DEVELOPMENT OF SUBROUTINE LIBRARY AND DATA TRANSFER INTERFACE FOR HIGH TEMPERATURE STRUCTURAL INTEGRITY-CREEP

by

Feng Tan

A thesis submitted to the University of Huddersfield

in partial fulfilment of the requirements for

the degree of Doctor of Philosophy

School of Computing and Engineering,

The University of Huddersfield

Augest, 2015

DECLARATION

This dissertation is the result of my own work and includes nothing, which is the outcome of work done in collaboration except where specifically indicated in the text. It has not been previously submitted, in part or whole, to any university of institution for any degree, diploma, or other qualification.

In accordance with the Department of Engineering guidelines, this thesis is does not exceed 40,000 words, and it contains fewer than 150 figures.

Signed: _____

Date: _____

Feng Tan

COPYRIGHT STATEMENT

ABSTRACT

Creep plays a critical role in the research of high temperature materials because it is the major failure form of high temperature devices. In the safety assessment of high temperature devices, creep failure is one of the key factors used to evaluate residual lifetime of metal components; however, creep analysis in practical applications is still a great challenge due to the lack of a unified theory of all materials. A number of researchers are conducting research into creep constitutive model based on either experimental approaches or computational approaches, but multifarious computational tools were used because the constitutive model is still in the exploration stage. Traditional commercial software could reach the required capability based on the development of user-developed codes; moreover, some in-house codes were proposed, but just used in a narrow scope. Therefore, the development of a novel universal creep finite element software needs to be carried out to meet the requirements of future research.

This research aims to develop required subroutines and interface for the proposed elastic-creep finite element software called High Temperature Structural Integrity-Creep (HITSI). Basic concepts and situations of creep and its computational tools have been reviewed. General knowledge of programming of finite element method has also been studied. A universal subroutine template of creep constitutive equations has been given to enable users to add their own equations directly. A high order and embedded numerical method called Runge-Kutta-Fehlberg (RKF) method has been applied and discussed in order to enhance the accuracy of traditional methods. A mathematical method used to improve the accuracy and efficiency of constitutive equations subroutine call normalization has been applied and discussed. Formatted input and output of purchased pre- and post-processor has been studied to develop the data transfer interface. Some auxiliary modules such as stress transformation and nodal load arrangement have been developed to satisfy the input conditions of constitutive equations subroutines and data transfer interface.

Acknowledgements

The author would like to express his sincere gratitude to his major supervisor Dr. Qiang Xu. Xu sets up the direction and fundamental frame for this research. His enthusiasm, encouragement and useful guidance make this research smooth and solid.

Moreover, the author also needs to thank co-supervisor Prof. Zhongyu Lu for her support of required software and devices.

Thanks to my colleague Dezheng Liu and my previous co-supervisor Reader Donglai Xu because of their useful help.

Finally, profound gratitude to my dear parents for their care and financial support.

CONTENTS

Word count: in total 36786 words which excludes the appendices

LIST OF FIGURES

LIST OF TABLES

LIST OF ACRONYMS

CDM	Continuum Damage Mechanics
CES	Constitutive Equations Subroutine
CGM	Cavity Growth Mechanism
CPT	Classical Plastic Theory
DTI	Data Transfer Interface
FEA	Finite Element Analysis
FEM	Finite Element Method
HAZ	Heat Affect Zone
KR	Kachanov-Rabotnov
NLC	Nodal Load Calculator
NMS	Numerical Method Subroutine
NOR_KR	Normalized Kachanov-Rabotnov
ODEs	Ordinary differential equations
PH	Perrin-Hayhurst
QX	Qiang Xu's
RK	Runge-Kutta
RK4	4^{th} order classical Runge-Kutta
RKM	Runge-Kutta-Merson
RKF	Runge-Kutta-Fehlberg
TRS	Transformation of Stress
TSC	Time Step Control
UMIST	University of Manchester Institute of Science and Technology

1 INTRODUCTION

1.1 Background

The earliest creep phenomenon of metal wire was observed in 1900s, when Andrade (1910) firstly identified the creep characteristics of several pure metals via a set of experiments in 1910. A few years later, people realised the metal components always have creep deformation when loaded in high temperature environment even though the loaded stress is much smaller than yield strength of material during the same temperature condition. The high temperature and high-pressure process industry was developed rapidly after 1920s, see (Lai, 1990, Boyer, 1996, Eason et al., 2015), in the same time, the research of creep has been paid more and more attention. In recent years, the issues of creep and creep damage are becoming more and more serious in high temperature industries. For example, in the catalytic cracking device of petroleum industry (Sadeghbeigi, 2012), the manufacturing materials of pressure vessels normally are carbon steel or carbon-molybdenum steel in which working temperature of carbon steel and carbon-molybdenum steel could reach 510 ℃ and 540 ℃ respectively. The major failure form of this type of device is creep. The weldments of steam pipework of power generation plant are extremely sensitive to creep because of the different metallurgical regions between the weld and base material (Wang et al., 2015). During welding, the heating and subsequent cooling of the weldments is the major factor to form different metallurgical regions between the weld and base material.

Finite element method (FEM) was introduced into creep damage research to validate the tentative constitutive model or to intuitively observe the visualized evolution process of

creep strain and damage. For example, finite element analysis (FEA) with a modified form of Kachanov-Rabotnov (KR) constitutive equations was carried out to simulate the damage evolution of modelling a specimen with cooling holes, its FEM results revealed that the existence of cooling holes causes the concentration of stress and strain (Yu et al., 2008). However, the existing situation of computational tools is not optimistic due to the lack of a unified theory of creep. The mainstream computational tools can be classified into two categories, one is commercial software and the other is in-house software. The present commercial FEA software such as ABAQUS (Jiang et al., 2015) or ANSYS (Ni et al., 2015) allows the customer to develop a user's subroutine to conduct creep damage analysis simulation. Although the development of in-house software is difficult, the controllability is much better than commercial software.

In order to obtain a useful computational tool, High Temperature Structural Integrity-Creep (HITSI), which is an elastic-creep finite element analysis system, was proposed during this research. HITSI consists of three parts, which are a) pre- and post-processor, b) solver and c) data transfer programme. FEMGV, a set of purchased software, was selected to be the pre- and post-processor. The solver was developed by the author and his colleague Dezheng Liu[1] (2015), in which case, the author makes contribution to the development of a series of subroutines such as constitutive equations subroutines, time-step control subroutine and numerical method subroutines. The author built the data transfer programme, which used to convert the files of FEMGV and the solver because of their different file format, independently.

This thesis starts with a detailed survey of the creep research domain. The knowledge of high temperature structural integrity theory is reviewed to understand the significances of creep failure in high temperature industry. The basic concepts of creep and creep damage, which include the mechanisms of creep deformation and rupture, constitutive models and the major research approaches, are reviewed in order to have a general understanding of creep damage analysis itself. The present situation of computation tools is introduced to assess the advantages and disadvantages of existing finite element software. Finally, basic knowledge of programming of FEM and a number of adopted general-purpose subroutines are studied to start this research.

[1] Dezheng Liu undertakes the work of overall structure design, assembly of subroutines and practical case test.

General computing science methodology, software development life cycle, is adopted in this research. Spiral model will be applied to the development of HITSI due to the author's research and Dezheng Liu's researches are interactive; however, for those subroutines and data transfer programme, the waterfall model is much better. A number of development processes such as requirements analysis, design, coding and testing were reported.

The relative theories, which are used to develop subroutines and programmes, will be studied and discussed in order to clear the requirements and framework of each subroutine and programme. A number of subroutines need to be developed, which are a) constitutive equations subroutines, b) numerical method subroutines, c) time-step control subroutine, d) normalization technique subroutine, and e) stress tensor transformation subroutine. All subroutines were held in a subroutine library called **Tan_library**. Due to the missing capability of nodal loads conversion, a simple calculation tool called Nodal Loads Calculator (NLC) was developed to assist the data transfer programme, which is named Data Transfer Interface (DTI).

Each subroutine and programme is reported through analysing goals, defining arguments and drawing flowchart. The goals analysis analysed the working environment of those subroutines and programmes in order to clarify their structures. Argument definitions present the size and type of each variable whatever it is a global variable or a local variable. The flowchart was converted to write the pseudo code for coding.

Due to the development of HITSI adopted a lot of existing code resources, which are written by Fortran, the programing language of this thesis is Fortran as well. Compared with C++, Python, Matlab, S-Plus etc. which all often used for scientific computing, the advantages of Fortran is obvious such as its portability and array handling, and Fortran is also the best for shared memory in parallelism.

Static verification and dynamic verification[2] are both performed for all subroutines and programmes. The static testing is performed through Code::Blocks platform to guarantee no coding mistake in this research. Some testing programmes are designed for those

[2] Static verification means the proofreading of the code itself, and dynamic verification means the execution of code.

developed subroutines. Only some classical testing cases are reported in this thesis due to the large amount of random testing data are restricted by the space.

The performances of numerical method subroutines, time-step control subroutine and normalization technique subroutine will be discussed in order to identify the accuracy and efficiency of these subroutines to guide the potential users. The implications of this research are summarized in the end.

1.2 Aim and Objectives

The overall project, of which this thesis forms a part, aims to establish a set of software system (HITSI) to handle complex geometric simulation of creep damage analysis for industrial requirements. HITSI focuses on offering expert advising to enhance the relative researches such as prediction of residual lifetime, determination of design code and development of constitutive equations. It intends to could be widely used for industrial producing such as to evaluate operating state of the steam pipework of power generation plant.

As a significant part of development of HITSI, this thesis aims to research and devise a series of indispensable components such as constitutive equations subroutines, time-step control procedure and data transfer interface for HITSI. The research objectives of this project are classified as follows:

1. Structural analysis of HITSI, which is used to understand what kinds of subroutine or programme are required;
2. Development of constitutive equations subroutine class, which is used to enable the capacity of creep damage analysis of the solver;
3. Development of numerical method subroutine class, which is used to assist constitutive equations subroutine class, and to quantitatively analyse applied novel numerical method;
4. Development of modular stress conversion subroutine, which is used to satisfy the input conditions of constitutive equations subroutine class;
5. Exploration of time-step control procedures and normalization technique, which is used to enhance the accurate and efficient performance of numerical integration;
6. Development of data transfer interface, which is used to connect pre- and post-processor and the solver.

1.3 Thesis Layout

The structure of this thesis is summarised below:

The second chapter presents a review of the context knowledge of this research, which includes a) a brief introduction of high temperature structural integrity theory; b) basic concepts of creep, continuum damage mechanics and creep constitutive models; c) the present research approaches and their tools; and d) basic knowledge of programming of FEM and adopted general purpose subroutines.

The third chapter presents the development strategy of this research, which includes development targets and development methodology. The development targets discuss the properties and functions of the developed subroutines and programmes. The development methodology briefly introduces the knowledge of general programme development methodology, and reports the adopted development model and activities.

The fourth chapter presents the understanding of the theories and knowledge used to develop those subroutines and programmes. These theories and knowledge involve mechanical engineering, numerical analysis and computing science, which are a) creep constitutive equations, b) stress transformation, c) numerical methods, d) time-step control procedures, e) normalization technique, f) nodal loads arrangement, and formatted input and output of pre- and post-processor and the solver.

The fifth chapter presents the created processes of required subroutines and programmes, which includes algorithm design, variables definitions and structural display. The algorithm is presented through a flowchart and the structure is explained via a pseudo code. Detailed source codes are not shown in text, but are attached in the appendix.

The sixth chapter presents the implementation of testing and verification, which includes descriptions of testing tools and cases, test results and discussions of the results.

The seventh chapter presents the exploration of the performance of numerical method subroutines, time-step control subroutines and normalization subroutines. The accuracy and efficiency of them will be quantitatively analysed.

The eighth chapter is conclusion, which presents the summaries of the contributions and outcomes of this research. The discussion of future works is also presented.

2 SURVEY OF RESEARCH DOMAIN

This chapter reviews the context knowledge of creep research domain. The inverted triangle narrative method was used to report the understanding of creep research. Figure 2-1 presents the relationship of each knowledge range from wide to narrow. Initial research of high temperature structural integrity was launched to redesign, remanufacture and extend lifetime of the active high temperature process equipment. In the high temperature failure, creep is one of the major failure factors. Normally, people understood the constitutive relation of creep through experiment and numerical simulation. An FE package is often used to perform such numerical simulation. Compared with commercial FE package, the in-house FE package has its own advantages.

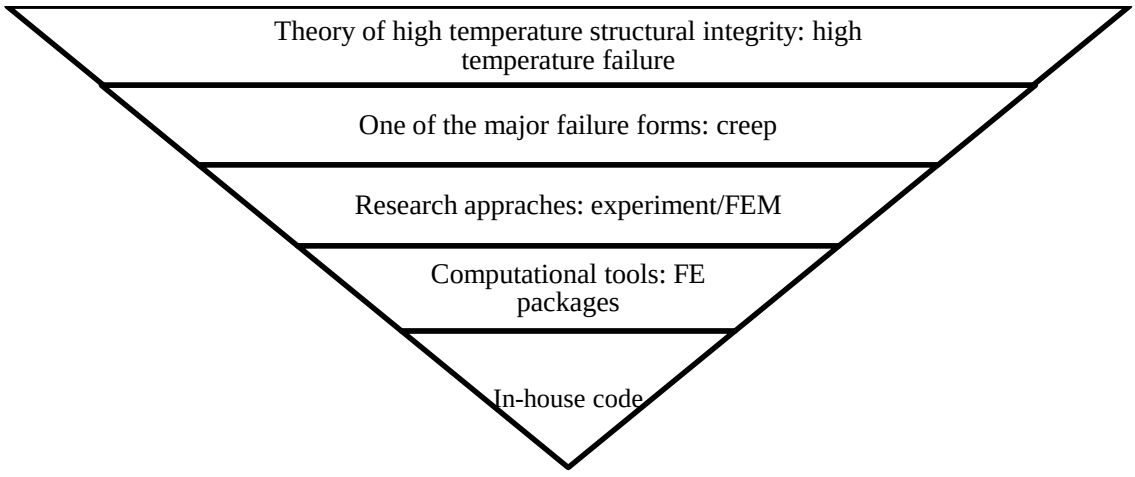

Figure 2-1 Relationship overview of theoretical knowledge of creep research domain

The concept and theory of high temperature structural integrity includes high temperature process and its materials, failure mechanisms and assessment standards. A brief introduction is displayed below.

2.1 Present Situation of High Temperature Industry

Modern industrial processes were developed toward the direction of high temperature and high pressure; however, the rise of corresponding knowledge lags the requirement of the processes. Many problems cannot be answered exactly and clearly according to the existing theories (Tu, 2003); for example, the steam pipework of power generation plant normally operating above 560 ℃, and its residual lifetime is still uncertain.

Creep, fatigue and corrosion are major factors affecting the high temperature structural integrity of metal. Compared with fatigue and corrosion, creep is especially serious in a high temperature working environment (Tu, 2007). Figure 2-2 indicates the operating temperature of plants since 1796s to 2000s. The structural integrity has been widely used with great success for the design, manufacture and failure prevention of modern constructions.

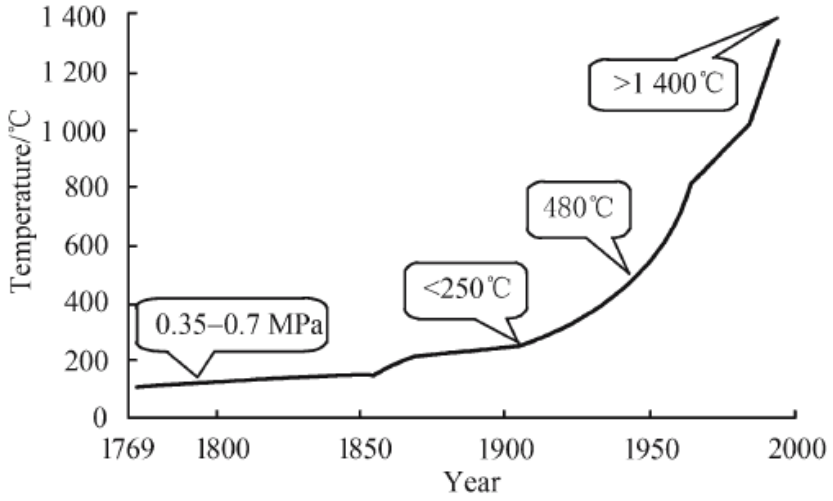

Figure 2-2 Temperature change tendency of operation of power plants (Tu, 2007)

The needs of structural integrity technology are influenced by the increase of service temperature of the structures. Besides the raised needs from high temperature plants, the development of high technology provides new challenges to the structural integrity technology (Tu, 2005). The risks of high pressure can be less worrying due to the development of modern design methods and computer technology; however, compared

with the high-pressure technology, the problems caused by the high temperature are more complex.

There is a lack of knowledge of high temperature equipment design either strength theory or process industry; actually, many problems could not be answered in the framework of existing disciplines, see (Berto et al., 2014, Islam et al., 2014, Quayyum et al., 2014, Zhu et al., 2014).

Creep research is significant and valuable; for example, Wen et al. (2014) presented a creep damage model from the micromechanics viewpoint. In order to mitigate the difficulty of calibrating many parameters in the existing damage evolution models, creep ductility exhaustion approach is employed to account for the accumulation of the creep damage.

As one of the major failure forms of high temperature structures, creep has two research branches; one is based on the classical plasticity theory and the other is based on the continuum damage mechanics. The application of continuum damage mechanics in creep research could be understood through some brief introductions of creep, damage, creep damage mechanisms and creep constitutive models.

2.2 Creep and Creep Damage

The concept of creep and continuum damage mechanics is briefly introduced in order to indicate the difference between creep and creep damage. This difference could be presented from the aspect of micro-mechanisms of creep deformation and creep rupture. As the core issues of this research, the constitutive models whether based on the classical plastic theory or the continuum damage mechanics were both studied.

2.2.1 Concept of creep

Creep is a type of deformation, which is a slow plastic deformation of metal caused by sustained stress at a certain temperature (Kassner, 2009). It is invisible in a relatively short time (counted by year) under room temperature; however, when the temperature exceeds one third of absolute melting temperature of metal, creep deformation will be exacerbated and can be measured. In order to distinguish the difference in creep deformation progress, it has been divided as three stages (Kassner, 2009). Figure 2-3 shows a typical creep curve. In the primary creep stage, the strain rate is relatively high,

but slows with increasing time. The secondary stage, which is known as steady state creep stage, is the most understood. The term "creep strain rate" typically refers to the secondary stage. The tertiary stage, creep deformation will be accelerated in a short time relative to previous stages. Both the primary and secondary stages are similar to the "pure" plastic deformations problem, but the tertiary stage is not. The damage-accumulation occurs due to accelerating creep because of grain boundaries defects.

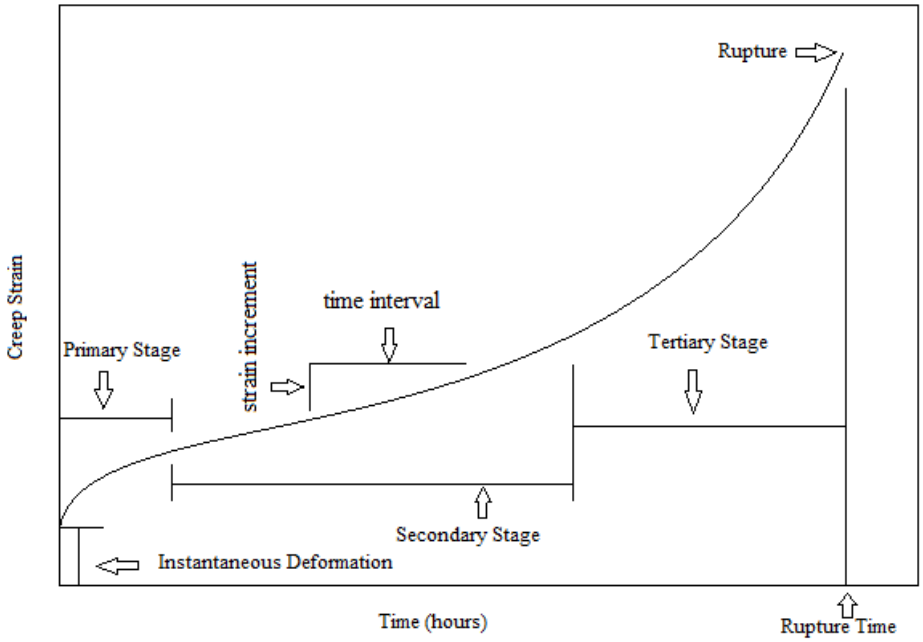

Figure 2-3 Evolution of creep deformation

2.2.2 Damage mechanics

A brief specification of continuum damage mechanics was addressed by Voyiadjis and Kattan (2002), assuming an axial pressure on a cylinder bar which is shown on the left of Figure 2-4, and a lot of voids and cracks appear on the bar. Voids and cracks represent the damage. Two different models were obtained if envisage removing all the damages out of the bar. As follows:

$$P = \sigma A \qquad (2\text{-}1)$$

$$P = \bar{\sigma}\bar{A} \qquad (2\text{-}2)$$

Where P is pressure, A is true area, $\bar{A}$ is theoretical area, σ is true stress and $\bar{\sigma}$ is nominal stress. Using the damage variable $\Phi = \frac{A-\bar{A}}{A}$ defined by Kachanov (1999),

$$\bar{\sigma} = \frac{\sigma}{1 - \Phi} \qquad (2\text{-}3)$$

Obviously, the value of damage variable must less than 1, because it is in the denominator. In addition, if the damage variable is equal to 1, that means the component has completely ruptured.

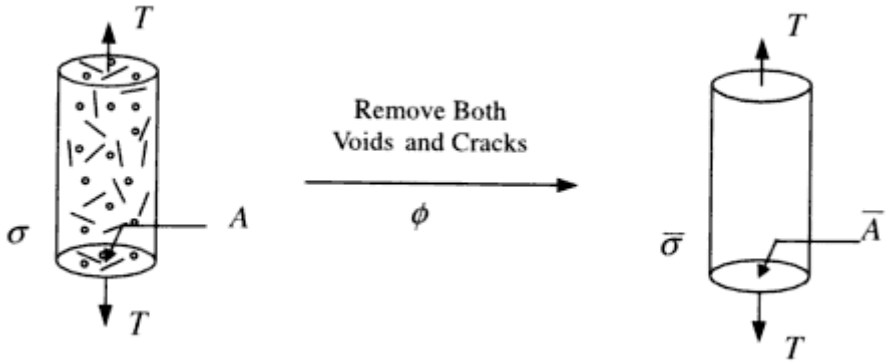

Figure 2-4 Definition of damage (Voyiadjis and Kattan, 2002)

2.2.3 Mechanisms of creep deformation and rupture

2.2.3.1 Mechanisms of creep deformation

The creep deformation may be influenced by temperature in two ways, which are a) dependence of the material constants on temperature and b) structural changes of the material. Generally, as temperature increases the creep strain rate goes up due to the increasing activation of the structural elements, in spite of the slip influenced strain hardening of the material (Skrzypek and Hetnarski, 1993, Gittus, 1975). When the temperature of the creep test is relatively low, the predominance of the slip type creep mechanism characteristic for the primary creep is observed. The various deformation mechanism regimes could be visualized versus stress and the holonomic temperature ranges in the universal deformation map, see Figure 2-5.

An increasing temperature causes the thermal activity of the dislocations in the crystalline lattice structure of the metal to rise. Hence, the dislocations can overcome the natural stiffness of the crystal and obstacles, to move through the lattice. The dislocation creep mechanism diminishes the strain hardening effect in the material although the cross-slip mechanism, which mainly characterizes the primary creep, still dominates. The higher temperature causes the dislocation creep to overpass the hardening mechanism; hence, the secondary creep phase may occur (Skrzypek and Hetnarski, 1993).

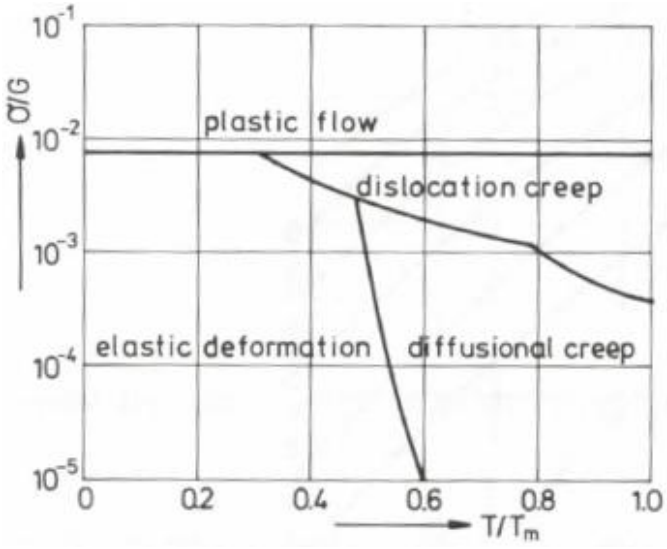

Figure 2-5 Typical deformation map of dislocation creep and diffusional creep (Skrzypek and Hetnarski, 1993)

It is worth to note that the dislocation creep, which in the fundamental creep mechanism in engineering structures, is strongly affected by stress. When the stress level is relatively low, the dislocation motion may slow down. However, if the test temperature is sufficiently high, the other diffusional creep mechanism allows continuing the creep process. The diffusional creep results in the drastic increase in the strain rate observed at the tertiary phase, preceding the creep rupture (Skrzypek and Hetnarski, 1993).

2.2.3.2 Mechanisms of creep rupture

The effect of temperature and stress level on the mechanism of rupture is shown in Figure 2-6. A ductile rupture is preceded by a reduction of the cross-sectional area due to the large creep strains essentially caused by slip deformations within the grains. This effect results in the fracture mechanism via the propagation of cracks nucleated at the grain boundaries and spread inwards from the surface. The ductile rupture mechanism occurs at high stress levels and low temperature regimes (Skrzypek and Hetnarski, 1993).

A brittle rupture is caused by the deterioration of the material due to the formation of voids and the corresponding reduction of the effective cross-sectional area below a critical value. The brittle rupture mechanism occurs at low stress levels and high temperatures. The overall geometric effect is not observed, since creep strains are small (Skrzypek and Hetnarski, 1993).

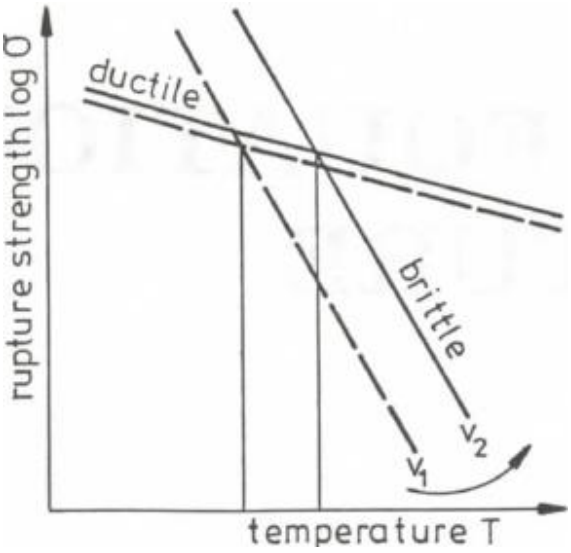

Figure 2-6 Typical effect of temperature and stress on mechanism of creep rupture (Skrzypek and Hetnarski, 1993)

2.2.4 Creep constitutive models

Higher operating temperature and stresses were adopted in chemical and petrochemical plants, power generation systems and so on; hence, the concern of strength design of components was moved to the viscoplastic performance of materials due to the prevention of creep failure. Different researchers proposed large amounts of creep constitutive equations.

The phenomenon of multi-axial creep behaviour is very close to classical plasticity; hence, classical plastic theory (CPT) was directly used in the multi-axial creep analysis during the first half of the 20th century. Taking into account the major factor of creep failure is the nucleation, growth and coalescence of cavities on the grain boundaries, see (Hayhurst and Leckie, 1984, Huddleston, 1985, Kassner and Hayes, 2003, Goodall and Skelton, 2004); the CPT-based model is limited since its derivation does not consider the physical damage factors.

Multi-axial creep design criteria using the models based on cavity growth mechanisms (CGM) was innovated by Hull and Rimmer (1959). The CGM-based models were improved by a lot of researchers from 1970s to 1980s, see (Rice and Tracey, 1969, Hayhurst, 1972, Gurson, 1977, Manjoine, 1975, Raj and Ashby, 1975, Ashby et al., 1978, Cocks and Ashby, 1980, Cocks and Ashby, 1981, Cocks and Ashby, 1982, Edward and Ashby, 1979, Cane, 2013, Cane, 1981, Cane, 1982, Tvergaard and Needleman, 1984). In recent years, further development has been made, see (Hales, 1994, Spindler, 2004,

Spindler, 1994, Kowalewski et al., 1994a, Spindler et al., 2001, Margolin et al., 1998, Ragab, 2002). Some of these models were applied in high temperature strength assessment procedures.

Kachanov (1958) initially developed a continuum damage mechanics (CDM) based model. The CDM-based model is developed on the phenomenological theory and mechanics theory contrasted with the CGM-based model. CDM-based method has been widely used during recent years due to the rapid development of modern computer technology and finite element analysis method, see (Othman et al., 1993, Hayhurst et al., 1994, Kowalewski et al., 1994a, Kowalewski et al., 1994b, Perrin and Hayhurst, 1996, Perrin and Hayhurst, 1999, Hyde et al., 1996, Hyde et al., 2004, Hyde et al., 2006, Xu, 2004, Xu, 2001, Ju-Shin and Gibbons, 1999, Jing et al., 2001a, Jing et al., 2003, Jing et al., 2001b). Three creep constitutive equations based on CDM-based model were selected in this research.

Creep and its deformation are widely applied in industry; for example, people predicted the residual lifetime of components through understanding the relationship of stress and strain. The availability of a procedure able to predict the residual life of plant devices is necessary to assist the management decisions about power plants' operation and maintenance scheduling. The major research approaches were reviewed to present some recent research situations.

2.3 Research Approaches of Creep Damage Analysis

Experimental and computational approaches are the major research approaches used to predict residual lifetime, and most time, people use them together. Some specific application examples of experimental and computational approaches are reviewed below.

Wang et al. (2014) proposed the creep test method of Babbitt and analysed the factors influencing the results of creep test according to the creep deformation of oil film bearing Babbitt in operation process. Based on this test, the creep characteristics of SnSb11Cu6 and SnSb8Cu4 were understood and the relationship between creep coefficients and stress has been obtained.

Yuan et al. (2014) proposed a modified method for attenuation coefficient calculation, and new parameters were calculated with inspecting signal acquired from creep specimen.

A uni-axial tension creep experiment with pure lead to verify the detecting ability of this new parameter has been conducted, and another creep inspecting experiment with P91 steel weldment was conducted to verify the practicability of the new parameter of a good distinguishing ability for different creep status. Moreover, Wang et al. (2014) verified the accuracy and reliability of creep deformation of Babbitt through finite element numerical simulation on the test specimens based on ANSYS, and carried out the creep characteristics of Babbitt of oil film bearing.

Zhang et al. (2015) carried out a uni-axial creep tensile tests using round bar specimens with a diameter of 10 mm at different stress levels at 566 °C. Moreover, Zhang et al. (2015) implemented the stress-regime dependent creep model and ductility in a ductility exhaustion based damage model, and analysed their influence on creep crack growth (CCG) behaviour of materials, and the CCG rate in a Cr–Mo–V steel over a wide range of C* was predicted by finite element analyses.

The experimental approach can display the whole process of creep evolution; however, it will spend a lot of financial resource and time for purchasing experiment equipment and awaiting experimental results. FEM as a computational approach greatly improved this defect, but it is seriously depending on the theoretical knowledge. A general computational tool does not exist; moreover, the traditional computational tools include the commercial software that need a user-defined subroutine, and the in-house software that has a narrow application scope.

2.4 Computational Tools of Creep Damage Analysis

The existing situation of FE packages for creep damage analysis has been reviewed by the author, see (Tan et al., 2012a). The creep analysis capability of commercial FE packages is unavailable. Users need to programme a user-defined subroutine to expand this capability. Moreover, some researchers chose to code their own in-house code to conduct the numerical simulation. A number of application examples of commercial software and in-house software are reviewed below.

2.4.1 Commercial Software

Becker et al. (2002) developed and used ABAQUS UMAT to implement benchmark tests against creep damage; they deemed creep rupture life could be predicted using continuum

damage mechanics. Geng et al. (2009) incorporated the modified KR creep damage constitutive equation into finite element program ABAQUS through its user subroutine to predict the creep damage and service life of a serviced steam pipeline made of 10CrMo910 heat resistance steel. Colombo et al. (1996) said that the creep analysis has been performed by the computer code ABAQUS and the damage evaluation has been carried out by means of in-house developed user's subroutine and post processor in his paper. Zhao et al. (2012) compiled the UMAT for ABAQUS to research factors affecting creep damage accumulation in ASME P92 steel welded joint is a typical example of coding user's subroutine.

Continuing damage mechanics has not been integrated in the commercial FE package. The primary and secondary creep stages can be analysed using the traditional plasticity theory, but the tertiary creep stage is unable to be simulated because the significant damage occurred. Users need to develop a user-defined subroutine, which is called UMAT to help ABAQUS for the analysis of creep damage. The commercial FE package can use a wide range of element types, material models and other facilities; for example, the efficient equation solvers that are not normally available in in-house FE package. Moreover, the commercial FE package does not currently allow the removal of failed elements from the boundary-value problem during the solution process.

2.4.2 In-house Software

DAMAGE XX (Hall, 1990) is an early in- house creep damage analysis solver developed at The University of Manchester Institute of Science and Technology (UMIST). This software is written based on Fortran 77, and the failed elements could be removed from the stiffness matrix. Runge-Kutta-Merson (RKM) method is applied. It is an important tool for the production of significant publications; these publications are not listed here for brevity. FEMGV is used as the pre- and post-processor.

DAMAGE XXX (Hayhurst, 2006) is a new advanced version of DAMAGE XX, which was also developed at UMIST. This package is not only added a 3D FEA function, but was also applied on a parallel computer. Hayhurst (2006) reported its validation via analysing creep failure in the heat affected zone (HAZ) region of Cr-Mo-V cross-welds and its application. FEMGV is also used as the pre- and post-processor.

FE-DAMAGE (Becker et al., 2002) is another in-house code developed at University of Nottingham. It has been used in research; for example, Becker et al. (2002) used the FE-DAMAGE program to present four different creep damage results in his paper. It is interesting to note that those results were compared with the result produced by ABAQUS for benchmarking. FEMGV is used as the pre- and post-processor.

HT∑ is an in-house FE package used for creep damage analysis. Ling et al. (2000) developed a subroutine based on RKM method for this programme. As an in-house code, the accuracy and efficiency of this programme had been proved through a thick cylinder problem.

An in-house code was developed at Tsing Hua University (Wang and Wang, 1996). The Euler's method has been used in this code. A critical time-step which small enough was used to reduce the errors caused by highly non-linear behaviour.

Haigihara and Miyazaki (2008) developed an in-house code, which used FEA for creep failure of coolant pipe in light water reactor due to local heating under severs accident condition. This result was also compared with the result of ABAQUS in this paper.

Commercial software and in-house software have their own advantages in practical applications. The commercial software has rich resource support such as element type, material parameters and advanced topological technique; however, it does not allow the failure element to be removed during the solving processing. Moreover, the required user-defined subroutine is also difficult to be programmed. The in-house software looks rudimentary, but the capability of removable failure element is allowed. Moreover, to programme in-house software is easier than to code a user-defined subroutine for commercial FE package. On the above observation, the author prefers and advocates the approach of developing in-house FE software.

2.5 Programming of Finite Element Method

In order to develop an in-house FEA software, some basic problems should be cleared firstly. The specific working targets need to be understood; the supervisor allocated the initial workloads directly, but the arrangement was modified in the following research. The background, basic equilibrium equations, programming language of developed in-

house software were reviewed below. A number of general-purpose subroutines were adopted in order to build this in-house software.

2.5.1 HITSI

Dr. Qiang Xu has established his own research group since Oct. 2011. The research projects include the development of in-house FEA software and the development of creep constitutive equations. Such in-house FEA software is the original conception of HITSI, and a primary literature review was addressed by Tan et al. (2012a).

The initial design scheme only focusses on the development of the solver, and the work of pre- and post-processing was arranged to the purchased software FEMGV; hence, in the earliest stage, the name of HITSI belongs to the solver. At the same time, the data transfer between the solver and FEMGV was considered which is to develop a text process programme. Later, the HITSI is defined a complete system include pre- and post-processor, the solver and data transfer interface, Figure 2-7 indicates processing sequence of HITSI.

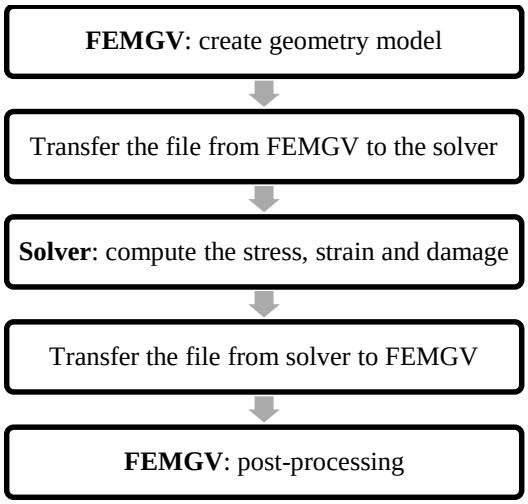

Figure 2-7 Operation process of HITSI

Due to the workload of programming, Xu allocated the task to two PhD candidates; the author's colleague Dezheng Liu undertakes the design and development of overall structure of the solver, and the author undertakes the development of subroutines and data transfer tool. The author has relative programming experience of data transfer tool because of his undergraduate final project.

In the beginning, the research was proposed based on a large amount of general purpose subroutines; for example, NAG (2009) subroutine library provides various type subroutine of numerical method, and Smith and Griffiths (2004) offer subroutines to handle global stiffness matrix, nodal freedom array and elastic stress-strain matrix. Following the pace of progress of the research, the situation has changed. The author found that it is not worth paying more attention to couple NAG subroutine library[3] and other general-purpose subroutines. At the same time, the author was suggested to promote the small data transfer tool.

2.5.2 Finite element method

The operation flow of the solver was proposed by Liu et al. (2012a), and the equilibrium equation of FEM for the solver can be expressed as:

$$U = K^{-1}(F_e + F_c) \qquad (2\text{-}4)$$

where U is nodal displacement; K is global stiffness matrix; F_e and F_c are nodal elastic force and nodal creep force respectively. Initially, the f_c is zero, and strain can be derived by (2-5);

$$\epsilon = BU \qquad (2\text{-}5)$$

where ϵ is total strain, B is strain-displacement matrix, and the elastic strain is one part of total strain;

$$\epsilon_e = \epsilon - \epsilon_c \qquad (2\text{-}6)$$

where ϵ_e, ϵ_c are elastic strain and plastic strain respectively, the elastic stress is;

$$\sigma_e = D\epsilon_e \qquad (2\text{-}7)$$

where σ_e is elastic stress, D is stress-strain matrix. Here, the creep strain was produced by creep constitutive equations $f(\sigma_e, \Delta_t)$;

[3] Numerical method subroutine library, which allowed users to define their own ordinary differential equations; however, it does not suit the author's research due to the material parameters have to be set as constants.

$$\epsilon_c = f(\sigma_e, \Delta_t) \quad (2\text{-}8)$$

where Δ_t is time interval; the virtual creep stress can be derived by (2-9)

$$\sigma_c = D\epsilon_c \quad (2\text{-}9)$$

where σ_c is creep stress, and finally, the nodal creep force equal to:

$$F_c = B^T\sigma_c \quad (2\text{-}10)$$

Based on the discussion above, the tasks of this research are clarified.

2.5.3 Programming Languages

Fortran is widely used for scientific/numerical computing purposes, and it is only used for such requirements. For example, it is still used for such tasks in embedded programming things like aircraft controllers, chemical plants (Sebesta, 2010). Compared with C++, Python, Matlab and S--Plus etc., which are all often used for scientific computing, a comparison of executive can be summarised as (Kupferschmid, 2009):

- Ease of use: Python, Matlab, then Fortran
- Debuggability: Python, Matlab, then NAG Fortran
- Portability: Fortran is the best , and over-elaborate C++ is the worst
- Software engineering: Fortran is the best, then Python
- Performance: Fortran or C++ (no overall difference), then Python and Matlab
- Parallelism: Fortran is the best for shared memory, few times be chosen for distributed memory
- Array handling: Fortran is the best, then Matlab and Python
- Text handling: Python is the best, then C++, then Fortran
- 'Computer science' : C++, then Python and Fortran
- 'System interfaces' : Pytho , then C++, then Fortran

Fortran 2003 is adopted in this research because the existing general-purpose subroutines, which are adopted by Dezheng Liu, were written by Fortran. It avoids the need to rewrite those adopted subroutines because Fortran 2003 is compatible with previous version even Fortran 66. This research involves a lot of computation of matrices; hence, using Fortran is easy to programme.

2.5.4 General purpose subroutines of HITSI

The development work of FEA software is really complex and difficult; however, a lot of existing theories and subroutines make this problem much easier than decades ago. Smith and Griffiths (2004) offered general purpose subroutines and functions held in a library called **new_library**. Some general-purpose subroutines and functions from **new_library** were adopted in a new library, which is called **HITSI_library**, in the development process of the solver. Table 2-1 indicates the subroutines and functions in alphabetical order, together with the meaning of their arguments. Arguments in bold are those returned by the subroutine.

Table 2-1 Description subset of new_library (Smith and Griffiths, 2004)

Name	Arguments	Description
bacsub	bk, **loads**	Return the complete Gaussian back substitution on displacement array **loads** from global stiffness matrix bk.
bandwidth	g	Function returns the maximum bandwidth for an element with steering vector g.
banred	**bk**, n	Return the Gaussian reduction on global stiffness matrix **bk** itself from number of degrees of freedom in the mesh n.
beemat	**bee**, deriv	Returns **bee** matrix for shape function derivatives derive.
bmataxi	**bee**, **radius**, coord, deriv, fun	Returns **bee** matrix and **radius** from element nodal coordinates coord, shape function derivatives derive and shape function fun.
deemat	**dee**, e, v	Returns elastic stress-strain **dee** matrix in 2D (plane strain) or 3D. e and v are Young's modulus and Poisson's ratio.
determinant	jac	Function returns the determinant of 2D or 3D square matrix jac.
formkv	**bk**, km, g, n	Returns global stiffness matrix **bk** from element stiffness matrix km and number of degrees of freedom in the mesh n. g is element steering vector.
formnf	**nf**	Returns nodal freedom array **nf** from boundary conditions input of 0 s and 1 s.
invert	**matrix**	Return the inverse of a small matrix called **matrix** onto itself.
num_to_g	num, nf, **g**	Returns the element steering vector **g** from the element node numbering num and the nodal freedom array nf.
sample	element, **s**, **wt**	Returns the local coordinates **s** and weighting coefficients **wt** for numerical integration of a finite element of type element.
shape_der	**der**, points, i	Returns the shape function derivatives **der** at the i^{th} integration point. points holds the local coordinates of the integration points.
shape_fun	**fun**, points, i	Returns the shape function **fun** at the i^{th} integration point. points holds the local coordinates of the integration points.

2.6 Summary

The present situation of high temperature industry has been reviewed, which the safety assessment and residual lifetime prediction are issues in high temperature devices, and creep is one of the major failure factors. General knowledge of creep such as creep deformation, creep failure mechanisms and creep constitutive models has been reviewed.

Experimental and computational approaches have their own advantages, but this research focusses on the computational approach. The present situation of computational tools has been reviewed, where multifarious computational tools were used due to the general finite element software could not be employed directly. The way to develop a universal finite element software has been proposed and the adopted programming method and general-purpose subroutine have been reviewed.

3 DEVELOPMENT STRATEGY OF HIGH TEMPERATURE STRUCTURAL INTEGRITY-CREEP

This chapter explains the development strategy of HITSI, which includes structure analysis, function definition and the development methodology. The structure analysis involves the understanding of overall structure of HITSI, and includes the understanding of overall algorithm of the solver; it aims to identify the specific research sub-tasks of this research. Based on the structure analysis, the type, function and compatibility of required subroutines and programmes will be discussed. The unique development methodology is introduced, which includes the stand-alone work of the author and the co-operation work with the author's colleague Dezheng Liu.

3.1 Overall structure of HITSI

HITSI includes four components, which are a) the solver, b) pre- and post-processor, c) data transfer programme and d) nodal loads calculator. Figure 3-1 presents the general operation model of HITSI, which could identify that the author's development work has two branches. The first branch is the development of independent programmes which are used to connect the solver and pre- and post-processor. The second branch is the development of dependent subroutines, which are used to realize the capability of creep

analysis of the solver. FEMGV was chosen to be the pre- and post-processor, which is shown in the Figure 3-1. The detailed responsibilities of each component are introduced separately.

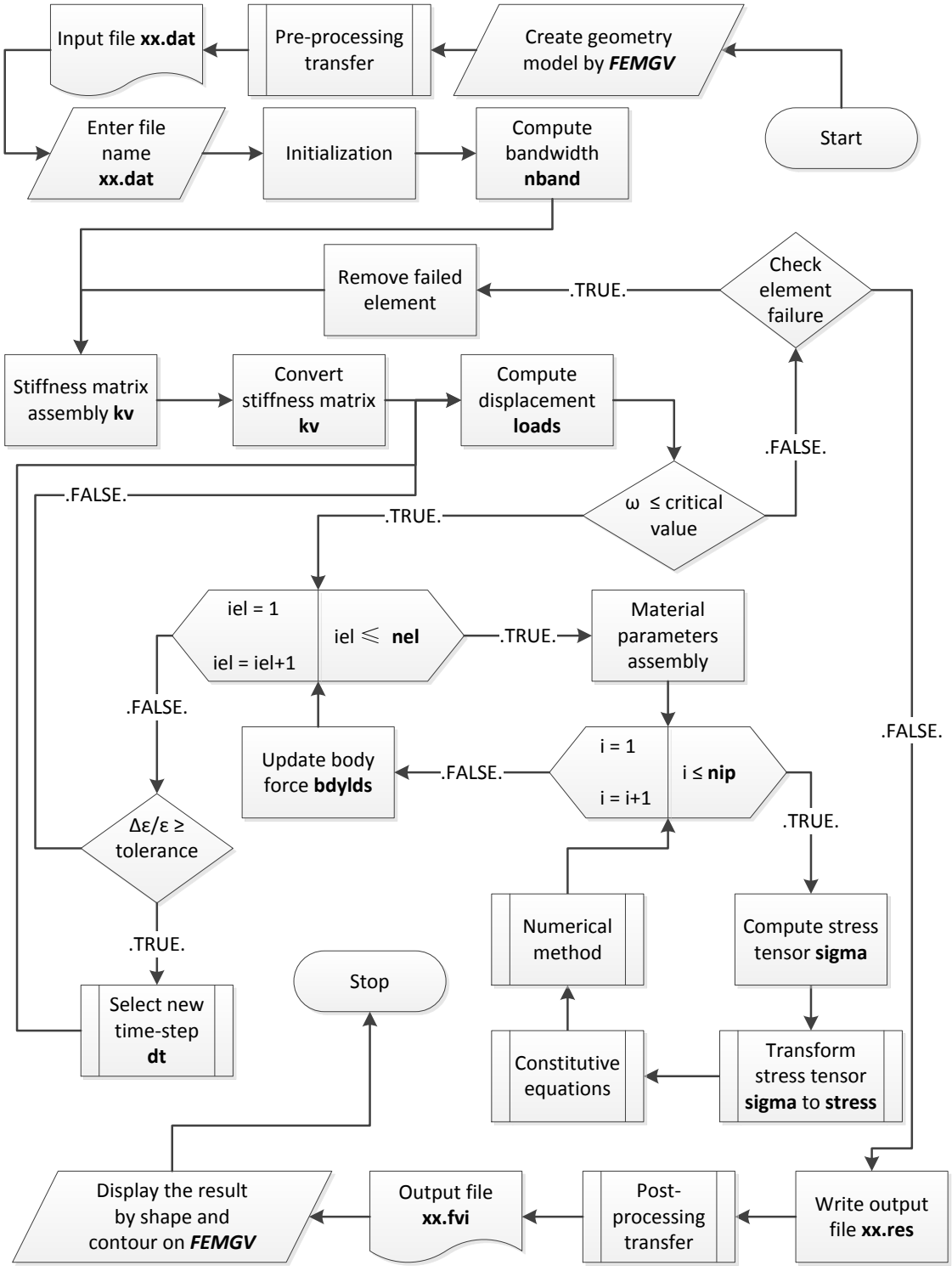

Figure 3-1 Overall structure of HITSI

3.1.1 The solver

The solver is the core of HITSI, which undertakes the major analysis work of creep research. It is an elastic-creep finite element method programme, which is developed by the author and his colleague Dezheng Liu. Dezheng Liu et al. (2012b, 2012a, 2013d, 2013c, 2013a, 2013e, 2013b) reviewed the programming knowledge of finite element method; conducted the overall algorithm design of the solver; and validated the solver.

Liu is the major developer who undertakes the work of overall algorithm design, coding and validation of the solver, in which case, the author undertakes the development work of subroutines. The solver was coded based on a number of existing general-purpose subroutines; thus, the author is required to not only develop the subroutines, which are needed but could not be found from literatures directly, but also keep the compatibility of the developed subroutines and those general purpose subroutines. All subroutines were developed based on the Fortran language.

3.1.2 Pre- and Post-processor

The solver is a programme without any geometrical and topological capability; hence, obtaining geometric models and visualized results is required. The solver has to work with a pre- and post-processor together in practical case, and FEMGV was selected to be the pre- and post-processor in this research. FEMGV (Manie and Wolthers, 2013) is a professional pre- and post-processor, which works in conjunction with two databases which both comprise an index file and a data file in binary format. One database applies for the Design environment and another one for the Results environment. The contents of both databases can also be represented as a so-called ‘Neutral file’ in ASCII text format.

As commercial software, unfortunately, FEMGV will not offer a specialized interface for the solver; therefore, formatted input and output of FEMGV should be researched due to the gap between FEMGV and HITSI (Tan et al., 2012b). The file format of Neutral file is very complex; hence, the understanding of formatted input and output was independent in order to reduce the development workload of the solver.

3.1.3 Data transfer interface

Due to a specialized interface not being offered by FEMGV, a data transfer programme was developed to fill this gap. The structures of both Neutral files based on the design environment and result environment are essentially the same but the contents of included data may vary by their very nature (Manie and Wolthers, 2013). For instance the Neutral file based on the design environment could not contain analysis results data which appears in the database (and the Neutral file) based on the results environment only; therefore, the data transfer processes of pre-processing transfer and post-processing transfer should be distinguished through two different branches.

In the pre-processing transfer, some problems could not be solved based on the present situation, which leads to the data transfer interface not being integrated into the solver. For example, a number of parameters used to describe the characteristics of creep could be defined through FEMGV directly; hence, these parameters have to be entered into the input file of the solver artificially. Moreover, the allocation of a concentrated force to each loaded node could not be supported by FEMGV, and this capability could not be realized yet through the data transfer interface itself due to this problem involving advanced topological algorithms.

In the post-processing transfer, a number of control parameters were defined by FEMGV; however, the solver will not produce these parameters directly because these are parameters depending on FEMGV only, but not involving any activities of creep damage analysis. Identification of the value of each control parameter is a challenge due to these parameters being required to not only to be analysed qualitatively based on theoretical knowledge, but also to be tested through real data.

3.1.4 Nodal loads calculator

In the setting of boundary conditions, a concentrated force should be represented by the equivalent nodal loads because of the nature of the solver (Liu et al., 2012a). The allocation of a concentrated force to each loaded node is a problem, which could not be solved by FEMGV or the data transfer programme; thus, a small calculator was proposed to obtain the converted value of each nodal load based on a concentrated force. Figure 3-1 does not display the position of nodal loads calculator since it is included in the process of

pre-processing transfer. Actually, the nodal loads calculator is used to avoid hand-calculation only.

3.2 Required subroutines of the solver

Figure 3-1 also presents the overall algorithm of the solver. The solver starts with parameter initialization such as the allocation of node coordinates, element definitions and constraint information. All of the work such as bandwidth calculation, stiffness matrix assembly and displacement calculation could be solved by those general-purpose subroutines; however, some works such as stress tensor transformation, constitutive equations integration and time-step control need to be satisfied through a number of self-developed subroutines.

The general-purpose subroutines make programme development much easier, but they could not cover everything. For example, a class of constitutive equations subroutines should be developed to return the creep strain and creep damage; however, the constitutive equations subroutine could not exist independently due to the solving process being dependent on numerical methods. Moreover, the stress tensor could not be applied to constitutive equations directly, which is also a significant problem.

It could be clearly seen from Figure 3-1; the time-step control method based on self-adaptive technique was used in the solver to enhance the accuracy of integration. Furthermore, a mathematical approach called normalization was considered to improve the accuracy and efficiency of constitutive equations. The detailed responsibilities of each subroutine were introduced separately.

3.2.1 Constitutive equations subroutine

Creep constitutive equations undertake the task to return creep strain and damage in the overall operation of the solver. The creep strain is used for the redistribution of elastic stress; and the creep damage is used for the determination of failure element. Generally, creep constitutive equations are ordinary differential equations based on the initial-value problem; hence, the numerical methods such as Euler's method, Runge-Kutta (RK) method were required to find its estimated solutions. Furthermore, complex description of stress state was introduced into the constitutive equations such as deviatoric stress tensor, principal stress and equivalent stress.

In the solver, the computing of stress and strain of each integration point and element was conducted one by one; hence, the arguments used to hold creep strain and damage should be stated as a three-dimensional deferred-shape real array which contains the data of strain and damage components, integration points and elements. Moreover, due to the factor of temperature is normally integrated into the material properties, the argument used to describe the temperature will be included in the array that is used to hold material properties. This research only includes three types of constitutive equations subroutines, which are Kachanov-Rabotnov (KR) equations, Perrin-Hayhurst (PH) equations and Qiang Xu's (QX) equations. KR equations are the typical version in creep damage mechanics, and many researchers developed their own constitutive equations based on this version. Compared with KR equations, PH equations were introduced more damage factors. It is also a widely used creep constitutive equations. QX equations are new version of PH equations, which the stress state functions have been reconsidered.

3.2.2 Numerical method subroutine

The solutions to the creep constitutive equations itself are creep strain rate and creep damage rate. Obtaining creep strain and damage is dependents on the numerical methods. A class of numerical method subroutines undertake the work to find estimated creep strain and damage according to a specific time-step. The time-step will affect the result based on its size due to all solutions produced by the numerical method being not exact solutions. Explicit-shape arrays were recommended in order to avoid the mismatch of the transmission between constitutive equations subroutine and numerical method subroutine because the constitutive equations subroutine is executed inside the numerical method subroutine.

The major arguments statement of the numerical method subroutine should follow the constitutive equations subroutine. Four single-step, explicit numerical methods, which are Euler's method, classical 4^{th} order Runge-Kutta (RK4) method, RKM method and RKF method, were selected in this research. The Euler's method is the easiest integration method. The family of RK methods are typical integration methods that be used for ordinary differential equations, and they are developed based on the Euler's method. RK4 method is the typical version of RK method family. RKM method is the classical version, which is already used in creep damage analysis area. RKF method is the embedded version that has more advantages of precision.

3.2.3 Stress tensor transformation subroutine

An equivalent creep strain was allocated to each coordinate through the deviatoric stress tensor components; this is the derivation of multi-axial constitutive equations. On the other hand, principal stress, equivalent stress and some self-defined stress were used to describe the stress state of creep deformation; hence, the understanding of those stresses is significant.

Deviatoric stress tensor, principal stress or equivalent stress could be obtained via the stress tensor, and may be produced through some separate existing subroutines; however, the efficiency will not be guaranteed because those subroutines have single and discrete features. A modular processed concept of those stresses was proposed in order to not only reduce the memory requirements of the solver, but also enhance the computing efficiency of the solver.

A subroutine, which is used to return the values of deviatoric stress tensor, principal stress and equivalent stress according to the present stress tensor components, is the stress transformation subroutine. Its operation process could be divided into three stages that in the first stage, stress tensor components were allocated to the local arguments; secondly, target functions were computed; finally, the results were updated into a global array. In here, deviatoric stress tensor, principal stress and equivalent stress will be obtained in sequence. This subroutine was designed based on three-dimensional stress system; hence, the two-dimensional stress system was realized through reducing the number of stress tensor components[4].

3.2.4 Time-step control procedure subroutine

Creep deformation of metals is a time-depending problem; moreover, the time-step dependence of the numerical method is high. Hence, the time-step control is extremely important in practical computational analysis. The time-step control procedure was divided into two parts, which are time-step acceptance and time-step selection. The time-step acceptance was integrated into the subroutine of numerical method due to it being the self-adaptive technique of numerical method; and, the time-step selection was coded as an independent subroutine. Figure 3-1 shows the overall time-step control method.

[4] It means τ_{yz}, τ_{zx} will be forced to zero in two-dimensional stress system.

3.2.5 Normalization technique subroutine

The normalization technique is used to promote the computing capability through reducing the stress and strain proportionally. A normalization scheme, which included the normalization of equilibrium equation of finite element method and the normalization of creep constitutive equations, was firstly proposed by Hayhurst et al. (1984). The normalization procedure improved the accuracy of numerical solution due to reducing the effect of numerical rounding errors (Hall, 1990). The advantages were not mentioned clearly in those papers; however, it is still worth researching the potential power of normalization in creep damage area even though nowadays the computational power has a huge improvement. The normalization subroutine is a special constitutive equation.

3.3 Research methodology

The definition and application of each research sub-task were given respectively to deeply understand how to organise the required programmes and subroutines. A unique research methodology was adopted depending on our research environment. Software development life cycle as the general software development methodology was used in this research.

3.3.1 Software development life cycle and activities

The software development life cycle consists of a number of distinct work phases, and each of those distinct work phases was called software development process or activity. Wallis (1985) identified what might now be termed the ‘traditional’ software development life cycle: requirements analysis, functional specification, design, coding, testing and maintenance. Almost one and half decades later, Jacobson et al. (1999) addressed their core workflow of the unified software development process: requirements, analysis, design, implementation and test. It is obviously; those two definitions did not have essential difference except the process of maintenance.

3.3.2 Software development model

Bell (2000) introduced a series of development models such as 1) seat-of-the-pants, do-it-yourself, or ad hoc, 2) waterfall, 3) prototyping, 4) formal methods and 5) spiral. The waterfall model is a sequential development approach that looks like a waterfall, and its typical form can be presented by Figure 3-2. Prototyping is a development approach, which builds a programme through repeated updating the prototype, and the first prototype is an incomplete version of the developed programme. However, prototyping is

not only a standalone development methodology, but also a selected parts of a larger development methodology such as incremental and spiral. The spiral model guides a team to adopt elements of one or more process models such as incremental, waterfall or evolutionary prototyping, and its typical model could be seen Figure 3-3. Spiral combines some key aspects of the waterfall model and the prototyping model in order to enhance the advantages of top-down and bottom-up concepts.

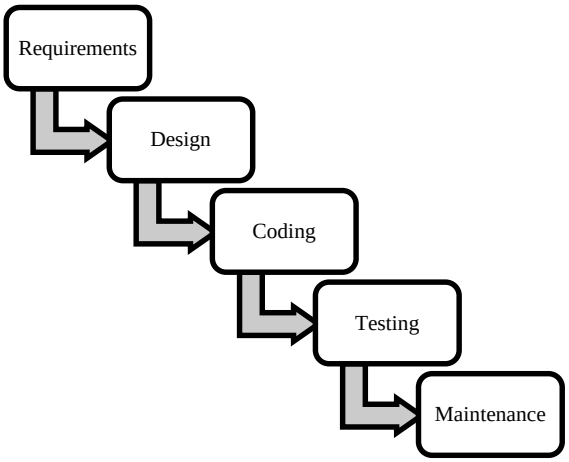

Figure 3-2 Typical activities of waterfall model

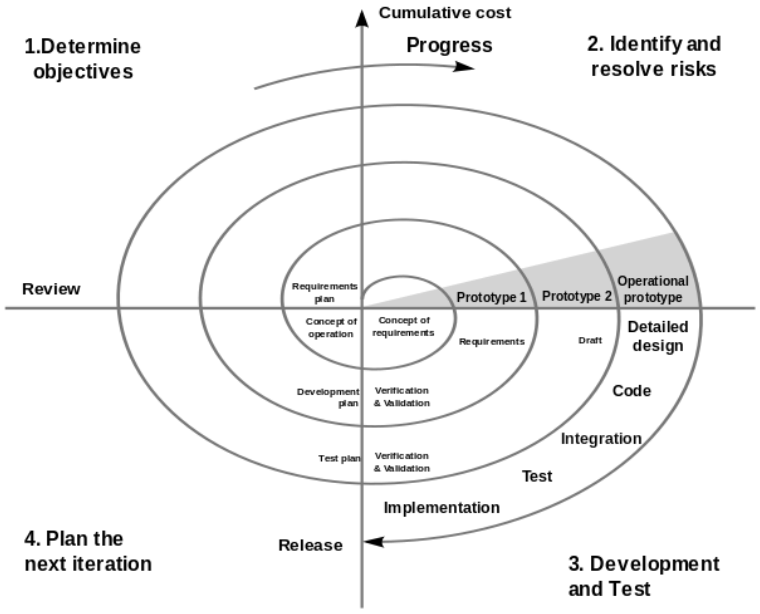

Figure 3-3 Typical activities of spiral model

This research adopted the spiral model due to its unique nature. The first prototype[5] of the solver is an elastic analysis programme, which is built through general-purpose

[5] In this thesis, this prototype named Prototype I.

subroutines. The programmer of Prototype I is my colleague Dezheng Liu, and his work offers a number of key conditions for this research. For example, a) Prototype I offered a set of unified input and output format, and this form will be used to design data transfer interface; b) Prototype I defined the data type of some key arguments such as stress and strain, and these data types will affect the framework of those subroutines developed by this research. According to the Prototype I, the author developed the constitutive equations subroutine, numerical method subroutine, stress tensor transformation subroutine, time-step control subroutine and normalization technique subroutine. These subroutines will be delivered to Prototype I to create the second Prototype[6]. Dezheng Liu will use prototype II to test the elastic-creep analysis capability in plane stress case, plane strain case and axisymmetric case.

Based on the feedback of Prototype II, data transfer interface and the author developed nodal loads calculator, and those subroutines will be optimized as well. Finally, all of these works will be delivered to Dezheng Liu to test the final version via a weldment case (Liu, 2015).

3.3.3 Requirements analysis

General requirements analysis of software development is determining the conditions to satisfy the new subroutines and programmes, and is taking account of the possibly conflicting requirements of the various stakeholders. According to the proposed flowchart of HITSI, a number of theories and knowledge need to be collected, understood and researched in order to develop relative subroutines and programmes. This work will be reported in detail in the Chapter 4. Liu (2015) gave feedback when he complete the Prototype I. The input and output file format, arguments definition and overall structure of Prototype I has been confirmed through his feedback.

3.3.4 Algorithm Design

The algorithms were designed based on the nature of theoretical knowledge and the position on the solver. The theoretical knowledge should be studied, and be analysed in order to identify those key arguments. A flowchart will be sketched to display the overall algorithm clearly. The structure will be presented through pseudo code.

[6] In this thesis, this prototype named Prototype II.

3.3.5 Testing and verification

The quality assessment of a software product includes either dynamic verification, i.e. testing, or static verification, i.e. review and inspections. The following issues must be highlighted in the validation of software (Olsen et al., 2001):

- Correct identification and expression of user requirements
- Correct implementation of the specified requirements
- Absence of problems with the code and the data
- Usability, completeness and level of updating of the documentation given to customers
- Maintainability of the product

The validation approach of this research contains both static validation and dynamic validation.

3.4 Summary

The overall structure and algorithm of HITSI has been identified. Spiral model and waterfall model will be used in this research. The sub-tasks fulfilled by this research could be summarised below:

- Data transfer interface, which used to fill the gap caused by formatted input and output of two different software;
- Nodal loads calculator, which is proposed to be the temporary solution of nodal loads arrangement module;
- Constitutive equations subroutine, which is a subroutine class used to return creep strain and damage for the main programme of the solver; moreover, it could enable the potential users to add their own constitutive equations directly;
- Numerical method subroutine, which is a subroutine class used to assist constitutive equations subroutine;
- Stress tensor transformation, which is used to obtain deviatoric stress tensor, principal stress and equivalent stress;
- Time-step control subroutine, which is used to assist numerical method subroutine to implement self-adaptive approach;

- Normalization subroutine, which is a special constitutive equations subroutine used to enhance the accuracy and efficiency of numerical integration.

4 RELATIVE THEORIES AND KNOWLEDGE APPLICATION

This chapter studies and analyses the relative theories and knowledge used to develop the required subroutines and programmes. These theories involve constitutive theory of creep deformation, numerical methods of engineering, classical plastic theory and computing science. Based on the definition of required subroutine/programme of this research, the knowledge could be specific to a) constitutive equations based on continuum damage mechanics; b) derivation of deviatoric stress tensor, principal stress and equivalent stress based on stress tensor; c) single-step explicit numerical methods of initial-value problem; d) self-adaptive approach of time-step and its control procedure; e) normalization technique; f) boundary conditions of nodal loads based on planar and axisymmetric element type; and g) formatted input and output of FEMGV and the solver.

4.1 Constitutive Equations

Knowledge of constitutive equations was used to develop the subroutine of constitutive equations. The equations itself and the definitions of dependent variables, independent variables and constants were introduced specifically in order to design arguments of this class of subroutine. Based on the analysis of constitutive equations, the input conditions of constitutive equations could be summarised as a) stress, b) material properties and c) initial creep strain and damage. Only stress needs to be discussed because material properties, initial creep strain and damage are constants.

A set of ordinary differential equations, which are used to describe the constitutive relation between material and deformation, is known as creep constitutive equations. The metal creep research has been researched almost since 100 years ago, Yao et al. (2007) reviewed a large amount of existing creep constitutive equations, and suggested such equations can be grouped into three categories, i.e. 1) CPT-based approach, 2) CGM-based approach, and 3) CDM-based approach.

In this research, three sets of constitutive equations, KR constitutive equations, PH constitutive equations and QX constitutive equations, which based on CDM, were selected to develop the class of constitutive equations subroutine.

4.1.1 Kachanov-Rabotnov constitutive equations

Based on the classical plasticity theory, Kachanov (1999) introduced a new concept named damage into creep research in 1958, and firstly proposed a uni-axial form of his creep damage constitutive equations. The modified form according to the theories of Rabotnov and Andrade was addressed by Hayhurst et al. (1984). The uni-axial form of modified equations is expressed as:

$$\dot{\epsilon} = A\left(\frac{\sigma}{1-\omega}\right)^{n} t^{m} \qquad (4\text{-}1)$$

$$\dot{\omega} = B\frac{\sigma^{\chi}}{(1-\omega)^{\phi}} t^{m} \qquad (4\text{-}2)$$

where $\dot{\epsilon}, \dot{\omega}$ are creep strain rate and creep damage rate respectively; A, B, n, m, χ, ϕ are creep material properties; σ is stress; and t is time function.

Odqivst (1974) proposed the multi-axial stress state form of Norton's equations, and (4-1) and (4-2) were revised by Hayhurst et al. (1984) according to the theory of Odqivst. The multi-axial form of equations is expressed as:

$$\dot{\epsilon_{ij}} = \frac{3S_{ij}}{2\sigma_e} A\left(\frac{\sigma_e}{1-\omega}\right)^{n} t^{m} \qquad (4\text{-}3)$$

$$\dot{\omega} = B\frac{\sigma_r{}^{\chi}}{(1-\omega)^{\phi}} t^{m} \qquad (4\text{-}4)$$

$$\sigma_r = \alpha\sigma_1 + (1-\alpha)\sigma_e \qquad (4\text{-}5)$$

where S_{ij} is deviatoric stress tensor; σ_e is effective stress which is equal to Von Mises stress; σ_r is rupture stress which represents stress state function; σ_1 is the first principal stress; and α is coefficient of stress state function.

4.1.2 Perrin-Hayhurst constitutive equations

Perrin and Hayhurst (1996) suggested a set of constitutive equations to describe the creep constitutive relation of a 0.5Cr-0.5Mo-0.25V ferritic steel over the temperature range 600-675℃. Three types of creep damage variables, i.e. cavitation damage (Cane, 1981), carbide precipitates (Othman et al., 1993) and primary creep (Kowalewski et al., 1994a) were introduced into this constitutive equations. The uni-axial form of his constitutive equations is expressed as:

$$\dot{\epsilon} = \mathrm{A}\sinh\left[\frac{\mathrm{B}\sigma(1-\mathrm{H})}{(1-\phi)(1-\omega)}\right] \quad (4\text{-}6)$$

$$\dot{\mathrm{H}} = \frac{\mathrm{h}}{\sigma}\left(1-\frac{\mathrm{H}}{\mathrm{H}^*}\right)\dot{\epsilon} \quad (4\text{-}7)$$

$$\dot{\phi} = \left(\frac{\mathrm{K_c}}{3}\right)(1-\phi)^4 \quad (4\text{-}8)$$

$$\dot{\omega} = \mathrm{C}\dot{\epsilon} \quad (4\text{-}9)$$

where $\dot{\epsilon}, \dot{\mathrm{H}}, \dot{\phi}, \dot{\omega}$ are creep strain rate and variables of damage rate respectively; $\mathrm{A}, \mathrm{B}, \mathrm{C}, \mathrm{h}, \mathrm{H}^*, \mathrm{K_c}$ are creep material properties; and σ is stress.

It was expanded to multi-axial form by Perrin and Hayhurst (1996):

$$\dot{\epsilon_{ij}} = \frac{3\mathrm{S_{ij}}}{2\sigma_e}\mathrm{A}\sinh\left[\frac{\mathrm{B}\sigma_e(1-\mathrm{H})}{(1-\phi)(1-\omega)}\right] \quad (4\text{-}10)$$

$$\dot{\mathrm{H}} = \left(\frac{\mathrm{h}\dot{\epsilon_e}}{\sigma_e}\right)\left(1-\frac{\mathrm{H}}{\mathrm{H}^*}\right) \quad (4\text{-}11)$$

$$\dot{\phi} = \left(\frac{\mathrm{K_c}}{3}\right)(1-\phi)^4 \quad (4\text{-}12)$$

$$\dot{\omega} = \mathrm{CN}\dot{\epsilon_e}\left(\frac{\sigma_1}{\sigma_e}\right)^{\mathrm{v}} \quad (4\text{-}13)$$

where S_{ij} is deviatoric stress tensor; σ_e is effective stress which equal to Von Mises stress; σ_1 is the first principal stress; $\dot{\epsilon_e}$ is equivalent strain rate; and N = 0, if $\sigma_1 < 0$, else N =1.

4.1.3 Qiang Xu's constitutive equations

Xu (2001, 2004) modified the stress state functions of PH equations. Based on his investigations and considerations, the researches of Huddleston (1993) and Spindler (2004) were introduced into his research. The uni-axial form of his equations is the same with PH equations, and the multi-axial form of his equations is expressed as:

$$\dot{\epsilon}_{ij} = \frac{3S_{ij}}{2\sigma_e}\mathrm{Asinh}\left[\frac{B\sigma_e(1-H)}{(1-\phi)(1-\omega_d)}\right] \quad (4\text{-}14)$$

$$\dot{H} = \left(\frac{h\dot{\epsilon}_e}{\sigma_e}\right)\left(1-\frac{H}{H^*}\right) \quad (4\text{-}15)$$

$$\dot{\phi} = \left(\frac{K_c}{3}\right)(1-\phi)^4 \quad (4\text{-}16)$$

$$\dot{\omega} = CN\dot{\epsilon}_e\left\{\exp\left[p\left(1-\frac{\sigma_1}{\sigma_e}\right)+q\left(\frac{1}{2}-\frac{3\sigma_m}{2\sigma_e}\right)\right]\right\}^{-1} \quad (4\text{-}17)$$

$$\dot{\omega_d} = \dot{\omega}\left(\frac{2\sigma_e}{3S_1}\right)^a \exp\left[b\left(\frac{3\sigma_m}{S_s}-1\right)\right] \quad (4\text{-}18)$$

$$\sigma_m = \frac{1}{3}(\sigma_1+\sigma_2+\sigma_3) \quad (4\text{-}19)$$

$$\sigma_m = \sigma_1 - \sigma_m \quad (4\text{-}20)$$

$$\sigma_m = \sqrt{{\sigma_1}^2+{\sigma_2}^2+{\sigma_3}^2} \quad (4\text{-}21)$$

where $\dot{\epsilon}_{ij}, \dot{H}, \dot{\phi}, \dot{\omega}, \dot{\omega_d}$ are creep strain rate and variables of damage rate respectively; A, B, C, h, H^*, K_c are creep material properties; S_{ij} is deviatoric stress tensor; σ_e is effective stress which equal to Von Mises stress; $\sigma_1, \sigma_2, \sigma_3$ are principal stress; σ_m is hydrostatic stress; $\dot{\epsilon}_e$ is equivalent strain rate; N = 0, if $\sigma_1 < 0$, else N =1; and a, b, p, q are stress state index.

Based on these discussions, creep constitutive equations could be considered simply to consist of derivatives, solutions, stress and coefficients. It might be appropriate to consider creep strain rate and damage rate as the derivative, to consider creep strain and damage as the solution, to consider deviatoric stress tensor, principal stress and equivalent stress as the stress, and to consider material properties as the coefficients.

4.2 Transformation of Stress Tensor

A series of stresses are involved in the operation of constitutive equations such as deviatoric stress tensor, principal stress and equivalent stress; furthermore, some self-defining stress states were also applied depending on the specific creep constitutive equations. For programming convenience, an independent subroutine with deviatoric stress tensor, principal stress and equivalent stress should be developed, and such self-defining stress states will be put in their corresponding subroutine of constitutive equations.

The concept of stress and stress tensor can be found in many books, see (Richards, 2001, Gere and Goodno, 2009); however, the challenge is how to arrange the computing order of each in order to save computing power. A appropriate sequence of computing was suggested by Chen (2007), and detailed formulas is expressed below.

4.2.1 Deviatoric stress tensor

A stress called hydrostatic stress is simply the average of the three normal stresses:

$$\sigma_0 = \frac{1}{3}\left(\sigma_x + \sigma_y + \sigma_z\right) \qquad (4\text{-}22)$$

where σ_0 is the hydrostatic stress; $\sigma_x,\ \sigma_y,\ \sigma_z$ are stress tensor components in x-direction, y-direction and z-direction respectively.

The deviatoric stress tensor is equal to the stress tensor minus the hydrostatic stress:

$$\left[s_{ij}\right] = \begin{bmatrix} \sigma_x - \sigma_0 & \tau_{xy} & \tau_{xz} \\ \tau_{yx} & \sigma_y - \sigma_0 & \tau_{yz} \\ \tau_{zx} & \tau_{zy} & \sigma_z - \sigma_0 \end{bmatrix} \qquad (4\text{-}23)$$

where $\left[s_{ij}\right]$ is the deviatoric stress tensor; $\begin{bmatrix} \sigma_x - \sigma_0 & \tau_{xy} & \tau_{xz} \\ \tau_{yx} & \sigma_y - \sigma_0 & \tau_{yz} \\ \tau_{zx} & \tau_{zy} & \sigma_z - \sigma_0 \end{bmatrix}$ is the deviatoric stress tensor components.

4.2.2 Principal stress

Like the stress tensor, the deviatoric stress tensor also has three invariants:

$$J_1 = \sigma_x - \sigma_0 + \sigma_y - \sigma_0 + \sigma_z - \sigma_0 = 0 \qquad (4\text{-}24)$$

$$J_2 = \frac{1}{6}\left[(\sigma_x - \sigma_y)^2 + (\sigma_y - \sigma_z)^2 + (\sigma_z - \sigma_x)^2 + 6(\tau_{xy}{}^2 + \tau_{yz}{}^2 + \tau_{zx}{}^2)\right] \quad (4\text{-}25)$$

$$J_3 = \begin{vmatrix} \sigma_x - \sigma_0 & \tau_{xy} & \tau_{xz} \\ \tau_{yx} & \sigma_y - \sigma_0 & \tau_{yz} \\ \tau_{zx} & \tau_{zy} & \sigma_z - \sigma_0 \end{vmatrix} \quad (4\text{-}26)$$

where J_1, J_2, J_3 are the invariants of deviatoric stress tensor.

The load angle was derived from the second invariant and the third invariant of deviatoric stress tensor:

$$\theta_\sigma = \frac{1}{3}\sin^{-1}\left[\frac{-\sqrt{27}J_3}{2J_2{}^{\frac{3}{2}}}\right] \quad (4\text{-}27)$$

where θ_σ is the load angle.

Principal stress was derived by the second invariant of stress tensor, load angle and hydrostatic stress:

$$\sigma_1 = \frac{2\sqrt{J_2}}{\sqrt{3}}\sin\left(\theta_\sigma + \frac{2\pi}{3}\right) + \sigma_0 \quad (4\text{-}28)$$

$$\sigma_2 = \frac{2\sqrt{J_2}}{\sqrt{3}}\sin(\theta_\sigma) + \sigma_0 \quad (4\text{-}29)$$

$$\sigma_3 = \frac{2\sqrt{J_2}}{\sqrt{3}}\sin\left(\theta_\sigma - \frac{2\pi}{3}\right) + \sigma_0 \quad (4\text{-}30)$$

where σ_1, σ_2, σ_3 are the first principal stress, the second principal stress and the third principal stress respectively.

4.2.3 Equivalent stress

The first, the second and the third principal stress can express the equivalent stress (von Mises stress):

$$\overline{\sigma} = \frac{1}{\sqrt{2}}\sqrt{(\sigma_1 - \sigma_2)^2 + (\sigma_2 - \sigma_3)^2 + (\sigma_3 - \sigma_1)^2} \quad (4\text{-}31)$$

where $\overline{\sigma}$ is the equivalent stress.

In summary, the compute steps has been reduced to the minimum requirements according to the sequence of a) hydrostatic stress; b) second and third invariant of deviatoric stress tensor; c) load angle; d) principal stress; and e) equivalent stress.

4.3 Numerical Methods

Differential equations are used to model problems that involve the change of some variable with respect to another. These problems require the solution to an initial-value problem—that is, the solution to a differential equation that satisfies a given initial condition (Chapra and Canale, 1998). The numerical methods of initial-value problem, which are used in creep damage analysis, are different with its mathematical instruction due to its integration process is not consecutive. The derivation of stress and the derivation of creep strain are carried out alternately; thus, the numerical integration of each element will be executed only once in each loop in order to keep the unity of time.

Ling et al (2000) and Hayhurst et al. (1984) both suggested that RKM method is the most suitable numerical method in creep damage analysis. However, a researcher (Wang and Wang, 1996) addressed that Euler's method could also be used for creep damage analysis if the time interval is small enough. In addition, RKF method was also mentioned by both Ling et al. (2000) and Hayhurst et al. (1984), and they think this method could improve the accuracy of integration but consume more computing power.

Four types of numerical method, Euler's method, RK4 method, RKM method and RKF method, were selected to develop the class of numerical method subroutine. Detailed formulas were presented in (Faires and Burden, 2013, Chapra and Canale, 1998, Hayhurst et al., 1984).

4.3.1 Euler's method

In computational science, the Euler's method is used for solving ordinary differential equations (ODEs) with a given initial value. It is the most basic explicit method for numerical integration of ODEs, and it is the simplest RK method. The Euler's method is a first order method, and it often serves as the basis to construct methods that are more complex.

The initial value problem of ODEs could be expressed as:

$$\frac{dy}{dt} = f(t, y), \qquad \text{for } a \le t \le b \tag{4-32}$$

where the initial condition is $y(a) = \alpha$.

The estimating equation of Euler's method is,

$$y_{i+1} = y_i + f(x_i, y_i)h \tag{4-33}$$

4.3.2 Classical 4th order Runge-Kutta method

The RK methods are an important family of implicit and explicit iterative methods, which are used for the approximation of solutions of ordinary differential equations. One member of the family of RK methods is often referred to as RK4.

The estimating equation of RK4 method is,

$$y_{i+1} = y_i + \frac{1}{6}(k_1 + 2k_2 + 2k_3 + k_4)h \tag{4-34}$$

where the coefficient could be estimated as:

$$k_1 = f(x_i, y_i) \tag{4-35}$$

$$k_2 = f\left(x_i + \frac{1}{2}h, y_i + \frac{1}{2}k_1\right) \tag{4-36}$$

$$k_3 = f\left(x_i + \frac{1}{2}h, y_i + \frac{1}{2}k_2\right) \tag{4-37}$$

$$k_4 = f(x_i + h, y_i + k_3) \tag{4-38}$$

4.3.3 Runge-Kutta-Merson method

RKM method was developed by Merson (1957), and it is based on the large family of RK methods. A Fortran code of the RKM method is available in the NAG library. The RKM method is the earliest proposed method belonging to the family of imbedded methods.

The estimating equation of RKM method is,

$$y_{i+1} = y_i + \frac{1}{6}(k_1 + 4k_4 + k_5)h \tag{4-39}$$

where the coefficient could be estimated as:

$$k_1 = f(x_i, y_i) \quad (4\text{-}40)$$

$$k_2 = f\left(x_i + \frac{1}{3}h, y_i + \frac{1}{3}k_1\right) \quad (4\text{-}41)$$

$$k_3 = f\left(x_i + \frac{1}{3}h, y_i + \frac{1}{6}(k_1 + k_2)\right) \quad (4\text{-}42)$$

$$k_4 = f\left(x_i + \frac{1}{2}h, y_i + \frac{1}{8}(k_1 + 3k_3)\right) \quad (4\text{-}43)$$

$$k_5 = f\left(x_i + h, y_i + \frac{1}{2}(k_1 - 3k_3 + 4k_4)\right) \quad (4\text{-}44)$$

4.3.4 Runge-Kutta-Fehlberg method

RKF method was developed by the German mathematician Erwin Fehlberg (1968, 1969), and it is based on the large family of RK methods. The novelty of Fehlberg's form is that the error in the solution can be estimated by using the higher-order embedded method to enhance the integration accuracy by performing one extra calculation.

The estimating equation of RKF method is,

$$y_{i+1} = y_i + \left(\frac{25}{216}k_1 + \frac{1408}{2565}k_3 + \frac{2197}{4104}k_4 - \frac{1}{5}k_5\right)h \quad (4\text{-}45)$$

where the coefficients could be estimated as:

$$k_1 = f(x_i, y_i) \quad (4\text{-}46)$$

$$k_2 = f\left(x_i + \frac{1}{4}h, y_i + \frac{1}{4}k_1\right) \quad (4\text{-}47)$$

$$k_3 = f\left(x_i + \frac{3}{8}h, y_i + \frac{3}{32}k_1 + \frac{9}{32}k_2\right) \quad (4\text{-}48)$$

$$k_4 = f\left(x_i + \frac{12}{13}h, y_i + \frac{1932}{2197}k_1 - \frac{7200}{2179}k_2 + \frac{7296}{2179}k_3\right) \quad (4\text{-}49)$$

$$k_5 = f\left(x_i + h, y_i + \frac{439}{216}k_1 - 8k_2 + \frac{3680}{513}k_3 - \frac{845}{4104}k_4\right) \quad (4\text{-}50)$$

$$k_6 = f\left(x_i + \frac{1}{2}h, y_i - \frac{8}{27}k_1 + 2k_2 - \frac{3544}{2565}k_3 + \frac{1859}{4104}k_4 - \frac{11}{40}k_5\right) \quad (4\text{-}51)$$

4.4 Time-step Control Procedure

Because of the discontinuities of numerical integration, the time-control approach should be discussed individually. The time-control approach includes time-step selection and time-step acceptance. The time-step acceptance was undertaken by the self-adaptive approach of numerical method, and the time-step selection will be designed according to the main programme of the solver. Due to the structure of the solver, the general self-adaptive approach should be applied through some appropriate amendment. Several time-step selection criteria and time-step acceptance criteria were introduced here, but only one of each was selected.

4.4.1 Time-step selection

Wang (1996) suggested a simple time-step selection method; once the time-step dissatisfies the requirement of accuracy; it will be reduced to half in the next iteration.

Hayhurst research group addressed their approach, see (Hayhurst et al., 1984, Hall, 1990, Hayhurst, 2006), which can be summarized as:

1) The first time-step was selected in user-specified;
2) The subsequent time-steps are $\delta = \min\left[\Delta_{T_p}, \Delta_{T_a}, \Delta_{T_r}\right]$.

where Δ_{T_p} is the previous time-step; $\Delta_{T_a} = \frac{0.1}{\dot{\omega}_{a\,max}}$ $(\omega < 0.8)$, the $\dot{\omega}_{a\,max}$ is the maximum damage rate of all elements; $\Delta_{T_r} = \frac{0.05}{\omega_{r\,max}}$ $(\omega \geq 0.8)$, $\omega_{r\,max} = \max\left|\frac{\dot{\omega}}{\omega}\right|$.

Ling et al. (2000) suggested the approach by a new consideration; they intend to increase the time-step when it was too small. Their approach can be expressed as: $\Delta t = \min[\Delta t_a, \Delta t_b]$, where $\Delta t_a = \Delta t_{n+1} = \delta_1 \Delta t_n$; $\Delta t_b = \frac{\delta_2}{\dot{\omega}_{max}}$.

4.4.2 Time-step acceptance

For the self-adaptive technique, a variable called local truncation error was introduced into the RKM method (Hayhurst et al., 1984, Ling et al., 2000). The local truncation can be expressed as:

$$\text{Local truncation error} = \frac{1}{30}(2k_1 - 9k_3 + 8k_4 - k_5) \qquad (4\text{-}52)$$

A criteria $\theta = \frac{E_\epsilon}{\epsilon_c} < 0.001$ was suggested by Hayhurst et al. (1984), where, E_ϵ is local truncation error, and ϵ_c creep strain. The tolerance of 0.001 is a recommended value, and it could be changed depending on the application's target tolerance.

Faires and Burden (2013) introduced the self-adaptive technique of RKF method. The 4th order function (4-45) was used for the result and 5th order function (4-53) was used to estimate the error.

$$\tilde{y}_{i+1} = y_i + \left(\frac{16}{135}k_1 + \frac{6656}{12825}k_3 + \frac{28561}{56430}k_4 - \frac{9}{50}k_5 + \frac{2}{55}k_6\right)h \qquad (4\text{-}53)$$

The time-step acceptance criterion is,

$$q = \left(\frac{\varepsilon h}{2|\overline{w}_{i+1} - w_{i+1}|}\right)^{\frac{1}{4}} \qquad (4\text{-}54)$$

where, $\overline{w}_{i+1}$ is 5th order function, w_{i+1} is 4th order function; when $q \geq 1$, the integration process will continue, else the integration process will be re-executed.

This research adopted Wang's time-step selection method and Ling's time-step acceptance procedure. This combination was selected because it is very easy to be programmed; moreover, it also avoids the repetitive checking of time-step. However, whether a constant time-step may affect the accumulation of damage is still an uncertain issue.

4.5 Normalization Technique

The normalization technique was used to enhance the accuracy and efficiency of numerical integration. Its specific method was introduced here to develop the normalized constitutive equations subroutine and to revise the main programme of the solver.

In order to reduce the numerical error, Hayhurst et al (1984) and Hall (1990) utilised a technique called normalization to modify the creep constitutive equations. In their approach, the stress, strain and time are divided by fixed proportion and are represented by new notations. All the new notations such as $\Sigma_{ij} = \frac{\sigma_{ij}}{\sigma_0}$, $E_{ij} = \frac{\epsilon_{ij}}{\epsilon_0}$, $S_{ij} = \frac{s_{ij}}{\sigma_0}$, $\epsilon_0 = \frac{\sigma_0}{e}$ are substituted to the creep constitutive equation, and the new obtained equations are used in the finite element programme.

For example, the Kachanov-Rabotnov constitutive equation can be expressed as:

$$\dot{\epsilon_{ij}} = \frac{3}{2} s_{ij} A \frac{{\sigma_e}^{n-1}}{(1-\omega)^n} t^m \qquad (4\text{-}55)$$

$$\dot{\omega} = B \frac{{\sigma_r}^{\chi}}{(1-\omega)^{\phi}} t^m \qquad (4\text{-}56)$$

To enable the time scale of (4-55) and (4-56) to be normalized in the same way and conveniently, a new constant was introduced:

$$B = \frac{B'}{(1+\phi)} \qquad (4\text{-}57)$$

To substitute the new notations mentioned above, (4-55) and (4-56) can be rewritten as:

$$\begin{aligned} \frac{dE_{ij}}{dt} = \frac{\dot{\epsilon_{ij}}}{\epsilon_0} &= \frac{1}{\epsilon_0} \times \frac{3}{2} \times \frac{s_{ij}}{\sigma_0} \times \sigma_0 \times A \times \frac{\frac{{\sigma_e}^{n-1}}{{\sigma_0}^{n-1}} \times {\sigma_0}^{n-1}}{(1-\omega)^n} t^m \\ &= \frac{3}{2} S_{ij} A \frac{e{\Sigma_e}^{n-1} {\sigma_0}^{n-1}}{(1-\omega)^n} t^m \end{aligned} \qquad (4\text{-}58)$$

$$\frac{d\omega}{dt} = \frac{B'}{(1+\phi)} \times \frac{\frac{{\sigma_r}^{\chi}}{{\sigma_0}^{\chi}} \times {\sigma_0}^{\chi}}{(1-\omega)^{\phi}} t^m = B' \frac{{\Sigma_r}^{\chi}}{(1+\phi)(1-\omega)^{\phi}} t^m {\sigma_0}^{\chi} \qquad (4\text{-}59)$$

A normalized time increment has been defined depending on the specific constitutive equations, and now is:

$$d\tau = Ae{\sigma_0}^{n-1} t^m dt \qquad (4\text{-}60)$$

To substitute the normalized time increment, (4-58) and (4-59) can be rewritten as:

$$\dot{E_{ij}} = \frac{dE_{ij}}{d\tau} = \frac{3}{2} S_{ij} \frac{{\Sigma_e}^{n-1}}{(1-\omega)^n} \qquad (4\text{-}61)$$

$$\dot{\omega} = \frac{d\omega}{d\tau} = B' \frac{{\Sigma_r}^{\chi}}{(1+\phi)(1-\omega)^{\phi}} \times \frac{{\sigma_0}^{\chi}}{Ae{\sigma_0}^{n-1}} \qquad (4\text{-}62)$$

In order to simplify (4-62) and to understand its physical significance, a concept called normalized creep failure strain has been introduced (Hall, 1990):

$$E_u = \frac{Ae}{B'} {\sigma_0}^{n-\chi-1} \qquad (4\text{-}63)$$

Hayhurst et al (1984) addressed the value of E_u equal to the normalized creep strain computed from the uni-axial form of (4-55) after the rupture life at the stress σ_0. In addition, the normalized rupture stress Σ_r can be obtained from

$$\sigma_r = [\alpha\sigma_1 + (1-\alpha)\sigma_e] \qquad (4\text{-}64)$$

The final equation form was written as:

$$\dot{E_{ij}} = \frac{3}{2} S_{ij} \frac{{\Sigma_e}^{n-1}}{(1-\omega)^n} \qquad (4\text{-}65)$$

$$\dot{\omega} = \frac{{\Sigma_r}^{\chi}}{E_u(1+\phi)(1-\omega)^{\phi}} \qquad (4\text{-}66)$$

Additionally, Hall (1990) suggested that (4-60) and (4-63) should be divided by 100 for creep strains measured in percent strain. This is not adopted here because the author's subroutine is based on engineer strain system. The main programme of the solver also needs to be revised due to the stresses and strains used in normalized constitutive equations are normalized, see (Hall, 1990).

4.6 Nodal Loads Conditions

Boundary conditions of nodal loads could not be produced by FEMGV directly because of its limited capability. At the same time, the nodal loads distribution method of the solver is not simple; hence, a small calculator of nodal loads was proposed to reduce the calculation of nodal loads arrangements.

Smith and Griffiths (2004) suggested the algorithms of nodal loads, where the axisymmetric elements are different from planar elements and three dimensional elements. Figure 4-1 indicates the nodal force distribution of 3-nodes triangle planar element and 4-nodes quadrilateral planar element. Figure 4-2 indicates the nodal force distribution of 3-nodes triangle axisymmetric element and 4-nodes quadrilateral axisymmetric element. It also presented the formula to calculate each force, in which F_1 and F_2 are nodal forces of each node; r_0 and r_1 are inner radius and outer radius respectively. The equivalent nodal loads of both planar elements and three-dimensional elements depend on the length of loaded side of planar elements or the area of loaded surface of three-dimensional elements,

and it has a fixed proportion on each node. However, the equivalent nodal loads of axisymmetric elements are depending on the position of the element[7].

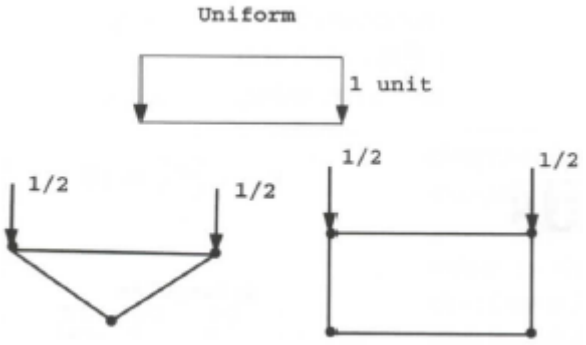

Figure 4-1 Nodal force distribution of 3-nodes triangle and 4-nodes quadrilateral planar element (Smith and Griffiths, 2004)

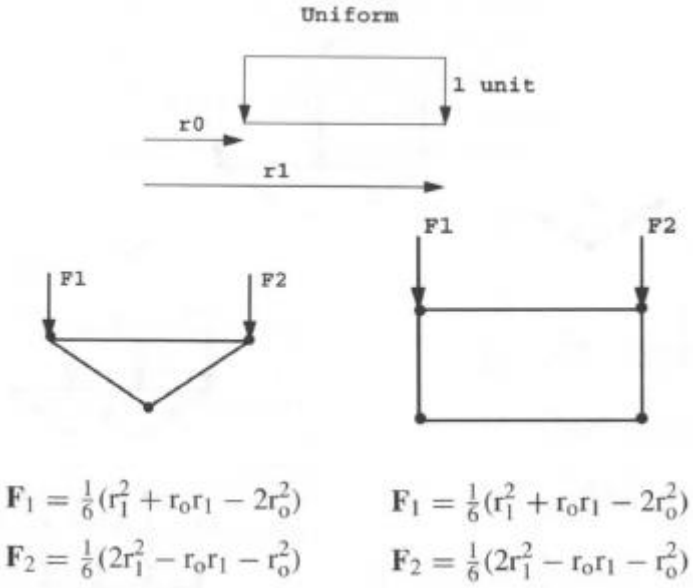

Figure 4-2 Nodal force distribution and formula of 3-nodes triangle and 4-nodes quadrilateral axisymmetric element (Smith and Griffiths, 2004)

Although this thesis does not involve the linear element type, its advantages in damage mechanics are still worth to mention. Compared with quadratic elements, using the linear elements will lose some precision; however, it is easy to remove the failure elements[8].

4.7 File Format of FEMGV and the solver

The file format of FEMGV and the solver is mismatched because of the formatted input and output. Not only the data format of FEMGV need to be understood, but also the data

[7] It means the distance from the axis not rotational symmetry to each loaded node of each loaded element.
[8] This is the knowledge about failure element remove, further discussion can be seen in my colleague's thesis (Liu, 2015).

structure need to be discussed due to the data accepted/produced by the solver could not cover all data types of FEMGV. Synchronization of formatted input and output between FEMGV and the solver is very important. The data types and data format of FEMGV were analysed, and only the data type of the solver was introduced because the solver has free data format. Selective understanding of the file format of FEMGV makes the research more efficient.

4.7.1 File format of FEMGV

FEMGV has two databases, one is for the design environment and the other is for the result environment. All of them can be recorded in ASCII text format called neutral file, and the overall structure of neutral file is shown in Figure 4-3. It could be summarised that each neutral file must include model set, data set and data record. Figure 4-3 shows that the model set includes model header and data set, the data set includes data set header and data record. The data set delimiter[9] is '-3' and the file delimiter is '9999'.

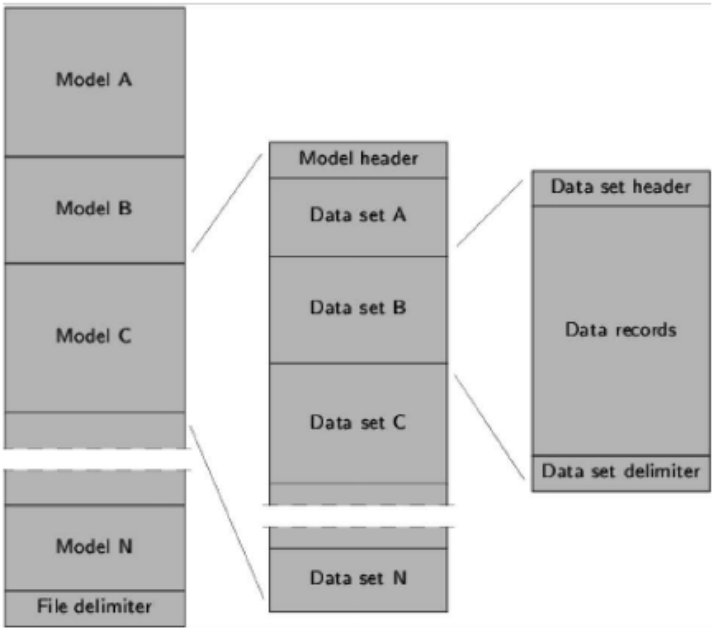

Figure 4-3 Overall structure of FEMGV (Manie and Wolthers, 2013)

Table 4-1 shows data categories of the Neutral file in the design environment. In this research, only node coordinates, part definition, element definition, constrains and concentrated forces were considered due to the data structure of the solver. Table 4-2 shows data categories of the Neutral file in the result environment. In this research, only

[9] Delimiter is a FEMGV defined characteristic, which used to recognise data type. It appears on the start of each recording line. Especially, a delimiter '9999' must be added on the end of each neutral file; the infinite loop will occur if it is never encountered.

node coordinates, element definition, and user-defined results were considered due to the data structure of the solver.

Table 4-1 Data categories of Neutral file in design environment

Record type	KEY	OPCODE	Name
Model header	1	C	Model name
Node coordinates	2	C	
Part definition	101	C	
Element definition	3	C	
Material properties	20	C	
Physical properties	102	C	
Constraints	103	C	
Generalized constraints	104	C	
Elastic supports	105	C	
Coordinate systems	107	C	
Concentrated forces	110	C	
Prescribed displacements	111	C	
Pressure load	112	C	
Temperature load	113	C	
Gravity load	114	C	
Centrifugal load	115	C	

Table 4-2 Data categories of Neutral file in result environment

Record type	KEY	OPCODE	Name
User header	111	U	User key
Project header	111	P	Project name
Model header	111	C, A, B	Model name
Node coordinates	2	C	
Element definition	3	C	
Node sets	18	C	
Element sets	19	C	
Material properties	20	C	
Integration point coordinates	30	C	
Nodal results local systems	32	C	
Element results local systems	33	C	
Integration point results local	34	C	
Nodal constraints	41	C	
Local coordinate systems	43	C	
Transformations	50	C	
Symbols	53	C	
User defined results	100	C, I, R	Load case name

Table 4-3 indicates the specific syntax of node coordinates, part definition, element definition, material properties, constraints and concentrated forces. Through the observation of these syntaxes, it is easy to find that the data types do not need a fixed

position in the file because a logic control algorithm could be designed based on the ‘key’ and ‘opcode’.

Table 4-3 Specific data format of Neutral file in design environment

Data set header
__ KEY OPCODE *Name* _____ IFORMT __ MAXOUT 1X I4 A1 6A1 61X I2 1X I2 *Data records*
Node coordinates
__ 2 C *Name* _____ IFORMT __ MAXOUT 1X I4 A1 6A1 61X I2 1X I2 __ -1 NODE X Y Z 1X I2 I10 3E14.7
Part definition
__ 101 C *Name* _____ IFORMT __ MAXOUT 1X I4 A1 6A1 61X I2 1X I2 __ -1 __ ANAME LPNR TYPE NE0 NE1 NANT 1X I2 1X 8A1 I10 I10 I10 I10 I10 __ -2 NODES 1X I2 10I10
Element definition
__ 3 C *Name* _____ IFORMT __ MAXOUT 1X I4 A1 6A1 61X I2 1X I2 __ -1 NUMBER TYPE GROUP MATERIAL _____ VARIANT PHYSICAL 1X I2 I10 I5 I5 I5 5X I5 I5 __ -2 NODES 1X I2 10I10
Constraints
__ 103 C *Name* _____ IFORMT __ MAXOUT 1X I4 A1 6A1 61X I2 1X I2 __ -1 __ ANAME LPNR __ NANT __ DOFS 1X I2 1X 8A1 I5 1X I5 1X 6I1 __ -2 NODES 1X I2 10I10
Concentrated forces
__ 110 C *Name* _____ IFORMT __ MAXOUT 1X I4 A1 6A1 61X I2 1X I2 __ -1 LCASE NODE DOF VALUE 1X I2 I5 I10 I5 E15.5

Table 4-4 indicates the specific syntax of node coordinates, element definition, and user defined results.

Table 4-4 Specific data format of Neutral file in result environment

Data set header
__ KEY OPCODE *Name* STPVAL _____ Text ICTYPE NUMSTP Analysis IFORMT 1X I4 A1 6A1 E12.5 12X 20A1 I2 I5 10A1 I2 *Data records*
Node coordinates
__ 2 C *Name* STPVAL _____ Text ICTYPE NUMSTP Analysis IFORMT 1X I4 A1 6A1 E12.5 12X 20A1 I2 I5 10A1 I2 __ -1 NODE X Y Z SYSID 1X I2 I10 3E12.5 I15

Element definition
__ 3 C *Name* STPVAL _____ Text ICTYPE NUMSTP Analysis IFORMT 1X I4 A1 6A1 E12.5 12X 20A1 I2 I5 10A1 I2 __ -1 NUMBER TYPE GROUP MATERIAL SYSID VARIANT 1X I2 I10 I5 I5 I5 I5 I5 __ -2 NODES 1X I2 10I10
User defined result
__ 100 C *Name* STPVAL _____ Text ICTYPE NUMSTP Analysis IFORMT 1X I4 A1 6A1 E12.5 12X 20A1 I2 I5 10A1 I2 *Attribute header* *Component definition* *[Attribute variant] . . .*
Attribute header
__ -4 __ *Name* NCOMPS IRTYPE NORCTY _____ *Orig* 1X I2 2X 8A1 I5 I5 I5 10X 8A1
Component definition
__ -5 __ *Name* MENU ICTYPE ICIND1 ICIND2 IEXIST ICNAME ICDATA 1X I2 2X 8A1 I5 I5 I5 I5 I5 8A1 8A1
Entity header of element at node
__ -1 ELEM TYPE GROUP IRECTY NODAL NSRF _____ ISYSTM 1X I2 I5 I5 I5 I5 I5 I5 5X I5 *Data records*
Entity header of element at integration point
__ -1 ELEM TYPE GROUP IRECTY NODAL NSRF INTEG ISYSTM 1X I2 I5 I5 I5 I5 I5 I5 I5 I5 __ -2 IPNT X Y Z 1X I2 I5 3E12.5 *Data records*
Entity data
__ -2 NUMB COMP1 COMP2 COMP3 COMP4 COMP5 COMP6 1X I2 I5 E12.5 E12.5 E12.5 E12.5 E12.5 E12.5

4.7.2 File format of the solver

The data structure of the solver in input environment follows the sequence displayed below:

1. Element type, number of element, number of nodes, material properties
2. Coordinates definition
3. Element definition
4. Constraint
5. Nodal loads

The data structure of the solver in output environment follows the sequence displayed below:

1. Element type, number of element, number of nodes, material properties
2. Coordinates definition

3. Element definition
4. Displacement
5. Body loads
6. Coordinates of integration point
7. Elastic stress
8. Elastic strain
9. Creep strain
10. Creep damage

4.8 Summary

The knowledge presented in this chapter involves a) constitutive equations based on continuum damage mechanics; b) derivation of deviatoric stress, principal stress and equivalent stress based on stress tensor; c) single-step explicit numerical method of initial value problem; d) self-adaptive approach of time-step and its control scheme; e) normalization technique; f) boundary conditions of nodal loads based on planar and axisymmetric element type; and g) formatted input and output of FEMGV and the solver has been studied and analysed.

KR constitutive equations, PH constitutive equations and QX constitutive equations were selected to develop the subroutine of constitutive equations. A critical computing sequence of a) hydrostatic stress, b) second and third invariants of deviatoric stress tensor, c) load angle, d) principal stress, and e) equivalent stress was chosen to develop the stress tensor transformation subroutine. Euler's method, RK4 method, RKM method and RKF method were selected to develop the class of numerical method subroutine. Wang's time-step selection method and Ling's time-step acceptance procedure were adopted as the time-step control procedure. The normalization technique used to normalize the constitutive equations was introduced. The nodal force distributions of 3-nodes triangle element and 4-nodes quadrilateral element were reported to develop the nodal load calculator. The file format of FEMGV and the solver were analysed to develop the data transfer interface.

5 DESIGN OF ALGORITHM AND STRUCTURE

This chapter presents the algorithm, arguments and structure of all developed subroutines and programmes of this research. The algorithm was designed based on the theories and knowledge, which are used to achieve the required function of each subroutine and programme; moreover, the structure was designed based on the structure of Prototype I. A number of variables were defined according to the variables of Prototype I due to the consideration of subroutine interface.

Table 5-1 presents brief summary descriptions of the involved theories of each subroutine and programme. The developing sequence of subroutines depends on their position in the solver from top to bottom, and depends on the priority of prototype version; hence, its order is a) stress tensor transformation subroutine; b) numerical method subroutine; c) constitutive equations subroutine; d) time-step control subroutine; e) normalization subroutine.

Table 5-1 Relative theories and knowledge of each subroutine and programme

Subroutine/Programme	Relative theories and knowledge
Constitutive equations subroutines	• Kachanov-Rabotnov constitutive equations • Perrin-Hayhurst constitutive equations • Qiang Xu's constitutive equations
Stress tensor transformation subroutine	• Conversion between stress tensor and deviatoric stress tensor • The derivation approach of principal stress, based on the invariants of deviatoric stress tensor • The derivation approach of equivalent stress, based on the principal stress
Numerical method subroutines	• Euler's method • 4^{th} order classical Runge-Kutta method • Runge-Kutta-Merson method • Runge-Kutta-Fehlberg method
Time-step control subroutine	• Time-step acceptance approach • Time-step selection approach • Self-adaptive technique of Runge-Kutta-Merson method • Self-adaptive technique of Runge-Kutta-Fehlberg method
Normalization technique subroutine	• Normalized Kachanov-Rabotnov constitutive equation • Normalized equilibrium equation of finite element method
Boundary conditions calculator	• The nodal loads equation of 3-nodes triangle element • The nodal loads equation of 4-nodes quadrilateral element
Data transfer programme	• File format of FEMGV • File format of the solver • Arguments arrangement of the solver

5.1 Transformation of Stress Tensor

The computing of constitutive equations involved a series of distinct stress forms; hence, a number of classical stress forms such as deviatoric stress tensor, principal stress and equivalent stress were integrated into a conversion subroutine called stress tensor transformation subroutine due to the consideration of efficiency and convenience.

5.1.1 Algorithm of TRS

A subroutine called Transformation of Stress (**TRS)** was required to convert the deviatoric stress tensor, the principal stress and the equivalent stress based on the normal

stress tensor. According to the theoretical analysis, the computing sequence is a) hydrostatic stress; b) second and third invariant of deviatoric stress tensor; c) load angle; d) principal stress; and e) equivalent stress in **TRS**.

Figure 5-1 shows the process of **TRS**, where it is obvious to see that the equations are the same for two-dimensional problem or three-dimensional problem, because **TRS** solves two-dimensional problem through reducing the three-dimensional equations. **cas_sel_4** represents two-dimensional problem and **cas_sel_6** represents two-dimensional problem. The key point is that only local variables are involved in the computations, and the stress components that are not required can be set as zero in the allocation of global variables[10].

Figure 5-1 also shows that when the process move into the first selection branch, if the selection reference is incorrect, a warning statement will be printed on screen, then the process move into the next statement. This can be seen as a self-checking function to help user to identify the error of input data. It does not affect the execution of subroutine itself. The same situation, which occurs on the later sections, will not be explained repeatedly.

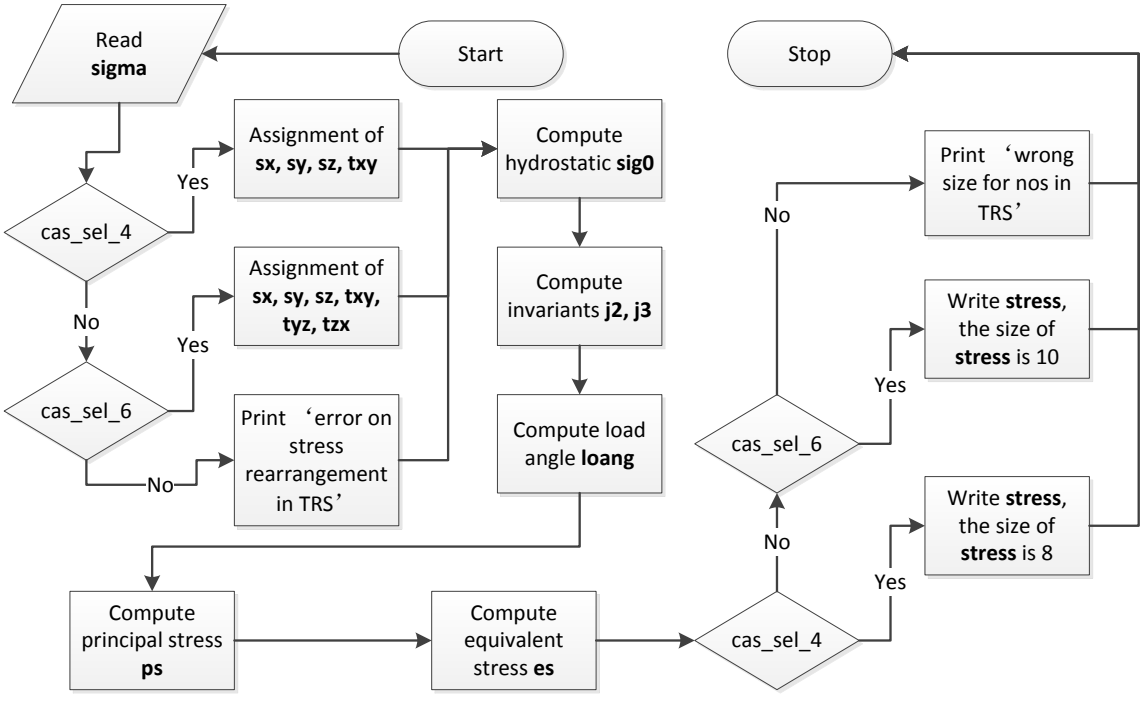

Figure 5-1 Algorithm of stress tensor transformation subroutine

[10] In FEM, the number of stress terms in 2D environment and 3D environment is different, two unrequired shear stress will be force to zero when the plane elements are applied.

TRS was called after the completion of elastic stress derivation; hence, the array stored elastic stress tensor will be delivered to **TRS** from the main programme of the solver. The number of stress components is changed after the conversion; therefore, a new array should be designed to store the deviatoric stress tensor, the principal stress and the equivalent stress. For the consideration of convenient debugging and maintenance, **TRS** used two sets of variable systems, one is a global variable, which is used to deliver data between main programme and subroutine; one is a local variable, which is used to prevent the memory mistakes. At the start of **TRS**, the global variable is allocated to a local variable, and the local variable will be allocated back to the global variable at the end of **TRS**.

5.1.2 Variables Definition

Table 5-2 presents the definition of all local variables, which is used for the local computations only. The deviatoric stress tensor components were not defined because their value can be represented by the component of stress tensor and the hydrostatic stress.

Table 5-2 Variables dictionary of TRS

Variables	Descriptions
Global variables	
sigma	Input data, which stored the normal and shear stress tensor. Its size and type should synchronize with the main programme of the solver. It is a deferred-shape array in order to ignore the number of stress component.
stress	Output data, which stored the deviatoric stress tensor, the principal stress and the equivalent stress. Its size and type should synchronize with the main programme of the solver. It is a deferred-shape array in order to ignore the number of stress component.
Local variables	
sx, sy, sz, txy, tyz, tzx	It is real, which means the components of stress tensor.
pi	It is real, which means the constant π.
j2, j3	It is real, which means the second and third stress invariants.
sig0	It is real, which means the hydrostatic stress.
loang	It is real, which means the load angle.
es	It is real, which means the equivalent stress.
ps(3)	It is real array, which stores the principal stress.
nos	It is integer, which means the number of stress terms.

The activity of deriving the unknown deviatoric stress tensor, principal stress and equivalent stress is a stress transformation process; hence, only two real arrays were needed in this subroutine, one is known stress and the other is unknown stress. The known stress was obtained from the main programme of the solver, so its setting can follow the

main programme to name it as **sigma**. The size of **sigma** depends on the specific problem, two-dimensional problem is 4 and the three-dimensional problem is 6.

The array of unknown stress was named **stress** in order to distinguish the known stress and unknown stress. Array **stress** stored deviatoric stress tensor, principal stress and equivalent stress; hence, its size equals to the sum of the number of deviatoric stress tensor components, the number of principal stress and the number of equivalent stress. The number of deviatoric stress tensor components depends on the specific problem, two-dimensional problem is 4 and the three-dimensional problem is 6. The number of principal stresses is 3 and the number of equivalent stresses is 1, so the size of **stress** is 8 in two-dimensional problem and 10 in three-dimensional problem.

5.1.3 Subroutine Structure

TRS consists of two parts, variable declaration and execution; its pseudo code was obtained through Figure 5-1 and Table 5-2. The detailed code is attached in appendix 10.2, and the pseudo code is displayed below.

```
1   Subroutine TRS (sigma, stress)
    Execution section:
2   obtain (sigma) from the main programme
3   nos ← ubound( sigma,1 )
4   pi ← 3.1415926
5   allocate (sigma) of (sx, sy, sz, txy, tyz, tzx)
6   sig0 ← (sx+sy+sz)/3.
7   j2 ← ((sx-sy)**2+(sy-sz)**2+(sz-sx)**2)/6.+txy**2+tyz**2+ tzx**2
8   j3 ← (sx-sig0)*(sy-sig0)*(sz-sig0)+2*txy*tyz*tzx-(sx-sig0)*tyz**2-(sz-
    sig0)*txy**2-(sy-sig0)*tzx**2
9   loang ← asin((-sqrt(27.)*j3)/(2*sqrt(j2**3)))/3
10  ps(1) ← 2*sqrt(j2)/sqrt(3.)*sin(loang+2*pi/3)+sig0
11  ps(2) ← 2*sqrt(j2)/sqrt(3.)*sin(loang)+sig0
12  ps(3) ← 2*sqrt(j2)/sqrt(3.)*sin(loang-2*pi/3)+sig0
13  es ← 1/sqrt(2.)*sqrt((ps(1)-ps(2))**2+(ps(2)-ps(3))**2+ (ps(3)-ps(1))**2)
14  update (stress)
15  return (stress) to main programme
```

5.2 Constitutive Equations

Constitutive equations subroutine is a class of subroutines, which has a unified template of subroutine structure and algorithm; and, three kinds of constitutive equations were selected in this research based on continuum damage mechanics.

5.2.1 Algorithm of CES

A class of subroutines called Constitutive Equations Subroutine (**CES**) was required to find the solution of creep constitutive equations, which includes creep strain rate and creep damage rate. According to the theoretical analysis, the computing sequence is a) creep strain rate of each direction; and b) creep damage rate in **CES**.

Figure 5-2 shows the process of **CES**, it is obvious to see that specific constitutive equations were divided into three cases, which are a) uni-axial case; b) two-dimensional case; and c) three-dimensional case. The subroutine will select the correct case according the variable representing the number of equations. **cas_sel_1** represents uni-axial form of constitutive equations, **cas_sel_8** represents multi-axial form of constitutive equations (four equations for strain components) and **cas_sel_10** represents multi-axial form of constitutive equations (six equations for strain components). **cas_sel_x**, **cas_sel_xx** and **cas_sel_xxx** are user-defined which recommend using the number of constitutive equations as the selection reference.

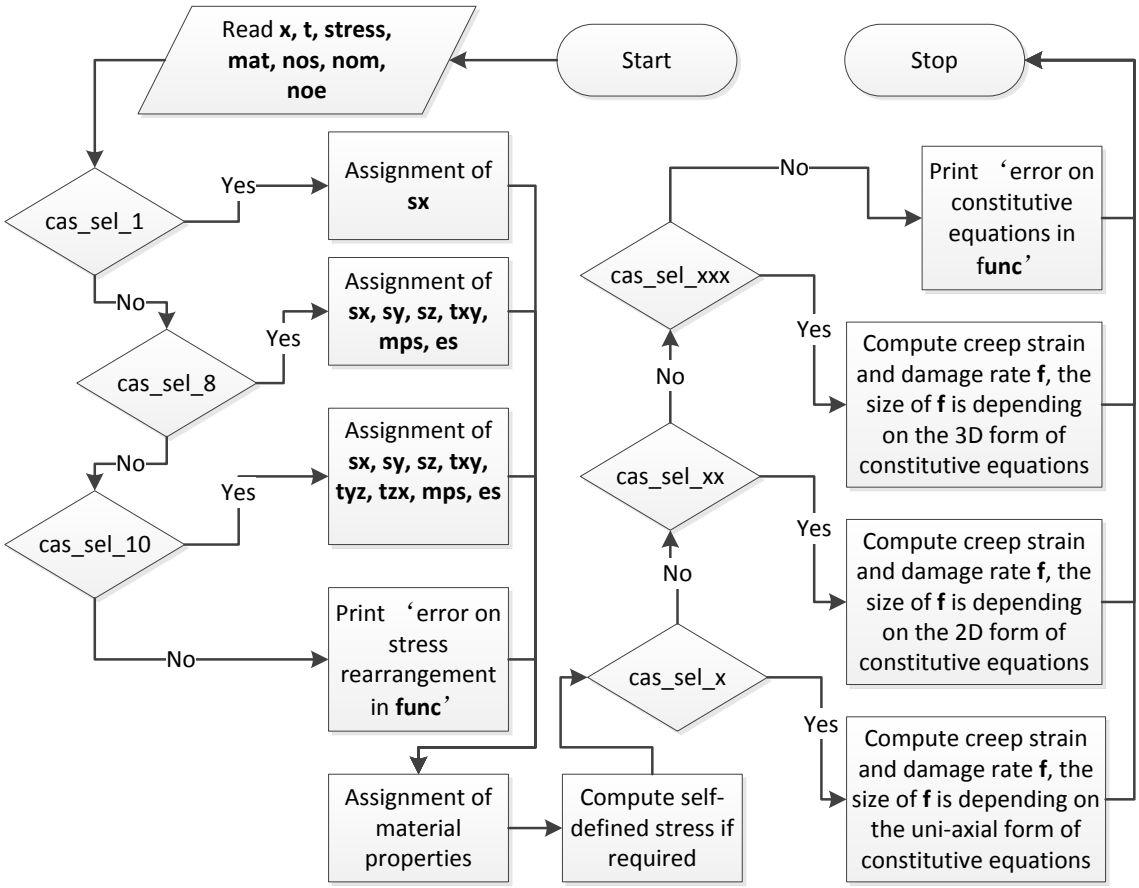

Figure 5-2 Algorithm of constitutive equaitons subroutines

CES was called in the numerical method subroutine; hence, the array stored converted stresses, material properties and the initial value of creep strain and damage will be delivered to **CES** from the numerical method subroutine. At the same time, the creep strain rate and creep damage rate will be returned to numerical method subroutine from **CES** as well.

In order to distinguish the problems of two-dimensions and three-dimensions, **CES** used two sets of variable systems, one is a global variable, which is used to deliver data between the main programme and subroutine; one is a local variable, which is used to offer required parameters depending on the number of equations. At the start of **CES**, the global variable will be allocated to a local variable to prepare the case selection of two-dimensional problem and three-dimensional problem. KR constitutive equations, PH constitutive equations and QX constitutive equations were selected to code in this research. **CES** has a template, which enables users to build their own constitutive equations subroutines[11].

5.2.2 Variables Definition

Table 5-3 presented the definition of all local variables, which is used to distinguish the two-dimensional problem and three-dimensional problem, and to prevent coding mistakes. The descriptions of the relationship of creep strain and stress are the constitutive equations; hence, the consideration of variables definition should contain all dependent variables, independent variables and parameters of creep constitutive equations.

Table 5-3 Variables dictionary of CES

Variables	Descriptions
Global variables	
f	Output data, which is an explicit-shape array used to store the creep strain rate and creep damage rate. Its size and type should synchronize with the variable of **noe**.
x	Input data, which is an explicit-shape array used to store the creep strain damage. Its size and type should synchronize with the variable of **noe**.
stress	Input data, which is an explicit-shape array used to store the deviatoric stress tensor, the principal stress and the equivalent stress. Its size and type should synchronize with the variable of **nos**, which point out the operation environment is two dimensional problem or three-dimensional problem.
mat	Input data, which is an explicit-shape array used to store the material property parameters. Its size and type should synchronize with the variable of **nom**.
t	Input data, which is a real used to store the value of present time.
nos	Control variables, which is a real used to represent the number of stress terms.

[11] The users must be familiar with some understanding of FEA and Fortran coding. Once the user completes their own constitutive equations subroutine, it is very easy to integrate their subroutine into HITSI.

nom	Control variables, which is a real used to represent the number of material properties.
noe	Control variables, which is a real used to represent the number of equations.
Local variables	
sx, sy, sz, txy,	It is real, which means the components of deviatoric stress tensor.
mps	It is real, which means the first principal stress.
es	It is real, which means the equivalent stress.
A, n, m, B, phi,	Samples, which are real, and means material property parameters

A real array named **x** used to store the value of creep strain and damage were needed in this subroutine. The initial value of **x** was obtained from main programme of the solver, in which the size of **x** depends on the number of equations called **noe**.

A real array named **f** used to store the value of creep strain rate and creep damage rate were stated in this subroutine. The value of **f** will be returned to the numerical method subroutine, in which the size of **f** depends on **noe** as well. A real array variable called **t** is used to hold the variable of time is also reserved due to some constitutive equations having integrated into time functions. Material properties were stored in the real array named **mat**. The initial value of **mat** was obtained from main programme of the solver, which the size of **mat** is depending on the number of material properties. A real array called **stress** was obtained from the main programme of the solver, and was derived by the subroutine of **TRS**.

5.2.3 Subroutine Structure

CES consists of two parts, variable declaration and execution; its pseudo code was obtained through Figure 5-2 and Table 5-3. The detailed code is attached in appendix 10.2, and the pseudo code sample of subroutine **KR** (using KR equations) is displayed below. The other two is subroutine **PH** (using PH equations) and **QX** (using QX equations).

```
1   Subroutine KR  (f, x, t, stress, mat, nos, nom, noe)
    Execution section:
2   obtain (nos, noe, nom, mat(nom), t, x(noe), stress(nos)) from numerical method
    subroutine
3   allocate the stress terms which stored in global variable to local variables
4   allocate the material properties which stored in global variable to local variables
5   rs ← alpha*mps+(1.-alpha)*es
6   select case ( problem type )
7   case ( uni-axial )
8   f(1) ← A*((sx/(1-x(2)))**n)*(t**m)
9   f(2) ← B*(sx**chi)/((1-x(2))**phi)*(t**m)
10  case ( 2D problem )
11  f(1) ← (3./2.)*(sx/es)*A*((es/(1-x(5)))**n)*(t**m)
12  f(2) ← (3./2.)*(sy/es)*A*((es/(1-x(5)))**n)*(t**m)
```

```
13  f(3) ← (3./2.)*(txy/es)*A*((es/(1-x(5)))**n)*(t**m)
14  f(4) ← (3./2.)*(sz/es)*A*((es/(1-x(5)))**n)*(t**m)
15  f(5) ← B*(rs**chi)/((1-x(5))**phi)*(t**m)
16  case ( 3D problem )
17  f(1) ← (3./2.)*(sx/es)*A*((es/(1-x(7)))**n)*(t**m)
18  f(2) ← (3./2.)*(sy/es)*A*((es/(1-x(7)))**n)*(t**m)
19  f(3) ← (3./2.)*(sz/es)*A*((es/(1-x(7)))**n)*(t**m)
20  f(4) ← (3./2.)*(txy/es)*A*((es/(1-x(7)))**n)*(t**m)
21  f(5) ← (3./2.)*(tyz/es)*A*((es/(1-x(7)))**n)*(t**m)
22  f(6) ← (3./2.)*(tzx/es)*A*((es/(1-x(7)))**n)*(t**m)
23  f(7) ← B*(rs**chi)/((1-x(7))**phi)*(t**m)
24  end select
25  return (f) to numerical method subroutine
```

5.3 Numerical Integration Method

Like the constitutive equations subroutine, the numerical method subroutine is also a class of subroutines, and has a unified template. Four kinds of numerical methods were selected in this research based on the single step method of ordinary differential equations.

5.3.1 Algorithm of NMS

A class of subroutine called Numerical Method Subroutine (**NMS**) was required to integrate the constitutive equations, which includes computing of functions' slope and updating of functions' solution. According to the theoretical analysis, the computing sequence is 1) estimates of slope depending on different time-steps; 2) mean slope; 3) solution updating in **NMS**. The self-adaptive approach will be discussed in section 5.4. Figure 5-3 shows the process of **NMS**, where it is obvious to see that the functions' slopes will be derived through the constitutive equations subroutine; hence, the dummy arguments list of **NMS** should consider the nature of constitutive equations subroutine. **NMS** solves two-dimensional problem and three-dimensional equations depending on the argument of the number of equations.

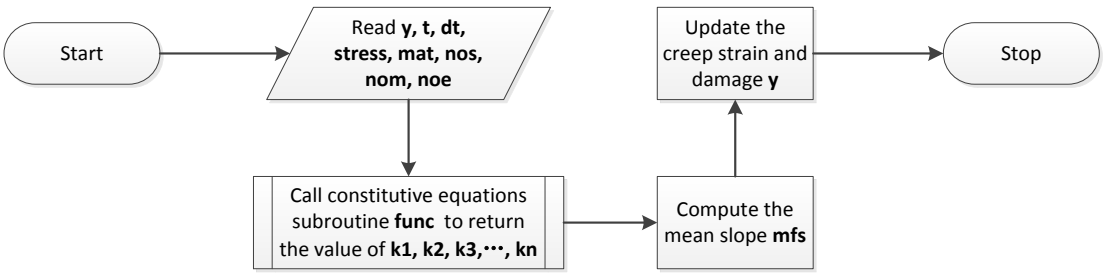

Figure 5-3 Algorithm of numerical method subroutines

NMS was called after the completion of stress conversion; hence, the array stored converted stresses will be delivered to **NMS** from the main programme of the solver. At the same time, the material properties, time-step size and the initial value of creep strain and damage will be delivered to **NMS** from the main programme of the solver as well.

NMS will not distinguish local variables and global variables since the major execution part will be done by external subroutines.

In this research, four types of numerical method subroutine were included in **NMS**, which are Euler's method, RK4 method, RKM method and RKF method. **NMS** has a template, which enables users to build their own numerical method subroutines as well.

5.3.2 Variables Definition

The activity of finding the solution of constitutive equations is the numerical integration process; hence, the consideration of variables definition should involve the numerical method itself and constitutive equations. The numerical method aims to find the solution of constitutive equations; therefore, a real array named **y** used to store the value of creep strain and damage were needed in this subroutine. The initial value of **y** was obtained from the main programme of the solver, so its setting can follow the main programme, where the size of **y** depends on the number of equations.

Table 5-4 presents the definition of all local variables, which is used to derive the mean slope of constitutive equations and to return the value of creep strain and damage.

Table 5-4 Variables dictionary of NMS

Variables	Descriptions
Global variables	
func	Name of constitutive equations subroutine.
y	Input and output data, which is an explicit-shape array used to store the creep strain damage. Its size and type should synchronize with the variable of **noe**.
stress	Input data, which is an explicit-shape array used to store the deviatoric stress tensor, the principal stress and the equivalent stress. Its size and type should synchronize with the variable of **nos**, which point out the operation environment is two dimensional problem or three-dimensional problem.
mat	Input data, which is an explicit-shape array used to store the material property parameters. Its size and type should synchronize with the variable of **nom**.
t	Input data, which is a real used to store the value of present time.
dt	Input data, which is a real used to store the time-step.
nos	Control variables, which is a real used to represent the number of stress terms.
nom	Control variables, which is a real used to represent the number of material properties.
noe	Control variables, which is a real used to represent the number of equations.

Local variables	
k_1, k_2, $k_3 \ldots k_n$	They are explicit-shape array used to store slopes of functions obtained using different time-step. Its size and type should synchronize with the variable of **noe**.
mfs	It is an explicit-shape array used to store mean slopes of functions. Its size and type should synchronize with the variable of **noe**.

External subroutine **func**, which represents the constitutive equations subroutine was used to find the creep strain rates and creep damage rates. All of these rates will be stored in the local variables named $\mathbf{k_n}$., which $\mathbf{k_n}$ are a series of variables. Due to the like with the external subroutine, the corresponding dummy arguments of the external subroutine must be included; for example, variables of **stress**, **mat**, **t**, **nos**, **nom** and **noe** have the same setting as the constitutive equations subroutine.

The time-step was stored in the real variable **dt**, and the value of **dt** was updated from the main programme of the solver during every iteration loops of constitutive equations.

5.3.3 Subroutine Structure

NMS consists of two parts, variable declaration and execution; its pseudo code was obtained through Figure 5-3 and Table 5-4. The detailed code is attached in appendix 10.2, and the pseudo code sample of subroutine **RK4** (using RK4 method) is displayed below. The other three is subroutine **EULER** (using Euler's method), **RKM** (using RKM method) and **RKF** (using RKF method).

```
1  Subroutine RK4 (func, y, t, dt, stress, mat, nos, nom, noe)
   external :: func  [name of constitutive equations subroutine]
   Execution section:
2  obtain ( nos, noe, nom, mat(nom), t, dt, y(noe), stress(nos) ) from the main programme
3  call func(k1,y,t,stress,mat,nos,nom,noe)
4  call func(k2,y+dt/2*k1,t+dt/2,stress,mat,nos,nom,noe)
5  call func(k3,y+dt/2*k2,t+dt/2,stress,mat,nos,nom,noe)
6  call func(k4,y+dt*k3,t+dt,stress,mat,nos,nom,noe)
7  mfs ← (k1+2*k2+2*k3+k4)/6
8  y ← y+mfs*dt
9  return (y) to main programme
```

5.4 Time-step Control Procedure

The solution of time-step control consists of time-step acceptance and time-step selection. Especially, the part of time-step acceptance was integrated into the numerical method subroutine. The time-step control is also reflected in the structure of the solver, which

means the new creep strain and damage will not be updated if the current integration is unaccepted.

5.4.1 Algorithm of TSC and self-adaptive approach

A subroutine called Time-Step Control (**TSC**) was required to give a new time-step for every iteration loop of constitutive equations. The new time-step selection criteria depend on the self-adaptive approach of the numerical method. The standard of self-adaptive technique was integrated into the numerical method subroutine. Figure 5-4 shows the co-operation of **TSC** and **NMS,** where the important controller of time-step control scheme was produced after the integration process and then go to the branch of stress updating decision. The only termination criterion is that the value of creep damage exceeds its critical value. Figure 5-5 shows the specific determination process of a new time-step. When the last integration process satisfies the standard of the self-adaptive technique, the new time-step equals to the old time-step; however, when the last integration process does not satisfy the standard of the self-adaptive technique, the new time-step equals to the half of the old time-step.

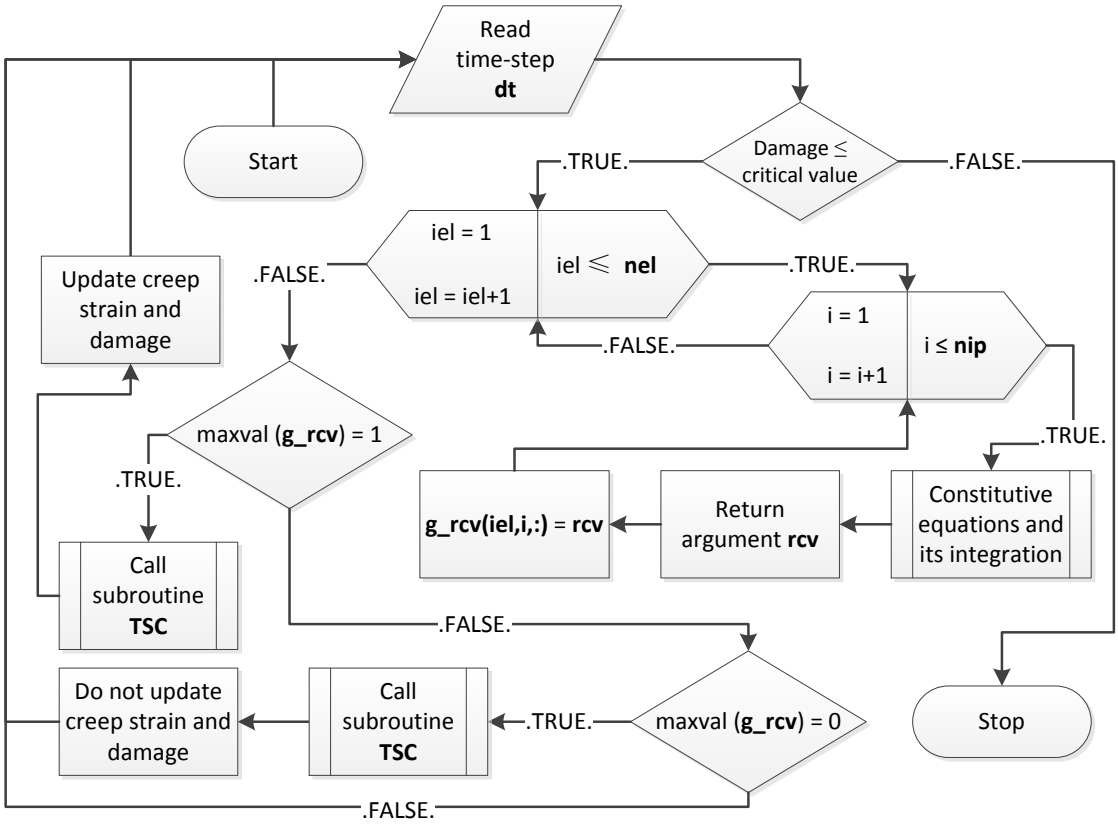

Figure 5-4 Algorithm of time-step control in the solver

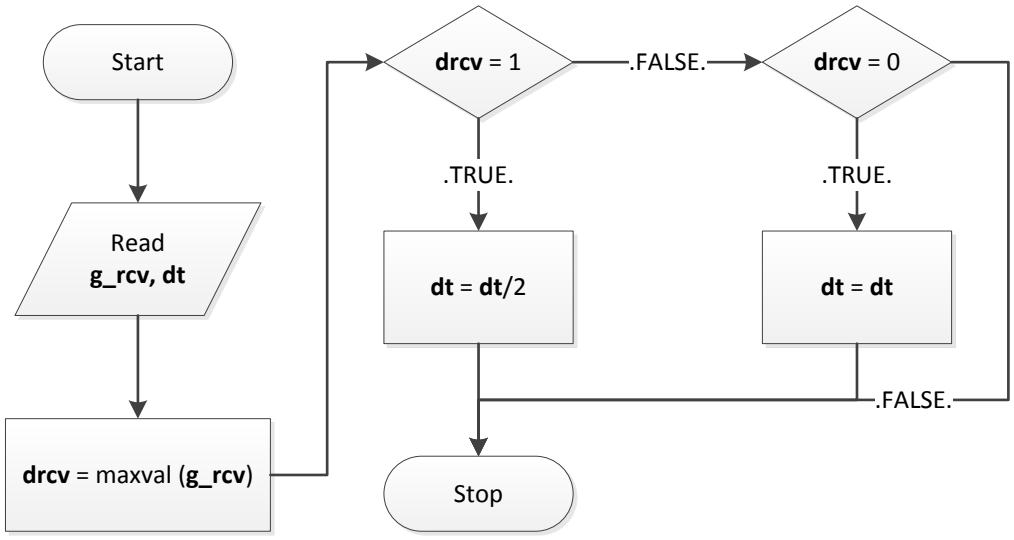

Figure 5-5 Algorithm of time-step selection subroutine

5.4.2 Variables Definition

The activity of determining a new time-step based on the controller of self-adaptive is time-step selection; hence, the consideration of variables definition should involve the controller and time-step itself. A three-dimensional deferred-shape real array named **g_rcv** used to store global controller of self-adaptive were needed in this subroutine. The value of each controller is 0 or 1 only, where '1' indicates that the present loop is not accepted and '0' indicates that the present loop is accepted.

The present time-step was stored in the real variable **dt**; however, in order to reduce the memory requirement, the new time-step will be returned to **dt** as well. Table 5-5 shows the definition of local variables **drcv**, which is used to find the maximum controller of self-adaptive time-step approach of all equations.

Table 5-5 Variables dictionary of TSC

Variables	Descriptions
Global variables	
g_rcv	Input data, which is a deferred-shape array used to store the global controller of self-adaptive time-step approach. Its size and type should synchronize with the main programme of the solver.
dt	Input and output data, which is a real used to store the time-step.
Local variables	
drcv	It is integer, which means the maximum controller of self-adaptive time-step approach.

The local variables, which were shown in Table 5-6, should be added to **NMS**. In order to determine the acceptance of self-adaptive approach, a real array **aoi**, whose size depends on the number of equations, was designed to hold all values of acceptance criteria. Due to

the nature of the acceptance criteria, the maximum value was chosen to be stored in the real variable **maoi**. Once the present iteration satisfies the acceptance criteria, the controller will be defined as '0', else the controller is '1'; hence, an integer variable called **rcv** was proposed to hold the controller of self-adaptive approach.

Table 5-6 Unique variables dictionary of self-adaptive approach in NMS

Variables	Descriptions
Local variables	
aoi	It is an explicit-shape array used to hold the acceptance of integration
maoi	It is a real, which means the maximum acceptance of integration
rcv	It is an integer, which means the controller produced by self-adaptive approach. Its value is 0 or 1 only.

5.4.3 Subroutine Structure

TSC consists of two parts, variable declaration and execution; its pseudo code was obtained through Figure 5-5 and Table 5-5. The detailed code is attached in appendix 10.2, and the pseudo code sample of subroutine **TSC** is displayed below:

```
1   Subroutine TSC (dt, g_rcv)
    Execution section:
2   obtain (dt, g_rcv) from the main programme
3   drcv ← maxval(g_rcv)
4   if drcv = 1
5   dt ← dt/2
6   else if  drcv = 0
7   dt ← dt
8   end if
9   return (dt) to main programme
```

New **NMS** consists of two parts, variable declaration and execution; its pseudo code was obtained through Figure 5-3, Table 5-4 and Table 5-6. The detailed code is attached in appendix 10.2, and the pseudo code sample of subroutine **RKM** is displayed below:

```
1   Subroutine RKM (func, y, t, dt, stress, mat, nos, nom, noe)
    external :: func  [name of constitutive equations subroutine]
    Execution section:
2   obtain ( nos, noe, nom, mat(nom), t, dt, y(noe), stress(nos) ) from the main
    programme
3   call func(k1,y,t,stress,mat,nos,nom,noe)
4   call func(k2,y+dt/3.*k1,t+dt/3.,stress,mat,nos,nom,noe)
5   call func(k3,y+dt/6.*(k1+k2),t+dt/3.,stress,mat,nos,nom,noe)
6   call func(k4,y+dt/8.*(k1+3*k3),t+dt/2.,stress,mat,nos,nom,noe)
7   call func(k5,y+dt/2.*(k1-3.*k3+4.*k4),t+dt,stress,mat,nos,nom, noe)
```

```
 8   mfs ← (k1+4.*k4+k5)/6.
 9   y ← y+mfs*dt
10   loer ← (2*k1-9*k3+8*k4-k5)/30.
11   aoi ← loer/mfs
12   maoi ← maxval(aoi)
13   if  maoi<0.001
14   rcv ← 0
15   else
16   rcv ← 1
17   return (y, rcv) to main programme
```

5.5 Normalization Scheme

The normalization technique contains a set of normalized constitutive equations and normalized equilibrium equations of the finite element method. The normalized constitutive equations can be coded based on constitutive equation subroutines template mentioned above. The normalized equilibrium equations of the finite element method were reflected in the structure of the solver, which means the elastic stress and Young's modulus will be 1 at the start of constitutive equations integration.

5.5.1 Algorithm of NOR_KR

A subroutine called Normalized KR (**NOR_KR**) was required to find the solution of normalized KR constitutive equations, which includes creep strain rate and creep damage rate. According to the theoretical analysis, the computing sequence is a) creep strain rate of each direction; and b) creep damage rate in **NOR_KR**. **NOR_KR** is also a kind of constitutive equations subroutine; hence, all settings of **CES** were followed in here except some unique variable definition.

5.5.2 Variables Definition

The major difference of variables definition of **CES** and **NOR_KR** is shown in Table 5-7. The definition of variable **stress** not only includes the deviatoric stress tensor, the principal stress and the equivalent stress, but also includes the internal pressure used for the normalization. Additionally, a number of new local variables were introduced; for example, a concept of rupture stress and creep fracture strain was used to reduce the computing workload. They were stored in real variables **rs** and **cfs** respectively. Young's modulus was included into the argument of material prooerties, and its value will be allocated to the real variable **e**.

Table 5-7 Unique variables dictionary of NOR_KR

Variables	Descriptions
Global variables	
stress	Input data, which is an explicit-shape array used to store the deviatoric stress tensor, the principal stress, the equivalent stress and the internal pressure. Its size and type should synchronize with the variable of **nos** plus 1.
Local variables	
A, n, m, B, phi, chi, alpha	They are real, which mean material property parameters.
ip	It is real, which means the internal pressure used for the normalization.
e	It is real, which means Young's modulus
rs	It is real, which means rupture stress
cfs	It is real, which means creep fracture strain

5.5.3 Subroutine Structure

NOR_KR consists of two parts, variable declaration and execution; its pseudo code was obtained through Figure 5-2, Table 5-3 and Table 5-7. The detailed code is attached in appendix 10.2, and the pseudo code is displayed below:

```
1   Subroutine NOR_KR (f, x, t, stress, mat, nos, nom, noe)
    Execution section:
2   passing (nos, noe, nom, mat(nom), t, x(noe), stress(nos)) from the main
    programme to subroutine XX
3   allocate ( stress(nos) ) to (sx, sy, sz, txy, tyz, tzx, mps, es, ip)
4   allocate ( mat(nom) ) to (A, n, m, B, phi, chi, alpha)
5   rs ← alpha*mps+(1.-alpha)*es
6   cfs ← (A*e/B)*(ip**(n-chi-1.))
7   select case ( problem type )
8   case ( 2D problem )
9   f(1) ← (3./2.)*(sx/es)*((es/(1-x(5)))**n)
10  f(2) ← (3./2.)*(sy/es)*((es/(1-x(5)))**n)
11  f(3) ← (3./2.)*(txy/es)*((es/(1-x(5)))**n)
12  f(4) ← (3./2.)*(sz/es)*((es/(1-x(5)))**n)
13  f(5) ← (rs**chi)/(cfs*(1.+phi)*((1.-x(5))**phi))
14  case ( 3D problem )
15  f(1) ← (3./2.)*(sx/es)*((es/(1-x(7)))**n)
16  f(2) ← (3./2.)*(sy/es)*((es/(1-x(7)))**n)
17  f(3) ← (3./2.)*(sz/es)*((es/(1-x(7)))**n)
18  f(4) ← (3./2.)*(txy/es)*((es/(1-x(7)))**n)
19  f(5) ← (3./2.)*(tyz/es)*((es/(1-x(7)))**n)
20  f(6) ← (3./2.)*(tzx/es)*((es/(1-x(7)))**n)
21  f(7) ← (rs**chi)/(cfs*(1.+phi)*((1.-x(7))**phi))
22  end select
23  return (f) to numerical method subroutine
```

5.6 Boundary Conditions of Nodal Loads

Boundary conditions calculator was used to obtain the boundary condition of nodal loads when conducting a practical case. It is a primary solution of nodal loads conversion subroutine.

5.6.1 Algorithm of NLC

A programme called Nodal Load Calculator (**NLC**) was required to give the arrangement of nodal loads distribution. FEMGV does not give the specific value of nodal load to each node, only a uniform load was recorded in the Neutral file of design environment. The unique conversion formulas of nodal loads were obtained from Smith and Griffiths (2004). 3-nodes triangle planar element, 4-nodes quadrilateral planar element, 3-nodes triangle axisymmetric element, 4-nodes quadrilateral axisymmetric element were considered in here.

Figure 5-6 shows the computing flow of nodal loads calculator. All input data were written into the input file, which is ending with the suffix '.dat'. The input file records the type of problem, number of loaded elements, uniform loads and the relative position of each node.

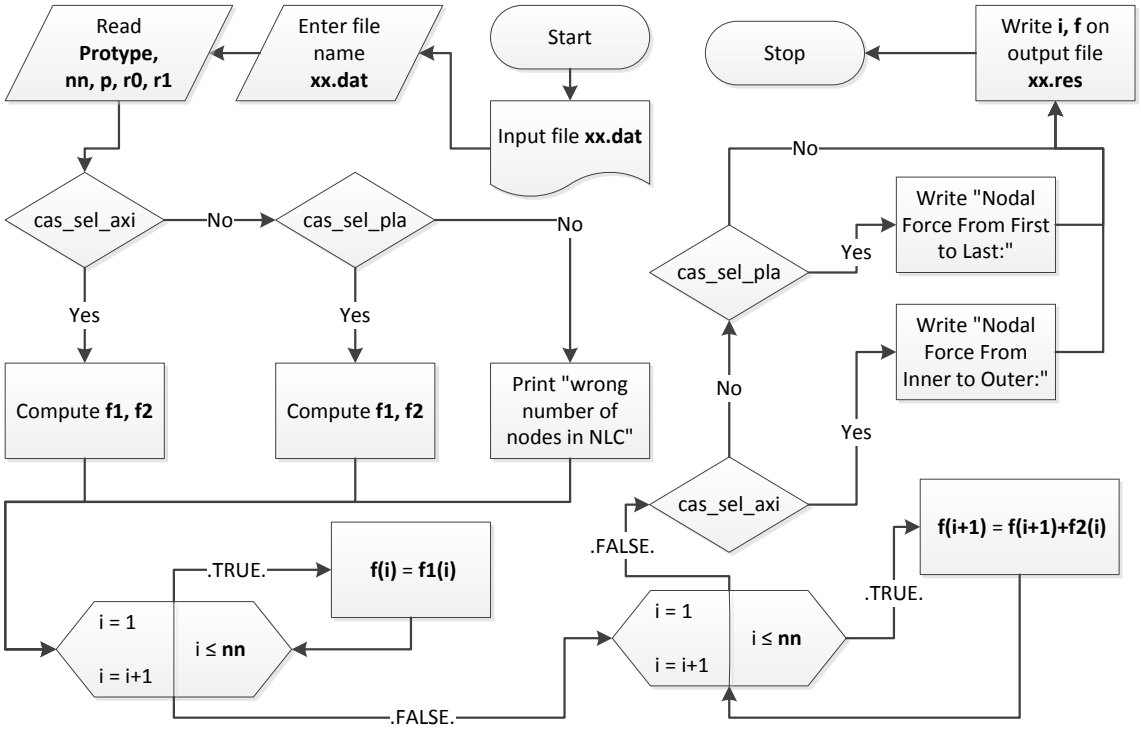

Figure 5-6 Algorithm of nodal loads calculator

The relative positions of the first node of each loaded element were processed and held firstly, and then the relative positions of the second node of each loaded element were processed and held.

5.6.2 Variables Definition

Two deferred-shape arrays called $\mathbf{r_0}$ and $\mathbf{r_1}$ were designed to hold the relative positions of the first and second node of each loaded element due to some nodes are corner nodes.

Table 5-8 shows that, two deferred-shape arrays called $\mathbf{f_1}$ and $\mathbf{f_2}$, which were used to store the first and second component of nodal loads of each node, were introduced due to the relative positions bring processed separately. Real array **f** was used to hold the result of nodal loads.

Table 5-8 Variables dictionary of NLC

Variables	Descriptions
filename1	It is character string, which means the name of input file.
protype	It is character string, which means the problem type.
p	It is real, which means the uniform loads.
f	Output data, which is a deferred-shape array used to store the nodal loads of each loaded node. Its size is depending on the argument of loaded nodes, which equals to the number of loaded elements **nn** plus one due to the nature of 3-nodes and 4-nodes element.
f_1	It is a deferred-shape array used to store the first component of nodal loads of each node. Its size should synchronize with the variable of **nn**.
f_2	It is a deferred-shape array used to store the second component of nodal loads of each node. Its size should synchronize with the variable of **nn**.
r_0	It is a deferred-shape array used to store the relative position of first node of each element. Its size should synchronize with the variable of **nn**.
r_1	It is a deferred-shape array used to store the relative position of second node of each element. Its size should synchronize with the variable of **nn**.
nn	It is integer, which means the number of loaded elements.
i	It is integer, which means the loops counter.
ierror	It is integer, which means the value of **iostat** statement.

5.6.3 Subroutine Structure

DTI consists of two parts, pre-processing transfer and post-processing transfer; its pseudo code was obtained through Figure 5-7 and Table 5-9. The detailed code is attached in appendix 10.2, and the pseudo code is displayed below:

```
1   Programme NLC
    Execution section:
2   read input file name (filename1)
3   open input and output file
4   read the type of problem (protype)
```

```
5   read the number of loaded elements (nn)
6   read the uniform loads (p)
7   allocate (f, f1, f2, r0, r1)
8   read the relative position of each node (r0, r1)
9   select case ( protype )
10  case ( 'axisymmetric' )
11  starting of counted-loops with (nn)
12  f1(i) ← (r1(i)**2+r0(i)*r1(i)-2*r0(i)**2)/6
13  f2(i) ← (2*r1(i)**2-r0(i)*r1(i)-r0(i)**2)/6
14  end loops
15  case ( 'planar' )
16  starting of counted-loops with (nn)
17  f1(i) ← (r1(i)-r0(i))/2
18  f2(i) ← f1(i)
19  end loops
20  end select
21  starting of counted-loops with (nn)
22  f(i) ← f1(i)
23  end loops
24  starting of counted-loops with (nn)
25  f(i+1) ← f(i+1)+f2(i)
26  end loops
27  f ← f*p
28  write (f) into output file
29  end programme
```

5.7 Pre- and Post-processing

Data transfer programme is the solution to unify the file format. It is an independent programme due to this unique research environment; however, it can be integrated into the solver in future research.

5.7.1 Algorithm of DTI

A programme called Data Transfer Interface (**DTI**) was required to transfer the data between FEMGV and the solver. Both the pre-processing transfer and post-processing transfer are integrated into one programme. The pre-processing transfer includes five basic data types, which are 1) node coordinates, 2) element definition, 3) part definition, 4) constraint and 5) concentrated force. The post-processing transfer includes nine data types, which are 1) node coordinates, 2) element definition, 3) material properties, 4) displacement, 5) body loads, 6) elastic stress, 7) elastic strain, 8) creep strain and 9) damage. Figure 5-7 shows the overall algorithm of DTI, where the left side is pre-processing transfer and the right side is post-processing transfer.

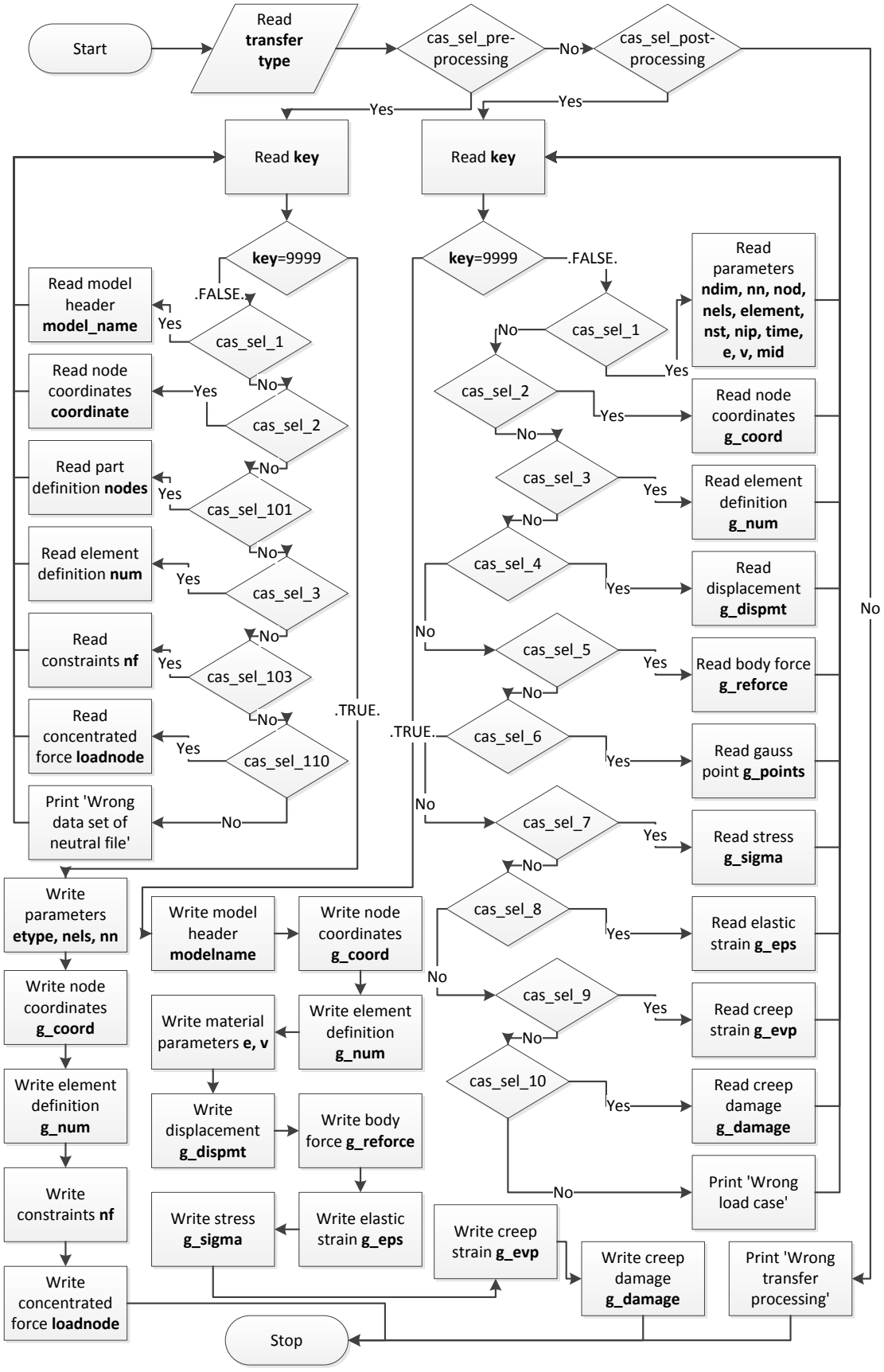

Figure 5-7 Overall algorithm of data transfer interface

The number of node and the number of element will not be offered through FEMGV directly; hence, in order to control the reading process of Neutral file of FEMGV logically, simple reading way of line by line is not enough because the loops is impossible to be controlled. However, a number of identifiers listed on the starting position of each line of the Neutral file guaranteed the possibility of logical reading.

The conversion of concentrated force was taken out to the previous section due to it involves the topological theory; the concentrated force could be converted automatically until a new algorithm to be proposed. The writing of Neutral file is a challenge because the solver will not offer a number of parameters required by FEMGV. The specific value of those parameters should be tested one by one according to the output data types of the solver.

5.7.2 Variables Definition

Table 5-9 indicates the definition of all variables used in **DTI**, and these variables were reported in order of appearance in the code from top to bottom. A character string **transfertype** was designed to distinguish the pre-processing transfer and post-processing transfer through the case selection. Both pre-processing transfer and post-processing transfer use the same character string $\mathbf{filename_1}$ and $\mathbf{filename_2}$ to represent the name of input file and output file. The logical reading was controlled through an integer variable **key**, and the value of **key** is dynamic because its value will be updated in every line.

The data records could be de defined according to the file format guidance of FEMGV. For examples, a) a real array **coordinate** is used to indicate the coordinate components and its size is 3; b) an integer array **nodes** size is 10, which was used to store the node number on the part; and c) three integer variables called $\mathbf{dof_1}$, $\mathbf{dof_2}$ and $\mathbf{dof_3}$ indicate which of the three degrees of freedom are constrained: 0 for not affected, 1 for affected.

Those obtained data mentioned above will not be written in file directly because of the restriction of the algorithm, but those defined variables will be updated during every loop; thus, a series of variables were proposed to hold the obtained data of every loop. For example, a two-dimensional real array **g_coord** is used to indicate the global coordinates, and its size of the first rank is 3 and the size of second rank should synchronize with the variable of number of nodes. The number of nodes was counted through the algorithm developed by the author.

A number of parameters should be tested before programming, and they are designed as integer variables such as **iformt**$_1$, **ncomps**$_1$ and **irtype**. These variables were used to write the Neutral file according to the structure of user defined results.

Table 5-9 Variables dictionary of DTI

Variables	Descriptions
transfertype	It is a character string used to indicate the transfer process type, which only has two value; value of 'pre-processing' represented the transfer process of FEMGV file, and value of 'post-processing' represented the transfer process of the solver file.
filename$_1$, filename$_2$	They are character string used to indicate the name of input files and the name of output files respectively; and their size are restricted to 30.
key	It is an integer variable used to identify the data type of FEMGV file; for example, '2' represented the definition of coordinate, '-1' represented the data itself and '9999' represented the termination of file.
opcode	It is a character used to assist **key**, and its value is always 'C'.
model_name, modelname	They are character string used to indicate the geometry model name of design environment and the geometry model name of result environment respectively in FEMGV; and their size are restricted to 30.
n_1, n_2, $n_{3,\dots}$,n_8	They are character variables, which represent an alphabet used to compose a random name.
iformt	It is an integer variable used to represent the format indicator of data; for example, '0' for a five digit format, '1' for a ten digit format, '2' for a ten digit with high precision nodal coordinates
maxout	It is an integer variable used to indicate the maximum number of data per line.
nn	It is an integer variable used to indicate the number of node.
coordinate	It is a real array used to indicate the coordinate components. Its size is 3.
g_coord	It is a two-dimensional real array used to indicate the global coordinates. The size of first rank is 3 and the size of second rank should synchronize with the variable of **nn**.
lpnr	It is an integer variable used to indicate the sequence number of the line, surface or body.
types	It is an integer variable used to indicates the type of the part; for example, '2' for a line, '3' for a surface and '4' for a body.
ne$_0$, ne$_1$	They are an integer variable used to indicate the number of the first element on the part and the number of the last element on the part respectively.
nant	It is an integer variable used to indicate the number nodes in the subsequent data continuation records.
nodes	It is an integer array and the size is 10, which used to store the node number on the part (for all parts except bodies).
nels	It is an integer variable used to indicate number of element
group	It is integer which used to indicate the element group number.
material	It is integer which used to indicate the material property number.
variant	It is integer which used to indicate the element variant number.
physical	It is integer which used to indicate the physical property number.
nod	It is an integer variable used to indicate the number of node in per element
num	It is a real array used to indicate the element definition. Its size is '3' or '4'.
g_num	It is a two-dimensional real array used to indicate the element definition. The size of first rank is 3 or 4 and the size of second rank should synchronize with the variable of **nels**.
g_mat	It is a real array used to indicate the global material type. Its size should synchronize with the variable of **nels**.
aname	It is character string, which represent the name of a point, line, surface, or body.

dof_1, dof_2, dof_3	They are integer, which indicate which of the three degrees of freedom are constrained: 0 for not affected, 1 for affected
nf	It is a two-dimensional real array used to indicate the constraint. The size of first rank is 2 or 3 and the size of second rank should synchronize with the variable of **nn**.
k	It is an integer variable used to indicate the node number.
lcase	It is an integer variable used to indicate the load case number.
elem	It is an integer variable used to indicate the node number
dof	It is an integer variable used to indicates the degree of freedom: 1 for FX, 2 for FY, 3 for FZ, 4 for MX, 5 for MY, 6 for MZ.
$value_1$	It is a real variable used to indicate the magnitude of the concentrated load.
loadnode	It is a real array used to indicate the nodal loads. Its size should synchronize with the variable of **j**.
etype	It is character string, which represent the type of element.
g_num_1	It is a two-dimensional real array used to indicate the element definition after conversion. The size of first rank is 3 or 4 and the size of second rank should synchronize with the variable of **nels**.
i, j	They are real variables used to be the counters.
ndim	It is an integer variable used to indicate the number of dimension.
element	It is character string, which represent the type of element.
nst	It is integer which used to indicate the number of stress components of stress tensor.
nip	It is integer which used to indicate the number of integration point.
time	It is a real variable used to indicate the time moment.
e	It is an integer variable used to indicate the Young's modulus.
v	It is an integer variable used to indicate the Poisson's ratio.
mid	It is integer which used to indicate the number of material types.
node	It is an integer variable used to indicate node number.
elemt	It is an integer variable used to indicate element number.
$material_1$	It is a real array used to indicate the material type. The size of first rank is 1 and the size of second rank should synchronize with the variable of **nels**.
g_dispmt	It is a two-dimensional real array used to indicate the global displacement. The size of first rank is 2 or 3 and the size of second rank should synchronize with the variable of **nels**.
g_reforce	It is a two-dimensional real array used to indicate the global body loads. The size of first rank is 2 or 3 and the size of second rank should synchronize with the variable of **nels**.
g_points	It is a three-dimensional real array used to indicate the global integration points coordinates. The size of first rank is 2, the size of second rank should synchronize with the variable of **nip** and the size of third rank should synchronize with the variable of **nels**.
point	It is an integer variable used to indicate the integration point number.
g_sigma	It is a three-dimensional real array used to indicate the global elastic stress. The size of first rank is 4 or 6, the size of second rank should synchronize with the variable of **nip** and the size of third rank should synchronize with the variable of **nels**.
g_eps	It is a three-dimensional real array used to indicate the global elastic strain. The size of first rank is 4 or 6, the size of second rank should synchronize with the variable of **nip** and the size of third rank should synchronize with the variable of **nels**.
g_evp	It is a three-dimensional real array used to indicate the global creep strain. The size of first rank is 4 or 6, the size of second rank should synchronize with the variable of **nip** and the size of third rank should synchronize with the variable of **nels**.
g_damage	It is a three-dimensional real array used to indicate the global creep damage. The size of first rank is 1, the size of second rank should synchronize with the variable of **nip** and the size of third rank should synchronize with the variable of **nels**.
$opcode_1$	It is a character variable used to assist **key**, and its value is always 'C'.

key_1, key_2, key_3,…,key_{11}	They are integer variable used to identify the data type of result environment of FEMGV file only; for example, '2' represented the definition of coordinate, '-1' represented the data itself and '9999' represented the termination of file.
The arguments of (**iformt$_1$, ncomps$_1$, irtype, norcty, menu, ictype, icind$_1$, icind$_2$, iexist, ncomps$_2$, icind$_3$, group$_1$, nsc, nlc**) are integer, which used to give the fixed constant to the file of result environment of FEMGV. They do not have any real meaning.	
ierror	It is integer, which means the value of **iostat** statement.

5.7.3 Subroutine Structure

DTI consists of two parts, pre-processing transfer and post-processing transfer; its pseudo code was obtained through Figure 5-7 and Table 5-9. The detailed code is attached in appendix 10.2, and the pseudo code is displayed below:

```
1   Programme DTI
    Execution section:
2   read the transfer process type (transfertype)
3   select case (transfertype)
4   case (pre-processing)
5   open input file
6   open output file
7   read model
8   start until-loops
9   read the controller of (key)
10  select case (key)
11  case (1, 2, 3, …, n; all case include the following until-loops)
12  start until-loops
13  read data set header
14  read data record
15  end until-loops
16  end select
17  end until-loops
18  write data in output file
19  case (post-processing)
20  open input file
21  open output file
22  start until-loops
23  read the controller of (key)
24  select case (key)
25  cases (include reading of parameters, coordinate, element definition, displacement,
    body force, integration point coordinate, elastic stress, elastic strain, creep strain
    and damage)
26  end select
27  write data in output file
28  end select
29  end programme
```

5.8 summary

Table 5-10 indicated the purpose of all subroutines and programmes that developed in this research. They could be specific to that 1) constitutive equations subroutines include KR, PH and QX; 2) stress transformation subroutine includes TRS; 3) numerical method subroutines include EULER, RK4, RKM and RKF; 4) Time-step control subroutine includes TSC; 5) normalization subroutine includes NOR_KR; 6) Nodal loads calculator is a programme named NLC; and 7) Data transfer interface is a programme as well called DTI. All subroutines developed in this research were held in a library called **Tan_library**.

Table 5-10 Purpose statement of developed subroutines and programmes

Name	Description
KR	**KR** returns the solution of Kachanov-Rabotnov constitutive equations under any environment such as uni-axial, 2D problem and 3D problem.
PH	**PH** returns the solution of Perrin-Hayhurst constitutive equations under any environment such as uni-axial, 2D problem and 3D problem.
QX	**QX** returns the solution of Qiang Xu's constitutive equations under any environment such as 2D problem and 3D problem.
TRS	**TRS** returns varies type of stress include: 1) deviatoric stress tensor, 2) principal stress, and 3) equivalent stress.
EULER	**EULER** returns the integration value of constitutive equations within a specific time interval according to Euler's method.
RK4	**RK4** returns the integration value of constitutive equations within a specific time interval according to 4^{th} order classical Runge-Kutta method.
RKM	**RKM** returns the integration value of constitutive equations within a specific time interval according to Runge-Kutta-Merson method. It includes self-adaptive technique.
RKF	**RKF** returns the integration value of constitutive equations within a specific time interval according to Runge-Kutta-Felhberg method. It includes self-adaptive technique.
TSC	**TSC** returns the next time interval when present iteration is over. It associated with **KRM** and **RKF**
NOR_KR	**NOR_KR** returns the solution of Kachanov-Rabotnov constitutive equation within normalization technique under any environment such as uni-axial, 2D problem and 3D
NLC	**NLC** output a data file saved the value of loads on each nodal.
DTI	**DTI** offers the correct input files for both the solver and FEMGV.

The algorithm of subroutines and programmes were designed based on the analysis of knowledge presented in Table 5-1. The flowcharts of each code were presented to show the detailed structure clearly. A number of important variables, which affected the input or output, were introduced specifically in order to serve the maintenance and enhancement of future. Except that, all variables were summarised into the tables for the convenience of reader. The order of development of subroutine and programme was arranged according to the timeline. The core part is creep constitutive equations; therefore, the first stage of research was launched around creep constitutive equations.

6 TESTING AND VERIFICATION

This chapter presents the testing and verification of all developed subroutines and programmes of this research. Proofreading and debug of code was implemented through Code::Blocks platform. The testing has two categories, one is for subroutines testing and the other is for programmes testing. Since the subroutine could not be executed by itself, a series of testing programmes were developed. Detailed user guidance of each testing programme was reported and future users can repeat those tests reported in this thesis.

Some subroutines such as numerical method subroutines or stress transformation subroutines were developed to assist constitutive equations subroutines; hence, the testing sequence is important. The testing sequence of subroutines depends on their position in the solver from top to bottom; hence, its testing order is a) stress tensor transformation subroutine; b) numerical method subroutine; c) constitutive equations subroutine; d) time-step control subroutine; e) normalization subroutine.

Table 6-1 indicates the intentions of those tests to explain the reasons and significances to conduct these testing. The testing of nodal loads calculator and data transfer interface will be demonstrated through the verification of the output result.

Table 6-1 Description of subroutine's testing

Objectives of Test	Description
Accuracy test of stress transformation subroutine	Unit test, in order to guarantee that the values of deviatoric stress components, principal stress and equivalent stress are correct in both two-dimensional and three-dimensional environment
Accuracy test of numerical method subroutine	Unit test, in order to guarantee that no coding error in each numerical method, and prove that the argument passing process is correct
Coupling test of stress transformation subroutine, numerical method subroutine and constitutive equations subroutines Accuracy test of constitutive equations subroutines	Integration test, in order to guarantee that the co-operation of stress transformation subroutine, numerical method subroutine and constitutive equations subroutines is smoothly, and prove that the output values of both two-dimensional and three-dimensional forms are qualified
Accuracy test of time-step selection subroutine Accuracy test of time-step acceptance part of numerical method subroutine Co-operation test of time-step selection and acceptance	Integration test, in order to guarantee that the co-operation test of time-step selection and acceptance is qualified, and to check the output value of time-step selection subroutine and acceptance part of numerical method subroutine
Accuracy test of normalization subroutine	Integration test, in order to guarantee that output values of both two-dimensional and three-dimensional forms are qualified

6.1 Subroutine of Transformation of Stress Tensor

6.1.1 Instruction of TRScheck

Programme **TRScheck** was developed in order to test the stress transformation subroutine. Figure 6-1 indicates the programme structure, and it could be found that the two-dimensional and three-dimensional environments were distinguished through a branch structure of **nos**; '4' represented two-dimensional environment and '6' represented three-dimensional environment. Those results will be written into a file named by the user. The user instruction is summarised below:

1. Create a new file named **xx.dat**;
2. The first line of input file is the number of stress components, whose value was restricted in '4' or '6';
3. The second line of input file is stress tensor components value;
4. Execute programme **TRScheck**
5. Enter **xx.dat** following the prompts;
6. Enter any output file name following the prompts;
7. Open the result file and see the result.

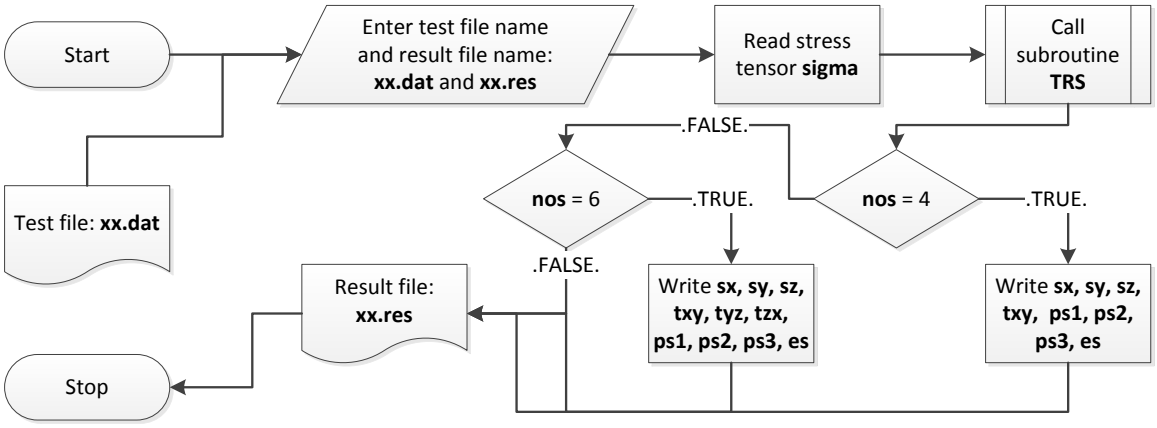

Figure 6-1 Algorithm of programme TRScheck

6.1.2 Statement of testing cases

Two sets of stress tensor, which include four components and six components respectively, were applied in this testing; the case of four components was used to test the two-dimensional environment, and the case of six components was used to test the two-dimensional environment. Table 6-2 shows the specific value of each stress tensor components.

Table 6-2 Stress tensor components used in testing (unit/MPa)

	σ_x/σ_r	σ_y/σ_z	σ_z/σ_θ	τ_{xy}/τ_{rz}	τ_{yz}	τ_{zx}
Test 1	84	-30	0	-32	-----	-----
Test 2	-10	0	7	9	0	5

These two sets of data will be implemented separately, and their computing results will be compared with the results obtained by hand calculation. The error was expected in following situations:

1. If the error occurred on the stress tensor allocation unit; a) the final result will be wrong when subroutine arranged a wrong value, and b) a warning text of "Error on stress rearrangement in TRS" will be presented on screen when the wrong size of stress tensor array was used;
2. If the error occurred on the functions computing unit, the corresponding value will be wrong;
3. If the error occurred on the result updating unit; a) the characteristic is the same with functions error, and static validation should be utilized in here, and b) a warning text of "wrong size for nos in TRS" will be presented on screen when the wrong size of stress tensor array was used.

6.1.3 Result and verification

6.1.3.1 Test 1

The test 1 used the input file TRS1.dat, this file was attached in appendix 10.4.1. Figure 6-2 shows the results of the first testing, which the deviatoric stress components are 66MPa in x-direction, -48MPa in y-direction, -32MPa in xy-direction and -18MPa in z-direction; the principal stresses are 92.368MPa of maximum principal stress and -38.368MPa of minimum principal stress; and the equivalent stress is 116.395MPa. The second principal stress is too small, and can be ignored as 0. This situation is suited for the theory of plane stress.

```
Deviator Stress Tensor:       XX              YY
                          6.6000000E+01-4.8000000E+01
Deviator Stress Tensor:       XY              ZZ
                         -3.2000000E+01-1.8000000E+01
Principal Stress:       1st             2nd             3rd
                    9.2368191E+01 5.4753034E-07-3.8368195E+01
Equivalent Stress:  1.1639588E+02
```

Figure 6-2 Results obtained from stress tensor components 80, -30, 0, -32

In order to verify those results, a hand calculation was conducted:

$$\sigma_0 = \frac{1}{3}(\sigma_x + \sigma_y + \sigma_z) = (84 - 30 + 0) \div 3 = 18$$

$$[s_{ij}] = \begin{bmatrix} \sigma_x - \sigma_0 & \tau_{xy} & \tau_{xz} \\ \tau_{yx} & \sigma_y - \sigma_0 & \tau_{yz} \\ \tau_{zx} & \tau_{zy} & \sigma_z - \sigma_0 \end{bmatrix} = \begin{bmatrix} 66 & -32 & 0 \\ -32 & -48 & 0 \\ 0 & 0 & -18 \end{bmatrix}$$

$$J_2 = \frac{1}{6}\left[(\sigma_x - \sigma_y)^2 + (\sigma_y - \sigma_z)^2 + (\sigma_z - \sigma_x)^2 + 6(\tau_{xy}{}^2 + \tau_{yz}{}^2 + \tau_{zx}{}^2)\right]$$
$$= (114^2 + 30^2 + 84^2 + 6 \times 32^2) \div 6 = 4516$$

$$J_3 = \begin{vmatrix} \sigma_x - \sigma_0 & \tau_{xy} & \tau_{xz} \\ \tau_{yx} & \sigma_y - \sigma_0 & \tau_{yz} \\ \tau_{zx} & \tau_{zy} & \sigma_z - \sigma_0 \end{vmatrix} = \begin{vmatrix} 66 & -32 & 0 \\ -32 & -48 & 0 \\ 0 & 0 & -18 \end{vmatrix} = 75456$$

$$\theta_\sigma = \frac{1}{3}\sin^{-1}\left[\frac{-\sqrt{27}J_3}{2J_2{}^{\frac{3}{2}}}\right] = \sin^{-1}\left[\frac{-\sqrt{27} \times 75456}{2 \times 4516^{\frac{3}{2}}}\right] \div 3 = -0.2341$$

$$\sigma_1 = \frac{2\sqrt{J_2}}{\sqrt{3}}\sin\left(\theta_\sigma + \frac{2\pi}{3}\right) + \sigma_0 = \frac{2\sqrt{4516}}{\sqrt{3}}\sin\left(-0.2341 + \frac{2\pi}{3}\right) + 18 = 92.3682$$

$$\sigma_2 = \frac{2\sqrt{J_2}}{\sqrt{3}}\sin(\theta_\sigma) + \sigma_0 = \frac{2\sqrt{4516}}{\sqrt{3}}\sin(-0.2341) + 18 = -5.02426 \times 10^{-5} \approx 0$$

$$\sigma_3 = \frac{2\sqrt{J_2}}{\sqrt{3}}\sin\left(\theta_\sigma - \frac{2\pi}{3}\right) + \sigma_0 = \frac{2\sqrt{4516}}{\sqrt{3}}\sin\left(-0.2341 - \frac{2\pi}{3}\right) + 18 = -38.3682$$

$$\bar{\sigma} = \frac{1}{\sqrt{2}}\sqrt{(\sigma_1 - \sigma_2)^2 + (\sigma_2 - \sigma_3)^2 + (\sigma_3 - \sigma_1)^2} = \sqrt{130.7364^2 + 38.3682^2 + 92.3682^2} \div \sqrt{2}$$
$$= 116.3959$$

It is obvious to see from Table 6-3 that the subroutine returned a correct value in two-dimensional environment.

Table 6-3 Comparison of computer results and hand calculation results

Result	Computer	Hand calculation
Deviatoric stress in x-direction	6.6000000E+01	66
Deviatoric stress in y-direction	-4.8000000E+01	-48
Deviatoric stress in xy-direction	-3.2000000E+01	-32
Deviatoric stress in z-direction	-1.8000000E+01	-18
First principal stress	9.2368191E+01	92.3682
Second principal stress	5.4753034E-07	0
Third principal stress	-3.8368195E+01	-38.3682
Equivalent stress	1.1639588E+02	116.3959

6.1.3.1 Test 2

The test 2 used the input file TRS2.dat, this file was attached in appendix 10.4.1. Figure 6-3 shows the results of the second testing, which the deviatoric stress components are -9MPa in x-direction, 1MPa in y-direction, 8MPa in z-direction, 9MPa in xy-direction, 0MPa in yz-direction and 5MPa in zx-direction. The principal stresses are 9.342MPa of the first principal stress, 3.767MPa of the second principal stress and -16.109MPa of the third principal stress; and the equivalent stress is 23.173MPa.

```
Deviator Stress Tensor:      XX            YY            ZZ
                        -9.0000000E+00 1.0000000E+00 8.0000000E+00
Deviator Stress Tensor:      XY            YZ            ZX
                         9.0000000E+00 0.0000000E+00 5.0000000E+00
Principal Stress:       1st            2nd            3rd
                   9.3424884E+00 3.7673038E+00-1.6109791E+01
Equivalent Stress:  2.3173262E+01
```

Figure 6-3 Results obtained from stress tensor components -10, 0, 7, 9, 0, 5

In order to verify those results, a hand calculation was conducted:

$$\sigma_0 = \frac{1}{3}(\sigma_x + \sigma_y + \sigma_z) = (-10 + 0 + 7) \div 3 = -1$$

$$[s_{ij}] = \begin{bmatrix} \sigma_x - \sigma_0 & \tau_{xy} & \tau_{xz} \\ \tau_{yx} & \sigma_y - \sigma_0 & \tau_{yz} \\ \tau_{zx} & \tau_{zy} & \sigma_z - \sigma_0 \end{bmatrix} = \begin{bmatrix} -9 & 9 & 5 \\ 9 & 1 & 0 \\ 5 & 0 & 8 \end{bmatrix}$$

$$J_2 = \frac{1}{6}\left[(\sigma_x - \sigma_y)^2 + (\sigma_y - \sigma_z)^2 + (\sigma_z - \sigma_x)^2 + 6(\tau_{xy}{}^2 + \tau_{yz}{}^2 + \tau_{zx}{}^2)\right]$$
$$= \left(10^2 + 7^2 + 17^2 + 6 \times (9^2 + 5^2)\right) \div 6 = 179$$

$$J_3 = \begin{vmatrix} \sigma_x - \sigma_0 & \tau_{xy} & \tau_{xz} \\ \tau_{yx} & \sigma_y - \sigma_0 & \tau_{yz} \\ \tau_{zx} & \tau_{zy} & \sigma_z - \sigma_0 \end{vmatrix} = \begin{vmatrix} -9 & 9 & 5 \\ 9 & 1 & 0 \\ 5 & 0 & 8 \end{vmatrix} = -745$$

$$\theta_\sigma = \frac{1}{3}\sin^{-1}\left[\frac{-\sqrt{27}J_3}{2J_2{}^{\frac{3}{2}}}\right] = \sin^{-1}\left[\frac{-\sqrt{27} \times -745}{2 \times 179^{\frac{3}{2}}}\right] \div 3 = 0.3137$$

$$\sigma_1 = \frac{2\sqrt{J_2}}{\sqrt{3}}\sin\left(\theta_\sigma + \frac{2\pi}{3}\right) + \sigma_0 = \frac{2\sqrt{179}}{\sqrt{3}}\sin\left(0.3137 + \frac{2\pi}{3}\right) - 1 = 9.3426$$

$$\sigma_2 = \frac{2\sqrt{J_2}}{\sqrt{3}}\sin(\theta_\sigma) + \sigma_0 = \frac{2\sqrt{179}}{\sqrt{3}}\sin(0.3137) - 1 = 3.7672$$

$$\sigma_3 = \frac{2\sqrt{J_2}}{\sqrt{3}}\sin\left(\theta_\sigma - \frac{2\pi}{3}\right) + \sigma_0 = \frac{2\sqrt{179}}{\sqrt{3}}\sin\left(0.3137 - \frac{2\pi}{3}\right) - 1 = -16.1098$$

$$\overline{\sigma} = \frac{1}{\sqrt{2}}\sqrt{(\sigma_1 - \sigma_2)^2 + (\sigma_2 - \sigma_3)^2 + (\sigma_3 - \sigma_1)^2} = \sqrt{5.5754^2 + 19.877^2 + 25.4524^2} \div \sqrt{2} = 23.1733$$

It is obvious to see from Table 6-4 that the subroutine returned a correct value in three-dimensional environment.

Table 6-4 Comparison of computer results and hand calculation results

Result	Computer	Hand calculation
Deviatoric stress in x-direction	-9.0000000E+00	-9
Deviatoric stress in y-direction	1.0000000E+00	1
Deviatoric stress in z-direction	8.0000000E+00	8
Deviatoric stress in xy-direction	9.0000000E+00	9
Deviatoric stress in yz-direction	0.0000000E+00	0
Deviatoric stress in zx-direction	5.0000000E+00	5
First principal stress	9.3424884E+00	9.3426
Second principal stress	3.7673038E+00	3.7672
Third principal stress	-1.6109791E+01	-16.1098
Equivalent stress	2.3173262E+01	23.1733

6.2 Subroutines of Numerical Integration Method

6.2.1 Instruction of NMScheck

Programme **NMScheck** was developed in order to test the numerical method subroutines. Figure 6-4 indicates the programme structure, and it could be found that the Euler's method, RK4 method, RKM method, and RKF method were distinguished through a case selection of **cas_sel_x**. For example, 'cas_sel_1' called the subroutine **EULER**, 'cas_sel_2' called the subroutine **RK4**, 'cas_sel_3' called the subroutine **RKM** and 'cas_sel_4' called the subroutine **RKF**. Those results will be written into a file named by the user. The user instruction is summarised below:

1. Create a new file named **xx.dat**;
2. The first line of input file is the type identifier of numerical method and loops counter, and the value of type identifier was restricted to '1' , '2' , '3' or '4';
3. The second line of input file is the initial value;
4. The third line of input file is the time-step;
5. Execute programme **NMScheck**
6. Enter **xx.dat** following the prompts;
7. Enter any output file name following the prompts;
8. Open the result file and see the result.

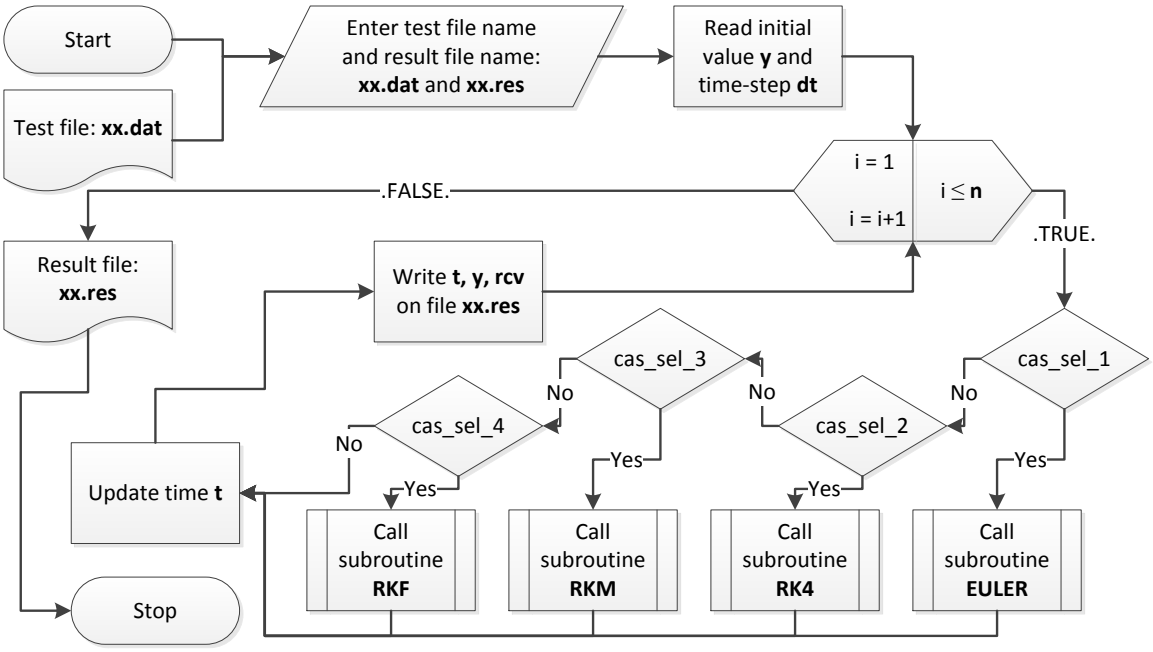

Figure 6-4 Algorithm of programme NMScheck

6.2.2 Statement of testing cases

The function $y' = y - t^2 + 1$ adopted from existing literature (Faires and Burden, 2013) was given to test numerical method subroutines. Such function was coded as a subroutine called **NTEST** according to the structure of constitutive equations subroutine. This testing was launched with given initial value Y_0 = 0.5 and time interval dt = 0.2. The result can present the preliminary understanding of accuracy of those numerical methods.

Subroutines **EULER, RK4, RKM** and **RKF** were tested separately, but the tested function is the same. Their computing results will be compared with the results obtained by existing literature (Faires and Burden, 2013). The used input files, which are EULER.dat, RK4.dat, RKM.dat and RKF.dat, are attached in appendix 10.4.2. The error was expected in the following situations:

1. If the error occurred on calling of an external function, then NMScheck will be aborted;
2. If the error occurred on the computing of slopes of function, then the final result will be wrong;
3. If the error occurred on the solution update, then the result will be wrong.

6.2.3 Result and verification

Figure 6-5 shows the result of each time moment of each numerical method. The result obtained from Euler's method is different with other three methods since the first step. The Euler's method has the error of almost 3.67% compared with the other methods. The results of other three numerical methods are very similar.

Euler's Method		RKM Method		RK4 Method		RKF Method	
time	y	time	y	time	y	time	y
0.20	0.8000000	0.20	0.8292987	0.20	0.8292933	0.20	0.8292985
0.40	1.1520000	0.40	1.2140880	0.40	1.2140762	0.40	1.2140875
0.60	1.5504000	0.60	1.6489412	0.60	1.6489220	0.60	1.6489403
0.80	1.9884800	0.80	2.1272307	0.80	2.1272027	0.80	2.1272291
1.00	2.4581760	1.00	2.6408609	1.00	2.6408227	1.00	2.6408586
1.20	2.9498112	1.20	3.1799444	1.20	3.1798942	1.20	3.1799409
1.40	3.4517734	1.40	3.7324042	1.40	3.7323401	1.40	3.7323992
1.60	3.9501281	1.60	4.2834898	1.60	4.2834095	1.60	4.2834828
1.80	4.4281538	1.80	4.8151848	1.80	4.8150857	1.80	4.8151751
2.00	4.8657845	2.00	5.3054839	2.00	5.3053630	2.00	5.3054705

Figure 6-5 Results obtained from EULER, RK4, RKM and RKF

The solutions of the testing function, which include exact solution and numerical solutions obtained from numerical methods, were given in Table 6-5. It is easy to find that,

all results are hundred percent correct through compare the Figure 6-5 with Table 6-5. Until now, the correctness of subroutine **EULER, RK4, RKM** and **RKF** were proved.

Table 6-5 Exact solution and numerical solution of numerical method of Euler's, RK4, RKM and RKF (Faires and Burden, 2013)

Time	Exact Solution	Euler's	RK4	RKM	RKF
0.0	0.5000000	0.5000000	0.5000000	0.5000000	0.5000000
0.2	0.8292986	0.8000000	0.8292933	0.8292987	0.8292985
0.4	1.2140877	1.1520000	1.2140762	1.2140880	1.2140875
0.6	1.6489406	1.5504000	1.6489220	1.6489412	1.6489403
0.8	2.1272295	1.9884800	2.1272027	2.1272307	2.1272291
1.0	2.6408591	2.4581760	2.6408227	2.6408609	2.6408586
1.2	3.1799415	2.9498112	3.1798942	3.1799444	3.1799409
1.4	3.7324000	3.4517734	3.7323401	3.7324042	3.7323992
1.6	4.2834838	3.9501281	4.2834095	4.2834898	4.2834828
1.8	4.8151763	4.4281538	4.8150857	4.8151848	4.8151751
2.0	5.3054720	4.8657845	5.3053630	5.3054839	5.3054705

6.3 Subroutines of Constitutive Equations

6.3.1 Instruction of CEScheck

Programme **CEScheck** was developed in order to test the constitutive equations subroutines. **Figure 6-6** indicates the programme structure, and it could be found that the KR equations, PH equations and QX equations were distinguished through a case selection of **cas_sel_x**, where 'cas_sel_1' called the subroutine **KR**, 'cas_sel_2' called the subroutine **PH** and 'cas_sel_3' called the subroutine **QX**. Only subroutine **RK4** is used in these tests because it does not include the part of self-adaptive approach but the accuracy is higher than subroutine **EULER**. Those results will be written into a file named by the user. The user instruction was summarised below:

1. Create a new file named **xx.dat**;
2. The first line of input file includes a) the type of constitutive equations, which '1' represented KR constitutive equations, '2' represented PH constitutive equations and '3' represented QX constitutive equations; b) the number of creep material property; c) the number of equations; d) number of stress terms; e) time-step; f) output loops; and g) output item;
3. The second line of input file is material properties;
4. The third line of input file is stress tensor components;
5. Execute programme **CEScheck;** Enter **xx.dat** following the prompts;

6. Enter any output file name following the prompts;
7. Open the result file and see the result.

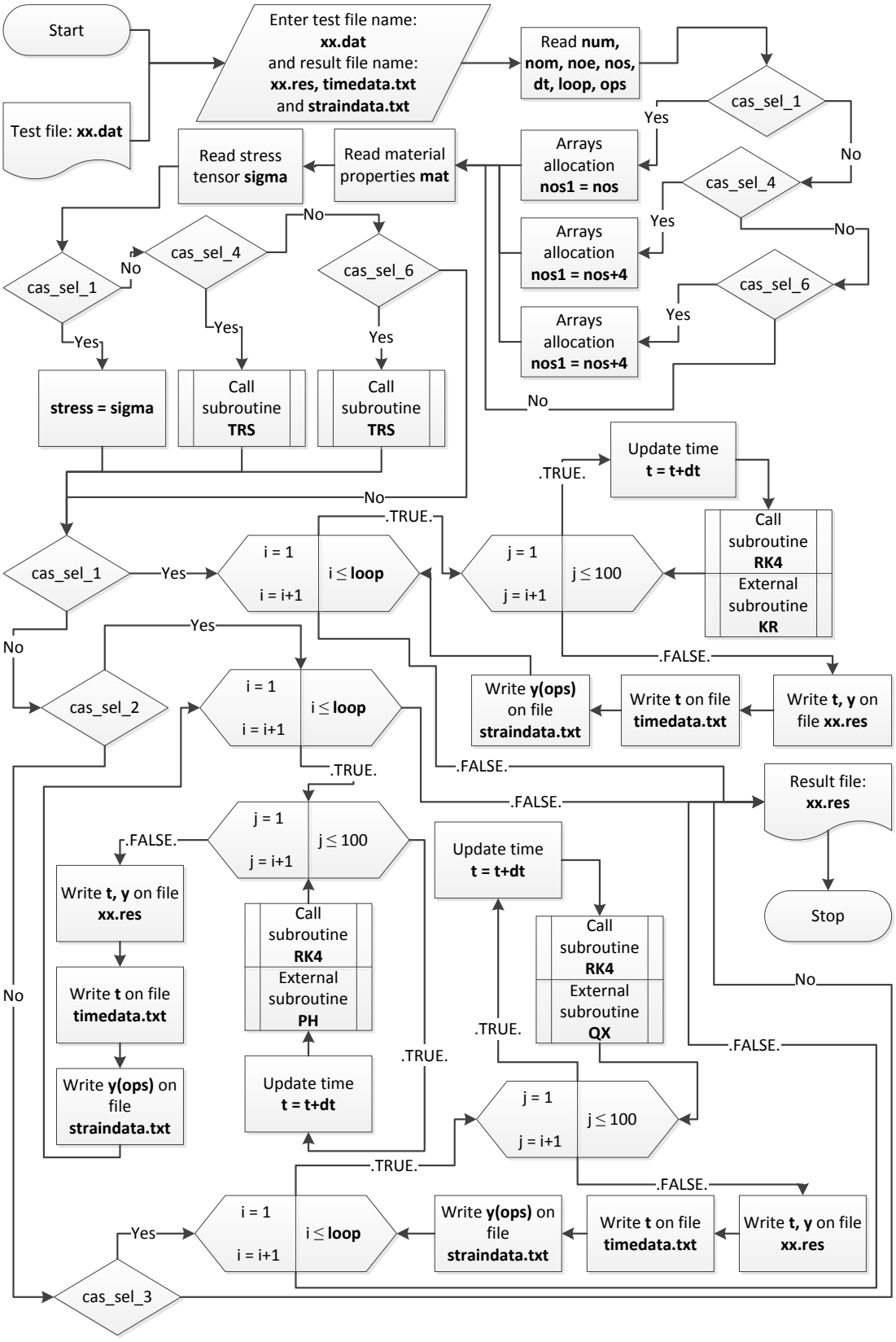

Figure 6-6 Algorithm of programme CEScheck

6.3.2 Statement of testing cases

Constitutive equations subroutines include both solutions of two-dimensional problem and three-dimensional problem; thus, the test of constitutive equations subroutines should be divided into three stages, which are uni-axial form testing, two-dimensional problem testing and three-dimensional problem testing[12]. All material properties used in these tests were adopted from the existing literature (Xu, 2001, Hyde et al., 2006). Table 6-6 indicates the detailed material properties for each creep constitutive equation.

Firstly, a single stress was given to test the performance of uni-axial equations of each subroutine (if applicable). Secondly, three sets of stresses included four components, which were obtained from Mohr's circle, were given to test each subroutine separately for two-dimensional problem. Finally, a set of stresses included six components, which direction will be changed according to the order of x-direction, y-direction and z-direction, were given to test each subroutine for three-dimensional problem.

Table 6-7 indicates the detailed value of stresses for each test of each creep constitutive equations. Due to this testing includes too much input files, a summary is given. Table 6-8 indicates the specific information of each input file to explain which test they have performed. Detailed input files are attached in appendix 10.4.3.

These files will be implemented separately, and their computing results will be compared with the creep strain curves obtained from existing literature. The error was expected in the following situations:

1. If the error occurred on the uni-axial form of constitutive equations; a) the result of creep strain and damage will be wrong when integration terminated, and b) the life time will have an obvious difference with the curves obtained from existing literature;
2. If the error occurred on the two-dimensional form of constitutive equations; a) the result of creep strain and damage based on the first set of stress will be wrong when integration terminated, and b) the their life time will not be the same;

[12] In here, the concepts of plane stress problem, plane strain problem and axisymmetric problem are not adopted due to all of them are considered as two-dimensional problem. For the subroutines of this research, those problems are the same because of the same number of stress terms (this research does not involve other more FEM knowledge unless otherwise stated).

3. If the error occurred on the three-dimensional form of constitutive equations, the three creep strain curves will not coincide.

Table 6-6 Material properties for each creep constitutive equations

Kachanov-Rabotnov constitutive equations						
A	**n**	**m**	**B**	**φ**	**χ**	**α**
1.092×10^{-20}	8.462	-4.754×10^{-4}	3.537×10^{-17}	7.346	6.789	0.215
Perrin-Hayhurst constitutive equations						
A	**B**	**h**	**K_c**	**H***	**C**	**ν**
6.216×10^{-8}	0.15	1.0×10^{4}	4.998×10^{-4}	0.35	2.0	1.32
Qiang Xu's constitutive equations						
A	**B**	**h**	**H***	**C**	**ν**	**a**
2.1618×10^{-9}	0.20524	2.4326×10^{5}	0.5929	1.8537	2.8	2
K_c	**b**	**p**	**q**			
9.2273×10^{-5}	2	2.5	1			

Table 6-7 Stress for each test of each creep constitutive equations

Uni-axial form		
KR	**PH**	
70MPa		
Two-dimensional problem		
	KR/PH	**QX**
0 degrees on the Mohr's circle	70MPa; 0MPa; 0MPa; 0MPa	60MPa; 0MPa; 0MPa; 0MPa
45 degrees on the Mohr's circle	59.74873734 MPa; 10.25126266	51.21320344MPa;
90 degrees on the Mohr's circle	35MPa; 35MPa; 35MPa; 0MPa	30MPa; 30MPa; 30MPa; 0MPa
Three dimensional problem		
	KR/PH	**QX**
Uni-axial on x-direction	70MPa; 0MPa; 0MPa; 0MPa;	60MPa; 0MPa; 0MPa; 0MPa;
Uni-axial on y-direction	0MPa; 70MPa; 0MPa; 0MPa;	0MPa; 60MPa; 0MPa; 0MPa;
Uni-axial on z-direction	0MPa; 0MPa; 70MPa; 0MPa;	0MPa; 0MPa; 60MPa; 0MPa;

Table 6-8 Summary information of each input file

File name	Description
KR1.dat; PH1.dat	Uni-axial creep constitutive equation test.
KR2.dat; KR3.dat; KR4.dat	Two-dimensional problem, included uni-axial case, biaxial case and pure shear case
PH2.dat; PH3.dat; PH4.dat	
QX1.dat; QX2.dat; QX3.dat	
KR5.dat; KR6.dat; KR7.dat	Three-dimensional problem, included x-direction uni-axial tensile, y-direction uni-axial tensile and z-direction uni-axial tensile
PH5.dat; PH6.dat; PH7.dat	
QX4.dat; QX5.dat; QX6.dat	

6.3.3 Result and verification

6.3.3.1 Testing of uni-axial form

The integration of constitutive equations will be terminated depending on a parameter that is named critical damage value. The theoretical critical damage value of KR equations is 1; however, when this value was employed by programme **CEScheck**, the result of damage is infinity. Through large amount of practices, a value of 0.85 was recommended, which is more suitable for the author's programme. Moreover, the value of 0.85 also has a limitation that the time-step must be less than 2 hours. Because the critical damage value and time-step is user-defined, the rupture strains obtained by this research are different with to that found in literature.

Figure 6-7 shows the result of uni-axial form of KR and PH. The lifetime of KR is almost 990 hours and the rupture strain is 0.18; and, the lifetime of PH is almost 980 hours and the rupture strain is 0.184. It is obvious to see that, these two kinds of constitutive equations have small distinction in the tendency of creep deformation.

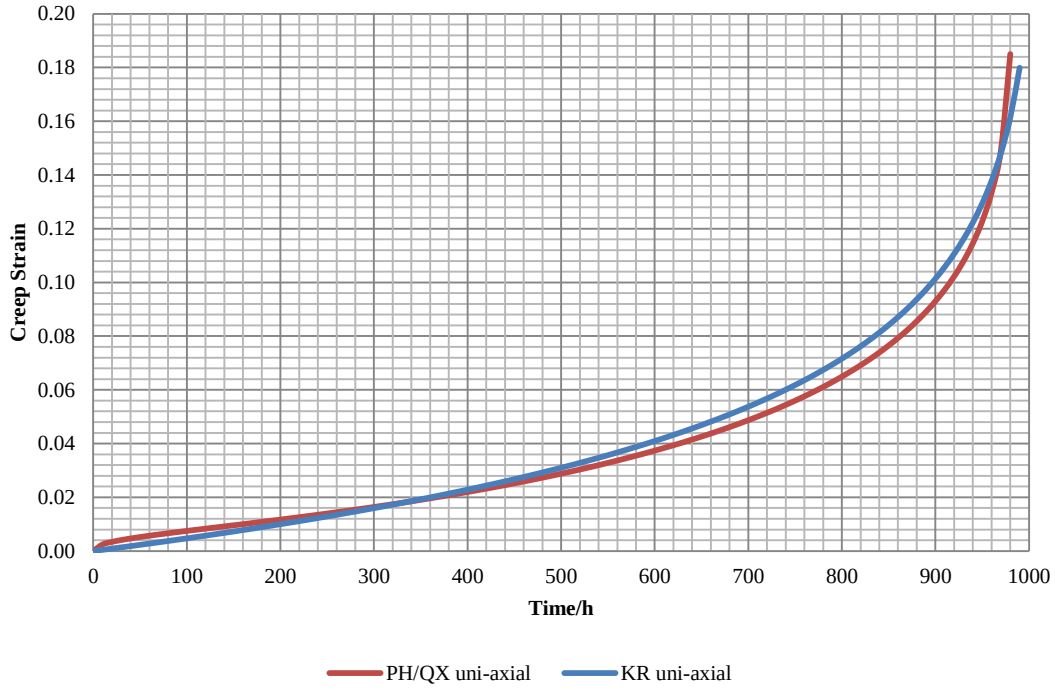

Figure 6-7 Creep strain curve of KR and PH based on single stress

Figure 6-8 presents the creep strain curves of KR and PH equations obtained from existing literature. Only the curves under 70MPa were used for the verification, and these two curves matched Figure 6-7 perfectly; for example, on KR curve, the creep strain was

accumulated to 0.1 after 900 hours, but on PH curve, the creep strain was accumulated to 0.1 after 920 hours. It could be said that the uni-axial forms of subroutine **KR** and **PH** are correct due to the outputted creep strain and damage being correct.

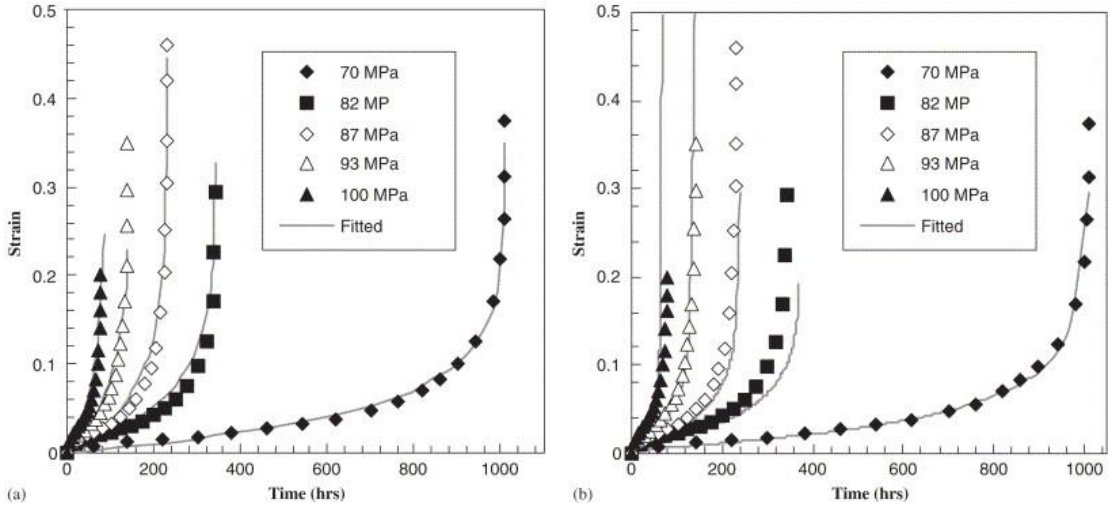

Figure 6-8[13] Creep strain curve obtained by Hyde; a) for KR, b) for PH (Hyde et al., 2006)

Testing of two-dimensional problem

Table 6-7 shows that, the stress tensor component on the x-direction is reduced according to the order of 0 degrees, 45 degrees and 90 degrees; hence, the corresponding creep strain in x-direction should be reduced according to the same sequence. To compare with the three curves shown on Figure 6-9, Figure 6-10 and Figure 6-12, it is easy to find that not only the creep strain was reduced but also the rupture time is the same. This phenomenon is correct because the loaded stresses are from the same Mohr's circle that has the same maximum principal stress and equivalent stress.

Due to the load of 0 degrees, Mohr's circle could be simply considered as a uni-axial tensile; hence the creep strain curve of 0 degrees on Mohr's circle should match the curve of uni-axial form. To compare with Figure 6-9 and Figure 6-8, Figure 6-10 and Figure 6-8, and Figure 6-12 and Figure 6-8, all curves of 0 degrees on Mohr's circle are matched with the curves of uni-axial form.

Figure 6-11 presents the creep strain curves of QX equations obtained from existing literature. Only the curve under the 60MPa was used for the verification, and the lifetime and creep strain of this curve are nearly 37000 hours and 0.18.

[13] In figure 6-8, the curve is fitted line, only the data under 70MPa both of a) and b) are used to verify the corresponding curve presented on figure 6-7.

It could be said that the two-dimensional forms of subroutine **KR, PH** and **QX** are correct due to not only the outputted creep strain and damage based on 0 degrees on Mohr's circle being correct, but also the reducing tendency of creep strain is matched.

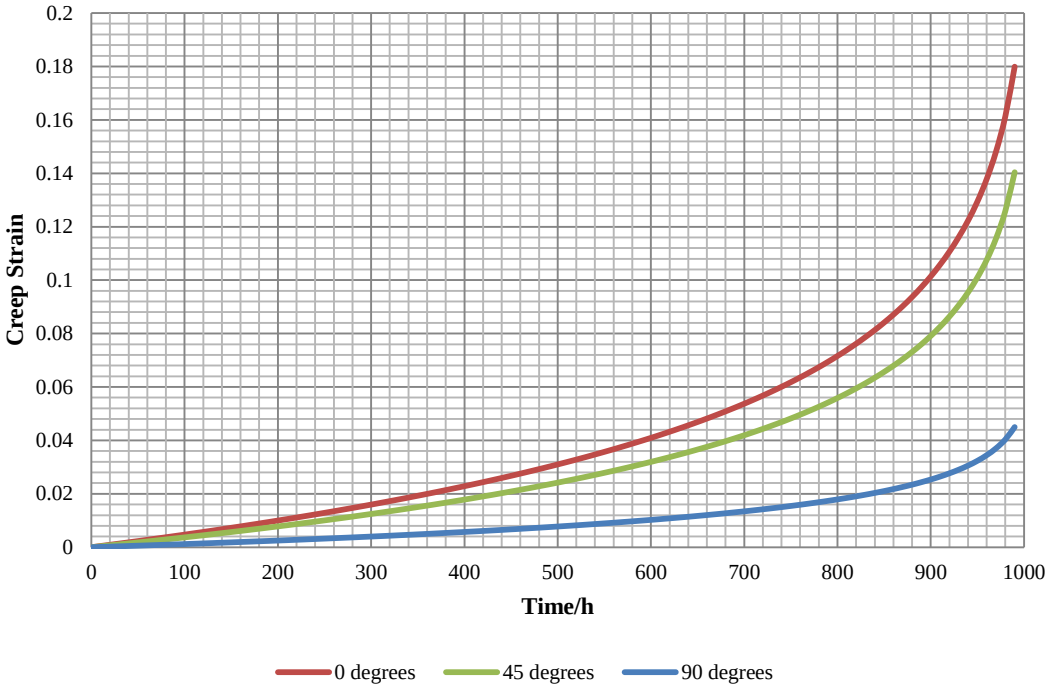

Figure 6-9 Creep strain curve of KR based on the stresses obtained from Mohr's circle in x-direction

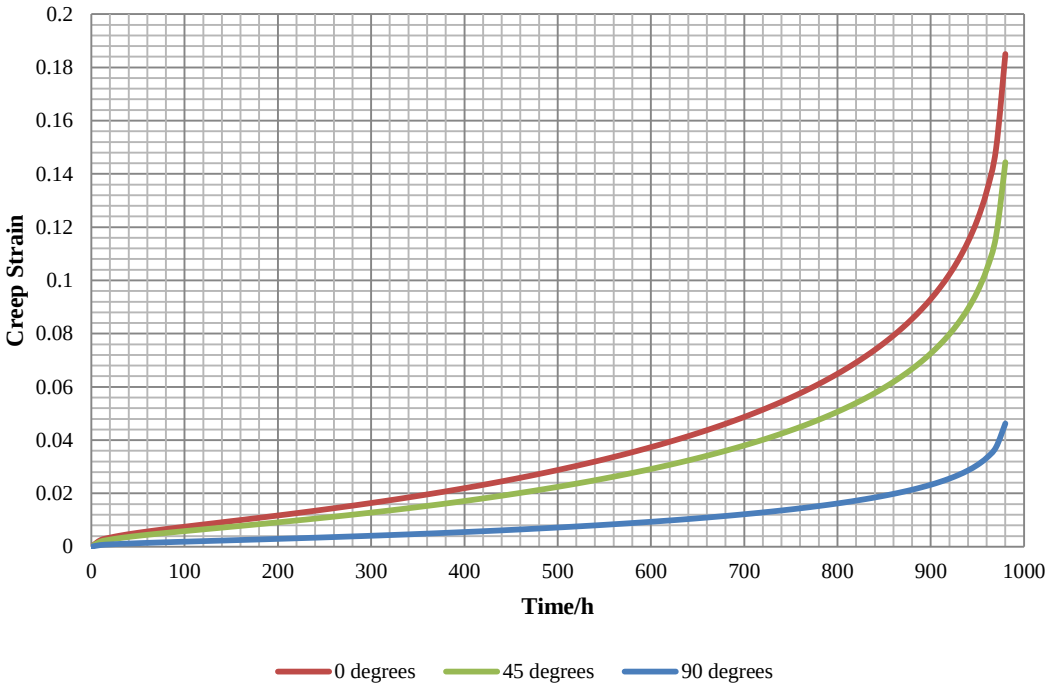

Figure 6-10 Creep strain curve of PH based on the stresses obtained from Mohr's circle in x-direction

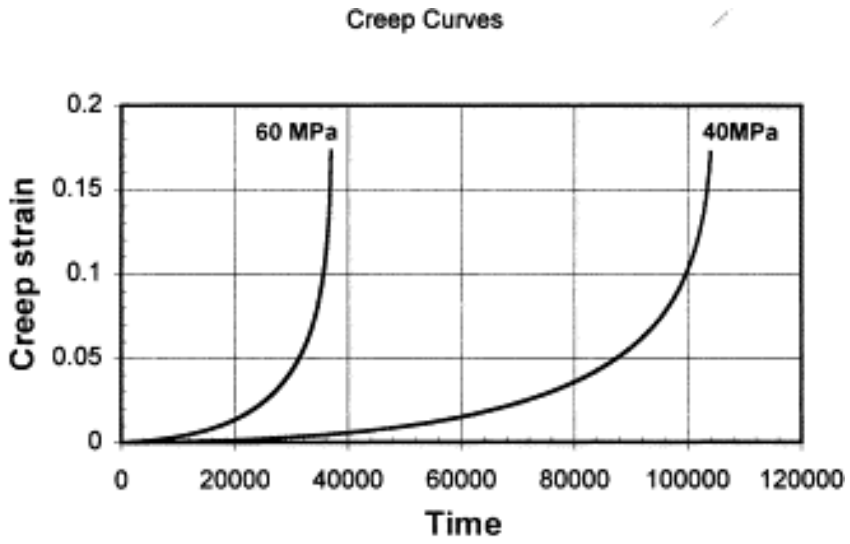

Figure 6-11 Creep strain curve obtained by Qiang Xu (Xu, 2001)

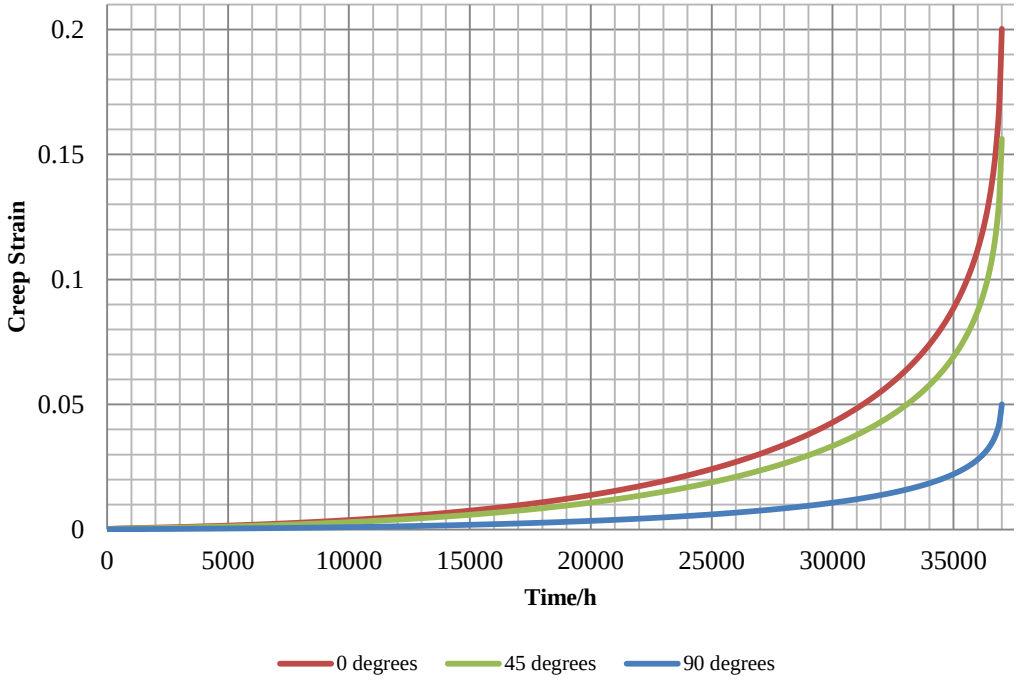

Figure 6-12 Creep strain curve of QX based on the stresses obtained from Mohr's circle in x-direction

Testing of three-dimensional problem

Table 6-7 shows that, the loaded stresses could be considered as the uni-axial tensile on x-direction, y-direction and z-direction respectively; hence, the corresponding creep strain in x-direction, y-direction and z-direction should be the same.

To compare with the three curves shown on Figure 6-13, Figure 6-14 and Figure 6-15, it is easy to find that those creep strain curves of x-direction, y-direction and z-direction are

hundred percent matched. This phenomenon is correct because the loaded stresses could be seen as the same, which only direction is different.

Due to the loads of x-direction, y-direction and z-direction could be simply considered as the same uni-axial tensile; hence, their creep strain curves of each direction should match the curve of uni-axial form. Compared with Figure 6-13 and Figure 6-8, all curves of x-direction, y-direction and z-direction are matched with the corresponding curves of uni-axial form. The same comparison can be used for Figure 6-14 and Figure 6-8; Figure 6-15 and Figure 6-8.

It could be said that the three-dimensional forms of subroutine **KR, PH** and **QX** are correct due to not only the outputted creep strain and damage based on x-direction, y-direction and z-direction is correct, but also the creep strain curves were coincided with each other.

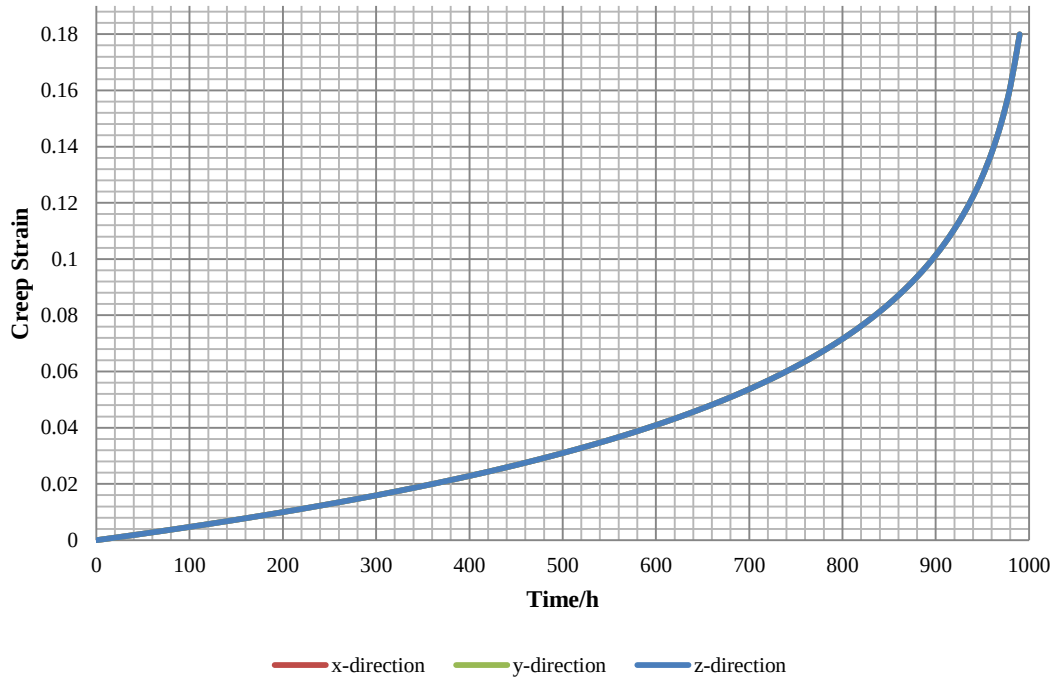

Figure 6-13 Creep strain curve of KR based on the uni-axial tensile load in x-direction, y-direction and z-direction

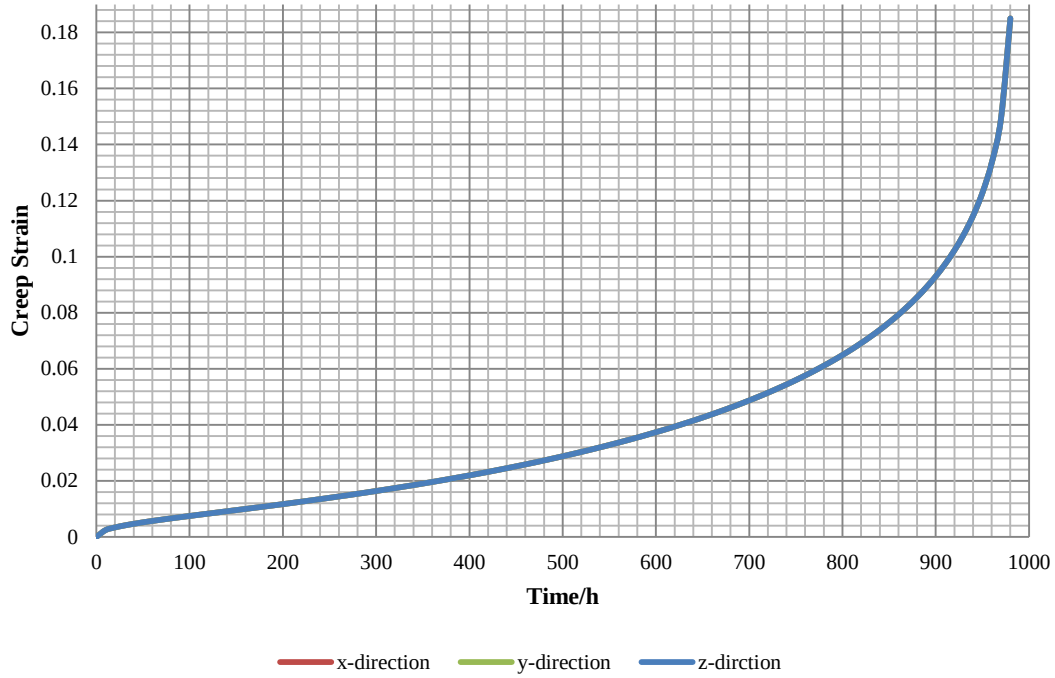

Figure 6-14 Creep strain curve of PH based on the uni-axial tensile load in x-direction, y-direction and z-direction

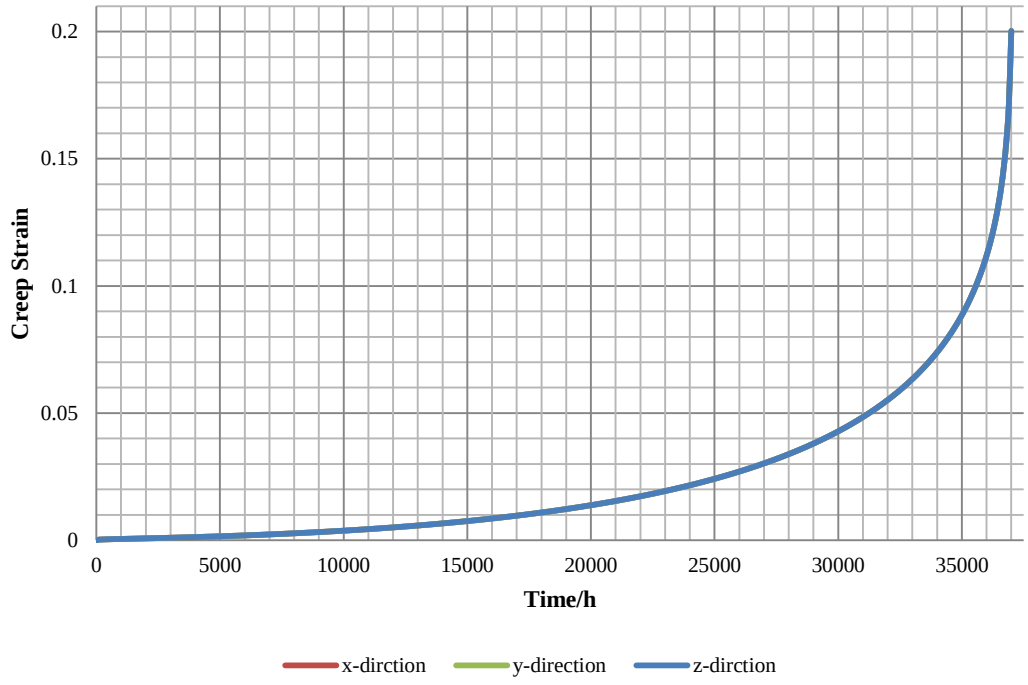

Figure 6-15 Creep strain curve of QX based on the uni-axial tensile load in x-direction, y-direction and z-direction

6.4 Subroutine of Time-step Control Procedure

6.4.1 Instruction of TSCcheck

Programme **TSCcheck** was developed in order to test the time-step control subroutines. Figure 6-16 indicated the programme structure, and it could be found that the RKM method and RKF method were distinguished through a case selection of **cas_sel_x**; 'cas_sel_1' called the subroutine **RKM** and 'cas_sel_2' called the subroutine **RKF**. PH is the only constitutive equations subroutine used in these tests in order to keep the same integrand. Those results will be written into a file named by the user. The user instruction is summarised below:

1. Create a new file named **xx.dat**;
2. The first line of input file includes a) the type of numerical method, which '1' represented Runge-Kutta-Merson method and '2' represented Runge-Kutta-Fehlberg method; b) the number of creep material property; c) the number of equations; d) number of stress terms; e) time-step; and f) output loops;
3. The second line of input file is material properties;
4. The third line of input file is stress tensor components;
5. Execute programme **TSCcheck**
6. Enter **xx.dat** following the prompts;
7. Enter any output file name following the prompts;
8. Open the result file and see the result.

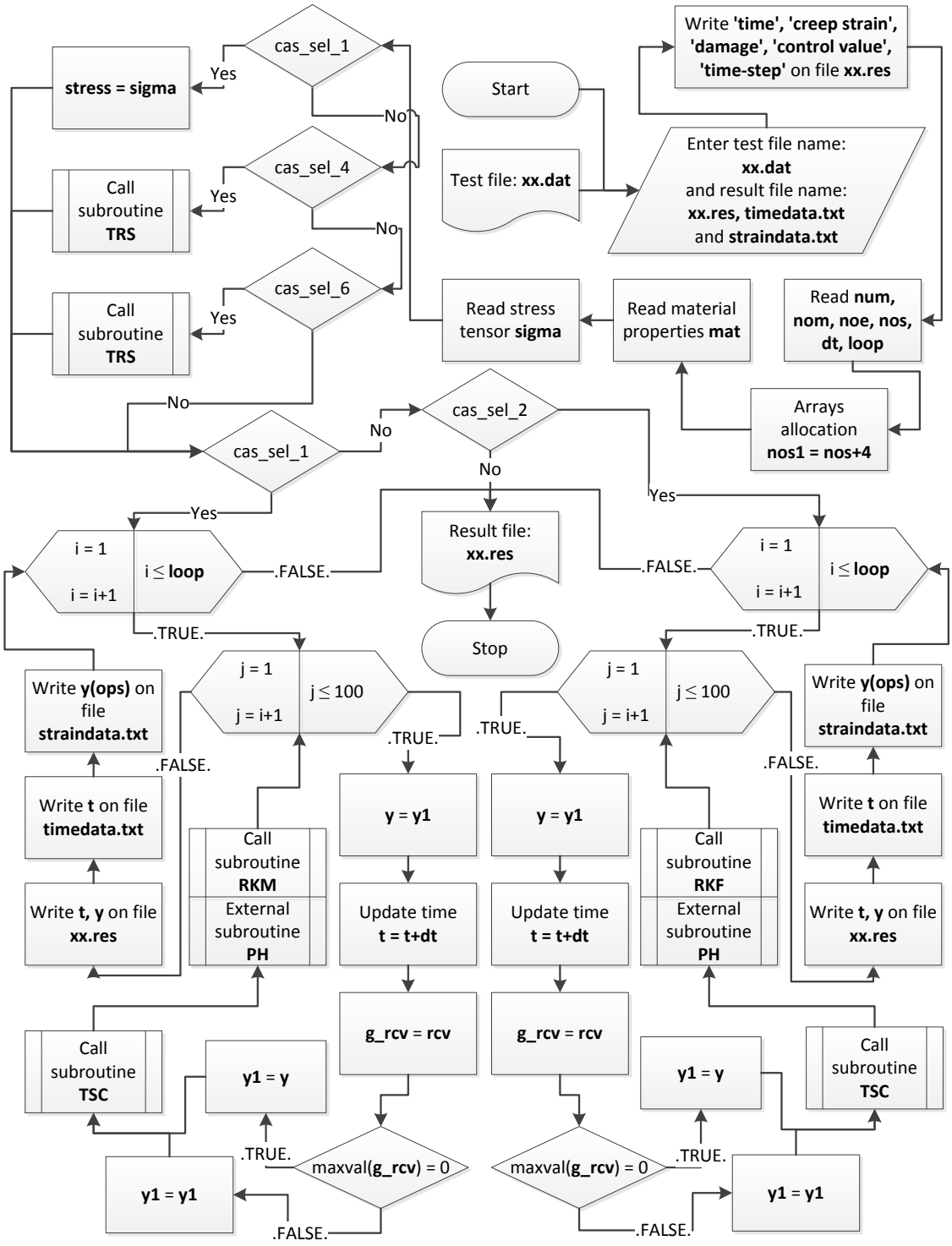

Figure 6-16 Algorithm of programme TSCcheck

6.4.2 Statement of testing cases

Two tests based on subroutine **RKM** and subroutine **RKF** were executed separately. The testing of time-step control module includes the test of performance of subroutine **TSC** and the test of self-adaptive function of subroutine **RKM** and **RKF**. The subroutine **PH** was adopted here to assist the numerical method subroutine.

The loaded stress is 70MPa and time-step is 0.1 hour. All material properties used in these tests were adopted from the existing literature (Xu, 2001, Hyde et al., 2006). Table 6-9 indicated the detailed material properties for each creep constitutive equation.

These two sets of data will be implemented separately, and their computing results will be compared by each other. The error was expected in the following situations:

1. If the error occurred on the subroutine **TSC**, the time-step will not be changed;
2. If the error occurred on the self-adaptive part of subroutine **RKM** and **RKF**, the time-step will not be changed as well.

Table 6-9 Material properties of Perrin-Hayhurst constitutive equations

Perrin-Hayhurst constitutive equations						
A	**B**	**h**	**K_c**	**H***	**C**	**ν**
6.216×10^{-8}	0.15	1.0×10^{4}	4.998×10^{-4}	0.35	2.0	1.32

6.4.3 Result and verification

6.4.3.1 Testing of subroutine TSC and RKM

Figure 6-17 and Figure 6-18 indicates creep strain, creep damage and time-step obtained from RKM method and RKF method respectively. It could be observed from Figure 6-17, the change of time-step is obvious based on the RKM method; however, it is easy to see that, all changes were happened after 860 hours. Through the Figure 6-7, the time of 860 hours is almost the starting of tertiary stage of creep deformation due to the creep strain has a rapid accumulation. This change tendency of time-step satisfies the basic physical and mathematical theories.

```
  time  creep strain    damage  control value time-step
 10.00 2.4646001E-03 4.9292005E-03    0 1.0000000E-01
..............
860.00 7.9288110E-02 1.5857623E-01    0 1.0000000E-01
865.55 8.0896113E-02 1.6179223E-01    0 5.0000000E-02
..............
920.55 1.0248352E-01 2.0496704E-01    0 5.0000000E-02
923.28 1.0392101E-01 2.0784203E-01    0 2.5000000E-02
..............
953.28 1.2574231E-01 2.5148464E-01    0 2.5000000E-02
955.46 1.2798373E-01 2.5596747E-01    0 1.2500000E-02
..............
970.46 1.4994061E-01 2.9988123E-01    0 1.2500000E-02
971.16 1.5142039E-01 3.0284079E-01    0 6.2500000E-03
..............
977.41 1.6993857E-01 3.3987716E-01    0 6.2500000E-03
```

Figure 6-17 Creep strain and damage, time-step obtained from RKM method

6.4.3.2 Testing of subroutine TSC and RKF

It could be observed from Figure 6-18, the change of time-step did not happen based on the RKF method; however, it is easy to see that, the final creep strain, creep damage and lifetime are correct. This reason is investigated in Chapter 7.

Figure 6-19 presents the creep strain curves obtained from this testing, which are based on RKM method and RKF method, coincided with each other. To compare with the PH curve shown in Figure 6-7, both the results obtained from this testing are correct. Therefore, the accuracy of RKF method was proved.

Figure 6-20 shows error between RKM method and RKF method, the error increased following the time increasing, but the absolute value of error is very small, and could be ignored.

Until now, the correctness of subroutine **TSC** and the self-adaptive function of subroutine **RKM** and **RKF** were proved. Figure 6-20 provides the primary understanding of the performance of two self-adaptive approaches.

time	creep strain	damage	control value	time-step
10.00	2.4646004E-03	4.9292009E-03	0	1.0000000E-01
.....				
110.00	7.9061874E-03	1.5812375E-02	0	1.0000000E-01
.....				
310.00	1.6905416E-02	3.3810834E-02	0	1.0000000E-01
.....				
410.00	2.2582907E-02	4.5165816E-02	0	1.0000000E-01
.....				
860.00	7.9288101E-02	1.5857621E-01	0	1.0000000E-01
.....				
940.00	1.1455693E-01	2.2911388E-01	0	1.0000000E-01
950.00	1.2273984E-01	2.4547969E-01	0	1.0000000E-01
960.00	1.3344732E-01	2.6689464E-01	0	1.0000000E-01
970.00	1.4934216E-01	2.9868433E-01	0	1.0000000E-01
980.00	1.8501404E-01	3.7002809E-01	0	1.0000000E-01

Figure 6-18 Creep strain and damage, time-step obtained from RKF method

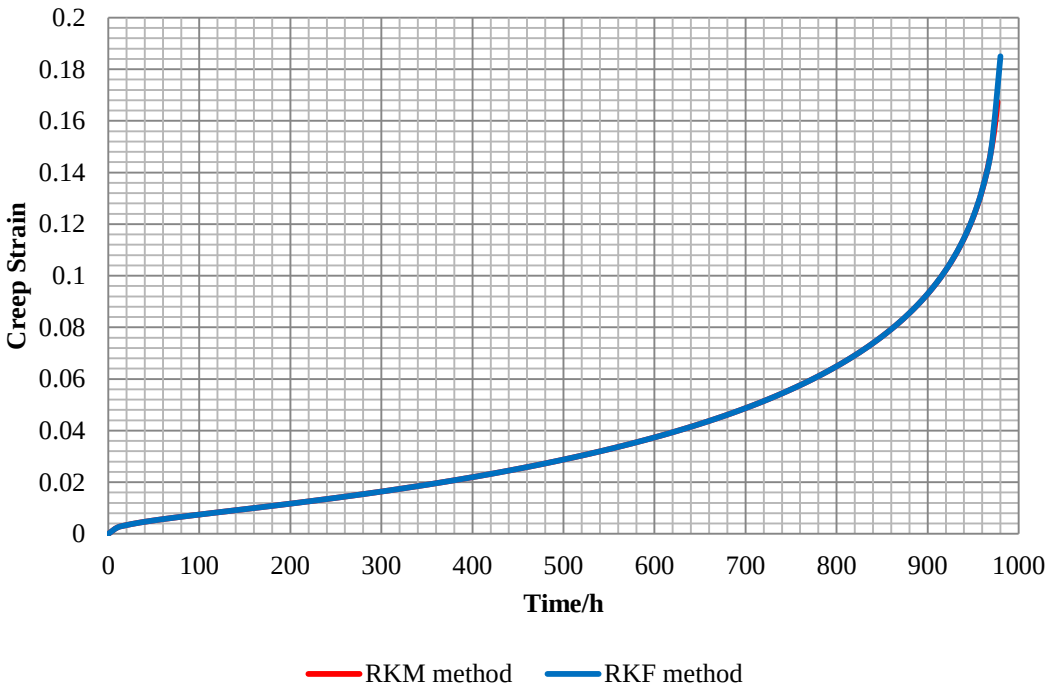

Figure 6-19 Creep strain curve comparison based on RKM method and RKF method

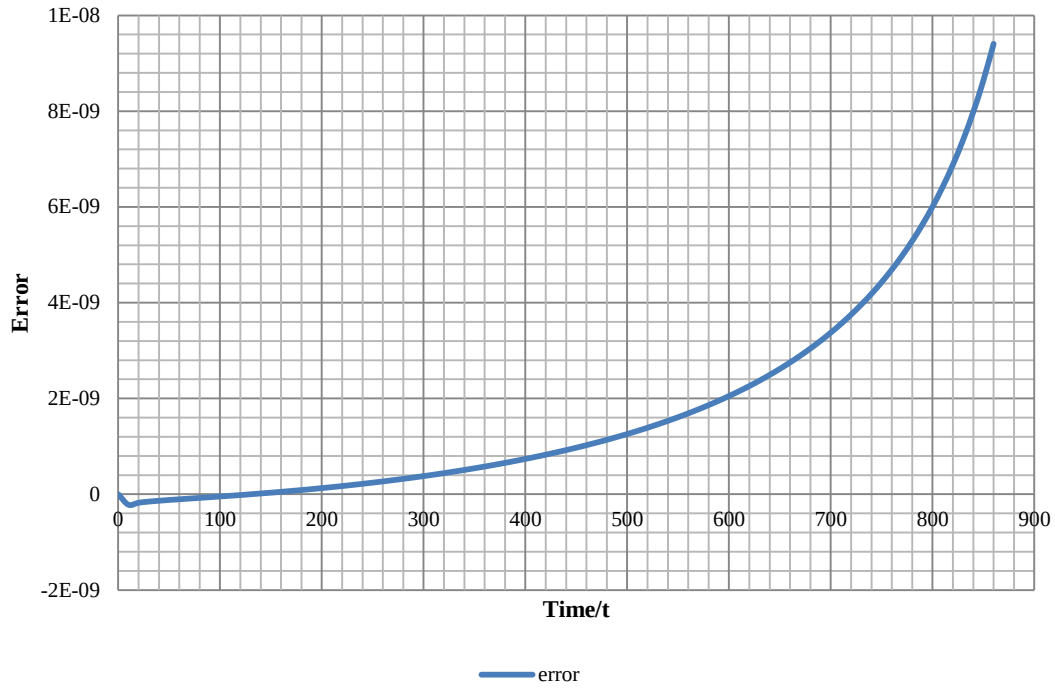

Figure 6-20 Error between RKM and RKF

6.5 Subroutine of Normalisation technique

6.5.1 Instruction of NORcheck

Programme **NORcheck** was developed in order to test the constitutive equations subroutines. Figure 6-21 indicated the programme structure, and it could be found that the two-dimensional problem and three-dimensional problem were distinguished through a case selection of **cas_sel_x**; 'cas_sel_9' is two-dimensional problem and 'cas_sel_11' is three-dimensional problem. **RK4** is the only numerical method subroutine used in these tests because it does not include the part of self-adaptive approach but the accuracy is higher than **EULER** is. Those results will be written into a file named by the user.

The user instruction was summarised below:

1. Create a new file named **xx.dat**;
2. The first line of input file includes a) the number of creep material property; b) the number of equations; c) number of stress terms; d) time-step; e) output loops; f) output item; and g) stress used for normalization;
3. The second line of input file is material properties;
4. The third line of input file is stress tensor components;

5. Execute programme **NORcheck**
6. Enter **xx.dat** following the prompts;
7. Enter any output file name following the prompts;
8. Open the result file and see the result.

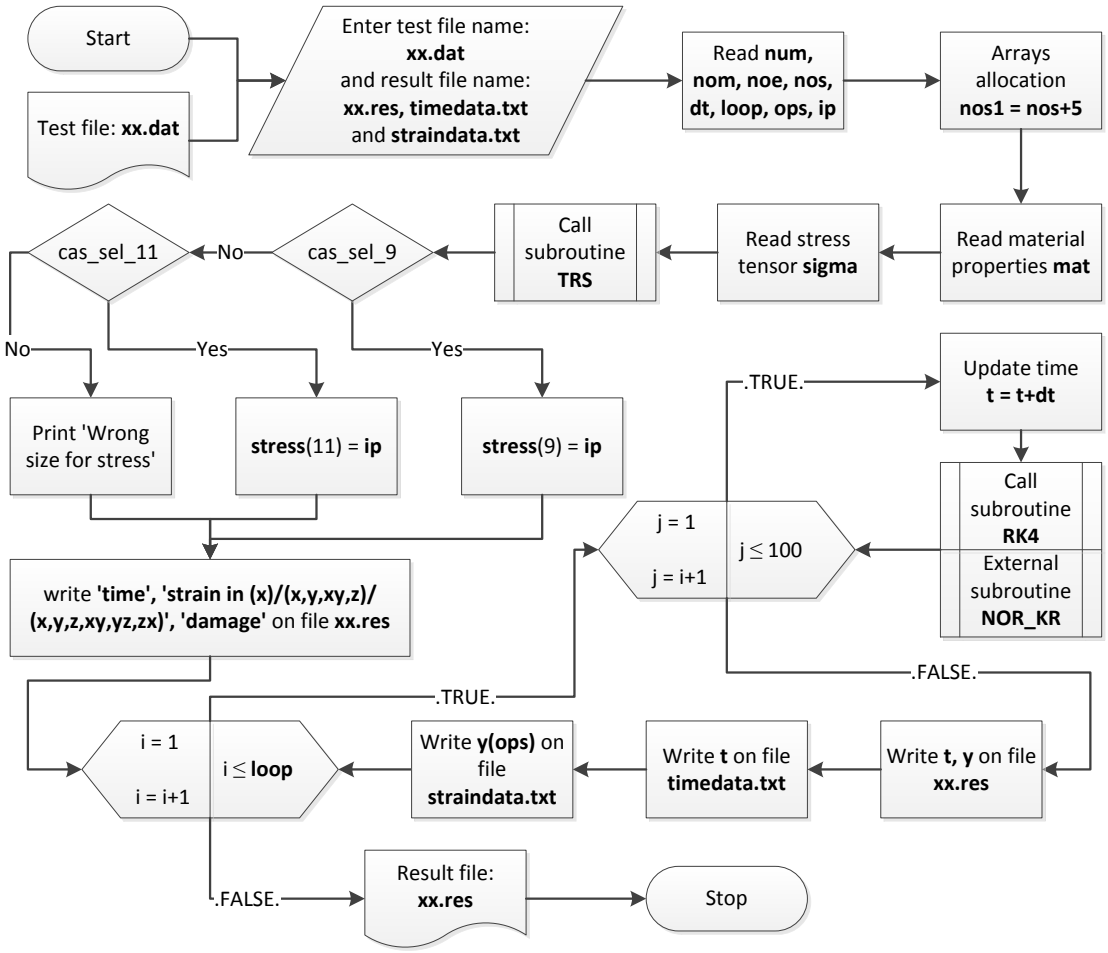

Figure 6-21 Algorithm of programme NORcheck

6.5.2 Statement of testing cases

Normalization subroutine includes both solutions of two-dimensional problem and three-dimensional problem; thus, the test of normalization subroutine should be divided into two stages, which are two-dimensional problem testing and three-dimensional problem testing. All material properties used in these tests were adopted from the existing literature (Xu, 2001, Hyde et al., 2006). Table 6-10 indicates the detailed material properties for normalized creep constitutive equations.

Firstly, three sets of stresses included four components, which were obtained from Mohr's circle, were used to test each subroutine separately for the two-dimensional problem.

Secondly, a set of stresses included six components, whose direction will be changed according to the order of x-direction, y-direction and z-direction, were given to test each subroutine for the three-dimensional problem. Table 6-11 indicates the detailed value of stresses for each test of each creep constitutive equations.

The error was expected in following situations:

1. If the error occurred on the two-dimensional form of constitutive equations; a) the result of creep strain and damage based on the first set of stress will be wrong when integration terminated, and b) the their life time will not be same;
2. If the error occurred on the three-dimensional form of constitutive equations, the three creep strain curves will not coincide.

Table 6-10 Material properties used to the test of normalized creep constitutive equations subroutine

NOR_KR	**A**	**n**	**m**	**B**
	1.092×10^{-20}	8.462	-4.754×10^{-4}	3.537×10^{-17}
	α	**φ**	**χ**	**E**
	0.215	7.346	6.789	170×10^{3}

Table 6-11 Stress for each test of normalized creep constitutive equations

Two-dimensional problem	
0 degrees on the Mohr's circle	70MPa; 0MPa; 0MPa; 0MPa
45 degrees on the Mohr's circle	59.74873734 MPa; 10.25126266 MPa; 24.74873734 MPa; 0MPa
90 degrees on the Mohr's circle	35MPa; 35MPa; 35MPa; 0MPa
Three dimensional problem	
Uni-axial on x-direction	70MPa; 0MPa; 0MPa; 0MPa; 0MPa; 0MPa
Uni-axial on y-direction	0MPa; 70MPa; 0MPa; 0MPa; 0MPa; 0MPa
Uni-axial on z-direction	0MPa; 0MPa; 70MPa; 0MPa; 0MPa; 0MPa

Result and verification

Table 6-7 shows that, the stress tensor component on the x-direction is reduced according to the order of 0 degrees, 45 degrees and 90 degrees; hence, the corresponding creep strain in x-direction should be reduced according to the same sequence. To compare with the three curves shown on Figure 6-22, it is easy to find that not only the creep strain was reduced but also the rupture time is the same. This phenomenon is correct because the loaded stresses are from the same Mohr's circle that has the same maximum principal stress and equivalent stress.

Table 6-7 shows that, the loaded stresses could be considered as the uni-axial tensile on x-direction, y-direction and z-direction respectively; hence, the corresponding creep strain in x-direction, y-direction and z-direction should be the same. To compare with the three curves shown on Figure 6-23, it is easy to find that those creep strain curves of x-direction, y-direction and z-direction are exactly matched. This phenomenon is correct because the loaded stresses could be seen as the same, in which only the direction is different.

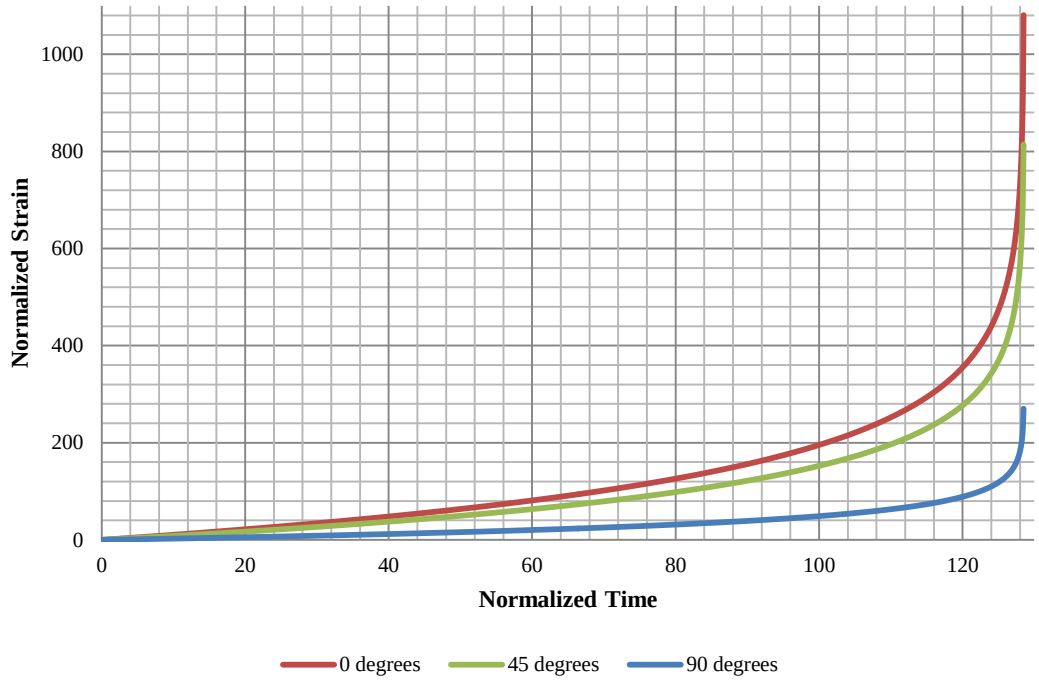

Figure 6-22 Creep strain curve of normalized constitutive equations based on the stresses obtained from Mohr's circle in x-direction

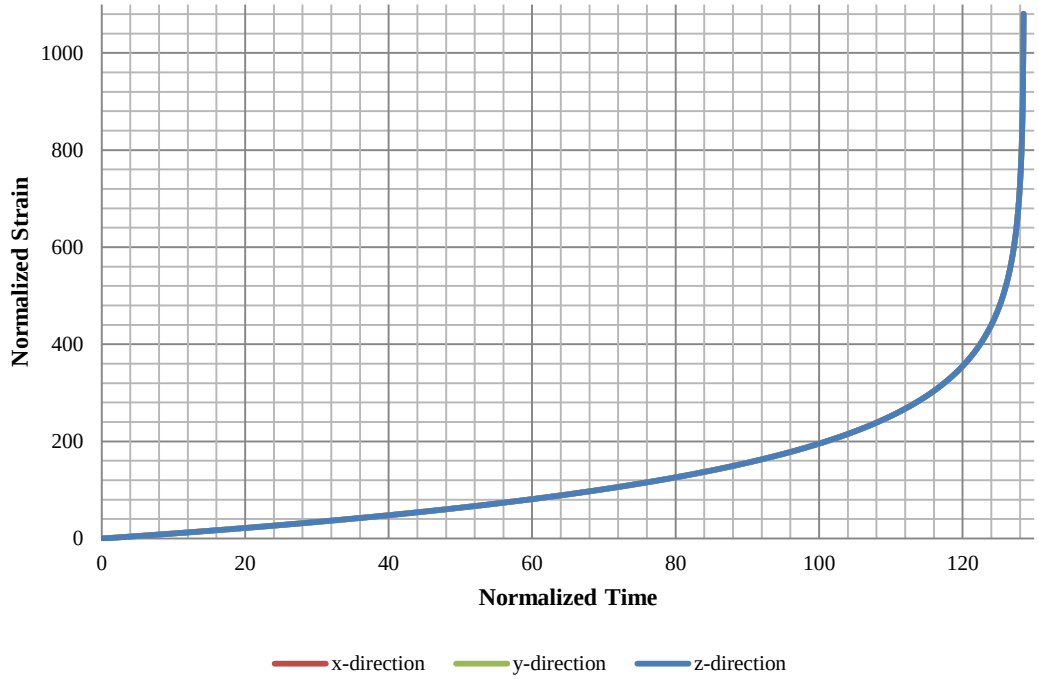

Figure 6-23 Creep strain curve of normalized constitutive equations based on the uni-axial tensile load in x-direction, y-direction and z-direction

6.6 Nodal Loads Calculator

6.6.1 Instruction of NLC

The nodal loads calculator is an independent programme; hence, it does not need to design a specific testing programme. The user instruction was summarised below:

1. Create a new file named **xx.dat**;
2. The first line of input file is the number of element;
3. The second line of input file is the expected pressure;
4. The third line of input file is the coordinates of loaded direction of the first loaded node of each loaded element;
5. The fourth line of input file is the coordinates of loaded direction of the second loaded node of each loaded element;
6. Execute programme **NLC**;
7. Enter **xx.dat** following the prompts;
8. Enter any output file name following the prompts;
9. Open the result file and see the result.

6.6.2 Statement testing cases

NLC has its restriction of element type. Only 3-nodes triangle element and 4-nodes quadrilateral element can be resolved within general planar and axisymmetric cases; hence, a 700N uniform force was set as the expected load. Two elements and three loaded nodes were supposed and the coordinates of loaded node are (0, 60), (20, 60) and (40, 60).

6.6.3 Result and verification

Figure 6-24 shows nodal forces loaded on axisymmetric and planar elements. The nodal forces of axisymmetric element are 4666.6N, 28000N and 23333N respectively. The nodal forces of planar element are 700N, 1400N and 700N respectively.

```
  axisymmetric result
Nodal Force From Inner to Outer:
   1 4.6666667E+03
   2 2.8000000E+04
   3 2.3333333E+04

  planar result
Nodal Force From First to Last:
   1 7.0000000E+02
   2 1.4000000E+03
   3 7.0000000E+02
```

Figure 6-24 Nodal forces depend on axisymmetric case and planar case

In order to verify those results, a hand calculation was conducted:

6.6.3.1 Planar problem

The nodal force loaded on first node of each element can be derived as,

$$f_{1\ node1} = \frac{(20-0)\times 70}{2} = 700$$

$$f_{1\ node2} = \frac{(40-20)\times 70}{2} = 700$$

The nodal force loaded on second node of each element can be derived as,

$$f_{2\ node1} = \frac{(40-20)\times 70}{2} = 700$$

$$f_{2\ node2} = \frac{(60-40)\times 70}{2} = 700$$

Since the first node of second element and second node of first element is the same node;

$$f_{\text{shared node}} = f_{1\ \text{node2}} + f_{2\ \text{node1}} = 1400$$

These three nodal forces can be summarised in the sequence 700N, 1400N and 700N.

6.6.3.2 Axisymmetric problem

The nodal force loaded on first node of each element can be derived as,

$$f_{1\ \text{node1}} = \frac{1}{6}({r_1}^2 + r_0 r_1 - 2{r_0}^2) \times P = \frac{1}{6} \times (20^2 + 0 \times 20 - 2 \times 0^2) \times 70 = 4666.669$$

$$f_{1\ \text{node2}} = \frac{1}{6}\left(2{r_1}^2 - r_0 r_1 - {r_0}^2\right) \times P = \frac{1}{6} \times (2 \times 20^2 - 0 \times 20 - 0^2) \times 70 = 9333.331$$

The nodal force loaded on second node of each element can be derived as,

$$f_{2\ \text{node1}} = \frac{1}{6}({r_1}^2 + r_0 r_1 - 2{r_0}^2) \times P = \frac{1}{6} \times (40^2 + 20 \times 40 - 2 \times 20^2) \times 70 = 18666.669$$

$$f_{2\ \text{node2}} = \frac{1}{6}\left(2{r_1}^2 - r_0 r_1 - {r_0}^2\right) \times P = \frac{1}{6} \times (2 \times 40^2 - 20 \times 40 - 20^2) \times 70 = 23333.331$$

Since the first node of second element and second node of first element is the same node; hence,

$$f_{\text{shared node}} = f_{1\ \text{node2}} + f_{2\ \text{node1}} = 28000$$

These three nodal forces can be summarised in the sequence 4666.669N, 28000N and 23333.331N.

To compare the hand calculation results and the results obtained from NLC, those two are the same; thus, the correctness of programme NLC was proved.

6.7 Data Transfer Interface

6.7.1 Instruction of DTI

The data transfer interface is an independent programme; hence, it does not need to design a specific testing programme. The input files of **DTI** include the Neutral file produced by FEMGV with the suffix of '.anl' and the data file produced by the solver with the suffix of '.res'. The execution of programme **DTI** is very simple, requiring just clicking and

entering information following the prompts. The required information includes a) the type of transfer process, which is restricted to the character of ‘pre-processing’ or ‘post-processing’; and b) the name of input and output file. The output files of **DTI** include the Neutral file produced for FEMGV with the suffix of ‘.fvi’ and the data file produced for the solver with the suffix of ‘.dat’.

6.7.2 Statement of testing cases

Two cases were implemented in this testing, one is for pre-processing transfer and the other is for post-processing transfer. Six elements geometry was created by FEMGV shown on the Figure 6-25. This geometry includes twelve nodes, 20mm in width and 60mm in height; its bottom was restricted in y-direction and left side was restricted in x-direction. A file called DTI.res, which is used to test the post-processing transfer, is attached in appendix 10.4.7.

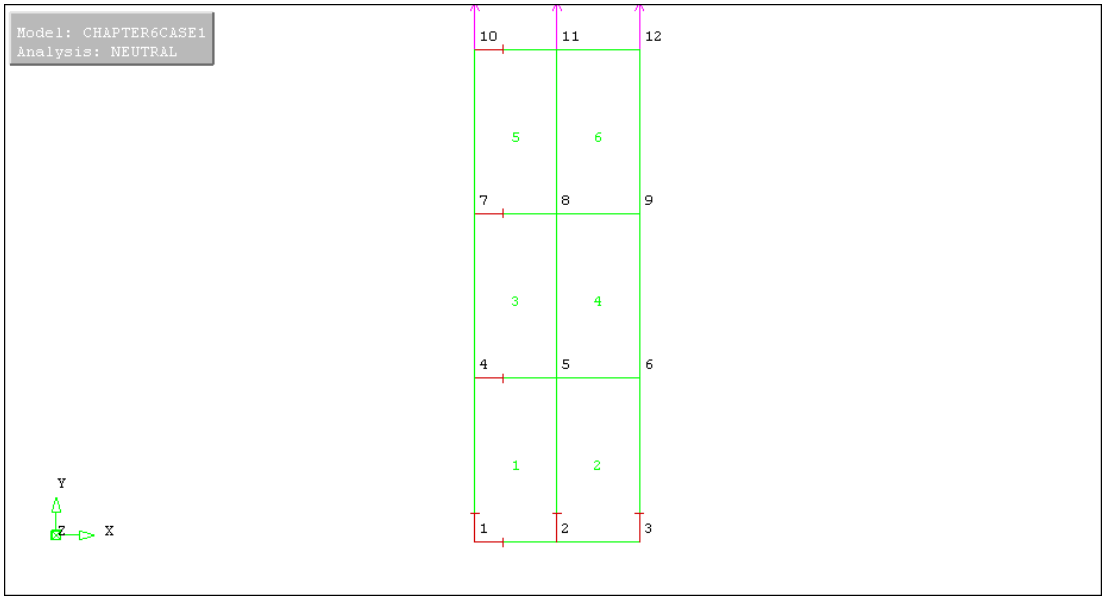

Figure 6-25 Geometry model used for pre-processing transfer test

6.7.3 Result and verification

The nodal force was applied on the top. The coordinates of this geometry was shown in the Table 6-12. Figure 6-26 presented the transfer result of pre-processing. To compare with Table 6-12, Figure 6-25 and Figure 6-26, it could be observed that, the node coordinate, element definition and constrains has been transferred directly and correctly; however, the nodal loads part needs need the help of NLC

Table 6-12 Coordinates of geometry used for pre-processing transfer

node	x-direction	y-direction	z-direction
1	0.0000000E+00	0.0000000E+00	0.0000000E+00
2	1.0000000E+01	0.0000000E+00	0.0000000E+00
3	2.0000000E+01	0.0000000E+00	0.0000000E+00
4	0.0000000E+00	2.0000000E+01	0.0000000E+00
5	1.0000000E+01	2.0000004E+01	0.0000000E+00
6	2.0000000E+01	2.0000000E+01	0.0000000E+00
7	0.0000000E+00	4.0000000E+01	0.0000000E+00
8	1.0000000E+01	4.0000000E+01	0.0000000E+00
9	2.0000000E+01	4.0000000E+01	0.0000000E+00
10	0.0000000E+00	6.0000000E+01	0.0000000E+00
11	1.0000000E+01	6.0000000E+01	0.0000000E+00
12	2.0000000E+01	6.0000000E+01	0.0000000E+00

```
quadrilateral             6          12
           1   0.0000000000000000          0.0000000000000000
           2   10.000000000000000          0.0000000000000000
           3   20.000000000000000          0.0000000000000000
           4   0.0000000000000000          20.000000000000000
           5   10.000000000000000          20.000004000000001
           6   20.000000000000000          20.000000000000000
           7   0.0000000000000000          40.000000000000000
           8   10.000000000000000          40.000000000000000
           9   20.000000000000000          40.000000000000000
          10   0.0000000000000000          60.000000000000000
          11   10.000000000000000          60.000000000000000
          12   20.000000000000000          60.000000000000000
           1           1           4           5           2
           2           2           5           6           3
           3           4           7           8           5
           4           5           8           9           6
           5           7          10          11           8
           6           8          11          12           9
           1           0           0
           2           1           0
           3           1           0
           4           0           1
           5           1           1
           6           1           1
           7           0           1
           8           1           1
           9           1           1
          10           0           1
          11           1           1
          12           1           1
           3
          10
          11
          12
```

Figure 6-26 Result file of pre-processing transfer

Figure 6-27, Figure 6-28, Figure 6-29, Figure 6-30 and Figure 6-31 all show the result of post-processing transfer. The contour of elastic stress, elastic strain, creep strain and creep damage look uneven because the actual change of value (result) occurs after the fourth effective digital. Detailed values could be found in appendix 10.4.7.

In order to understand this problem[14], Figure 6-28 is used as the example to show the details. From 10.4.7, the elastic stress in y-direction could be summarized as:

Table 6-13 Elastic stress in y-direction

Element 1	Integration point 1	39.999982450985883
	Integration point 2	39.999982450985883
	Integration point 3	39.999982450985883
Element 2	Integration point 1	39.999994616455837
	Integration point 2	39.999994616455837
	Integration point 3	39.999994616455837
Element 3	Integration point 1	40.000052893418300
	Integration point 2	40.000052893418300
	Integration point 3	40.000052893418300
Element 4	Integration point 1	39.999965270768499
	Integration point 2	39.999965270768499
	Integration point 3	39.999965270768499
Element 5	Integration point 1	40.000434237458435
	Integration point 2	40.000434237458435
	Integration point 3	40.000434237458435
Element 6	Integration point 1	40.000298000417629
	Integration point 2	40.000298000417629
	Integration point 3	40.000298000417629
Element 7	Integration point 1	40.000358204542437
	Integration point 2	40.000358204542437
	Integration point 3	40.000358204542437
Element 8	Integration point 1	40.000316568443083
	Integration point 2	40.000316568443083
	Integration point 3	40.000316568443083

From the scales of Figure 6-28, the stress represented by the dark blue is less than the stress represented by light blue; moreover, from Figure 6-27, the elements on the above of square is element 1 to 4 from left side to right side. Compared with Table 6-13, the observation satisfies the actual values. Use the same method, Figure 6-29, Figure 6-30 and Figure 6-31 can be proved as well.

[14] Figure 6-28 to 6-31 are a same problem about meaningless scales. The scales of FEMGV normally reserves three effective digitals, but the actual change occurs after the fourth effective digital.

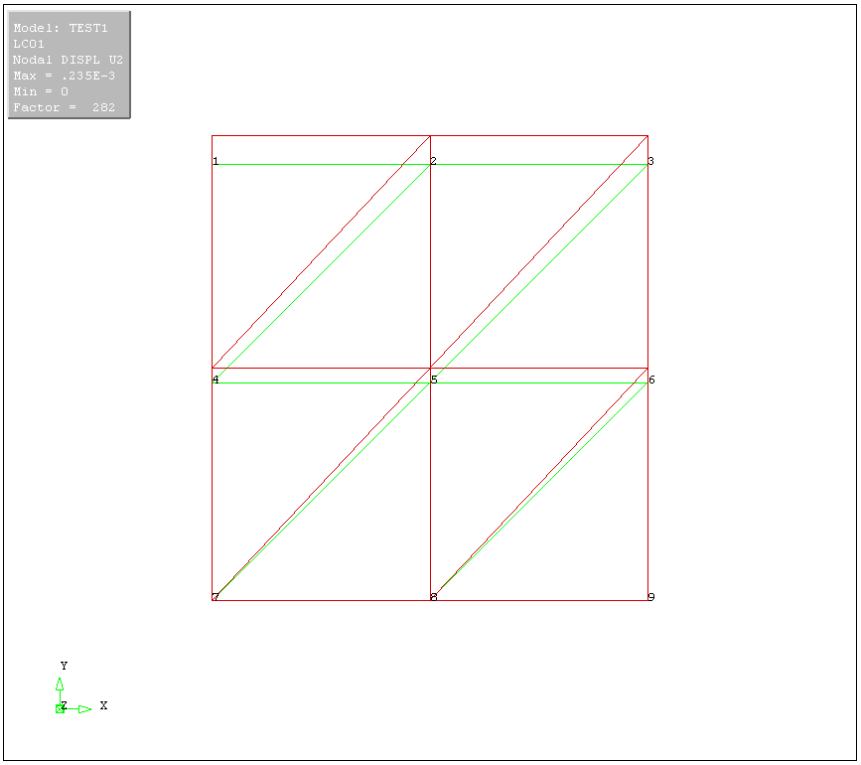

Figure 6-27 Displacement shape in y-direction

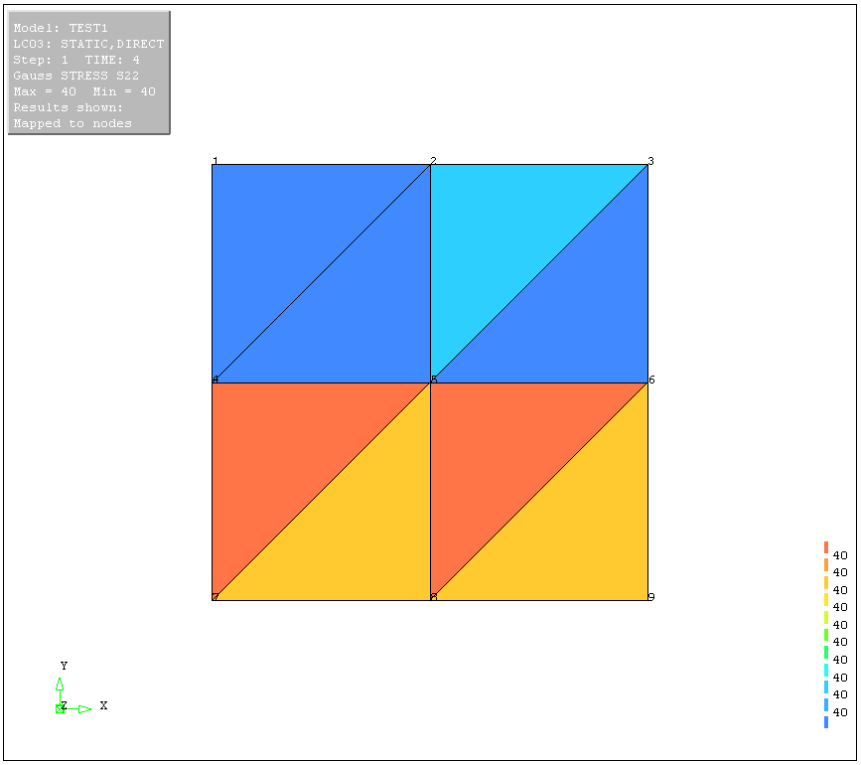

Figure 6-28 Contour of elastic stress y-direction

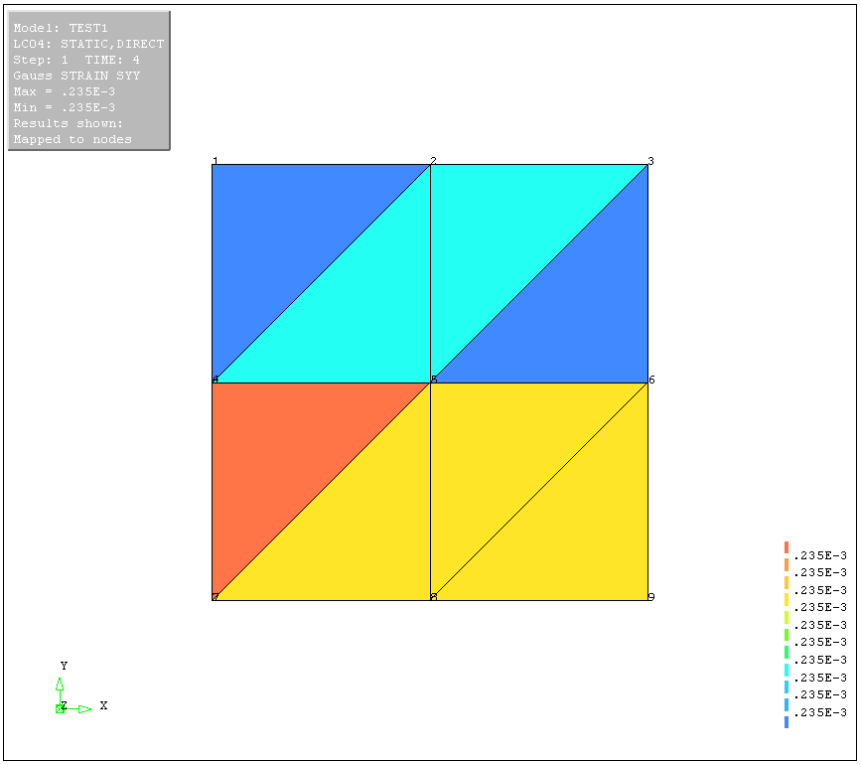

Figure 6-29 Contour of elastic strain y-direction

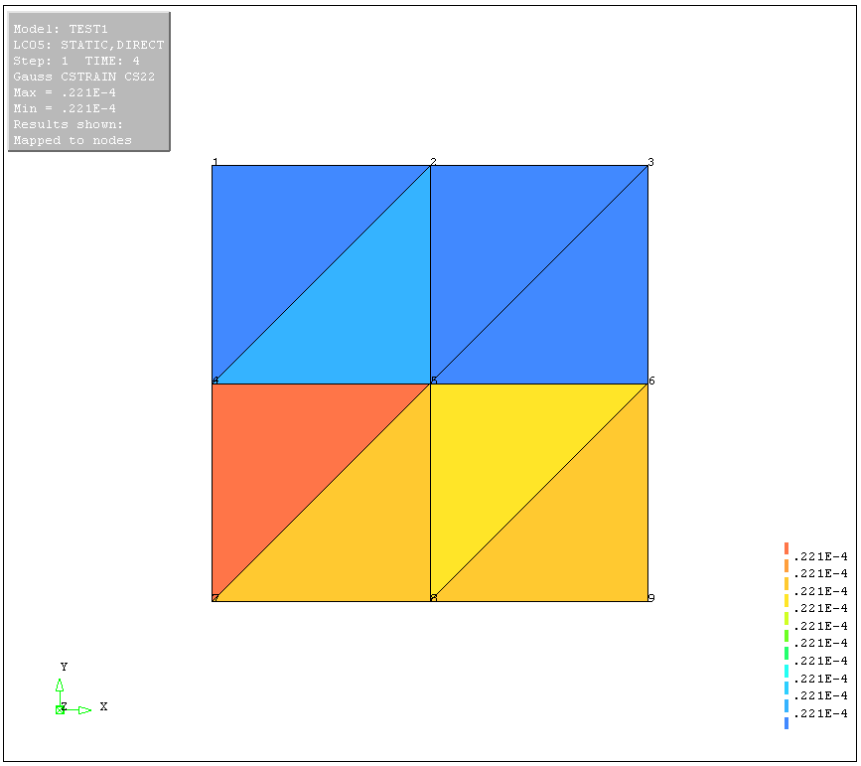

Figure 6-30 Contour of creep strain y-direction

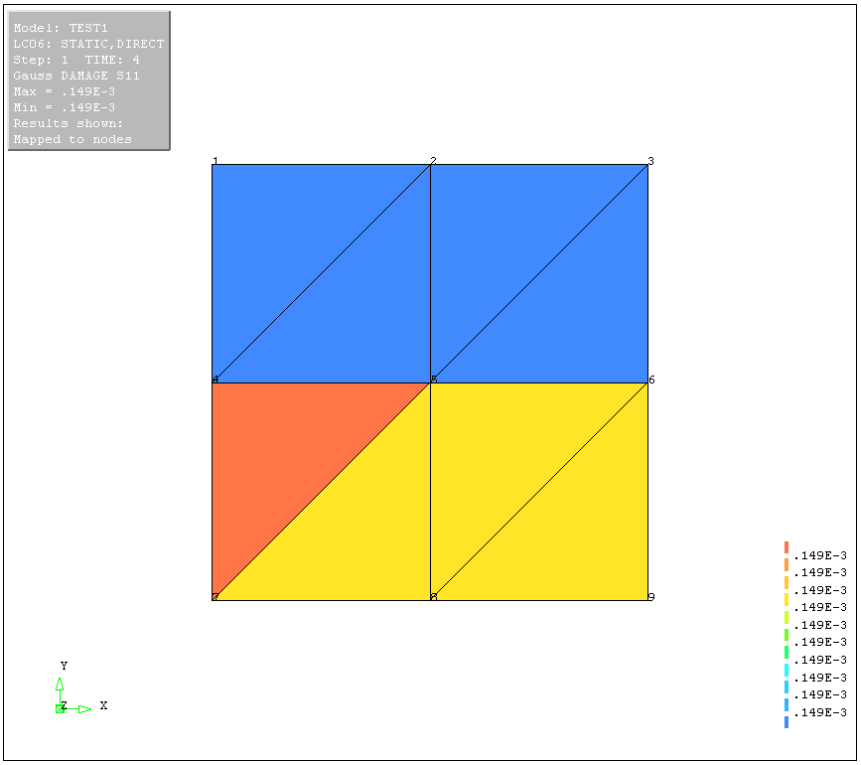

Figure 6-31 Contour of creep damage

6.8 Summary

The tests of subroutines and programmes were designed based on the major concerned problems. The flowcharts of each test programme were presented to show the detailed structure clearly. A number of test cases, which include the input and output, were introduced specifically in order to validate the developed subroutines and programmes in this research. Most results were summarised into tables and curves for the convenience of the reader. The order of testing of subroutine and programme was arranged according to the independence of each subroutine. The core part is creep constitutive equations; therefore, the first stage of research was launched around creep constitutive equations. In total, 10 subroutines and 2 programmes were developed, and these subroutines and programmes could be divided into 7 categories.

Table 6-14 indicates the validation purpose and method of all subroutines and programmes which developed in this research. They are specific to that 1) constitutive equations subroutines include **KR, PH** and **QX**; 2) stress transformation subroutine **TRS**; 3) numerical method subroutines include **EULER, RK4, RKM** and **RKF**; 4) Time-step control subroutine **TSC**; 5) normalization subroutine includes **NOR_KR**; 6) Nodal loads calculator is a programme **NLC**; and 7) Data transfer interface is a programme **DTI**.

Table 6-14 Summary of each tests

Name	Description
TRS	Two set of supposed stress tensor each have four components and six components were given. Hand calculation was reported to compare with the results obtained from TRS.
EULER	A supposed function was coded based on CESL for this testing. Its exact solution was given to compare with the result obtained from each subroutine. It can not only prove the correctness of each subroutine, but also understand the accuracy of each numerical method.
RK4	
RKM	
RKF	
KR	Three set of material properties were given for each constitutive equation. A single stress state was implemented firstly to prove the correctness of the uni-axial form of constitutive equations.
PH	The plane stress state, which includes uni-axial tensile, bia-axial, and pure shear, was implemented secondly to prove the correctness of each subroutine within 2D environment.
QX	The 3D stress state, which contains the uni-axial tensile of each normal stress direction, was implemented to prove the correctness of each subroutine within 3D environment.
TSC	A supposed time-step control value was given to test the time-step will be reduced when required. Additionally, TSC was implemented with PH and RKM to present the time-step control procedure.
NOR_KR	The plane stress state, which includes uni-axial tensile, bia-axial, and pure shear, was implemented secondly to prove the correctness of each subroutine within 2D environment. The 3D stress state, which contains the uni-axial tensile of each normal stress direction, was implemented to prove the correctness of each subroutine within 3D environment.
NLC	A simple case was supposed, and its loaded node number, inner radius and outer radius of each node were given to NLC. Hand calculation was reported to prove the correctness of NLC.
DTI	A simple square case was simulated; the pre- and post-processing transfer was proved through the solver and FEMGV.

7 EXPLORATION OF THE PERFORMANCE OF NUMERICAL METHODS, TIME-STEP CONTROL PROCEDURE AND NORMALIZATION SCHEME

This chapter explores a series of performance problems of the subroutines developed by this research. This discussion includes the performance exploration of numerical method subroutines, time-step control subroutine and normalization subroutine. Numerical method subroutines, currently includes **EULER** subroutine, **RK4** subroutine, **RKM** subroutine and **RKF** subroutine; however, the quantitative analysis of those numerical methods is deficient. The sensitivity of self-adaptive method of RKF method is an issue; it was observed that the self-adaptive method of RKF method is invalid in the testing of time-step control subroutine. Thus, it will be covered in this chapter. Normalization approach is suggested to enhance the accuracy and power of computation. It expands (or shrinks) proportionately the independent variables and dependent variables through mathematical theory. The quantitative analysis was conducted to show the effects of the normalization approach through a set of experiments.

7.1 Performance of Numerical Integration Methods

The author has addressed application of RKF method for creep damage analysis; however, its practical performance is uncertain. In order to have an intuitive understanding, some general approaches were applied to conduct a set of comparative tests. Traditionally, RKM method is normally suggested in creep damage analysis due to its superiority of accuracy (Hayhurst et al., 1984, Ling et al., 2000); moreover, Euler's method is also used by some researchers based on a critical time-step size (Wang and Wang, 1996). The RK4 method was also discussed. In here, only the performance of numerical method subroutines was focused on; thus, the mathematical superiorities of the numerical method itself will not be addressed.

7.1.1 Experiments design

Euler's method was used to find the closest exact solution since its accuracy could be improved by constantly reducing the time-step size; it aims to set a basis for comparison. RK4 method, RKM method and RKF method will be implemented separately, which are based on the time-step of 0.1 hour to ensure the consistency of the experiments. Due to the performance of integration accuracy being the only issue concerned in this experiment, a unified termination time-line was set which is 980 hours.

The uni-axial form of Perrin's constitutive equations was used to implement these experiments. The same material and loaded stress will be adopted in order to ensure the consistency of this experiment. 70MPa uniform stress was applied, and the material properties are presented in Table 7-1. Only the error of creep strain will be compared to give a quantitative result of the performance of each numerical method subroutine. A programme called **NMSexp** was developed to conduct these experiments and subroutine **EULER, RK4, RKM** and **RKF** were used; its detailed source code was attached in appendix 10.5.1. For Euler's method only, five time-steps, which are 1 hour, 0.1 hour, 0.01 hour, 0.001 hour and 0.0001 hour, were used to conduct the continuous reduction of time-step size.

Table 7-1 Material properties of Perrin-Hayhurst constitutive equations

A	B	h	K_c	H*	C	ν
6.216×10^{-8}	0.15	1.0×10^{4}	4.998×10^{-4}	0.35	2.0	1.32

7.1.2 Result and discussion

According to the experimental setting, the first discussion was given to the finding of the closest exact solution. Table 7-2 shows the five sets of running time and final rupture strain based on Euler's method. When the time-step reduced until 0.001 hour, the computing speed of Euler's method was rapidly decreased; the improvement of rupture strain almost could be ignored in practical applications once the time-step is less than 0.01 hour. However, because this is an exploratory experiment, the results obtained based on 0.0001 hour were used to be the comparative basis.

Table 7-2 Running time and rupture strain based on Euler's method using time-step size of 1 hour, 0.1 hour, 0.01 hour, 0.001 hour and 0.0001 hour

Time-step/hours	Running	Increase	Rupture strain	Improvement
1	3.12×10^{-2}	base	0.160324134	base
0.1	4.68×10^{-2}	50%	0.182306477	13.71%
0.01	9.36×10^{-2}	200%	0.184732961	15.22%
0.001	5.93×10^{-1}	1800.64%	0.184987248	15.38%
0.0001	5.46	17400%	0.185012809	15.40%

Moreover, Figure 7-1 and Figure 7-2 shows an intuitive understanding of the changes based on the reduction of time-step. The creep strains based on the time-step of 1 hour were deducted from the creep strains based on the time-step of 0.1 hour, 0.01 hour, 0.001 hour and 0.0001 hour separately according to the corresponding time point; hence, four relative error curves of creep strain could be plotted. Figure 7-1 shows the relative error curves of Euler's method using time-steps of 0.1 hour, 0.01 hour, 0.001 hour and 0.0001 hour, which only displayed the data of 0 hour to 800 hour. It is obvious to see that, the relative error of time-step 1 hour vs 0.1 hour has an obvious difference with the other three curves. However, the differences between the relative error of time-step 1 hour vs 0.01 hour, the relative error of time-step 1 hour vs 0.001 hour and the relative error of time-step 1 hour vs 0.0001 hour are not large.

In order to find the minor difference, those curves were zoomed in the time range of five hours, and the new curves were display on Figure 7-2. It could be seen that, there is almost no difference when the time-step is reduced to less than 0.001 hour; however, due to the critical requirement of accuracy, the creep strain curve of time-step 0.0001 hours was adopted to be similar to the exact solution.

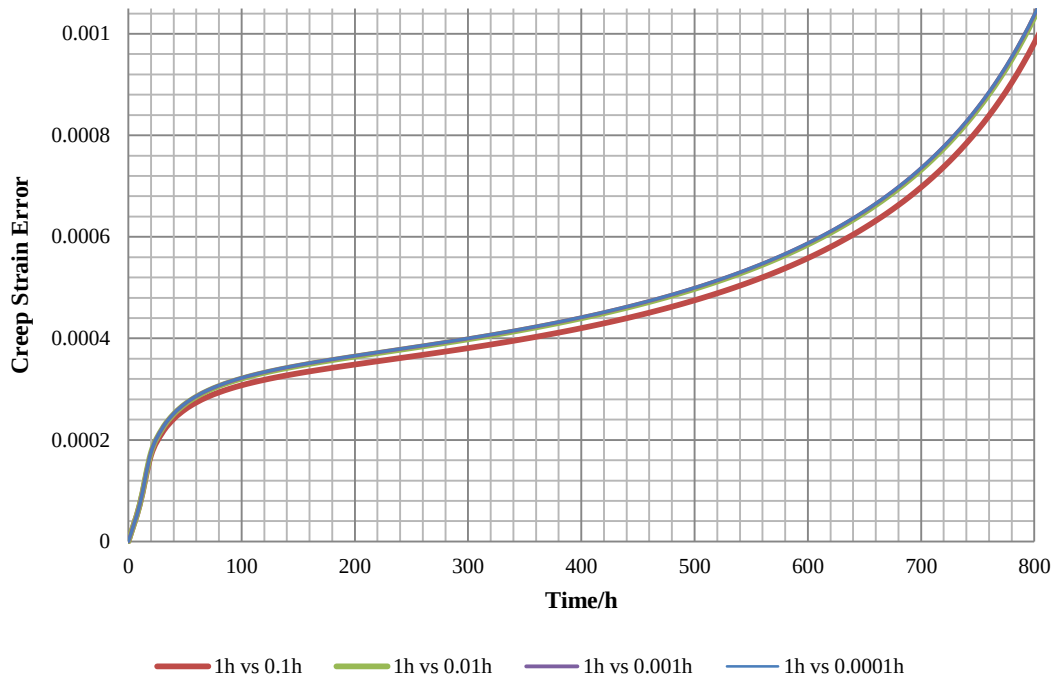

Figure 7-1 Relative error curves of Euler's method using time-step of 0.1 hour, 0.01 hour, 0.001 hour and 0.0001 hour, only displayed from 0 hour to 800 hours

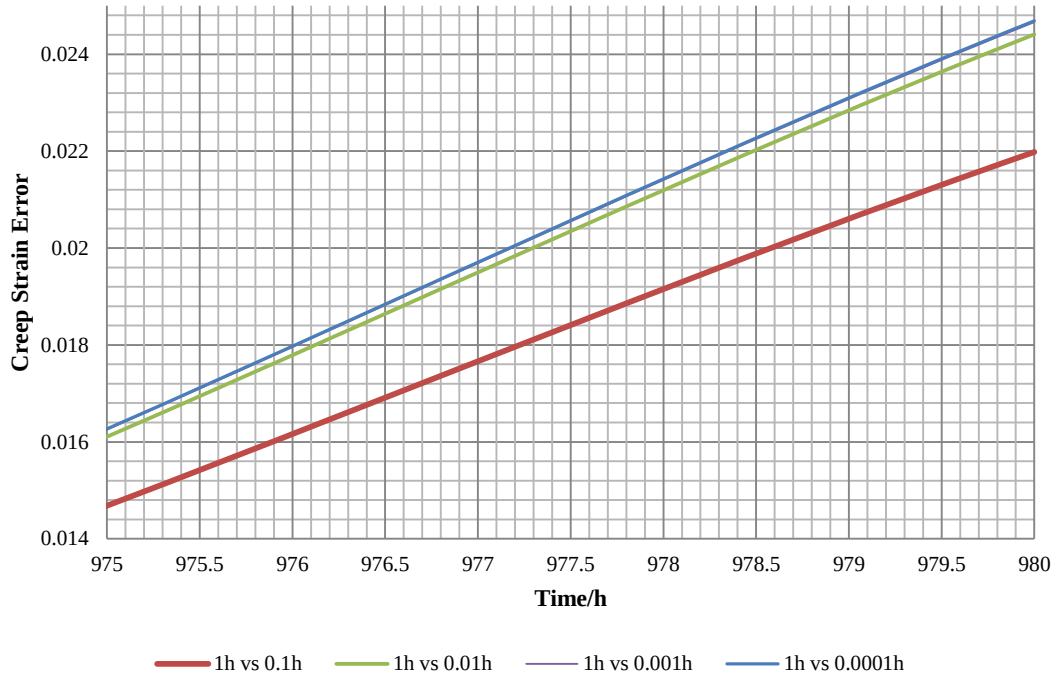

Figure 7-2 Relative error curve of Euler's method using time-step of 0.1 hour, 0.01 hour, 0.001 hour and 0.0001 hour, only displayed from 975 hours to 980 hours

The second discussion will be given to the accuracy and efficiency comparison of subroutine **RK4, RKM** and **RKF**. Figure 7-3 presents the relative error curves based on RK4 method, RKM method and RKF method respectively from 0 to 800 hours. The curve

of RK4 increased across the curve of RKF; its increased tendency has a change at 400 hours because the default value of time-step is unsuitable from this time moment. The curves based on RKM and RKF have the same performance in the initial 300 hours; however, a significant change occurs at this time moment, where the relative error of RKM increased faster than the value of RKF.

Moreover, the subsequent changes can be observed from Figure 7-4 which presents the relative error curves based on RK4 method, RKM method and RKF method respectively from 960 to 980 hours. In summary, subroutine **RKF** has a poor start but its accuracy could be guaranteed during whole integration process; subroutine **RKM** needs the reduction of time-step size to keep the same accuracy of RKM. Based on the almost same running time of subroutine **RK4, RKM** and **RKF**, it could be identified that subroutine **RKF** has the best performance based on constant time-step size in this exploration.

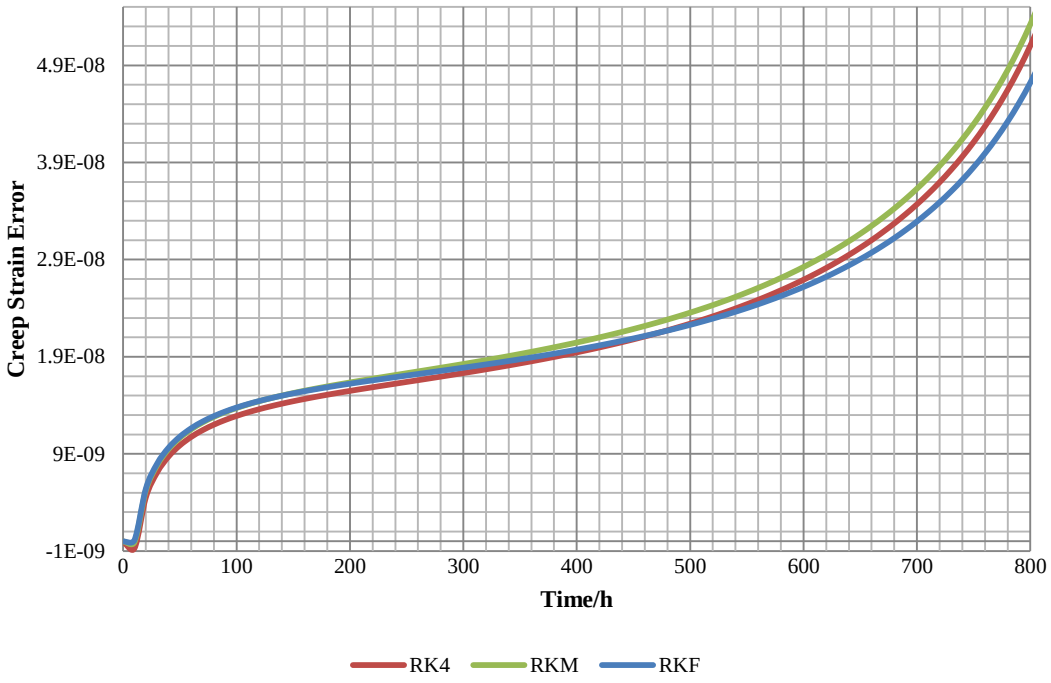

Figure 7-3 Relative error curves of RK4 method, RKM method and RKF method, which based on the Euler's method using time-step of 0.0001 hour, only displayed from 0 hours to 800 hours

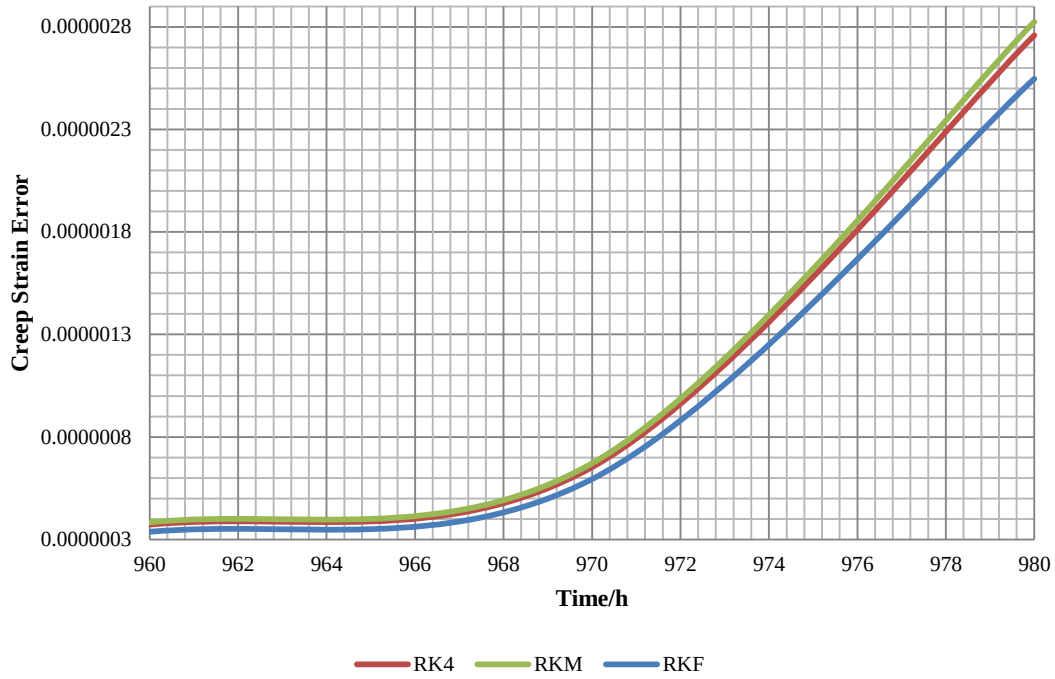

Figure 7-4 Relative error curves of RK4 method, RKM method and RKF method, which based on the Euler's method using time-step of 0.0001 hour, only displayed from 960 hours to 980 hours

7.2 Self-adaptive approach of Runge-Kutta-Fehlberg method

The unexpected performance of self-adaptive method of RKF method was observed in the testing of time-step control subroutine. The accuracy of RKF method and the acceptance criteria self-adaptive method of RKF method were questioned. The accuracy of RKF method has been proved through further discussion in 6.4.3.2 and 7.1.2; however, the acceptance criteria of RKF method were still suspected. The self-adaptive method of RKF method will be discussed in order to clear this uncertain factor.

7.2.1 Experiments design

The acceptance of time-step may not be critical enough in creep damage analysis environment; hence, a set of comparative experiments were conducted to find the answer. The function $y' = \mathrm{y} - t^2 + 1$ was adopted to prove that the time-step acceptance of RKF method is available. Such function was coded as a subroutine called **NTEST** according to the structure of constitutive equations subroutine. Two tests were launched, where the given initial value is $\mathrm{Y}_0 = 0.5$, and time interval are dt = 0.2 and dt = 0.4 respectively.

The uni-axial form of Perrin's constitutive equations was selected to implement the comparative experiments. 70MPa uniform stress was applied, and the material properties are presented in Table 7-3 and the time-steps are 1 hour and 2 hour respectively.

A programme called **TSCexp** was developed to conduct these experiments; its detailed source code is attached in appendix 10.5.2.

Table 7-3 Material properties of Perrin-Hayhurst constitutive equations

A	B	h	K_c	H*	C	ν
6.216×10^{-8}	0.15	1.0×10^{4}	4.998×10^{-4}	0.35	2.0	1.32

7.2.2 Result and discussion

The results produced by subroutine **NTEST** could identify the working status of the self-adaptive approach of the RKF method. Figure 7-5 presented the results based on subroutine **NTEST** using initial time-step of 0.2 and 0.4 respectively. When the initial time-step of 0.2 was employed, the subsequent time-step size does not change. It could be said that the time-step of 0.2 satisfies the predict accuracy requirements. When the initial time-step of 0.4 was employed, the subsequent time-step size was reduced to 0.2, and it will not change any more until to the end. Obviously, the self-adaptive technique of RKF method is working.

```
initial time-step: 0.2                          initial time-step: 0.4
time   solution    cVal  time-step       time   solution    cVal  time-step
0.20 8.2929854E-01    0 2.0000000E-01    0.00 5.0000000E-01    1 2.0000000E-01
0.40 1.2140875E+00    0 2.0000000E-01    0.20 8.2929854E-01    0 2.0000000E-01
0.60 1.6489403E+00    0 2.0000000E-01    0.40 1.2140875E+00    0 2.0000000E-01
0.80 2.1272291E+00    0 2.0000000E-01    0.60 1.6489403E+00    0 2.0000000E-01
1.00 2.6408586E+00    0 2.0000000E-01    0.80 2.1272291E+00    0 2.0000000E-01
1.20 3.1799409E+00    0 2.0000000E-01    1.00 2.6408586E+00    0 2.0000000E-01
1.40 3.7323992E+00    0 2.0000000E-01    1.20 3.1799409E+00    0 2.0000000E-01
1.60 4.2834828E+00    0 2.0000000E-01    1.40 3.7323992E+00    0 2.0000000E-01
1.80 4.8151751E+00    0 2.0000000E-01    1.60 4.2834828E+00    0 2.0000000E-01
2.00 5.3054705E+00    0 2.0000000E-01    1.80 4.8151751E+00    0 2.0000000E-01
```

Figure 7-5 Results of subroutine NTEST based on the initial time-step of 0.2 and 0.4

Further experiments have been conducted. Figure 7-6 presented the results of subroutine **PH** based on the initial time-step of 1 hour and 2 hour. The first significant observation is that the time-step size never changed during the whole integration process whether based on the initial time-step of 1 hour or 2 hour. This shows that the acceptance criteria should be identified in order to satisfy the requirements of creep damage analysis. The second significant observation is that the accuracy has been decreased when the time-step size

increased. This shows that the acceptance criteria should be identified in order to satisfy the requirements of creep damage analysis again.

initial time-step: 1				initial time-step: 2			
time	strain	cVal	time-step	time	strain	cVal	time-step
10.00	2.4653859E-03	0	1.0000000E+00	10.00	2.3493084E-03	0	2.0000000E+00
20.00	3.3974839E-03	0	1.0000000E+00	20.00	3.2466488E-03	0	2.0000000E+00
.....				.....			
510.00	2.9558457E-02	0	1.0000000E+00	510.00	2.9284559E-02	0	2.0000000E+00
520.00	3.0344719E-02	0	1.0000000E+00	520.00	3.0067218E-02	0	2.0000000E+00
.....				.....			
890.00	8.9121623E-02	0	1.0000000E+00	890.00	8.8245869E-02	0	2.0000000E+00
900.00	9.3029249E-02	0	1.0000000E+00	900.00	9.2071921E-02	0	2.0000000E+00
910.00	9.7378887E-02	0	1.0000000E+00	910.00	9.6319680E-02	0	2.0000000E+00
.....				.....			
960.00	1.3354620E-01	0	1.0000000E+00	960.00	1.3087474E-01	0	2.0000000E+00
970.00	1.4950233E-01	0	1.0000000E+00	970.00	1.4529057E-01	0	2.0000000E+00
980.00	1.8558258E-01	0	1.0000000E+00	980.00	1.7302943E-01	0	2.0000000E+00

Figure 7-6 Results of subroutine PH based on the initial time-step of 1 hour and 2 hour

7.3 Normalization Technique

A mathematical approach called normalization was applied in this research in order to enhance the accuracy and efficiency of integration of constitutive equations; however, the practical performance of this subroutine should be quantitatively analysed before being recommended to users. In here, only the performance of normalization subroutine was focused on; thus, the mathematical superiority of normalization itself will not be discussed.

7.3.1 Experiments design

In order to test the performance of normalization subroutine, a set of comparative tests were proposed. The same material and loaded stress will be adopted in order to ensure the consistency of this experiment. Table 7-4 shows the detailed parameters of material property; moreover, it could be seen that the argument of B is different between non-normalized material properties and normalized material properties. This arguments should be normalized according to Eq. (4-57) before being used. 70MPa was adopted as the loaded stress and the time-step size is 1 hour. Since the results produced by normalization subroutine are normalized, they should be converted to the non-normalized value to perform the final comparison. The running time, creep strain and damage will be compared to give a quantitative result of the performance of normalization subroutine.

Programme **CEScheck** and **NORcheck** were used to conduct this experiment, and Microsoft Excel was use to convert those normalized results and to produce visualized results.

Table 7-4 Material properties used for the performance investigation of normalized subroutine

Non-normalized material properties						
A	n	m	B	φ	χ	α
1.092×10^{-20}	8.462	$-4.754\times10^{-}$	3.537×10^{-17}	7.346	6.789	0.215
Normalized material properties						
A	n	m	B'	φ	χ	α
1.092×10^{-20}	8.462	$-4.754\times10^{-}$	4.238×10^{-18}	7.346	6.789	0.215

7.3.2 Results and discussion

The normalization technique enhances the integration of the creep constitutive equations through redefinition of time, stress and strain[15]; hence, this first discussion was given to the changing scope of time-step size. According to the time scale conversion formula (7-1), the real time could be obtained.

$$t = \left[\frac{\tau(m+1)}{AE{\sigma_0}^{n-1}}\right]^{\left(\frac{1}{m+1}\right)} \qquad (7\text{-}1)$$

where, t is real time; τ is normalized time; σ_0 is the stress used for normalization; and, E is the Young's modulus;

Figure 7-7 presents converted results of normalized time of the whole integration process. The phenomenon was observed, where the equivalent real time rapidly changed during the first 200 hours then going into the steady state in the rest. It means the critical requirement of time-steps of creep tertiary stage based on non-normalized method was ameliorated. The changing scope of equivalent real time is between 0.9183 hour and 0.9219 hour; only 0.36% change occurred during almost 1200 hours. Compared with time-step used for non-normalized method, the mean error will not exceed 8%; therefore,

[15] The redefined time, stress and strain are called normalized time, stress and strain; hence, the time-step used for integrating normalized constitutive equations is different with the real time-step. The normalized time-step does not use the unit such as seconds, minute or hour, and each normalized time-step does not represent a fixed real time-step. Generally, a normalized time-step can be seen as a solution of an ungiven function of real time-step.

this exploration could continue since the consistency of experiment was essentially ensured.

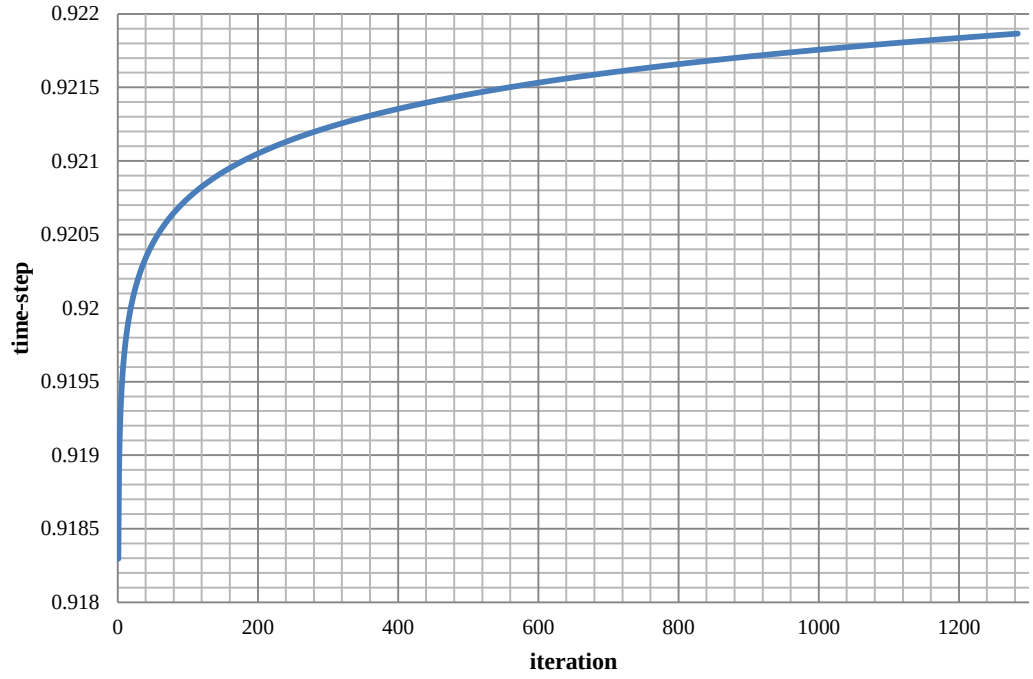

Figure 7-7 Equivalent real times of normalized method

The normal creep strain and lifetime based on both the normalization method and non-normalization method are shown on Figure 7-8. The enhancements of the normalization technique could be summarised according to lifetime, creep strain and running time. The lifetime obtained by the normalized technique is almost 1200 hours, but the result from the non-normalized method is just over 1000 hours; thus, the lifetime has 17.14% improvement through the normalized approach. The creep strain obtained by the normalized technique is over 0.37, but the result from the non-normalized method is 0.33; hence, the creep rupture strain has 14.24% improvement through the normalized approach. The normalized time-step was set as 0.1 and the non-normalized time-step was set as 1 hour; however, the running time based on the normalization subroutine is only 27.13% more than the normal constitutive equations subroutine. Therefore, the efficiency of subroutine has been greatly improved.

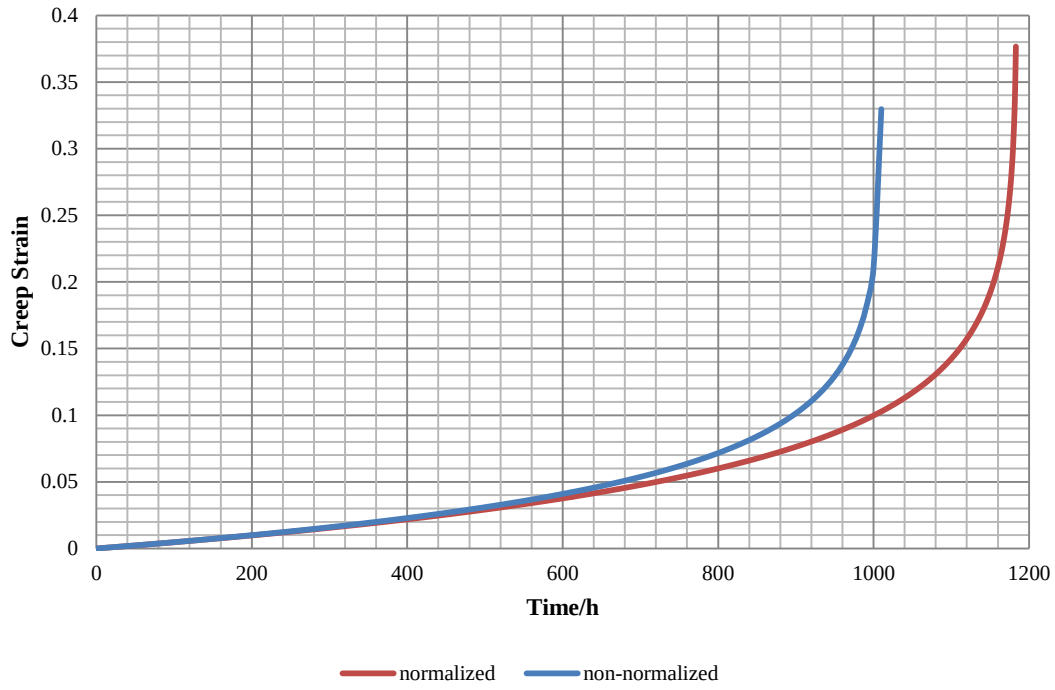

Figure 7-8 Creep strain curve using normalized approach and non-normalized approach

7.4 Summary

The performance exploration of numerical method subroutines, time-step control subroutine and normalization subroutine has been conducted. Based on the constant time-step size, subroutine RKF has the best performance of accuracy and efficiency; however, there is an issue with its self-adaptive technique. The existing acceptance criteria of time-step size do not satisfy the accuracy requirements of creep damage analysis. The normalization technique could enhance the accuracy and efficiency of numerical integration of creep constitutive equations a lot; however, it should be studied first since it involves the modification of the original constitutive equations.

8 CONCLUSION AND FURTHER WORK

This chapter summarizes the major contributions of this research. Some novel methods and techniques have been used and explored. A number of theoretical analysis process, subroutine/programme development process and subroutine/programme validation process are presented. The major outcomes of this research are summarized. In the end, further development of HITSI is also discussed.

8.1 Contributions

This research contributed to develop a novel in-house finite element solver for the research of creep damage analysis. A unified finite element software does not exist in present creep research domain due to the lack of unified constitutive model that could be applied in all materials. In view of this situation, this in-house code was developed based on these superiorities listed below.

- General models, whether plane stress, plane strain, axisymmetric or three-dimension are considered;
- Its scalability is guaranteed due to the standardized programming applied. It enables the potential user to apply their own constitutive equations directly without any understanding of programming of finite element method;
- Diversified numerical methods are available, enables more advanced and accurate integration methods to be introduced into creep damage analysis.

The second contribution of this research is the development of prototype of the data transfer interface that called **DTI**. All of the traditional commercial finite element software such as ABAQUS, ANSYS and FEMGV have their own data transfer interface to keep their compatibility and versatility. **DTI** not only fills the gap between the solver and FEMGV, but also enables the solver to be commercialized based on the platform of FEMGV in the future. Moreover, the **DTI** will not restricted to FEMGV only, it can be interfaced (through a formatted neutral text file) with any commercial FE package.

8.2 Conclusion

In order to fulfil the aim and objectives of this research, a number of issues have been studied and investigated, which include modular conversion of applied stress, standardized programming of constitutive equations and numerical methods, accuracy enhancement through self-adaptive method and normalization method. Moreover, formatted input and output has also been studied in order to build a useful logical control data transfer interface. General software development processes including requirement analysis, design and testing have been performed. The major outcomes of this research are summarised below.

A unified programming standard of the subroutine of constitutive equations has been identified. The key parts of the constitutive equations such as stress functions, time functions and temperature functions have been classified. Three explicit-shape real arrays were used to hold the arguments of creep strain and damage, stress and material properties. Due to the argument of time is a dynamic variable, an independent real variable was arranged to it. Normally, the temperature is a constant, so it was included as a function in the material properties.

The matched numerical method subroutines have been developed. Based on the present structure of these numerical method subroutines, the arguments of creep strain rate and creep damage rate will never appear on the main programme of the solver; hence, the memory of those arguments will be released when the present integration process is complete.

A modular conversion subroutine of stress has been developed. Compared with using the existing subroutines, its advantages are obvious; for example, the memory requirement of the main programme of the solver was reduced. Due to the embedded design, the system

of two-dimensions and three-dimensions do not need to be distinguished. The dimensions could be changed automatically.

The research of accuracy and efficiency based on numerical methods has been done. The instability during serving the constitutive equations of RK4 method was observed; moreover, the sensitivity of the self-adaptive technique of RKF method has been discussed. In summary, the attempt of introducing new numerical methods does not achieve the expectation.

The improvement of efficiency and accuracy has been tried from a mathematical aspect. A significant result was obtained, and could be summarized as:

- No more than 8% reduction of time-step leads to an increase of 27.13% in computational speed;
- The lifetime and rupture strain have the increases of 17.4% and 14.24% based on this method.

The boundary conditions of nodal loads and element definition have been solved. The general algorithm used to allocate a concentrated force to loaded nodes has been studied; however, the method of allocation itself is a challenge that leads to the pre-processing transfer being not perfect. A temporary solution, called nodal loads calculator has been developed in view of this situation.

8.3 Further Work

The research work presented in this thesis is a primary stage of the development of HITSI since this system has been developed three years ago from nothing. Further research could be conducted based on HITSI to enhance its capability.

The further works related to this research are listed below:

1. The capability of the data transfer interface should be enhanced. Especially, the conversion of nodal loads should be integrated into the data transfer interface to enable the pre-processing transfer to be accomplished automatically. Furthermore, based on the present algorithm developed by the author, a number of data types could be included such as more element types, more constraint types and more

load types. To enable the data transfer interface to be visualized is another research direction.

2. More creep constitutive equations and numerical methods could be collected and coded based on the standardized subroutine template developed by the author. This enhancement of constitutive equations subroutine library and numerical method subroutine library enables HITSI to be commercialized.

9 REFERENCE

ANDRADE, E. D. C. 1910. On the viscous flow in metals, and allied phenomena. *Proceedings of the Royal Society of London. Series A, Containing Papers of a Mathematical and Physical Character*, 1-12.

ASHBY, M. F., EDWARD, G. H., DAVENPORT, J. & VERRALL, R. A. 1978. Application of bound theorems for creeping solids and their application to large strain diffusional flow. *Acta Metallurgica,* 26**,** 1379-1388.

BECKER, A. A., HYDE, T. H., SUN, W. & ANDERSSON, P. 2002. Benchmarks for finite element analysis of creep continuum damage mechanics. *Computational Materials Science,* 25**,** 34-41.

BELL, D. 2000. *Software engineering: a programming approach,* New York; Harlow, England, Addison Wesley.

BERTO, F., GALLO, P. & LAZZARIN, P. 2014. High temperature fatigue tests of un-notched and notched specimens made of 40CrMoV13. 9 steel. *Materials & Design,* 63**,** 609-619.

BOYER, R. R. 1996. An overview on the use of titanium in the aerospace industry. *Materials Science and Engineering: A,* 213**,** 103-114.

CANE, B. J. 1981. Creep fracture of dispersion strengthened low alloy ferritic steels. *Acta Metallurgica,* 29**,** 1581-1591.

CANE, B. J. Creep damage accumulation and failure criteria increep brittle ferritic weldment structures. Proceedings of International Conference on Welding Technology for Energy Applications. American Welding Society, Gatlingburg Tennessee, 1982. 623.

CANE, B. J. Creep damage accumulation and fracture under multiaxial stresses. ICF5, Cannes (France) 1981, 2013.

CHAPRA, S. C. & CANALE, R. P. 1998. *Numerical methods for engineers: with programming and software applications,* Boston, WCB/McGraw-Hill.

CHEN, M. 2007. *Elasticity and Plasticity,* Beijing, Science Press

COCKS, A. C. F. & ASHBY, M. F. 1980. Intergranular fracture during power-law creep under multiaxial stresses. *Metal science,* 14**,** 395-402.

COCKS, A. C. F. & ASHBY, M. F. 1981. *Creep fracture by void growth,* Springer.

COCKS, A. C. F. & ASHBY, M. F. 1982. Creep fracture by coupled power-law creep and diffusion under multiaxial stress. *Metal Science,* 16**,** 465-474.

COLOMBO, P. P., GARZILLO, A., MERIGGI, M., PONZONI, C. & SAMPIETRI, C. 1996. Creep and damage analysis of a serviced tee intersection in a boiler header: comparison between numerical and experimental results. *International journal of pressure vessels and piping,* 66**,** 243-251.

EASON, T. J., BOND, L. J. & LOZEV, M. G. Structural health monitoring of localized internal corrosion in high temperature piping for oil industry. Volume 34, 2015. AIP Publishing, 863-873.

EDWARD, G. H. & ASHBY, M. F. 1979. Intergranular fracture during power-law creep. *Acta Metallurgica,* 27**,** 1505-1518.

FAIRES, J. D. & BURDEN, R. L. 2013. *Numerical methods,* Pacific Grove, Calif.?, Brooks/Cole.

FEHLBERG, E. 1968. Classical fifth-, sixth-, seventh-, and eighth-order Runge-Kutta formulas with stepsize control. NASA TR R-287.

FEHLBERG, E. 1969. Low-order classical Runge-Kutta formulas with stepsize control and their application to some heat transfer problems. NASA TR R-315.

GENG, L. Y., GONG, J. M., LIU, D. & JIANG, Y. 2009. Damage analysis and life prediction of a main steam pipeline at elevated temperature based on creep damage mechanics. *Sustainable Power Generation and Supply.* IEEE.

GERE, J. M. & GOODNO, B. J. 2009. *Mechanics of materials,* Stamford, Conn, Cengage Learning.

GITTUS, J. 1975. *Creep, viscoelasticity and creep fracture in solids,* London, Applied Science Publishers.

GOODALL, I. W. & SKELTON, R. P. 2004. The importance of multiaxial stress in creep deformation and rupture. *Fatigue & Fracture of Engineering Materials & Structures,* 27**,** 267-272.

GURSON, A. L. 1977. Continuum theory of ductile rupture by void nucleation and growth: Part I—Yield criteria and flow rules for porous ductile media. *Journal of engineering materials and technology,* 99**,** 2-15.

HAGIHARA, S. & MIYAZAKI, N. 2008. Finite element analysis for creep failure of coolant pipe in light water reactor due to local heating under severe accident condition. *Nuclear Engineering and Design,* 238**,** 33-40.

HALES, R. 1994. The role of cavity growth mechanisms in determining creep rupture under multiaxial stresses. *Fatigue & Fracture of Engineering Materials & Structures,* 17**,** 579-591.

HALL, F. R. 1990. *Development of continuum damage mechanics models to predict the creep deformation and failure of high temperature structures.* Doctoral Thesis (Ph.D.) University of Sheffield.

HAYHURST, D. R. 1972. Creep rupture under multi-axial states of stress. *Journal of the Mechanics and Physics of Solids,* 20**,** 381-382.

HAYHURST, D. R., DIMMER, P. R. & MORRISON, C. J. 1984. Development of Continuum Damage in the Creep Rupture of Notched Bars. *Philosophical Transactions of the Royal Society of London A: Mathematical, Physical and Engineering Sciences,* 311**,** 103-129.

HAYHURST, D. R., DYSON, B. F. & LIN, J. 1994. Breakdown of the skeletal stress technique for lifetime prediction of notched tension bars due to creep crack growth. *Engineering fracture mechanics,* 49**,** 711-726.

HAYHURST, D. R. & LECKIE, F. A. 1984. Behaviour of materials at high temperatures. *Mechanical behaviour of materials- IV***,** 1195-1211.

HAYHURST, J. 2006 *Creep lifetime predictions of welded structures using parallel processing algorithms.* Doctoral Thesis (Ph.D.) University of Manchester.

HUDDLESTON, R. L. 1985. An Improved Multiaxial Creep--Rupture Strength Criterion. *J. Pressure Vessel Technol.(Trans. ASME),* 107**,** 421-429.

HUDDLESTON, R. L. 1993. Assessment of an Improved Multiaxial Strength Theory Based on Creep-Rupture Data for Type 316 Stainless Steel. *Journal of Pressure Vessel Technology,* 115**,** 177-184.

HULL, D. & RIMMER, D. E. 1959. The growth of grain-boundary voids under stress. *Philosophical Magazine,* 4**,** 673-687.

HYDE, T. H., BECKER, A. A., SUN, W. & WILLIAMS, J. A. 2006. Finite-element creep damage analyses of P91 pipes. *International Journal of Pressure Vessels and Piping,* 83**,** 853-863.

HYDE, T. H., SUN, W. & BECKER, A. A. 2004. Effect of geometry change on the creep failure life of a thick-walled CrMoV pipe with a circumferential weldment. *International journal of pressure vessels and piping,* 81**,** 363-371.

HYDE, T. H., XIA, L. & BECKER, A. A. 1996. Prediction of creep failure in aeroengine materials under multi-axial stress states. *International journal of mechanical sciences,* 38**,** 385-403.

ISLAM, N., QUAYYUM, S. & HASSAN, T. A Unified Constitutive Model for High Temperature Multiaxial Creep-Fatigue and Ratcheting Response Simulation of Alloy 617. ASME 2014 Pressure Vessels and Piping Conference, 2014. American Society of Mechanical Engineers, V005T11A026-V005T11A026.

JACOBSON, I., BOOCH, G. & RUMBAUGH, J. 1999. *The unified software development process,* Harlow; Reading, Mass, Addison-Wesley.

JIANG, W., ZHANG, W., ZHANG, G., LUO, Y., ZHANG, Y. C., WOO, W. & TU, S. 2015. Creep damage and crack initiation in P92–BNi2 brazed joint. *Materials & Design,* 72**,** 63-71.

JING, J., MENG, G., SUN, Y. & XIA, S. 2003. An effective continuum damage mechanics model for creep–fatigue life assessment of a steam turbine rotor. *International Journal of Pressure Vessels and Piping,* 80**,** 389-396.

JING, J., SUN, Y., XIA, S. & FENG, G. 2001a. A continuum damage mechanics model on low cycle fatigue life assessment of steam turbine rotor. *International journal of pressure vessels and piping,* 78**,** 59-64.

JING, J., XIA, S. & SUN, Y. 2001b. Nonlinear continuum damage mechanics model on high temperature creep analysis. *JOURNAL OF PROPULSION TECHNOLOGY-BEIJING-,* 22**,** 139-141.

JU-SHIN, H. & GIBBONS, T. B. 1999. The Prediction of Creep Failure in Notched Tubes Using Continuum Damage Mechanics. *ASME-PUBLICATIONS-PVP,* 388**,** 171-176.

KACHANOV, L. M. 1958. On creep rupture time. *Izv. Acad. Nauk SSSR, Otd. Techn. Nauk,* 8**,** 26-31.

KACHANOV, L. M. 1999. Rupture time under creep conditions. *International journal of fracture,* 97**,** 11-18.

KASSNER, M. E. 2009. *Fundamentals of Creep in Metals and Alloys,* GB, Elsevier Science.

KASSNER, M. E. & HAYES, T. A. 2003. Creep cavitation in metals. *International Journal of Plasticity,* 19**,** 1715-1748.

KOWALEWSKI, Z. L., HAYHURST, D. R. & DYSON, B. F. 1994a. Mechanisms-based creep constitutive equations for an aluminium alloy. *The Journal of Strain Analysis for Engineering Design,* 29**,** 309-316.

KOWALEWSKI, Z. L., LIN, J. & HAYHURST, D. R. 1994b. Experimental and theoretical evaluation of a high-accuracy uni-axial creep testpiece with slit extensometer ridges. *International journal of mechanical sciences,* 36**,** 751-769.

KUPFERSCHMID, M. 2009. *Classical Fortran: programming for engineering and scientific applications,* Boca Raton, Fla; London, CRC.

LAI, G. Y. 1990. High temperature corrosion of engineering alloys.

LING, X., TU, S. & GONG, J. 2000. Application of Runge–Kutta–Merson algorithm for creep damage analysis. *International Journal of Pressure Vessels and Piping,* 77**,** 243-248.

LIU, D. 2015. *The development of finite element software for creep damage analysis.* Ph.D. Ph.D. thesis, University of Huddersfield.

LIU, D., XU, Q. & LU, Z. 2013a. Research in the development of finite element software for creep damage analysis. *Journal of communication and computer,* 2013.

LIU, D., XU, Q., LU, Z., BARRANS, S. & GLOVER, I. 2013b. The development of finite element software for creep deformation and damage analysis of weldment. *6th International 'HIDA' Conference: Life/Defect Assessment & Failures in High Temperature Plant.* Nagasaki, Japan.

LIU, D., XU, Q., LU, Z. & XU, D. 2012a. Research in the development of computational FE software for creep damage mechanics. *18th International Conference on Automation and Computing.* loughborough, UK: IEEE.

LIU, D., XU, Q., LU, Z. & XU, D. 2012b. The review of computational FE software for creep damage mechanics. *Advanced Materials Research,* 510**,** 495-499.

LIU, D., XU, Q., LU, Z., XU, D. & TAN, F. 2013c. The development of finite element analysis software for creep damage analysis. *The 2013 World Congress in Computer Science, Computer Engineering, and Applied Computing.* Nevada, USA.

LIU, D., XU, Q., LU, Z., XU, D. & TAN, F. 2013d. The validation of computational FE software for creep damage mechanics. *Advanced Materials Research,* 744**,** 205-210.

LIU, D., XU, Q., LU, Z., XU, D. & XU, Q. 2013e. The techniques in developing finite element software for creep damage analysis. *Advanced Materials Research,* 744**,** 199-204.

MANIE, J. & WOLTHERS, A. 2013. User's Manual - Pre- and Postprocessing. *In:* MANIE, J. & WOLTHERS, A. (eds.). Netherlands: TNO DIANA bv.

MANJOINE, M. J. 1975. Ductility indices at elevated temperature. *Journal of Engineering Materials and Technology,* 97**,** 156-161.

MARGOLIN, B. Z., KARZOV, G. P., SHVETSOVA, V. A. & KOSTYLEV, V. I. 1998. Modelling for transcrystalline and intercrystalline fracture by void nucleation and growth. *Fatigue & Fracture of Engineering Materials & Structures,* 21**,** 123-137.

MERSON, R. H. An operational method for the study of integration processes. Proc. Symp. Data Processing, 1957. 1-25.

NAG 2009. NAG Library Manual, Mark 22.

NI, Y. Z., LAN, X., XU, H. & MAO, X. P. 2015. Study on creep mechanical behaviour of P92 steel under multiaxial stress state. *Materials at High Temperatures*.

ODQVIST, F. K. G. 1974. *Mathematical theory of creep and creep rupture,* Clarendon Press Oxford.

OLSEN, E. W., CONSOLINI, L. & HAUG, M. 2001. *Software quality approaches: testing, verification, and validation,* New York, Springer.

OTHMAN, A. M., HAYHURST, D. R. & DYSON, B. F. 1993. Skeletal point stresses in circumferentially notched tension bars undergoing tertiary creep modelled with physically based constitutive equations. *Proceedings of the Royal Society of London. Series A: Mathematical and Physical Sciences,* 441**,** 343-358.

PERRIN, I. J. & HAYHURST, D. R. 1996. Creep constitutive equations for a 0.5 Cr–0.5 Mo–0.25 V ferritic steel in the temperature range 600–675 C. *The Journal of Strain Analysis for Engineering Design,* 31**,** 299-314.

PERRIN, I. J. & HAYHURST, D. R. 1999. Continuum damage mechanics analyses of type IV creep failure in ferritic steel crossweld specimens. *International journal of pressure vessels and piping,* 76**,** 599-617.

QUAYYUM, S., SENGUPTA, M., CHOI, G., LISSENDEN, C. J. & HASSAN, T. 2014. High Temperature Multiaxial Creep-Fatigue and Creep-Ratcheting Behavior of Alloy 617. *Challenges In Mechanics of Time-Dependent Materials and Processes in Conventional and Multifunctional Materials, Volume 2*. Springer.

RAGAB, A. R. 2002. Creep rupture due to material damage by cavitation. *Journal of engineering materials and technology,* 124**,** 199-205.

RAJ, R. & ASHBY, M. F. 1975. Intergranular fracture at elevated temperature. *Acta Metallurgica,* 23**,** 653-666.

RICE, J. R. & TRACEY, D. M. 1969. On the ductile enlargement of voids in triaxial stress fields∗. *Journal of the Mechanics and Physics of Solids,* 17**,** 201-217.

RICHARDS, R. 2001. *Principles of solid mechanics,* Boca Raton, Fla; London, CRC.

SADEGHBEIGI, R. 2012. *Fluid catalytic cracking handbook: an expert guide to the practical operation, design, and optimization of FCC units*, Elsevier.

SEBESTA, R. W. 2010. *Concepts of programming languages,* Upper Saddle River, N.J; London, Pearson.

SKRZYPEK, J. J. & HETNARSKI, R. B. 1993. *Plasticity and creep: theory, examples, and problems,* Boca Raton,Fla, Begell House.

SMITH, I. M. & GRIFFITHS, D. V. 2004. *Programming the finite element method,* Hoboken, NJ, Wiley.

SPINDLER, M. W. 1994. The Multi-axial Creep of Austenitic Stainless Steels, Nuclear Electric Report TIGM. REP/0014/94.

SPINDLER, M. W. 2004. The multiaxial and uniaxial creep ductility of Type 304 steel as a function of stress and strain rate. *Materials at high Temperatures,* 21**,** 47-54.

SPINDLER, M. W., HALES, R. & SKELTON, R. P. 2001. The multiaxial creep ductility of an ex-service Type 316H stainless steel. *BOOK-INSTITUTE OF MATERIALS,* 769**,** 679-690.

TAN, F., XU, Q., LU, Z., BARRANS, S. 2013a. The development of computational FE system for creep damage analysis of weldment. *6th International 'HIDA' Conference: Life/Defect Assessment & Failures in High Temperature Plant*. Nagasaki, Japan.

TAN, F., XU, Q., LU, Z. & LIU, D. 2013b. Practical guidance on the application of RK integration method in finite element analysis of creep damage problem. *The 2013 World Congress in Computer Science, Computer Engineering, and Applied Computing*. Nevada, USA.

TAN, F., XU, Q., LU, Z. & XU, D. 2012a. Literature review on the development of computational software system for creep damage analysis for weldment. *Advanced Materials Research,* 510**,** 490-494.

TAN, F., XU, Q., LU, Z. & XU, D. 2012b. The preliminary development of computational software system for creep damage analysis in weldment. *18th International Conference on Automation and Computing.* loughborough, UK: IEEE.

TAN, F., XU, Q., LU, Z., XU, D. & LIU, D. 2013c. The Validation of Computational Software System for Creep Damage Analysis. *Advanced Materials Research,* 744**,** 449-454.

TU, S. 2003. *High temperature structural integrity,* Beijing, Scientific press.

TU, S. 2005. New Need of structural integrity technology for high temperature applications. *Journal of Pressure Equipment and Systems,* 3**,** 26-39.

TU, S. 2007. Emerging challenges to structural integrity technology for high-temperature applications. *Frontiers of Mechanical Engineering in China,* 2**,** 375-387.

TVERGAARD, V. & NEEDLEMAN, A. 1984. Analysis of the cup-cone fracture in a round tensile bar. *Acta metallurgica,* 32**,** 157-169.

VOYIADJIS, G. Z. & KATTAN, P. I. 2002. *Damage mechanics with finite elements: practical applications with computer tools,* New York, Springer.

WALLIS, P. J. L. 1985. *The Software development process,* Maidenhead, Pergamon Infotech.

WANG, J. M., XUE, Y. W., LI, W. H., WEI, A. Z. & CAO, Y. F. 2014. Study on creep characteristics of oil film bearing Babbitt. *Materials Research Innovations,* 18**,** S2-16-S2-21.

WANG, X. & WANG, X. 1996. Finite element analysis on creep damage. *Computers & Structures,* 60**,** 781-786.

WANG, X., XU, Q., YU, S., LIU, H., HU, L. & REN, Y. 2015. Laves-phase evolution during aging in fine grained heat-affected zone of a tungsten-strengthened 9% Cr steel weldment. *Journal of Materials Processing Technology,* 219**,** 60-69.

WEN, J., TU, S., GAO, X. & REDDY, J. N. 2014. New model for creep damage analysis and its application to creep crack growth simulations. *Materials Science and Technology,* 30**,** 32-37.

XU, Q. 2001. Creep damage constitutive equations for multi-axial states of stress for 0.5Cr0.5Mo0.25V ferritic steel at 590°C. *Theoretical and Applied Fracture Mechanics,* 36**,** 99-107.

XU, Q. 2004. The development of validation methodology of multi-axial creep damage constitutive equations and its application to 0.5Cr0.5Mo0.25V ferritic steel at 590 °C. *Nuclear Engineering and Design*, 228**,** 97-106.

YAO, H., XUAN, F., WANG, Z. & TU, S. 2007. A review of creep analysis and design under multi-axial stress states. *Nuclear Engineering and Design*, 237**,** 1969-1986.

YU, Q., YUE, Z. & WEN, Z. 2008. Creep damage evolution in a modeling specimen of nickel-based single crystal superalloys air-cooled blades. *Materials Science and Engineering: A*, 477**,** 319-327.

YUAN, K., HAN, Z., ZHOU, H. & ZHONG, Y. 2014. Ultrasonic Inspection for Metal Creep Based on Flight-Time-Attenuation. *Applied Mechanics and Materials*, 599**,** 164-168.

ZHANG, J., WANG, G., XUAN, F. & TU, S. 2015. The influence of stress-regime dependent creep model and ductility in the prediction of creep crack growth rate in Cr–Mo–V steel. *Materials & Design*, 65**,** 644-651.

ZHAO, L., JING, H., XU, L., AN, J. & XIAO, G. 2012. Numerical investigation of factors affecting creep damage accumulation in ASME P92 steel welded joint. *Materials & Design*, 34**,** 566-575.

ZHU, H., WEI, T., CARR, D., HARRISON, R., EDWARDS, L., SEO, D., MARUYAMA, K. & DARGUSCH, M. S. 2014. Microstructural design for thermal creep and radiation damage resistance of titanium aluminide alloys for high-temperature nuclear structural applications. *Current Opinion in Solid State and Materials Science*, 18**,** 269-278.

10 APPENDICES

10.1 Publication list of this research

Seven papers were published during this research, only five is the first author:

1) Tan, F., Xu, Q., Lu, Z. Y., & Xu, D. L. (2012). Literature review on the development of computational software system for creep damage analysis for weldment. Advanced Materials Research, 510, 490-494.
2) Tan, F., Xu, Q., Lu, Z., & Xu, D. (2012). The preliminary development of computational software system for creep damage analysis in weldment. Paper presented at the 18th International Conference on Automation and Computing, loughborough, UK.
3) Tan, F., Xu, Q., Lu, Z. Y., Xu, D. L., & Liu, D. Z. (2013). The Validation of Computational Software System for Creep Damage Analysis. Advanced Materials Research, 744, 449-454.
4) Tan, F., Xu, Q., Lu, Z., Xu, D., & Liu, D. (2013). Practical guidance on the application of RK integration method in finite element analysis of creep damage problem. Paper presented at the The 2013 World Congress in Computer Science, Computer Engineering, and Applied Computing, Nevada, USA.
5) Tan, F., Xu, Q., Lu, Z., Barrans, S. (2013). The development of computational FE system for creep damage analysis of weldment. Paper presented at the 6th International 'HIDA' Conference: Life/Defect Assessment & Failures in High Temperature Plant, Nagasaki, Japan.
6) Liu, D. Z., Xu, Q., Lu, Z. Y., Xu, D. L., & Tan, F. (2013). The validation of computational FE software for creep damage mechanics. Advanced Materials Research, 744, 205-210.
7) Liu, D., Xu, Q., Lu, Z., Xu, D., & Tan, F. (2013). The development of finite element analysis software for creep damage analysis. Paper presented at the The 2013 World Congress in Computer Science, Computer Engineering, and Applied Computing, Nevada,USA. Reference

10.2 Source Code of Tan_library，NLC and DTI

```
      module tan_library
        contains
        !
        ! Purpose:
        !   Coded for the develpment of HITSI, and it includes:
        !     1. KR, PH, QX (constitutive equations)
        !     2. EULER, RK4, RKM, RKF (numerical integration method)
        !     3. TRS (transformation of stress tensor)
        !     4. TSC (time-step control procedure)
        !     5. NOR_KR (normalization scheme)
        ! Record of revisions:
        !     Date           Programmer         Description of change
        !     ====           ==========         =====================
        !  21/09/2014          F. Tan               Original code
        !
      subroutine TRS (sigma,stress)
        implicit none
        doubleprecision, intent(in) :: sigma(:)
        doubleprecision, intent(out) :: stress(:)
        doubleprecision :: sx, sy, sz, txy, tyz, tzx, pi, j2, j3,
     &  sig0, loang, es
        doubleprecision, dimension(3) :: ps
        integer :: nos
        nos = ubound(sigma,1)
        pi = 3.1415926
!---------------------stress terms rearrangement---------------------
        select case (nos)
        case (4)
            sx = sigma(1); sy = sigma(2); sz = sigma(4)
            txy = sigma(3); tyz = 0.0; tzx = 0.0
            case (6)
                sx = sigma(1); sy = sigma(2); sz = sigma(3)
                txy = sigma(4); tyz = sigma(5); tzx = sigma(6)
                case default
                    print*, "Error on stress rearrangement in TRS"
                    end select
!----------------------------hydrostatic-----------------------------
        sig0 = (sx+sy+sz)/3.
!----------------------------invariants------------------------------
        j2 = ((sx-sy)**2+(sy-sz)**2+(sz-sx)**2)/6.+txy**2+tyz**2+
     &       tzx**2
        j3 = (sx-sig0)*(sy-sig0)*(sz-sig0)+2*txy*tyz*tzx-
     &      (sx-sig0)*tyz**2-(sz-sig0)*txy**2-(sy-sig0)*tzx**2
!----------------------------load angle------------------------------
        loang = asin((-sqrt(27.)*j3)/(2.*sqrt(j2**3)))/3.
!-------------------------principal stress---------------------------
        ps(1) = 2.*sqrt(j2)/sqrt(3.)*sin(loang+2.*pi/3.)+sig0
        ps(2) = 2.*sqrt(j2)/sqrt(3.)*sin(loang)+sig0
        ps(3) = 2.*sqrt(j2)/sqrt(3.)*sin(loang-2.*pi/3.)+sig0
!-------------------------equivalent stress--------------------------
        es = 1/sqrt(2.)*
     &      sqrt((ps(1)-ps(2))**2+(ps(2)-ps(3))**2+(ps(3)-ps(1))**2)
!-------------------------result updatiing----------------------------
        select case (nos)
        case (4)
            stress(1) = sx-sig0; stress(2) = sy-sig0; stress(3) = txy
            stress(4) = sz-sig0; stress(5) = ps(1); stress(6) = ps(2)
            stress(7) = ps(3); stress(8) = es
        case (6)
```

```
            stress(1) = sx-sig0; stress(2) = sy-sig0; stress(4) = txy
            stress(3) = sz-sig0; stress(5) = tyz; stress(6) = tzx
            stress(7) = ps(1); stress(8) = ps(2); stress(9) = ps(3)
            stress(10) = es
            case default
                print*, "wrong size for nos in TRS"
                end select
                return
                end subroutine TRS

      subroutine EULER (func,y,t,dt,stress,mat,nos,nom,noe)
        implicit none
        external :: func
        integer, intent(in) :: nos, nom, noe
        doubleprecision, intent(inout) :: y(noe)
        doubleprecision, intent(in) :: stress(nos), mat(nom)
        doubleprecision, intent(in) :: t, dt
        doubleprecision :: k(noe)
!-------------------------derivative solving-------------------------
        call func(k,y,t,stress,mat,nos,nom,noe)
!-------------------------increment update---------------------------
        y = y+k*dt
        return
        end subroutine EULER

      subroutine RK4 (func,y,t,dt,stress,mat,nos,nom,noe)
        implicit none
        external :: func
        integer, intent(in) :: nos, nom, noe
        doubleprecision, intent(inout) :: y(noe)
        doubleprecision, intent(in) :: stress(nos), mat(nom)
        doubleprecision, intent(in) :: t, dt
          doubleprecision :: k1(noe), k2(noe), k3(noe), k4(noe),
     &                       mfs(noe)
!-------------------------derivative solving-------------------------
          call func(k1,y,t,stress,mat,nos,nom,noe)
          call func(k2,y+dt/2*k1,t+dt/2,stress,mat,nos,nom,noe)
          call func(k3,y+dt/2*k2,t+dt/2,stress,mat,nos,nom,noe)
          call func(k4,y+dt*k3,t+dt,stress,mat,nos,nom,noe)
!-------------------------increment update---------------------------
          mfs = (k1+2*k2+2*k3+k4)/6
          y = y+mfs*dt
          return
          end subroutine RK4

      subroutine RKM (func,y,t,dt,stress,mat,nos,nom,noe,rcv)
        implicit none
        external :: func
        integer, intent(in) :: nos, nom, noe
        doubleprecision, intent(inout) :: y(noe)
        doubleprecision, intent(in) :: stress(nos), mat(nom)
        doubleprecision, intent(in) :: t, dt
        integer, intent(out) :: rcv
        doubleprecision :: maoi
        doubleprecision :: k1(noe), k2(noe), k3(noe), k4(noe),
     &                     k5(noe), mfs(noe), loer(noe), aoi(noe)
!-------------------------derivative solving-------------------------
          call func(k1,y,t,stress,mat,nos,nom,noe)
          call func(k2,y+dt/3.*k1,t+dt/3.,stress,mat,nos,nom,noe)
          call func(k3,y+dt/6.*(k1+k2),t+dt/3.,stress,mat,nos,nom,
     &              noe)
```

```
          call func(k4,y+dt/8.*(k1+3*k3),t+dt/2.,stress,mat,nos,nom,
     &              noe)
          call func(k5,y+dt/2.*(k1-3.*k3+4.*k4),t+dt,stress,mat,nos,
     &              nom,noe)
!-------------------------increment update-------------------------
          mfs = (k1+4.*k4+k5)/6.
          y = y+mfs*dt
!----------------------self-adaptive technique----------------------
          loer = (2*k1-9*k3+8*k4-k5)/30.
          aoi = loer/mfs
          maoi = maxval(aoi)
          if (maoi<0.001) then
            rcv = 0
            else
                rcv = 1
                end if
                return
                end subroutine RKM

      subroutine RKF (func,y,t,dt,stress,mat,nos,nom,noe,rcv)
        implicit none
        external :: func
        integer, intent(in) :: nos, nom, noe
        doubleprecision, intent(inout) :: y(noe)
        doubleprecision, intent(in) :: stress(nos), mat(nom)
        doubleprecision, intent(in) :: t, dt
        integer, intent(out) :: rcv
        doubleprecision :: maoi
        doubleprecision :: k1(noe), k2(noe), k3(noe), k4(noe),
     &                     k5(noe), k6(noe), mfs4(noe), mfs5(noe),
     &                     aoi(noe), ybar(noe)
!-------------------------derivative solving-----------------------
        call func(k1,y,t,stress,mat,nos,nom,noe)
        call func(k2,y+dt/4.*k1,t+dt/4.,stress,mat,nos,nom,noe)
        call func(k3,y+dt/32.*(3.*k1+9.*k2),t+3.*dt/8.,stress,mat,
     &            nos,nom,noe)
        call func(k4,y+dt/2197.*(1932.*k1-7200.*k2+7296.*k3),
     &            t+12.*dt/13.,stress,mat,nos,nom,noe)
        call func(k5,y+dt/4104.*(8341.*k1-32832.*k2+29440.*k3-
     &            845.*k4),t+dt,stress,mat,nos,nom,noe)
        call func(k6,y+dt*(-(8./27.)*k1+2*k2-(3544./2565.)*k3+(1859./
     &           4104.)*k4-(11./40.)*k5),t+dt/2.,stress,mat,nos,nom,
     &           noe)
        mfs4 = (25./216.)*k1+(1408./2565.)*k3+(2197./4104.)*k4-
     &         (1./5.)*k5
        mfs5 = (16./135.)*k1+(6656./12825.)*k3+(28561./56430.)*k4-
     &         (9./50.)*k5+(2./55.)*k6
!-------------------------increment update-------------------------
        ybar = y+mfs4*dt
        y = y+mfs5*dt
!----------------------self-adaptive technique----------------------
        aoi = (dt*1e-5/(2.*abs(y-ybar)))**0.25
        maoi = maxval(aoi)
        if (maoi<1) then
            rcv = 1
            else
                rcv = 0
                end if
                return
                end subroutine RKF
```

```
      subroutine KR (f,x,t,stress,mat,nos,nom,noe)
        implicit none
        integer, intent(in) :: nos, nom, noe
        doubleprecision, intent(in) :: stress(nos), mat(nom), x(noe)
        doubleprecision, intent(in) :: t
        doubleprecision, intent(out) :: f(noe)
        doubleprecision :: sx, sy, sz, txy, tyz, tzx, mps, es
        doubleprecision :: A, n, m, B, phi, chi, alpha, rs
!---------------------stress terms rearrangement---------------------
        select case (nos)
        case (1)
            sx = stress(1); sy = 0.0; sz = 0.0; txy = 0.0
            tyz = 0.0; tzx = 0.0; mps = 0.0; es = 0.0
            case (8)
                sx = stress(1); sy = stress(2); txy = stress(3)
                sz = stress(4); mps = stress(5); es = stress(8)
                tyz = 0.0; tzx = 0.0
                case (10)
                    sx = stress(1); sy = stress(2); sz = stress(3)
                    txy = stress(4); tyz = stress(5); tzx = stress(6)
                    mps = stress(7); es = stress(10)
                    case default
                        print*, "Error on stress rearrangement in KR"
                        end select
!------------------material properties rearrangement-----------------
        A=mat(1); n=mat(2); m=mat(3); B=mat(4); phi=mat(5); chi=mat(6)
        alpha=mat(7); rs=alpha*mps+(1.-alpha)*es
!--------------------creep constitutive equations--------------------
        select case (noe)
        case (2)
            f(1)=A*((sx/(1-x(2)))**n)*(t**m)
            f(2)=B*(sx**chi)/((1-x(2))**phi)*(t**m)
            case (5)
                f(1)=(3./2.)*(sx/es)*A*((es/(1-x(5)))**n)*(t**m)
                f(2)=(3./2.)*(sy/es)*A*((es/(1-x(5)))**n)*(t**m)
                f(3)=(3./2.)*(txy/es)*A*((es/(1-x(5)))**n)*(t**m)
                f(4)=(3./2.)*(sz/es)*A*((es/(1-x(5)))**n)*(t**m)
                f(5)=B*(rs**chi)/((1-x(5))**phi)*(t**m)
                case (7)
                    f(1)=(3./2.)*(sx/es)*A*((es/(1-x(7)))**n)*(t**m)
                    f(2)=(3./2.)*(sy/es)*A*((es/(1-x(7)))**n)*(t**m)
                    f(3)=(3./2.)*(sz/es)*A*((es/(1-x(7)))**n)*(t**m)
                    f(4)=(3./2.)*(txy/es)*A*((es/(1-x(7)))**n)*(t**m)
                    f(5)=(3./2.)*(tyz/es)*A*((es/(1-x(7)))**n)*(t**m)
                    f(6)=(3./2.)*(tzx/es)*A*((es/(1-x(7)))**n)*(t**m)
                    f(7)=B*(rs**chi)/((1-x(7))**phi)*(t**m)
                    case default
                    print*, "Error on constitutive equations in KR"
                        end select
                        return
                        end subroutine KR

      subroutine PH (f,x,t,stress,mat,nos,nom,noe)
        implicit none
        integer, intent(in) :: nos, nom, noe
        doubleprecision, intent(in) :: stress(nos), mat(nom), x(noe)
        doubleprecision, intent(in) :: t
        doubleprecision, intent(out) :: f(noe)
        doubleprecision :: sx, sy, sz, txy, tyz, tzx, mps, es
        doubleprecision :: A, B, C, h, Hstar, Kc, v
        integer :: N
```

```
!---------------------stress terms rearrangement---------------------
        select case (nos)
        case (1)
            sx = stress(1); sy = 0.0; sz = 0.0; txy = 0.0
            tyz = 0.0; tzx = 0.0; mps = 0.0; es = 0.0
            case (8)
                sx = stress(1); sy = stress(2); txy = stress(3)
                sz = stress(4); mps = stress(5); es = stress(8)
                tyz = 0.0; tzx = 0.0
                case (10)
                    sx = stress(1); sy = stress(2); sz = stress(3)
                    txy = stress(4); tyz = stress(5); tzx = stress(6)
                    mps = stress(7); es = stress(10)
                    case default
                        print*, "Error on stress rearrangement in PH"
                        end select
!------------------material properties rearrangement-----------------
        A=mat(1); B=mat(2); C=mat(3); h=mat(4); Hstar=mat(5)
         Kc=mat(6); v=mat(7)
        if (mps>0) then
            N=1
            else if (mps<=0) then
                N=0
                end if
!---------------------creep constitutive equations--------------------
        select case (noe)
        case (4)
            f(1) = A*sinh((B*sx*(1-x(2)))/((1-x(3))*(1-x(4))))
            f(2) = h*f(1)/sx*(1.-(x(2)/Hstar))
            f(3) = Kc/3.*(1-x(3))**4
            f(4) = C*f(1)
        case (8)
            f(1) = (3./2.)*(sx/es)*A*
     &             sinh((B*es*(1-x(6)))/((1-x(7))*(1-x(8))))
            f(2) = (3./2.)*(sy/es)*A*
     &             sinh((B*es*(1-x(6)))/((1-x(7))*(1-x(8))))
            f(3) = (3./2.)*(txy/es)*A*
     &             sinh((B*es*(1-x(6)))/((1-x(7))*(1-x(8))))
            f(4) = (3./2.)*(sz/es)*A*
     &             sinh((B*es*(1-x(6)))/((1-x(7))*(1-x(8))))
            f(5) = sqrt((2./3.)*(f(1)**2+f(2)**2+2*f(3)**2+f(4)**2))
            f(6) = h*f(5)/es*(1.-(x(6)/Hstar))
            f(7) = Kc/3.*(1.-x(7))**4
            f(8) = C*N*f(5)*(mps/es)**v
            case (10)
                f(1) = (3./2.)*(sx/es)*A*
     &                 sinh((B*es*(1-x(8)))/((1-x(9))*(1-x(10))))
                f(2) = (3./2.)*(sy/es)*A*
     &                 sinh((B*es*(1-x(8)))/((1-x(9))*(1-x(10))))
                f(3) = (3./2.)*(sz/es)*A*
     &                 sinh((B*es*(1-x(8)))/((1-x(9))*(1-x(10))))
                f(4) = (3./2.)*(txy/es)*A*
     &                 sinh((B*es*(1-x(8)))/((1-x(9))*(1-x(10))))
                f(5) = (3./2.)*(tyz/es)*A*
     &                 sinh((B*es*(1-x(8)))/((1-x(9))*(1-x(10))))
                f(6) = (3./2.)*(tzx/es)*A*
     &                 sinh((B*es*(1-x(8)))/((1-x(9))*(1-x(10))))
                f(7) = sqrt((2./3.)*(f(1)**2+f(2)**2+f(3)**2+
     &                   2*f(4)**2+2.*f(5)**2+2*f(6)**2))
                f(8)=h*f(7)/es*(1.-(x(8)/Hstar))
                f(9)=Kc/3.*(1-x(9))**4
```

```
                f(10)=C*N*f(7)*(mps/es)**v
                case default
                    print*, "Error on constitutive equations in PH"
                    end select
                    return
                    end subroutine PH

      subroutine QX (f,x,t,stress,mat,nos,nom,noe)
        implicit none
        integer, intent(in) :: nos, nom, noe
        doubleprecision, intent(in) :: stress(nos), mat(nom), x(noe)
        doubleprecision, intent(in) :: t
        doubleprecision, intent(out) :: f(noe)
        doubleprecision :: sx, sy, sz, txy, tyz, tzx, ps1, ps2, ps3,
     &                     es, Ss, sm, S1
        doubleprecision :: A, B, C, h, Hstar, Kc, v, a1, b1, p, q
        integer :: N
!---------------------stress terms rearrangement---------------------
        select case (nos)
        case (8)
            sx = stress(1); sy = stress(2); txy = stress(3)
            sz = stress(4); ps1 = stress(5); ps2 = stress(6)
            ps3 = stress(7); es = stress(8); tyz = 0.0; tzx = 0.0
            case (10)
                sx = stress(1); sy = stress(2); sz = stress(3)
                txy = stress(4); tyz = stress(5); tzx = stress(6)
                ps1 = stress(7); ps2 = stress(8); ps3 = stress(9)
                es = stress(10)
                case default
                    print*, "Error on stress rearrangement in QX"
                    end select
!------------------material properties rearrangement-----------------
        A = mat(1); B = mat(2); C = mat(3); h = mat(4);
        Hstar = mat(5); Kc = mat(6); v = mat(7); a1 = mat(8);
        b1 = mat(9); p = mat(10); q = mat(11)
        sm = (ps1+ps2+ps3)/3; Ss = sqrt(ps1**2+ps2**2+ps3**2)
        S1 = ps1-sm
        if (ps1>0) then
            N=1
            else if (ps1<=0) then
                N=0
                end if
!--------------------creep constitutive equations--------------------
        select case (noe)
        case (9)
            f(1) = (3./2.)*(sx/es)*A*
     &             sinh((B*es*(1-x(6)))/((1-x(7))*(1-x(9))))
            f(2) = (3./2.)*(sy/es)*A*
     &             sinh((B*es*(1-x(6)))/((1-x(7))*(1-x(9))))
            f(3) = (3./2.)*(txy/es)*A*
     &             sinh((B*es*(1-x(6)))/((1-x(7))*(1-x(9))))
            f(4) = (3./2.)*(sz/es)*A*
     &             sinh((B*es*(1-x(6)))/((1-x(7))*(1-x(9))))
            f(5) = sqrt((2./3.)*(f(1)**2+f(2)**2+2*f(3)**2+f(4)**2))
            f(6) = h*f(5)/es*(1.-(x(6)/Hstar))
            f(7) = Kc/3.*(1.-x(7))**4
            f(8) = C*N*f(5)*
     &             (exp(p*(1-(ps1/es))+q*(0.5-1.5*(sm/es))))**(-1)
            f(9) = f(8)*((2./3.)*(es/S1))**a1*exp(b1*(3.*sm/Ss-1.))
            case (11)
                f(1) = (3./2.)*(sx/es)*A*
```

```
     &                 sinh((B*es*(1-x(8)))/((1-x(9))*(1-x(11))))
                f(2) = (3./2.)*(sy/es)*A*
     &                 sinh((B*es*(1-x(8)))/((1-x(9))*(1-x(11))))
                f(3) = (3./2.)*(sz/es)*A*
     &                 sinh((B*es*(1-x(8)))/((1-x(9))*(1-x(11))))
                f(4) = (3./2.)*(txy/es)*A*
     &                 sinh((B*es*(1-x(8)))/((1-x(9))*(1-x(11))))
                f(5) = (3./2.)*(tyz/es)*A*
     &                 sinh((B*es*(1-x(8)))/((1-x(9))*(1-x(11))))
                f(6) = (3./2.)*(tzx/es)*A*
     &                 sinh((B*es*(1-x(8)))/((1-x(9))*(1-x(11))))
                f(7) = sqrt((2./3.)*(f(1)**2+f(2)**2+f(3)**2+
     &                   2*f(4)**2+2.*f(5)**2+2*f(6)**2))
                f(8) = h*f(7)/es*(1.-(x(8)/Hstar))
                f(9) = Kc/3.*(1-x(9))**4
                f(10) = C*N*f(7)*
     &                (exp(p*(1-(ps1/es))+q*(0.5-1.5*(sm/es))))**(-1)
                f(11) =f(10)*((2./3.)*(es/S1))**a1*
     &                 exp(b1*(3.*sm/Ss-1.))
                case default
                    print*, "Error on constitutive equations in QX"
                    end select
                    return
                    end subroutine QX

      subroutine TSC (dt,g_rcv)
        implicit none
        integer, intent(in) :: g_rcv(:,:,:)
        doubleprecision, intent(inout) :: dt
        doubleprecision :: drcv
        drcv = maxval(g_rcv)
!-------------------------reduce time-step-------------------------
        if (drcv == 1) then
            dt = dt/2
            else if (drcv == 0) then
                dt = dt
                end if
                end subroutine TSC

      subroutine NOR_KR (f,x,t,stress,mat,nos,nom,noe)
        implicit none
        integer, intent(in) :: nos, nom, noe
        doubleprecision, intent(in) :: stress(nos), mat(nom),
     &                                 x(noe)
        doubleprecision, intent(in) :: t
        doubleprecision, intent(out) ::  f(noe)
        doubleprecision :: sx, sy, sz, txy, tyz, tzx, mps, es, cfs
        doubleprecision :: A, n, m, B, phi, chi, alpha, rs, e, ip
!--------------------stress terms rearrangement--------------------
        select case (nos)
        case (9)
            sx = stress(1); sy = stress(2); txy = stress(3)
            sz = stress(4); mps = stress(5); es = stress(8)
            tyz = 0.0; tzx = 0.0; ip = stress(9)
            case (11)
                sx = stress(1); sy = stress(2); sz = stress(3)
                txy = stress(4); tyz = stress(5); tzx = stress(6)
                mps = stress(7); es = stress(10); ip = stress(11)
                case default
                    print*, "Error on stress rearrangement in NOR_KR"
                       end select
```

```
!------------------material properties rearrangement-----------------
        A = mat(1); n = mat(2); m = mat(3); B = mat(4); phi = mat(5)
        chi = mat(6); alpha = mat(7); e =  mat(8)
        rs = alpha*mps+(1.-alpha)*es; cfs = (A*e/B)*(ip**(n-chi-1.))
!--------------------creep constitutive equations--------------------
        select case (noe)
        case (5)
            f(1)=(3./2.)*(sx/es)*((es/(1-x(5)))**n)
            f(2)=(3./2.)*(sy/es)*((es/(1-x(5)))**n)
            f(3)=(3./2.)*(txy/es)*((es/(1-x(5)))**n)
            f(4)=(3./2.)*(sz/es)*((es/(1-x(5)))**n)
            f(5) = (rs**chi)/(cfs*(1.+phi)*((1.-x(5))**phi))
            case (7)
                f(1)=(3./2.)*(sx/es)*((es/(1-x(7)))**n)
                f(2)=(3./2.)*(sy/es)*((es/(1-x(7)))**n)
                f(3)=(3./2.)*(sz/es)*((es/(1-x(7)))**n)
                f(4)=(3./2.)*(txy/es)*((es/(1-x(7)))**n)
                f(5)=(3./2.)*(tyz/es)*((es/(1-x(7)))**n)
                f(6)=(3./2.)*(tzx/es)*((es/(1-x(7)))**n)
                f(7) = (rs**chi)/(cfs*(1.+phi)*((1.-x(7))**phi))
                case default
                    print*, "Error on constitutive equations in KR"
                    end select
                    return
                    end subroutine NOR_KR
        end module tan_library

      program NLC
        !
        ! Purpose:
        !   To calculate the nodal force of each loaded node
        !
        ! Record of revisions:
        !     Date           Programmer         Description of change
        !     ====           ==========         =====================
        !  21/09/2014          F. Tan               Original code
        !
        implicit none
        character(len=16) :: filename1, protype
        integer :: nn, i, ierror
        doubleprecision :: p
        doubleprecision, allocatable :: f1(:),f2(:),r1(:),r0(:),f(:)
        write(*,*) 'Please Enter The Input File Name: '
        read (*,*) filename1
        open(10,file=filename1,status='old',action='read',
     &       iostat=ierror)
        open(11,file='loadsdata.res',status='replace',action='write',
     &       iostat=ierror)
!-------------------------read input file--------------------------
        read (10,*) protype
        read (10,*) nn, p
        allocate (f1(nn), f2(nn), r0(nn), r1(nn), f(nn+1))
        read (10,*) (r0(i), i=1,nn)
        read (10,*) (r1(i), i=1,nn)
!------------------assemble nodal loads of each node----------------
        select case (protype)
        case ('axisymmetric')
        do i = 1, nn
            f1(i) = (r1(i)**2+r0(i)*r1(i)-2*r0(i)**2)/6
            f2(i) = (2*r1(i)**2-r0(i)*r1(i)-r0(i)**2)/6
            end do
```

```
            case ('planar')
                do i = 1, nn
                    f1(i) = (r1(i)-r0(i))/2
                    f2(i) = f1(i)
                    end do
                    case default
                        print*, "wrong number of nodes in NLC"
                        end select
!-----------------assemble nodal loads of each elememt--------------
        f = 0.
        do i = 1, nn
            f(i) = f1(i)
            end do
            do i = 1, nn
                f(i+1) = f(i+1)+f2(i)
                end do
                f=f*p
!-----------------output nodal loads of loaded nodes----------------
                select case (protype)
                case ('axisymmetric')
                    write(11,*) "Nodal Force From Inner to Outer:"
                    case ('planar')
                        write(11,*) "Nodal Force From First to Last:"
                        end select
                        do i = 1, nn+1
                            write (11,'(i5,es14.7)') i, f(i)
                            end do
                            end program NLC

      program DTI
        !
        ! Purpose:
        !   To transfer the data from neutral file of FEMGV to HITSI,
        !   and to transfer the data from result file of HITSI to
        !   FEMGV
        !
        ! Record of revisions:
        !     Date           Programmer         Description of change
        !     ====           ==========         =====================
        !  21/09/2014          F. Tan               Original code
        !
        implicit none
        integer :: ierror
        character (len=30) :: filename1, filename2, model_name
        integer :: key, iformt, maxout, nn, i, lpnr, types, ne0, ne1,
     &    nant, nels, group, material, variant, physical, nod, k,
     &    nant1, dof1, dof2, dof3, lcase, elem, dof, j
        character :: opcode, n1, n2, n3, n4, n5, n6, n7, n8
        doubleprecision, dimension(3) :: coordinate
        integer, dimension(10) :: nodes
        doubleprecision, allocatable :: g_coord(:,:)
        integer, allocatable :: num(:), g_num1(:,:), g_num(:,:),
     &    g_mat(:), nf(:,:), loadnode(:)
        character (len=15) :: etype, aname
        character (len=30) :: modelname,transfertype
        integer ::   nst, nip, node, elemt, point, ndim,
     &     mid
        doubleprecision :: time,e,v, value1
        character (len=15) :: element
        doubleprecision, allocatable ::  g_dispmt(:,:),
     &  g_reforce(:,:), g_points(:,:,:), g_sigma(:,:,:), g_eps(:,:,:)
```

```
     &   , g_evp(:,:,:), g_damage(:,:)
         integer, allocatable :: material1(:)
         integer :: key1=111, key2=2, key3=3, key4=20, key5=100,
     &   key6=-1, key7=-2, key8=-3, group1=1, nsc=0, nlc=0, key9=-4,
     &   key10=-5, key11=9999, iformt1=2
         integer :: ncomps1=2,irtype=1,norcty=0,menu=1,ictype=2,
     &   icind1=1, icind2=1, iexist=0, ncomps2=4, icind3=3
         character :: opcode1='C'
         write (*,*) 'Please Enter The Transfer Processing'
         read (*,*) transfertype
         select case (transfertype)
!-----------------------pre-processing transfer----------------------
         case ('pre-processing')
         write (*,*) 'Please Enter The Input File Name: '
         read (*,*) filename1
         write (*,*) 'Please Enter The Output File Name: '
         read (*,*) filename2
         open (unit=10, file=filename1, status='old', action='read',
     &   iostat=ierror)
         open (unit=11, file=filename2, status='replace',
     &   action='write', iostat=ierror)
         do
             read (10,99990) key
             if (key==9999) exit
             select case (key)
             case (1)
                 backspace (unit=10)
                 read (10,99999) key, opcode, model_name
             case (2)
                 open (unit=12, file='temp.txt')
                 backspace (unit=10)
                 read (10,99998) key, opcode, n1, n2, n3, n4, n5, n6,
     &           iformt, maxout
                 do
                     read (10,99989) key
                     if (key==-3) exit
                     backspace (unit=10)
                     read (10,99997) key, nn, coordinate(:)
                     write(12,99988) coordinate(:)
                     end do
                     allocate (g_coord(3,nn))
                     rewind (unit=12)
                     do i=1,nn
                         read (12,99988) g_coord(:,i)
                         end do
                         close (unit=12, status='delete')
             case (101)
                 backspace (unit=10)
                 read (10,99998) key, opcode, n1, n2, n3, n4, n5, n6,
     &           iformt, maxout
                 read (10,99996) key, n1, n2, n3, n4, n5, n6, n7, n8,
     &           lpnr, types, ne0, ne1, nant
                 do
                     read (10,99989) key
                     if (key==-3) exit
                     backspace (unit=10)
                     read (10,99995) key, nodes(:)
                     end do
             case (3)
                 open (unit=12, file='temp.dat')
                 open (unit=13, file='temp1.dat')
```

```
            backspace (unit=10)
            read (10,99998) key, opcode, n1, n2, n3, n4, n5, n6,
&           iformt, maxout
            do
                read (10,99989) key
                if (key==-3) exit
                backspace (unit=10)
                read (10,99994) key, nels, types, group,
&               material, variant, physical
                if (types==3) then
                    nod = 3
                    else if (types==5) then
                        nod = 4
                        else if (types==13) then
                            nod = 8
                            else
                                write (*,*) 'Wrong element type'
                                end if
                allocate (num(nod))
                read (10,99995) key, num(:)
                write (12,99987) num(:)
                write (13,99986) material
                deallocate (num)
                end do
                allocate (g_num(nod,nels),g_mat(nels))
                rewind (unit=12)
                rewind (unit=13)
                do i=1,nels
                    read (12,99987) g_num(:,i)
                    read (13,99986) g_mat(i)
                    end do
                    close (unit=12, status='delete')
                    close (unit=13, status='delete')
        case (103)
        backspace (unit=10)
        read (10,99998) key, opcode, n1, n2, n3, n4, n5, n6,
&       iformt, maxout
        allocate (nf(3,nn))
        nf = 1; nant1 = 0
        do
            read (10,99989) key
            if (key==-3) exit
            backspace (unit=10)
            nodes = 0
            read (10,99985) key,aname,lpnr,nant,dof1,dof2,dof3
            read (10,99995) key, nodes(:)
            nant1=nant1+nant
            if (dof1 == 0) then
                nf = nf
                else if (dof1 == 1) then
                    open (unit=12, file='temp.txt')
                    do i = 1, nant
                        write (12,99993) nodes(i),dof1-1
                        end do
                        rewind (unit=12)
                        do i = 1, nant
                            read (12,99993)k, nf(1,k)
                            end do
                            close (unit=12, status='delete')
                            end if
            if (dof2 == 0) then
```

```
                    nf = nf
                    else if (dof2 == 1) then
                        open (unit=12, file='temp.txt')
                        do i = 1, nant
                            write (12,99993) nodes(i),dof2-1
                            end do
                            rewind (unit=12)
                            do i = 1, nant
                                read (12,99993)k, nf(2,k)
                                end do
                                close (unit=12, status='delete')
                                end if
                if (dof3 == 0) then
                    nf = nf
                    else if (dof3 == 1) then
                        open (unit=12, file='temp.txt')
                        do i = 1, nant
                            write (12,99993) nodes(i),dof3-1
                            end do
                            rewind (unit=12)
                            do i = 1, nant
                                read (12,99993)k, nf(3,k)
                                end do
                                close (unit=12, status='delete')
                                end if
                    end do
                    close (unit=12, status='delete')
            case (110)
                j = 0
                open (unit=12, file='temp.txt')
                backspace (unit=10)
                read (10,99998) key, opcode, n1, n2, n3, n4, n5, n6,
     &          iformt, maxout
                do
                    j=j+1
                    read (10,99989) key
                    if (key==-3) exit
                    backspace (unit=10)
                    read (10,99992) key, lcase, elem, dof, value1
                    write (12,99991) elem
                    end do
                    rewind (unit=12)
                    j = j-1
                    allocate (loadnode(j))
                    loadnode = 0
                    do i=1, j
                        read (12,99991) loadnode(i)
                        end do
                    close (unit=12, status='delete')
                case default
                    write (*,*) 'Wrong data set of neutral file'
                    end select
                    end do
99999 format (1X,I4,A1,A30)
99998 format (1X,I4,A1,6A1,61X,I2,1X,I2)
99997 format (1X,I2,I10,3E14.7)
99996 format (1X,I2,1X,8A1,I10,I10,I10,I10,I10)
99995 format (1X,I2,10I10)
99994 format (1X,I2,I10,I5,I5,I5,5X,I5,I5)
99993 format (1X,4I10)
99992 format (1X,I2,I5,I10,I5,E15.5)
```

```
99991 format (1X,I10)
99990 format (1X,I4)
99989 format (1X,I2)
99988 format (3ES14.7)
99987 format (10I10)
99986 format (I5)
99985 format (1X,I2,1X,A8,I5,1X,I5,1X,3I1)
!--------------------translation of element type---------------------
      if (nod==3) then
        etype = 'triangle'
        else if (nod==4) then
            etype = 'quadrilateral'
            else if (nod==8) then
                etype = 'hexahedron'
                else
                    write (*,*) 'Cannot translate element type'
                    end if
!--------------------the input data file of HITSI--------------------
      write (11,*) etype, nels, nn
      if (etype == 'triangle') then
        do i=1,nn
            write(11,*) i, g_coord(1,i), g_coord(2,i)
            end do
            else if (etype == 'quadrilateral') then
                do i=1,nn
                    write(11,*) i, g_coord(1,i), g_coord(2,i)
                    end do
                    else if (etype == 'hexahedron') then
                        do i=1,nn
                            write(11,*) i, g_coord(:,i)
                            end do
                            else
                write (*,*) 'Error occurs on rewrite of coordinate'
                            end if
      allocate (g_num1(nod,nels))
      g_num1 = 0
      if (etype == 'triangle') then
        do i=1,nels
            if (mod(i,2)==0) then
                g_num1(1,i) = g_num(2,i)
                g_num1(2,i) = g_num(3,i)
                g_num1(3,i) = g_num(1,i)
                else
                    g_num1(:,i) = g_num(:,i)
                    end if
            write(11,*) i, g_num1(:,i)
            end do
            else if (etype == 'quadrilateral') then
                do i=1,nels
                g_num1(1,i) = g_num(1,i)
                g_num1(2,i) = g_num(4,i)
                g_num1(3,i) = g_num(3,i)
                g_num1(4,i) = g_num(2,i)
                    write(11,*) i, g_num1(:,i)
                    end do
                    else if (etype == 'hexahedron') then
                        do i=1,nels
                            write(11,*) i, g_num(:,i)
                            end do
                            else
            write(*,*)'Error occurs on rewrite of element definition'
```

```
                        end if
            if (nod == 3) then
            do i = 1,nn
                write(11,*) i, nf(1,i), nf(2,i)
                end do
                else if (nod == 4) then
                    do i = 1,nn
                        write(11,*) i, nf(1,i), nf(2,i)
                        end do
                        else if (nod == 8) then
                            do i = 1,nn
                                write(11,*) i, nf(:,i)
                                end do
                                end if
            write (11,*)j
            do i =1,j
                write(11,*) loadnode(i)
                end do
!----------------------post-processing transfer----------------------
        case ('post-processing')
        write (*,*) 'Please Enter The model Name: '
        read (*,*) modelname
        write (*,*) 'Please Enter The Input File Name: '
        read (*,*) filename1
        write (*,*) 'Please Enter The Output File Name: '
        read (*,*) filename2
        open (unit=10, file=filename1, status='old', action='read',
     &  iostat=ierror)
        open (unit=11, file=filename2, status='replace',
     &  action='write', iostat=ierror)
        do
            read (10,99969) key
99969 format(1X,I4)
            if (key==9999) exit
            select case (key)
                case (1)
                    read (10,*)ndim, nn, nod, nels, element, nst,
     &              nip, time,e,v,mid
                case (2)
                    allocate (g_coord(ndim,nn))
                    do i=1,nn
                        read (10,*)node, g_coord(:,i)
                        end do
                case (3)
                    allocate (g_num(nod,nels), material1(nels))
                    do i=1,nels
                        read (10,*)elemt, g_num(:,i),material1(i)
                        end do
                case (4)
                    allocate (g_dispmt(ndim,nn))
                    do i=1,nn
                        read (10,*)node, g_dispmt(:,i)
                        end do
                case (5)
                    allocate (g_reforce(ndim,nn))
                    do i=1,nn
                        read (10,*)node, g_reforce(:,i)
                        end do
                case (6)
                    allocate (g_points(ndim,nip,nels))
                    do i=1,nels
```

```
                        read (10,*) node
                        do k=1,nip
                        read (10,*)point,g_points(:,k,i)
                        end do
                        end do
                    case (7)
                        allocate (g_sigma(nst,nip,nels))
                        do i=1,nels
                            read (10,*) node
                            do k=1,nip
                            read (10,*)point,g_sigma(:,k,i)
                            end do
                            end do
                    case (8)
                        allocate (g_eps(nst,nip,nels))
                        do i=1,nels
                            read (10,*) node
                            do k=1,nip
                            read (10,*)point,g_eps(:,k,i)
                            end do
                            end do
                    case (9)
                        allocate (g_evp(nst,nip,nels))
                        do i=1,nels
                            read (10,*) node
                            do k=1,nip
                            read (10,*)point,g_evp(:,k,i)
                            end do
                            end do
                    case (10)
                        allocate (g_damage(nip,nels))
                        do i=1,nels
                            read (10,*) node
                            do k=1,nip
                            read (10,*)point,g_damage(k,i)
                            end do
                            end do
                            case default
                                write (*,*) 'Wrong load case'
                                end select
                                end do
!----------------------the input data of FEMGV-----------------------
        if (element=='triangle') then
            types=7
            else if (element=='quadrilateral') then
                types=9
                else if (element=='hexahedron') then
                    types=1
                    else
                        write(*,*)'Wrong element type'
                        end if
        write (11,99968) key1,opcode1,modelname
        write (11,99967) key2,opcode1,iformt1
        do i=1,nn
        write (11,99966) key6,i,g_coord(:,i)
        end do
        write (11,99965) key8
        write (11,99967) key3,opcode1,iformt1
        do i=1,nels
            write (11,99964) key6,i,types,group1,material1(i)
            write (11,99963) key7,g_num(:,i)
```

```
      end do
   write (11,99965) key8
   write (11,99967) key4,opcode1
   do i=1,mid
       write (11,99962) key6,group1,nsc,nlc,i,e,v
       end do
   write (11,99965) key8
   write (11,99961) key5,opcode1,'LC01  ',time
   write (11,99960) key9,'DISPL   ',ncomps1,irtype,norcty
   write (11,99959) key10,'U1      ',menu,ictype,icind1,icind2,
&  iexist
   write (11,99959) key10,'U2      ',menu,ictype,ictype,icind2,
&  iexist
   do i=1,nn
       write (11,99958) key6,i,g_dispmt(:,i)
       end do
   write (11,99965) key8
   write (11,99961) key5,opcode1,'LC02  ',time
   write (11,99960) key9,'R-FORCE ',ncomps1,irtype,norcty
   write (11,99959) key10,'R1      ',menu,ictype,icind1,icind2,
&  iexist
   write (11,99959) key10,'R2      ',menu,ictype,ictype,icind2,
&  iexist
   do i=1,nn
       write (11,99958) key6,i,g_reforce(:,i)
       end do
   write (11,99965) key8
   write (11,99956) key5,opcode1,'LC03  ',time,
&  'STATIC,DIRECT       ',irtype,irtype,nsc
   write (11,99960) key9,'STRESS  ',ncomps2,ncomps2,norcty
   write (11,99959) key10,'S11     ',menu,ncomps2,icind1,icind2,
&  iexist
   write (11,99959) key10,'S22     ',menu,ncomps2,ncomps1,
&  ncomps1,iexist
   write (11,99959) key10,'S12     ',menu,ncomps2,icind1,
&  ncomps1,iexist
   write (11,99959) key10,'S33      ',menu,ncomps2,icind3,
&  icind3,iexist
   do i=1,nels
       write(11,99955)key6,i,types,icind1,icind2,iexist,icind1,
&      nip,nsc
       do k=1,nip
           write (11,99958) key7,k,g_points(:,k,i)
           write (11,99958) key7,k,g_sigma(:,k,i)
           end do
           end do
   write (11,99965) key8
   write (11,99956) key5,opcode1,'LC04  ',time,
&  'STATIC,DIRECT       ',irtype,irtype,nsc
   write (11,99960) key9,'STRAIN  ',ncomps2,ncomps2,norcty
   write (11,99959) key10,'SXX     ',menu,ncomps2,icind1,icind2,
&  iexist
   write (11,99959) key10,'SYY     ',menu,ncomps2,ncomps1,
&  ncomps1,iexist
   write (11,99959) key10,'SXY     ',menu,ncomps2,icind1,
&  ncomps1,iexist
   write (11,99959) key10,'SZZ      ',menu,ncomps2,icind3,
&  icind3,iexist
   do i=1,nels
       write(11,99955)key6,i,types,icind1,icind2,iexist,icind1,
&      nip,nsc
```

```
            do k=1,nip
                write (11,99958) key7,k,g_points(:,k,i)
                write (11,99958) key7,k,g_eps(:,k,i)
                end do
                end do
         write (11,99965) key8
         write (11,99956) key5,opcode1,'LC05  ',time,
     &  'STATIC,DIRECT       ',irtype,irtype,nsc
         write (11,99960) key9,'CSTRAIN ',ncomps2,ncomps2,norcty
         write (11,99959) key10,'CS11    ',menu,ncomps2,icind1,icind2,
     &  iexist
         write (11,99959) key10,'CS22    ',menu,ncomps2,ncomps1,
     &  ncomps1,iexist
         write (11,99959) key10,'CS12    ',menu,ncomps2,icind1,
     &  ncomps1,iexist
         write (11,99959) key10,'CS33    ',menu,ncomps2,icind3,
     &  icind3,iexist
         do i=1,nels
            write(11,99955)key6,i,types,icind1,icind2,iexist,icind1,
     &      nip,nsc
            do k=1,nip
                write (11,99958) key7,k,g_points(:,k,i)
                write (11,99958) key7,k,g_evp(:,k,i)
                end do
                end do
         write (11,99965) key8
         write (11,99956) key5,opcode1,'LC06  ',time,
     &  'STATIC,DIRECT       ',irtype,irtype,nsc
         write (11,99960) key9,'DAMAGE  ',ncomps2,ncomps2,norcty
         write (11,99959) key10,'S11     ',menu,ncomps2,icind1,icind2,
     &  iexist
         write (11,99959) key10,'S22     ',menu,ncomps2,ncomps1,
     &  ncomps1,iexist
         write (11,99959) key10,'S12     ',menu,ncomps2,icind1,
     &  ncomps1,iexist
         write (11,99959) key10,'S33      ',menu,ncomps2,icind3,
     &  icind3,iexist
         do i=1,nels
            write(11,99955)key6,i,types,icind1,icind2,iexist,icind1,
     &      nip,nsc
            do k=1,nip
                write (11,99958) key7,k,g_points(:,k,i)
                write (11,99958) key7,k,g_damage(k,i)
                end do
                end do
         write (11,99965) key8
         write (11,99957) key11
99968 format(1X,I4,A1,A30)
99967 format(1X,I4,A1,67X,I2)
99966 format(1X,I2,I10,3ES14.7)
99965 format(1X,I2)
99964 format(1X,I2,I10,I5,I5,I5)
99963 format(1X,I2,10I10)
99962 format(1X,I2,I2,I2,I2,I5,ES12.5,ES12.5)
99961 format(1X,I4,A1,A6,ES12.5)
99960 format(1X,I2,2X,A8,I5,I5,I5)
99959 format(1X,I2,2X,A8,I5,I5,I5,I5,I5)
99958 format(1X,I2,I5,6ES12.5)
99957 format(1X,I4)
99956 format(1X,I4,A1,A6,ES12.5,12X,A20,I2,I5,10X,I2)
99955 format(1X,I2,I5,I5,I5,I5,I5,I5,I5,I5)
```

```
case default
    write (*,*) 'Wrong transfer processing'
    end select
end program DTI
```

10.3 Source Code of Validation Programmes

10.3.1 Stress Transformation Subroutine

```
      program TRScheck
        use validation
        implicit none
        character(len=16) :: filename1, filename2
        doubleprecision, allocatable :: sigma(:), stress(:)
        integer :: nos, i, ierror
        write(*,*) 'Please Enter The Input File Name: '
        read (*,*) filename1
        write(*,*) 'Please Enter The Output File Name: '
        read (*,*) filename2
        open(10,file=filename1,status='old',action='read',iostat=ierror)
        open(11,file=filename2,status='replace',action='write',
     &       iostat=ierror)
        read (10,*) nos
        allocate (sigma(nos), stress(nos+4))
        read (10,*) (sigma(i), i=1,nos)
        call TRS(sigma,stress)
        if (nos==4) then
            write (11,99999) 'Deviator Stress Tensor:', 'XX','YY'
            write (11,99998) stress(1), stress(2)
            write (11,99999) 'Deviator Stress Tensor:', 'XY','ZZ'
            write (11,99998) stress(3), stress(4)
            write (11,99997) 'Principal Stress:', '1st', '2nd', '3rd'
            write (11,99996) stress(5), stress(6), stress(7)
            write (11,99995) 'Equivalent Stress:', stress(8)
            else if (nos==6) then
                write (11,99994) 'Deviator Stress Tensor:', 'XX', 'YY',
     &                           'ZZ'
                write (11,99998) stress(1), stress(2), stress(3)
                write (11,99994) 'Deviator Stress Tensor:', 'XY', 'YZ',
     &                           'ZX'
                write (11,99998) stress(4), stress(5), stress(6)
                write (11,99997) 'Principal Stress:','1st', '2nd', '3rd'
                write (11,99996) stress(7), stress(8), stress(9)
                write (11,99995) 'Equivalent Stress:', stress(10)
                end if
99999  format(a23,6x,a2,12x,a2)
99998  format(23x,3es14.7)
99997  format(a17,6x,a3,12x,a3,12x,a3)
99996  format(17x,3es14.7)
99995  format(a18,1x,es14.7)
99994  format(a23,6x,a2,12x,a2,12x,a2)
                end program TRScheck
```

10.3.2 Numerical Method Subroutines

Main programme

```
      program NMScheck
        use validation
        implicit none
        character(len=16) :: filename1, filename2
        integer :: nos, nom, noe, num, rcv, i, n, ierror
        doubleprecision, allocatable :: y(:), stress(:), mat(:)
        doubleprecision :: t, dt
```

```
   write(*,*) 'Please Enter The Input File Name: '
   read (*,*) filename1
   write(*,*) 'Please Enter The Output File Name: '
   read (*,*) filename2
   open(10,file=filename1,status='old',action='read',iostat=ierror)
   open(11,file=filename2,status='replace',action='write',
&       iostat=ierror)
   nos = 1; nom = 1; noe = 1
   read(10,*) num, n
   allocate (y(noe), stress(nos), mat(nom))
   read(10,*) y
   read(10,*) dt
   t = 0.
   write(11,'(1x,a4,5x,a1,5x,a10)')'time', 'y', 'acceptance'
   do i = 1, n
   select case (num)
   case (1)
       call EULER (ntest,y,t,dt,stress,mat,nos,nom,noe)
   case (2)
       call RK4 (ntest,y,t,dt,stress,mat,nos,nom,noe)
   case (3)
       call RKM (ntest,y,t,dt,stress,mat,nos,nom,noe,rcv)
   case (4)
       call RKF (ntest,y,t,dt,stress,mat,nos,nom,noe,rcv)
       end select
       t = t+dt
       write(11,'(f5.2,f10.7,1x,I5)')t, y, rcv
       end do
 end program NMScheck
```

Test formulation

```
   subroutine ntest (f,x,t,stress,mat,nos,nom,noe)
     implicit none
     integer, intent(in) :: nos, nom, noe
     doubleprecision, intent(in) :: stress(nos), mat(nom),
&                                   x(noe)
     doubleprecision, intent(in) :: t
     doubleprecision, intent(out) ::  f(noe)
     f = x-t**2+1
     return
     end subroutine ntest
```

10.3.3 Constitutive Equations Subroutines

```
 program CEScheck
   use validation
   implicit none
   character(len=16) :: filename1, filename2
   integer :: nos, nom, noe, i, num, loop, j, ops, ierror
   doubleprecision, allocatable ::sigma(:),stress(:),mat(:),y(:)
   doubleprecision :: t,dt
   open(10,file=filename1,status='old',action='read',iostat=ierror)
   open(11,file=filename2,status='replace',action='write',
&       iostat=ierror)
   open(12,file='timedata',status='replace',action='write',
&       iostat=ierror)
   open(13,file='straindata',status='replace',action='write',
&       iostat=ierror)
   read (10,*) num, nom, noe, nos, dt, loop, ops
```

```
   select case (nos)
   case (1)
       allocate(sigma(nos), stress(nos), mat(nom), y(noe))
       case (4)
           allocate(sigma(nos), stress(nos+4), mat(nom), y(noe))
           case (6)
               allocate(sigma(nos), stress(nos+4), mat(nom),y(noe))
               end select
   read(10,*) mat
   read(10,*) sigma
   t = 0.
   y = 0.
   select case (nos)
   case(1)
       stress = sigma
       case(4)
           call TRS(sigma,stress)
           case(6)
               call TRS(sigma,stress)
               end select
   select case (num)
   case (1)
   write (11,*) 'time','strain in (x)/(x,y,xy,z)/(x,y,z,xy,yz,zx)',
&               'damage'
   do i = 1, loop
       do j = 1, 100
           t = t+dt
           call RK4(KR,y,t,dt,stress,mat,nos,nom,noe)
           end do
       write(11,'(f9.2,7es12.5)') t,y
       write(12,*) t
       write(13,*) y(ops)
       end do
   case (2)
   write (11,*) 'time','strain in (x)/(x,y,xy,z)/(x,y,z,xy,yz,zx)',
&               'hardening','coarsening','damage'
   do i = 1, loop
       do j = 1, 100
       t = t+dt
       call RK4(PH,y,t,dt,stress,mat,nos,nom,noe)
       end do
       write(11,'(f9.2,10es12.5)') t,y
       write(12,*) t
       write(13,*) y(ops)
       end do
   case (3)
   write (11,*) 'time','strain in (x)/(x,y,xy,z)/(x,y,z,xy,yz,zx)',
&               'hardening','coarsening','damage1','damage2'
   do i = 1, loop
       do j = 1, 100
       t = t+dt
       call RK4(QX,y,t,dt,stress,mat,nos,nom,noe)
       end do
       write(11,'(f9.2,11es12.5)') t,y
       write(12,*) t
       write(13,*) y(ops)
       end do
       end select
 end program CEScheck
```

10.3.4 Time-step Control Subroutines

```
 program TSCcheck
   use validation
   implicit none
   character(len=16) :: filename1, filename2
   integer :: nos, nos1, nom, noe, i, num, loop, j, rcv, ierror
   doubleprecision, allocatable ::sigma(:),stress(:),mat(:),y(:)
&    ,y1(:)
   integer :: g_rcv(1,1,1)
   doubleprecision :: t,dt
   write(*,*) 'Please Enter The Input File Name: '
   read (*,*) filename1
   write(*,*) 'Please Enter The Output File Name: '
   read (*,*) filename2
   open(10,file=filename1,status='old',action='read',iostat=ierror)
   open(11,file=filename2,status='replace',action='write',
&       iostat=ierror)
   open(12,file='timedata.txt',status='replace',action='write',
&       iostat=ierror)
   open(13,file='straindata.txt',status='replace',action='write',
&       iostat=ierror)
   write(11,*) 'time','creep strain','damage','control value',
&   'time-step'
   read (10,*) num, nom, noe, nos, dt, loop
   nos1 = nos+4
   allocate(sigma(nos), stress(nos1), mat(nom), y(noe), y1(noe))
   read(10,*) mat
   read(10,*) sigma
   t = 0.
   y1 = 0.
   select case (nos)
   case(1)
       stress = sigma
       case(4)
           call TRS(sigma,stress)
           case(6)
               call TRS(sigma,stress)
               end select
   select case (num)
   case (1)
   do i = 1, loop
       do j = 1, 100
           y=y1
       t = t+dt
       call RKM(PH,y,t,dt,stress,mat,nos1,nom,noe,rcv)
       g_rcv = rcv
       if (maxval(g_rcv)==0) then
           y1 = y
           else
               y1 = y1
               end if
       call TSC(dt,g_rcv)
       end do
       write(11,'(f9.2,2es14.7,i5,es14.7)') t, y(1), y(8), g_rcv,dt
       write(12,*) t
       write(13,*) y(1)
       end do
   case (2)
       do i = 1, loop
       do j = 1, 100
```

```
            y=y1
        t = t+dt
        call RKF(PH,y,t,dt,stress,mat,nos1,nom,noe,rcv)
        g_rcv = rcv
        if (maxval(g_rcv)==0) then
            y1 = y
            else
                y1 = y1
                end if
        call TSS(dt,g_rcv)
        end do
        write(11,'(f9.2,2es14.7,i5,es14.7)') t, y(1), y(8), g_rcv,dt
        write(12,*) t
        write(13,*) y(1)
        end do
        case default
            write(*,*)'Wrong case in this testing'
        end select
end program TSCcheck
```

10.3.5 Normalization subroutine

```
program NORcheck
  use validation
  implicit none
  character(len=16) :: filename1, filename2
  integer :: nos, nos1, nom, noe, i, loop, j, ops, ierror
  doubleprecision, allocatable ::sigma(:),stress(:),mat(:),y(:)
  doubleprecision :: t, dt, ip
  write(*,*) 'Please Enter The Input File Name: '
  read (*,*) filename1
  write(*,*) 'Please Enter The Output File Name: '
  read (*,*) filename2
  open(10,file=filename1,status='old',action='read',iostat=ierror)
  open(11,file=filename2,status='replace',action='write',
&       iostat=ierror)
  open(12,file='timedata.txt',status='replace',action='write',
&       iostat=ierror)
  open(13,file='straindata.txt',status='replace',action='write',
&       iostat=ierror)
  read (10,*) nom, noe, nos, dt, loop, ops, ip
  nos1 = nos+5
  allocate(sigma(nos), stress(nos1), mat(nom), y(noe))
  read(10,*) mat
  read(10,*) sigma
  t = 0.
  y = 0.
  stress = 0
  call TRS(sigma,stress)
  select case (nos1)
  case (9)
      stress(9) = ip
      case (11)
          stress(11) = ip
          case default
              write(*,*) 'Wrong size for stress'
              end select
  write (11,*) 'time','strain in (x)/(x,y,xy,z)/(x,y,z,xy,yz,zx)',
&              'damage'
  do i = 1, loop
      do j = 1, 100
```

```
            t = t+dt
            call RK4(NOR_KR,y,t,dt,stress,mat,nos1,nom,noe)
            end do
        write(11,'(f9.5,7es12.5)') t,y
        write(12,*) t
        write(13,*) y(ops)
        end do
end program NORcheck
```

10.4 Input file of validation cases

10.4.1 Stress Transformation Subroutine

TRS1.dat

```
number of stress tensor component
4
component in x-direction; y-direction; xy-direction; z-direction
80 -30 -32 0
```

TRS2.dat

```
number of stress tensor component
6
component in x-direction; y-direction; z-direction;
             xy-direction; yz-direction; zx-direction
-10 0 7 9 0 5
```

10.4.2 Numerical Method Subroutines

EULER.dat

```
type of numerical method; number of iteration
1, 10
initial value of y
0.5
Time interval
0.2
```

RK4.dat

```
type of numerical method; number of iteration
2, 10
initial value of y
0.5
Time interval
0.2
```

RKM.dat

```
type of numerical method; number of iteration
3, 10
initial value of y
0.5
Time interval
0.2
```

RKF.dat

```
type of numerical method; number of iteration
4, 10
initial value of y
0.5
Time interval
0.2
```

10.4.3 Constitutive Equations Subroutines

KR1.dat

```
type of equations; number of material properties; number of
equations
1, 7, 2
number of stress terms; time interval; number of iteration;
output strain
1, 0.1, 101, 1
material property
1.092e-20, 8.462, -4.754e-4, 3.537e-17, 7.346, 6.789, 0.215
Stress tensor component
70
```

PH1.dat

```
type of equations; number of material properties; number of
equations
2, 7, 4
number of stress terms; time interval; number of iteration;
output strain
1, 0.1, 100, 1
material property
6.216e-8, 0.15, 2.0, 1.0e4, 0.35, 4.998e-4, 1.32
Stress tensor component
70
```

KR2.dat

```
type of equations; number of material properties; number of
equations
1, 7, 5
number of stress terms; time interval; number of iteration;
output strain
4, 0.1, 101, 1
material property
1.092e-20, 8.462, -4.754e-4, 3.537e-17, 7.346, 6.789, 0.215
Stress tensor component
70, 0, 0, 0
```

KR3.dat

type of equations; number of material properties; number of equations
1, 7, 5
number of stress terms; time interval; number of iteration; output strain
4, 0.1, 101, 1
material property
1.092e-20, 8.462, -4.754e-4, 3.537e-17, 7.346, 6.789, 0.215
Stress tensor component
59.74873734, 10.25126266, 24.74873734, 0

KR4.dat

type of equations; number of material properties; number of equations
1, 7, 5
number of stress terms; time interval; number of iteration; output strain
4, 0.1, 101, 1
material property
1.092e-20, 8.462, -4.754e-4, 3.537e-17, 7.346, 6.789, 0.215
Stress tensor component
35, 35, 35, 0

PH2.dat

type of equations; number of material properties; number of equations
2, 7, 8
number of stress terms; time interval; number of iteration; output strain
4, 0.1, 100, 1
material property
6.216e-8, 0.15, 2.0, 1.0e4, 0.35, 4.998e-4, 1.32
Stress tensor component
70, 0, 0, 0

PH3.dat

type of equations; number of material properties; number of equations
2, 7, 8
number of stress terms; time interval; number of iteration; output strain
4, 0.1, 100, 1
material property
6.216e-8, 0.15, 2.0, 1.0e4, 0.35, 4.998e-4, 1.32
Stress tensor component
59.74873734, 10.25126266, 24.74873734, 0

PH4.dat

type of equations; number of material properties; number of equations
2, 7, 8
number of stress terms; time interval; number of iteration; output strain
4, 0.1, 100, 1
material property
6.216e-8, 0.15, 2.0, 1.0e4, 0.35, 4.998e-4, 1.32
Stress tensor component
35, 35, 35, 0

QX1.dat

type of equations; number of material properties; number of equations
3, 11, 9
number of stress terms; time interval; number of iteration; output strain
4, 0.1, 370, 1
material property
2.1618e-9, 0.20524, 1.8537, 2.4326e5, 0.5929, 9.2273e-5, 2.8, 2 2, 2.5, 1,
Stress tensor component
60, 0, 0, 0

QX2.dat

type of equations; number of material properties; number of equations
3, 11, 9
number of stress terms; time interval; number of iteration; output strain
4, 0.1, 370, 1
material property
2.1618e-9, 0.20524, 1.8537, 2.4326e5, 0.5929, 9.2273e-5, 2.8, 2 2, 2.5, 1
Stress tensor component
51.21320344, 8.78679656, 21.21320344, 0

QX3.dat

type of equations; number of material properties; number of equations
3, 11, 9
number of stress terms; time interval; number of iteration; output strain
4, 0.1, 370, 1
material property
2.1618e-9, 0.20524, 1.8537, 2.4326e5, 0.5929, 9.2273e-5, 2.8, 2, 2, 2.5, 1
Stress tensor component
30, 30, 30, 0

KR5.dat

```
type of equations; number of material properties; number of equations
1, 7, 7
number of stress terms; time interval; number of iteration; output strain
6, 0.1, 101, 1
material property
1.092e-20, 8.462, -4.754e-4, 3.537e-17, 7.346, 6.789, 0.215
Stress tensor component
70, 0, 0, 0, 0, 0
```

KR6.dat

```
type of equations; number of material properties; number of equations
1, 7, 7
number of stress terms; time interval; number of iteration; output strain
6, 0.1, 101, 2
material property
1.092e-20, 8.462, -4.754e-4, 3.537e-17, 7.346, 6.789, 0.215
Stress tensor component
0, 70, 0, 0, 0, 0
```

KR7.dat

```
type of equations; number of material properties; number of equations
1, 7, 7
number of stress terms; time interval; number of iteration; output strain
6, 0.1, 101, 3
material property
1.092e-20, 8.462, -4.754e-4, 3.537e-17, 7.346, 6.789, 0.215
Stress tensor component
0, 0, 70, 0, 0, 0
```

PH5.dat

```
type of equations; number of material properties; number of equations
2, 7, 10
number of stress terms; time interval; number of iteration; output strain
6, 0.1, 100, 1
material property
6.216e-8, 0.15, 2.0, 1.0e4, 0.35, 4.998e-4, 1.32
Stress tensor component
70, 0, 0, 0, 0, 0
```

PH6.dat

```
type of equations; number of material properties; number of
equations
2, 7, 10
number of stress terms; time interval; number of iteration;
output strain
6, 0.1, 100, 2
material property
6.216e-8, 0.15, 2.0, 1.0e4, 0.35, 4.998e-4, 1.32
Stress tensor component
0, 70, 0, 0, 0, 0
```

PH7.dat

```
type of equations; number of material properties; number of
equations
2, 7, 10
number of stress terms; time interval; number of iteration;
output strain
6, 0.1, 100, 3
material property
6.216e-8, 0.15, 2.0, 1.0e4, 0.35, 4.998e-4, 1.32
Stress tensor component
0, 0, 70, 0, 0, 0
```

QX4.dat

```
type of equations; number of material properties; number of
equations
3, 11, 11
number of stress terms; time interval; number of iteration;
output strain
6, 0.1, 370, 1
material property
2.1618e-9, 0.20524, 1.8537, 2.4326e5, 0.5929, 9.2273e-5, 2.8, 2 2,
2.5, 1,
Stress tensor component
60, 0, 0, 0, 0, 0
```

QX5.dat

```
type of equations; number of material properties; number of
equations
3, 11, 11
number of stress terms; time interval; number of iteration;
output strain
6, 0.1, 370, 2
material property
2.1618e-9, 0.20524, 1.8537, 2.4326e5, 0.5929, 9.2273e-5, 2.8, 2 2,
2.5, 1,
Stress tensor component
0, 60, 0, 0, 0
```

QX6.dat

```
type of equations; number of material properties; number of
equations
3, 11, 11
number of stress terms; time interval; number of iteration;
output strain
6, 0.1, 370, 3
material property
2.1618e-9, 0.20524, 1.8537, 2.4326e5, 0.5929, 9.2273e-5, 2.8, 2 2,
2.5, 1,
Stress tensor component
0, 0, 60, 0, 0, 0
```

10.4.4 Time-step Control Subroutines

TSC1.dat

```
type of numerical method; number of material properties; number
of equations
1, 7, 8
number of stress terms; time interval; number of iteration
4, 0.1, 135
material property
6.216e-8, 0.15, 2.0, 1.0e4, 0.35, 4.998e-4, 1.32
Stress tensor component
70, 0, 0, 0
```

TSC2.dat

```
type of numerical method; number of material properties; number
of equations
2, 7, 8
number of stress terms; time interval; number of iteration
4, 0.1, 100
material property
6.216e-8, 0.15, 2.0, 1.0e4, 0.35, 4.998e-4, 1.32
Stress tensor component
70, 0, 0, 0
```

10.4.5 Normalization Subroutine

NOR1.dat

```
number of material properties; number of equations
8, 5
number of stress terms; time interval; number of iteration;
output strain; stress used to normalize
4, 0.001, 1285, 1, 70
material property
1.092e-20, 8.462, -4.754e-4, 3.537e-17, 7.346, 6.789, 0.215,
170e3
Stress tensor component
1, 0, 0, 0
```

NOR2.dat

```
number of material properties; number of equations
8, 5
number of stress terms; time interval; number of iteration;
output strain; stress used to normalize
4, 0.001, 1285, 1, 70
material property
1.092e-20, 8.462, -4.754e-4, 3.537e-17, 7.346, 6.789, 0.215,
170e3
Stress tensor component
0.8535339, 0.1464466, 0.3535534, 0
```

NOR3.dat

```
number of material properties; number of equations
8, 5
number of stress terms; time interval; number of iteration;
output strain; stress used to normalize
4, 0.001, 1285, 1, 70
material property
1.092e-20, 8.462, -4.754e-4, 3.537e-17, 7.346, 6.789, 0.215,
170e3
Stress tensor component
0.5, 0.5, 0.5, 0
```

NOR4.dat

```
number of material properties; number of equations
8, 7
number of stress terms; time interval; number of iteration;
output strain; stress used to normalize
6, 0.001, 1285, 1, 70
material property
1.092e-20, 8.462, -4.754e-4, 3.537e-17, 7.346, 6.789, 0.215,
170e3
Stress tensor component
1, 0, 0, 0, 0, 0
```

NOR5.dat

```
number of material properties; number of equations
8, 7
number of stress terms; time interval; number of iteration;
output strain; stress used to normalize
6, 0.001, 1285, 2, 70
material property
1.092e-20, 8.462, -4.754e-4, 3.537e-17, 7.346, 6.789, 0.215,
170e3
Stress tensor component
0, 1, 0, 0, 0, 0
```

NOR6.dat

```
number of material properties; number of equations
8, 7
number of stress terms; time interval; number of iteration;
output strain; stress used to normalize
6, 0.001, 1285, 3, 70
material property
1.092e-20, 8.462, -4.754e-4, 3.537e-17, 7.346, 6.789, 0.215,
170e3
Stress tensor component
0, 0, 1, 0, 0, 0
```

10.4.6 Nodal Loads Calculator

NLC1.dat

```
case type
axisymmetric
the number of element; expected pressure
2, 70
the coordinate of first loaded node of each element
0, 20
the coordinate of second loaded node of each element
20, 40
```

NLC2.dat

```
case type
axisymmetric
the number of element; expected pressure
2, 70
the coordinate of first loaded node of each element
0, 20
the coordinate of second loaded node of each element
20, 40
```

10.4.7 Data Transfer Interface

DTI.anl

```
    1CCHAPTER6CASE1
    2CNODES
2
 -1         1 0.0000000E+00 0.0000000E+00 0.0000000E+00
 -1         2 1.0000000E+01 0.0000000E+00 0.0000000E+00
 -1         3 2.0000000E+01 0.0000000E+00 0.0000000E+00
 -1         4 0.0000000E+00 2.0000000E+01 0.0000000E+00
 -1         5 1.0000000E+01 2.0000004E+01 0.0000000E+00
 -1         6 2.0000000E+01 2.0000000E+01 0.0000000E+00
 -1         7 0.0000000E+00 4.0000000E+01 0.0000000E+00
 -1         8 1.0000000E+01 4.0000000E+01 0.0000000E+00
 -1         9 2.0000000E+01 4.0000000E+01 0.0000000E+00
 -1        10 0.0000000E+00 6.0000000E+01 0.0000000E+00
 -1        11 1.0000000E+01 6.0000000E+01 0.0000000E+00
```

```
 -1        12 2.0000000E+01 6.0000000E+01 0.0000000E+00
 -3
  101CPARTS
2 10
 -1 S1               1         3         1         6        12
 -2           1           2           3           4           5           6
7         8         9        10
 -2        11        12
 -3
    3CELEM
2 10
 -1         1    5    1    0         1    0
 -2         1         2         5         4
 -1         2    5    1    0         1    0
 -2         2         3         6         5
 -1         3    5    1    0         1    0
 -2         4         5         8         7
 -1         4    5    1    0         1    0
 -2         5         6         9         8
 -1         5    5    1    0         1    0
 -2         7         8        11        10
 -1         6    5    1    0         1    0
 -2         8         9        12        11
 -3
  103CCONS
2 10
 -1 CO1          1     3 010000
 -2         1         2         3
 -1 CO2          2     4 100000
 -2         1         4         7        10
 -3
  110CFORCE
2
 -1    1        10    2    4.00000E+01
 -1    1        11    2    4.00000E+01
 -1    1        12    2    4.00000E+01
 -3
 9999
```

DTI.res

```
    1
           2                9                3                8 triangle
4                  3     4.0000000000000000             170000.00000000000
0.29999999999999999   1
    2
           1   0.0000000000000000        0.0000000000000000
           2   0.5000000000000000        0.0000000000000000
           3   1.0000000000000000        0.0000000000000000
           4   0.0000000000000000       -0.5000000000000000
           5   0.5000000000000000       -0.5000000000000000
           6   1.0000000000000000       -0.5000000000000000
           7   0.0000000000000000       -1.0000000000000000
           8   0.5000000000000000       -1.0000000000000000
           9   1.0000000000000000       -1.0000000000000000
    3
           1           1           2           4           1
           2           5           4           2           1
           3           2           3           5           1
           4           6           5           3           1
```

```
        5           4           5           7           1
        6           8           7           5           1
        7           5           6           8           1
        8           9           8           6           1
4
        1   0.0000000000000000        2.3529559699592964E-004
        2  -3.5294510140535674E-005   2.3529547191359065E-004
        3  -7.0588929020547932E-005   2.3529529971266598E-004
        4   0.0000000000000000        1.1764846739377730E-004
        5  -3.5294938801932329E-005   1.1764818115249679E-004
        6  -7.0589601799141090E-005   1.1764817023585826E-004
        7   0.0000000000000000        0.0000000000000000
        8  -3.5295200112439891E-005   0.0000000000000000
        9  -7.0590198180171275E-005   0.0000000000000000
5
        1   0.0000000000000000        14.026700958263500
        2   2.2449187546680349E-005   28.053553235984012
        3  -3.3004068692148731        14.026720532648463
        4   0.0000000000000000        7.0721646593341347E-004
        5  -1.0795115519224296E-004   8.6553111922960113E-004
        6  -6.6010202122043777        5.9565404199979355E-004
        7   0.0000000000000000        0.0000000000000000
        8  -5.8657859545174773E-005   0.0000000000000000
        9  -3.3006308926241514        0.0000000000000000
6
        1
        1   0.0000000000000000      -0.25000000000000000
        2  0.25000000000000000        0.0000000000000000
        3  0.25000000000000000      -0.25000000000000000
        2
        1  0.50000000000000000      -0.25000000000000000
        2  0.25000000000000000      -0.50000000000000000
        3  0.25000000000000000      -0.25000000000000000
        3
        1  0.50000000000000000      -0.25000000000000000
        2  0.75000000000000000        0.0000000000000000
        3  0.75000000000000000      -0.25000000000000000
        4
        1   1.0000000000000000      -0.25000000000000000
        2  0.75000000000000000      -0.50000000000000000
        3  0.75000000000000000      -0.25000000000000000
        5
        1   0.0000000000000000      -0.75000000000000000
        2  0.25000000000000000      -0.50000000000000000
        3  0.25000000000000000      -0.75000000000000000
        6
        1  0.50000000000000000      -0.75000000000000000
        2  0.25000000000000000       -1.0000000000000000
        3  0.25000000000000000      -0.75000000000000000
        7
        1  0.50000000000000000      -0.75000000000000000
        2  0.75000000000000000      -0.50000000000000000
        3  0.75000000000000000      -0.75000000000000000
        8
        1   1.0000000000000000      -0.75000000000000000
        2  0.75000000000000000       -1.0000000000000000
        3  0.75000000000000000      -0.75000000000000000
7
        1
        1  -1.3871248636299072E-004   39.999982450985883
           -1.6356921252612814E-005   0.0000000000000000
```

```
          2  -1.3871248636299072E-004   39.999982450985883
             -1.6356921252612814E-005   0.0000000000000000
          3  -1.3871248636299072E-004   39.999982450985883
             -1.6356921252612814E-005   0.0000000000000000
          2
          1  -2.8080772024097200E-004   39.999994616455837
              1.8624169035410889E-005   0.0000000000000000
          2  -2.8080772024097200E-004   39.999994616455837
              1.8624169035410889E-005   0.0000000000000000
          3  -2.8080772024097200E-004   39.999994616455837
              1.8624169035410889E-005   0.0000000000000000
          3
          1  -8.6551178675620122E-005   40.000052893418300
              3.3537138642561455E-005   0.0000000000000000
          2  -8.6551178675620122E-005   40.000052893418300
              3.3537138642561455E-005   0.0000000000000000
          3  -8.6551178675620122E-005   40.000052893418300
              3.3537138642561455E-005   0.0000000000000000
          4
          1  -1.9583782042964515E-004   39.999965270768499
              8.6551178682847007E-005   0.0000000000000000
          2  -1.9583782042964515E-004   39.999965270768499
              8.6551178682847007E-005   0.0000000000000000
          3  -1.9583782042964515E-004   39.999965270768499
              8.6551178682847007E-005   0.0000000000000000
          5
          1  -1.4892141945921367E-004   40.000434237458435
             -3.7431552065636927E-005   0.0000000000000000
          2  -1.4892141945921367E-004   40.000434237458435
             -3.7431552065636927E-005   0.0000000000000000
          3  -1.4892141945921367E-004   40.000434237458435
             -3.7431552065636927E-005   0.0000000000000000
          6
          1  -2.7863810427497526E-004   40.000298000417629
              3.4171374065676948E-005   0.0000000000000000
          2  -2.7863810427497526E-004   40.000298000417629
              3.4171374065676948E-005   0.0000000000000000
          3  -2.7863810427497526E-004   40.000298000417629
              3.4171374065676948E-005   0.0000000000000000
          7
          1  -7.7957688247920487E-005   40.000358204542437
              3.2743813641517213E-005   0.0000000000000000
          2  -7.7957688247920487E-005   40.000358204542437
              3.2743813641517213E-005   0.0000000000000000
          3  -7.7957688247920487E-005   40.000358204542437
              3.2743813641517213E-005   0.0000000000000000
          8
          1  -2.0437249574634109E-004   40.000316568443083
              7.7988288562739295E-005   0.0000000000000000
          2  -2.0437249574634109E-004   40.000316568443083
              7.7988288562739295E-005   0.0000000000000000
          3  -2.0437249574634109E-004   40.000316568443083
              7.7988288562739295E-005   0.0000000000000000
8
          1
          1  -7.0589020281071349E-005   2.3529425920430468E-004
             -2.5016467798113715E-010   0.0000000000000000
          2  -7.0589020281071349E-005   2.3529425920430468E-004
             -2.5016467798113715E-010   0.0000000000000000
          3  -7.0589020281071349E-005   2.3529425920430468E-004
             -2.5016467798113715E-010   0.0000000000000000
```

```
 2
 1  -7.0589877603864659E-005   2.3529458152218770E-004
     2.8484023230628420E-010   0.0000000000000000
 2  -7.0589877603864659E-005   2.3529458152218770E-004
     2.8484023230628420E-010   0.0000000000000000
 3  -7.0589877603864659E-005   2.3529458152218770E-004
     2.8484023230628420E-010   0.0000000000000000
 3
 1  -7.0588837760024515E-005   2.3529458152218770E-004
     5.1292094394505758E-010   0.0000000000000000
 2  -7.0588837760024515E-005   2.3529458152218770E-004
     5.1292094394505758E-010   0.0000000000000000
 3  -7.0588837760024515E-005   2.3529458152218770E-004
     5.1292094394505758E-010   0.0000000000000000
 4
 1  -7.0589325994417521E-005   2.3529425895361546E-004
     1.3237239092670720E-009   0.0000000000000000
 2  -7.0589325994417521E-005   2.3529425895361546E-004
     1.3237239092670720E-009   0.0000000000000000
 3  -7.0589325994417521E-005   2.3529425895361546E-004
     1.3237239092670720E-009   0.0000000000000000
 5
 1  -7.0589877603864659E-005   2.3529693478755459E-004
    -5.7248256100385884E-010   0.0000000000000000
 2  -7.0589877603864659E-005   2.3529693478755459E-004
    -5.7248256100385884E-010   0.0000000000000000
 3  -7.0589877603864659E-005   2.3529693478755459E-004
    -5.7248256100385884E-010   0.0000000000000000
 6
 1  -7.0590400224879781E-005   2.3529636230499359E-004
     5.2262101512211799E-010   0.0000000000000000
 2  -7.0590400224879781E-005   2.3529636230499359E-004
     5.2262101512211799E-010   0.0000000000000000
 3  -7.0590400224879781E-005   2.3529636230499359E-004
     5.2262101512211799E-010   0.0000000000000000
 7
 1  -7.0589325994417521E-005   2.3529636230499359E-004
     5.0078773804673390E-010   0.0000000000000000
 2  -7.0589325994417521E-005   2.3529636230499359E-004
     5.0078773804673390E-010   0.0000000000000000
 3  -7.0589325994417521E-005   2.3529636230499359E-004
     5.0078773804673390E-010   0.0000000000000000
 8
 1  -7.0589996135462770E-005   2.3529634047171651E-004
     1.1927620603713069E-009   0.0000000000000000
 2  -7.0589996135462770E-005   2.3529634047171651E-004
     1.1927620603713069E-009   0.0000000000000000
 3  -7.0589996135462770E-005   2.3529634047171651E-004
     1.1927620603713069E-009   0.0000000000000000
9
 1
 1  -1.1041841831367080E-005   2.2083568790380394E-005
    -2.7091404587803183E-011  -1.1041726959013309E-005
 2  -1.1041841831367080E-005   2.2083568790380394E-005
    -2.7091404587803183E-011  -1.1041726959013309E-005
 3  -1.1041841831367080E-005   2.2083568790380394E-005
    -2.7091404587803183E-011  -1.1041726959013309E-005
 2
 1  -1.1042001021190423E-005   2.2083769494679590E-005
     3.0846785077030556E-011  -1.1041768473489160E-005
 2  -1.1042001021190423E-005   2.2083769494679590E-005
```

```
          3.0846785077030556E-011  -1.1041768473489160E-005
        3 -1.1042001021190423E-005   2.2083769494679590E-005
          3.0846785077030556E-011  -1.1041768473489160E-005
        3
        1 -1.1041873740043404E-005   2.2083675803952752E-005
          5.5546613741522707E-011  -1.1041802063909341E-005
        2 -1.1041873740043404E-005   2.2083675803952752E-005
          5.5546613741522707E-011  -1.1041802063909341E-005
        3 -1.1041873740043404E-005   2.2083675803952752E-005
          5.5546613741522707E-011  -1.1041802063909341E-005
        4
        1 -1.1041879209793359E-005   2.2083596239717113E-005
          1.4335186984880532E-010  -1.1041717029923754E-005
        2 -1.1041879209793359E-005   2.2083596239717113E-005
          1.4335186984880532E-010  -1.1041717029923754E-005
        3 -1.1041879209793359E-005   2.2083596239717113E-005
          1.4335186984880532E-010  -1.1041717029923754E-005
        5
        1 -1.1042397112448653E-005   2.2084670893445599E-005
         -6.1998974660913957E-011  -1.1042273780996950E-005
        2 -1.1042397112448653E-005   2.2084670893445599E-005
         -6.1998974660913957E-011  -1.1042273780996950E-005
        3 -1.1042397112448653E-005   2.2084670893445599E-005
         -6.1998974660913957E-011  -1.1042273780996950E-005
        6
        1 -1.1042364726447949E-005   2.2084498696061860E-005
          5.6598706156802205E-011  -1.1042133969613915E-005
        2 -1.1042364726447949E-005   2.2084498696061860E-005
          5.6598706156802205E-011  -1.1042133969613915E-005
        3 -1.1042364726447949E-005   2.2084498696061860E-005
          5.6598706156802205E-011  -1.1042133969613915E-005
        7
        1 -1.1042233232891617E-005   2.2084401904589847E-005
          5.4234027004756503E-011  -1.1042168671698227E-005
        2 -1.1042233232891617E-005   2.2084401904589847E-005
          5.4234027004756503E-011  -1.1042168671698227E-005
        3 -1.1042233232891617E-005   2.2084401904589847E-005
          5.4234027004756503E-011  -1.1042168671698227E-005
        8
        1 -1.1042311588084041E-005   2.2084453923378754E-005
          1.2917330510133722E-010  -1.1042142335294716E-005
        2 -1.1042311588084041E-005   2.2084453923378754E-005
          1.2917330510133722E-010  -1.1042142335294716E-005
        3 -1.1042311588084041E-005   2.2084453923378754E-005
          1.2917330510133722E-010  -1.1042142335294716E-005
10
        1
        1   1.4898054512460445E-004
        2   1.4898054512460445E-004
        3   1.4898054512460445E-004
        2
        1   1.4898094184059548E-004
        2   1.4898094184059548E-004
        3   1.4898094184059548E-004
        3
        1   1.4898096363601544E-004
        2   1.4898096363601544E-004
        3   1.4898096363601544E-004
        4
        1   1.4898053839544465E-004
        2   1.4898053839544465E-004
```

```
            3   1.4898053839544465E-004
            5
            1   1.4898396952228488E-004
            2   1.4898396952228488E-004
            3   1.4898396952228488E-004
            6
            1   1.4898322201493304E-004
            2   1.4898322201493304E-004
            3   1.4898322201493304E-004
            7
            1   1.4898324453124583E-004
            2   1.4898324453124583E-004
            3   1.4898324453124583E-004
            8
            1   1.4898320239903099E-004
            2   1.4898320239903099E-004
            3   1.4898320239903099E-004
9999
```

10.5 Source code of performance exploration Programme

10.5.1 Performance of numerical method

```
 program NMSexp
   use validation
   implicit none
   character(len=16) :: filename1, filename2
   integer :: nos, nos1, nom, noe, i, num, loop, j, ierror, loop1,
&    rcv
   doubleprecision, allocatable ::sigma(:),stress(:),mat(:),y(:)
   doubleprecision :: t,dt,start,finish
   write(*,*) 'Please Enter The Input File Name: '
   read (*,*) filename1
   write(*,*) 'Please Enter The Output File Name: '
   read (*,*) filename2
   open(10,file=filename1,status='old',action='read',iostat=ierror)
   open(11,file=filename2,status='replace',action='write',
&       iostat=ierror)
   open(12,file='timedata.txt',status='replace',action='write',
&       iostat=ierror)
   open(13,file='straindata.txt',status='replace',action='write',
&       iostat=ierror)
   open(14,file='damagedata.txt',status='replace',action='write',
&       iostat=ierror)
         call cpu_time(start)
   read (10,*) num, nom, noe, nos, dt, loop, loop1
   nos1 = nos+4
   allocate(sigma(nos), stress(nos1), mat(nom), y(noe))
   read(10,*) mat
   read(10,*) sigma
   t = 0.
   y = 0.
   call TRS(sigma,stress)
   select case (num)
   case (1)
   write (11,*) 'time','strain in (x)/(x,y,xy,z)/(x,y,z,xy,yz,zx)',
&               'hardening','coarsening','damage'
   do i = 1, loop
       do j = 1, loop1
       t = t+dt
       call EULER(PH,y,t,dt,stress,mat,nos1,nom,noe)
       end do
       write(11,'(f9.2,10es12.5)') t,y
       write(12,*) t
       write(13,*) y(1)
       write(14,*) y(8)
       end do
   case (2)
   write (11,*) 'time','strain in (x)/(x,y,xy,z)/(x,y,z,xy,yz,zx)',
&               'hardening','coarsening','damage'
   do i = 1, loop
       do j = 1, 100
       t = t+dt
       call RK4(PH,y,t,dt,stress,mat,nos1,nom,noe)
       end do
       write(11,'(f9.2,10es12.5)') t,y
       write(12,*) t
       write(13,*) y(1)
       write(14,*) y(8)
```

```
            end do
         case (3)
         write (11,*) 'time','strain in (x)/(x,y,xy,z)/(x,y,z,xy,yz,zx)',
     &               'hardening','coarsening','damage'
         do i = 1, loop
             do j = 1, 100
             t = t+dt
             call RKM(PH,y,t,dt,stress,mat,nos1,nom,noe,rcv)
             end do
             write(11,'(f9.2,10es12.5)') t,y
             write(12,*) t
             write(13,*) y(1)
             write(14,*) y(8)
             end do
         case (4)
         write (11,*) 'time','strain in (x)/(x,y,xy,z)/(x,y,z,xy,yz,zx)',
     &               'hardening','coarsening','damage'
         do i = 1, loop
             do j = 1, 100
             t = t+dt
             call RKF(PH,y,t,dt,stress,mat,nos1,nom,noe,rcv)
             end do
             write(11,'(f9.2,10es12.5)') t,y
             write(12,*) t
             write(13,*) y(1)
             write(14,*) y(8)
             end do
             end select
             call cpu_time(finish)
             print '("Time = ",es14.7," seconds.")',finish-start
      end program NMSexp
```

10.5.2 Performance of time-step control

```
      program TSCexp
        use validation
        implicit none
        character(len=16) :: filename1, filename2
        integer :: nos, nos1, nom, noe, i, num, loop, j, rcv, ierror,
     &  loop1
        doubleprecision, allocatable ::sigma(:),stress(:),mat(:),y(:)
     &    ,y1(:)
        integer :: g_rcv(1,1,1)
        doubleprecision :: t,dt
        write(*,*) 'Please Enter The Input File Name: '
        read (*,*) filename1
        write(*,*) 'Please Enter The Output File Name: '
        read (*,*) filename2
open(10,file=filename1,status='old',action='read',iostat=ierror)
        open(11,file=filename2,status='replace',action='write',
     &       iostat=ierror)
        open(12,file='timedata.txt',status='replace',action='write',
     &       iostat=ierror)
        open(13,file='straindata.txt',status='replace',action='write',
     &       iostat=ierror)
        write(11,*) 'time','  strain','  control value','  time-step'
        read (10,*) num, nom, noe, nos, dt, loop, loop1
        nos1 = nos+4
        allocate(sigma(nos), stress(nos1), mat(nom), y(noe), y1(noe))
        read (10,*) y1
```

```
  read (10,*) mat
  read (10,*) sigma
  t = 0.
  select case (nos)
  case(1)
      stress = sigma
      case(4)
          call TRS(sigma,stress)
          case(6)
              call TRS(sigma,stress)
              end select
  select case (num)
  case (1)
  do i = 1, loop
      y=y1
      call RKF(ntest,y,t,dt,stress,mat,nos1,nom,noe,rcv)
      g_rcv = rcv
      call TSC(dt,g_rcv)
      if (maxval(g_rcv)==0) then
          y1 = y
          t = t+dt
          else
              y1 = y1
              t = t
              end if
      write(11,'(f9.2,es14.7,i5,es14.7)') t, y1, g_rcv,dt
      write(12,*) dt
      write(13,*) y(1)
      end do
  case (2)
      do i = 1, loop
      do j = 1, loop1
          y=y1
      t = t+dt
      call RKF(PH,y,t,dt,stress,mat,nos1,nom,noe,rcv)
      g_rcv = rcv
      if (maxval(g_rcv)==0) then
          y1 = y
          else
              y1 = y1
              end if
      call TSC(dt,g_rcv)
      end do
      write(11,'(f9.2,es14.7,i5,es14.7)') t, y(1), g_rcv,dt
      write(12,*) t
      write(13,*) y(1)
      end do
      case default
          write(*,*)'Wrong case in this testing'
      end select
end program TSCexp
```