
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

1	Introduction	1
1.1	Remarks on Current Project	1
1.2	Grant-in-Aid Data	2
1.2.1	Investigator and institution	2
1.2.2	Annual budgets	2
1.2.3	Publications on this research	2
1.2.4	Key words	3
2	Finite Element Mesh Preparation for Agricultural Tires	5
2.1	Introduction	5
2.2	Mesh Generation for Smooth Tires	5
2.2.1	Data preparation	5
2.2.2	Example of mesh generation	6
2.3	Mesh Generation for Traction Tires	6
2.3.1	Data preparation	6
2.3.2	Example of mesh generation	8
2.4	Concluding Remarks	8
3	Discrete Element Method	13
3.1	Introduction	13
3.2	Principle of DEM	13
3.2.1	Relative displacement between DEM elements	13
3.2.2	Contact mechanics	14
3.2.3	Assembly of contact equations	16
3.2.4	Parameter setup	17
3.3	DEM Application	17
3.4	Concluding Remarks	18

4	Finite Element-Discrete Element Method	19
4.1	Introduction	19
4.2	FE-DE Contact Analysis	20
4.2.1	Finite element contact analysis	20
4.2.2	Discrete element contact analysis	21
4.2.3	Finite element-discrete element contact	21
4.3	Dynamic explicit analysis	23
4.4	Contact Algorithm and Coding	24
4.4.1	Algorithm of FE-DE contact analysis	24
4.4.2	Program flow	24
4.5	Numerical Experiment	26
4.5.1	Problem statement	26
4.5.2	Case A result	26
4.5.3	Case B result	28
4.5.4	Case C result	28
4.6	Result and Discussion	30
4.7	Extension to 2D Tire Rolling Problem	33
4.7.1	Strategy for tire rotation	33
4.7.2	Result of deformation field	33
4.7.3	Result of vertical tire reaction	35
4.8	Possible Extension to 3D Analysis	36
4.9	Concluding Remarks	36
5	Parallel Processing Method	37
5.1	Introduction	37
5.2	MPI and PVM	37
5.3	Analysis of Dynamic FE-DEM Program	38
5.3.1	Current performance	38
5.3.2	Possible parallel strategy for proposed FE-DEM	40
5.4	Concluding Remarks	41
6	Conclusion	43
A	Program List	49
A.1	Main Program	49
A.2	Fundamental Subroutine Programs	58

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 existence of solution is strictly or mathematically guaranteed. 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 existence 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].

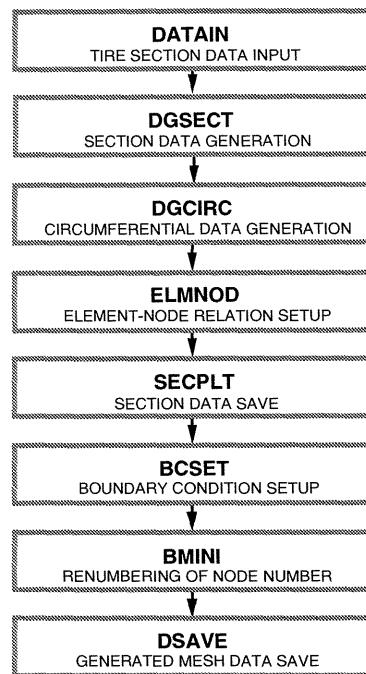


Figure 2-1: Mesh Generation for Smooth Tires

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 existence 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

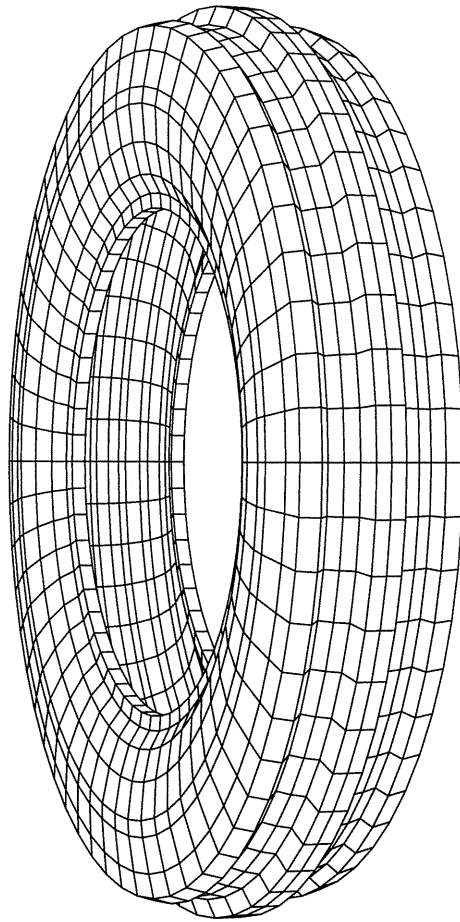


Figure 2-2: FE Mesh for Rib Tire

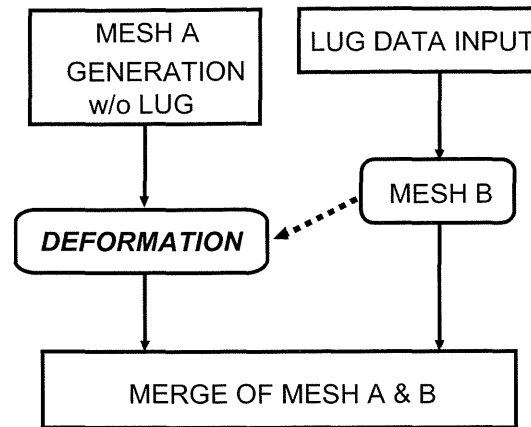


Figure 2-3: Mesh Generation for Traction Tires

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.

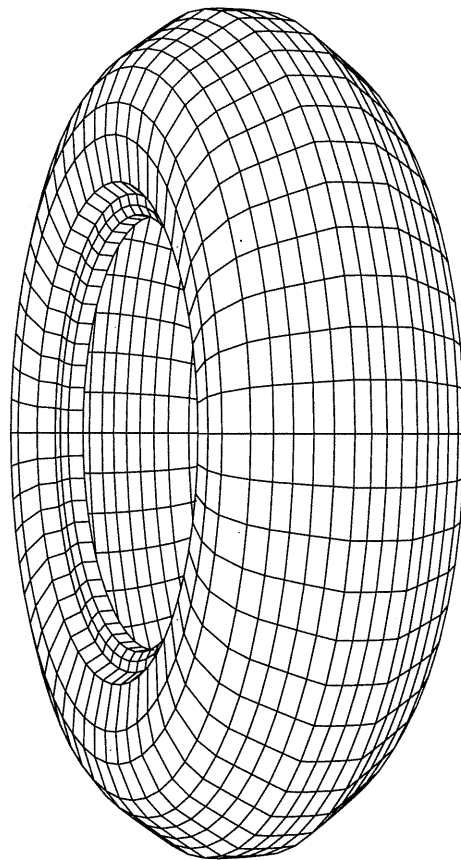


Figure 2-4: Mesh A without Connection Preparation

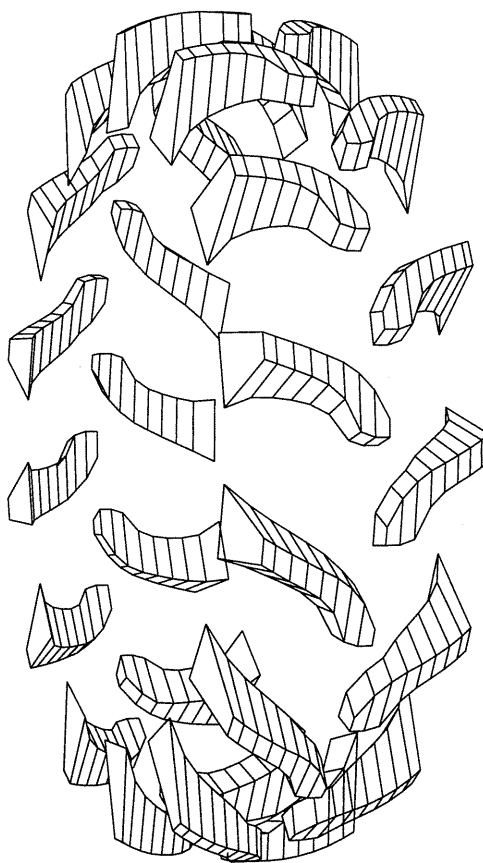


Figure 2-5: Generated Mesh B

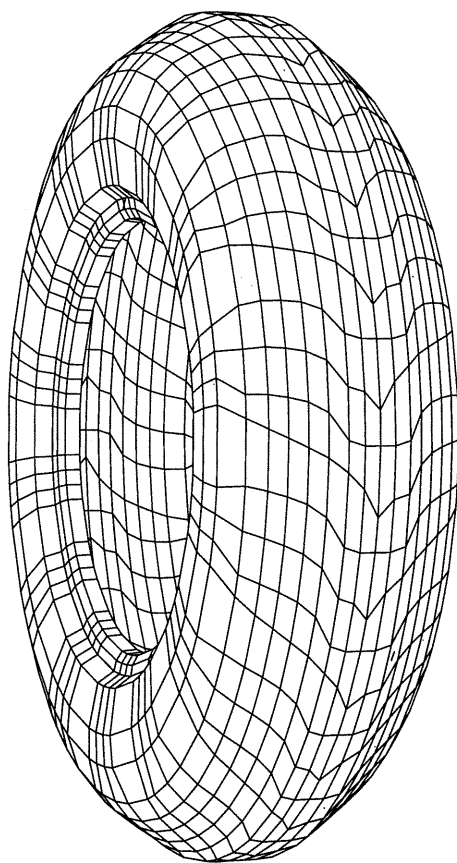


Figure 2-6: Mesh A with Connection Preparation

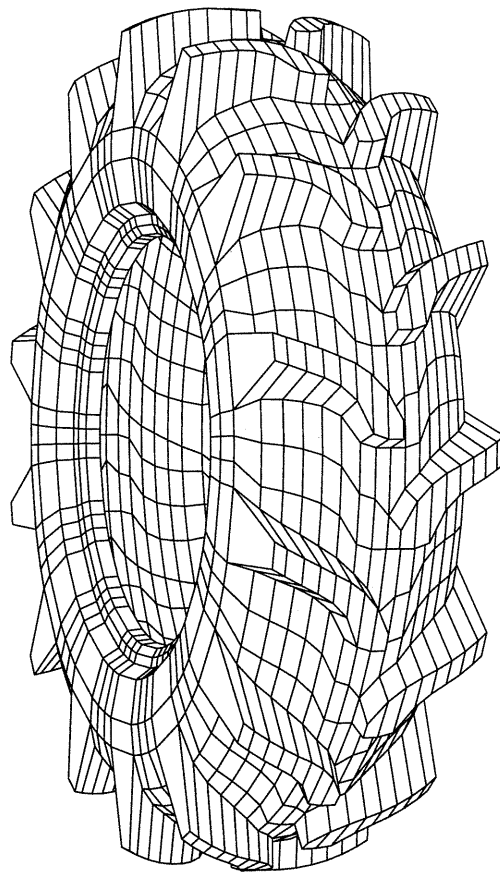


Figure 2-7: FE Mesh for Traction Tire

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 coordinate system of $x-y$ and local coordinate 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

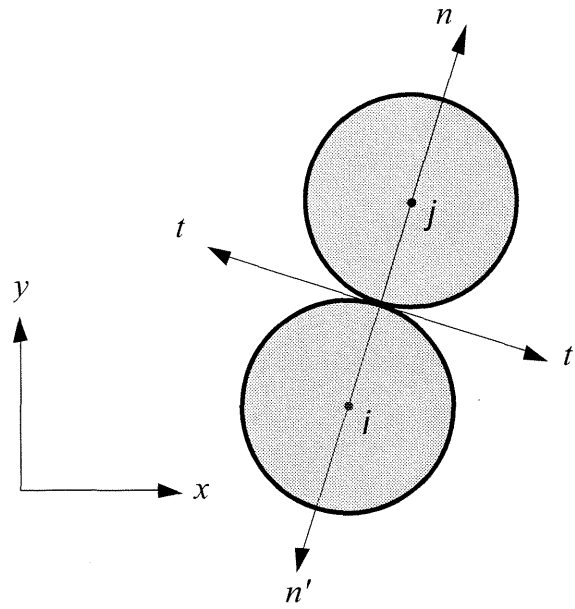


Figure 3-1: DEM-DEM Local Axis

local coordinate system as;

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

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

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;

$$\Delta f_n = K_n \Delta u_n \quad (3.3)$$

$$\Delta d_n = C_n \Delta \dot{u}_n \quad (3.4)$$

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

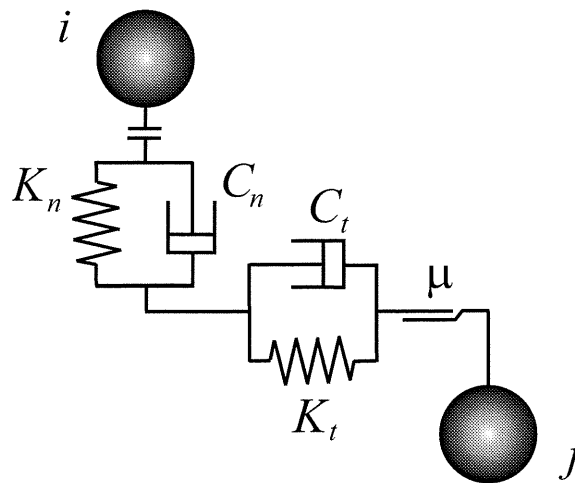


Figure 3-2: DEM model

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

$${}^t f_n = {}^{t-1} f_n + \Delta f_n \quad (3.5)$$

$${}^t d_n = \Delta d_n \quad (3.6)$$

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

$${}^t F_n = {}^t f_n + {}^t d_n \quad (3.7)$$

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;

$$\Delta f_t = K_t \Delta u_t \quad (3.8)$$

$$\Delta d_t = C_t \Delta \dot{u}_t \quad (3.9)$$

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);

$${}^t f_t = {}^{t-1} f_t + \Delta f_t \quad (3.10)$$

$${}^t d_t = \Delta d_t \quad (3.11)$$

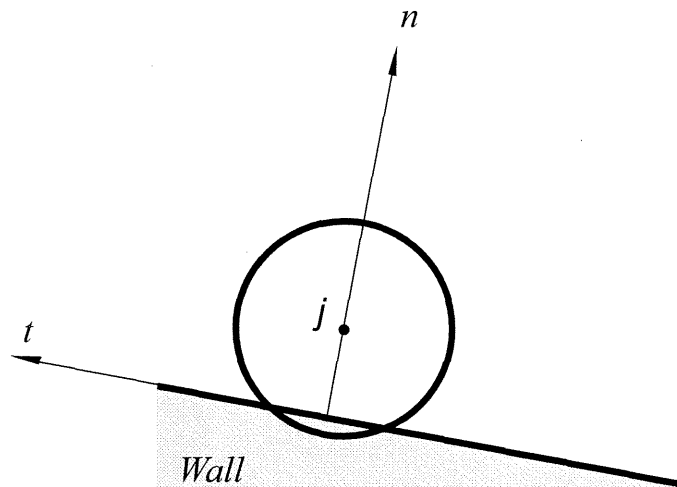


Figure 3-3: DEM-Wall Contact

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

$${}^t f_t = {}^t d_t = 0 \quad ({}^t f_n \leq 0) \quad (3.12)$$

$${}^t f_t = \mu {}^t f_n ({}^t f_t / |{}^t f_t|); {}^t d_t = 0 \quad ({}^t f_t \geq \mu {}^t f_n) \quad (3.13)$$

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

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

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;

$${}^t X_i = M_i {}^t a_i^x \quad (3.15)$$

$${}^t Y_i = M_i {}^t a_i^y \quad (3.16)$$

$${}^t N_i = I_i {}^t \dot{\phi}_i \quad (3.17)$$

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 acceleration 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];

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

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

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;

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

3.3 DEM Application

As an example of DEM application, the simulated result of rigid wheel with lug running over lunar soil (regolith) 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

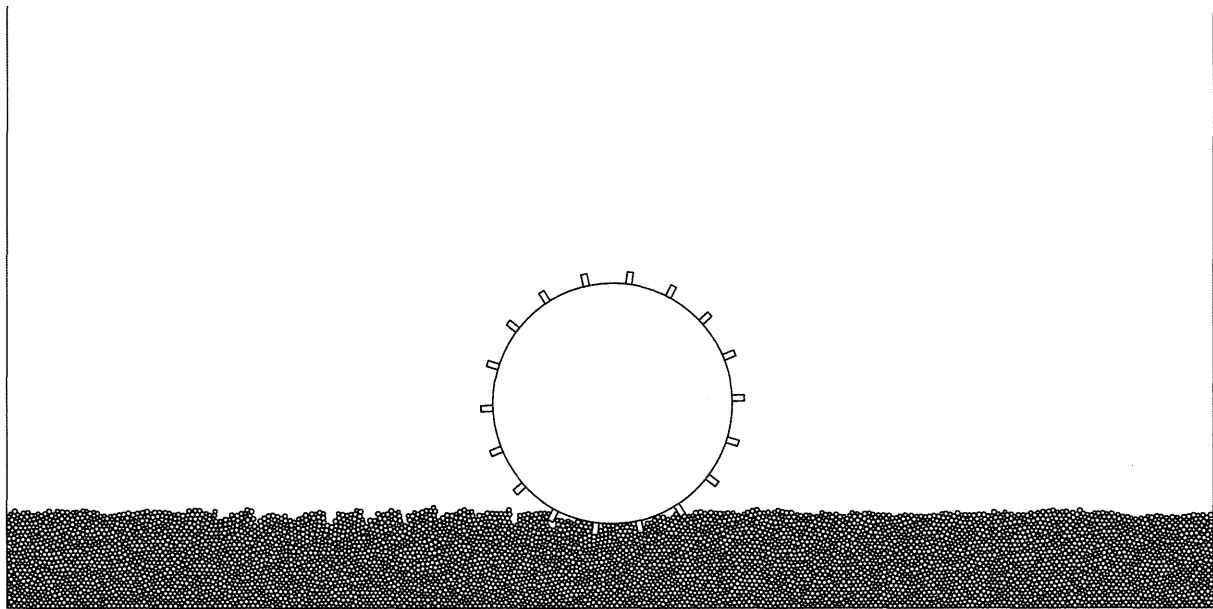


Figure 3-4: Example of DEM application

used. The total number of DEM elements for lunar soil is 6986. It is noted that the clear existence 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 proportion 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. Aube[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.

$$\Delta R_3^n = \alpha \Delta u_n \quad (4.1)$$

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;

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

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;

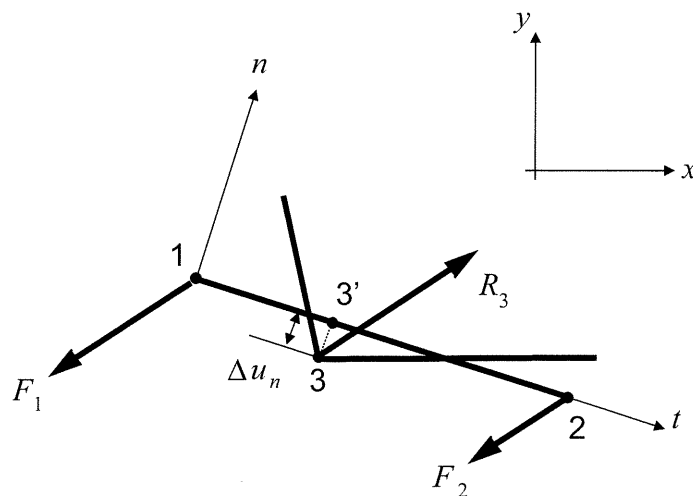


Figure 4-1: FEM Contact

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

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 ${}^t R_3^s$ at a time t , after updating the tangential reaction ${}^t R_3^s = {}^{t-1} R_3^s + \Delta R_3^s$ and comparing it with the corresponding condition of $\mu {}^t R_3^n$ where ${}^t R_3^s = \mu {}^t R_3^n$ if ${}^t R_3^s > \mu {}^t R_3^n$ and ${}^t R_3^n = {}^{t-1} R_3^n + \Delta R_3^n$.

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

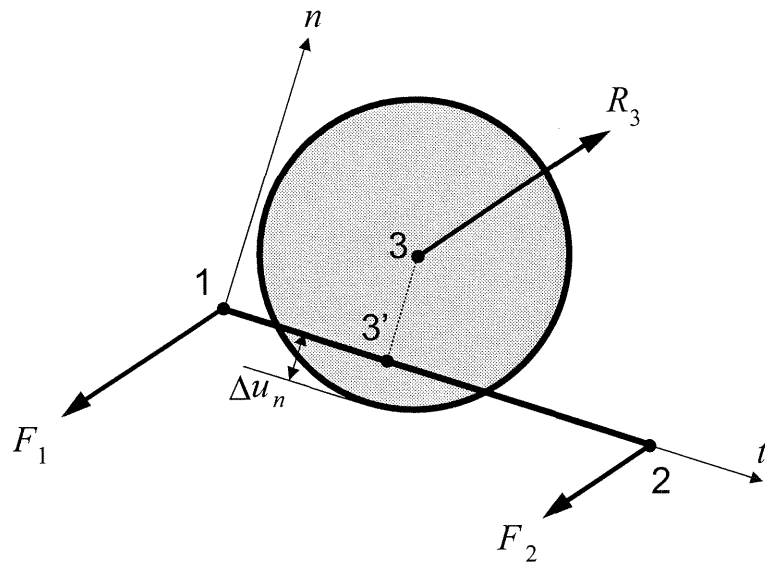


Figure 4-2: FEM-DEM Contact

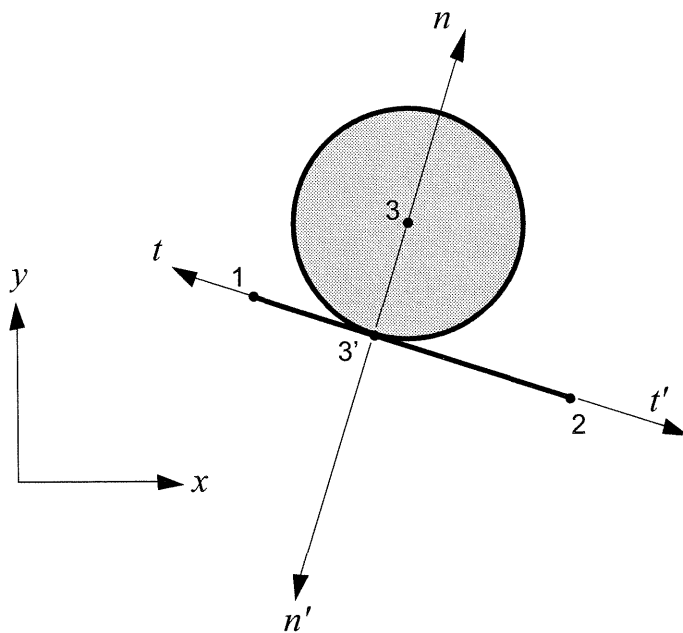


Figure 4-3: FEM-DEM Local Axis

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;

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

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

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 difference 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.

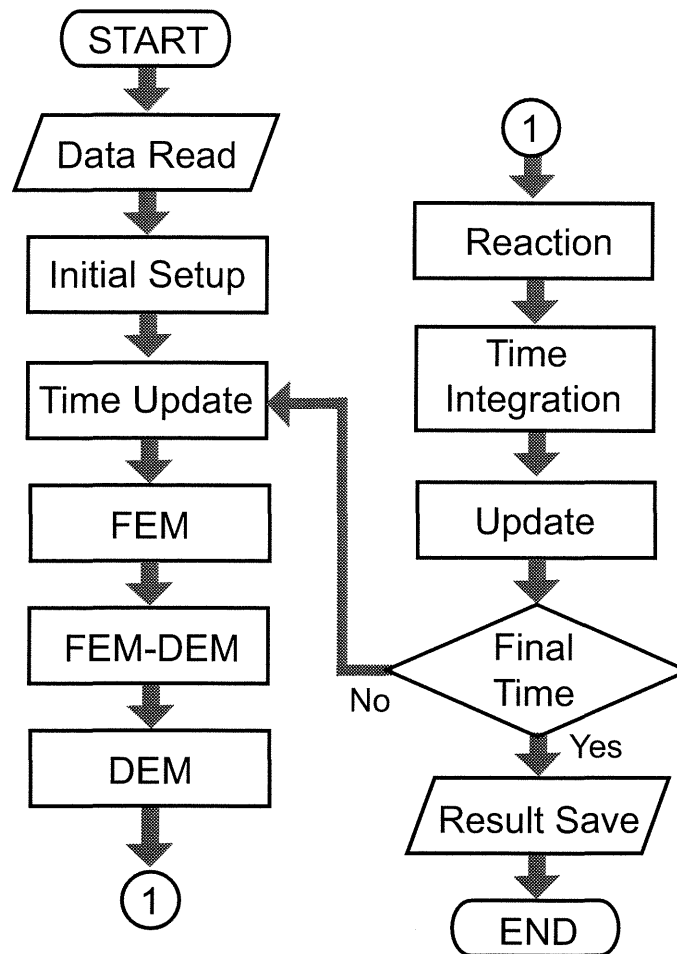


Figure 4-4: Schematic program flow of FE-DE analysis

Brief flow of analysis can be summarized as; acceleration, velocity and current displacement of a 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

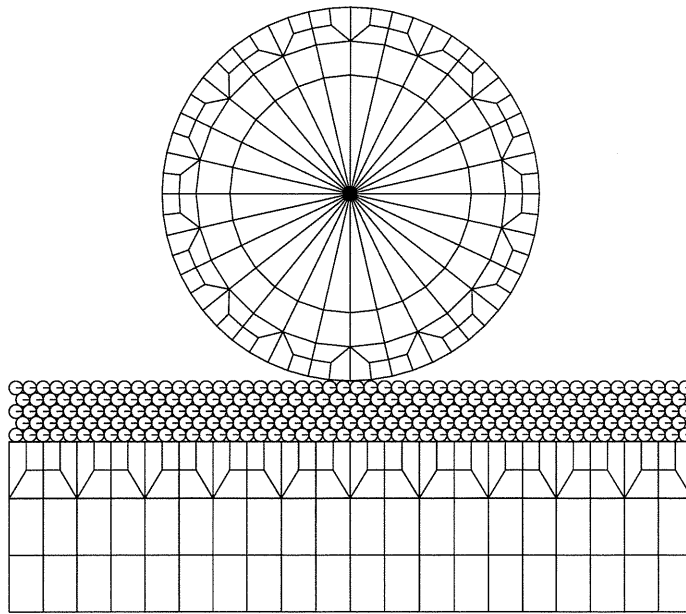


Figure 4-5: Initial Mesh Configuration of Case A

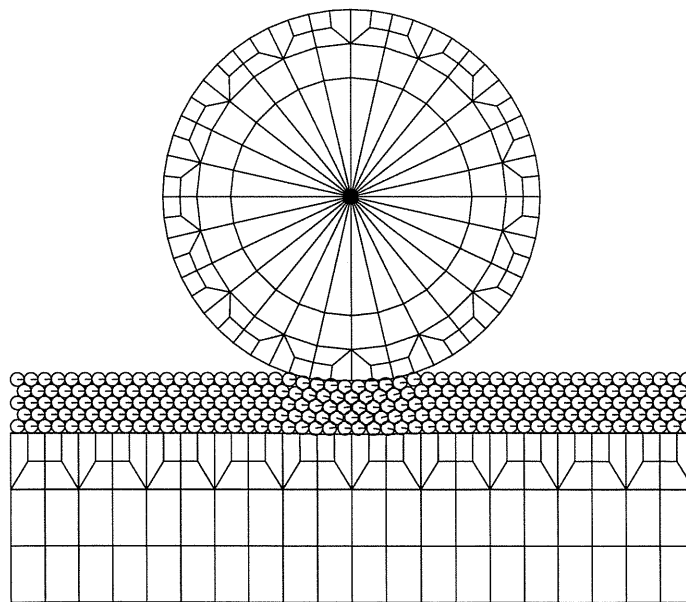


Figure 4-6: Deformation after 8191 time steps

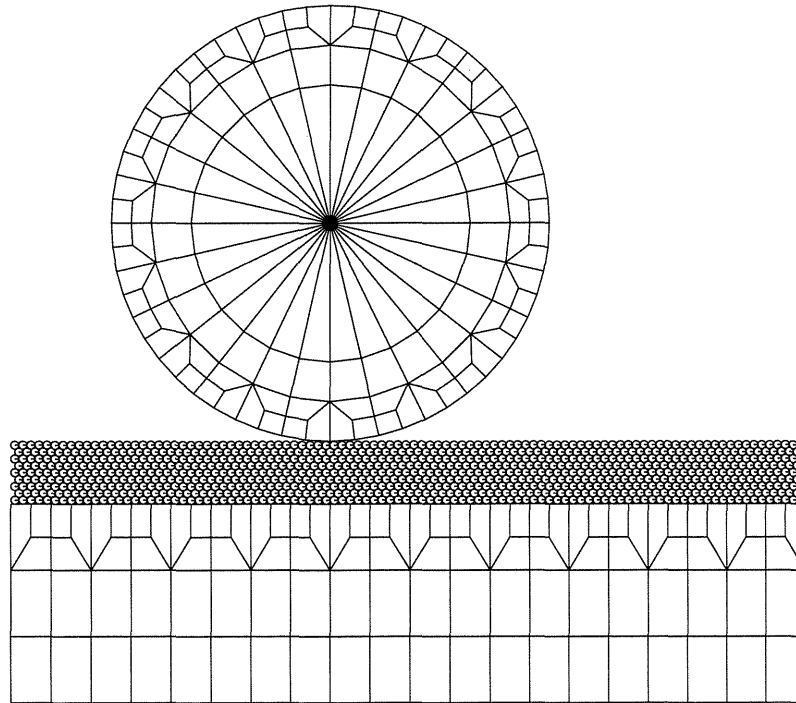


Figure 4-7: Initial Mesh Configuration of Case B

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

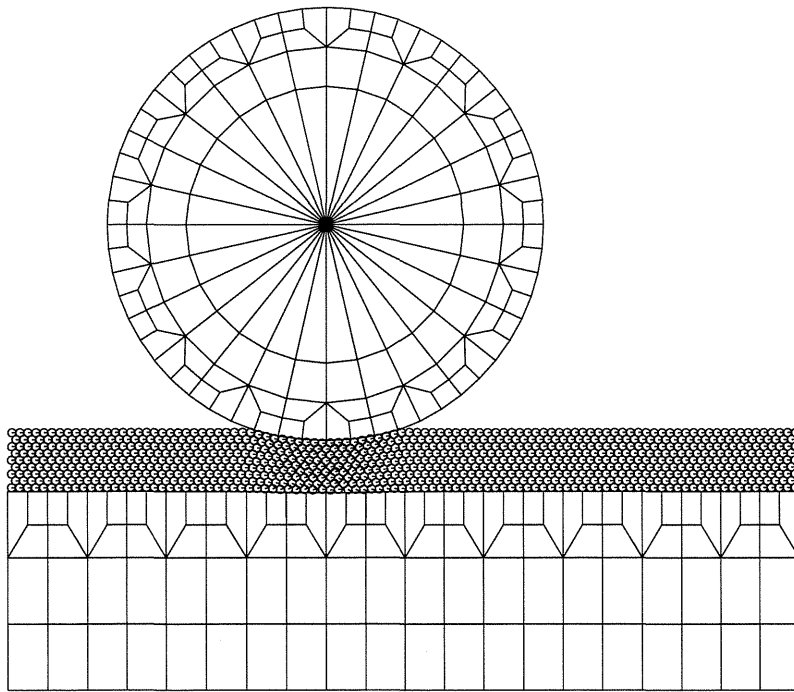


Figure 4-8: Deformation after 8482 time steps

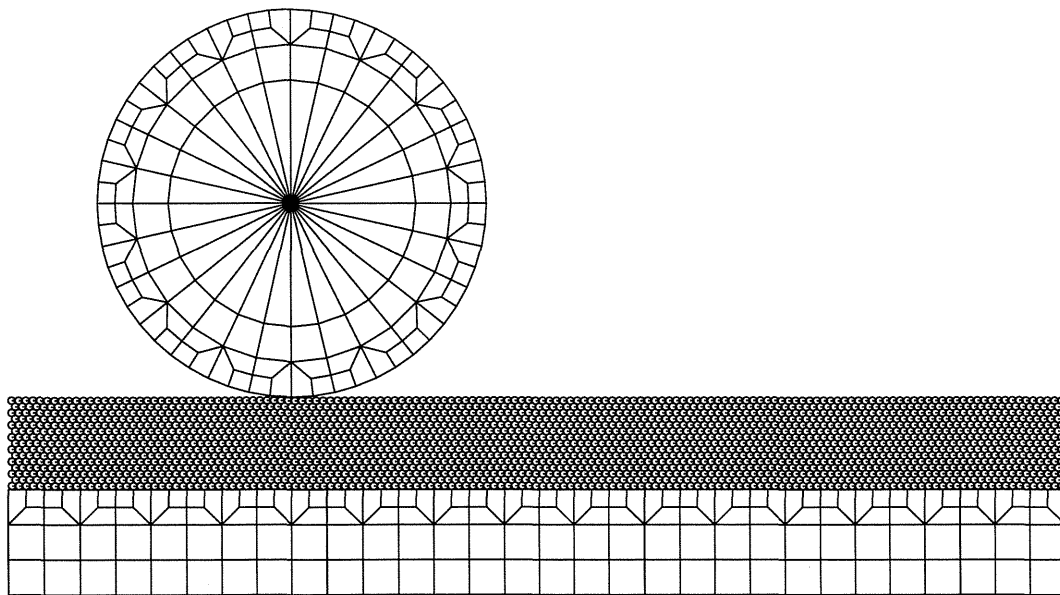


Figure 4-9: Initial Mesh Configuration of Case C

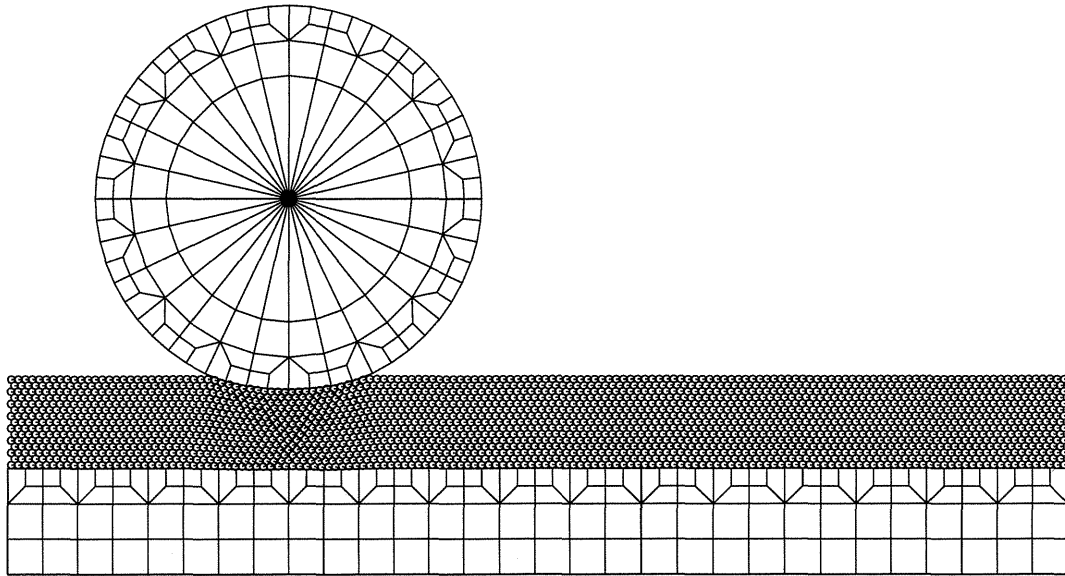


Figure 4-10: Deformation after 9828 time steps

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 further 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: $Y = 0.065677X +$

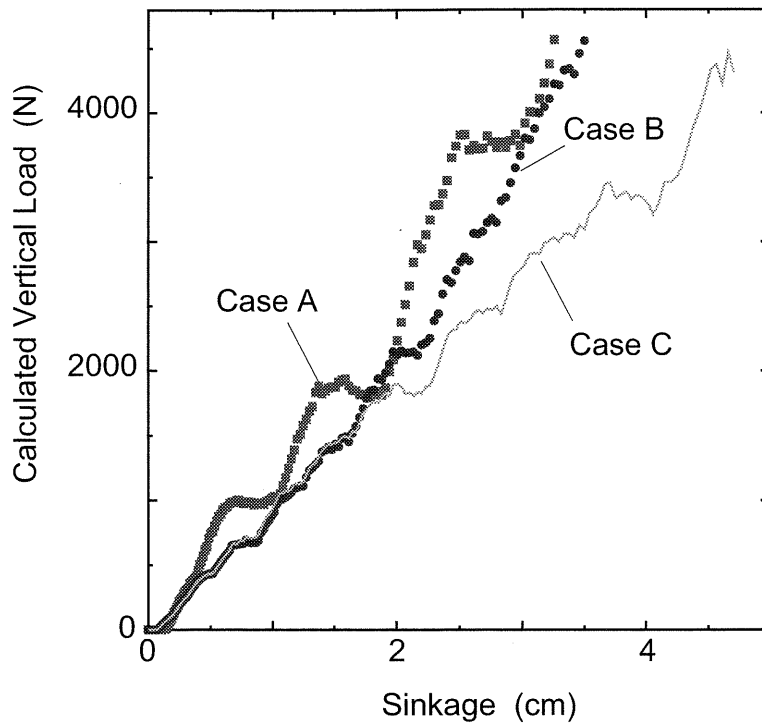


Figure 4-11: Load-sinkage result

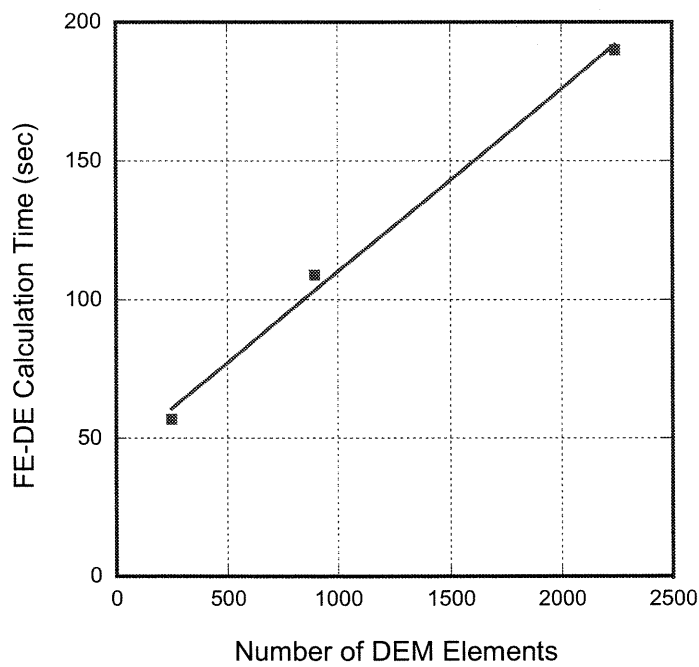


Figure 4-12: Calculation time in terms of DEM elements

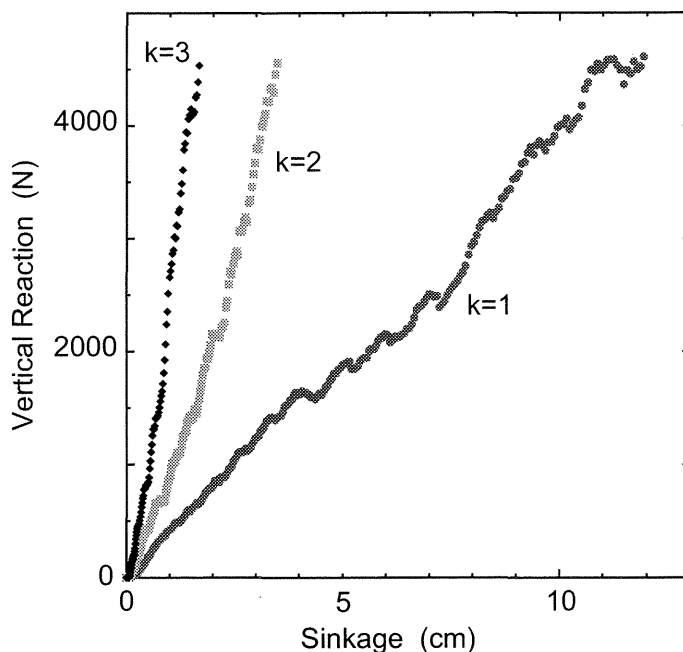


Figure 4-13: Effect of DEM parameter on calculated load

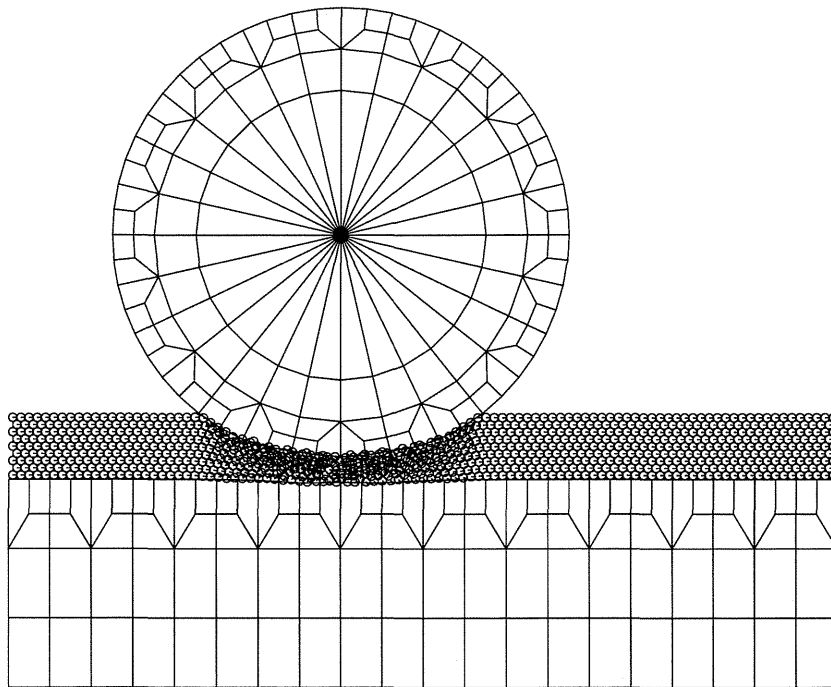
44.518 ($R^2 = 0.99724$) though there exists the increase in FEM element in Case C. Therefore, it is estimated that the larger the total number of DEM elements used, the more total calculation time consumed.

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

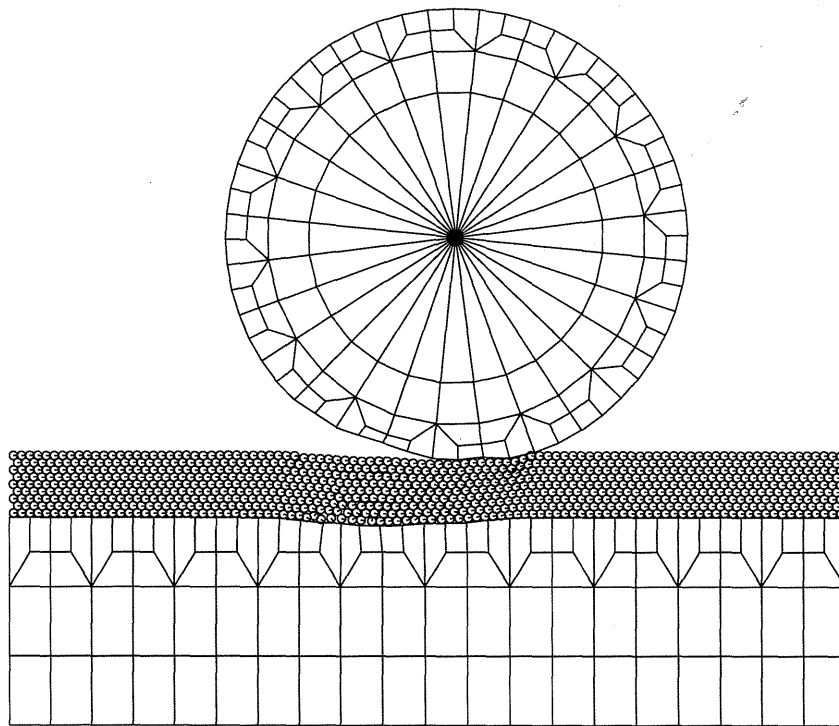


Figure 4-15: Result of tire rotation at 0.1 sec

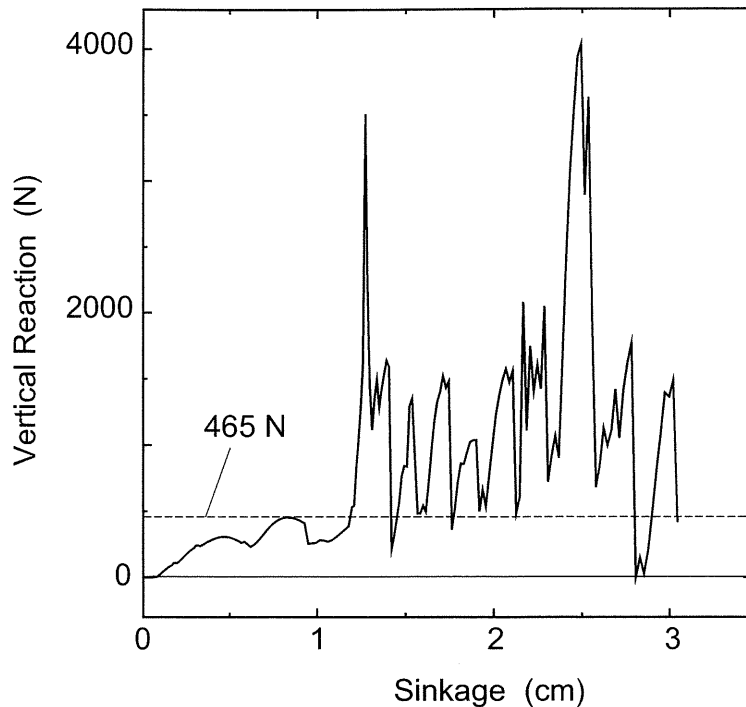


Figure 4-16: Result of total vertical reaction

(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 suffered 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 existence 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 developed 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

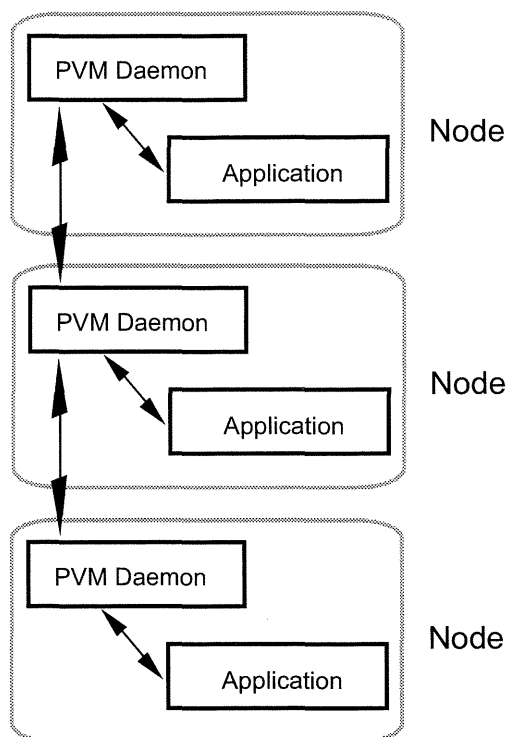


Figure 5-1: PVM model

parallel computers. Fig.5-2 shows the idea of MPI model. In MPI environment, the allotted 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 summarized 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.

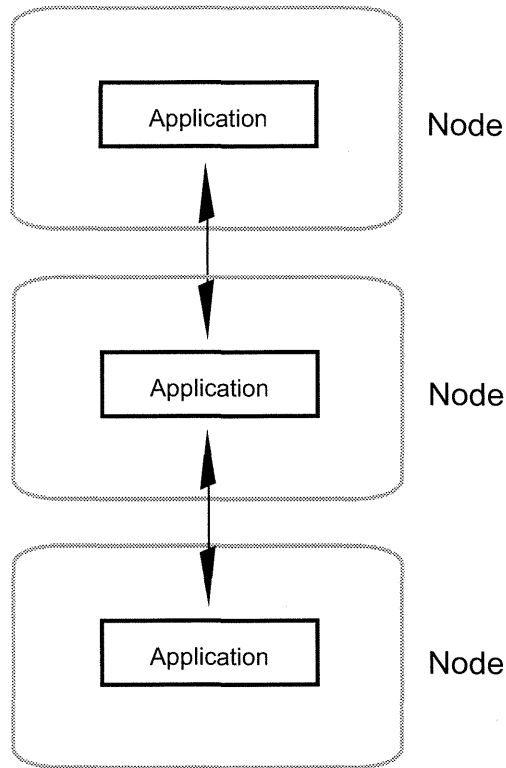


Figure 5-2: MPI model

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	pctx_
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_

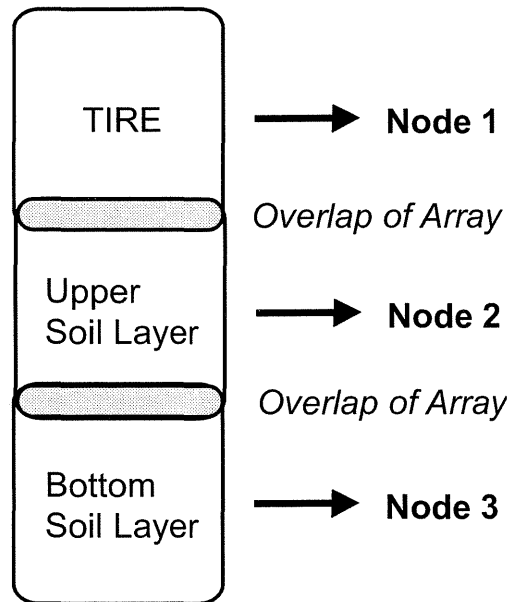


Figure 5-3: Schematic strategy for FE-DEM program

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 divided and processed by each CPU in parallel system separately.
- Contact check and reaction calculation part is divided 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 sub-routines 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 divide 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 sub-routines 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 divide 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

```
1 C-----
2 C
3 C Program: fdem99.f
4 C
5 C 2D Elasto-plastic & Dynamic FEM+DEM Analysis Code
6 C Explicit Time Integration & Lumped Mass
7 C
8 C by Hiroshi NAKASHIMA
9 C originally for VT-Alpha300/433AXP on Linux/Alpha
10 C
11 C Master Program: fdem99.f
12 C Library Program: exsub.f
13 C
14 C Compile Option:
15 C (Intel): ifc -O2 fdem99.f exsub.f -lm -o fdem99
16 C
17 C Ver. 3.1 May 8, 2003
18 C-----
19 IMPLICIT REAL(8) (A-H,O-Z)
20 C
21 PARAMETER (NPMAX=1000,NEMAX=1000,INO=700,NFMAX=500)
22 parameter (ngmax=npmax*2)
23 parameter (ni=2000,nc=20000)
24 PARAMETER (MAXBN=100)
25 C PARAMETER (NGX=250,NGY=400)
26 C
27 CHARACTER (20) FILE1
28 common /elem/np,ne,np0,ne0,np1,ne1,npd
29 common /convu/r2d,d2r
30 common /demscl/zmax,wmax,zmin
31 common /com/ dt,fri,frw,ev,ew,po,pow,so,g,de,pi
32 common /posi/ x0(ni),z0(ni),qq(ni)
33 C common /posp/xp(ni),zp(ni),qp(ni)
34 common /velo/ u0(ni),v0(ni),f0(ni)
35 common /forl/ xf(ni),zf(ni),of(ni)
36 common /lforc/ en(ni,14),es(ni,14)
37 common /wepr/ rr(ni),wei(ni),pmi(ni)
38 common /celx/ n,idx,idzi,ipz,w,c,ncl(nc),nncl(ni)
39 common /accel/ du0(ni),dv0(ni),df0(ni)
40 common /demcc/ je(ni,14)
41 common /dpmm/ u(ni+3),v(ni+3),f(ni+3)
42 common /tpara1/ef1,pof1
43 common /tpara2/ef2,pof2
44 common /spara1/ef11,pos11
45 common /presdd/dispres
46 common /tpara3/rhof1,rhof2
47 common /spara2/rhos1
48 common /femer/ lnode(NEMAX,5)
49 common /sscal/ zfmin
50 common /itest/ izin,ixin
51 common /cchek/ ifcont(npmax)
52 common /nods/ isnode(npmax)
53 C common /bcstat/nbc(npmax,2)
54 C
55 REAL*8 DEE(3,3),COORD(4,2),VSAMP(4,2)
56 REAL*8 XJAC(2,2),XJAC1(2,2)
57 REAL*8 DER(2,4),DERIV(2,4),BEE(3,8),ELD(8)
58 REAL*8 EPS(4),SIGMA(4),BT(8,3),ELOAD(8),BLOAD(8)
59 REAL*8 SX(NEMAX,4),SY(NEMAX,4),TXY(NEMAX,4),SZ(NEMAX,4)
60 REAL*8 EX(NEMAX,4),EY(NEMAX,4),GXY(NEMAX,4),EZ(NEMAX,4)
61 real*8 stress(4),btbd(8,8),dbee(3,8),ekm(8,8)
62 C real*8 fun(8)
63 real*8 alax(ngmax,ngmax),blax(ngmax),vw(ngmax)
64 real*8 wload(ngmax),sinks(ngmax)
65 DIMENSION isp(2,NEMAX),itype(NEMAX),knf(nemax,8)
66 DIMENSION LG(8)
67 DIMENSION IDEM(NPMAX)
68 DIMENSION IBC(MAXBN),nbc(npmax,2)
69 C intrinsic dtme
70 real tarray(2),tresult
71 C
72 DATA IDEE,IBEE,idee,IH/4*3/
73 DATA IJAC,IJAC1,IDER,IDERIV,IT/5*2/
74 DATA ICOORD,NOD/2*4/,IBT,IDOF,ibtbd,ikm/4*8/,NGP/2/
75 C call etime(tarray,result)
76 PI =4.0d0*ATAN(1.0d0)
77 D2R=PI/180.0d0
78 R2D=180.0d0/PI
79 G =9.80665d2
80 write(*,*) ' FDEM step number='
81 read(*,*) istep
82 C
83 igauss=ngp*ngp
84 do j=1,igauss
85 do i=1,nemax
86 sx(i,j)=0.0d0
87 sy(i,j)=0.0d0
88 txy(i,j)=0.0d0
89 sz(i,j)=0.0d0
90 ex(i,j)=0.0d0
91 ey(i,j)=0.0d0
92 gxy(i,j)=0.0d0
93 ez(i,j)=0.0d0
94 end do
95 end do
96 C
97 C Initial value for DEM
98 C
99 do i=1,npmax
100 nbc(i,1)=0
101 nbc(i,2)=0
102 end do
103 C IDEM: element index of FEM or DEM
104 C
105 CALL NULVEI (IDEM,NEMAX)
106 C-----
107 C FEM Data Load
108 C-----
109 call ALLDAT(NBC,NB,IBC)
110 C
111 do i=1,NPMAX
112 u(i) =0.0d0
113 v(i) =0.0d0
114 f(i) =0.0d0
115 u0(i) =0.0d0
116 v0(i) =0.0d0
117 f0(i) =0.0d0
118 du0(i) =0.0d0
119 dv0(i) =0.0d0
120 df0(i) =0.0d0
121 end do
122 C
123 C KNF: DOF indicator for Node Num in an element
124 C
125 do i=1,ne
126 jq=0
127 do j=1,4
128 jq=jq+1
129 k=2*lnode(i,j)-1
```

```

130         knf(i,jq)=k
131         jq=jq+1
132         k=2*lnode(i,j)
133         knf(i,jq)=k
134     end do
135 end do
136 C
137 C NBC: location of free movement in serial number
138 C
139 C CALL SETBC2(NBC,NPMAX,NBV,NO,IBF,NF,NB,NF)
140 CALL SURFFE(isp,ittype,nsf)
141 C=====
142 C DEM Data Load
143 C=====
144 NUMELM=npd
145 N=npd
146 call DEDATA
147 C
148 do i=NE+1,NUMELM
149 do j=1,4
150 en(i,j)=0.0d0
151 es(i,j)=0.0d0
152 je(i,j)=0
153 end do
154 end do
155 C
156 C Index for FEM(idem(*)=0) or DEM(idem(*)=element #)
157 C
158 NEDF=NE+NUMELM
159 NPDPF=NP+NUMELM
160 c write(*,*) 'NP,NE,NPDF,NEDF',np,ne,npdf,nedf
161 c stop
162 do I=1,NUMELM
163 IDEM(NE+I)=NP+I
164 end do
165 call GAUSSV(VSAMP,igaus)
166 C=====
167 C Explicit Integration Loop
168 C=====
169 TI=0.0d0
170 C
171 zini=z0(155)
172 k4=0
173 C
174 DO 10 JJ=1,ISTEP
175 TI=TI+DT
176 write(*,*)
177 write(*,9103) jj,TI
178 9103 format('STEP',i6,'') TIME='e12.3','[sec]')
179 C
180 C Time increment: dt
181 C Current Time: ti
182 C
183 C ** Initialize of Force Vector for each time step
184 C
185 do i=1,NPDF
186 xf(i)=0.0d0
187 zf(i)=0.0d0
188 of(i)=0.0d0
189 wei(i)=0.0d0
190 pmi(i)=0.0d0
191 end do
192 C
193 C ** Setup of DEM grid
194 C
195 call fposir(rmax)
196 call inmat
197 call ncel
198 C-----
199 C Elemental Stress-Strain Relationship
200 C-----
201 if(jj.eq.1) allw=0.0d0
202 C*****NE: All FE Elem Num
203 DO 30 IE=1,NE
204 C*****
205 C ** For FEM
206 C
207 ste =0.0d0
208 AREA=0.0d0
209 call nulvec(BLOAD,idof)
210 call nulvec(eps,4)
211 CALL GCOUNT(LG,IE,LNODE,NBC,NEMAX,NPMAX)
212 DO I=1,4
213 ILL=LNODE(IE,I)
214 COORD(I,1)=x0(ILL)
215 COORD(I,2)=z0(ILL)
216 if(nbc(ill,1).ne.1) then
217 ELD(2*I-1)=u(ILL)
218 else
219 eld(2*I-1)=0.0d0
220 end if
221 if(nbc(ill,2).ne.1) then
222 ELD(2*I )=v(ILL)
223
224 else
225 eld(2*I )=0.0d0
226 end if
227 END DO
228 CALL NULL(DEE,IDE,IE,IH,IH)
229 C
230 C TIRE: 1--NE0; SOIL: NE0+1--
231 C
232 if(IE.gt.NE0) then
233 EE1=ef11
234 VV1=pos11
235 rho=rhos1
236 end if
237 if(IE.le.NE0) then
238 if(lnode(ie,5).le.10) then
239 EE1=ef1
240 VV1=pof1
241 rho=rhof1
242 else
243 EE1=ef2
244 VV1=pof2
245 rho=rhof2
246 end if
247 end if
248 CALL FMDEPS(DEE,IDE,EE1,VV1)
249 call null(ekm,idof,idof,idof)
250 c
251 c Gauss Integration Loop
252 C-----
253 DO 50 IG=1,ngp*ngp
254 write(*,*) 'ie,ig=',ie,ig
255 CALL FORMLV(DER,IDER,VSAMP,4,IG)
256 CALL MAT-
257 MUX(DER,IDER,COORD,ICOORD,XJAC,IJAC,
258 & IT,NOD,IT)
259 CALL TWOBY2(XJAC,IJAC,XJAC1,IJAC1,DET)
260 CALL MAT-
261 MUX(XJAC1,IJAC1,DER,IDER,DERIV,IDERIV,
262 & IT,IT,NOD)
263 CALL NULL(BEE,IBEE,IH,IDOF)
264 CALL FORMB(BEE,IBEE,DERIV,IDERIV,NOD)
265 QUOT=DET
266 if(det.le.0.0d0) then
267 write(*,*) 'FEM ER-
268 ROR in Elem ',ie,'* DET=',det
269 stop
270 end if
271 c
272 c area = pho
273 AREA=AREA+QUOT*RHO
274 C
275 c sigma = D epsilon
276 CALL MVMULT(DEE,IDE,EPS,IH,IH,SIGMA)
277
278 CALL MATRAN(BT,IBT,BEE,IBEE,IH,IDOF)
279 CALL MVMULT(BT,IBT,SIGMA,IDOF,IH,ELOAD)
280
281 BLOAD = int BT*SIGMA dV
282 DO K=1,idof
283 BLOAD(K)=BLOAD(K)+ELOAD(K)*QUOT
284 END DO
285 C-----
286 C End of Gauss Integration Loop
287 C-----
288 50 CONTINUE
289 C
290 c global mass & inertia term for FEM
291 C
292 do k=1,4
293 ill=lnode(ie,k)
294 wei(ill)=wei(ill)+area*0.25d0
295 pmi(ill)=1.0d0
296 end do
297 C
298 c Weight of Tire Calculation
299 C
300 if((jj.eq.1).and.(ie.le.ne0)) allw=allw+area*g/100.0d0
301 C
302 c subtract BT*sigma from FORCE vector
303 C
304 J1=0
305 DO J=1,4
306 LI=LNODE(ie,J)
307 DO K=1,2
308 J1=J1+1
309 if(k.eq.1) then
310 xf(li)=xf(li)-bload(j1)
311 else
312 zf(li)=zf(li)-bload(j1)
313

```

```

313           end if
314           END DO
315           END DO
316 C-----
317 C                               End of FE Element Loop
318 C-----

319 30      CONTINUE
320 C=====
321 C                               Contact of FEM-DEM Check
322 C=====
323 c       write(*,*) 'FE-DE Contact in ',jj
324 c
325       elimi=1.0d-3
326 c       jcont=0
327       do i=1,npdf
328         ifcont(i)=0
329       end do
330 C-----
331 C                               Loop of FEM Surface Element
332 C
333       DO 44 I=1,NSF
334 C                               L1--LNODE(*,4);L2--
335 LNODE(*,3)
336 C                               L3--Location of Surface
337 C                               L3=1 for tire; L3=2 for soil
338       L3 =itype(I)
339       L1 =ISP(1,I)
340       L2 =ISP(2,I)
341       XL1 =X0(L1)
342       XL2 =X0(L2)
343       YL1 =Z0(L1)
344       YL2 =Z0(L2)
345
346       delx=XL2-XL1
347       dely=YL2-YL1
348 C
349 C                               ZLO: Boundary element length
350 C
351       ZLO=dsqrt(delx*delx+dely*dely)
352 C-----
353 C                               --- DEM ELEMENT LOOP
354 C-----
355       do 46 L=NE+1,NEDF
356 C                               Select surface DEM
357 C
358       if(lnode(L,5).eq.0) goto 46
359       if((L3.eq.1).and.(lnode(L,5).eq.100)) goto 46
360       if((L3.eq.2).and.(lnode(L,5).eq.99)) goto 46
361 C
362 c                               lk: DEM Elem number
363 c
364 c                               lk=idem(L)
365 c                               if(ifcont(lk).ne.0) goto 46
366 c
367 c                               Coord of DEM elem
368 c
369 c                               UDEX=X0(LK)
370 c                               UDEZ=Z0(LK)
371 c                               if(UDEX.GT.(XL2+0.001)) goto 46
372 c                               if(UDEX.LT.(XL1-0.001)) goto 46
373 C
374 C                               contact check based on up-
375 C                               dated coordinate
376 C
377 c                               CALL LNORML
378 c                               (UDEX,UDEZ,UPXL,UPZL,sinl,cosl,XL1,YL1,XL2,YL2,L3)
379 C
380 C                               GZAI=-1.0d0+2.0d0*UPXL/ZLO
381 C
382 C                               If DEM elem is out of Boundary Line Element:
383 C
384 C                               IF(gzai.Lt.-1.001d0) GOTO 46
385 C                               IF(gzai.Gt. 1.001d0) GOTO 46
386 C
387 C                               RR(j) = Radius of DEM ; REGI = Length of nor-
388 C                               mal foot
389 C
390 c                               gap=UPZL-RR(LK)
391 C
392 c                               if(gap.gt.0.0d0) then
393 c                               if(ifcont(lk).ne.0) then
394 c                               ifcont(lk)=0
395 c                               call detach(LK,14,L3)
396 c                               end if
397 c                               goto 46
398 c                               end if
399 C
400 c                               if(lk.ge.500) then
401 c                               write(*,9008) LK,gap
402 c                               end if
403 c                               format('---
404 contact at ',i4, ' ** gap =',f12.8)
405 C
406 C
407 C
408 C
409 C
410 C
411 C
412 C
413 C
414 C
415 C
416 46      continue
417 C
418 44      continue
419 C
420       tvert=0.0d0
421       do L=NE+1,NEDF
422 C
423 c                               ---> lk: node number
424 c                               lk=idem(L)
425 c                               if(ifcont(lk).eq.7) then
426 c                               write(*,*) 'dem num=',lk
427 c                               tvert=tvert+dabs(zf(lk))
428 c                               end if
429 c                               end do
430 C
431 C                               End of FEM-DEM Calculation
432 C
433 C                               DEM-DEM Calculation
434 C-----
435       do L=NE+1,NEDF
436 c                               iel=idem(L)
437 c                               write(*,*) 'DEM ',iel
438 C
439 c                               call wcont(iel)
440 c                               call pcontx(iel,rmax)
441 c                               end do
442 C
443 C                               End of DEM-DEM
444 C
445 C-----
446 C                               Time Integration           NPDF: All Nodes
447 C-----
448 C
449 c                               coordinate update
450 C
451 c                               call tinteg(npdf,np0,np,g,dt,nbc)
452 C
453 c                               write(*,8099)
454 C
455 c                               do i=1,npdf
456 c                               x0(i)=x0(i)+u(i)
457 c                               z0(i)=z0(i)+v(i)
458 c                               qq(i)=qq(i)+f(i)
459 C
460 c                               if((i.eq.1).or.(i.eq.221)) then
461 c                               write(*,8088) i,x0(i),z0(i),u(i),v(i),xf(i),zf(i)
462 c                               format('i --x,z,u,v,xf,zf---')
463 c                               format(i5,6f12.3)
464 c                               end if
465 c                               end do
466 C
467 c                               if(mod(jj,50).eq.0) then
468 c                               k4=k4+1
469 c                               sinks(k4)=-(z0(155)-zini)
470 c                               wload(k4)=tvert
471 c                               end if
472 C
473 c                               write(*,*) 'Current Tire Sink [cm] =',
474 c                               z0(155)-zini,v(1)
475 c                               write(*,*) 'Current Tire Load [N] =',tvert
476 c                               write(*,*) 'Total Tire Load [N] =',allw,ti,jj
477 C
478 C
479 C
480 C
481 C
482 C
483 C
484 C
485 C
486 C
487 C
488 C
489 C
490 C
491 C
492 C
493 C
494 C
495 C
496 C
497 C
498 C
499 C
500 C
501 C
502 C
503 C
504 C
505 C
506 C
507 C
508 C
509 C
510 C
511 C
512 C
513 C
514 C
515 C
516 C
517 C
518 C
519 C
520 C
521 C
522 C
523 C
524 C
525 C
526 C
527 C
528 C
529 C
530 C
531 C
532 C
533 C
534 C
535 C
536 C
537 C
538 C
539 C
540 C
541 C
542 C
543 C
544 C
545 C
546 C
547 C
548 C
549 C
550 C
551 C
552 C
553 C
554 C
555 C
556 C
557 C
558 C
559 C
560 C
561 C
562 C
563 C
564 C
565 C
566 C
567 C
568 C
569 C
570 C
571 C
572 C
573 C
574 C
575 C
576 C
577 C
578 C
579 C
580 C
581 C
582 C
583 C
584 C
585 C
586 C
587 C
588 C
589 C
590 C
591 C
592 C
593 C
594 C
595 C
596 C
597 C
598 C
599 C
600 C
601 C
602 C
603 C
604 C
605 C
606 C
607 C
608 C
609 C
610 C
611 C
612 C
613 C
614 C
615 C
616 C
617 C
618 C
619 C
620 C
621 C
622 C
623 C
624 C
625 C
626 C
627 C
628 C
629 C
630 C
631 C
632 C
633 C
634 C
635 C
636 C
637 C
638 C
639 C
640 C
641 C
642 C
643 C
644 C
645 C
646 C
647 C
648 C
649 C
650 C
651 C
652 C
653 C
654 C
655 C
656 C
657 C
658 C
659 C
660 C
661 C
662 C
663 C
664 C
665 C
666 C
667 C
668 C
669 C
670 C
671 C
672 C
673 C
674 C
675 C
676 C
677 C
678 C
679 C
680 C
681 C
682 C
683 C
684 C
685 C
686 C
687 C
688 C
689 C
690 C
691 C
692 C
693 C
694 C
695 C
696 C
697 C
698 C
699 C
700 C
701 C
702 C
703 C
704 C
705 C
706 C
707 C
708 C
709 C
710 C
711 C
712 C
713 C
714 C
715 C
716 C
717 C
718 C
719 C
720 C
721 C
722 C
723 C
724 C
725 C
726 C
727 C
728 C
729 C
730 C
731 C
732 C
733 C
734 C
735 C
736 C
737 C
738 C
739 C
740 C
741 C
742 C
743 C
744 C
745 C
746 C
747 C
748 C
749 C
750 C
751 C
752 C
753 C
754 C
755 C
756 C
757 C
758 C
759 C
760 C
761 C
762 C
763 C
764 C
765 C
766 C
767 C
768 C
769 C
770 C
771 C
772 C
773 C
774 C
775 C
776 C
777 C
778 C
779 C
780 C
781 C
782 C
783 C
784 C
785 C
786 C
787 C
788 C
789 C
790 C
791 C
792 C
793 C
794 C
795 C
796 C
797 C
798 C
799 C
800 C
801 C
802 C
803 C
804 C
805 C
806 C
807 C
808 C
809 C
810 C
811 C
812 C
813 C
814 C
815 C
816 C
817 C
818 C
819 C
820 C
821 C
822 C
823 C
824 C
825 C
826 C
827 C
828 C
829 C
830 C
831 C
832 C
833 C
834 C
835 C
836 C
837 C
838 C
839 C
840 C
841 C
842 C
843 C
844 C
845 C
846 C
847 C
848 C
849 C
850 C
851 C
852 C
853 C
854 C
855 C
856 C
857 C
858 C
859 C
860 C
861 C
862 C
863 C
864 C
865 C
866 C
867 C
868 C
869 C
870 C
871 C
872 C
873 C
874 C
875 C
876 C
877 C
878 C
879 C
880 C
881 C
882 C
883 C
884 C
885 C
886 C
887 C
888 C
889 C
890 C
891 C
892 C
893 C
894 C
895 C
896 C
897 C
898 C
899 C
900 C
901 C
902 C
903 C
904 C
905 C
906 C
907 C
908 C
909 C
910 C
911 C
912 C
913 C
914 C
915 C
916 C
917 C
918 C
919 C
920 C
921 C
922 C
923 C
924 C
925 C
926 C
927 C
928 C
929 C
930 C
931 C
932 C
933 C
934 C
935 C
936 C
937 C
938 C
939 C
940 C
941 C
942 C
943 C
944 C
945 C
946 C
947 C
948 C
949 C
950 C
951 C
952 C
953 C
954 C
955 C
956 C
957 C
958 C
959 C
960 C
961 C
962 C
963 C
964 C
965 C
966 C
967 C
968 C
969 C
970 C
971 C
972 C
973 C
974 C
975 C
976 C
977 C
978 C
979 C
980 C
981 C
982 C
983 C
984 C
985 C
986 C
987 C
988 C
989 C
990 C
991 C
992 C
993 C
994 C
995 C
996 C
997 C
998 C
999 C

```

```

488      end do
489      if(allw.le.tvert) goto 868
490 C=====
491      850      continue
492 C=====
493      C          END OF TIME INTEGRATION LOOP
494 C=====
495      10      CONTINUE
496 C
497      1200  format(i5,2x,5F12.5)
498      1201  format(i5,2x,7F12.4)
499      1205  format(i5,2x,3F12.4)
500      1206  format(6i5)
501      1207  format(2i4,1x,F12.5)
502      1000  FORMAT(i5,2x,4F10.5)
503 C
504 C      Result File Save
505 C
506      868  continue
507
508      FILE1='fdemtestx.res'
509      OPEN(14,FILE=FILE1)
510      write(14,*) NP0,NE0,NP1,NE1,NPD
511      do i=1,NPDF
512          write(14,1205) i,x0(i),z0(i),qq(i)
513      end do
514      do i=1,NEDF
515          write(14,1206) i,(lnode(i,k),k=1,5)
516      end do
517      close(14)
518 C
519      OPEN(16,FILE='fdemtestx.stp')
520      write(16,1206) istep
521      close(16)
522      sbuf=0.0d0
523      open(14,FILE='fdemloadx.res')
524      write(14,*) k4+1
525      write(14,*) sbuf,sbuf
526      do i=1,k4
527          write(14,*) sinks(i),wload(i)
528      end do
529      close(14)
530      STOP
531      END
532 C=====
533      subroutine tinteg(nxx,np0,np,g,dt,nbc)
534      implicit real*8(A-H,O-Z)
535      PARAMETER(NPMAX=1000)
536      parameter (ni=2000)
537      common /posi/x0(ni),z0(ni),qq(ni)
538      common /velo/u0(ni),v0(ni),f0(ni)
539      common /for1/xf(ni),zf(ni),of(ni)
540      common /wep/rr(ni),wei(ni),pmi(ni)
541      common /accel/du0(ni),dv0(ni),df0(ni)
542      common /dpmn/u(ni+3),v(ni+3),f(ni+3)
543      dimension nbc(npmax,2)
544
545      do i=1,nxx
546          ww=wei(i)*1.0d0
547          if(nbc(i,2).ne.1) then
548              if(wei(i).eq.0.0d0) then
549                  write(*,*) 'ERROR: wei 2=0 at ',i
550                  stop
551              end if
552              if(i.gt.np0) then
553                  accl=zf(i)/ww
554              else
555                  accl=(zf(i)-wei(i)*g)/ww
556              end if
557              v0(i)=v0(i)+(dv0(i)+accl)*.5d0*dt
558              dv0(i)=accl
559          else
560              dv0(i)=0.0d0
561              v0(i)=0.0d0
562          end if
563
564          if(nbc(i,1).ne.1) then
565              if(wei(i).eq.0.0d0) then
566                  write(*,*) 'ERROR: wei 1=0 at ',i
567                  stop
568              end if
569              accl=xf(i)/ww
570              u0(i)=u0(i)+(du0(i)+accl)*.5d0*dt
571              du0(i)=accl
572          else
573              du0(i)=0.0d0
574              u0(i)=0.0d0
575          end if
576
577          if(pmi(i).ne.0.0d0) then
578              accl=of(i)/pmi(i)
579              f0(i)=f0(i)+(df0(i)+accl)*.5d0*dt
580              df0(i)=accl
581          else
582              df0(i)=0.0d0
583              f0(i)=0.0d0
584          end if
585
586          if(nbc(i,2).ne.1) then
587              vold=v(i)
588              v(i)=(v0(i)*dt+v(i))/2.0d0
589              vratio=v(i)/vold
590          else
591              v(i)=0.0d0
592          end if
593
594          if(nbc(i,1).ne.1) then
595              uold=u(i)
596              u(i)=(u0(i)*dt+u(i))/2.0d0
597              uratio=u(i)/uold
598          else
599              u(i)=0.0d0
600          end if
601
602          f(i)=(f0(i)*dt+f(i))/2.0d0
603          if((i.eq.155).or.(i.eq.221)) then
604              write(*,9008) i,xf(i),du0(i),u0(i),u(i)
605              write(*,9009) i,zf(i),dv0(i),v0(i),v(i)
606          end if
607          format('Node[',i4,'] :xf,du0,u0,u=',4f14.4)
608          format('Node[',i4,'] :zf,dv0,v0,v=',4f14.4)
609      end do
610      return
611      end
612 C=====
613 SUBROUTINE LNORML(X,Y,XL,YL,sinl,cosl,X1,Y1,X2,Y2,itype)
614 IMPLICIT REAL*8 (A-H,O-Z)
615 C
616 C      .3(X,Y)
617 C 2-----1
618 C
619 C XL,YL: Calculated Local Coordinate of 3 w.r.t. 2(Tire)
620 C XL,YL: Calculated Local Coordinate of 3 w.r.t. 1(Soil)
621 C THETA: Calculated angle of 1-2 axis to horizontal axis
622 C
623 DX=X2-X1
624 DY=Y2-Y1
625 DL=dsqrt(dx*dx+dy*dy)
626 c
627 sinl=dy/dl
628 cosl=dx/dl
629 C
630 if(itype.eq.1) then
631     XL=(X2-X)*COS1+(Y2-Y)*SIN1
632     YL=-(X2-X)*SIN1+(Y2-Y)*COS1
633 end if
634 if(itype.eq.2) then
635     XL=(X-X1)*COS1+(Y-Y1)*SIN1
636     YL=-(X-X1)*SIN1+(Y-Y1)*COS1
637 end if
638 RETURN
639 END
640 C
641 C=====
642 SUBROUTINE LCORD2(GX,GX0,GXN,Y0,YN)
643 C
644 C Find GZAI value on Local Coordinate on Line Element
645 C
646 C Y0: Before Contact; YN: After Con-
647 tact w/o modification
648 C GX: Found GZAI-value on modified contact
649 C GX0: Before Contact GZAI; GXN: After Con-
650 tact GZAI on YN
651 C If GX>=1 or <=0, outside of line element
652 C
653 IMPLICIT REAL*8 (A-H,O-Z)
654 if(y0.eq.yn) then
655     gx=gx0
656 else
657     GX=(Y0+GXN-YN*GX0)/(Y0-YN)
658 end if
659 RETURN
660 END
661 C=====
662 SUBROUTINE LFUNC2(GX,SS1,SS2)
663 C
664 C Local GZAI --> Shape Function of N_1, N_2
665 C
666 IMPLICIT REAL*8 (A-H,O-Z)
667 SS1=0.5d0*(1.0d0-GX)
668 SS2=0.5d0*(1.0d0+GX)
669 RETURN

```

```

664      END
665 C=====
666      SUBROUTINE ALLDAT(NBC,NB,IBC)
667 C
668      IMPLICIT REAL*8 (A-H,O-Z)
669      PARAMETER (NPMAX=1000,NEMAX=1000)
670      PARAMETER (MAXEN=100)
671      parameter (ni=2000)
672      common /elem/np,ne,np0,ne0,np1,nel,npd
673      common /demscl/zmax,wmax,zmin
674      common /posi/x0(ni),z0(ni),qq(ni)
675      common /wepr/rr(ni),wei(ni),pmi(ni)
676      common /femer/lnode(NEMAX,5)
677      common /tpara1/ef1,pof1
678      common /tpara2/ef2,pof2
679      common /spara1/ef11,pof11
680      common /presdd/dispre
681      common /tpara3/rhof1,rhof2
682      common /spara2/rhos1
683      common /sscal/zfmin
684 C      REAL*8 FX(*),FY(*)
685      DIMENSION NBC(NPMAX,*),IBC(MAXEN)
686      CHARACTER*20 DNAME
687 C
688 C      Input data file should be linked to FORT.5
689 C
690      WRITE(*,*) ' ==> Mesh Data File Name ?'
691 395 READ(*, '(1A14)') DNAME
692      IF(DNAME.EQ.'') THEN
693          WRITE(*,*) ' ... Invalid File Name. Re-
694 enter please!.'
695      GOTO 395
696      ENDIF
697      OPEN(5,FILE=DNAME,STATUS='OLD')
698 C
699      READ(5,*) NP0,NE0,NP1,NE1,NPD
700 c      write(*,*) NP0,NE0,NP1,NE1,NPD
701 C
702      NP=NP0+NP1
703      NE=NE0+NE1
704      NP99=NP+NPD
705      NE99=NE+NPD
706 C
707      zmax= 10000.0
708      wmax=-10000.0
709      zfmin=10000.0
710      DO I=1,NP99
711          READ(5,*) IDUMMY,X0(I),Z0(I),RR(I)
712          if(.i.gt.np0).and(.i.le.np) then
713              if(wmax.lt.dabs(x0(i))) wmax=dabs(x0(i))
714          end if
715          if(.i.gt.np) then
716              if(zmax.lt.dabs(z0(i))) zmax=dabs(z0(i))
717          end if
718          if(.i.le.np) then
719              if(zfmin.gt.z0(i)) zfmin=z0(i)
720          end if
721      END DO
722 C
723      DO I=1,NE99
724          READ(5,*) IDUMMY,(LNODE(I,K),K=1,5)
725      END DO
726 C
727      CLOSE(5)
728 C-----
729 -----
730 C      Input data file should be linked to FORT.5
731 C
732 C      WRITE(*,*) ' ==> FE B.C and Parameter File Name ?'
733 C 396 READ(*,*) DNAME1
734 C      IF(DNAME1.EQ.'') THEN
735 C          WRITE(*,*) ' ... Invalid File Name. Re-
736 enter please!.'
737 C      GOTO 396
738 C      ENDIF
739      OPEN(15,FILE='allmesh2.prm',STATUS='OLD')
740 C
741      READ(15,*) NB
742 C
743      DO K=1,3
744          DO I=1,NB
745              NBC(I,K)=0
746          END DO
747      END DO
748      DO I=1,NB
749          READ(15,*) IBC(I),NBC(IBC(I),1),NBC(IBC(I),2)
750 c      write(*,*) IBC(I),NBC(IBC(I),1),NBC(IBC(I),2)
751      END DO
752 C
753 c      tire: young's mod, posson's ratio, density
754 c      rim : young's mod, posson's ratio, density
755 c      soil: young's mod, posson's ratio, density
756 C
757      READ(15,*) ef1 ,pof1 ,rhof1
758      READ(15,*) ef2 ,pof2 ,rhof2
759      READ(15,*) ef11, pos11, rhos1
760 C
761 C      Prescribed Displacement Condition
762 C      ---> load condition
763      read(15,*) dispre
764 C
765      close(15)
766
767      RETURN
768      END
769 C=====
770      SUBROUTINE DEDATA
771 C
772 C      READ INITIAL ELEMENT POSITION AND VELOCITY --- DEM
773 C
774      IMPLICIT REAL*8 (A-H,O-Z)
775      common /conn/dt,fri,frw,ev,ew,po,pow,so,g,de,pi
776
777      OPEN(12,FILE='sdemx.dat',STATUS='OLD')
778 C
779      de : density of DEM element
780      dt : time increment
781 C
782      READ(12,*) de, dt
783 C
784      ev : Young's Modulus for DEM element
785      ew : Young's Modulus for FE wall
786      po : Poisson's Ratio for DEM element
787      pow: Poisson's Ratio for FE wall
788      fri: Coulomb Friction Coef for DEM
789      frw: Coulomb Friction Coef for FE wall
790 C
791      READ(12,*) ev,ew,po,pow,fri,frw
792      write(*,*) ev,ew,po,pow,fri,frw
793 C
794      CLOSE(12)
795      RETURN
796      END
797 C=====
798      SUBROUTINE SURFFE(isp,itYPE,nsf)
799 C
800      DEFINE FEM SURFACE W.R.T. DEM
801 C
802      IMPLICIT REAL*8 (A-H,O-Z)
803      PARAMETER (NEMAX=1000)
804      common /elem/np,ne,np0,ne0,np1,nel,npd
805      common /femer/lnode(NEMAX,5)
806      ccc common /nodesf/isnode(npmax),NSU
807      integer ISP(2,nemax), itype(nemax)
808
809      INS=0
810      DO I=1,NE
811          if(lnode(i,5).ne.1) goto 100
812
813          J =LNODE(I,2)
814          J1=LNODE(I,1)
815          INS=INS+1
816          ISEE(INS)=I
817          itype=1 for tire; itype=2 for soil
818
819          if(I.le.ne0) then
820              itype(INS)=1
821              ISP(1,INS)=J1
822              ISP(2,INS)=J
823          else
824              itype(INS)=2
825              ISP(1,INS)=J
826              ISP(2,INS)=J1
827          end if
828          write(*,*) ' Soil top=',I
829      END DO
830      continue
831      NSF=INS
832      RETURN
833      END
834 C=====
835      SUBROUTINE EXTFOR(NP0,NPX,dispre)
836 C
837      External Force Calc.
838 C
839      parameter (ni=2000,nc=20000)
840      IMPLICIT REAL*8 (A-H,O-Z)
841      common /for1/xf(ni),zf(ni),of(ni)
842
843      DO I=1,NP0
844          if(NPX.eq.i) then
845              xf(I)=0.0d0
846              zf(I)=zf(i)-dispre
847              of(I)=-dispre
848          end if
849      END DO

```

```

845         write(*,*) 'gravity force=',i,zf(i)
846     end if
847 END DO
848 RETURN
849 END
850 C=====
851 SUBROUTINE EXTDIS (dispre)
852 C
853 C External Force Calc.
854 C
855 IMPLICIT REAL*8 (A-H,O-Z)
856 PARAMETER (NEMAX=1000)
857 parameter (ni=2000)
858 common /elemmp/np,ne,np0,ne0,np1,nel,npd
859 common /dpmu/u(ni+3),v(ni+3),f(ni+3)
860 common /femer/lnode(NEMAX,5)
861 dimension irim(ni)
862 do i=1,ni
863     irim(i)=0
864 end do
865
866 do i=1,ne0
867     if(lnode(i,5).eq.11) then
868         do k=1,3
869             do k=1,4
870                 j=lnode(i,k)
871                 if(irim(j).eq.0) then
872                     u(j)=u(j)
873                     v(j)=v(j)-dispre
874                     f(j)=f(j)
875                     irim(j)=1
876                     if(j.eq.155) then
877                         write(*,*) 'j,u,v=',j,u(j),v(j)
878                     end if
879                 end if
880             end do
881         end if
882     end do
883 END
884 C=====
885 SUBROUTINE FSPARM (ALAX, IV, EKM, IKM, IE, NEMAX, KNF)
886 C
887 C This subroutine assembles the element stiffness
888 C matrix into the gloval matrix stored as ALAX
889 C
890 IMPLICIT REAL*8 (A-H,O-Z)
891 real*8 ALAX(IV,IV),EKM(IKM,IKM)
892 INTEGER KNF(NEMAX,*)
893 DO I=1,8
894     J=KNF(IE,I)
895     DO L=1,8
896         JL=KNF(IE,L)
897         IF (J.NE.0.AND.J1.NE.0) THEN
898             ALAX(J,J1)=ALAX(J,J1)+EKM(I,L)
899         END IF
900     end do
901 end do
902 RETURN
903 END
904 C=====
905 subroutine fposir(rmax)
906 implicit real*8 (a-h,o-z)
907 parameter (ni=2000,nc=20000)
908 common /elemmp/np,ne,np0,ne0,np1,nel,npd
909 common /conm/dt,fri,frw,ev,ew,po,pow,so,g,de,pi
910 common /wepr/rr(ni),wei(ni),pmi(ni)
911 common /celx/n,idx,idzi,ipz,w,c,ncl(nc),nncl(ni)
912 common /posi/x0(ni),z0(ni),qq(ni)
913 common /demsc/zmax,wmax,zmin
914 common /itest/izn,ixin
915 c
916 c w=wmax
917 c
918 c n0=np
919 rmax=rr(np+1)
920 rmin=rr(np+1)
921 zmax=z0(np+1)
922 zmin=z0(np+1)
923 c
924 do i=2,npd
925     k=np+i
926     if(rr(k).gt.rmax) rmax=rr(k)
927     if(rr(k).lt.rmin) rmin=rr(k)
928     if(z0(k).gt.zmax) zmax=z0(k)
929     if(z0(k).lt.zmin) zmin=z0(k)
930 end do
931 zz=zmax-zmin
932 c=rmin*1.35d0
933 idx=idint(wmax/c)+1
934 idzi=idint(zz/c)+1
935 ixin=idx
936 izin=idzi
937
938 return
939 end
940 C=====
941 subroutine ncel
942 implicit real*8 (a-h,o-z)
943 parameter (ni=2000,nc=20000)
944 common /elemmp/np,ne,np0,ne0,np1,nel,npd
945 common /celx/n,idx,idzi,ipz,w,c,ncl(nc),nncl(ni)
946 common /posi/x0(ni),z0(ni),qq(ni)
947 common /lforc/en(ni,14),es(ni,14)
948 common /demcc/je(ni,14)
949 common /dpmu/u(ni+3),v(ni+3),f(ni+3)
950 common /sscal/zfmin
951 c
952 do ib=1,(idx*idzi)
953     ncl(ib)=0
954 end do
955 do i=1,n
956     k=np+i
957     zst=z0(k)-zfmin
958     ib=idint(zst/c)*idx+idint(x0(k)/c)+1
959     ncl(ib)=k
960     nncl(k)=ib
961 end do
962 return
963 end
964 C=====
965 subroutine inmat
966 implicit real*8 (a-h,o-z)
967 parameter (ni=2000,nc=20000)
968 common /conm/dt,fri,frw,ev,ew,po,pow,so,g,de,pi
969 common /wepr/rr(ni),wei(ni),pmi(ni)
970 common /celx/n,idx,idzi,ipz,w,c,ncl(nc),nncl(ni)
971 common /elemmp/np,ne,np0,ne0,np1,nel,npd
972 c
973 so=1.0d0/2.0d0/(1.0d0+po)
974 do i=1,npd
975     k=np+i
976     Mass=wei(k)
977     Moment of Inertia=wei(k)*r^2/2
978
979     wei(k)=pi*rr(k)*rr(k)*de
980     pmi(k)=wei(k)*rr(k)*rr(k)*0.5d0
981 end do
982 return
983 end
984 C=====
985 subroutine wcont(i)
986 implicit real*8 (a-h,o-z)
987 c
988 parameter (ni=2000,nc=20000)
989 common /demsc/zmax,wmax,zmin
990 common /wepr/rr(ni),wei(ni),pmi(ni)
991 common /celx/n,idx,idzi,ipz,w,c,ncl(nc),nncl(ni)
992 common /dpmu/u(ni+3),v(ni+3),f(ni+3)
993 common /posi/x0(ni),z0(ni),qq(ni)
994 common /velo/u0(ni),v0(ni),f0(ni)
995 common /forl/xf(ni),zf(ni),of(ni)
996 common /lforc/en(ni,14),es(ni,14)
997 common /elemmp/np,ne,np0,ne0,np1,nel,npd
998 common /demcc/je(ni,14)
999 c
1000 xwi=x0(i)
1001 rwi=rr(i)
1002 c --- left wall (left wall is on x=0)
1003 c
1004 jk=11
1005 j=np+npd+1
1006 if(xwi.le.rwi) then
1007     as=0.0d0
1008     ac=-1.0d0
1009     gap=dabs(xwi)
1010     je(i,jk)=j
1011     call actf1(i,j,jk,as,ac,gap)
1012 else
1013     en(i,jk)=0.0d0
1014     es(i,jk)=0.0d0
1015     je(i,jk)=0
1016 end if
1017 c
1018 c --- under wall
1019 c
1020 c jk=12
1021 c j=n+2
1022 c if(zi.lt.ri) then
1023     as=-1.0d0
1024     ac=0.0d0
1025     gap=dabs(zi)
1026     je(i,jk)=n+2
1027     call actf1(i,j,jk,as,ac,gap)
1028 else

```

```

1029 c      en(i,jk)=0.0d0
1030 c      es(i,jk)=0.0d0
1031 c      je(i,jk)=0
1032 c      end if
1033 c
1034 c --- right wall (wmax: width of soil bin)
1035 c
1036 c      jk=13
1037 c      j=np+npd+3
1038 c      if((xwi+rwi).ge.wmax) then
1039 c          as= 0.0d0
1040 c          ac= 1.0d0
1041 c          gap=dabs(xwi-wmax)
1042 c          je(i,jk)=j
1043 c          call actfl(i,j,jk,as,ac,gap)
1044 c      else
1045 c          en(i,jk)=0.0d0
1046 c          es(i,jk)=0.0d0
1047 c          je(i,jk)=0
1048 c      end if
1049 c      return
1050 c      end
1051 C=====
1052 c      subroutine pcontx(i,rmax)
1053 c      implicit real*8 (a-h,o-z)
1054 c      parameter (ni=2000,nc=20000)
1055 c      common /elem/np,ne,np0,ne0,np1,nel,mpd
1056 c      common /wepr/rr(ni),wei(ni),pmi(ni)
1057 c      common /celx/n,idx,idzi,ipz,w,c,ncl(nc),nncl(ni)
1058 c      common /posi/x0(ni),z0(ni),qq(ni)
1059 c      common /for1/xf(ni),zf(ni),of(ni)
1060 c      common /lforc/en(ni,14),es(ni,14)
1061 c      common /demcc/je(ni,14)
1062 c      common /demsc/zmax,wmax,zmin
1063 c      common /sscal/zfmin
1064 c
1065 c      xi=x0(i)
1066 c      zi=z0(i)-zfmin
1067 c      ri=rr(i)
1068 c
1069 c      lup=indint((zi+2.d0*rmax)/c)
1070 c      lun=indint((zi-2.d0*rmax)/c)
1071 c      llf=indint((xi-2.d0*rmax)/c)
1072 c      lrg=indint((xi+2.d0*rmax)/c)
1073 c
1074 c      if(lun.lt.0) lun=0
1075 c      if(llf.lt.0) llf=0
1076 c      if(lrg.ge.idx) lrg=idx-1
1077 c      do 90 lz=lun,lup
1078 c          do 80 lx=llf,lrg
1079 c              ib=lz*idx+lx+1
1080 c              j=ncl(ib)
1081 c              if((j.eq.0).or.(j.eq.i)) goto 80
1082 c              do 11 jj=1,10
1083 c                  if (je(i,jj).eq.j) then
1084 c                      jk=jj
1085 c                      goto 70
1086 c                  end if
1087 c              continue
1088 c              do 12 jj=1,10
1089 c                  if (je(i,jj).eq.0) then
1090 c                      jk=jj
1091 c                      je(i,jj)=j
1092 c                      goto 70
1093 c                  end if
1094 c              continue
1095 c          12  continue
1096 c          xj=x0(j)
1097 c          zj=z0(j)-zfmin
1098 c          rj=rr(j)
1099 c          gap=dsqrt((xi-xj)*(xi-xj)+(zi-zj)*(zi-zj))
1100 c          if (gap.lt.(ri+rj)) then
1101 c              if (i.gt.j) then
1102 c                  ac=(xj-xi)/gap
1103 c                  as=(zj-zi)/gap
1104 c                  j0=0
1105 c                  do 555 jj=1,10
1106 c                      if (je(j,jj).eq.i) then
1107 c                          j0=jj
1108 c                          goto 554
1109 c                      endif
1110 c                  continue
1111 c              call actfl(i,j,jk,as,ac,gap)
1112 c              en(j,j0)=en(i,jk)
1113 c              es(j,j0)=es(i,jk)
1114 c              j0=0
1115 c          endif
1116 c      else
1117 c          en(i,jk)=0.d0
1118 c          es(i,jk)=0.d0
1119 c          je(i,jk)=0
1120 c      endif
1121 c      80  continue
1122 c
1123 c      90  continue
1124 c      return
1125 c      end
1126 C=====
1127 c      subroutine actflb(i,j,jk,as,ac,gap)
1128 c      implicit real*8 (a-h,o-z)
1129 c      parameter (ni=2000,nc=20000)
1130 c      common /elem/np,ne,np0,ne0,np1,nel,mpd
1131 c      common /conm/dt,fri,frw,ew,ew,po,pow,so,g,de,pi
1132 c      common /wepr/rr(ni),wei(ni),pmi(ni)
1133 c      common /celx/n,idx,idzi,ipz,w,c,ncl(nc),nncl(ni)
1134 c      common /posi/x0(ni),z0(ni),qq(ni)
1135 c      common /velo/u0(ni),v0(ni),f0(ni)
1136 c      common /for1/xf(ni),zf(ni),of(ni)
1137 c      common /lforc/en(ni,14),es(ni,14)
1138 c      common /demcc/je(ni,14)
1139 c      common /dpmm/u(ni+3),v(ni+3),f(ni+3)
1140 c
1141 c      nn=np+npd
1142 c      ri=rr(i)
1143 c      alpha=1.0d0
1144 c      beta=1.0d0
1145 c      if((j-nn).le.0) then
1146 c          rj=rr(j)
1147 c          dis=ri+rj-gap
1148 c          wei3=2.d0*wei(i)*wei(j)/(wei(i)+wei(j))
1149 c      else
1150 c          rj=0.0d0
1151 c          dis=ri-gap
1152 c          wei3=wei(i)
1153 c      end if
1154 c      enn=en(i,jk)
1155 c      if(enn.le.0.0d0) enn=1.0d0
1156 c      pois2 =1.0d0-po*po
1157 c      pois21=1.0d0-pow*pow
1158 c      if((j-nn).le.0) then
1159 c          exx1=pois2/ev
1160 c          exx2=pois2/ew
1161 c          requi=ri*rj/(ri+rj)
1162 c          bal=dsqrt(4.0d0*(exx1+exx2)*requi*enn/pi)
1163 c          elx1=dlog(4.0d0*ri/bal)-0.5d0
1164 c          elx2=dlog(4.0d0*rj/bal)-0.5d0
1165 c          eknn=pi/(2.0d0*(exx1*elx1+exx2*elx2))
1166 c          ekss=eknn*so
1167 c          vnn=beta*dsqrt(4.0d0*wei3*eknn)
1168 c          vss=so*vnn
1169 c      else
1170 c          exx1=pois2/ev
1171 c          exx2=pois21/ew
1172 c          requi=ri*rj/(ri+rj)
1173 c          if(rj.eq.0.0d0) requi=ri
1174 c          bal=dsqrt(4.0d0*(exx1+exx2)*requi*enn/pi)
1175 c          elx1=dlog(4.0d0*ri/bal)-0.5d0
1176 c          if(rj.eq.0.0d0) then
1177 c              elx2=0.0d0
1178 c          else
1179 c              elx2=dlog(4.0d0*rj/bal)-0.5d0
1180 c          end if
1181 c          eknn=alpha*pi/(2.0d0*(exx1*elx1+exx2*elx2))
1182 c          ekss=eknn*so
1183 c          vnn=beta*dsqrt(4.0d0*wei3*eknn)
1184 c          vss=so*vnn
1185 c      end if
1186 c
1187 c      if(eknn.ne.0.0d0) then
1188 c          ddt=1.0d-1*dsqrt(wei3/eknn)
1189 c          if(ddt.lt.dt) then
1190 c              write(6,*) 'dt > ddt=',ddt,i,j,jk,eknn,wei(i)
1191 c              stop
1192 c          end if
1193 c      end if
1194 c
1195 c      ui=u(i)
1196 c      uj=u(j)
1197 c      vi=v(i)
1198 c      vj=v(j)
1199 c      fi=f(i)
1200 c      fj=f(j)
1201 c
1202 c      un= (ui-uj)*ac+(vi-vj)*as
1203 c      us=- (ui-uj)*as+(vi-vj)*ac+(ri*fi+rj*fj)
1204 c
1205 c      if(en(i,jk).eq.0.d0) then
1206 c          if(un.ne.0.d0) us=us*dis/un
1207 c          un=dis
1208 c      end if
1209 c
1210 c      en(i,jk)=en(i,jk)+eknn*un
1211 c      es(i,jk)=es(i,jk)+ekss*us
1212 c
1213 c      dn=vnn*un/dt
1214 c      ds=vss*us/dt

```



```

1210 if(en(i,jk).lt.0.0d0) then
1211   en(i,jk)=0.0d0
1212   es(i,jk)=0.0d0
1213   dn=0.0d0
1214   ds=0.0d0
1215   je(i,jk)=0
1216   return
1217 else if ((j-nn).le.0) then
1218   frc=fri
1219 else
1220   frc=frw
1221 end if
1222 hn=en(i,jk)+dn
1223 hs=es(i,jk)+ds

1224 if(dabs(hs-frc*hn).gt.0.0d0) then
1225   hs=frc*dsign(hn,hs)
1226   ds=0.0d0
1227 end if
1228 xf(i)=-hn*ac+hs*as+xf(i)
1229 zf(i)=-hn*as+hs*ac+zf(i)
1230 of(i)= of(i)-ri*hs
1231 c write(*,*) ' * DEM-DEM: Cont Elem at ',i

1232 if(jk.le.10) then
1233   xf(j)=hn*ac-hs*as+xf(j)
1234   zf(j)=hn*as+hs*ac+zf(j)
1235   of(j)=of(j)-rj*hs
1236 c   if((j.eq.315).or.(j.eq.314)) then
1237 c     write(*,9000) i,j,zf(j)
1238 c   end if
1239 c   if((j.eq.363).or.(j.eq.364)) then
1240 c     write(*,9000) i,j,zf(j)
1241 c   end if
1242 end if
1243 9000 format(' * DEM-
1244 DEM: Cont Elem[' ,i4,' ][' ,i4,' ] zf=' ,e15.7)
1245 return
1246 end
C=====
1247 subroutine actd-
1248 fxb(i,j,jk,as,ac,gap,rx,rxz,rz,uj,vj,itt)
1249 implicit real*8 (a-h,o-z)
1250 PARAMETER (NPMAX=1000)
1251 parameter (ni=2000,nc=20000)
1252 common /elem/np,ne,np0,ne0,npl,nel,npd
1253 common /conm/dt,fri,frw,ew,po,pow,so,g,de,pi
1254 common /wepr/rr(ni),wei(ni),pmi(ni)
1255 common /celx/n,idx,idzi,ipz,w,c,ncl(nc),nncl(ni)
1256 common /posi/x0(ni),z0(ni),qq(ni)
1257 common /velo/u0(ni),v0(ni),f0(ni)
1258 common /forl/xf(ni),zf(ni),of(ni)
1259 common /lforc/en(ni,14),es(ni,14)
1260 common /demcc/je(ni,14)
1261 common /dpmm/u(ni+3),v(ni+3),f(ni+3)
1262 common /tpara1/ef1,pof1
1263 common /tpara2/ef2,pof2
1264 common /spara1/ef11,pos11
1265 common /spara2/rhos1
1266 common /cchek/ifcont(npmax)
1267 c
1268 c i: DEM, j: FEM
1270 c
1271 beta=1.0d0
1272 alpha=1.0d0
1273 if(itt.eq.1) then
1274   frc=fri
1275 else
1276   frc=frw
1277 end if
1278 pow =pos11
1279 ew =ef11
1280 c
1281 ri=rr(i)
1282 rj=0.0d0
1283 dis=gap

1284 wei3=wei(i)
1285 c
1286 enn=en(i,jk)
1287 if(enn.le.0.0d0) enn=1.0d0
1288 pois2=1.0d0-po*po
1289 pois21=1.0d0-pow*pow

1290 exx1=pois2/ev
1291 exx2=pois21/ew
1292 requi=ri*rj/(ri+rj)
1293 if(rj.eq.0.0d0) then
1294   requi=ri
1295 end if
1296 bal=dsqrt(4.0d0*(exx1+exx2)*requi*enn/pi)
1297 elx1=dlog(4.0d0*ri/bal)-0.5d0
1298 if(rj.eq.0.0d0) then

1299   elx2=0.0d0
1300 else
1301   elx2=dlog(4.0d0*rj/bal)-0.5d0
1302 end if
1303 eknn=pi/(2.0d0*(exx1*elx1+exx2*elx2))
1304 vnn=beta*dsqrt(4.0d0*wei3*eknn)
1305 vss=so*vnn
1306 eknn=alpha*eknn
1307 ekss=eknn*so

1308 ui=u(i)
1309 vi=v(i)
1310 fi=f(i)
1311 fj=0.0d0
1312 c
1313 c normal & tangential relative displacement
1314 c
1315 if(itt.eq.1) then
1316   us=(ui-uj)*ac-(vi-vj)*as+(ri*fi+rj*fj)
1317   un=(ui-uj)*as-(vi-vj)*ac
1318 end if
1319 if(itt.eq.2) then
1320   us=(ui-uj)*ac+(vi-vj)*as+(ri*fi+rj*fj)
1321   un=(ui-uj)*as+(vi-vj)*ac
1322 end if
1323 c if((i.eq.318).or.(i.eq.316)) then
1324 c   write(*,9003) i,ui-uj,vi-vj
1325 9003 format(' i,dui,dvi=' ,i5,2f12.7)
1326 c   write(*,9001) un,us,vnn,vss,fi
1327 c   end if
1328 9001 format(' un,us,vnn,vss,fi=' ,5f12.7)
1329 c
1330 if(itt.eq.1) ifcont(I)=7
1331 if(itt.eq.2) ifcont(I)=1

1332 if(en(i,jk).eq.0.0d0) then
1333   if(un.ne.0.0d0) us=us*dis/un
1334   un=gap
1335   end if
1336 c
1337 c Total normal force=en; Total tangential force=es
1338 c
1339 en(i,jk)=-eknn*un+en(i,jk)
1340 es(i,jk)=-ekss*us+es(i,jk)

1341 dn=vnn*un/dt
1342 ds=vss*us/dt
1343 c
1344 c Tentative
1345 c Total normal reaction hn; Total tangential reaction hs
1346 c
1347 hn=en(i,jk)
1348 hss=es(i,jk)
1349 c
1350 c Coulomb Friction check
1351 c
1352 cccc hnab=dabs(hn)
1353 if(dabs(hss-frc*hn).gt.0.0d0) then
1354   hss=frc*dsign(hn,hss)
1355   ds=0.0d0
1356 end if
1357 c
1358 c Final
1359 c Total normal reaction hn; Total tangential reaction hs
1360 c
1361 hn =hn +dn
1362 hs =hss +ds

1363 if(itt.eq.1) then
1364   rxx= hn*as-hs*ac
1365   rzz=-hn*ac-hs*as
1366 else
1367   rxx=-hn*as+hs*ac
1368   rzz= hn*ac+hs*as
1369 end if

1370 xf(i)=xf(i)+rxx
1371 zf(i)=zf(i)+rzz
1372 of(i)=of(i)+ri*hs

1373 if(i.ge.500) then
1374   write(*,9900) i,xf(i),zf(i)
1375 end if
1376 if((i.ge.310).and.(i.le.319)) then
1377   write(*,9900) i,xf(i),zf(i)
1378 end if
1379 9900 format('FE-
1380 DE cont at DEM',i4,' xf=' ,e12.4,' zf=' ,e12.4)
1381 return
1382 end
C=====
1383 subroutine detach(i,jk,itt)
1384 implicit real*8 (a-h,o-z)
1385

```

```

1386 PARAMETER (NPMAX=1000)
1387 parameter (ni=2000,nc=20000)
1388 common /elem/np,ne,np0,ne0,np1,nel,npj
1389 common /conn/dt,fri,frw,ew,po,pow,so,g,de,pi
1390 common /wepr/rr(ni),wei(ni),pmi(ni)
1391 common /celx/n,idx,idxi,ipz,w,c,ncl(nc),nncl(ni)
1392 common /posi/x0(ni),z0(ni),qq(ni)
1393 common /velo/u0(ni),v0(ni),f0(ni)
1394 common /for1/xf(ni),zf(ni),of(ni)
1395 common /lforc/en(ni,14),es(ni,14)
1396 common /demcc/je(ni,14)
1397 common /dpmu/u(ni+3),v(ni+3),f(ni+3)
1398 common /tpara1/ef1,pof1
1399 common /tpara2/ef2,pof2
1400 common /spara1/ef11,pos11
1401 common /spara2/rhos1
1402 common /cchek/ifcont(npmax)

1403 if(en(i,jk).le.0.0d0) then
1404   en(i,jk)=0.0d0
1405   es(i,jk)=0.0d0
1406   ifcont(I)=0
1407 end if
1408 xf(i)=xf(i)
1409 zf(i)=zf(i)
1410 of(i)=of(i)
1411 if(i.ge.500) then
1412   write(*,9904) i
1413 end if
1414 if((i.ge.310).and.(i.le.319)) then
1415   write(*,9904) i
1416 end if
1417 9904 format(' >>> FE-DE cont deleted at DEM',i4)
1418 return
1419 end
1420 C=====
1421 subroutine actf1(i,j,jk,as,ac,gap)
1422 implicit real*8 (a-h,o-z)
1423 parameter (ni=2000,nc=20000)
1424 common /elem/np,ne,np0,ne0,np1,nel,npj
1425 common /conn/dt,fri,frw,ew,po,pow,so,g,de,pi
1426 common /wepr/rr(ni),wei(ni),pmi(ni)
1427 common /celx/n,idx,idxi,ipz,w,c,ncl(nc),nncl(ni)
1428 common /posi/x0(ni),z0(ni),qq(ni)
1429 common /velo/u0(ni),v0(ni),f0(ni)
1430 common /for1/xf(ni),zf(ni),of(ni)
1431 common /lforc/en(ni,14),es(ni,14)
1432 common /demcc/je(ni,14)
1433 common /dpmu/u(ni+3),v(ni+3),f(ni+3)
1434 c write(*,*) 'n,idx,idxi,po,pow=',n,idx,idxi,po,pow
1435 c
1436 nn=np+npj
1437 ri=rr(i)
1438 beta=1.0d0
1439 if((j-nn).le.0) then
1440   rj=rr(j)
1441   dis=ri+rj-gap
1442   wei3=2.d0*wei(i)*wei(j)/(wei(i)+wei(j))
1443 else
1444   rj=0.0d0
1445   dis=ri-gap
1446   wei3=wei(i)
1447 end if
1448 enn=en(i,jk)
1449 if(enn.le.0.0d0) enn=1.0d0
1450 if((j-nn).le.0) then
1451   b1=(3.d0/2.d0/ev*ri*rj/(ri+rj))*(1.d0-po*po)
1452   & *enn)**(1.d0/3.d0)
1453   eknn=2.d0/3.d0*b1*ev/(1.d0-po*po)
1454   ekss=eknn*so
1455   vnn=beta*dsqrt(4.d0*wei3*eknn)
1456   vss=so*vnn
1457 else
1458   b1=((3.d0/4.d0*ri*(1.d0-po*po)/ev+(1.d0-
1459 pow*po)/ew)
1460   & *enn)**(1.d0/3.d0)
1461   eknn=4.d0/3.d0*b1*ev*ew/((1.d0-po*po)*ew+(1.d0-
1462 pow*po)*ev)
1463   ekss=eknn*so
1464   vnn=beta*dsqrt(4.d0*wei3*eknn)
1465   vss=so*vnn
1466 end if
1467 if(eknn.ne.0.0d0) then
1468   ddt=1.0d-1*dsqrt(wei3/eknn)
1469   if(ddt.lt.dt) then
1470     write(6,*) 'dt > ddt=',ddt,i,j,jk,eknn,wei(i)
1471     stop
1472   end if
1473 end if
1474 c
1475 ui=u(i)
1476 uj=u(j)
1477 vi=v(i)
1478 vj=v(j)
1479 fi=f(i)
1480 fj=f(j)
1481 un=(ui-uj)*ac+(vi-vj)*as
1482 us=- (ui-uj)*as+(vi-vj)*ac+(ri*fi+rj*fj)
1483 c
1484 if(en(i,jk).eq.0.0d0) then
1485   if(un.ne.0.0d0) us=us*dis/un
1486   un=dis
1487 end if
1488 c
1489 en(i,jk)=en(i,jk)+eknn*un
1490 es(i,jk)=es(i,jk)+ekss*us
1491 c
1492 dn=vnn*un/dt
1493 ds=vss*us/dt
1494 if(en(i,jk).lt.0.0d0) then
1495   en(i,jk)=0.0d0
1496   es(i,jk)=0.0d0
1497   dn=0.0d0
1498   ds=0.0d0
1499   je(i,jk)=0
1500   return
1501 else if ((j-nn).le.0) then
1502   frc=fri
1503 else
1504   frc=frw
1505 end if
1506 hn=en(i,jk)+dn
1507 hs=es(i,jk)+ds
1508 if(dabs(hs-frc*hn).gt.0.0d0) then
1509   hs=frc*dsign(hn,hs)
1510   ds=0.0d0
1511 end if
1512 xf(i)=-hn*ac+hs*as+xf(i)
1513 zf(i)=-hn*as+hs*ac+zf(i)
1514 of(i)=of(i)-ri*hs
1515 c write(*,*) ' * DEM-DEM: Cont Elem at ',i
1516 if(jk.le.10) then
1517   xf(j)=hn*ac+hs*as+xf(j)
1518   zf(j)=hn*as+hs*ac+zf(j)
1519   of(j)=of(j)-rj*hs
1520 c if((j.eq.315).or.(j.eq.314)) then
1521   write(*,9000) i,j,zf(j)
1522 c end if
1523 c if((j.eq.363).or.(j.eq.364)) then
1524   write(*,9000) i,j,zf(j)
1525 c end if
1526 end if
1527 9000 format(' * DEM-
1528 DEM: Cont Elem[' ,i4,'] [' ,i4,'] zf=',e15.7)
1529 return
1530 end
1531 C=====
1532 subroutine actdfx(i,j,jk,as,ac,gap,rxz,rzz,uj,vj,itt)
1533 implicit real*8 (a-h,o-z)
1534 PARAMETER (NPMAX=1000)
1535 parameter (ni=2000,nc=20000)
1536 common /elem/np,ne,np0,ne0,np1,nel,npj
1537 common /conn/dt,fri,frw,ew,po,pow,so,g,de,pi
1538 common /wepr/rr(ni),wei(ni),pmi(ni)
1539 common /celx/n,idx,idxi,ipz,w,c,ncl(nc),nncl(ni)
1540 common /posi/x0(ni),z0(ni),qq(ni)
1541 common /velo/u0(ni),v0(ni),f0(ni)
1542 common /for1/xf(ni),zf(ni),of(ni)
1543 common /lforc/en(ni,14),es(ni,14)
1544 common /demcc/je(ni,14)
1545 common /dpmu/u(ni+3),v(ni+3),f(ni+3)
1546 common /tpara1/ef1,pof1
1547 common /tpara2/ef2,pof2
1548 common /spara1/ef11,pos11
1549 common /spara2/rhos1
1550 common /cchek/ifcont(npmax)
1551 c
1552 i: DEM, j: FEM
1553 c
1554 beta=1.0d-1
1555 alpha=1.0d0
1556 if(itt.eq.1) then
1557   frc=fri
1558 else
1559   frc=frw
1560 end if
1561 pow=pos11
1562 ew=ef11
1563 c
1564 ri=rr(i)
1565 rj=0.0d0

```

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].

```

1566      dis=gap
1567      wei3=wei(i)
1568      enn=en(i,jk)
1569      if(enn.le.0.0d0) enn=1.0d0
1570      po2 =1.0d0-po*po
1571      po21=1.0d0-pow*pow
1572      b1=((3.d0/4.d0*ri*(po2/ev+po21/ew))
1573      &      *enn)**(1.d0/3.d0)
1574      eknn=4.d0/3.d0*b1*ev*ew/(po2*ew+po21*ev)
1575      eknn=alpha*eknn
1576      ekss=eknn*so
1577      vnn=beta*dsqrt(2.d0*wei3*eknn)
1578      vss=vnn*dsqrt(so)
1579      c
1580      ui=u(i)
1581      vi=v(i)
1582      fi=f(i)
1583      fj=0.0d0
1584      c
1585      c normal & tangential relative displacement
1586      c
1587      if(itt.eq.1) then
1588          us=- (ui-uj)*ac-(vi-vj)*as+(ri*fi+rj*fj)
1589          un= (ui-uj)*as-(vi-vj)*ac
1590      end if
1591      if(itt.eq.2) then
1592          us= (ui-uj)*ac+(vi-vj)*as+(ri*fi+rj*fj)
1593          un=- (ui-uj)*as+(vi-vj)*ac
1594      end if
1595
1596      if(itt.eq.1) ifcont(I)=7
1597      if(itt.eq.2) ifcont(I)=1
1598
1599      if(en(i,jk).eq.0.0d0) then
1600          if(un.ne.0.0d0) us=us*dis/un
1601          un=gap
1602      end if
1603      c
1604      c Total normal force=en; Total tangential force=es
1605      c
1606      en(i,jk)=-eknn*un+en(i,jk)
1607      es(i,jk)=-ekss*us+es(i,jk)
1608      dn=vnn*un/dt
1609      ds=vss*us/dt
1610      c
1611      c Tentative
1612      c Total normal reaction hn; Total tangential reaction hs
1613      c
1614      hn =en(i,jk)
1615      hss=es(i,jk)
1616      c
1617      c Coulomb Friction check
1618      c
1619      ccccc      hnab=dabs(hn)
1620      if(dabs(hss-frc*hn).gt.0.0d0) then
1621          hss=frc*dsgn(hn,hss)
1622          ds=0.0d0
1623      end if
1624      c
1625      c Final
1626      c Total normal reaction hn; Total tangential reaction hs
1627      c
1628      hn =hn +dn
1629      hs =hss +ds
1630      if(itt.eq.1) then
1631          rxx= hn*as-hs*ac
1632          rzz=-hn*ac-hs*as
1633      else
1634          rxx=-hn*as+hs*ac
1635          rzz= hn*ac+hs*as
1636      end if
1637      xf(i)=xf(i)+rxx
1638      zf(i)=zf(i)+rzz
1639      of(i)=of(i)+ri*hs
1640
1641      if(i.ge.500) then
1642          write(*,9900) i,xf(i),zf(i)
1643      end if
1644      if((i.ge.310).and.(i.le.319)) then
1645          write(*,9900) i,xf(i),zf(i)
1646      end if
1647      9900 format('FE-
DE cont at DEM',i4,' xf=',e12.4,' zf=',e12.4)
return
end

```

```

1          SUBROUTINE INVAR (STRESS, SIGM,DSBAR, THETA)
2      C
3      C This routine forms the stress invariants (2-D Mises)
4      C
5      IMPLICIT REAL (8) (A-H,O-Z)
6      REAL (8) :: STRESS(*)
7      SX =STRESS (1)
8      SY =STRESS (2)
9      TXY=STRESS (3)
10     SZ =STRESS (4)
11     SIGM=(SX+SY+SZ)/3.0d0
12     DSBl=((SX-SY)*(SX-SY)+(SY-SZ)*(SY-SZ)+(SZ-SX)*(SZ-
SX)+
13     &      6.0d0*TXY*TXY)/2.0d0
14
15     DSBAR=SQRT(dsbl)
16     IF (DSBAR.EQ.0.d0) THEN
17         THETA=0.d0
18     ELSE
19         DX = (2.0d0*SX-SY-SZ)/3.d0
20         DY = (2.0d0*SY-SZ-SX)/3.d0
21         DZ = (2.0d0*SZ-SX-SY)/3.d0
22         XJ3 =DX*DY*DZ-DZ*TXY*TXY
23         SINX=-13.5*XJ3/(DSBAR*DSBAR*DSBAR)
24         IF (SINX.GT. 1.0d0) SINX= 1.d0
25         IF (SINX.LT.-1.0d0) SINX=-1.d0
26         THETA=ASIN(SINX)/3.0d0
27     END IF
28     RETURN
29     END
30     SUBROUTINE NULVEC(A,IA)
31     IMPLICIT REAL (8) (A-H,O-Z)
32     REAL (8) :: A(*)
33     C
34     IF (MOD (IA,2).EQ.0) THEN
35         DO I=1,IA,2
36             A(I )=0.0d0
37             A(I+1)=0.0d0
38         END DO
39     ELSE
40         DO I=1,IA
41             A(I)=0.0d0
42         END DO
43     END IF
44     RETURN
45     END
46     SUBROUTINE NULVEI (IXA,IA)
47     IMPLICIT REAL (8) (A-H,O-Z)
48     integer :: IXA(*)
49     C
50     IF (MOD (IA,2).EQ.0) THEN
51         DO I=1,IA,2
52             IXA(I )=0
53             IXA(I+1)=0
54         END DO
55     ELSE
56         DO I=1,IA
57             IXA(I)=0
58         END DO
59     END IF
60     RETURN
61     END
62     SUBROUTINE NULL(A,IA,M,N)
63     C
64     C This subroutine nulls a 2-d array
65     C
66     IMPLICIT REAL (8) (A-H,O-Z)
67     REAL (8) :: A(IA,*)
68     DO I=1,N
69         DO J=1,M
70             A(J,I)=0.D0
71         end do
72     end do
73     RETURN
74     END
75     SUBROUTINE NULL2(a,ia,ig)
76     IMPLICIT REAL (8) (A-H,O-Z)

```

```

77      REAL(8):: a(ia,*)
78      C
79      do i=1,ig
80          do j=1,ia
81              a(j,i)=0.0d0
82          end do
83      end do
84      RETURN
85      END
86      SUBROUTINE NULL2I(ia,ian,ig)
87      C      IMPLICIT REAL*8 (A-H,O-Z)
88      integer:: ia(ian,*)
89      C
90      do i=1,ig
91          do j=1,ian
92              ia(j,i)=0
93          end do
94      end do
95      RETURN
96      END
97      SUBROUTINE GAUSS(SAMP,ISAMP,NGP)
98      C
99      C      This subroutine provides the weights and sam-
100      pling points
101      C      for gauss-legendre quadrature
102      C
103      IMPLICIT REAL(8) (A-H,O-Z)
104      REAL(8) ::SAMP(ISAMP,*)
105      if(ngp=1) goto 1
106      if(ngp=2) goto 2
107      1      SAMP(1,1)=0.DO
108      SAMP(1,2)=2.DO
109      GO TO 100
110      2      SAMP(1,1)=-1./DSQRT(3.DO)
111      SAMP(2,1)=-SAMP(1,1)
112      SAMP(1,2)=1.DO
113      SAMP(2,2)=1.DO
114      100      CONTINUE
115      RETURN
116      END
117      SUBROUTINE GAUSSV(VSAMP,IGP)
118      C
119      C      This subroutine provides the weights and sam-
120      pling points
121      C      for gauss-legendre quadrature ... Continu-
122      ous Version
123      C
124      IMPLICIT REAL(8) (A-H,O-Z)
125      REAL(8):: VSAMP(IGP,*)
126      VSAMP(1,1)=-1.d0/DSQRT(3.DO)
127      VSAMP(1,2)=-1.d0/DSQRT(3.DO)
128      VSAMP(2,1)= 1.d0/DSQRT(3.DO)
129      VSAMP(2,2)=-1.d0/DSQRT(3.DO)
130      VSAMP(3,1)= 1.d0/DSQRT(3.DO)
131      VSAMP(3,2)= 1.d0/DSQRT(3.DO)
132      VSAMP(4,1)=-1.d0/DSQRT(3.DO)
133      VSAMP(4,2)= 1.d0/DSQRT(3.DO)
134      RETURN
135      END
136      SUBROUTINE FORMLN(DER2,IDER,SAMP,ISAMP,I,J)
137      C
138      C      This subroutine forms the shape functions and their
139      C      derivatives for 4-noded quadrilateral elements
140      C
141      IMPLICIT REAL(8) (A-H,O-Z)
142      REAL(8):: DER2(IDER,*) ,SAMP(ISAMP,*)
143      REAL(8):: FUN(4)
144      C
145      C      samp(1,1)=1/sqrt(3) , samp(2,1)=-samp(1,1)
146      C
147      ETA= SAMP(I,1)
148      XI = SAMP(J,1)
149      ETAM =0.25d0*(1.0-ETA)
150      ETAP =0.25d0*(1.0+ETA)
151      XIM =0.25d0*(1.0-XI )
152      XIP =0.25d0*(1.0+XI )
153      FUN(1)=4.0d0*XIM*ETAM
154      FUN(2)=4.0d0*XIP*ETAM
155      FUN(3)=4.0d0*XIP*ETAP
156      FUN(4)=4.0d0*XIM*ETAP
157      DER2(1,1)=-ETAM
158      DER2(1,2)= ETAM
159      DER2(1,3)= ETAP
160      DER2(1,4)=-ETAP
161      DER2(2,1)=-XIM
162      DER2(2,2)=-XIP
163      DER2(2,3)= XIP
164      DER2(2,4)= XIM
165      RETURN
166      END
167      SUBROUTINE FORMLV(DER2,IDER,VSAMP,ISAMP,IG)
168      C
169      C      This subroutine forms the shape functions and their
170      C      derivatives for 4-noded quadrilateral elements
171      C
172      IMPLICIT REAL(8) (A-H,O-Z)
173      REAL(8):: DER2(IDER,*) ,VSAMP(ISAMP,*)
174      REAL(8):: FUN(4)
175      C
176      C      4---3      Integration Order
177      C      | |---> xi-axis
178      C      1---2
179      C
180      XI = VSAMP(IG,1)
181      ETA= VSAMP(IG,2)
182      C
183      ETAM =0.25d0*(1.0-ETA)
184      ETAP =0.25d0*(1.0+ETA)
185      XIM =0.25d0*(1.0-XI )
186      XIP =0.25d0*(1.0+XI )
187      C      FUN(1)=4.0d0*XIM*ETAM
188      C      FUN(2)=4.0d0*XIP*ETAM
189      C      FUN(3)=4.0d0*XIP*ETAP
190      C      FUN(4)=4.0d0*XIM*ETAP
191      DER2(1,1)=-ETAM
192      DER2(1,2)= ETAM
193      DER2(1,3)= ETAP
194      DER2(1,4)=-ETAP
195      DER2(2,1)=-XIM
196      DER2(2,2)=-XIP
197      DER2(2,3)= XIP
198      DER2(2,4)= XIM
199      RETURN
200      END
201      SUBROUTINE FORML1(DER2,IDER)
202      C
203      C      This subroutine forms the shape functions and their
204      C      derivatives for 4-noded quadrilateral elements
205      C
206      IMPLICIT REAL(8) (A-H,O-Z)
207      REAL(8):: DER2(IDER,*)
208      REAL(8):: FUN(4)
209      C
210      C      4---3      Integration Order
211      C      | |---> xi-axis
212      C      1---2
213      C
214      XI = 0.0d0
215      ETA= 0.0d0
216      C
217      ETAM =0.25d0*(1.0d0-ETA)
218      ETAP =0.25d0*(1.0d0+ETA)
219      XIM =0.25d0*(1.0d0-XI )
220      XIP =0.25d0*(1.0d0+XI )
221      C      FUN(1)=4.0d0*XIM*ETAM
222      C      FUN(2)=4.0d0*XIP*ETAM
223      C      FUN(3)=4.0d0*XIP*ETAP
224      C      FUN(4)=4.0d0*XIM*ETAP
225      DER2(1,1)=-ETAM
226      DER2(1,2)= ETAM
227      DER2(1,3)= ETAP
228      DER2(1,4)=-ETAP
229      DER2(2,1)=-XIM
230      DER2(2,2)=-XIP
231      DER2(2,3)= XIP
232      DER2(2,4)= XIM
233      RETURN
234      END
235      SUBROUTINE MATMUX(A,IA,B,IB,C,IC,L,M,N)
236      IMPLICIT REAL(8) (A-H,O-Z)
237      C
238      C      This subroutine forms the product of two matrices
239      C
240      REAL(8):: A(IA,*) ,B(IB,*) ,C(IC,*)
241      DO I=1,L
242          DO J=1,N
243              X=0.0D+0
244              DO K=1,M
245                  X=X+A(I,K)*B(K,J)
246              end do
247              C(I,J)=X
248          end do
249      end do
250      RETURN
251      END
252      SUBROUTINE MATRAN(A,IA,B,IB,M,N)
253      C
254      C      This subroutine forms the transpose of a matrix
255      C
256      IMPLICIT REAL(8) (A-H,O-Z)
257      REAL(8):: A(IA,*) ,B(IB,*)
258      DO I=1,M
259          DO J=1,N

```

```

260       A(J,I)=B(I,J)
261     end do
262   end do
263   RETURN
264   END
265   SUBROUTINE MSMULT(A,IA,C,M,N)
266   C
267   C   This subroutine multiplies a matrix by a scalar
268   C
269   IMPLICIT REAL(8) (A-H,O-Z)
270   REAL(8):: A(IA,*),C
271   DO J=1,N
272     DO I=1,M
273       A(I,J)=A(I,J)*C
274     end do
275   end do
276   RETURN
277   END
278   SUBROUTINE MATADD(A,IA,B,IB,M,N)
279   C
280   C   This subroutine adds two equal sized arrays
281   C
282   IMPLICIT REAL(8) (A-H,O-Z)
283   REAL(8):: A(IA,*),B(IB,*),C
284   DO J=1,N
285     DO I=1,M
286       A(I,J)=A(I,J)+B(I,J)
287     end do
288   end do
289   RETURN
290   END
291   SUBROUTINE MOCOUP(PHI,DSBAR,THETA,DQ1,DQ2,DQ3)
292   C
293   C   This subroutine forms the derivatives of a mohr-
294   coulomb
295   C   potential function with re-
296   spect to the three invariants
297   C   psi in degrees
298   C
299   IMPLICIT REAL(8) (A-H,O-Z)
300   PSIR=PHI*4.*DATAN(1.D0)/180.
301   SNTH=DSIN(THETA)
302   SNPS=DSIN(PSIR)
303   SQ3 =DSQRT(3.D0)
304   DQ1 =SNPS
305   IF (DABS(SNTH).GT. .49D0 ) THEN
306     C1=1.D0
307     IF (SNTH.LT.0.D0 ) C1=-1.D0
308     DQ2=(SQ3*.5d0-C1*SNPS*.5d0/SQ3)*SQ3*.5d0/DSBAR
309     DQ3=0.D0
310   ELSE
311     CSTH =DCOS(THETA)
312     CS3TH=DCOS(3.D0*THETA)
313     TN3TH=DTAN(3.D0*THETA)
314     TNTH=SNTH/CSTH
315     DQ2=SQ3*CSTH/DSBAR*((1.+TNTH*TN3TH)+SNPS*(TN3TH-
316     TNTH)/SQ3)*.5
317     DQ3=1.5d0*(SQ3*SNTH+SNPS*CSTH)/(CS3TH*DSBAR*DSBAR)
318   ENDIF
319   RETURN
320   END
321   SUBROUTINE MOCOUP(PHI,C,SIGM,DSBAR,THETA,F)
322   C
323   C   This subroutine calculates
324   lates the value of the yield function
325   C   for a mohr-coulomb material (phi in degrees)
326   C
327   IMPLICIT REAL(8) (A-H,O-Z)
328   PHIR=PHI*4.*DATAN(1.D0)/180.d0
329   SNPH=DSIN(PHIR)
330   CSPH=DCOS(PHIR)
331   CSTH=DCOS(THETA)
332   SNTH=DSIN(THETA)
333   F=SNPH*SIGM+DSBAR*(CSTH/DSQRT(3.D0)-SNTH*SNPH/3.d0)-
334   C*CSPH
335   RETURN
336   END
337   SUBROUTINE MVMULT(AM,IM,V,K,L,Y)
338   C
339   C   This subroutine multiplies a matrix by a vector
340   C
341   IMPLICIT REAL(8) (A-H,O-Z)
342   REAL(8):: AM(IM,*),V(*),Y(*)
343   DO I=1,K
344     X=0.D0
345     DO J=1,L
346       X=X+AM(I,J)*V(J)
347     end do
348     Y(I)=X
349   end do
350   RETURN
351   END
352   SUBROUTINE VECADD(A,B,C,N)
353   C
354   C   This subroutine adds vectors a+b=c
355   C
356   IMPLICIT REAL(8) (A-H,O-Z)
357   REAL(8):: A(*),B(*),C(*)
358   if(mod(n,2).eq.0) then
359     DO I=1,N,2
360       C(I )=A(I )+B(I )
361       C(I+1)=A(I+1)+B(I+1)
362     end do
363   else
364     DO I=1,N
365       C(I)=A(I)+B(I)
366     end do
367   end if
368   RETURN
369   END
370   SUBROUTINE VECOP(A,B,N)
371   C
372   C   This subroutine copies vector a into vector b
373   C
374   IMPLICIT REAL(8) (A-H,O-Z)
375   REAL(8):: A(*),B(*)
376   if(mod(n,2).eq.0) then
377     DO I=1,N,2
378       B(I )=A(I )
379       B(I+1)=A(I+1)
380     end do
381   else
382     DO I=1,N
383       B(I)=A(I)
384     end do
385   end if
386   RETURN
387   END
388   SUBROUTINE MATCOP(A,B,N,M)
389   C
390   C   This subroutine copies vector a into vector b
391   C
392   IMPLICIT REAL(8) (A-H,O-Z)
393   REAL(8):: A(M,*),B(M,*),C(*)
394   DO J=1,N
395     DO I=1,M
396       B(I,J)=A(I,J)
397     end do
398   end do
399   RETURN
400   END
401   SUBROUTINE GCOUNT(LG,IE,LNODE,NBC,NEMAX,NPMAX)
402   IMPLICIT REAL(8) (A-H,O-Z)
403   integer:: LNODE(NEMAX,*),NBC(NPMAX,*),LG(*)
404   INC=0
405   DO I=1,4
406     LL=LNODE(IE,I)
407     write(*,*) 'node=',ll
408     DO K=1,2
409       INC =INC+1
410       NOD1=NBC(LL,K)
411       IF(NOD1.EQ.0) THEN
412         LG(INC)=0
413       ELSE
414         LG(INC)=NOD1
415       END IF
416     write(*,*) 'lg(inc)=' ,lg(inc)
417   end do
418   END DO
419   C
420   RETURN
421   END
422   SUBROUTINE SETBC2(NBC,NPMAX,NBV,NO,IBF,NP,NB,NF)
423   C
424   IMPLICIT REAL(8) (A-H,O-Z)
425   INTEGER:: NBC(NPMAX,*),NBV(*),NO(*),IBF(*)
426   C
427   C   Calculation for Total Degrees of Freedom
428   L=0
429   DO I=1,NP
430     DO K=1,2
431       IF(NBC(I,K).EQ.1) GOTO 500
432       L=L+1
433       NBC(I,K)=L
434       NBV(L)=I
435     write(*,*) 'i,k,nbc(i,k)=' ,i,k,nbc(i,k)
436     DO J=1,NF
437       IF(I.EQ.IBF(J)) NO(J)=L
438     end do
439     GOTO 10
440   500   NBC(I,K)=0
441   10   end do
442   end do
443   NB=L
444   C
445   stop

```

```

445     RETURN
446     END
447     SUBROUTINE TWOBY2 (YJAC, IJAC, YJAC1, IJAC1, DET)
448 C
449 C     INVERSE OF 2 X 2 MATRIX
450 C
451     IMPLICIT REAL(8) (A-H,O-Z)
452     REAL(8):: YJAC(IJAC,*), YJAC1(IJAC1,*)
453     DET=YJAC(1,1)*YJAC(2,2)-YJAC(1,2)*YJAC(2,1)
454     YJAC1(1,1)= YJAC(2,2)
455     YJAC1(1,2)=-YJAC(1,2)
456     YJAC1(2,1)=-YJAC(2,1)
457     YJAC1(2,2)= YJAC(1,1)
458     DO K=1,2
459         DO L=1,2
460             YJAC1(K,L)=YJAC1(K,L)/DET
461         end do
462     end do
463     RETURN
464     END
465     SUBROUTINE FMDEPS (DEE, IDEE, E, V)
466 C
467 C     2D PLAIN STRAIN ELASTIC D-MATRIX GENERATION
468 C
469     IMPLICIT REAL(8) (A-H,O-Z)
470     REAL(8):: DEE (IDEE,*)
471 C
472     V1=1.0d0-V
473     C=E/(1.0d0+V)*(1.0d0-2.0d0*V)
474     DEE(1,1)=V1*C
475     DEE(2,2)=V1*C
476     DEE(3,3)=0.5d0*C*(1.0d0-2.0d0*V)
477     DEE(1,2)=V*C
478     DEE(2,1)=V*C
479     DEE(1,3)=0.0d0
480     DEE(3,1)=0.0d0
481     DEE(2,3)=0.0d0
482     DEE(3,2)=0.0d0
483     RETURN
484     END
485     SUBROUTINE FORMB (BEE, IBEE, DERIV, IDERIV, NOD)
486 C
487 C     This subroutine forms B-Matrix for Plain Strain
488 C
489     IMPLICIT REAL(8) (A-H,O-Z)
490     REAL(8):: BEE (IBEE,*), DERIV (IDERIV,*)
491     DO M=1,NOD
492         K=2*M
493         L=K-1
494         X=DERIV(1,M)
495         BEE(1,L)=X
496         BEE(3,K)=X
497         Y=DERIV(2,M)
498         BEE(2,K)=Y
499         BEE(3,L)=Y
500     END DO
501     RETURN
502     END
503     SUBROUTINE VMPL (E, V, STRESS, PL)
504 C
505 C     This routine forms the plastic matrix for a von-
506 Mises material
507 C
508     IMPLICIT REAL(8) (A-H,O-Z)
509     REAL(8):: STRESS (*), TERM(4), PL(4,*)
510     SX =STRESS(1)
511     SY =STRESS(2)
512     TXY=STRESS(3)
513     SZ =STRESS(4)
514     DSBAL=( (SX-SY)*(SX-SY) + (SY-SZ)*(SY-SZ) + (SZ-SX)*(SZ-
515 SX)
516     &      +6.0d0*TXY*TXY)/2.d0
517     DSBAR=SQRT (DSBAL)
518     EE1=1.5d0*E/((1.0+V)*DSBAR*DSBAR)
519     TERM(1)=(2.0d0*SX-SY-SZ)/3.d0
520     TERM(2)=(2.0d0*SY-SZ-SX)/3.d0
521     TERM(3)=TXY
522     TERM(4)=(2.0d0*SZ-SX-SY)/3.d0
523     DO I=1,4
524         DO J=1,4
525             PL(I,J)=TERM(I)*TERM(J)*EE1
526         PL(J,I)=PL(I,J)
527         end do
528     end do
529     RETURN
530     END
531     SUBROUTINE FKDIAG (KDIAG, IG, IDOF)
532     IMPLICIT REAL(8) (A-H,O-Z)
533     INTEGER:: KDIAG (*), IG (*)
534     DO I=1, IDOF
535         IWPL=1
536         IF (IG(I).EQ.0) GOTO 1
537         DO J=1, IDOF
538             IF (IG(J).EQ.0) GOTO 2
539             IM=IG(I)-IG(J)+1
540             IF (IM.GT.IWPL) IWPL=IM
541         CONTINUE
542         end do
543         K=IG(I)
544         IF (IWPL.GT.KDIAG(K)) KDIAG(K)=IWPL
545     CONTINUE
546     end do
547     RETURN
548     END
549     SUBROUTINE EC-
550 MAT (ECM, IECM, TN, ITN, TNT, ITNT, FUN, NOD, NODOF)
551 C
552 C     FORM CONSISTENT MASS MATRIX
553 C
554     IMPLICIT REAL(8) (A-H,O-Z)
555     REAL(8):: ECM (IECM,*), TN (ITN,*), TNT (ITNT,*), FUN (*)
556     IDOF=NOD*NODOF
557     DO I=1, IDOF
558         DO J=1, NODOF
559             TNT (I, J)=0.0D0
560             TN (J, I)=TNT (I, J)
561         end do
562     end do
563     DO I=1, NOD
564         DO J=1, NODOF
565             TNT ((I-1)*NODOF+J, J)=FUN (I)
566             TN (J, (I-1)*NODOF+J)=FUN (I)
567         end do
568     end do
569     DO I=1, IDOF
570         DO J=1, IDOF
571             X=0.0D0
572             DO K=1, NODOF
573                 X=X+TNT (I, K)*TN (K, J)
574             end do
575             ECM (I, J)=X
576         end do
577     end do
578     RETURN
579     END
580     SUBROUTINE FORMKV (BK, TKM, IKM, IG, N, IDOF)
581 C
582 C     STORE IN UPPER TRIANGLE AS A VECTOR BK(N*(IW+1))
583 C
584     IMPLICIT REAL(8) (A-H,O-Z)
585     REAL(8):: BK (*), TKM (IKM,*)
586     INTEGER:: IG (*)
587     DO I=1, IDOF
588         IF (IG(I).EQ.0) GOTO 1
589         DO J=1, IDOF
590             IF (IG(J).EQ.0) GOTO 5
591             ICD=IG(J)-IG(I)+1
592             IF ((ICD-1).GT.0) goto 5
593             IVAL=N*(ICD-1)+IG(I)
594             BK (IVAL)=BK (IVAL)+TKM (I, J)
595         CONTINUE
596         end do
597     CONTINUE
598     end do
599     RETURN
600     END
601     SUBROUTINE FSPARV (BK, TKM, IKM, IG, KDIAG, IDOF)
602 C
603 C     GLOBAL MATRIX STORED IN A VECTOR
604 C
605     IMPLICIT REAL(8) (A-H,O-Z)
606     INTEGER:: KDIAG (*), IG (*)
607     REAL(8):: BK (*), TKM (IKM,*)
608     DO I=1, IDOF
609         K=IG(I)
610         IF (K.EQ.0) GOTO 1
611         DO J=1, IDOF
612             IF (IG(J).EQ.0) GOTO 2
613             IW=K-IG(J)
614             IF (IW.LT.0) GOTO 2
615             IVAL=KDIAG(K)-IW
616             BK (IVAL)=BK (IVAL)+TKM (I, J)
617         CONTINUE
618         end do
619     CONTINUE
620     end do
621     RETURN
622     END
623     SUBROUTINE SPARIN (A, N, KDIAG)
624     IMPLICIT REAL(8) (A-H,O-Z)
625     REAL(8):: A (*), X

```

```

626     INTEGER ::KDIAG(*)
627     A(1)=DSQRT(A(1))
628     DO I=2,N
629         KI=KDIAG(I )-I
630         L =KDIAG(I-1)-KI+1
631         DO J=L,I
632             X=A(KI+J)
633             KJ=KDIAG(J)-J
634             IF(J.EQ.1) GOTO 2
635             LBAR=KDIAG(J-1)-KJ+1
636             LBAR=MAX0(L,LBAR)
637             IF(LBAR.EQ.J) GOTO 2
638             M=J-1
639             DO K=LBAR,M
640                 X=X-A(KI+K)*A(KJ+K)
641             end do
642         2     continue
643             A(KI+J)=X/A(KJ+J)
644         end do
645         A(KI+I)=DSQRT(X)
646     END DO
647     RETURN
648     END
649     SUBROUTINE SPABAC(A,B,N,KDIAG)
650     IMPLICIT REAL(8) (A-H,O-Z)
651     REAL(8) :: A(*),B(*)
652     INTEGER :: KDIAG(*)
653     B(1)=B(1)/A(1)
654     DO I=2,N
655         KI=KDIAG(I)-I
656         L =KDIAG(I-1)-KI+1
657         X=B(I)
658         IF(L.EQ.I) GOTO 1
659         M=I-1
660         DO J=L,M
661             X=X-A(KI+J)*B(J)
662         end do
663     1     continue
664         B(I)=X/A(KI+I)
665     end do
666     C
667     DO IT=2,N
668         I=N+2-IT
669         KI=KDIAG(I)-I
670         X=B(I)/A(KI+I)
671         B(I)=X
672         L=KDIAG(I-1)-KI+1
673         IF(L.EQ.I) GOTO 3
674         M=I-1
675         DO K=L,M
676             B(K)=B(K)-X*A(KI+K)
677         END DO
678     3     CONTINUE
679     end do
680     B(1)=B(1)/A(1)
681     RETURN
682     END

```