx,idzi,ipz,w,c,ncl(nc),nncl(ni)
      common /posi/x0(ni),z0(ni),qq(ni)
      common /velo/u0(ni),v0(ni),f0(ni)
      common /for1/xf(ni),zf(ni),of(ni)
      common /lforc/en(ni,14),es(ni,14)
      common /demcc/je(ni,14)
      common /dpmm/u(ni+3),v(ni+3),f(ni+3)
      common /tpara1/ef1,pof1
      common /tpara2/ef2,pof2
      common /spara1/ef11,pos11
      common /spara2/rhos1
      common /cchek/ifcont(npmax)
c
c     i: DEM,  j: FEM
c
      beta=1.0d-1
      alpha=1.0d0
      if(itt.eq.1) then
         frc=fri
      else
         frc=frw
      end if
      pow =pos11
      ew  =ef11
c
      ri=rr(i)
      rj=0.0d0
```

```
      dis=gap

      wei3=wei(i)
      enn=en(i,jk)
      if(enn.le.0.0d0) enn=1.0d0
      po2 =1.0d0-po*po
      po21=1.0d0-pow*pow
      b1=((3.d0/4.d0*ri*(po2/ev+po21/ew))
     &     *enn)**(1.d0/3.d0)
      eknn=4.d0/3.d0*b1*ev*ew/(po2*ew+po21*ev)
      eknn=alpha*eknn
      ekss=eknn*so
      vnn=beta*dsqrt(2.d0*wei3*eknn)
      vss=vnn*dsqrt(so)
c
      ui=u(i)
      vi=v(i)
      fi=f(i)
      fj=0.0d0
c
c    normal & tangential relative displacement
c
      if(itt.eq.1) then
         us=-(ui-uj)*ac-(vi-vj)*as+(ri*fi+rj*fj)
         un= (ui-uj)*as-(vi-vj)*ac
      end if
      if(itt.eq.2) then
         us= (ui-uj)*ac+(vi-vj)*as+(ri*fi+rj*fj)
         un=-(ui-uj)*as+(vi-vj)*ac
      end if

      if(itt.eq.1) ifcont(I)=7
      if(itt.eq.2) ifcont(I)=1

      if(en(i,jk).eq.0.0d0) then
         if(un.ne.0.0d0) us=us*dis/un
         un=gap
      end if
c
c    Total normal force=en; Total tangential force=es
c
      en(i,jk)=-eknn*un+en(i,jk)
      es(i,jk)=-ekss*us+es(i,jk)
      dn=vnn*un/dt
      ds=vss*us/dt
c
c    Tentative
C    Total normal reaction hn; Total tangential reaction hs
c
      hn =en(i,jk)
      hss=es(i,jk)

c
c    Coulomb Friction check
c
cccc      hnab=dabs(hn)
      if(dabs(hss-frc*hn).gt.0.0d0) then
         hss=frc*dsign(hn,hss)
         ds=0.0d0
      end if
c
c    Final
c    Total normal reaction hn; Total tangential reaction hs
c
      hn =hn   +dn
      hs =hss  +ds
      if(itt.eq.1) then
         rxx= hn*as-hs*ac
         rzz=-hn*ac-hs*as
      else
         rxx=-hn*as+hs*ac
         rzz= hn*ac+hs*as
      end if
      xf(i)=xf(i)+rxx
      zf(i)=zf(i)+rzz
      of(i)=of(i)+ri*hs

      if(i.ge.500) then
         write(*,9900) i,xf(i),zf(i)
      end if
      if((i.ge.310).and.(i.le.319)) then
         write(*,9900) i,xf(i),zf(i)
      end if
 9900 format('FE-
DE cont at DEM',i4,'  xf=',e12.4,' zf=',e12.4)
      return
      end
```

A.2 Fundamental Subroutine Programs

The following list is a set of subroutines included when compiling the main program. Note that the original program is shown in Ref[21].

```
      SUBROUTINE INVAR(STRESS,SIGM,DSBAR,THETA)
C
C     This routine forms the stress invariants (2-D Mises)
C
      IMPLICIT REAL(8) (A-H,O-Z)
      REAL(8):: STRESS(*)
      SX =STRESS(1)
      SY =STRESS(2)
      TXY=STRESS(3)
      SZ =STRESS(4)
      SIGM=(SX+SY+SZ)/3.0d0
      DSB1=((SX-SY)*(SX-SY)+(SY-SZ)*(SY-SZ)+(SZ-SX)*(SZ-
SX)+
     &       6.0d0*TXY*TXY)/2.0d0

      DSBAR=SQRT(dsb1)
      IF(DSBAR.EQ.0.d0) THEN
         THETA=0.d0
      ELSE
         DX  =(2.0d0*SX-SY-SZ)/3.d0
         DY  =(2.0d0*SY-SZ-SX)/3.d0
         DZ  =(2.0d0*SZ-SX-SY)/3.d0
         XJ3 =DX*DY*DZ-DZ*TXY*TXY
         SINX=-13.5*XJ3/(DSBAR*DSBAR*DSBAR)
         IF(SINX.GT. 1.0d0) SINX= 1.d0
         IF(SINX.LT.-1.0d0) SINX=-1.d0
         THETA=ASIN(SINX)/3.0d0
      END IF
      RETURN
      END
      SUBROUTINE NULVEC(A,IA)
      IMPLICIT REAL(8) (A-H,O-Z)
      REAL(8):: A(*)
C
      IF(MOD(IA,2).EQ.0) THEN
         DO I=1,IA,2
            A(I  )=0.0d0
            A(I+1)=0.0d0
         END DO
      ELSE
         DO I=1,IA
            A(I)=0.0d0
         END DO
      END IF
      RETURN
      END
      SUBROUTINE NULVEI(IXA,IA)
      IMPLICIT REAL(8) (A-H,O-Z)
      integer:: IXA(*)
C
      IF(MOD(IA,2).EQ.0) THEN
         DO I=1,IA,2
            IXA(I  )=0
            IXA(I+1)=0
         END DO
      ELSE
         DO I=1,IA
            IXA(I)=0
         END DO
      END IF
      RETURN
      END
      SUBROUTINE NULL(A,IA,M,N)
C
C       This subroutine nulls a 2-d arry
C
      IMPLICIT REAL(8) (A-H,O-Z)
      REAL(8):: A(IA,*)
      DO I=1,N
         DO J=1,M
            A(J,I)=0.D0
         end do
      end do
      RETURN
      END
      SUBROUTINE NULL2(a,ia,ig)
      IMPLICIT REAL(8) (A-H,O-Z)
```

```
      REAL(8):: a(ia,*)
C
      do i=1,ig
         do j=1,ia
            a(j,i)=0.0d0
         end do
      end do
      RETURN
      END
      SUBROUTINE NULL2I(ia,ian,ig)
C      IMPLICIT REAL*8 (A-H,O-Z)
      integer:: ia(ian,*)
C
      do i=1,ig
         do j=1,ian
            ia(j,i)=0
         end do
      end do
      RETURN
      END
      SUBROUTINE GAUSS(SAMP,ISAMP,NGP)
C
C       This subroutine provides the weights and sam-
pling points
C       for gauss-legendre quadrature
C
      IMPLICIT REAL(8) (A-H,O-Z)
      REAL(8) ::SAMP(ISAMP,*)
      if(ngp==1) goto 1
      if(ngp==2) goto 2

1     SAMP(1,1)=0.D0
      SAMP(1,2)=2.D0
      GO TO 100
2     SAMP(1,1)=-1./DSQRT(3.D0)
      SAMP(2,1)=-SAMP(1,1)
      SAMP(1,2)=1.D0
      SAMP(2,2)=1.D0
100   CONTINUE
      RETURN
      END
      SUBROUTINE GAUSSV(VSAMP,IGP)
C
C       This subroutine provides the weights and sam-
pling points
C       for gauss-legendre quadrature ... Continu-
ous Version
C
      IMPLICIT REAL(8) (A-H,O-Z)
      REAL(8):: VSAMP(IGP,*)
      VSAMP(1,1)=-1.d0/DSQRT(3.D0)
      VSAMP(1,2)=-1.d0/DSQRT(3.D0)
      VSAMP(2,1)= 1.d0/DSQRT(3.D0)
      VSAMP(2,2)=-1.d0/DSQRT(3.D0)
      VSAMP(3,1)= 1.d0/DSQRT(3.D0)
      VSAMP(3,2)= 1.d0/DSQRT(3.D0)
      VSAMP(4,1)=-1.d0/DSQRT(3.D0)
      VSAMP(4,2)= 1.d0/DSQRT(3.D0)
      RETURN
      END
      SUBROUTINE FORMLN(DER2,IDER,SAMP,ISAMP,I,J)
C
C     This subroutine forms the shape functions and their
C     derivatives for 4-noded quadrilateral elements
C
      IMPLICIT REAL(8) (A-H,O-Z)
      REAL(8):: DER2(IDER,*),SAMP(ISAMP,*)
      REAL(8):: FUN(4)
C
C     samp(1,1)=1/sqrt(3), samp(2,1)=-samp(1,1)
C

      ETA= SAMP(I,1)
      XI = SAMP(J,1)
      ETAM  =0.25d0*(1.0-ETA)
      ETAP  =0.25d0*(1.0+ETA)
      XIM   =0.25d0*(1.0-XI )
      XIP   =0.25d0*(1.0+XI )
      FUN(1)=4.0d0*XIM*ETAM
      FUN(2)=4.0d0*XIP*ETAM
      FUN(3)=4.0d0*XIP*ETAP
      FUN(4)=4.0d0*XIM*ETAP
      DER2(1,1)=-ETAM
      DER2(1,2)= ETAM
      DER2(1,3)= ETAP
      DER2(1,4)=-ETAP
      DER2(2,1)=-XIM
      DER2(2,2)=-XIP
      DER2(2,3)= XIP
      DER2(2,4)= XIM
      RETURN
      END
      SUBROUTINE FORMLV(DER2,IDER,VSAMP,ISAMP,IG)
C
C     This subroutine forms the shape functions and their
C     derivatives for 4-noded quadrilateral elements
C
      IMPLICIT REAL(8) (A-H,O-Z)
      REAL(8):: DER2(IDER,*),VSAMP(ISAMP,*)
      REAL(8):: FUN(4)
C
C       4---3                Integration Order
C       |   |--> xi-axis
C       1---2
C
      XI = VSAMP(IG,1)
      ETA= VSAMP(IG,2)
C
      ETAM  =0.25d0*(1.0-ETA)
      ETAP  =0.25d0*(1.0+ETA)
      XIM   =0.25d0*(1.0-XI )
      XIP   =0.25d0*(1.0+XI )
c      FUN(1)=4.0d0*XIM*ETAM
c      FUN(2)=4.0d0*XIP*ETAM
c      FUN(3)=4.0d0*XIP*ETAP
c      FUN(4)=4.0d0*XIM*ETAP
      DER2(1,1)=-ETAM
      DER2(1,2)= ETAM
      DER2(1,3)= ETAP
      DER2(1,4)=-ETAP
      DER2(2,1)=-XIM
      DER2(2,2)=-XIP
      DER2(2,3)= XIP
      DER2(2,4)= XIM
      RETURN
      END
      SUBROUTINE FORML1(DER2,IDER)
C
C     This subroutine forms the shape functions and their
C     derivatives for 4-noded quadrilateral elements
C
      IMPLICIT REAL(8) (A-H,O-Z)
      REAL(8):: DER2(IDER,*)
      REAL(8):: FUN(4)
C
C       4---3                Integration Order
C       |   |--> xi-axis
C       1---2
C
      XI = 0.0d0
      ETA= 0.0d0
C
      ETAM  =0.25d0*(1.0d0-ETA)
      ETAP  =0.25d0*(1.0d0+ETA)
      XIM   =0.25d0*(1.0d0-XI )
      XIP   =0.25d0*(1.0d0+XI )
c      FUN(1)=4.0d0*XIM*ETAM
c      FUN(2)=4.0d0*XIP*ETAM
c      FUN(3)=4.0d0*XIP*ETAP
c      FUN(4)=4.0d0*XIM*ETAP
      DER2(1,1)=-ETAM
      DER2(1,2)= ETAM
      DER2(1,3)= ETAP
      DER2(1,4)=-ETAP
      DER2(2,1)=-XIM
      DER2(2,2)=-XIP
      DER2(2,3)= XIP
      DER2(2,4)= XIM
      RETURN
      END
      SUBROUTINE MATMUX(A,IA,B,IB,C,IC,L,M,N)
      IMPLICIT REAL(8) (A-H,O-Z)
C
C      This subroutine forms the product of two matrices
C
      REAL(8):: A(IA,*),B(IB,*),C(IC,*)
      DO I=1,L
         DO J=1,N
            X=0.0D+0
            DO K=1,M
               X=X+A(I,K)*B(K,J)
            end do
            C(I,J)=X
         end do
      end do

      RETURN
      END
      SUBROUTINE MATRAN(A,IA,B,IB,M,N)
C
C      This subroutine forms the transpose of a matrix
C
      IMPLICIT REAL(8) (A-H,O-Z)
      REAL(8):: A(IA,*),B(IB,*)
      DO I=1,M
         DO J=1,N
```

```
            A(J,I)=B(I,J)
         end do
      end do
      RETURN
      END
      SUBROUTINE MSMULT(A,IA,C,M,N)
C
C      This subroutine multiplies a matrix by a scalar
C
      IMPLICIT REAL(8) (A-H,O-Z)
      REAL(8):: A(IA,*),C
      DO J=1,N
         DO I=1,M
            A(I,J)=A(I,J)*C
         end do
      end do
      RETURN
      END
      SUBROUTINE MATADD(A,IA,B,IB,M,N)
C
C      This subroutine adds two equal sized arrays
c
      IMPLICIT REAL(8) (A-H,O-Z)
      REAL(8):: A(IA,*),B(IB,*)
      DO J=1,N
         DO I=1,M
            A(I,J)=A(I,J)+B(I,J)
         end do
      end do
      RETURN
      END
      SUBROUTINE MOCOUQ(PSI,DSBAR,THETA,DQ1,DQ2,DQ3)
C
C      This subroutine forms the deribatives of a mohr-
coulomb
C      potential function with re-
spect to the three invariants
C      psi in degrees
C
      IMPLICIT REAL(8) (A-H,O-Z)
      PSIR=PSI*4.*DATAN(1.D0)/180.
      SNTH=DSIN(THETA)
      SNPS=DSIN(PSIR)
      SQ3 =DSQRT(3.D0)
      DQ1 =SNPS
      IF(DABS(SNTH).GT..49D0 )THEN
        C1=1.D0
        IF(SNTH.LT.0.D0 )C1=-1.D0
        DQ2=(SQ3*.5d0-C1*SNPS*.5d0/SQ3)*SQ3*.5d0/DSBAR
        DQ3=0.D0
      ELSE
        CSTH =DCOS(THETA)
        CS3TH=DCOS(3.D0*THETA)
        TN3TH=DTAN(3.D0*THETA)
        TNTH=SNTH/CSTH
        DQ2=SQ3*CSTH/DSBAR*((1.+TNTH*TN3TH)+SNPS*(TN3TH-
TNTH)/SQ3)*.5
        DQ3=1.5d0*(SQ3*SNTH+SNPS*CSTH)/(CS3TH*DSBAR*DSBAR)
      ENDIF
      RETURN
      END
      SUBROUTINE MOCOUF(PHI,C,SIGM,DSBAR,THETA,F)
C
C      This subroutinecalcu-
lates the value of the yield function
C      for a mohr-coulomb material (phi in  degrees)
C
      IMPLICIT REAL(8) (A-H,O-Z)
      PHIR=PHI*4.d0*DATAN(1.0D0)/180.d0
      SNPH=DSIN(PHIR)
      CSPH=DCOS(PHIR)
      CSTH=DCOS(THETA)
      SNTH=DSIN(THETA)
      F=SNPH*SIGM+DSBAR*(CSTH/DSQRT(3.D0)-SNTH*SNPH/3.d0)-
C*CSPH
      RETURN
      END
      SUBROUTINE MVMULT(AM,IM,V,K,L,Y)
C
C      This subroutine multiplies a matrix by a vector
C
      IMPLICIT REAL(8) (A-H,O-Z)
      REAL(8):: AM(IM,*),V(*),Y(*)
      DO I=1,K
         X=0.D0
         DO J=1,L
            X=X+AM(I,J)*V(J)
         end do
         Y(I)=X
      end do
      RETURN
      END
      SUBROUTINE VECADD(A,B,C,N)
C
C      This subroutine adds vectors  a+b=c
C
      IMPLICIT REAL(8) (A-H,O-Z)
      REAL(8):: A(*),B(*),C(*)
      if(mod(n,2).eq.0) then
         DO I=1,N,2
            C(I   )=A(I   )+B(I   )
            C(I+1)=A(I+1)+B(I+1)
         end do
      else
         DO I=1,N
            C(I)=A(I)+B(I)
         end do
      end if
      RETURN
      END
      SUBROUTINE VECCOP(A,B,N)
C
C      This subroutine copies vector a into vector b
C
      IMPLICIT REAL(8) (A-H,O-Z)
      REAL(8):: A(*),B(*)
      if(mod(n,2).eq.0) then
         DO I=1,N,2
            B(I   )=A(I   )
            B(I+1)=A(I+1)
         end do
      else
         DO I=1,N
            B(I)=A(I)
         end do
      end if
      RETURN
      END
      SUBROUTINE MATCOP(A,B,N,M)
C
C      This subroutine copies vector a into vector b
C
      IMPLICIT REAL(8) (A-H,O-Z)
      REAL(8):: A(M,*),B(M,*)
      DO J=1,N
         DO I=1,M
            B(I,J)=A(I,J)
         end do
      end do
      RETURN
      END
      SUBROUTINE GCOUNT(LG,IE,LNODE,NBC,NEMAX,NPMAX)
      IMPLICIT REAL(8) (A-H,O-Z)
      integer::LNODE(NEMAX,*),NBC(NPMAX,*),LG(*)
      INC=0
      DO I=1,4
         LL=LNODE(IE,I)
c         write(*,*) 'node=',ll
         DO K=1,2
            INC =INC+1
            NOD1=NBC(LL,K)
            IF(NOD1.EQ.0) THEN
               LG(INC)=0
            ELSE
               LG(INC)=NOD1
            END IF
c            write(*,*) 'lg(inc)=',lg(inc)
         end do
      END DO
C
      RETURN
      END
      SUBROUTINE SETBC2(NBC,NPMAX,NBV,NO,IBF,NP,NB,NF)
C
      IMPLICIT REAL(8) (A-H,O-Z)
      INTEGER:: NBC(NPMAX,*),NBV(*),NO(*),IBF(*)
C
C              Calculation for Total Degrees of Freedom
      L=0
      DO I=1,NP
        DO K=1,2
        IF(NBC(I,K).EQ.1) GOTO 500
           L=L+1
           NBC(I,K)=L
           NBV(L)=I
c           write(*,*) 'i,k,nbc(i,k)=',i,k,nbc(i,k)
           DO J=1,NF
              IF(I.EQ.IBF(J)) NO(J)=L
        end do
        GOTO 10
500     NBC(I,K)=0
10    end do
      end do
      NB=L
C      stop
```

```
      RETURN
      END
      SUBROUTINE TWOBY2(YJAC,IJAC,YJAC1,IJAC1,DET)
C
C     INVERSE OF 2 X 2 MATRIX
C
      IMPLICIT REAL(8) (A-H,O-Z)
      REAL(8):: YJAC(IJAC,*),YJAC1(IJAC1,*)
      DET=YJAC(1,1)*YJAC(2,2)-YJAC(1,2)*YJAC(2,1)
      YJAC1(1,1)= YJAC(2,2)
      YJAC1(1,2)=-YJAC(1,2)
      YJAC1(2,1)=-YJAC(2,1)
      YJAC1(2,2)= YJAC(1,1)
      DO K=1,2
         DO L=1,2
            YJAC1(K,L)=YJAC1(K,L)/DET
         end do
      end do
      RETURN
      END
      SUBROUTINE FMDEPS(DEE,IDEE,E,V)
C
C     2D PLAIN STRAIN ELASTIC D-MATRIX GENERATION
C
      IMPLICIT REAL(8) (A-H,O-Z)
      REAL(8):: DEE(IDEE,*)
C
      V1=1.0d0-V
      C=E/((1.0d0+V)*(1.0d0-2.0d0*V))
      DEE(1,1)=V1*C
      DEE(2,2)=V1*C
      DEE(3,3)=0.5d0*C*(1.0d0-2.0d0*V)
      DEE(1,2)=V*C
      DEE(2,1)=V*C
      DEE(1,3)=0.0d0
      DEE(3,1)=0.0d0
      DEE(2,3)=0.0d0
      DEE(3,2)=0.0d0
      RETURN
      END
      SUBROUTINE FORMB(BEE,IBEE,DERIV,IDERIV,NOD)
C
C     This subroutine forms B-Matrix for Plain Strain
C
      IMPLICIT REAL(8) (A-H,O-Z)
      REAL(8):: BEE(IBEE,*),DERIV(IDERIV,*)
      DO M=1,NOD
         K=2*M
         L=K-1
         X=DERIV(1,M)
         BEE(1,L)=X
         BEE(3,K)=X
         Y=DERIV(2,M)
         BEE(2,K)=Y
         BEE(3,L)=Y
      END DO
      RETURN
      END
      SUBROUTINE VMPL(E,V,STRESS,PL)
C
C     This routine forms the plastic matrix for a von-
Mises material
C
      IMPLICIT REAL(8) (A-H,O-Z)
      REAL(8):: STRESS(*),TERM(4),PL(4,*)
      SX =STRESS(1)
      SY =STRESS(2)
      TXY=STRESS(3)
      SZ =STRESS(4)
      DSBA1=((SX-SY)*(SX-SY)+(SY-SZ)*(SY-SZ)+(SZ-SX)*(SZ-
SX)
     &       +6.0d0*TXY*TXY)/2.d0

      DSBAR=SQRT(DSBA1)
      EE1=1.5d0*E/((1.0+V)*DSBAR*DSBAR)
      TERM(1)=(2.0d0*SX-SY-SZ)/3.d0
      TERM(2)=(2.0d0*SY-SZ-SX)/3.d0
      TERM(3)=TXY
      TERM(4)=(2.0d0*SZ-SX-SY)/3.d0
      DO I=1,4
         DO J=1,4
            PL(I,J)=TERM(I)*TERM(J)*EE1
            PL(J,I)=PL(I,J)
         end do
      end do

      RETURN
      END
      SUBROUTINE FKDIAG(KDIAG,IG,IDOF)
      IMPLICIT REAL(8) (A-H,O-Z)
      INTEGER:: KDIAG(*),IG(*)
      DO I=1,IDOF
         IWP1=1
         IF(IG(I).EQ.0) GOTO 1
         DO J=1,IDOF
            IF(IG(J).EQ.0) GOTO 2
            IM=IG(I)-IG(J)+1
            IF(IM.GT.IWP1) IWP1=IM
 2       CONTINUE
         end do
         K=IG(I)
         IF(IWP1.GT.KDIAG(K)) KDIAG(K)=IWP1
 1    CONTINUE
      end do
      RETURN
      END
      SUBROUTINE EC-
MAT(ECM,IECM,TN,ITN,TNT,ITNT,FUN,NOD,NODOF)
C
C     FORM CONSISTENT MASS MATRIX
C
      IMPLICIT REAL(8) (A-H,O-Z)
      REAL(8):: ECM(IECM,*),TN(ITN,*),TNT(ITNT,*),FUN(*)
      IDOF=NOD*NODOF
      DO I=1,IDOF
         DO J=1,NODOF
            TNT(I,J)=0.0D0
            TN(J,I)=TNT(I,J)
         end do
      end do
      DO I=1,NOD
         DO J=1,NODOF
            TNT((I-1)*NODOF+J,J)=FUN(I)
            TN (J,(I-1)*NODOF+J)=FUN(I)
         end do
      end do

      DO I=1,IDOF
         DO J=1,IDOF
            X=0.0D0
            DO K=1,NODOF
               X=X+TNT(I,K)*TN(K,J)
            end do

            ECM(I,J)=X
         end do
      end do

      RETURN
      END
      SUBROUTINE FORMKV(BK,TKM,IKM,IG,N,IDOF)
C
C     STORE IN UPPER TRIANGLE AS A VECTOR BK(N*(IW+1))
C
      IMPLICIT REAL(8) (A-H,O-Z)
      REAL(8):: BK(*),TKM(IKM,*)
      INTEGER:: IG(*)
      DO I=1,IDOF
         IF(IG(I).EQ.0) GOTO 1
         DO J=1,IDOF
            IF(IG(J).EQ.0) GOTO 5
            ICD=IG(J)-IG(I)+1
            IF((ICD-1).gt.0) goto 5
            IVAL=N*(ICD-1)+IG(I)
            BK(IVAL)=BK(IVAL)+TKM(I,J)
 5       CONTINUE
         end do
 1    CONTINUE
      end do
      RETURN
      END
      SUBROUTINE FSPARV(BK,TKM,IKM,IG,KDIAG,IDOF)
C
C     GLOBAL MATRIX STORED IN A VECTOR
C
      IMPLICIT REAL(8) (A-H,O-Z)
      INTEGER::KDIAG(*),IG(*)
      REAL(8)::BK(*),TKM(IKM,*)
      DO I=1,IDOF
         K=IG(I)
         IF(K.EQ.0) GOTO 1
         DO J=1,IDOF
            IF(IG(J).EQ.0) GOTO 2
            IW=K-IG(J)
            IF(IW.LT.0) GOTO 2
            IVAL=KDIAG(K)-IW
            BK(IVAL)=BK(IVAL)+TKM(I,J)
 2       CONTINUE
         end do
 1    CONTINUE
      end do
      RETURN
      END
      SUBROUTINE SPARIN(A,N,KDIAG)
      IMPLICIT REAL(8) (A-H,O-Z)
      REAL(8) ::A(*),X
```

```
      INTEGER ::KDIAG(*)
      A(1)=DSQRT(A(1))
      DO I=2,N
        KI=KDIAG(I  )-I
        L =KDIAG(I-1)-KI+1
        DO J=L,I
          X=A(KI+J)
          KJ=KDIAG(J)-J
          IF(J.EQ.1) GOTO 2
          LBAR=KDIAG(J-1)-KJ+1
          LBAR=MAX0(L,LBAR)
          IF(LBAR.EQ.J) GOTO 2
          M=J-1
          DO K=LBAR,M
            X=X-A(KI+K)*A(KJ+K)
          end do
 2        continue
          A(KI+J)=X/A(KJ+J)
        end do
        A(KI+I)=DSQRT(X)
      END DO
      RETURN
      END
      SUBROUTINE SPABAC(A,B,N,KDIAG)
      IMPLICIT REAL(8) (A-H,O-Z)
      REAL(8) :: A(*),B(*)
      INTEGER :: KDIAG(*)
      B(1)=B(1)/A(1)
      DO I=2,N
        KI=KDIAG(I)-I
        L =KDIAG(I-1)-KI+1
        X=B(I)
        IF(L.EQ.I) GOTO 1
        M=I-1
        DO J=L,M
           X=X-A(KI+J)*B(J)
        end do
 1      continue
        B(I)=X/A(KI+I)
      end do
C
      DO IT=2,N
        I=N+2-IT
        KI=KDIAG(I)-I
        X=B(I)/A(KI+I)
        B(I)=X
        L=KDIAG(I-1)-KI+1
        IF(L.EQ.I) GOTO 3
        M=I-1
        DO K=L,M
          B(K)=B(K)-X*A(KI+K)
        END DO
3     CONTINUE
      end do
      B(1)=B(1)/A(1)
      RETURN
      END
```