

THE UNIVERSITY OF CALGARY

**A Dynamic Model of Skeletal Muscle**

by

Johnny Mattar

A THESIS

SUBMITTED TO THE FACULTY OF GRADUATE STUDIES  
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE  
DEGREE OF MASTER OF SCIENCE

DEPARTMENT OF MECHANICAL ENGINEERING

CALGARY, ALBERTA

DECEMBER, 1993

© Johnny Mattar 1993



National Library  
of Canada

Bibliothèque nationale  
du Canada

Acquisitions and  
Bibliographic Services Branch

Direction des acquisitions et  
des services bibliographiques

395 Wellington Street  
Ottawa, Ontario  
K1A 0N4

395, rue Wellington  
Ottawa (Ontario)  
K1A 0N4

*Your file* *Votre référence*

*Our file* *Notre référence*

**The author has granted an irrevocable non-exclusive licence allowing the National Library of Canada to reproduce, loan, distribute or sell copies of his/her thesis by any means and in any form or format, making this thesis available to interested persons.**

**L'auteur a accordé une licence irrévocable et non exclusive permettant à la Bibliothèque nationale du Canada de reproduire, prêter, distribuer ou vendre des copies de sa thèse de quelque manière et sous quelque forme que ce soit pour mettre des exemplaires de cette thèse à la disposition des personnes intéressées.**

**The author retains ownership of the copyright in his/her thesis. Neither the thesis nor substantial extracts from it may be printed or otherwise reproduced without his/her permission.**

**L'auteur conserve la propriété du droit d'auteur qui protège sa thèse. Ni la thèse ni des extraits substantiels de celle-ci ne doivent être imprimés ou autrement reproduits sans son autorisation.**

ISBN 0-315-93944-3

**Canada**

Name JOHNNY MATTAR

Dissertation Abstracts International is arranged by broad, general subject categories. Please select the one subject which most nearly describes the content of your dissertation. Enter the corresponding four-digit code in the spaces provided.

MECHANICAL ENGINEERING

SUBJECT TERM

0548

**U·M·I**

SUBJECT CODE

**Subject Categories**

**THE HUMANITIES AND SOCIAL SCIENCES**

**COMMUNICATIONS AND THE ARTS**

Architecture ..... 0729  
 Art History ..... 0377  
 Cinema ..... 0900  
 Dance ..... 0378  
 Fine Arts ..... 0357  
 Information Science ..... 0723  
 Journalism ..... 0391  
 Library Science ..... 0399  
 Mass Communications ..... 0708  
 Music ..... 0413  
 Speech Communication ..... 0459  
 Theater ..... 0465

**EDUCATION**

General ..... 0515  
 Administration ..... 0514  
 Adult and Continuing ..... 0516  
 Agricultural ..... 0517  
 Art ..... 0273  
 Bilingual and Multicultural ..... 0282  
 Business ..... 0688  
 Community College ..... 0275  
 Curriculum and Instruction ..... 0727  
 Early Childhood ..... 0518  
 Elementary ..... 0524  
 Finance ..... 0277  
 Guidance and Counseling ..... 0519  
 Health ..... 0680  
 Higher ..... 0745  
 History of ..... 0520  
 Home Economics ..... 0278  
 Industrial ..... 0521  
 Language and Literature ..... 0279  
 Mathematics ..... 0280  
 Music ..... 0522  
 Philosophy of ..... 0998  
 Physical ..... 0523

Psychology ..... 0525  
 Reading ..... 0535  
 Religious ..... 0527  
 Sciences ..... 0714  
 Secondary ..... 0533  
 Social Sciences ..... 0534  
 Sociology of ..... 0340  
 Special ..... 0529  
 Teacher Training ..... 0530  
 Technology ..... 0710  
 Tests and Measurements ..... 0288  
 Vocational ..... 0747

**LANGUAGE, LITERATURE AND LINGUISTICS**

Language  
 General ..... 0679  
 Ancient ..... 0289  
 Linguistics ..... 0290  
 Modern ..... 0291  
 Literature  
 General ..... 0401  
 Classical ..... 0294  
 Comparative ..... 0295  
 Medieval ..... 0297  
 Modern ..... 0298  
 African ..... 0316  
 American ..... 0591  
 Asian ..... 0305  
 Canadian (English) ..... 0352  
 Canadian (French) ..... 0355  
 English ..... 0593  
 Germanic ..... 0311  
 Latin American ..... 0312  
 Middle Eastern ..... 0315  
 Romance ..... 0313  
 Slavic and East European ..... 0314

**PHILOSOPHY, RELIGION AND THEOLOGY**

Philosophy ..... 0422  
 Religion  
 General ..... 0318  
 Biblical Studies ..... 0321  
 Clergy ..... 0319  
 History of ..... 0320  
 Philosophy of ..... 0322  
 Theology ..... 0469

**SOCIAL SCIENCES**

American Studies ..... 0323  
 Anthropology  
 Archaeology ..... 0324  
 Cultural ..... 0326  
 Physical ..... 0327  
 Business Administration  
 General ..... 0310  
 Accounting ..... 0272  
 Banking ..... 0770  
 Management ..... 0454  
 Marketing ..... 0338  
 Canadian Studies ..... 0385  
 Economics  
 General ..... 0501  
 Agricultural ..... 0503  
 Commerce-Business ..... 0505  
 Finance ..... 0508  
 History ..... 0509  
 Labor ..... 0510  
 Theory ..... 0511  
 Folklore ..... 0358  
 Geography ..... 0366  
 Gerontology ..... 0351  
 History  
 General ..... 0578

Ancient ..... 0579  
 Medieval ..... 0581  
 Modern ..... 0582  
 Black ..... 0328  
 African ..... 0331  
 Asia, Australia and Oceania ..... 0332  
 Canadian ..... 0334  
 European ..... 0335  
 Latin American ..... 0336  
 Middle Eastern ..... 0333  
 United States ..... 0337  
 History of Science ..... 0585  
 Law ..... 0398  
 Political Science  
 General ..... 0615  
 International Law and  
 Relations ..... 0616  
 Public Administration ..... 0617  
 Recreation ..... 0814  
 Social Work ..... 0452  
 Sociology  
 General ..... 0626  
 Criminology and Penology ..... 0627  
 Demography ..... 0938  
 Ethnic and Racial Studies ..... 0631  
 Individual and Family  
 Studies ..... 0628  
 Industrial and Labor  
 Relations ..... 0629  
 Public and Social Welfare ..... 0630  
 Social Structure and  
 Development ..... 0700  
 Theory and Methods ..... 0344  
 Transportation ..... 0709  
 Urban and Regional Planning ..... 0999  
 Women's Studies ..... 0453

**THE SCIENCES AND ENGINEERING**

**BIOLOGICAL SCIENCES**

Agriculture  
 General ..... 0473  
 Agronomy ..... 0285  
 Animal Culture and  
 Nutrition ..... 0475  
 Animal Pathology ..... 0476  
 Food Science and  
 Technology ..... 0359  
 Forestry and Wildlife ..... 0478  
 Plant Culture ..... 0479  
 Plant Pathology ..... 0480  
 Plant Physiology ..... 0817  
 Range Management ..... 0777  
 Wood Technology ..... 0746  
 Biology  
 General ..... 0306  
 Anatomy ..... 0287  
 Biostatistics ..... 0308  
 Botany ..... 0309  
 Cell ..... 0379  
 Ecology ..... 0329  
 Entomology ..... 0353  
 Genetics ..... 0369  
 Limnology ..... 0793  
 Microbiology ..... 0410  
 Molecular ..... 0307  
 Neuroscience ..... 0317  
 Oceanography ..... 0416  
 Physiology ..... 0433  
 Radiation ..... 0821  
 Veterinary Science ..... 0778  
 Zoology ..... 0472  
 Biophysics  
 General ..... 0786  
 Medical ..... 0760

Geodesy ..... 0370  
 Geology ..... 0372  
 Geophysics ..... 0373  
 Hydrology ..... 0388  
 Mineralogy ..... 0411  
 Paleobotany ..... 0345  
 Paleocology ..... 0426  
 Paleontology ..... 0418  
 Paleozoology ..... 0985  
 Palynology ..... 0427  
 Physical Geography ..... 0368  
 Physical Oceanography ..... 0415

**HEALTH AND ENVIRONMENTAL SCIENCES**

Environmental Sciences ..... 0768  
 Health Sciences  
 General ..... 0566  
 Audiology ..... 0300  
 Chemotherapy ..... 0992  
 Dentistry ..... 0567  
 Education ..... 0350  
 Hospital Management ..... 0769  
 Human Development ..... 0758  
 Immunology ..... 0982  
 Medicine and Surgery ..... 0564  
 Mental Health ..... 0347  
 Nursing ..... 0569  
 Nutrition ..... 0570  
 Obstetrics and Gynecology ..... 0380  
 Occupational Health and  
 Therapy ..... 0354  
 Ophthalmology ..... 0381  
 Pathology ..... 0571  
 Pharmacology ..... 0419  
 Pharmacy ..... 0572  
 Physical Therapy ..... 0382  
 Public Health ..... 0573  
 Radiology ..... 0574  
 Recreation ..... 0575

Speech Pathology ..... 0460  
 Toxicology ..... 0383  
 Home Economics ..... 0386

**PHYSICAL SCIENCES**

**Pure Sciences**  
 Chemistry  
 General ..... 0485  
 Agricultural ..... 0749  
 Analytical ..... 0486  
 Biochemistry ..... 0487  
 Inorganic ..... 0488  
 Nuclear ..... 0738  
 Organic ..... 0490  
 Pharmaceutical ..... 0491  
 Physical ..... 0494  
 Polymer ..... 0495  
 Radiation ..... 0754  
 Mathematics ..... 0405  
 Physics  
 General ..... 0605  
 Acoustics ..... 0986  
 Astronomy and  
 Astrophysics ..... 0606  
 Atmospheric Science ..... 0608  
 Atomic ..... 0748  
 Electronics and Electricity ..... 0607  
 Elementary Particles and  
 High Energy ..... 0798  
 Fluid and Plasma ..... 0759  
 Molecular ..... 0609  
 Nuclear ..... 0610  
 Optics ..... 0752  
 Radiation ..... 0756  
 Solid State ..... 0611  
 Statistics ..... 0463

**Applied Sciences**  
 Applied Mechanics ..... 0346  
 Computer Science ..... 0984

Engineering  
 General ..... 0537  
 Aerospace ..... 0538  
 Agricultural ..... 0539  
 Automotive ..... 0540  
 Biomedical ..... 0541  
 Chemical ..... 0542  
 Civil ..... 0543  
 Electronics and Electrical ..... 0544  
 Heat and Thermodynamics ..... 0348  
 Hydraulic ..... 0545  
 Industrial ..... 0546  
 Marine ..... 0547  
 Materials Science ..... 0794  
 Mechanical ..... 0548  
 Metallurgy ..... 0743  
 Mining ..... 0551  
 Nuclear ..... 0552  
 Packaging ..... 0549  
 Petroleum ..... 0765  
 Sanitary and Municipal ..... 0554  
 System Science ..... 0790  
 Geotechnology ..... 0428  
 Operations Research ..... 0796  
 Plastics Technology ..... 0795  
 Textile Technology ..... 0994

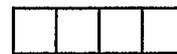
**PSYCHOLOGY**

General ..... 0621  
 Behavioral ..... 0384  
 Clinical ..... 0622  
 Developmental ..... 0620  
 Experimental ..... 0623  
 Industrial ..... 0624  
 Personality ..... 0625  
 Physiological ..... 0989  
 Psychobiology ..... 0349  
 Psychometrics ..... 0632  
 Social ..... 0451



Nom \_\_\_\_\_

Dissertation Abstracts International est organisé en catégories de sujets. Veuillez s.v.p. choisir le sujet qui décrit le mieux votre thèse et inscrivez le code numérique approprié dans l'espace réservé ci-dessous.



U·M·I

SUJET

CODE DE SUJET

Catégories par sujets

**HUMANITÉS ET SCIENCES SOCIALES**

**COMMUNICATIONS ET LES ARTS**

Architecture	0729
Beaux-arts	0357
Bibliothéconomie	0399
Cinéma	0900
Communication verbale	0459
Communications	0708
Danse	0378
Histoire de l'art	0377
Journalisme	0391
Musique	0413
Sciences de l'information	0723
Théâtre	0465

**ÉDUCATION**

Généralités	515
Administration	0514
Art	0273
Collèges communautaires	0275
Commerce	0688
Économie domestique	0278
Éducation permanente	0516
Éducation préscolaire	0518
Éducation sanitaire	0680
Enseignement agricole	0517
Enseignement bilingue et multiculturel	0282
Enseignement industriel	0521
Enseignement primaire	0524
Enseignement professionnel	0747
Enseignement religieux	0527
Enseignement secondaire	0533
Enseignement spécial	0529
Enseignement supérieur	0745
Évaluation	0288
Finances	0277
Formation des enseignants	0530
Histoire de l'éducation	0520
Langues et littérature	0279

Lecture	0535
Mathématiques	0280
Musique	0522
Orientation et consultation	0519
Philosophie de l'éducation	0998
Physique	0523
Programmes d'études et enseignement	0727
Psychologie	0525
Sciences	0714
Sciences sociales	0534
Sociologie de l'éducation	0340
Technologie	0710

**LANGUE, LITTÉRATURE ET LINGUISTIQUE**

Langues	
Généralités	0679
Anciennes	0289
Linguistique	0290
Modernes	0291
Littérature	
Généralités	0401
Anciennes	0294
Comparée	0295
Médiévale	0297
Moderne	0298
Africaine	0316
Américaine	0591
Anglaise	0593
Asiatique	0305
Canadienne (Anglaise)	0352
Canadienne (Française)	0355
Germanique	0311
Latino-américaine	0312
Moyen-orientale	0315
Romane	0313
Slave et est-européenne	0314

**PHILOSOPHIE, RELIGION ET THÉOLOGIE**

Philosophie	0422
Religion	
Généralités	0318
Clergé	0319
Études bibliques	0321
Histoire des religions	0320
Philosophie de la religion	0322
Théologie	0469

**SCIENCES SOCIALES**

Anthropologie	
Archéologie	0324
Culturelle	0326
Physique	0327
Droit	0398
Économie	
Généralités	0501
Commerce-Affaires	0505
Économie agricole	0503
Économie du travail	0510
Finances	0508
Histoire	0509
Théorie	0511
Études américaines	0323
Études canadiennes	0385
Études féministes	0453
Folklore	0358
Géographie	0366
Gérontologie	0351
Gestion des affaires	
Généralités	0310
Administration	0454
Banques	0770
Comptabilité	0272
Marketing	0338
Histoire	
Histoire générale	0578

Ancienne	0579
Médiévale	0581
Moderne	0582
Histoire des noirs	0328
Africaine	0331
Canadienne	0334
États-Unis	0337
Européenne	0335
Moyen-orientale	0333
Latino-américaine	0336
Asie, Australie et Océanie	0332
Histoire des sciences	0585
Loisirs	0814
Planification urbaine et régionale	0999
Science politique	
Généralités	0615
Administration publique	0617
Droit et relations internationales	0616
Sociologie	
Généralités	0626
Aide et bien-être social	0630
Criminologie et établissements pénitentiaires	0627
Démographie	0938
Études de l'individu et de la famille	0628
Études des relations interethniques et des relations raciales	0631
Structure et développement social	0700
Théorie et méthodes	0344
Travail et relations industrielles	0629
Transports	0709
Travail social	0452

**SCIENCES ET INGÉNIERIE**

**SCIENCES BIOLOGIQUES**

Agriculture	
Généralités	0473
Agronomie	0285
Alimentation et technologie alimentaire	0359
Culture	0479
Élevage et alimentation	0475
Exploitation des péturages	0777
Pathologie animale	0476
Pathologie végétale	0480
Physiologie végétale	0817
Sylviculture et faune	0478
Technologie du bois	0746
Biologie	
Généralités	0306
Anatomie	0287
Biologie (Statistiques)	0308
Biologie moléculaire	0307
Botanique	0309
Cellule	0379
Écologie	0329
Entomologie	0353
Génétiq	0369
Limnologie	0793
Microbiologie	0410
Neurologie	0317
Océanographie	0416
Physiologie	0433
Radiation	0821
Science vétérinaire	0778
Zoologie	0472
Biophysique	
Généralités	0786
Médicale	0760

Géologie	0372
Géophysique	0373
Hydrologie	0388
Minéralogie	0411
Océanographie physique	0415
Paléobotanique	0345
Paléocologie	0426
Paléontologie	0418
Paléozoologie	0985
Palynologie	0427

**SCIENCES DE LA SANTÉ ET DE L'ENVIRONNEMENT**

Économie domestique	0386
Sciences de l'environnement	0768
Sciences de la santé	
Généralités	0566
Administration des hôpitaux	0769
Alimentation et nutrition	0570
Audiologie	0300
Chimiothérapie	0992
Dentisterie	0567
Développement humain	0758
Enseignement	0350
Immunologie	0982
Loisirs	0575
Médecine du travail et thérapie	0354
Médecine et chirurgie	0564
Obstétrique et gynécologie	0380
Ophtalmologie	0381
Orthophonie	0460
Pathologie	0571
Pharmacie	0572
Pharmacologie	0419
Physiothérapie	0382
Radiologie	0574
Santé mentale	0347
Santé publique	0573
Soins infirmiers	0569
Toxicologie	0383

**SCIENCES DE LA TERRE**

Biogéochimie	0425
Géochimie	0996
Géodésie	0370
Géographie physique	0368

**SCIENCES PHYSIQUES**

Sciences Pures	
Chimie	
Généralités	0485
Biochimie	487
Chimie agricole	0749
Chimie analytique	0486
Chimie minérale	0488
Chimie nucléaire	0738
Chimie organique	0490
Chimie pharmaceutique	0491
Physique	0494
Polymères	0495
Radiation	0754
Mathématiques	0405
Physique	
Généralités	0605
Acoustique	0986
Astronomie et astrophysique	0606
Électronique et électricité	0607
Fluides et plasma	0759
Météorologie	0608
Optique	0752
Particules (Physique nucléaire)	0798
Physique atomique	0748
Physique de l'état solide	0611
Physique moléculaire	0609
Physique nucléaire	0610
Radiation	0756
Statistiques	0463
Sciences Appliquées Et Technologie	
Informatique	0984
Ingénierie	
Généralités	0537
Agricole	0539
Automobile	0540

Biomédicale	0541
Chaleur et thermodynamique	0348
Conditionnement (Emballage)	0549
Génie aérospatial	0538
Génie chimique	0542
Génie civil	0543
Génie électronique et électrique	0544
Génie industriel	0546
Génie mécanique	0548
Génie nucléaire	0552
Ingénierie des systèmes	0790
Mécanique navale	0547
Métallurgie	0743
Science des matériaux	0794
Technique du pétrole	0765
Technique minière	0551
Techniques sanitaires et municipales	0554
Technologie hydraulique	0545
Mécanique appliquée	0346
Géotechnologie	0428
Matières plastiques (Technologie)	0795
Recherche opérationnelle	0796
Textiles et tissus (Technologie)	0794

**PSYCHOLOGIE**

Généralités	0621
Personnalité	0625
Psychobiologie	0349
Psychologie clinique	0622
Psychologie du comportement	0384
Psychologie du développement	0620
Psychologie expérimentale	0623
Psychologie industrielle	0624
Psychologie physiologique	0989
Psychologie sociale	0451
Psychométrie	0632



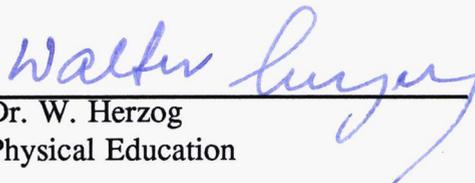
**THE UNIVERSITY OF CALGARY  
FACULTY OF GRADUATE STUDIES**

The undersigned certify that they have read, and recommend to the Faculty of Graduate Studies for acceptance, a thesis entitled "A Dynamic Model of Skeletal Muscle" submitted by Johnny Mattar in partial fulfillment of the requirements for the degree of Master of Science.



---

Supervisor, Dr. M. Epstein  
Mechanical Engineering



---

Dr. W. Herzog  
Physical Education



---

Dr. B. MacIntosh  
Physical Education

17<sup>th</sup> December, 1993

---

Date

## **Abstract**

A two-dimensional dynamic muscle model is described and assessed qualitatively. The model is based on a physical principle, namely the principle of virtual work, and includes a kinematic constraint of constant volume.

Paying special attention to physical consistency and simplicity, this model incorporates consistent equilibrium considerations which relate the external muscle forces to the internal fibre forces. The equations describing the model are expressed by a set of non linear algebraic equations which are solved by the Newton-Raphson iterative method.

The model incorporates properties of the individual muscle fibres down to the sarcomere level, such as the force-length and force-velocity relationships.

Keeping with the presently accepted view of muscle structure and function, the assumptions underlying the model are:

- (a) muscle fibres are one-dimensional entities and
- (b) the mechanical muscle behaviour is a reflection of the active and passive muscle fibre characteristics exclusively.

## Acknowledgements

Many thanks to

Dr. Marcelo Epstein, for his patience, for the generous sharing of his time and insights, and for being a friend.

Dr. Walter Herzog, for sharing his experimental knowledge of muscle behaviour.

Fekete Associates Inc., for their generous sponsorship and support of my work.

Mrs. Sydney Fielder, for an excellent job in typing parts of the manuscript, but most of all for being a friend.

To my Canadian family; Louis, Sheila, Pierre, Monique and Andrew, thanks for providing me with a home away from home.

Many thanks also to my family, for their patience, understanding and unceasing support.

Thank you also to all those who helped me in many ways, particularly

Brad Stephens  
Karel Zaoral  
Alex Berze

In memory of my grandmother, Raïfa.

## Table of Contents

Approval Page	ii
Abstract	iii
Dedication	iv
Acknowledgments	v
Table of Contents	vi
List of Tables	ix
List of Figures	x
<b>1. Introduction</b>	<b>1</b>
<b>2. The Principle of Virtual Work</b>	<b>4</b>
2.1 The equivalence of the virtual work equations to the equilibrium equations.	7
2.1.1 Equilibrium Equation Consideration	7
2.1.2 Virtual Work Consideration	8
2.2 Problem Formulation	11
2.2.1 Kinematics	11
2.2.2 Time Dependent Quantities	13
2.2.3 Equilibrium Equations	14
2.3 Newton-Raphson iterative method for non linear system of equations	15
2.4 Discussion	17
<b>3. Kinematic Constraints</b>	<b>18</b>
3.1 Maxima and Minima	18
3.2 Lagrange Multipliers and Constraints	20
3.3 Implementation of Lagrange Multipliers	23

3.4	A Typical Constraint	25
3.5	Discussion	27
<b>4.</b>	<b>Time Dependent Problems and Dynamics</b>	<b>29</b>
4.1	Introduction	29
4.2	Implicit Time Integration Methods	31
4.2.1	The Houbolt Method	31
4.2.2	The Wilson- $\theta$ Method	32
4.2.3	The Newmark Method	34
4.3	Implementation of the Wilson Method	36
<b>5.</b>	<b>Computer Implementation of Theory</b>	<b>38</b>
5.1	Equation Implementation	38
5.1.1	Virtual Work Implementation	39
5.1.2	Constraint Implementation	41
5.1.3	Dynamic Implementation	42
5.1.4	Adaptation for Muscle Behaviour	44
5.1.5	Discussion	44
5.2	Program Input	45
5.3	Program Output	46
<b>6.</b>	<b>An Introduction to Skeletal Muscle</b>	<b>47</b>
6.1	The Structure of Skeletal Muscle	47
6.2	Muscle Models	50
6.2.1	Functional Muscle Models	50
6.2.2	Geometric Muscle Models	52
<b>7.</b>	<b>Dynamic Muscle Model</b>	<b>55</b>
7.1	Model Assumptions and Simplifications	55

7.2	Dynamic Muscle Model Geometry and Kinematics	57
7.3	Fibre Force-Length Relation	60
7.4	Fibre Force-Velocity Relation	62
7.5	Discussion	63
<b>8.</b>	<b>Dynamic Muscle Model Exploration</b>	<b>65</b>
8.1	Introduction	65
8.2	DMM Consistency	66
8.3	Real Muscle Experiments	70
8.4	DMM Capabilities	73
<b>9.</b>	<b>Summary and Recommendations</b>	<b>81</b>
	<b>Bibliography</b>	<b>84</b>
	<b>Appendix A</b>	<b>87</b>
	<b>Appendix B</b>	<b>109</b>

## List of Tables

5.1	Skeleton listing of program highlighting functions	39
5.2	Listing of some function calls from mnewt ( )	40
5.3	Modifications to function calls due to constraint	42
5.4	Listing of some function calls relevant to the dynamic terms	43
8.1	Errors in some of the nodal displacement	69
8.2	Initial data for contrived muscle	73

## List of Figures

2.1	Particle acted upon by several forces.	4
2.2	Body Loaded at several points.	6
2.3	Variable X-section bar subjected to one dimensional forces.	7
2.4	Thin section of same bar subjected to horizontal loads.	9
2.5	Bar i - j before and after deformation.	11
3.1	Two-dimensional structure of Muscle Fibres between parallel tendon plates.	25
3.2	Two-dimensional structure after deformation.	26
4.1	Linear variation of acceleration.	32
6.1	The general structure of striated muscle. Adapted from Jolley and Purslow (1989).	48
7.1	Trapezoidal Muscle Geometry	57
7.2	Typical muscle geometry subjected to incompressibility constraint	59
7.3	Fibre Force-Length Profile	61
8.1	Typical muscle modelled with 4 panels	67
8.2	Typical muscle modelled with 8 panels	68
8.3	"Real" muscle experiment	71
8.4	The response of a muscle to slow and fast stimulation. (No inertia, Velocity dependence included)	74
8.5	The response of a muscle to slow and fast stimulation. (No velocity dependence, Inertia included)	75
8.6	The response of a muscle to slow and fast stimulation. (Inertia and Velocity dependence included)	76
8.7	The response of a velocity dependent muscle to slow stimulation. (with and without inertia)	77
8.8	The response of a velocity dependent muscle to fast stimulation. (with and without inertia)	78

## 1. Introduction

Muscles are the components of the body which are capable of active contraction. The contraction of skeletal muscle acting on the skeletal system is responsible for vertebrate locomotion and movement. The study of muscle contraction is an interesting one which can be undertaken from many aspects; biological, chemical or mechanical to name a few. The scope of this thesis allows for the study to be done from a mechanical point of view. The transmission of forces produced by the cells through the tendinous sheaths is of prime concern.

It is hoped that studying muscle behaviour from a mechanical point of view, will add another dimension to the body of knowledge already available in the fields of biochemistry, muscle physiology, neurology, etc. Ultimately, a better understanding of any problem leads to an increase in the number of ways of solving that problem. In the biomechanical field, this could lead to improvements in rehabilitation procedures and the design of prostheses.

Muscle fibres generate forces as a result of stimuli. The mechanics of muscular contraction on a microscopic scale, as well as the associated active and passive muscle fibre characteristics, are well understood both from an experimental and theoretical point of view. The most widely accepted theories of muscular contraction on a microscopic scale, that is the Cross Bridge Theory and the Sliding Filament Theory, are results of relatively recent research by Huxley [1957,1974] and Gordon et al [1966].

The earliest reference to models of entire muscle dates back to the early 17<sup>th</sup> century. Niels Stenson (1638 - 1686) recognised the pinnate structure of muscle. He

demonstrated, based on geometrical arguments, that muscles could contract without changing their volume. This was in contradiction to the views held by his contemporaries that the animal spirit was the cause of muscle shortening.

Muscle models can be divided into two groups; functional and geometrical muscle models. Only geometrical muscle models are of interest in the context of this thesis.

The currently most cited (pinnate) muscle models have been proposed by Woittiez et al. [1983,1984] and Otten [1985, 1987a, 1987b, 1988]. In general, these models are based on kinematic constraints, that is, the mode of muscle deformation is preordained, with the imposed mode of muscle deformation showing a close resemblance to the work of Stenson. The models incorporate the present knowledge of active and passive muscle fibre behaviour, and are essentially based on the assumptions of one-dimensional fibres with uni-directional fibre activity, non-interaction between neighbouring fibres, and the constancy of volume.

Anton [1991] developed a model ( straight line model ) based on the principle of virtual work in which he made use of the present knowledge of active and passive muscle fibre behaviour. His model, however, did not take into account any dynamic effects.

It is, therefore, the goal of this thesis to develop a muscle model which is based on a physical principle and will also incorporate dynamic effects. It is envisioned that the muscle will be qualitative in nature. The model will be formulated using the principle of virtual work and the kinematic constraint of constant volume will be implemented by means of the method of Lagrange Multipliers. The primary concern in developing the muscle model is directed toward its internal consistency. The present knowledge of active

and passive muscle fibre behaviour will be assumed to be correct.

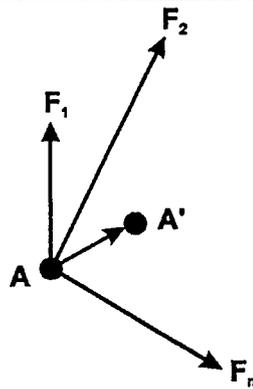
The muscle model will be able to incorporate a variable number of fibre bundles (from two to infinity), but selection of the number of fibre bundles is somewhat limited by computer processing time.

Unreferenced model comparisons to experimental results and experimental observations are also made with respect to experiments conducted independently of this thesis by Dr. Walter Herzog in his laboratory at the University of Calgary.

The theoretical aspects of the model will be derived in the early chapters. A brief introduction to muscle structure, a literature review of muscle models and the muscle model derivation will follow. The study concludes with a number of examples which highlight the capabilities of the model.

## 2. The Principle of Virtual Work

The principle of virtual work, which was first formulated by Johann J. Bernoulli<sup>1</sup>, offers an elegant way of investigating the conditions of equilibrium for a given system [Pestel & Thomson]. The term "virtual" is used because the work is done, or would be done, by the external force(s) for the imagined displacements on the system.



**Figure 2.1** Particle acted upon by several forces.

Consider a particle acted upon by several forces  $F_1, F_2, \dots, F_n$  as shown in figure 2.1. Assume that the particle undergoes a small displacement from A to A'. This displacement is possible, but it will not necessarily take place. The forces may be balanced and the particle at rest, or the particle may move under the action of the given forces in a direction different from that of A-A'. The displacement considered is therefore an imaginary displacement; it is called a *virtual displacement* and is denoted by  $\delta u$ . The symbol  $\delta u$  represents a differential of the first order; it is used to distinguish the virtual

---

<sup>1</sup> Born 1667 in Basel, Switzerland; died 1748 in the same place. The name *virtual work* was first mentioned in a letter by Bernoulli to P. Varignon (1640 - 1718), who is known for his theorem concerning the addition of force couples. It is interesting that in the Russian literature "virtual" is translated as "possible." Instead of "virtual" the famous mathematician C. F. Gauss (1777 - 1855) used the word "facultative."

displacement from the displacement  $\delta\mathbf{u}$  which would take place under actual motion. Virtual displacements may be used to determine whether the conditions of equilibrium of a particle are satisfied.

The work of each of the forces  $\mathbf{F}_1, \mathbf{F}_2, \dots, \mathbf{F}_n$  during the virtual displacement  $\delta\mathbf{u}$  is called *virtual work*. The virtual work of all the forces acting on the particle of figure 2.1 is

$$\begin{aligned}\delta U &= \mathbf{F}_1 \cdot \delta\mathbf{u} + \mathbf{F}_2 \cdot \delta\mathbf{u} + \dots + \mathbf{F}_n \cdot \delta\mathbf{u} \\ &= (\mathbf{F}_1 + \mathbf{F}_2 + \dots + \mathbf{F}_n) \cdot \delta\mathbf{u} \\ &= \mathbf{R} \cdot \delta\mathbf{u} \quad \dots (2.1)\end{aligned}$$

where  $\mathbf{R}$  is the resultant of the given forces. Thus the total virtual work of the forces  $\mathbf{F}_1, \mathbf{F}_2, \dots, \mathbf{F}_n$  is equal to the virtual work of their resultant  $\mathbf{R}$ .

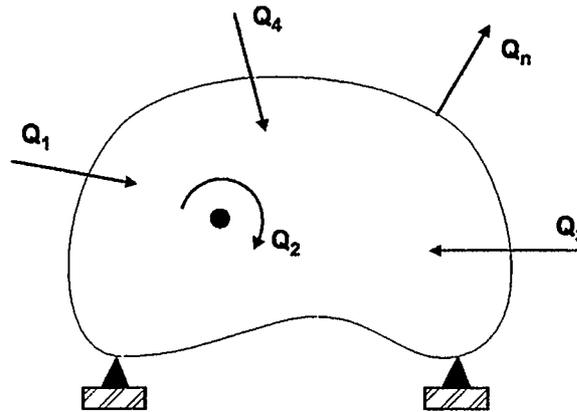
The principle of virtual work for a particle states that, *if a particle is in equilibrium, the total virtual work of the forces acting on the particle is zero for any virtual displacement of the particle*. This condition is necessary: if the particle is in equilibrium, the resultant  $\mathbf{R}$  of the forces is zero, and it follows from eqn. (2.1) that the total virtual work  $\delta U$  is zero. The condition is also sufficient: if the total virtual work  $\delta U$  is zero for any virtual displacement, the scalar product  $\mathbf{R} \cdot \delta\mathbf{u}$  is zero for any  $\delta\mathbf{u}$ , and the resultant  $\mathbf{R}$  must be zero.

In the case of a rigid body, the principle of virtual work states that, *if a rigid body is in equilibrium, the total virtual work of the external forces and couples acting on the rigid body is zero for any virtual displacement of the body*. The condition is necessary: if the body is in equilibrium, all the particles forming the body are in equilibrium and the

total virtual work of the forces acting on all the particles must be zero; but we have seen in the preceding section that the total work of the internal forces is zero; the total work of the external forces must therefore also be zero. The condition is also sufficient.

The principle of virtual work may be extended to the case of a *system of connected rigid bodies*. If the system remains connected during the virtual displacement, *only the work of the external forces need to be considered*, since the total work of the internal forces at the various connections is zero [Beer & Johnston].

Turning our attention to deformable bodies, consider a body under a system of generalised forces shown in figure 2.2



**Figure 2.2** Body loaded at several points.

The principle of virtual work is stated mathematically as

$$\delta U = Q_m \delta q_m$$

where  $\delta U$  represents the the work done internally by the Q-induced stresses under the strains engendered by the virtual deformation  $\delta \mathbf{u}$ . In words, the principle of virtual work

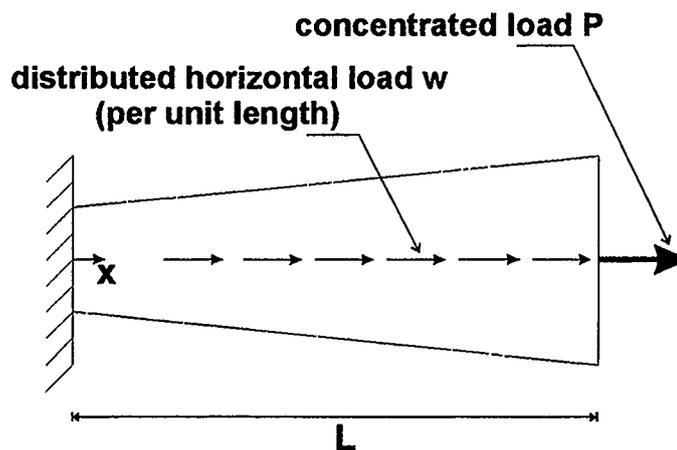
may be stated as follows [Bisplinghoff et al.]:

*If a deformable body is in equilibrium and remains in equilibrium while it is subjected to a virtual distortion compatible with the geometrical constraints, the virtual work done by the external forces is equal to the virtual work done by the internal stresses.*

### 2.1 The equivalence of the virtual work equations to the equilibrium equations.

It can be easily shown that the principle of virtual work (PVW) is a valid substitute for the usual equilibrium equations by means of a simple one dimensional example [Epstein].

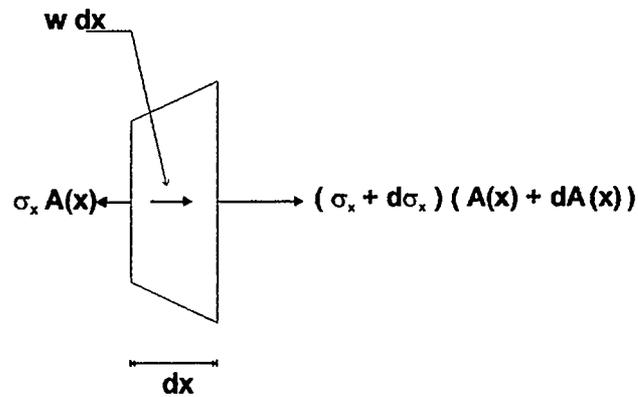
A bar of variable cross section  $A(x)$  is subjected to horizontal loads as shown in figure 2.3.



**Figure 2.3** Variable X-section bar subjected to one dimensional forces

#### 2.1.1 Equilibrium Equation Consideration:

Consider a thin section of the bar of thickness  $dx$



From the equilibrium of forces in the x direction

$$\sum F_x = 0 \quad \dots(2.2)$$

or

$$-\sigma_x A + (\sigma_x + d\sigma_x)(A + dA) + w dx = 0 \quad \dots (2.3)$$

dividing eqn. (2.3) by  $dx$  and rearranging the resultant, we obtain,

$$\frac{d(\sigma_x A)}{dx} + w = 0 \quad \dots (2.4)$$

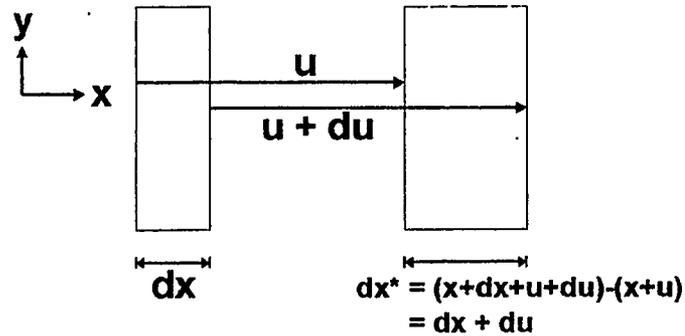
### 2.1.2 Virtual Work Consideration:

Consider the strain on a bar as shown in figure 2.4

By definition

$$\epsilon_x = \frac{dx^* - dx}{dx} = \frac{du}{dx} \quad \dots (2.5)$$

thus for a given virtual displacement,  $\delta u$ , the corresponding virtual strain is



**Figure 2.4** Thin section of same bar subjected to horizontal loads

$$\delta \epsilon_x = \frac{d(\delta u)}{dx} \quad \dots (2.6)$$

The internal virtual work (I.V.W.) is defined as

$$IVW = \int_V (\delta \epsilon)^T (\sigma) dV = \int_0^L \frac{d(\delta u)}{dx} \sigma A dx \quad \dots (2.7)$$

The external virtual work (E.V.W.) is defined as

$$EVW = \int_0^L \delta u \cdot w dx + \delta u|_{x=L} \cdot P \quad \dots (2.8)$$

From the principle of virtual work, the internal virtual work is equal to the external virtual work for **all** compatible virtual displacements.

$$\int_0^L \frac{d(\delta u)}{dx} \sigma A dx = \int_0^L \delta u \cdot w dx + \delta u|_{x=L} \cdot P \quad \dots (2.9)$$

Consider the left hand side, and apply the divergence theorem (integrate by parts).

Substitute the result in eqn (2.9) to give

$$\int_0^L \frac{d(\delta u)}{dx} \sigma A dx = \int_0^L \delta u \cdot w dx + \delta u \Big|_{x=L} \cdot P \quad \dots (2.10)$$

This is an identity which is valid for all  $\delta u$ 's provided they do not violate the constraint ( i.e.  $\delta u$  vanishes at  $x = 0$  ). Any equality obtained from an identity by choosing particular values of the variables is valid. Choose all those  $\delta u$  's which vanish at both ends. These do not violate the constraints, with the added advantage that only the integral terms remain. This can be concluded for all such  $\delta u$  's

$$\int_0^L \delta u \cdot \frac{d(\sigma A)}{dx} dx = \int_0^L \delta u \cdot w dx \quad \dots (2.11)$$

Eqn (2.11) is true for a wide range only if the multipliers of  $\delta u$  under each integral are the same<sup>1</sup>. Thus we conclude

$$-\frac{d(\sigma A)}{dx} = w \quad \dots (2.12)$$

which is the equilibrium equation.

The boundary condition of force (namely, the stress at the right end must be  $P/A$ ) is also obtained from eqn (2.10) at  $x = L$ .

It will also be shown (Chapter 3) that the P.V.W. offers an advantage over equilibrium equation derivation by allowing the addition of constraint equations to the problem. ( e.g. constant volume panels, or displacement along a specified path ).

---

<sup>1</sup> Formally, we are applying the the so-called fundamental lemma of the calculus of variations.

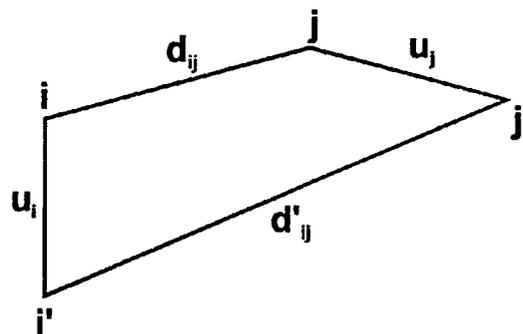
## 2.2 Problem Formulation:

The particular problem that will be considered in this thesis involves a two-dimensional space truss (used to model a muscle). The deformation due to a loading experienced by this truss is not restricted in magnitude. It must, however, remain in an in-plane direction. The constitutive equation can also be modified to represent different types of materials. The virtual work equations will be developed for this truss system and the simplicity of the method will be highlighted.

Consider a pin-jointed space truss having  $M$  nodes. Denote by  $N$  the number of free (unrestrained) joints so that  $M-N$  is the total number of supports. In the present analysis, only fixed supports are dealt with [Epstein & Tene].

External forces act on the nodes and do not vary during the process of deformation, neither in magnitude, nor in direction.

### 2.2.1 Kinematics:




---

**Figure 2.5** Bar  $i$ - $j$  before and after deformation.

Let  $i$  and  $j$  (Fig 2.5) be two nodes connected by a bar  $i$ - $j$ . After deformation the

position of the nodes will be  $i'$  and  $j'$  respectively. Denote by  $\mathbf{u}_i$  the displacement of node  $i$ , by  $\mathbf{u}_j$  the displacement of node  $j$  and by  $\mathbf{d}_{ij}$  and  $\mathbf{d}'_{ij}$  the vectors from node  $i$  to node  $j$  before and after deformation, respectively.

Using vector equalities,

$$\mathbf{d}_{ij} + \mathbf{u}_j = \mathbf{u}_i + \mathbf{d}'_{ij} \quad \dots (2.13)$$

$$\text{or } \mathbf{d}'_{ij} = \mathbf{d}_{ij} + (\mathbf{u}_j - \mathbf{u}_i) \quad \dots (2.14)$$

are obtained. The length of  $\mathbf{d}'_{ij}$  is obtained from

$$d'_{ij}{}^2 = d_{ij}{}^2 + 2\mathbf{d}_{ij} \cdot (\mathbf{u}_j - \mathbf{u}_i) + (\mathbf{u}_j - \mathbf{u}_i) \cdot (\mathbf{u}_j - \mathbf{u}_i) \quad \dots (2.15)$$

$$\text{or } (d'_{ij} + d_{ij})(d'_{ij} - d_{ij}) = 2\mathbf{d}_{ij} \cdot (\mathbf{u}_j - \mathbf{u}_i) + (\mathbf{u}_j - \mathbf{u}_i) \cdot (\mathbf{u}_j - \mathbf{u}_i) \quad \dots (2.16)$$

in which  $d'_{ij}$  and  $d_{ij}$  are the lengths of  $\mathbf{d}'_{ij}$  and  $\mathbf{d}_{ij}$  respectively.

Denote

$$e_{ij} = d'_{ij} - d_{ij} \quad \dots (2.17)$$

Eqns. (2.16) and (2.17) yield

$$(2d_{ij} + e_{ij}) \cdot e_{ij} = 2\mathbf{d}_{ij} \cdot (\mathbf{u}_j - \mathbf{u}_i) + (\mathbf{u}_j - \mathbf{u}_i) \cdot (\mathbf{u}_j - \mathbf{u}_i) \quad \dots (2.18)$$

rearranging eqn. (2.18) yields a non linear equation,

$$e_{ij} = -d_{ij} + \sqrt{d_{ij}{}^2 + 2\mathbf{d}_{ij} \cdot (\mathbf{u}_j - \mathbf{u}_i) + (\mathbf{u}_j - \mathbf{u}_i) \cdot (\mathbf{u}_j - \mathbf{u}_i)} \quad \dots (2.19)$$

where

$e_{ij}$  is the physical elongation of the bar ,

$\mathbf{d}_{ij}$  is the vector from node  $i$  to node  $j$  ,

$d_{ij}$  is the length of  $\mathbf{d}_{ij}$  ,

$\mathbf{u}_i$  and  $\mathbf{u}_j$  are the displacement vectors of node  $i$  and node  $j$  respectively.

### 2.2.2 Time Dependent Quantities:

The nature of the problem being dealt with involves velocity and acceleration terms. The velocity term, which is the time rate of change of elongation, is derived in this section. The acceleration term will be derived in chapter 4.

Recall eqn. (2.15)

$$d'_{ij}{}^2 = d_{ij}{}^2 + 2\mathbf{d}_{ij} \cdot (\mathbf{u}_j - \mathbf{u}_i) + (\mathbf{u}_j - \mathbf{u}_i) \cdot (\mathbf{u}_j - \mathbf{u}_i)$$

differentiating w.r.t time we obtain

$$\frac{\partial (d'_{ij})^2}{\partial t} = \left( \frac{\partial (d'_{ij})}{\partial \mathbf{u}_j} \cdot \dot{\mathbf{u}}_j + \frac{\partial (d'_{ij})}{\partial \mathbf{u}_i} \cdot \dot{\mathbf{u}}_i \right) \quad \dots (2.20)$$

further simplification and substitution of equations (2.15) and (2.17) gives the non linear result

$$\dot{e}_{ij} = \dot{d}'_{ij} = \frac{(\mathbf{d}_{ij} + \mathbf{u}_j + \mathbf{u}_i) \cdot (\dot{\mathbf{u}}_j - \dot{\mathbf{u}}_i)}{d'_{ij}} \quad \dots (2.21)$$

### 2.2.3 Equilibrium Equations:

Let

$$\mathbf{N}_{ij} = f(\mathbf{e}_{ij}, \dot{\mathbf{e}}_{ij}) \quad \dots (2.22)$$

represent the relationship between elongations and internal forces in the bars. Denote by  $F_i$  the force acting on node  $i$ . The variational form of the equilibrium equation of the structure is

$$\sum_{j>1}^M \sum_{i=1}^N \mathbf{N}_{ij}(\mathbf{e}_{ij}, \dot{\mathbf{e}}_{ij}) \cdot \delta \mathbf{e}_{ij} - \sum_{i=1}^N \mathbf{F}_i \cdot \delta \mathbf{u}_i = 0 \quad \dots (2.23)$$

for all  $\delta \mathbf{u}$  's compatible with supports.

$$\sum_{j>1}^M \sum_{i=1}^N \mathbf{N}_{ij}(\mathbf{e}_{ij}, \dot{\mathbf{e}}_{ij}) \cdot \delta \mathbf{e}_{ij} \quad \text{is the internal virtual work of the system of bars,}$$

$$\sum_{i=1}^N \mathbf{F}_i \cdot \delta \mathbf{u}_i = 0 \quad \text{is the external virtual work of the same system,}$$

$$\delta \mathbf{e}_{ij} = \frac{(\mathbf{d}_{ij} + \mathbf{u}_j + \mathbf{u}_i) \cdot (\delta \mathbf{u}_j - \delta \mathbf{u}_i)}{\mathbf{d}'_{ij}} \quad \text{is the variation of } \mathbf{e}_{ij}.$$

Equations (2.23) represent a non linear system of equations. These exact equilibrium equations are formed by equating to zero the coefficients of each of the independent variations  $\delta \mathbf{u}_i$ . They are valid for arbitrarily large strains and displacements (eqns. 2.19 & 2.21) and make use of exact kinematic equations (strain-displacement relations). The resulting non linear equations are written using the known initial geometry of the structure. It should be noted that the strain-displacement relations can be represented by Hooke's law for small strains, or by any other physically valid strain-displacement relation (e.g. constitutive equation for a rubber like material) for large strains.

Equation (2.23), now takes the form

$$\sum_{i=1}^N \phi_i(\mathbf{u}_1, \mathbf{u}_2, \dots, \mathbf{u}_N) \cdot \delta \mathbf{u}_i = 0 \quad \dots (2.24 \text{ a})$$

Since this equation is an identity, each of the coefficients of the  $\delta \mathbf{u}$ 's will vanish

identically and the remaining coefficients are merely the equilibrium equations.

$$\phi_i(\mathbf{u}_1, \mathbf{u}_2, \dots, \mathbf{u}_N) = 0 \quad i = 1, 2, \dots, N \quad \dots (2.24 \text{ b})$$

One of the best methods of solution available is the Newton-Raphson method. The method has achieved a wide range of acceptance, particularly for large displacements and stability analyses. Obviously, in order to achieve such a degree of acceptance, this solution procedure must possess some excellent characteristics. The most important of these characteristics is the ability of the procedure to converge for highly non linear behaviour.[Tillerson et. al.]. This method will be discussed in the next section.

### 2.3 Newton-Raphson iterative method for non linear system of equations

Consider a system of K non linear equations ( equivalent to eqn. (2.24) ) to be solved for K unknowns, i.e. K functional relations to be zeroed, involving variables  $u_i$ , for  $i = 1, 2, \dots, K$ .

$$f_i(u_1, u_2, u_3, \dots, u_K) = 0, \quad i = 1, 2, 3, \dots, K \quad \dots (2.25)$$

Let  $\mathbf{U}$  denote the entire vector of values  $u_i$ . Then, in the neighbourhood of  $\mathbf{U}$ , each of the functions  $f_i$  can be expanded in Taylor Series:

$$f_i(\mathbf{U} + \delta \mathbf{U}) = f_i(\mathbf{U}) + \sum_{j=1}^K \frac{\partial f_i}{\partial u_j} \cdot \Delta u_j + 0.(\Delta \mathbf{U})^2 \quad i=1, 2, \dots, K \quad \dots (2.26)$$

Neglecting terms of the order  $\Delta \mathbf{U}^2$  and higher, we obtain a set of linear equations for the corrections  $\Delta \mathbf{U}$  that eventually yield the solution, namely,

$$\sum_{j=1}^K a_{ij} \Delta u_j = b_i, \quad i = 1, 2, \dots, K \quad \dots (2.27)$$

where,

$$b_i = -f_i$$

and

$$a_{ij} = \frac{\partial f_i}{\partial u_j} = \begin{vmatrix} \frac{\partial f_1}{\partial u_1} & \frac{\partial f_1}{\partial u_2} & \dots & \frac{\partial f_1}{\partial u_N} \\ \cdot & \cdot & \dots & \cdot \\ \cdot & \cdot & \dots & \cdot \\ \frac{\partial f_N}{\partial u_1} & \frac{\partial f_N}{\partial u_2} & \dots & \frac{\partial f_N}{\partial u_N} \end{vmatrix}$$

$a_{ij}$  is called the Jacobian.

The quantity  $b_i$  represents the vector of the values of each function, whereas  $a_{ij}$  represents the matrix of the partial derivatives of the same functions (  $a_{ij}$  is also referred to as the stiffness matrix ).

Equation (2.27) is solved by LU decomposition and the corrections,  $\Delta U$ , are then added to the solution vector,  $U$ :

$$u_i^{\text{new}} = u_i^{\text{old}} + \Delta u_i, \quad i = 1, 2, \dots, K \quad \dots (2.28)$$

The process is iterated to convergence [Press et. al.]. The relative error of each function and its arguments is checked until satisfactory convergence is achieved.

**2.4 Discussion:**

Having introduced the principle of virtual work and derived the equations of equilibrium for a two dimensional truss, the next step is the introduction of kinematic constraints. Specifically, the method of Lagrange Multipliers is used to link these constraints into the problem. This method is introduced in chapter 3 and the required modifications to the equilibrium equations are discussed.

The introduction of inertia terms is done in chapter 4. A brief review of the relevant methods is undertaken and one particular method, the Wilson method, is implemented.

### 3. Kinematic Constraints

The introduction of the method of Lagrange Multipliers forms the core of this chapter. It is preceded by a brief review of the determination of maxima and minima as relevant to the calculus of variations. The mathematical derivation of a typical kinematic constraint, relevant to the muscle model, is given at the end of this chapter.

#### 3.1 Maxima and Minima:

Applications of the *calculus of variations* are concerned chiefly with the determination of maxima and minima of certain expressions involving unknown functions. Certain techniques involved are analogous to procedures in differential calculus, which are briefly reviewed in this section [Hilderbrand].

The problem of determining maximum and minimum values of a function  $y = f(x)$  for values of  $x$  in a certain interval  $(a,b)$  is an important one in differential calculus. If in that interval,  $f(x)$  has a continuous derivative, then a *necessary* condition for the existence of a maximum or minimum at a point  $x_0$  *inside*  $(a,b)$  is that  $dy/dx = 0$ , at  $x_0$ . A *sufficient* condition that  $y$  be a maximum (or a minimum) at  $x_0$ , relative to values at neighbouring points, is that, in addition  $d^2y/dx^2 < 0$  (or  $d^2y/dx^2 > 0$ ) at that point.

If  $z$  is a function of two independent variables, say  $z = f(x,y)$ , in a region  $\mathbb{R}$ , and if the partial derivatives  $\partial z/\partial x$  and  $\partial z/\partial y$  exist and are continuous throughout  $\mathbb{R}$ , then *necessary* conditions that  $z$  possesses a relative maximum or minimum at an interior point  $(x_0, y_0)$  of  $\mathbb{R}$ , are that  $\partial z/\partial x = 0$  and  $\partial z/\partial y = 0$  simultaneously at  $(x_0, y_0)$ . These two requirements are equivalent to the single requirement that

$$dz = \frac{\partial z}{\partial x} dx + \frac{\partial z}{\partial y} dy = 0 \quad \dots (3.1)$$

at a point  $(x_0, y_0)$  for arbitrary values of both  $dx$  and  $dy$ . *Sufficient* conditions for either a maximum (or minimum) involve certain inequalities among the second partial derivatives

$$\frac{\partial^2 z}{\partial x^2} < 0 \quad \wedge \quad \frac{\partial^2 z}{\partial x^2} \frac{\partial^2 z}{\partial y^2} - \left( \frac{\partial^2 z}{\partial x \partial y} \right)^2 > 0 \quad \dots (3.2a)$$

or

$$\frac{\partial^2 z}{\partial x^2} > 0 \quad \wedge \quad \frac{\partial^2 z}{\partial x^2} \frac{\partial^2 z}{\partial y^2} - \left( \frac{\partial^2 z}{\partial x \partial y} \right)^2 > 0 \quad \dots (3.2b)$$

More generally, a *necessary* condition that a continuously differentiable function  $f(x_1, x_2, \dots, x_p)$  of  $p$  variables  $x_1, x_2, \dots, x_p$  have a relative maximum or minimum value at an interior point of a region is that

$$df = \frac{\partial f}{\partial x_1} dx_1 + \frac{\partial f}{\partial x_2} dx_2 + \dots + \frac{\partial f}{\partial x_p} dx_p = 0 \quad \dots (3.3)$$

at that point, for all *permissible* values of the differentials  $dx_1, dx_2, \dots, dx_p$ .

At a point satisfying eqn. (3.3), the function  $f$  is said to be *stationary*.

If the  $p$  variables are all independent, the  $p$  differentials can be assigned arbitrarily, and it follows that eqn (3.3) is then equivalent to the  $p$  conditions

$$\frac{\partial f}{\partial x_1} = \frac{\partial f}{\partial x_2} = \dots = \frac{\partial f}{\partial x_p} = 0 \quad \dots (3.4)$$

*Sufficient* conditions that values of the variables satisfying eqns. (3.3) or (3.4) actually determine maxima (or minima) involve certain inequalities similar to those shown in eqns. (3.2a) or (3.2b).

### 3.2 Lagrange Multipliers and Constraints:

Suppose that  $p$  variables are not independent, but are related by, say,  $R$  conditions each of the form

$$\psi_k(x_1, x_2, \dots, x_p) = 0$$

Then, at least theoretically, these  $R$  equations generally can be solved to express  $R$  of the variables in terms of the  $p-R$  remaining variables, and hence to express  $f$  and  $df$  in terms of  $p-R$  independent variables and their differentials. Alternatively,  $R$  linear relations among the  $p$  differentials can be obtained by differentiation. These conditions permit the expression of  $R$  of the *differentials* as linear combinations of the differentials of the  $p-R$  independent variables. If eqn. (3.3) is expressed in terms of these differentials, their coefficients must then vanish, giving  $p-R$  conditions for stationary values of  $f$  which supplement the  $R$  constraint equations.

A convenient alternative procedure used in these cases consists of the introduction of the so-called *Lagrange Multipliers*. To illustrate their use, consider the problem of obtaining stationary values of  $f(x, y, z)$

$$df \equiv f_x dx + f_y dy + f_z dz \quad \dots (3.5)$$

subject to the two constraints

$$\psi_1(x, y, z) = 0 \quad \dots (3.6a)$$

$$\psi_2(x, y, z) = 0 \quad \dots (3.6b)$$

Since the three variables must satisfy the two auxiliary conditions eqns. (3.6a,b), only one variable can be considered as independent. Eqns. (3.6a,b) imply the differential relations

$$\psi_{1x} dx + \psi_{1y} dy + \psi_{1z} dz = 0 \quad \dots (3.7a)$$

$$\psi_{2x} dx + \psi_{2y} dy + \psi_{2z} dz = 0 \quad \dots (3.7b)$$

The procedure outlined above would consist of first solving eqns. (3.7a,b) for, say, dx and dy in terms of dz (if this is possible) and of introducing the results into eqn. (3.5), to give a result of the form

$$df \equiv (\dots) dz = 0$$

Since dz can be assigned arbitrarily, the vanishing of the indicated expression in the parentheses in this form is the desired condition that f be stationary when eqns. (3.6a,b) are satisfied.

As an alternative procedure, eqns. (3.7a) and (3.7b) are multiplied respectively by the quantities  $\lambda_1$  and  $\lambda_2$  to be specified presently, and the results added to eqn. (3.5). Since the right-handed members are all zeros, there follows

$$\begin{aligned}
 & (f_x + \lambda_1 \psi_{1x} + \lambda_2 \psi_{2x}) dx + (f_y + \lambda_1 \psi_{1y} + \lambda_2 \psi_{2y}) dy \\
 & + (f_z + \lambda_1 \psi_{1z} + \lambda_2 \psi_{2z}) dz = 0 \quad \dots (3.8)
 \end{aligned}$$

for arbitrary values of  $\lambda_1$  and  $\lambda_2$ . Let  $\lambda_1$  and  $\lambda_2$  be determined so that two of the parentheses in eqn. (3.8) vanish<sup>1</sup>. Then the differential multiplying the remaining parenthesis can be arbitrarily assigned, and hence that parenthesis must also vanish. Thus we must have

$$\frac{\partial f}{\partial x} + \lambda_1 \frac{\partial \psi_1}{\partial x} + \lambda_2 \frac{\partial \psi_2}{\partial x} = 0 \quad \dots (3.9a)$$

$$\frac{\partial f}{\partial y} + \lambda_1 \frac{\partial \psi_1}{\partial y} + \lambda_2 \frac{\partial \psi_2}{\partial y} = 0 \quad \dots (3.9b)$$

$$\frac{\partial f}{\partial z} + \lambda_1 \frac{\partial \psi_1}{\partial z} + \lambda_2 \frac{\partial \psi_2}{\partial z} = 0 \quad \dots (3.9c)$$

Eqns. (3.9a,b,c) and eqns. (3.6a,b) comprise five equations determining  $x$ ,  $y$ ,  $z$  and  $\lambda_1$ ,  $\lambda_2$ . The quantities  $\lambda_1$  and  $\lambda_2$  are known as the *Lagrange Multipliers*. Their introduction frequently simplifies the relevant algebra in problems of the type considered. In many applications, they are found to have physical significance as well.

It should be noted that there are other types of problems which are greatly simplified by the introduction of Lagrange multipliers. These will not be discussed, as they are not relevant in the context of this thesis.

---

<sup>1</sup>If two of the parentheses do not vanish, then the functions  $\phi_1$  and  $\phi_2$  would be *functionally dependent*, so that the two constraint eqns. (3.6a) and (3.6b) would be either equivalent or incompatible.

### 3.3 Implementation of Lagrange Multipliers:

This section deals with the implementation of Lagrange multipliers in the context of the thesis. The notation used is consistent with that of chapter 2. Consequently, the terms defined in the previous section are renamed. The terms from the previous section are stated first and those from the previous chapter are shown as equivalent.

$$\mathbf{f} \equiv \phi (u_i)$$

$$\psi \equiv g (u_i) \text{ and}$$

x, y, and z are equivalent to  $u_i$  .

The introduction of Lagrange Multipliers modifies the virtual work calculations and increases the vector of unknowns by one entry per kinematic constraint.

Recall eqn. (2.24 a)

$$\sum_{i=1}^N \phi^i(u_1, u_2, \dots, u_N) \cdot \delta u_i = 0$$

which is an expression for the virtual work in terms of the known initial geometry.

By introducing a constraint condition, say,  $g (u_1, u_2, \dots, u_N) = 0$  to the system, eqn. (2.24 a) is modified and takes the form of

$$\sum_{i=1}^N \left( \phi^i \delta u_i + \lambda \frac{\partial g}{\partial u_i} \delta u_i \right) + g \delta \lambda = 0 \quad \dots (3.10)$$

This is consistent with the derivations shown in eqns. (3.9a,b,c) and (3.6a,b). Since eqn. (3.10) is an identity, the coefficients of  $\delta u$  and  $\delta \lambda$  must vanish identically, giving the equilibrium equations:

$$\Psi^i = \phi^i + \lambda \frac{\partial g}{\partial u_i} = 0$$

$$g = 0$$

for all  $\delta u$ 's compatible with supports.

In the formulation of the modified stiffness matrix, recall eqn. (2.27).

$$\sum_{i=1}^N a_{ij} \Delta u_j = b_i$$

where  $i = 1, 2, \dots, N$  and  $N$  is the number of unknown variables. The addition of a constraint condition results in an increase of the vector of unknowns by one. i.e.  $i = 1, 2, \dots, N+1$ . The Jacobian  $a_{ij}$ , which is essentially a matrix of the partial derivatives of the equilibrium equations, is also modified to give

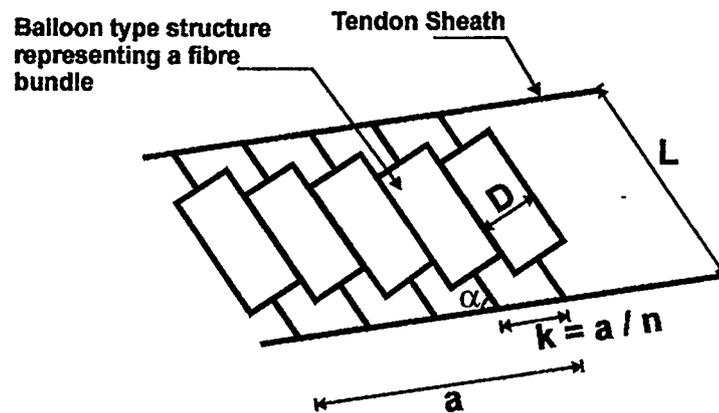
$$a_{ij} = \left[ \begin{array}{cc|c} \frac{\partial \phi^i}{\partial u^j} + \lambda \frac{\partial^2 g}{\partial u^i \partial u^j} & & \frac{\partial g}{\partial u^j} \\ \hline & & \\ \hline & \frac{\partial g}{\partial u^i} & 0 \end{array} \right]$$

Since the exact locations of each constraint equation entry in the matrix  $a_{ij}$  are known, the modifications to the computer program are therefore simple to implement. This will be shown in chapter 5.

### 3.4 A Typical Constraint:

The constancy of volume during muscle contraction forms an important part of this thesis. There have been many publications in the journals of biomechanics and physiology that deal with this issue. In this sub-section, the mathematical foundation of a constant volume (i.e. incompressible) "panel" will be examined.

Consider an initial geometry as shown in figure 3.1



**Figure 3.1** Two-dimensional structure of muscle fibres between parallel tendon plates.

where

$D$  is the diameter of a fibre bundle,

$k$  can be related to the stiffness of the tendon sheath,

$n$  is the number of fibre bundles,

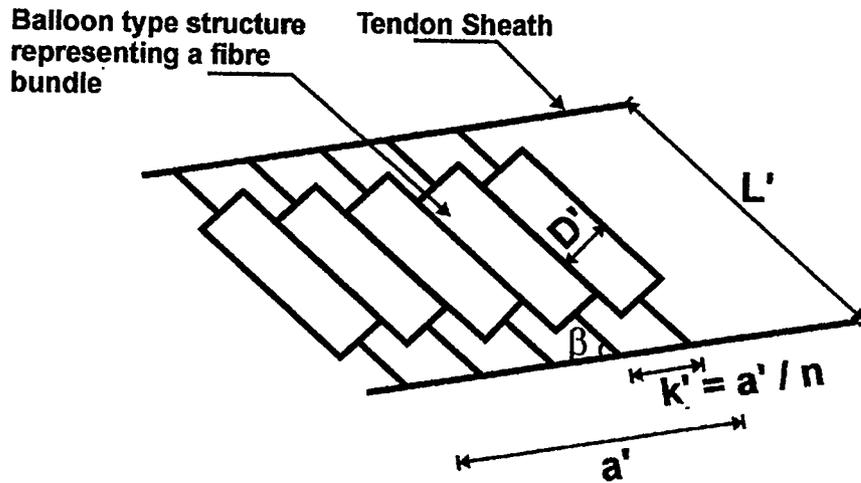
$\alpha$  is the angle formed between the fibre bundle and the tendon sheath,

$L$  is the distance between tendon sheaths,

$a$  is the distance along the tendon sheath.

By using simple trigonometry, the parameter  $k$  is calculated as  $k = D/\sin \alpha$ . This can also be represented by  $k = a/n$ .

Consider this same structure after a slight deformation as shown in figure 3.2.



**Figure 3.2** Two-dimensional structure after deformation.

Assuming that the "balloons" remain in contact and preserve their volume, then  $k' = a'/n$ , or alternatively  $k' = D'/\sin \beta$ .

From the condition of incompressibility, (or constant volume),  $D'^2 L' = D^2 L$ .

Re-arranging and substituting gives,

$$\frac{k^2 \sin^2 \alpha}{k'^2 \sin^2 \beta} L = L'$$

Further rearranging yields a form of the equation which is easily incorporated in the computer program:

$$L a^2 \sin^2 \alpha - L' a'^2 \sin^2 \beta = 0$$

The sine of the angle is calculated from the vector product of the two intersecting vectors, the fibre bundle and the tendon sheath. The mathematical form of this is:

$$\mathbf{A} \times \mathbf{B} = AB \sin(\theta)$$

where  $\mathbf{A}$  and  $\mathbf{B}$  are vectors representing the lengths  $L$  (or  $L'$ ) and  $k$  (or  $k'$ ) respectively.

The angle  $\theta$  represents the angle  $\alpha$  (or  $\beta$ ).

The quantities  $a$  and  $L$  are easily obtained from the initial geometry and the quantities  $a'$  and  $L'$  are calculated in terms of the displacements.

### 3.5 Discussion:

In highlighting the simplicity of the method of Lagrange Multipliers, it is appropriate to point out that the imposed incompressibility constraint would be extremely difficult to implement by means of static analysis (performed on a node by node basis).

Another advantage, already mentioned, is the relative ease with which the constraint can be implemented. This is largely due to the modelling approach employed in the thesis.

Recalling eqn. (3.10)

$$\sum_{i=1}^N (\phi^i \delta u_i + \lambda \frac{\partial g}{\partial u^i} \delta u_i) + g \delta \lambda = 0 \quad \dots (3.10)$$

It can be seen, from a careful examination of this equation and keeping in mind the derivations of section 3.2, that different types (different  $g$  functions) and numbers of constraint can be easily incorporated, thereby altering the nature of the problem being

considered. This renders the method of Lagrange Multipliers a simple, flexible and attractive alternative.

## 4. Time Dependent Problems and Dynamics

### 4.1 Introduction

In many engineering applications the inclusion of dynamic terms in the analysis of structures is essential. The term 'dynamic transient situations' implies that the inertia terms must be included in the equations of equilibrium. In certain instances widely different displacements are obtained from a quasi-static analysis as compared to a complete dynamic transient solution [Owen]. Examples where such inertia effects are important include the impact loading of structures due to collision, shock blast or explosive conditions, in which cases the loading is of high intensity and is applied for a short time period only. The type of structural loading can have a profound influence on the choice of numerical method most suitable for solution of the problem.

The transient response analysis of structural systems generally involves the discrete modelling of the structure by either finite element or finite difference methods and the solution of large, second-order differential equations by direct time integration techniques. Although finite difference solutions compare well in accuracy with those obtained by use of the finite element method, the difficulties encountered in problems with complicated geometries make the finite difference method unattractive.

Based on the general approach for solving nonlinear problems adopted in chapter 2, however, the resulting system of governing equations for the transient dynamic problem becomes

$$\mathbf{M} \ddot{\mathbf{u}}_1 + \phi_1(\mathbf{u}_1, \dot{\mathbf{u}}_1) = 0 \quad \dots (4.1)$$

in which the dots denote differentiation in time, and  $\mathbf{u}$  stands for a set of parameters

describing the displacements. Matrix  $\mathbf{M}$  is the structural mass matrix (generally independent of time or displacement). The information relevant to the applied or activating forces, the displacements and the velocity is contained in  $\Phi_i ( \mathbf{u}_i, \dot{\mathbf{u}}_i )$ .

The procedures for the numerical solution of eqn (4.1) can be divided into two classifications [Bathe and Wilson]: *direct integration* and *modal superposition* techniques. The direct integration methods can be further subdivided into *explicit* and *implicit* methods. Although the direct integration and modal superposition techniques may at first appear to be different, they are, in fact, closely related and the choice of method is a question of numerical efficiency. However, the modal superposition technique is used for linear problems only.

In the methods of directly integrating the general dynamic equation (4.1), assumptions are made about the variation of either the displacements or accelerations during small time intervals [Warburton]; e.g. it may be assumed that during a small interval the displacement is a cubic function of time or the acceleration varies linearly. With the aid of these assumptions the set of  $n$  second-order differential equations (4.1) is replaced by  $n$  simultaneous equations, in general; the solution of the latter gives the displacements at the end of the short time interval for known conditions at the beginning of the interval. Successive application of this procedure leads to a complete solution. A review of three of the methods used in implicit time integration is undertaken in the next sub-chapter.

## 4.2 Implicit Time Integration Methods:

In this section, three implicit time integration methods are described. In particular, the Houbolt, Wilson- $\theta$  and Newmark methods are presented.

### 4.2.1 The Houbolt Method:

The method is mainly of historical significance, but it was one of the first procedures developed for the computer analysis of aircraft dynamics [Argyris & Mlejnek].

It is based upon the following system of equations:

$$\mathbf{M} \ddot{\mathbf{u}}_{p+1} + \mathbf{C} \dot{\mathbf{u}}_{p+1} + \mathbf{K} \mathbf{u}_{p+1} = \mathbf{F}_{p+1} \quad \dots (4.2)$$

$$\ddot{\mathbf{u}}_{p+1} = \frac{1}{\Delta t^2} (2\mathbf{u}_{p+1} - 5\mathbf{u}_p + 4\mathbf{u}_{p-1} - \mathbf{u}_{p-2}) \quad \dots (4.3)$$

$$\dot{\mathbf{u}}_{p+1} = \frac{1}{6 \Delta t} (11\mathbf{u}_{p+1} - 18\mathbf{u}_p + 9\mathbf{u}_{p-1} - 2\mathbf{u}_{p-2}) \quad \dots (4.4)$$

where the subscript  $p+1$  denotes the time  $t_{p+1} = t_p + \Delta t$  for which a solution is sought. By substituting eqns. (4.3) and (4.4) into eqn. (4.2), an equation determining  $\mathbf{u}_{p+1}$  in terms of  $\mathbf{u}_p$ ,  $\mathbf{u}_{p-1}$  and  $\mathbf{u}_{p-2}$  is obtained. This algorithm describes a 3-step method of 2<sup>nd</sup> order. As in the case for all multi-step procedures, the Houbolt method requires an initialisation since at the starting point in time  $t_0$ , only  $\mathbf{u}_0$ ,  $\dot{\mathbf{u}}_0$  and  $\ddot{\mathbf{u}}_0$  are known (from the equation of motion). For a consistent application of eqns. (4.3) and (4.4), however,  $\mathbf{u}_1$  and  $\mathbf{u}_2$  are required. Thus, one should begin either with a single-step procedure or introduce some auxiliary equations. By exploiting the known initial displacement  $\mathbf{u}_0$  and velocity  $\dot{\mathbf{u}}_0$ , it

is not difficult to obtain the following starting values:

$$\ddot{\mathbf{u}}_0 = \mathbf{M}^{-1}[\mathbf{F} - \mathbf{C}\dot{\mathbf{u}}_0 - \mathbf{K}\mathbf{u}_0] \quad \dots (4.5)$$

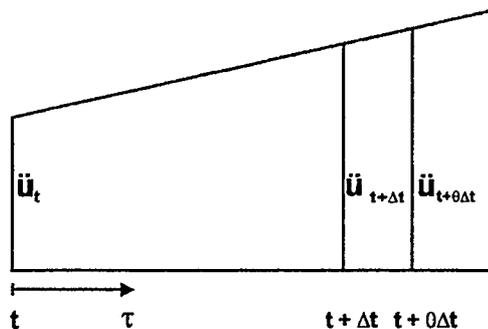
$$\mathbf{u}_{-1} = \Delta t^2 \ddot{\mathbf{u}}_0 + 2\mathbf{u}_0 - \mathbf{u}_1 \quad \dots (4.6)$$

$$\mathbf{u}_{-2} = 6\Delta t^2 \ddot{\mathbf{u}}_0 + 6\mathbf{u}_{-1} - 3\mathbf{u}_0 - 2\mathbf{u}_1 \quad \dots (4.7)$$

Following their substitution into equation (4.3) and (4.4), these equations clearly determine the unknown vectors  $\mathbf{u}_{-1}$  and  $\mathbf{u}_{-2}$  in terms of the known  $\mathbf{u}_0$  and  $\dot{\mathbf{u}}_0$ . The Houbolt method is unconditionally stable, exact to the 2<sup>nd</sup> order and obviously implicit. The algorithm possesses strong algorithmic damping which produces a stabilizing effect in some cases, but can also falsify the solution.

#### 4.2.2 The Wilson- $\theta$ Method:

This method is essentially an extension of the linear acceleration technique in which a linear variation of acceleration from time  $t_p$  to time  $t_{p+1}$  is assumed. In the Wilson method the acceleration is assumed to be linear from time  $t$  to time  $t + \theta\Delta t$ , where  $\theta \geq 1$ , as shown in figure 4.1



**Figure 4.1** Linear variation of acceleration.

For  $\theta = 1$  the method reduces to the linear acceleration scheme, but in this case the

method is only conditionally stable [Owen]. For unconditional stability  $\theta \geq 1.37$ . For any time  $\tau$  so that  $0 \leq \tau \leq \theta\Delta t$ , we have from figure 4.1 that

$$\ddot{\mathbf{u}}_{t+\tau} = \ddot{\mathbf{u}}_t + \frac{\tau}{\theta\Delta t} (\ddot{\mathbf{u}}_{t+\theta\Delta t} - \ddot{\mathbf{u}}_t) \quad \dots (4.8)$$

Integrating eqn. (4.8) gives

$$\dot{\mathbf{u}}_{t+\tau} = \dot{\mathbf{u}}_t + \ddot{\mathbf{u}}_t \tau + \frac{\tau^2}{2\theta\Delta t} (\ddot{\mathbf{u}}_{t+\theta\Delta t} - \ddot{\mathbf{u}}_t) \quad \dots (4.9)$$

and integrating again gives

$$\mathbf{u}_{t+\tau} = \mathbf{u}_t + \tau \dot{\mathbf{u}}_t + \frac{\tau^2}{2} \ddot{\mathbf{u}}_t + \frac{\tau^3}{6\theta\Delta t} (\ddot{\mathbf{u}}_{t+\theta\Delta t} - \ddot{\mathbf{u}}_t) \quad \dots (4.10)$$

For the particular time  $\tau = \theta\Delta t$ , eqns. (4.9) and (4.10) give

$$\dot{\mathbf{u}}_{t+\theta\Delta t} = \dot{\mathbf{u}}_t + \frac{\theta\Delta t}{2} (\ddot{\mathbf{u}}_{t+\theta\Delta t} + \ddot{\mathbf{u}}_t) \quad \dots (4.11)$$

$$\mathbf{u}_{t+\theta\Delta t} = \mathbf{u}_t + \theta\Delta t \dot{\mathbf{u}}_t + \frac{\theta^2\Delta t^2}{6} (\ddot{\mathbf{u}}_{t+\theta\Delta t} + 2\ddot{\mathbf{u}}_t) \quad \dots (4.12)$$

from which we can solve for  $\ddot{\mathbf{u}}_{t+\theta\Delta t}$  and  $\dot{\mathbf{u}}_{t+\theta\Delta t}$  in terms of  $\mathbf{u}_{t+\theta\Delta t}$  :

$$\ddot{\mathbf{u}}_{t+\theta\Delta t} = \frac{6}{\theta^2\Delta t^2} (\mathbf{u}_{t+\theta\Delta t} - \mathbf{u}_t) - \frac{6}{\theta\Delta t} \dot{\mathbf{u}}_t + 2\ddot{\mathbf{u}}_t \quad \dots (4.13)$$

$$\dot{\mathbf{u}}_{t+\theta\Delta t} = \frac{3}{\theta\Delta t} (\mathbf{u}_{t+\theta\Delta t} - \mathbf{u}_t) - 2\dot{\mathbf{u}}_t - \frac{\theta\Delta t}{2} \ddot{\mathbf{u}}_t \quad \dots (4.14)$$

To obtain the displacements, velocities and accelerations at time  $t + \Delta t$ , the equilibrium equations are considered at time  $t + \theta\Delta t$ . This requires the projection of the applied load vector to time  $t + \theta\Delta t$ , which is performed linearly as

$$\mathbf{F}_{t+\theta\Delta t} = \mathbf{F}_t + \theta (\mathbf{F}_{t+\Delta t} - \mathbf{F}_t)$$

Equation (4.1) then becomes

$$\mathbf{M} \ddot{\mathbf{u}}_{t+\theta\Delta t} + \phi_i(\mathbf{u}_{t+\theta\Delta t}, \dot{\mathbf{u}}_{t+\theta\Delta t}) = 0 \quad \dots (4.15)$$

Substitution of eqns. (4.13) and (4.14) into eqn. (4.15) gives a system of simultaneous equations which may be solved for  $\mathbf{u}_{t+\theta\Delta t}$ .

Substituting  $\mathbf{u}_{t+\theta\Delta t}$  into eqn. (4.13) gives  $\ddot{\mathbf{u}}_{t+\theta\Delta t}$ , which is then employed in eqns. (4.8)-(4.10), all evaluated at  $\tau = \Delta t$ , to give

$$\ddot{\mathbf{u}}_{t+\Delta t} = b_1 (\mathbf{u}_{t+\theta\Delta t} - \mathbf{u}_t) + b_2 \dot{\mathbf{u}}_t + b_3 \ddot{\mathbf{u}}_t \quad \dots (4.16)$$

$$\dot{\mathbf{u}}_{t+\Delta t} = \dot{\mathbf{u}}_t + b_4 (\ddot{\mathbf{u}}_{t+\Delta t} + \ddot{\mathbf{u}}_t) \quad \dots (4.17)$$

$$\mathbf{u}_{t+\Delta t} = \mathbf{u}_t + \Delta t \dot{\mathbf{u}}_t + b_5 (\ddot{\mathbf{u}}_{t+\Delta t} + 2\ddot{\mathbf{u}}_t) \quad \dots (4.18)$$

where

$$b_1 = \frac{6}{\theta^3 \Delta t^2}$$

$$b_2 = \frac{-6}{\theta^2 \Delta t}$$

$$b_3 = 1 - \frac{3}{\theta}$$

$$b_4 = \frac{\Delta t}{2}$$

$$b_5 = \frac{\Delta t^2}{6} \quad \dots (4.19 \text{ a - e})$$

It is noted from eqns. (4.16)-(4.18) that no special starting algorithm is required, since  $\mathbf{u}_{t+\Delta t}$ ,  $\dot{\mathbf{u}}_{t+\Delta t}$  and  $\ddot{\mathbf{u}}_{t+\Delta t}$  are all expressed in terms of the same quantities at time  $t$  only.

### 4.2.3 The Newmark Method:

This method is also an extension of the linear acceleration method. The following

assumptions are used:

$$\dot{\mathbf{u}}_{t+\Delta t} = \dot{\mathbf{u}}_t + [(1-\delta)\ddot{\mathbf{u}}_t + \delta\ddot{\mathbf{u}}_{t+\Delta t}]\Delta t \quad \dots (4.20)$$

$$\mathbf{u}_{t+\Delta t} = \mathbf{u}_t + \Delta t \dot{\mathbf{u}}_t + \left[ \left( \frac{1}{2} - \alpha \right) \ddot{\mathbf{u}}_t + \alpha \ddot{\mathbf{u}}_{t+\Delta t} \right] \Delta t \quad \dots (4.21)$$

where  $\alpha$  and  $\delta$  are parameters that can be determined to obtain integration accuracy and stability. When  $\delta = 1/2$  and  $\alpha = 1/6$ , this method reduces to the linear acceleration method [Owen]. Newmark originally proposed as an unconditionally stable scheme the constant average acceleration method, in which case  $\delta = 1/2$  and  $\alpha = 1/4$ . For a solution of displacements, velocities and accelerations at time  $t + \Delta t$ , the equilibrium equations (4.1) are also considered at time  $t + \Delta t$ :

$$\mathbf{M} \ddot{\mathbf{u}}_{t+\Delta t} + \phi_1(\mathbf{u}_{t+\Delta t}, \dot{\mathbf{u}}_{t+\Delta t}) = 0 \quad \dots (4.22)$$

An identical approach to the one considered in the Wilson method is adopted here. Eqn (4.21) is solved for  $\ddot{\mathbf{u}}_{t+\Delta t}$  in terms of  $\mathbf{u}_{t+\Delta t}$ . Substituting the result in eqn (4.20), a set of equations for  $\ddot{\mathbf{u}}_{t+\Delta t}$  and  $\dot{\mathbf{u}}_{t+\Delta t}$ , each in terms of the unknown displacement  $\mathbf{u}_{t+\Delta t}$ , is obtained. Substituting these expressions for  $\ddot{\mathbf{u}}_{t+\Delta t}$  and  $\dot{\mathbf{u}}_{t+\Delta t}$  into eqn. (4.22) gives a system of simultaneous equations which may be solved to give  $\mathbf{u}_{t+\Delta t}$ . By use of eqns. (4.20) and (4.21) all quantities  $t + \Delta t$  can be finally expressed as

$$\ddot{\mathbf{u}}_{t+\Delta t} = \mathbf{b}_1 (\mathbf{u}_{t+\Delta t} - \mathbf{u}_t) + \mathbf{b}_2 \dot{\mathbf{u}}_t + \mathbf{b}_3 \ddot{\mathbf{u}}_t \quad \dots (4.23)$$

$$\dot{\mathbf{u}}_{t+\Delta t} = \dot{\mathbf{u}}_t + \mathbf{b}_4 \ddot{\mathbf{u}}_t + \mathbf{b}_5 \ddot{\mathbf{u}}_{t+\Delta t} \quad \dots (4.24)$$

$$\mathbf{u}_{t+\Delta t} = \mathbf{u}_t + (\mathbf{u}_{t+\Delta t} - \mathbf{u}_t) \quad \dots (4.25)$$

where

$$b_1 = \frac{1}{\alpha \Delta t^2}$$

$$b_2 = \frac{-1}{\alpha \Delta t}$$

$$b_3 = -\left(\frac{1}{2\alpha} - 1\right)$$

$$b_4 = \Delta t (1 - \delta)$$

$$b_5 = \delta \Delta t \quad \dots (4.26 \text{ a-e})$$

It should be noted that the numerical time integration algorithms for the Newmark and the Wilson methods are identical, provided that the appropriate values of the constants  $b_1$  -  $b_5$  are employed: (eqns (4.26 a-e) and (4.19 a-e)). Thus, both methods can be easily incorporated in the same computer code.

In the context of this thesis, the Wilson method has been used. The implementation of this method is discussed in the next section.

### 4.3 Implementation of the Wilson Method:

The implementation of the Wilson method is dealt with in this section. All necessary modifications to the computer program are discussed in Chapter 5.

Recall eqn. (2.24 a)

$$\sum_{i=1}^N \phi^i(\mathbf{u}_1, \mathbf{u}_2, \dots, \mathbf{u}_N) \cdot \delta \mathbf{u}_i = 0$$

which is an expression for the virtual work in terms of the known initial geometry. The introduction of the dynamic terms requires several modifications to the assembly of eqn. (2.24 a).

Recall eqn (4.1)

$$\mathbf{M} \ddot{\mathbf{u}}_1 + \phi_1(\mathbf{u}_1, \dot{\mathbf{u}}_1) = \mathbf{0}$$

It can be clearly seen that a representation of the mass properties is needed. The mass matrix is formulated using a lumped mass approach instead of a distributed mass approach. In assembling a lumped mass matrix, the mass of each bar of the system being considered is calculated from its density and volume. One half of the calculated mass is then assigned to each node of that bar. e.g.  $m_1 = \frac{1}{2}\rho V_1$ ,

where  $V_1 = \text{Cross Sectional Area of bar 1} * \text{length of bar 1}$ .

(All bars have the same density and this remains constant throughout.)

The equations developed in section 4.2.2 are applied in the manner described in that section. These serve to modify eqns (2.24 a) into the following form:

$$\sum_{i=1}^N \phi_i(\mathbf{u}, \dot{\mathbf{u}}, \ddot{\mathbf{u}}) \cdot \delta \mathbf{u}_i = 0 \quad \dots (4.27)$$

where  $\mathbf{u}$  is the vector of unknown displacements

$\dot{\mathbf{u}}$  is a function of  $\mathbf{u}$

$\ddot{\mathbf{u}}$  is also a function of  $\mathbf{u}$

Equation (4.27) which is equivalent to eqn (4.15), developed in section 4.2.2, may be solved by the Newton-Raphson method described in chapter 2.

## **5. Computer Implementation of Theory**

The implementation of the theory developed in chapters 2, 3 and 4 will be dealt with in this section; this will be divided into two sub-sections. In the first, the relevant equations that need to be solved by the Newton-Raphson method are implemented. The latter half deals with the program's input and output.

A BORLAND C++ 3.0 compiler and its integrated environment provides the platform on which the program is developed. The program has been developed to run in a DOS environment and requires a math co-processor. The "C" programming language, hereafter referred to as just "C", is used to implement the necessary code. This choice is greatly influenced by the current popularity of "C" over other languages (e.g. FORTRAN or BASIC). It is appropriate to point out that in "C", the term function refers to what is commonly called a subroutine. In the context of this chapter, the term function will be used in this sense instead of its mathematical sense.

A complete listing of the source code of one working version is given in appendix A. Listings of the function calls in the main section and those related to the Newton method are given in tables 5.1 and 5.2 respectively.

### **5.1 Equation Implementation:**

The implementation of the equations of P.V.W., developed in chapter 2, will form the core of this sub-section. The necessary modifications to the program due to the constraint and the dynamic terms introduced in chapters 3 and 4 are also discussed.

---

```

main()
{
    .
    Program Inputs;    /* A set of function calls to handle the program input */
    .
    for( time = start to duration; increment dt )
    {
    history(time, duration);    /* function that returns a scaling parameter */
    .
    mnewt();            /* Newton-Raphson function call */
    }
    .
    Program Output;    /* A set of function calls to handle the program output */
}

```

---

**Table 5.1** Skeleton listing of program highlighting functions.

### 5.1.1 Virtual Work Implementation:

Recall eqns. (2.24 a) and (2.24 b) which are the equations of virtual work and equilibrium equations respectively.

$$\sum_{i=1}^N \phi_i(\mathbf{u}_1, \mathbf{u}_2, \dots, \mathbf{u}_N) \cdot \delta \mathbf{u}_i = 0$$

$$\phi_i(\mathbf{u}_1, \mathbf{u}_2, \dots, \mathbf{u}_N) = 0 \quad i = 1, 2, \dots, N$$

Also recall eqn. (2.27)

$$\sum_{j=1}^N a_{ij} \Delta u_j = b_i \quad i = 1, 2, \dots, N$$

The Newton-Raphson method requires the user to supply a matrix of coefficients  $\mathbf{a}_{ij}$  and a vector  $\mathbf{b}_i$ . In essence these correspond to the matrix of the partial derivatives of the

equilibrium equations and the numerical values of the equilibrium equations respectively.

---

```

mnewt ( )
{
    for ( K = 1 to # of iterations; increment 1 )
    {
        callfunc ( ) /* function which supplies  $\mathbf{a}_{ij}$  and  $\mathbf{b}_i$  */
        ludcmp ( ) /* LU decomposition */
        lubksb ( ) /* LU backsubstitution */
    }
}

callfunc ( )
{
    for ( i=1 to N; increment 1 )
    {
        vw ( ) /* call the virtual work function */
    }
}

vw ( )
{
    eij ( ) /* call the strain function */
    kij ( ) /* call the constitutive equation function */
}

```

---

**Table 5.2** Listing of some function calls from mnewt().

As can be seen from Table 5.2, the quantities  $\mathbf{a}_{ij}$  and  $\mathbf{b}_i$  are assembled in callfunc(). This is done by repeatedly calling the function vw(). This function calculates the virtual work for each node based on eqn (2.24 a).

It is possible to obtain the value of any  $\mathbf{b}_i$  by evaluating it with a convenient combination of  $\delta u$  's and all the  $\mathbf{u}$  values at that step. e.g. for a case where  $N = 4$ , the value of  $\mathbf{b}_1$  would be obtained by setting  $\delta u_1 = 1$  with all other  $\delta u$  's = 0. The function  $vw()$  is then called with these  $\mathbf{u}$  and  $\delta u$  values. Similarly, the quantity  $\mathbf{a}_{ij}$  is obtained by a convenient combination of  $\mathbf{u}$  and  $\delta u$  values. In this case, however, both values of  $\mathbf{u}$  and  $\delta u$  are varied sequentially. This is equivalent to a numerical partial differentiation. Specifically, a central difference differentiation technique is used; the average of a forward and backward differentiation is taken to give each value of  $\mathbf{a}_{ij}$ . The calculated values are returned to  $mnewt()$  and solved by LU decomposition and backsubstitution.

It is apparent that this method is computationally expensive, but the simplicity of its implementation offers a great advantage. Its flexibility will also become obvious when the necessary modifications due to the constraint and dynamic terms are shown.

### 5.1.2 Constraint Implementation:

The constraint that will be implemented in this section is the incompressibility constraint developed in chapter 3.4. Since the modifications to the program occur in  $callfunc$ , a pertinent modified function listing will be provided in table 5.3.

The values of  $\mathbf{a}_{ij}$  and  $\mathbf{b}_i$  are obtained in a similar manner as described in the preceding section. The equilibrium equations are now eqns. (3.10) and the vector of unknowns is increased by one entry for every constraint. Since this implementation would be extremely wasteful (computationally expensive), the form of the Jacobian derived in chapter 3.4 will be implemented instead.

---

```

callfunc()
{
    for( w = 1 to # of constraints; increment w )
    {
        g(); /* call the constraint */
    }

    for( i = 1 to N, increment i )
    {
        vw(); /* call vw */
    }
}
vw()
{
    eij(); /* call strain function */
    kij(); /* call constitutive relation */
    .
    vw = vw + dg; /*modify vw by the partial derivative of the constraint */
}

```

---

**Table 5.3** Modifications to function calls due to constraint.

The exact location of each constraint equation entry in the matrix  $\mathbf{a}_{ij}$  is known. A numerical differentiation is thus possible with minimal changes to the function, as shown in table 5.3. The price to pay is the extra parameter that has to be passed to `vw()`. No other modifications are required, however.

### 5.1.3 Dynamic Implementation:

The modifications that are associated with the introduction of a dynamic term are slightly more complex. They occur at all stages of the program, and add correction terms at each stage. This results in the following changes to tables 5.1 and 5.3.

In the case of an isometric muscle contraction, there are no external forces present.

---

```

main()
{
    Program Inputs;    /* A set of functions that handles program inputs */
    Mass;              /* Assemble the lumped mass matrix */
    for( time = start to duration; increment dt )
    {
        history();
        mnewt();
        Back Projection; /* application of eqns. 4.16 - 4.19 */
    }
    Program Outputs();
}
mnewt();
{
    for( k = 1 to # of iterations; increment k )
    {
        callfunc()
        Solve;          /* solve by LU decomposition */
    }
}
callfunc()
{
    for( w = 1 to # of constraints; increment w )
    {
        g();           /* call constraint function */
    }
    for( i = 1 to N; increment i )
    {
        vw();         /* call virtual work function */
    }
}
vw()
{
    Forward Projection; /* application of eqns. 4.13 - 4.14 */
    eij();              /* call strain function */
    kij();              /* call constitutive equation function */
    .
    vw - (m.a)*du      /* Subtract inertia term */
}

```

---

**Table 5.4** Listing of some function calls relevant to the dynamic terms.

Hence, the external virtual work consists of merely the inertia term (mass\*acceleration\*du).

#### **5.1.4 Adaptation for Muscle Behaviour:**

It is a relatively simple matter to change the constitutive equation in the  $k_{ij}()$  function. This allows the study of different types of materials. Specifically, the force-length profile employed will be discussed in the next chapter. It is of the form shown below,

$$F_f(L, V) = F_f(L) F_f(V)$$

where  $F_f(L)$  represents the isometric force output of a fibre, and  $F_f(V)$  its force - velocity relationship during concentric contraction.

#### **5.1.5 Discussion:**

It is apparent that the approach adopted in this thesis is extremely simple and offers a great deal of flexibility. Different constraint equations can be easily implemented, thereby altering the nature of the problem being examined. It would be just as simple to change the constitutive equation, to study the behaviour of different types of material.

These advantages are not readily available if one was to use a commercial finite element package. In particular, in the study of muscles, (large deformation, large strain active material) there are hardly any elements that provide such capabilities.

## 5.2 Program Input:

A brief description of the functions that handle the program inputs will be undertaken in this section.

"readinput ( )" : An interactive function allowing the user to impose a displacement or a velocity on a node in a specific direction at run time. This feature is useful since all displacements are normally reset to zero at the start of each run. In most cases, an initial zero guess is sufficient. However, the number of iterations is greatly reduced when a sensible initial guess is chosen.

"innodes ( )" : This function can be run interactively or can allow the user to read in prepared data files. The nodal information of the structure is input at this point. The properties of the connecting bars are input and the dimension of length is calculated. These properties include the diameter, the depth, node connections and the material type. The information is stored in the files 'nodes.dat' and 'connect.dat'.

"support ( )" : The user is able to read in a prepared data file or use the function interactively. A file titled 'support.dat' contains the prepared information. The nodal supports of the structure are input at this stage.

"forces ( )" : The function allows user interaction as well as reading in a prepared data file. The information is stored in 'force.dat' and contains all the nodal force data. For the example of isometric muscle contraction, the value of all the forces (external) is set to zero.

"foptimum ( )" : At the start of every experiment, the user is prompted to enter the

optimum muscle length. This enables the calculation of the optimum length of each of the fibre bundles.

A sample of the input data files required for one run is given in appendix B.

### **5.3 Program Output:**

Several output files are generated for every run. These files cover the displacement, velocity, acceleration, force observed in the tendon, relative changes in the fibre lengths and the relative changes in angles of pinnation<sup>1</sup>. All the output files are generated with a \*.\$\$\$ extension.

---

<sup>1</sup>The definition of the angle of pinnation differs from that used in bio-mechanics. Here it refers to the relative angle between the fibre bundle and the tendon sheath.

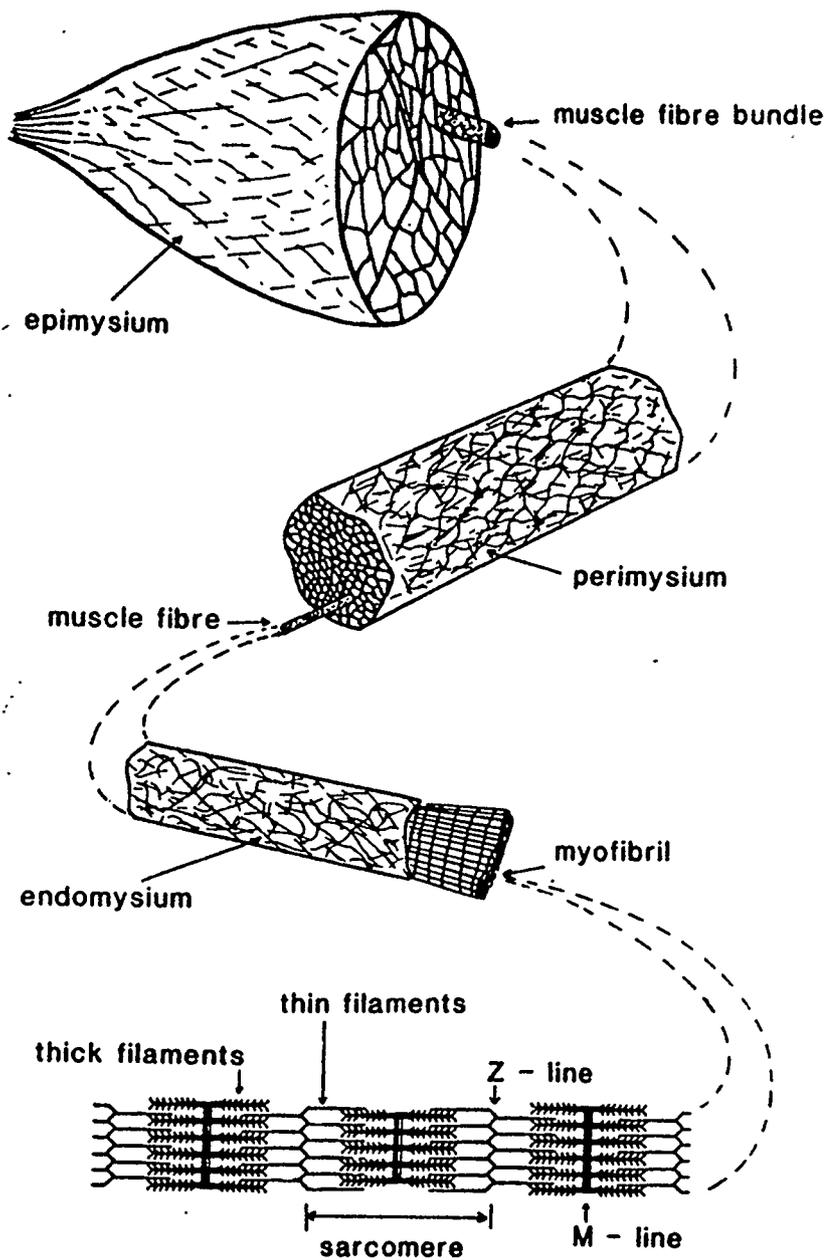
## 6. An Introduction to Skeletal Muscle

### 6.1 The structure of skeletal muscle:

Muscles are the components of the body which are capable of active contraction and they are sometimes referred to as force generators. There are three types of muscles which can be readily identified: skeletal, heart and smooth muscle. The voluntary, skeletal or striated muscles perform tasks as diverse as maintaining the posture of the spine, producing rapid and powerful movements of the limbs and executing the precise and finely judged movement of the eye [Purslow & Duance].

An appreciation of the structure-function relationships is clearly of great importance in understanding how a striated muscle works to fulfil its required functions.

In general, the structure of vertebrate striated muscle is as indicated in figure 6.1. Much of the description that follows is also given by Bagshaw [1982], with a more detailed review given by Schmalbruch [1985]. Muscle is a hierarchical structure, comprising many identical subunits that are aggregated to form the basic unit from which the next level of structure is then built. Surrounding the whole muscle there is a connective tissue sheath, the *epimysium*, which is continuous with the tendons. The whole muscle is divided internally into bundles of muscle fibres that run along its length, each surrounded by the next connective tissue structure, the *perimysium*. The perimysia of adjacent bundles merge to form a continuous network across the muscle, which connects to the epimysium at the surface of the muscle at irregular intervals. Muscle fibre bundles are typically 1-10 mm across. They tend, like the muscle fibres comprising them, to be irregularly polygonal in cross-section rather than circular. Each bundle contains many



**Figure 6.1** The general structure of striated muscle. Adapted from Jolley and Purslow (1989).

individual muscle fibres, each one ensheathed in its own connective tissue envelope, the *endomysium*. Each muscle fibre is a giant, multi-nucleated cell, which runs the entire length of the muscle, and is typically 10-100  $\mu\text{m}$  in diameter.

The cell cytoplasm is arranged into strands of myofibrils, each having a diameter of  $1\mu\text{m}$ . The myofibrils can be subdivided further into myofilaments which, in turn, are composed of sarcomeres, each of which has a length of approximately  $2.5\mu\text{m}$ . It is the structure and regular arrangement of the sarcomeres that gives the myofibrils and, in fact, skeletal muscle fibres their striated appearance; an individual sarcomere extends from the middle of one light region, called Z-Band, to the next within the dark-light striation pattern of the myofibrils. Sarcomeres constitute the smallest contractile unit within a muscle. They are composed of interdigitating thick ( $12\text{nm}$  diameter) and thin ( $5\text{nm}$  diameter) filaments bounded by Z-Bands. It is the action of thick and thin filaments sliding past each other that is responsible for the contraction of muscle tissue. This contractile tendency and associated force generation under stimulation is explained by the theory of cross-bridges [Purslow & Duance].

The unit of muscular activity is the single twitch, produced in response to a nerve impulse or an electric shock. The ordinary picture of a muscle twitch shows a gradual rising phase, a blunt maximum and gradual relaxation. By repeating the stimulus at a sufficient frequency, the resulting twitches can be fused into an apparently smooth maintained contraction, which begins to decline soon after the last stimulus [Hill, 1950]. In general, the fibre force increases along with the stimulation frequency, but it does not increase beyond a certain maximum value.

## **6.2 Muscle Models:**

Muscle models will be grouped into functional muscle models and geometrical muscle models. Only a brief review of functional models will be undertaken. The review will concentrate on geometrical models instead, since the model developed in this thesis is essentially geometrical in nature. Some of the review that follows is also given by Anton [1991].

Functional muscle models consist of a number of equations which are obtained from statistical modelling of experimental data. These models, for example, give a description of the force-length or the force-velocity relationships. Due to the lack of a well founded theoretical basis, these models often contain free parameters, which are available for adjustment when the models are used. These free parameters are then adjusted, often through optimization procedures, to have the muscle behave in a predetermined manner.

Geometrical muscle models, on the other hand, take account of the muscle geometry in deriving the equations which describe the muscle behaviour, and they attempt to explain differing muscle geometries on muscle performance. In general, they depend on relations describing the muscle fibre behaviour, where these relations are often related to the functional muscle models above, but which are now specific to fibres.

### **6.2.1 Functional Muscle Models:**

Functional muscle models are commonly employed for investigations akin to the musculo-skeletal system than to questions about the inner workings of muscles. In their

purest form, these models intentionally ignore any detailed current knowledge about muscle behaviour. They rather centre on a specific actual or perceived functional role of muscles in the musculo-skeletal system. Their mathematical and/or physical description is based on this functional role.

Most functional muscle models are derivations of Hill's Equation or Hill's Three Element Model of muscular contraction. Hill [1938] derived an empirical equation for tetanized (frog sartorius) muscle, which expresses a hyperbolic relation between the muscular contractile velocity and the applied load. Hill's investigation was restricted to muscle optimal length, but this was later generalized by Abbott and Wilkie [1953] to varying lengths. The maximum isometric force was shown to be dependent on the muscle length. Hill also proposed a mechanical model which was intended to "visualize" the basic features of his empirical equation. The model consisted of a contractile element in series with an elastic element. In more recent times, the model has been extended and its elements have been related to the Sliding Filament Theory. The most basic extension to Hill's original model is the addition of a parallel elastic element, and the extended model is usually referred to as Hill's Three-Element Model [Fung, 1970]. In this model, the series elastic element is commonly associated with the intrinsic elasticity of the actin and myosin molecules and cross-bridges, while the parallel elastic element is related to connective tissues, cell membrane, etc.. Often a viscous element is added in parallel to Hill's Three Element Model in order to give the model the velocity dependence of Hill's Equation when the active element is described as being independent of the shortening velocity. However, Hill [1938], [1950] and [1970] objected to the interpretation which

attributed the lower fibre tensions for higher shortening velocities to viscous effects.

Herzog [1987] used a modification of the original Hill's Equation to estimate individual muscle forces in situations where different muscles are acting together in various activities. The modifications consisted in adding further parameters which describe the muscle's angle of pinnation, its state of activation, physiological cross sectional area and muscle force constant, i.e. the maximum isometric force a muscle can exert per unit physiological cross sectional area.

### **6.2.2 Geometrical Muscle Models:**

Geometrical muscle models date as far back as the early 17<sup>th</sup> Century. The Danish scientist Niels Stensen (1638-1686) was first in formulating a mechanical model of muscular contraction, using Euclidean geometry. Stensen stated that the muscles consist of pinnate structures, each of which is composed of equally long muscle fibres forming a parallelepipedon between parallel tendons, and that muscle fibres are separated by transverse bands of tissue connected with the membrane wrapping the muscle [Kardel, 1990]. With his model, Stensen was able to demonstrate that muscles could contract without changing their volume. This concept was too remote from the generally held belief - handed down from antiquity with great authority by René Descartes (1596-1650) in *De Homine* - that a substance, the 'animal spirit', entered from the brain through hollow nerves to make the muscles swell and contract [Kardel, 1990].

Benninghoff and Rollhäuser [1952] produced a paper in which they kept constant the volume of the modelled pinnate muscle by leaving the surfaces of insertion of the

fibres unchanged. Based on trigonometric consideration of individual fibre deformations, the statement was made that the maximal economical angle of fibre pinnation would be 30 degrees, if the muscle fibres shortened by half their original lengths.

Hatze [1978] and Gans [1982] have both produced models in which the volume is kept constant.

Heukelom et al. [1979] investigated the relationship between the structure of muscle and its function, especially with regard to the influence of the internal pressure when a muscle contracts. They estimated the internal pressure in pinnate muscles at 10 kPa.

The most widely accepted geometrical muscle model in the literature today is the one by Woittiez et al [1984]. The model allows the construction of complex three-dimensional muscles with a wide variety of architecture. Specifically, he uses a geometry which consists of two kite shaped tendon sheaths (with opposite geometrical orientation at the top and bottom) which in general are not of equal size and are not parallel. The fibres are allowed to have varying angles of pinnation. The muscle volume, kept constant throughout, is divided into segments for which, at different muscle lengths, muscle fibre forces, shortening velocities, maximal power etc. are calculated. The segmental fibre forces are added after correcting for the angle of pinnation at different muscle lengths and result in the length dependent total muscle force. Woittiez reports excellent agreement between the model generated muscle force-length and force-velocity relations and those obtained by experiments on Wistar rats.

Otten [1988] presented a muscle model which accounts for fibre curvature, tendon

elasticity and internal muscle pressure. The model consists essentially of six sub-units in a pinnate geometrical arrangement, where each unit is formulated as a Hill type model. The tendon sheaths tie the ends of the sub-units together at either side. Comparing his model predictions to experiments performed on cat vastus lateralis, Otten obtains good agreement. He also observes that the inclusion of tendon elasticity in the model shifts the muscle force length curves to higher lengths.

Anton [1991], developed a model based on a physical principle. He used the principle of virtual work to derive the equations of equilibrium. In his thesis, he reports a good agreement between his results from the straight line model and Woittiez's results. In the same thesis, he also developed another model based on a constitutive description of muscle tissue and the theory of deformable continua. He called this a continuum model.

## **7. Dynamic Muscle Model**

A two dimensional Dynamic Muscle Model (DMM) of uni-pinnate muscles is developed in this chapter. The model is built on a physical foundation. This is achieved by using the principle of virtual work. In the context of the current model, the principle equates the incremental internal energy change to the incremental work performed by the muscle force, for any incremental deformation of the muscle consistent with the kinematic constraints. It should be restated at this point that the model is essentially qualitative. Consequently, the force-length, and force-velocity relationships used will be the ones most widely accepted in the literature.

The underlying assumptions and simplifications are similar to those encountered in the treatment by Woittiez et al. [1984] and by Anton [1991]. Due to the lack of a general constitutive theory of muscle tissue these assumptions are required in order to reduce the number of degrees of freedom inherent to the model and to guarantee a solution. These will determine to a large extent how the final model will perform, and they therefore, constitute an important part of the model.

### **7.1 Model Assumptions and Simplifications:**

Several assumptions are made which have to be clearly stated at the outset. Some of these appear to be unnatural and contradict physical and experimental evidence.

The first assumption and simplification describes the tendinous sheath as having a tensile stiffness and assigns to it a constant modulus of elasticity. Fibres, while they are allowed to contract or elongate freely, are taken to be stiff in bending. These

simplifications are counter to physiological evidence. As an aside, the mechanical properties of the tendinous sheath are assumed to change as it de-crimps and the network of collagen fibres re-orientates [Purslow, 1989].

The tendinous sheaths are allowed to deform in length as a result of the above discussion. This assumption will be discussed in more detail in the context of the kinematic constraint of constant volume.

The volume of entire muscle and of any volumetric element of muscle tissue remains unchanged, that is, the muscle tissue is incompressible. This incompressibility of muscles under stimulation has been shown by Abbott and Baskin [1962].

The assumptions concerning muscle fibre function will be discussed later in this chapter. For now though, fibres are assumed to extend over the entire distance between the two tendon sheaths and to possess a thickness which is small compared to the dimensions of the muscle. They are also assumed to have identical histo-chemical and physiological characteristics, in contrast to reality.

The only origin of forces is seen in the active and passive fibre force-length property. These forces act in the fibre direction only.

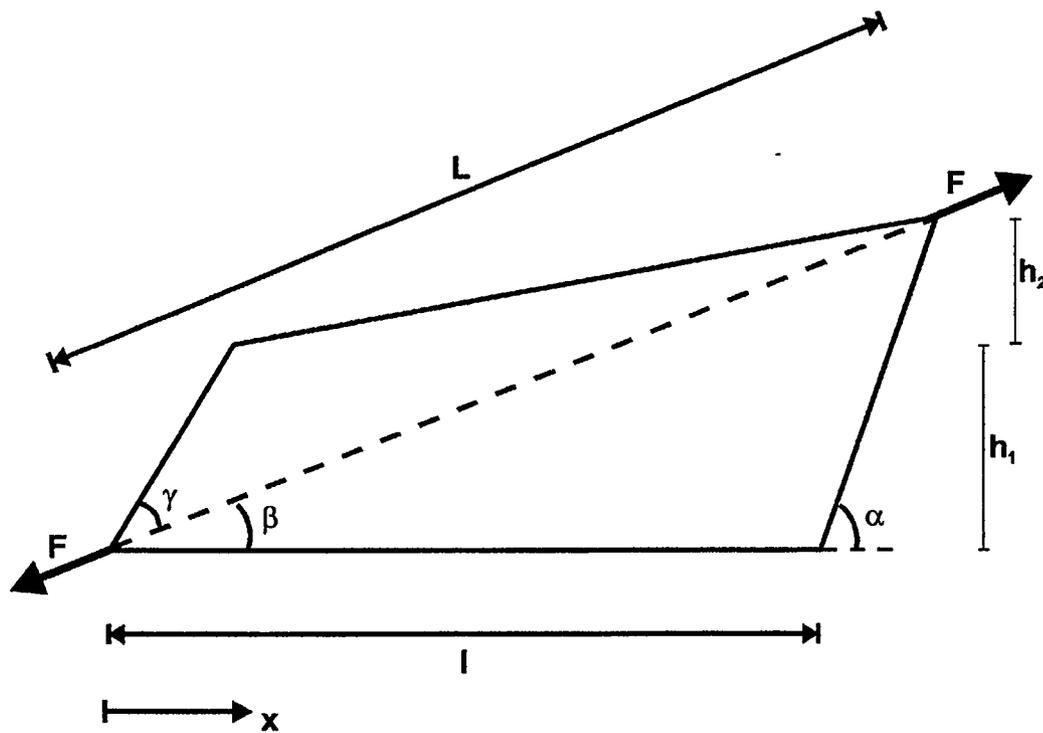
The fibres are initially stimulated by the same amounts, but are then free to contract in a manner consistent with the imposed kinematic constraint.

The muscle geometry that can be considered has to be limited only by the anatomical considerations. Caution should be exercised, since it would be easy to conceive an unrealistic muscle geometry and still obtain some solution.

The geometry of DMM will be considered in the next section. A close comparison

will be made with the model presented by Woittiez et al. [1984] and the one derived by Anton [1991]. These are the only two models encountered which take the broad approach adopted here. The work by Anton [1991] provides an excellent model, the Straight Line Model, which is based on a physical principle but it does not consider the velocity dependence.

## 7.2 Dynamic Muscle Model Geometry and Kinematics:



**Figure 7.1** Trapezoidal Muscle Geometry

The general muscle, whose cross section takes on a trapezoidal shape, is displayed in figure 7.1. The parameter  $h_2$  can be varied to make the geometry more anatomically

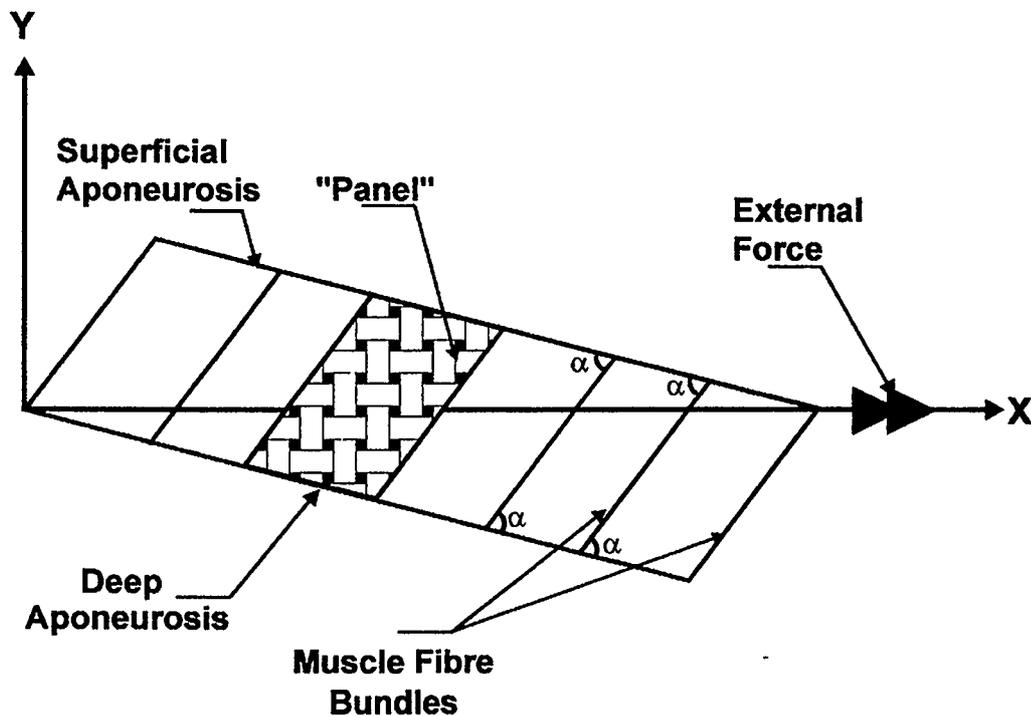
realistic. The only assumption made about the depth or muscle thickness, measured perpendicularly to the drawing plane, is that at any point along the length of the muscle,  $x$ , the depth of the top tendon sheath and the bottom one are identical. In other words, the volume of each segment is essentially the volume of a cube. The corresponding surface area, averaged for the top and bottom sheaths, of each segment is used to obtain the diameter of each fibre bundle; the width distribution of the tendon sheath, from which the area of each segment is calculated, determines the diameter of each fibre bundle.

Woittiez et al. [1984] used kite-like shapes to represent the tendon sheath geometries with the kites at the top and bottom having an opposite orientation. In their calculations of the volume, the trapezium is transformed into a rectangle by averaging the widths of the top and bottom tendon sheaths for each cross section. Consequently, the approach taken in the modelling of the tendon plate geometry in the DMM can be considered realistic.

Turning our attention to the kinematic constraint, the incompressibility constraint, derived in chapter 3, some of the definitions need to be clarified.

The angle of pinnation, referring to figure 7.1, is defined as the parameter  $\alpha$ . This is in agreement with Otten [1988] and Anton [1991] but in contrast with Woittiez [1984] who uses  $\gamma$  for this purpose. The choice of the angle  $\alpha$  will be shown to simplify the implementation of the incompressibility constraint.

The kinematic constraint is shown in the context of a typical muscle geometry in figure 7.2.



**Figure 7.2** Typical muscle geometry subjected to incompressibility constraint.

In the context of this model, the average of the four equivalent angles,  $\alpha$ , is taken as the angle of pinnation. In calculating the volume of each segment, the average of the two tendon sheath lengths and the average length of the adjacent fibre bundles is used. This is repeated for each volume segment, until the whole muscle is examined.

Each of the fibre bundles shown in figure 7.2 is assumed to be a "balloon" type structure, introduced in chapter 3. They are, in reality, fibre bundles which keep their volume constant and remain in contact at all times. The cross sectional area of each bundle is obtained from the method described above .

In the implementation of the incompressibility constraint, there is a fundamental

difference which needs to be highlighted. Considering a segment of muscle tissue, the current model allows the tendon plate to stretch, this being an integral part of the constraint. This feature is not present in the models of Woittiez [1984] and Anton [1991]. It is thought that the effect of straightening out of the crimps in the tendon sheath only has an effect at the initial stages of stimulation.

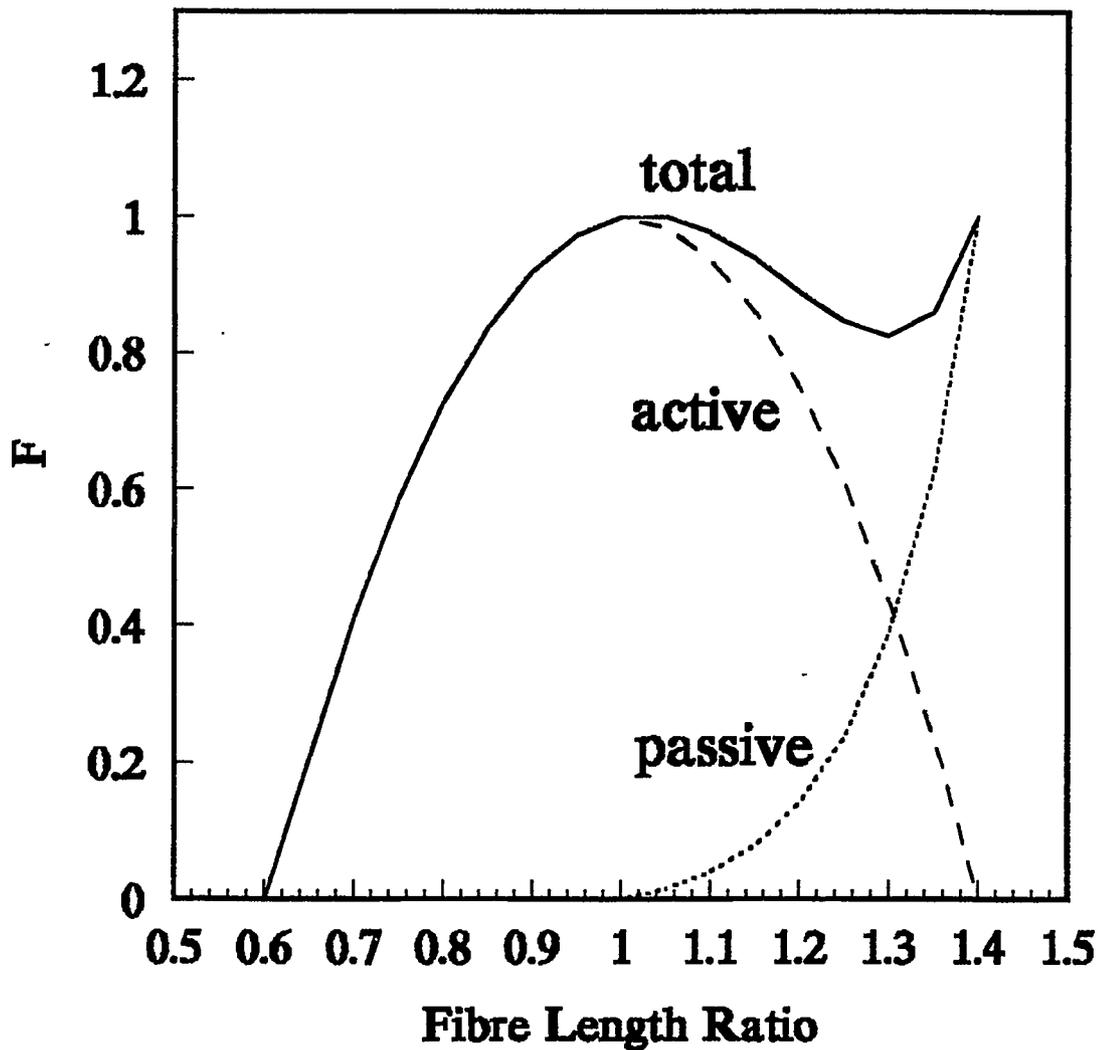
### 7.3 Fibre Force-Length Relation:

Experimental evidence indicates that muscle fibre forces depend on a fibre's length, the amount of stimulation and the time rate of the fibre length changes.

Muscle fibre force-length curves have been established experimentally by bringing individual muscle fibres to different absolute lengths and measuring the fibre forces with and without tetanic stimulation. Figure 7.2 displays the general form of measured fibre forces under stimulation (solid line - total muscle fibre force) and without stimulation (dotted line - passive muscle fibre force). The difference in force between the total and passive muscle fibre force is commonly called the active fibre force (dashed line). The absolute fibre length at which fibres reach their maximum active force is called optimal length. An unstimulated fibre left to itself will take on a length, called resting length, which is close or equal to its optimal length.

For this model, the fibre force-length relationship will be implemented in two stages: the active force profile will be taken from Woittiez [1984] and the passive force profile from Anton [1991].

$$f_{\text{active}} = ( -6.25 \lambda^2 + 12.5 \lambda - 5.25 )$$



**Figure 7.3** Fibre Force-Length Profile.

$$f_{\text{passive}} = 3.289 \cdot 10^{-6} e^{9.037 \lambda} - 0.02766$$

$$f(\lambda) = f_{\text{active}} + f_{\text{passive}}$$

The active force profile can be explained in terms of sarcomere behaviour, if we assume that the sarcomeres, linked in series within fibres, are identical.

A review by Purslow and Duan [1989] concludes that the epimysium,

perimysium and endomysium all contribute significantly towards the passive elasticity. This could be used as an explanation for the passive force profile.

#### 7.4 Fibre Force-Velocity Relation:

The muscle is assumed to behave according to an adaptation of Hill's [1938] equation:

$$(F + a) (v + b) = (gF_0 + a)b \quad \dots (7.1)$$

where

$$a/F_0 = b/v_0 = 0.25 \quad \dots (7.2)$$

and  $a$  and  $b$  are Hill's thermodynamic constants.  $F_0$  is the maximal active isometric force at optimal length,  $v_0$  is the maximal velocity of muscle shortening and  $g$  represents the maximal isometric force which a muscle can exert as a function of its length and is normalized to  $F_0$ .

The parameter  $g$  mentioned here is equivalent to the active force profile discussed in the preceding section.

$$g = ( -6.25 \lambda + 12.5 \lambda - 5.25 ) \quad \dots (7.3)$$

For ease of computer implementation, eqn. 7.1 and eqn 7.2 are combined to give a force-velocity profile:

$$F_f(v) = 2 \quad v / v_{\max} \leq -1$$

$$F_f(v) = 2 - ( 1 - v / v_{\max} ) / ( 1 + 4v / v_{\max} ) \quad -1 \leq v / v_{\max} \leq 0$$

$$F_f(v) = ( 1 + v / v_{\max} ) / ( 1 - 4v / v_{\max} ) \quad 0 \leq v / v_{\max} \leq 1$$

$$F_f(v) = 0 \quad v / v_{\max} > 1$$

This type of approximation is justified since the model is qualitative at this stage of development.

Previous models have assumed that either muscle length and velocity were independent of each other (Zajac, 1989) or that velocity was dependent upon muscle length (Pierrynowski and Morrison, 1985). The suitability of separating the length and velocity functions was recently tested by Rindos [1988] where cat plantar flexor muscle length and the velocity were controlled. Differences between predicted forces output using independent muscle length and velocity functions and the experimental data rarely exceeded 7% of maximal force. (The discussion was taken from a paper by Scott and Winter [1991]. Please refer to this paper for references of original publications by the authors mentioned above.)

The present DMM assumes no interdependence between muscle length and velocity thus the total fibre force,

$$F_f(L,V) = F_f(L) \cdot F_f(V) \quad \dots (7.4)$$

## 7.5 Discussion:

Anton [1991], in his derivation of the straight line model, has shown that the model behaviour is to a large extent indifferent to the absolute muscle geometry. This implies that a simple geometry (rectangular with two triangles at each end) will be representative of other geometries. Such a simplification lends credence to the tests that have been done qualitatively, which are the subject of the next chapter. The purpose of those tests is twofold: the first is to show that the model is consistent and the second is

to highlight its capabilities. It should be noted that the model has also been tested on realistic muscle geometries. e.g. the cat medial gastrocnemius can be modelled using parallel tendon sheaths and a uniform depth.

Another useful feature of the model which has not yet been mentioned is the activation parameter. This serves to modify eqn. 7.4 into the following form

$$F(a, L, v) = a \cdot F_f(L) \cdot F_f(v)$$

where ' $a$ ' is the activation parameter and serves as a scaling factor. The parameter ' $a$ ' can follow different time histories and this will also be exploited in the next chapter.

The inertia term has been incorporated into the model. The purpose of this is to show that it has a negligible effect, or otherwise, on the total deformation of the whole muscle during isometric contraction. Again the tests are included in the next chapter. It is envisioned that the inertia term would have a significant effect in a case where the fully stimulated muscle is suddenly subjected to an imposed displacement. This, however, is beyond the scope of the thesis and forms one of the recommendations for further work.

The equilibrium equations, while they are not directly mentioned in this chapter, have been derived from physical principles in chapters 3 and 4. It is of interest that Woittiez [1984] treats this topic in one sentence, saying that "the product of the force of each fibre and the cosine of its angle results in the force contribution of each fibre in the direction of the length axis of the muscle." Anton [1991] on the other hand used a physical principle to derive his equilibrium equations. The same principle, the principle of virtual work, has been used for this model.

## 8. Dynamic Muscle Model Exploration

### 8.1 Introduction:

This chapter is devoted entirely to testing the DMM qualitatively and highlighting its capabilities. It is of the utmost importance that the model be shown to be consistent. This is done by comparing two cases in which the volume and all in-plane dimensions are kept the same. In the first case, the muscle is modelled using 4 panels and in the second, 8 panels are used. Obviously, an increase in the number of panels produces more information about the deformation of the whole muscle, but a consistent model is expected to converge to an identical solution. i.e. corresponding nodes should experience identical displacements. This type of reasoning is similar to the one used in finite element analyses, where an increase in the number of elements increases the accuracy of the results.

Unreferenced model comparisons to experimental results and experimental observations are made with respect to experiments conducted independently of this thesis, on the cat medial gastrocnemus, by Dr. Walter Herzog in his laboratory at the University of Calgary

The model is also tested with a single panel constraint to demonstrate its capabilities and keep the CPU time to a minimum. The choice of a single panel should in no way be viewed as a limitation of the model. It is done in order to simplify the analysis of the simulated results. The effect of including the velocity dependence and the inertia terms is examined and is compared to a static analysis case. Different stimulation rates are also considered, by varying the rate at which the parameter  $a$ , introduced in the previous chapter, is changed from 0 to 1 ( where 1 represents full stimulation ).

The assumptions and values that are used in these tests are introduced in this section.

The constant which relates the physiological cross sectional area with the maximal force  $F_0$  is taken as  $0.4 \text{ N/mm}^2$ . The maximal velocity of shortening,  $v_0$ , is taken as  $8\text{s}^{-1} \cdot L_f$  where  $L_f$  is the muscle fibre length. These values are taken from Yoshihuku and Herzog [1990] and are considered to be typical values of muscle properties. In testing the model qualitatively, these values are adequate.

The Force-Length and Force-Velocity profiles are the same as those introduced in the previous chapter.

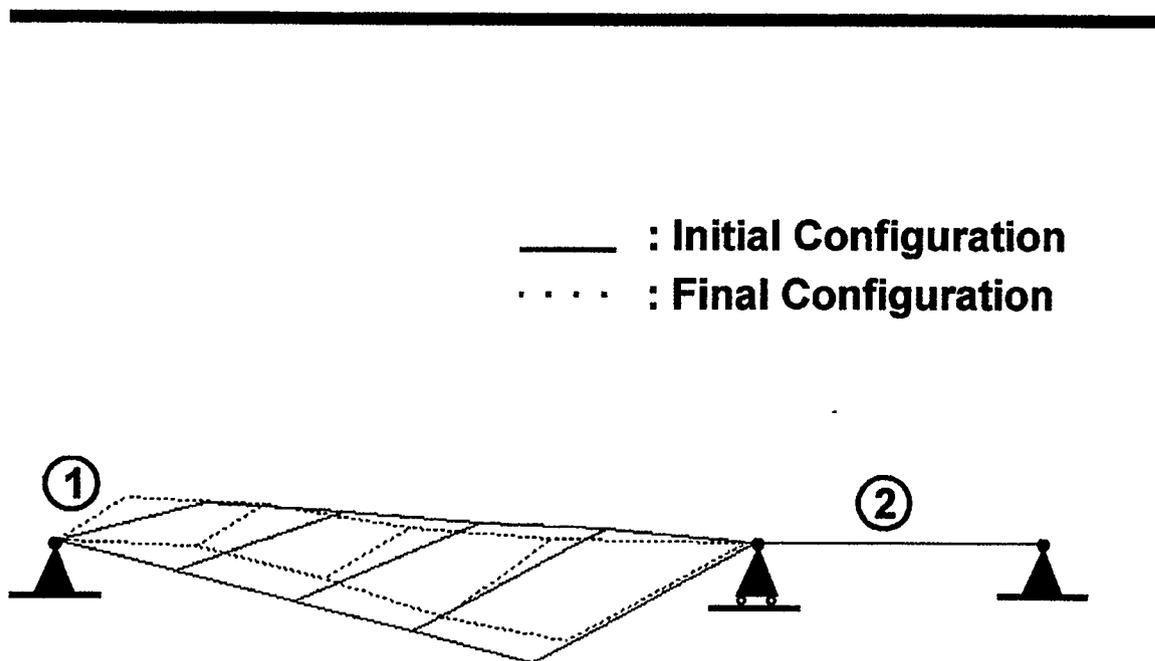
The density of the muscle is assumed to be the same as the density of water. This is significant when the inertia terms are considered, since the mass is calculated from the density and the volume of each fibre bundle.

Unless otherwise stated, the modulus of elasticity of the tendon is taken as  $600 \text{ N/mm}^2$  and that of the tendon sheath as  $1200 \text{ N/mm}^2$  [Private Communication].

## **8.2 DMM Consistency:**

In this sub-chapter, a typical muscle is modelled using 4 and 8 panels. The in-plane dimensions of the muscle are kept constant. The depth of the tendon plate in the case of 4 panels is obtained by averaging the values used in the case of 8 panels.

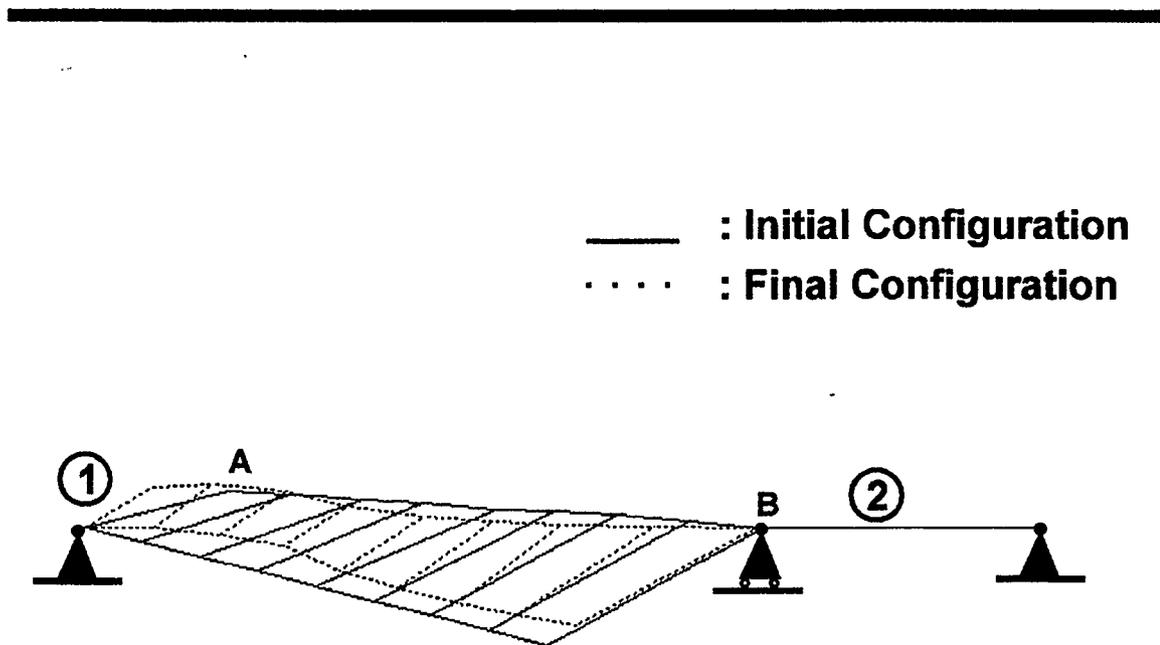
The results are shown in figures 8.1 and 8.2 respectively. It can be observed that the deformations are very close to each other in both cases. There are errors of less than 2% in the final lengths, shown in table 8.1 for selected components only, which can be



## INITIAL & FINAL MUSCLE CONFIGURATIONS

---

**Figure 8.1** Typical muscle modelled with 4 panels.



## INITIAL & FINAL MUSCLE CONFIGURATIONS

---

**Figure 8.2** Typical muscle modelled with 8 panels.

	Length of Fibre Bundle #1			Length of Tendon	
	Initial	Final		Initial	Final
4 Panels	12.53	6.64		23.25	23.91
8 Panels	12.53	6.55		23.25	23.84
% difference in length between 4 and 8 panels		<b>1.4</b>			<b>0.29</b>

**Table 8.1** Errors in some of the nodal displacements.

attributed to the averaging of the tendon sheath depths. The averaging of the angles during the implementation of the constraint is also a contributing factor.

The figures also show that the tendon / muscle attachment is free to move in a horizontal direction but constrained from moving in the vertical direction. This is done in order to speed up the convergence of the problem, but it should be noted that similar results are expected in the case where that joint is not supported.

By careful consideration of figures 8.1 and 8.2, it can also be observed that panel areas are not preserved, so that the incompressibility constraint seems to be doing the job of accurately representing volume conservation. This must be accompanied by a significant bulging out of the muscle, a phenomenon that is observed experimentally.

Having shown that the model is consistent, an attempt is now made to model a cat medial gastrocnemius muscle. The data is obtained from an independent set of

experiments performed by Dr. Walter Herzog.

### 8.3 Real Muscle Experiment:

The parameters that are used to model this muscle are the same as the ones mentioned in the introduction to this chapter. The modulus of elasticity of the tendon sheath is assumed to be 1200 N/mm<sup>2</sup> instead of 600 N/mm<sup>2</sup>. The tendon sheath has a uniform depth of 7.1 mm spanning 60% of its length and it tapers down to 2.5 mm on either end, at the origin and the muscle / tendon junction. The dimensions of the tendon are given as 0.75mm x 3mm wide. The muscle length is calculated as 50 mm (tendon plate length is approx. 40 mm) and the initial angle of pinnation is 30 degrees. The stimulation is increased from 0 to 1 in 1 second, in steps of 0.025.

It is evident, from figure 8.3, that the panel areas are not conserved. This implies a bulging out of the muscle, which is observed experimentally.

In order to get an appreciation of the magnitude of the values involved, the force that is experienced at the tendon is calculated. This also serves as a comparison against data measured in the lab. The force in the tendon is calculated, using the following formula:

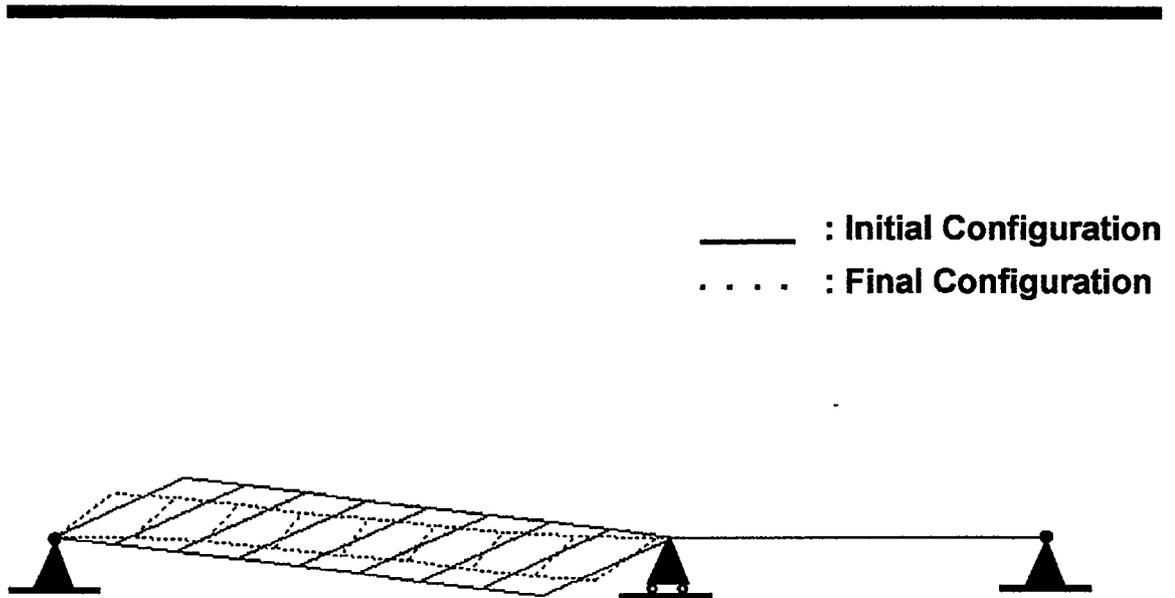
$$\text{Force} = E.A.(\Delta l / l).\cos \alpha_0$$

where  $E$  is the modulus of elasticity,

$A$  is the area of the tendon,

$\Delta l / l$  represents the strain in the tendon

$\alpha_0$  is the initial angle of pinnation (muscle is relaxed).



### INITIAL & FINAL MUSCLE CONFIGURATIONS

---

**Figure 8.3** "Real" muscle experiment.

The inclusion of the term  $\cos \alpha_0$  is to correct for the area. i.e. the difference between the area of the tendon sheath and the physiological cross sectional area of the fibre bundles. The force is calculated as 106 N, compared to 93 N obtained experimentally, which is equivalent to an error of 10%. This error is accounted for by the approximations that have to be made with regards to the dimensions of the tendon, tendon sheath and the choice of modulus of elasticity. The constant chosen to relate the maximal force to the physiological cross sectional area,  $0.4 \text{ N/mm}^2$ , also has a direct influence on the generated results.

It should be stressed at this point that the above comparison was performed to merely show that the model generates results of the right order of magnitude. These results depend, to a very large extent on the values that are input. In this case, the values required are those of muscle outer dimensions, tendon dimensions, tendon sheath depth and the moduli of elasticity of tendon and tendon sheath.

Having shown the model to be consistent and to generate values in the right order of magnitude, we now turn our attention to highlighting the capabilities of the model. As mentioned in the introduction, the series of tests that are performed in the next section involve a single panel and two fibre bundles. The areas of the fibre bundles are taken to be symmetrical. This simplifies the analysis, since the displacements are also symmetric. Hence, the general behaviour of the whole contrived muscle can be traced by simply examining the path of a single node.

#### 8.4 DMM Capabilities:

A contrived muscle geometry is used to test the model and highlight its capabilities. In this sub-chapter, the effect of including the velocity dependence and the inertia term is examined. The dimensions of the muscle are listed in table 8.2.

---

Length of tendon sheath	= 20.0 mm
Depth of tendon sheath	= 14.2 mm
Diameter of fibre bundle	= 14.2 mm
Initial fibre length	= 12.9 mm
Initial angle of pinnation	= 26 degrees

---

**Table 8.2** Initial data for contrived muscle.

The muscle is stimulated at two different rates. The first rate of stimulation produces a velocity of shortening of 8.5 mm/s, hereafter referred to as slow stimulation, whereas second rate yields a value of 50 mm/s, hereafter referred to as fast stimulation. A static case refers to a muscle which has no velocity dependence or inertia terms. In this case, the muscle is gradually stimulated to its maximum level over the entire time span. The total deflection for a static case is first determined and the stimulation rates are then calculated by dividing the total deflection over the desired time to achieve full stimulation.

The displacement results will be shown graphically for one of the nodes, in the positive horizontal direction ( figures 8.4 to 8.8 ).

The results obtained in figures 8.4 to 8.8 are as expected for materials of this type. Whereas the introduction of the velocity dependence shows a stabilizing effect, the inertia term causes the response to overshoot. This is clearly seen in figures 8.7 and 8.8 where

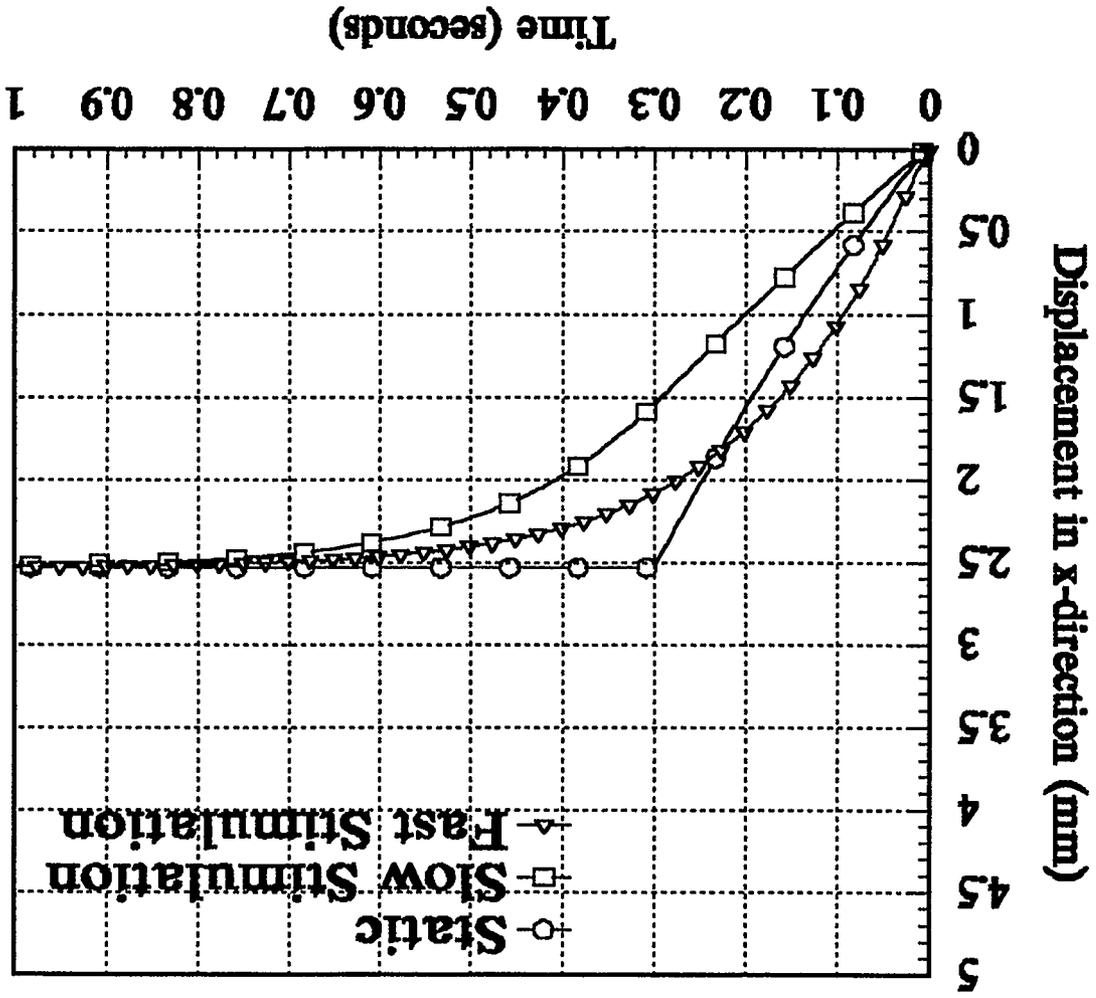
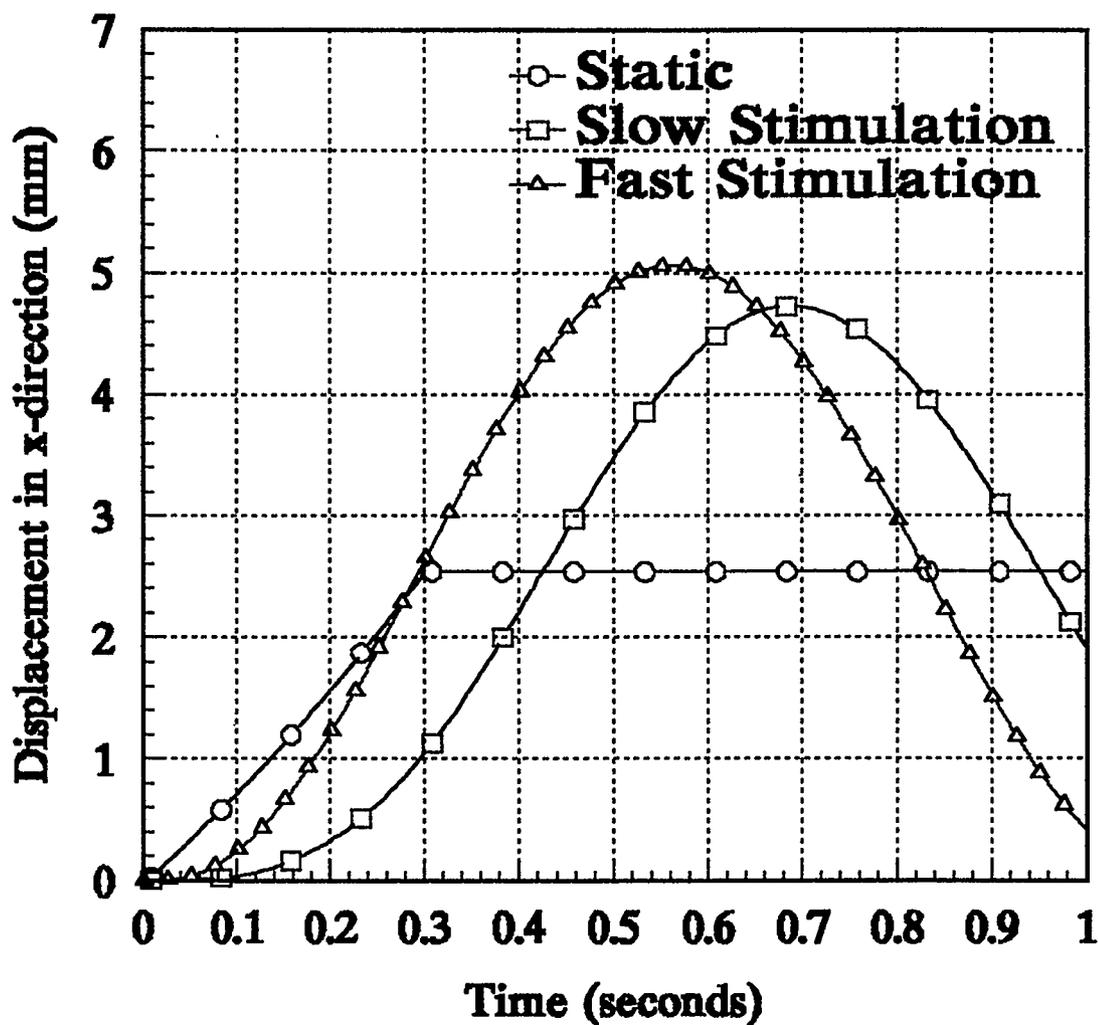
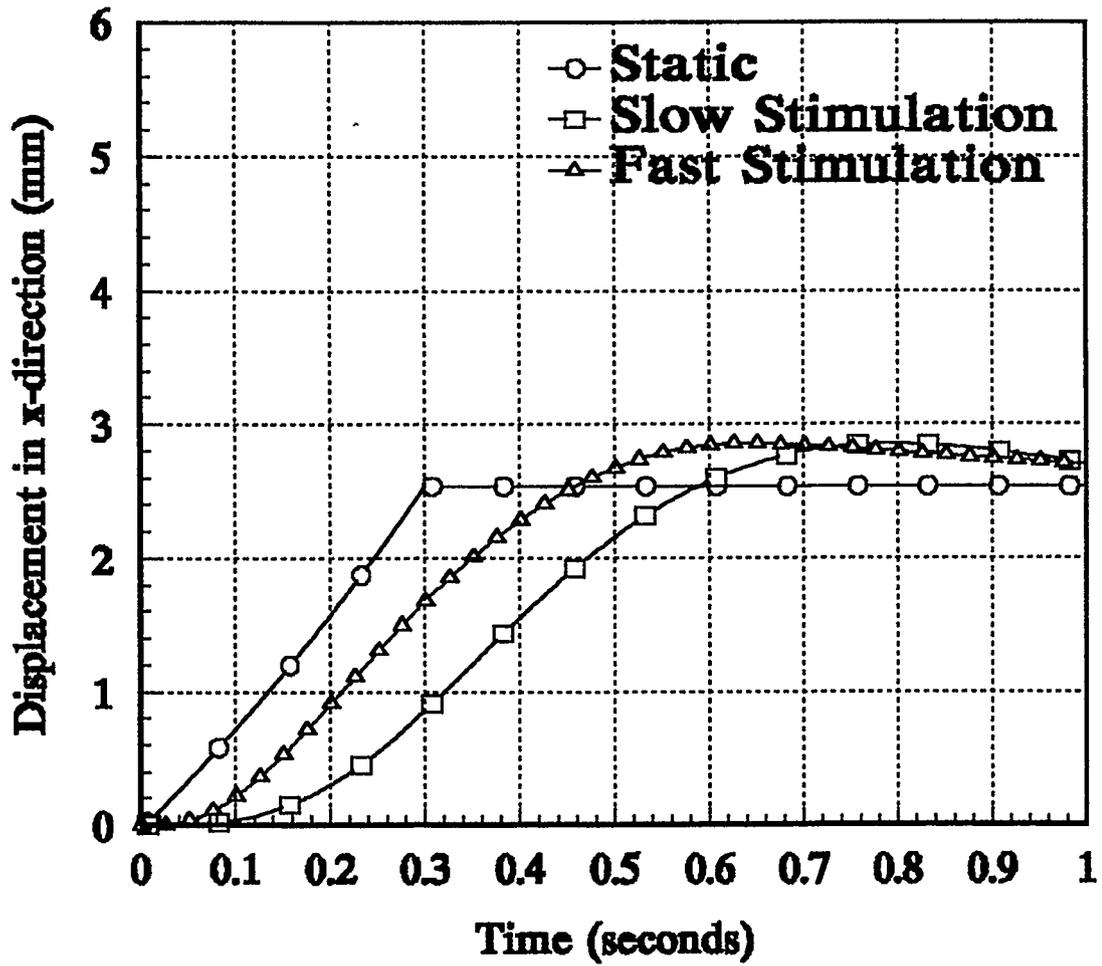


Figure 8.4 The response of a muscle to slow and fast stimulation. (No inertia, Velocity dependence included)



---

Figure 8.5 The response of a muscle to slow and fast stimulation.  
(No velocity dependence, Inertia included)



---

**Figure 8.6** The response of a muscle to slow and fast stimulation.  
(Inertia and Velocity dependence included)

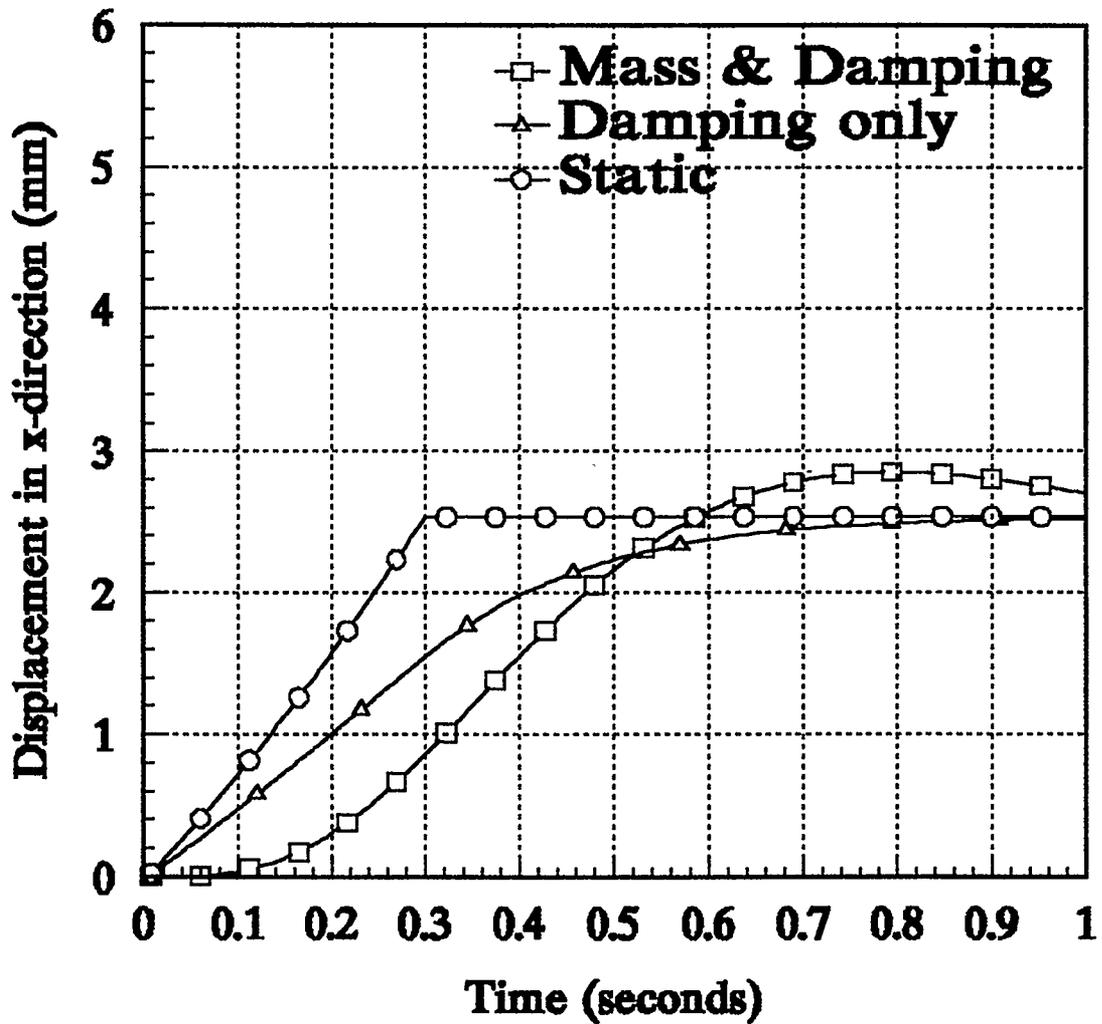
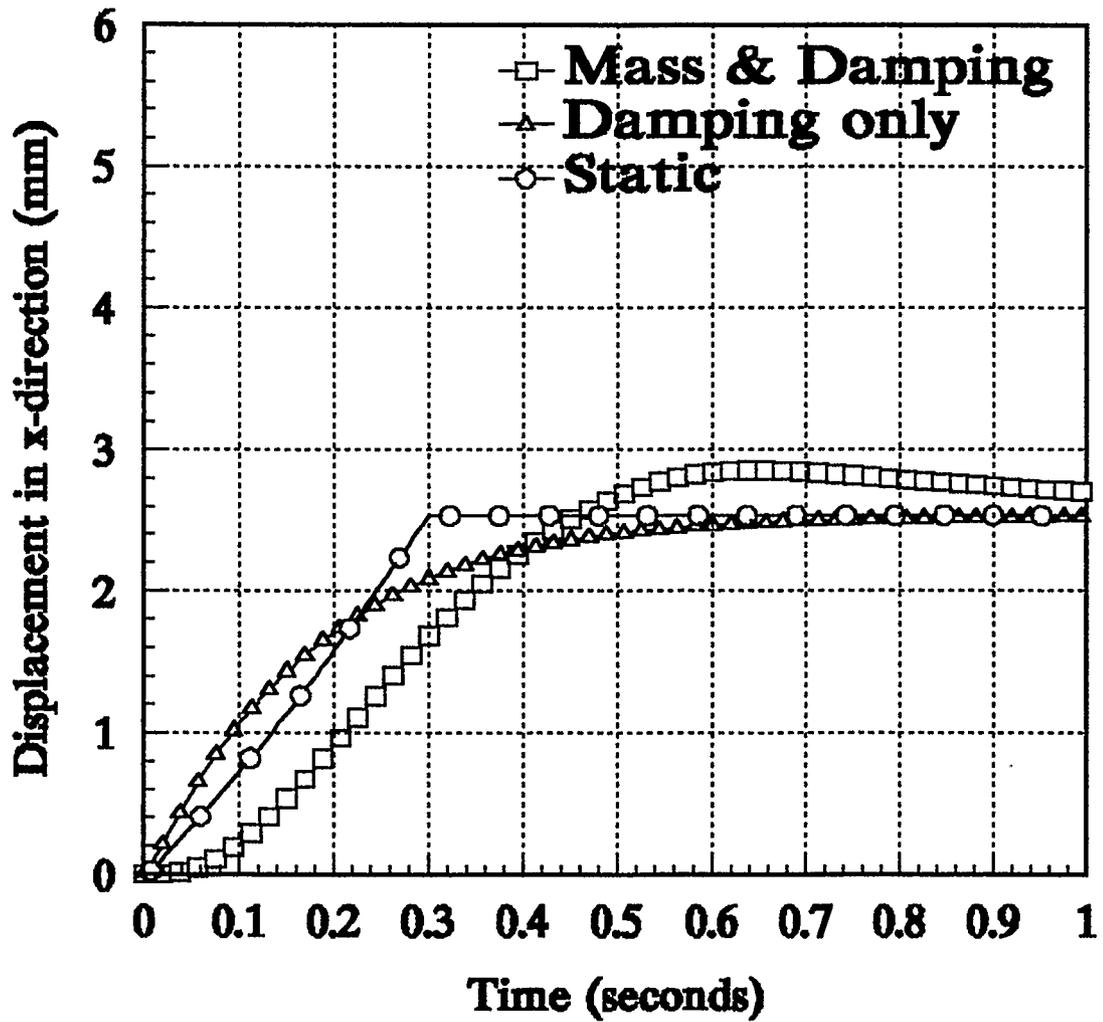


Figure 8.7 The response of a velocity dependent muscle to slow stimulation.  
( with and without Inertia )



---

**Figure 8.8** The response of a velocity dependent muscle to a fast stimulation.  
( with and without Inertia )

the response with and without inertia terms is compared to a static analysis. However, in both these graphs, it can be seen that the overshoot is of a small order of magnitude and after 1 second, the solutions converge to a similar value.

Another observation, as seen from these examples, is that the response of muscles is critically damped. This result is not altogether surprising if one considers the smooth functioning of most of our everyday muscular activities.

The example in which the inertia term is considered without any velocity dependence is not a realistic one (figure 8.5). It does, however, serve as further evidence of the correct functioning of the model. This combination is equivalent to a spring-mass mechanical system which would oscillate for ever, in the absence of any resistance.

The results of the combination seen in figure 8.6 seem to be the most realistic. The path followed by the node is representative of the build up of force in the muscle. The exact relation between stimulation and maximum force build up is not known. Experimentally, however, when a muscle is stimulated, there is a time lag before the maximum force of that configuration is observed [private communication]. This is evident from the graph.

In closing, it should be noted that the model has been shown to be consistent and exhibits the right type of behaviour. As with any model, the results are directly related to the inputs, whereas the behaviour is dependent upon the assumptions and simplifications made at the outset. The model would require extensive testing and a review of some of the assumptions and simplifications made before any claims about its quantitative worth are made. The preliminary indications are that the forces and displacements are of the

right order of magnitude.

In the next chapter, a summary of the objectives and achievements will be reiterated. Recommendations for further work are suggested, including ways to improve the quantitative aspect of the model.

## 9. Summary and Recommendations

A two dimensional dynamic muscle model has been successfully developed within the guidelines set at the beginning. Based on a physical principle, the principle of virtual work, the model has been shown to be consistent. The method of Lagrange Multipliers has been used to implement the kinematic constraint of incompressibility. The equations describing the model are expressed by a set of non linear algebraic equations.

The model is based on the underlying assumption that muscle fibres are the sole force generators and that their mechanical behaviour is a reflection of the active and passive muscle fibre characteristics exclusively.

The velocity dependence is based on an empirical relation formulated by Hill [1938] and the fibre force-length characteristics are taken from Woittiez et al. [1984] and Anton [1991]. In the context of this thesis, these relations are assumed to hold true over the specified ranges.

The effect of the inertia term on the behaviour of a contrived muscle has also been examined. As expected, this has a negligible effect after a long period of time, 1 second, but there is a visible overshoot at the early stimulation time.

The introduction of the velocity dependence has a stabilizing effect on the response of the whole system. This is to be expected, especially when one considers the smooth functioning of the whole musculo-skeletal system.

Other parameters that are relevant to the muscle tissue, tendon sheaths and the tendon are given at the start of each section. These values are approximations and it would be unrealistic to expect accurate, quantitative results based on those inputs.

Consequently, the model can only be tested qualitatively against some of the other models cited in the literature.

Nevertheless, the model has been tested against some data from independent experiments on the medial gastrocnemius of a cat. Initial analysis of these simulations yields results of the same order of magnitude. However, the model has to be tested extensively before any claims are made.

Having met the initial objectives of developing a consistent model, there are certain assumptions and simplifications that can be re assessed in order to improve the model.

The implementation of the incompressibility constraint is performed by the averaging of pinnation angles and panel lengths. In general, when the tendon sheaths are parallel, or almost so, this method represents an adequate way of implementation. As the model becomes trapezoidal in nature, there is a systematic error introduced due to averaging. This could be overcome by increasing the number of panels, but the increased CPU cost could be prohibitive.

Another area in which the model might be improved is in the modelling of the tendon sheath characteristics, since these form an integral part of the incompressibility constraint. It is believed that the tendon sheath de-crimps when it is subjected to a tensile load. Based on this assumption, the force-displacement characteristic of the tendon sheath could be modelled as being exponential.

Having seen the capabilities and limitations of the model, it is appropriate to mention some areas in which this model could be applied.

Based on the ability to model different stimulation rates, this model could be employed in trying to find some empirical relationship between stimulation-activation-force output.

By means of some modifications to the computer code and a reassessment of some of the assumptions, the model could also be modified to look at isotonic muscle behaviour. Specifically, the case in which a fully contracted muscle experiences a sudden displacement of one of its ends, could be studied. The response of the whole muscle from the point of view of inertia and "damping" could be traced.

## Bibliography

- Abbott, B.C. and Wilkie, D.R. (1953)** The relation between velocity of shortening and the tension-length curve of skeletal muscle. *Journal of Physiology*, 120: 214-223.
- Abbott, B.C. and Baskin, R.J. (1962)** Volume changes in frog muscle during contraction. *Journal of Physiology (London)*. 161: 379-391.
- Anton, M.G. (1991)** Mechanical Models of Skeletal Muscle. PhD Thesis, University of Calgary.
- Bagshaw, C.R. (1982)** Muscle Contraction. Chapman and Hall, London.
- Beer, F.P. and Johnston Jr., E.R.** Vector Mechanics for Engineers, 3<sup>rd</sup> edition. Chapter 10, Statics and Dynamics. MCGRAW-HILL BOOK COMPANY.
- Benninghoff, A. and Rollhäuser, H. (1952)** Zur inneren Mechanik des gefiederten Muskels. *Pflügers Archiv für die gesmate Physiologie*, 254: 527-548.
- Bisplinghoff, R.L., Mar, J.W. and Pian, T.H.H. (1965)** Statics of Deformable Solids. Addison-Wesley Publishing Company, Inc.
- Epstein, M. (1985)** Introduction to the Finite Element Method in Elasticity. Course notes for ENME 551, Department of Mechanical Engineering, University of Calgary.
- Epstein, M. and Tene, Y. (1971)** NONLINEAR ANALYSIS OF PIN-JOINTED SPACE TRUSSES. *Proceedings of the American Society of Civil Engineers*, ST 9: 2189-2202.
- Fung, Y.C. (1970)** Mechanics of the heart muscle. *Journal of Biomechanics*, 3: 381-404.
- Gans, C. (1982)** Fibre architecture and muscle function. *Exercise and Sport Sciences Review*, 10: 160-207.
- Gordon, A.M., Huxley, A.F., Julian, F.J. (1966)** The variation in isometric tension with sarcomere length in vertebrate muscle fibres. *Journal of Physiology*, 184: 170-192.
- Hatze, H. (1976)** The complete optimization of the human motion. *Mathematical Biosciences*, 28: 99-135.
- Herzog, W. (1987)** Individual muscle force estimates using a non-linear optimal design. *Journal of Neuroscience Methods*, 21: 167-179.
- Heukelom, B., van der Stelt, A., Diegenbach, P.C. (1979)** A simple anatomical model of muscle and the effects of internal pressure. *Bulletin of Mathematical Biology*, 41: 791-802.

- Hilderbrand, F. B. (1965)** Methods of Applied Mathematics. 2<sup>nd</sup> edition, Chapter 2.  
PRENTICE-HALL, INC.
- Hill, A.V., (1938)** The heat of shortening and the dynamic constants of muscle.  
Proceedings of the Royal Society of London. Series B: Biological Sciences  
(London), 126: 136-195.
- Hill, A.V. (1950)** Mechanics of the Contractile Element of Muscle.  
Nature, 166, 415-419.
- Huxley, A.F., (1957)** Muscle structure and theories of contraction. Progress in Biophysics and  
Biophysical Chemistry, 7: 255-318.
- Huxley, A.F., (1974)** Muscular contraction. Journal of Physiology, 243: 1-43.
- Kardel, T. (1990)** Niels Stensen's geometrical theory of muscle contraction (1667): A  
reappraisal. Journal of Biomechanics, 23: 953-965.
- Lieber, R. L. and Blevins, F. T. (1989)** Skeletal muscle architecture of the rabbit hindlimb:  
functional implications of muscle design. J. Morph. 199, 93-101.
- Otten, E. (1985)** Morphometrics and force-length relations of skeletal muscles. In A. Winter et  
al. (eds) International Series on Biomechanics (ISB), Biomechanics, Volume IX-A,  
Champaign, Ill.: Human Kinetic Publishers. pp. 27-32.
- Otten, E. (1987a)** Optimal design of vertebrate and insect sarcomeres. Journal of Morphology,  
191: 49-62.
- Otten, E. (1987b)** A myocybernetic model of the jaw system of the rat. Journal of Neuroscience  
Methods, 21: 287-302.
- Otten, E. (1988)** Concepts and models of functional architecture in skeletal muscle. Exercise  
and Sport Sciences Reviews, 16: 89-137.
- Pestel, E.C. and Thomson, W.T.** Statics, Chapter 6.  
McGRAW-HILL BOOK COMPANY.
- Press, W.H. et. al. (1990)** Numerical Recipes in C, The Art of Scientific Computing.  
CAMBRIDGE UNIVERSITY PRESS.
- Purslow, P.P and Duance, V.C.** Structure and Function of Intramuscular Connective Tissue.  
Connective Tissue Matrix Part 2. Edited by D.W.L.Hukins  
CRC Press, Inc.

- Sagan, H. (1969)** Introduction to the Calculus of Variations, Chapter 6.  
McGRAW-HILL BOOK COMPANY.
- Schmalbruch, H. (1985)** Skeletal Muscle.  
Springer-Verlag, Berlin.
- Scott, S.H. and Winter, D.A. (1991)** A comparison of three muscle pinnation assumptions and their effect on isometric and isotonic force. *Journal of Biomechanics*, 24: 163-167.
- Tillerson, J.R., Stricklin, J.A. and Haisler, W.E. (1973)** Numerical Methods for the solution of non linear problems in Structural Analysis. A.S.M.E. AMD-Vol. 6
- Wickiewicz, T. L., Roy, R. R., Powell, P. J. and Edgerton, V. R. (1983)** Muscle architecture of the human lower limb. *Clin. Orthop. Rel. Res.* 179, 317-325.
- Woittiez, R.D., Huijing, P.A., Rozendal, R.H. (1983)** Influence of muscle architecture on the length-force diagram: a model and its verification. *Pflügers Archiv für die gesamte Physiologie*, 397: 73-74.
- Woittiez, R.D., Huijing, P.A., Rozendal, R.H. (1984)** A three-dimensional model: A quantified relation between form and function of skeletal muscles. *Journal of Morphology*, 182: 95-113.
- Yoshihuku, Y. and Herzog, W. (1990)** Optimal design parameters of the bicycle-rider system for maximal muscle power output. *Journal of Biomechanics*, 23: 1069-1079.

## Appendix A:

```
/* ===== */
/*
The listing below is a representation of a program which uses the
Newton-Raphson method to solve n simultaneous nonlinear equations.
The program can handle geometrically nonlinear trusses subjected
to static loading and provides damping capabilities.
date: Oct. 25,1993

The function definitions and utilities are stored in their
respective .h files, & are called up with a #include
A table of incidence stores the initial bar information
    { 3 structures are used: line(), node(), & info() }
The information which is relevant to the physical problem can be
prepared and input in the following files:
    NODES.DAT -> node numbers and coordinates
    CONNECT.DAT -> node connectivity and bar diameter
    SUPPORT.DAT -> external node supports
    FORCE.DAT -> external forces on nodes
This information can also be input interactively at run time:
    nodal information & bar dimensions are input in "innodes"
    forces on each node are input in "forces"
    constraints on each node are input in "support"
Initial guesses are input in "readinput"
The length of connecting lines is calculated in "lent"
A force history can be specified in "history"
Mnewt calls up the equation solvers, the virtual work routine
    "vw" as well as perform the partial differentiation .
(Mnewt -> Callfunc -> g
    -> vw -> eij )
    -> kij )

The strain is calculated in "eij"
The internal forces are calculated in "kij"
A General Constraint is introduced in "g". This increases the vector
of unknowns by one per constraint introduced and acts to modify the
internal virtual work. This is commonly known as the Lagrange
Multiplier Method.
All unknowns are updated after each iteration, but a convergence
check is only performed on the displacement variables.
Data is stored in TEST.DAT & OUT.DAT */
/* ===== */

#include <stdio.h>
#include <math.h>
#include <alloc.h>
#include <stdlib.h>
#include <conio.h>
#include <dos.h>
#include <npsio.h> /* contains all function definitions */
#include <utils.h> /* contains all the utilities */

struct lines{
    int node1;
    int node2;
    int type; /* specifies material property (fibre or tendon) */
    float dx;
    float dy;
    float dx1; /* initial distance */
    float dy1;
    float length; /* non negative */
    float inlen; /* initial length */
    float ejk; /* strain */
    float ejkdot; /* time derivative of strain */
};
```

```

float dejk;          /* delta strain */
float intf;          /* internal force */
float area;          /* X-sectional area of each bar */
float fopt;          /* Optimal fibre lengths */
float diam;          /* Tendon plate thickness */
float depth;         /* Third dimension */
float angle;         /* Pinnation angle of panel */
} line[60];          /* up to 30 lines */

struct nodes{
float x;             /* x coordinate of node */
float y;             /* y coordinate of node */
float forces;        /* store forces */
float iforce;        /* initial force */
float inval;         /* initial constraint value */
float n1;            /* define panel nodes*/
float n2;
float n3;
float n4;
} node[60];          /* up to 30 nodes */

struct general{
int connect;         /* number of interconnections */
int total;           /* total number of nodes */
int kc;              /* number of modified indices */
float result;        /* force history parameter */
int icon;            /* number of panels */
float active;        /* level of muscle activation */
float oplength;      /* muscle optimum length */
float lambda;
float ymod2;         /* Youngs' Modulus for tendon */
} info[1];

FILE *fp;
FILE *fp5;
FILE *fp6;
FILE *fp7;
FILE *fp8;
FILE *fp9;

/* The MAIN section of the program */
/* ----- */
main()
{
int count, nodes, connect, index, numcon, i, p, q;
int ntrial, ww, nd, dir, icon, j, counter;
float *u, *extdp, *uold, *uddot, *value, *n1, *n2, *n3, *n4, *pan;
float *outdp, *udot, *lratio, active, ymod2, *pangle;
float *udotp, *xp, *yp, ymod, mlength, ratio;
float dur, dt, tt, stm, result, oplength, activation;
textmode(C4350);
clrscr();
printf("\nDeleting TEST.$$\n");
system("del test.$$$");
printf("\n*****");
printf("\n*");
printf("\n* ALL NODAL POINTS should be input in NODES.DAT");
printf("\n* THE NODAL CONNECTIONS and BAR DIAMETERS are entered in CONNECT.DAT");
printf("\n* THE NODAL CONSTRAINTS should be entered in SUPPORT.DAT");
printf("\n* THE NODAL FORCES should be entered in FORCE.DAT");
printf("\n* ALL BARS HAVE THE SAME YOUNGS' MODULUS");
printf("\n* ALL BARS HAVE THE SAME DENSITY ( 1 kg/m^3 = 10E-6 g/mm^3 )");

```

```

printf("\n* ASSUMING ZERO INITIAL VELOCITY                                     *");
printf("\n*                                                                 *");
printf("\n*****");
printf("\n");
    printf("\nModulus of Elasticity of the Tendon Plate? (approx. 600N/mm^2) ");
    scanf("%f", &ymod);
    printf("\nModulus of Elasticity of the Tendon ? (approx. 1200 N/mm^2) ");
    scanf("%f", &ymod2);
    printf("\nWhat is the optimum muscle length ? ");
    scanf("%f",&oplength);
    printf("\nWhat is the activation level ? ");
    scanf("%f",&active);
    info[1].ymod2 = ymod2;
    info[1].active = active;
    info[1].oplength = oplength;
    printf("\nHow many nodal points ? ");
    scanf("%d", &nodes);
    if( nodes <= 1 ) nrerror("Cannot calculate the length at a point");
    info[1].total = nodes; /* store total number of nodes */
    printf("\nHow many inter-connections ? ");
    scanf("%d", &connect);
    info[1].connect = connect;          /* store number of connections */
    printf("\nHow many panels ?");
    scanf("%d", &icon);
    info[1].icon = icon;               /* store number of panels */
    index = 2*(nodes-1)+2;            /* modified # of indices */
    index = index + icon;             /* add one parameter per constraint */
    info[1].kc = index;               /* store converted node number */
    printf("\nHow many external supports ?");
    scanf("%d", &numcon);
    u = vector(1,index);              /* Memory Allocation */
    uold = vector(1,index);
    udot = vector(1,index);
    uddot = vector(1,index);
    udotp = vector(1,index);
    extdp = vector(1,index);
    outdp = vector(1,index);
    lratio = vector(1,connect);
    fp = fopen("TEST.$$$", "w");
        fprintf(fp,"Youngs' modulus of Tendon Plate = %1.4f N/mm^2\n",ymod);
        fprintf(fp,"Youngs' modulus of Tendon = %1.4f N/mm^2\n",ymod2);
        fprintf(fp,"Number of Nodes = %d\n",nodes);
        fprintf(fp,"Number of Connections = %d\n",connect);
        fprintf(fp,"\n\n");
        fclose(fp);
    for( count=1; count<=index; count++ )          /* initialize all variables */
    {
        u[count] = 0;
        uold[count] = 0;
        udot[count] = 0;
        extdp[count] = 1;                          /* initialize all nodes as unconstrained */
        outdp[count] = 0;
        uddot[count] = 0;
        udotp[count] = 0;
        node[count].x = 0;
        node[count].y = 0;
        node[count].forces = 0;
        node[count].iforce = 0;
    }
    for( count=1; count<=icon; count++ )
    {
        node[count].n1 = 0;

```

```

        node[count].n2 = 0;
        node[count].n3 = 0;
        node[count].n4 = 0;
        node[count].inval = 0;
    }
    for( count=1; count<=connect; count++ )
    {
        lratio[count] = 0;
        line[count].ejk = 0;
        line[count].dejk = 0;
        line[count].intf = 0;
        line[count].inlen = 0;
        line[count].ejkdot = 0;
        line[count].length = 0;
    }
    readinput( u, udot, index );
    innodes( nodes, connect );
    for( count=1; count<=connect; count++ )
    {
        line[count].inlen = line[count].length;    /* store initial length */
        line[count].dx1 = line[count].dx;
        line[count].dy1 = line[count].dy;
    }
    /* Calculate Muscle Length */
    mlength = node[nodes].x - node[1].x;
    info[1].lambda = mlength / oplength;
    /* ----- */
    support( numcon, extdp );
    counter = index - icon;
    forces( nodes, counter );
    printf("\nMUSCLE length = %1.2f    L/L0 = %1.2f    Activation = %1.2f\n", mlength, info[1].lambda, info[1].active );
    foptimum( connect );
    for( count=1; count<=index; count++ )
    {
        node[count].iforce = node[count].forces;    /* store force magnitude */
    }
    /* read in panel data */
    if( icon != 0 )
    {
        n1 = vector(1,icon);
        n2 = vector(1,icon);
        n3 = vector(1,icon);
        n4 = vector(1,icon);
        pan = vector(1,icon);
        value = vector(1,icon);
        if((fp6=fopen("panel.dat","r"))==NULL)
        {
            printf("\nMissing Data File (PANEL.DAT) ! ");
            exit(0);
        }
        else
        {
            i = 1;
            while( !feof(fp6) && (i<=icon) )
            {
                fscanf(fp6,"%f %f %f %f %f",&pan[i],&n1[i],&n2[i],&n3[i],&n4[i]);
                node[i].n1 = n1[i];
                node[i].n2 = n2[i];
                node[i].n3 = n3[i];
                node[i].n4 = n4[i];
                i++;
            }
        }
    }

```

```

    }

    for( j=1; j<=icon; j++ )
    {
        g(u,value,index,j,n1,n2,n3,n4);
        node[j].inval = value[j];
    }
    free_vector(n1,1,icon);
    free_vector(n2,1,icon);
    free_vector(n3,1,icon);
    free_vector(n4,1,icon);
    free_vector(pan,1,icon);
    free_vector(value,1,icon);
}
pangle=vector(1,icon);
for( i=1; i<=icon; i++ )
{
    pangle[i] = line[i].angle;
    pangle[i] = pangle[i]*57.29578;          /* convert to degrees */
    printf("\nAngle of Pinnation[%d] = %1.2f",i,pangle[i]);
}
printf("\n");
out( index, icon );          /* open output data file */
printf("\nWhat is the time duration of the force history ?\n");
scanf("%f",&dur);
printf("\nWhat is the starting time ( >= 0 ) ? \n");
scanf("%f",&stm);
printf("\nWhat is the value of delta t ? \n");
scanf("%f",&dt);
printf("\nHow many iterations ?\n");
scanf("%d", &ntrial);
if( ntrial == 0 ) nerror("No iterations possible");
fp7 = fopen("nodes2.$$$","w");
fprintf(fp7,"\n");
fclose(fp7);
fp8 = fopen("ratio.$$$","w");
fprintf(fp8,"\nActivation   Initial Length(mm)   Final Length(mm)   Ratio");
fclose(fp8);
fp9 = fopen("disp.$$$","w");
fprintf(fp9,"\nAct.           Displacements (of all NODES)");
fprintf(fp9,"\n");
fclose(fp9);

for( tt=stm; tt<dur; tt+=dt )
{
    history( tt, dt );
    result = info[1].result;
    printf("\n");
    printf("The scaling parameter at t= %1.4f is %1.4f", tt, result);

    if( tt == stm )
    {
        for( i=1; i<=index; i++ )
        {
            uold[i] = u[i];
        }
        for( i=1; i<=index; i++ )
        {
            udot[i] = (u[i]-uold[i])/dt;
        }
        activation = 0;
        output( index, icon, u, tt, pangle, activation );
    }
}

```

```

printf("\nThe Solution is :\n");
printf("\nu[i]    uold[i]    udot[i]");
for( i=1; i<=index; i++ )
{
    printf("\n%1.3f    %1.3f    %1.3f",u[i],uold[i],udot[i]);
}
printf("\n");
}
else
{
    for( i=1; i<=index; i++ )
    {
        uold[i] = u[i];
    }
    printf("\n");
    mnewt( u, extdp, index, ymod, uold, ntrial, dt);
    activation = result*info[1].active;
    fp9 = fopen("disp.$$$","a+");
    fprintf(fp9,"\n");
    fprintf(fp9,"%1.3f    ",activation);
    for( i=1; i<=index; i++ )
    {
        fprintf(fp9,"%1.2f            ",u[i]);
    }
    fclose(fp9);
    for( i=1; i<=icon; i++ )
    {
        pangle[i] = line[i].angle;
        pangle[i] = pangle[i]*57.29578;    /* convert to degrees */
        printf("\nAngle of Pinnation[%d] = %1.2f",i,pangle[i]);
    }
    output( index, icon, u, tt, pangle, activation );
    printf("\nThe Solution is :\n");
    printf("\nu[i]    uold[i]    udot[i]");
    for( i=1; i<=index; i++ )
    {
        printf("\n%1.3f    %1.3f    %1.3f",u[i],uold[i],udot[i]);
    }
    printf("\n");
    xp = vector(1,nodes);
    yp = vector(1,nodes);
    fp7 = fopen("nodes2.$$$","a+");
    fprintf(fp7,"\n");
    for( i=1; i<=nodes; i++ )
    {
        p = 2*i-1;
        q = 2*i;
        /* increments due to iterations */
        xp[i] = node[i].x + u[p];
        yp[i] = node[i].y + u[q];
        fprintf(fp7,"\n%d            %1.3f    %1.3f",i,xp[i],yp[i]);
    }
    fprintf(fp7,"\n");
    fclose(fp7);
    free_vector(xp,1,nodes);
    free_vector(yp,1,nodes);
    fp8 = fopen("ratio.$$$","a+");
    fprintf(fp8,"\n");
    for( i=1; i<=connect; i++ )
    {
        ratio = line[i].length/line[i].inlen;
        fprintf(fp8,"\n%1.2f            %3.2f    %3.2f    %3.2f", activation, line[i].inlen, line[i].length, ratio);
    }
}

```

```

    }
    fprintf(fp8,"\\n");
    fclose(fp8);
}
}
printf("\\nNode Node Initial Length(mm) Final Length(mm)");
for( i=1; i<=connect; i++ )
{
    printf("\\n%2d %2d %3.2f %3.2f",line[i].node1,line[i].node2,line[i].inlen,line[i].length);
}
printf("\\n");
printf("\\nDone !\\n");
free_vector(u,1,index);
free_vector(uold,1,index);
free_vector(udot,1,index);
free_vector(uddot,1,index);
free_vector(udotp,1,index);
free_vector(extdp,1,index);
free_vector(outdp,1,index);
free_vector(pangle,1,icon);
free_vector(lratio,1,index);
exit(0);
return;
}

/* INITIAL TABLE OF INTEGERS */
/* ----- */
void readinput (float *u, float *udot, int index)
{
    int i, node, direc, count;
    float ndisp, nvel, val;
    printf("\\nHow many imposed displacements ? -> ");
    scanf("%f", &ndisp);
    printf("\\nWhich node is displaced ? In which direction ? By how much ?\\n");
    for( i=1; i<=ndisp; i++ )
    {
        scanf("%d %d %f",&node,&direc,&val);
        count = 2*(node-1) + direc;
        u[count] = val;
    }
    printf("\\nHow many imposed velocities ? -> ");
    scanf("%f", &nvel);
    printf("\\nAt which node ? In which direction ? Value ?\\n");
    for( i=1; i<=nvel; i++ )
    {
        scanf("%d %d %f",&node,&direc,&val);
        count = 2*(node-1) + direc;
        udot[count] = val;
    }
    printf("\\nGuesstimate of u[U1, U2, ...] is: \\n");
    for( i=1; i<=index; i++ )
    {
        printf("%6.1f", u[i]);
    }
    printf("\\n");
    return;
}

/* INPUT NODE INFORMATION IN THIS SECTION */
/* ----- */
void innodes(int nodes, int connect )
{

```

```

FILE *fp1;
FILE *fp2;
int i, j, k, index, type;
int n=1;
float *xc, *yc, diam, area, depth;
index = info[1].kc;
xc = vector(1,index);
yc = vector(1,index);
if(( fp1 = fopen("n2.dat","r") == NULL)
{
    printf("\n          NOTE: RECTANGULAR COORDINATE SYSTEM !");
    printf("\nSpecify a node number and its coordinates e.g. 1 (x1,y1)");
    printf("\n");
    for( i=1; i<=nodes; i++ )          /* input nodes & coordinates here */
    {
        scanf("%d %f %f", &n, &xc[i], &yc[i]);
        node[i].x = xc[i];
        node[i].y = yc[i];
    }
}
else
{
    printf("\nNode number and its coordinates e.g. 1 (x1,y1)");
    while( !feof( fp1 ) && (n <= nodes))
    {
        fscanf(fp1,"%d %f %f",&n,&xc[n],&yc[n]);
        printf("\n%d          %1.2f          %1.2f",n,xc[n],yc[n]);
        node[n].x = xc[n];
        node[n].y = yc[n];
        n++;
    }
    if( n<=nodes ) perror("\nWarning! Not all nodes read! Check NODES.DAT");
}
fclose(fp1);
printf("\n\n");
n = 1;
if(( fp2 = fopen("connect.dat","r") == NULL)
{
    printf("Bar#      Type      Diam.      Area(mm^2)\n");
    for( i=1; i<=connect; i++ )
    {
        printf("Please Specify Node Connection, Bar Diameter & Material Type(1or2)");
        scanf("%d %d %f %f %d", &j,&k,&diam,&depth,&type);          /* specify connectivity & diameter & type */
        line[i].node1 = j;          /* store node number for each line */
        line[i].node2 = k;          /* "" */
        line[i].type = type;
        area = 22*diam*diam/28;
        printf("%d          %d          %1.3f          %1.2f\n",i,type,diam,area);
        line[i].area = area;
        line[i].diam = diam;
        line[i].depth = depth;
    }
}
else
{
    printf("Bar#      Type      Diam      Area(mm^2)\n");
    while( !feof( fp2 ) && (n <= connect))
    {
        fscanf(fp1,"%d %d %f %d %f",&j,&k,&diam,&type,&depth);
        line[n].node1 = j;          /* store node number for each line */
        line[n].node2 = k;          /* "" */
        line[n].type = type;
    }
}
}

```

```

        area = 22*diam*diam/28;
        printf("%d      %d      %1.3f      %1.2f\n",n,type,diam,area);
        line[n].area = area;
        line[n].diam = diam;
        line[n].depth = depth;
        n++;
    }
    if( n<=connect ) nerror("\nWarning! Not all connections read! Check CONNECT.DAT");
    }
    fclose(fp2);
    printf("\n");
    lent( xc, yc );          /* calculate length */
    free_vector(xc,1,index);
    free_vector(yc,1,index);
    return ;
}

/* CALCULATE THE LENGTH OF EACH SEGMENT */
/* ----- */
void lent( float *xc, float *yc )
{
    int i, j, k, connect;
    float xdist, ydist, xlen, ylen;
    float *length;
    connect = info[1].connect;          /* `import' value of connect */
    length = vector(1,connect);
    for( i=1; i<=connect; i++)
    {
        j = line[i].node1;          /* import value */
        k = line[i].node2;          /* import value */
        xdist = (xc[k] - xc[j]);
        xlen = xdist*xdist;          /* calculate length of x */
        ydist = (yc[k] - yc[j]);
        ylen = ydist*ydist;          /* calculate length of y */
        length[i] = sqrt(xlen+ylen); /* line length */
        line[i].length = length[i]; /* store value */
        line[i].dx = xdist;
        line[i].dy = ydist;
    }
    free_vector(length,1,connect);
    return;
}

/* INPUT THE NODAL SUPPORTS */
/* ----- */
void support( int numcon, float *extdp )
{
    FILE *fp3;
    int i, k, nd, dir, index;
    float value;
    index = info[1].kc;
    if(( fp3 = fopen("support.dat","r") ) == NULL)
    {
        float far **ddu;
        ddu = matrix(1,index,1,index);
        printf("\nWhich nodes are supported ?\n");
        printf("specify node #, direction & value.\n");
        printf("");
        for( i=1; i<=numcon; i++ )
        {
            scanf("%d %d %f", &nd, &dir, &value);
            ddu[nd][dir] = value;
        }
    }
}

```

```

        printf("ddu[%d][%d] = %1.3f\n", nd, dir, value);
        printf("");
        k = 2 * (nd-1) + dir;
        extdp[k] = value;
    }
    free_matrix(ddu,1,index,1,index);
}
else
{
    i=1;
    printf("\nNode Direction Support");
    while( !feof( fp3 ) && ( i <= numcon))
    {
        fscanf(fp3,"%d %d %f",&nd,&dir,&value);
        printf("\n%2d      %5d          %5.2f",nd,dir,value);
        k = 2 * (nd-1) + dir;
        extdp[k] = value;
        i++;
    }
    if( i<=numcon ) perror("\nWarning! Not all connections read! Check CONSTRNT.DAT");
}
fclose(fp3);
printf("\n\n");
return;
}

/* INPUT ALL THE NODAL FORCES (EXTERNAL) */
/* ----- */
void forces( int nodes, int index )
{
    FILE *fp4;
    int i, j, k, nd, dir;
    float value;
    if(( fp4 = fopen("force.dat","r")) == NULL)
    {
        float far **force;
        force = matrix(1,index,1,index);
        printf("\nInput the forces for all the nodes.");
        printf("\n      e.g. force[i][j] = force on node i in j direction.");
        printf("\n      & j varying between 1 (horizontal x ) & 2 (vertical y ).");
        printf("\n");
        for( i=1; i<=nodes; i++)
        {
            for( j=1; j<=2; j++ )
            {
                printf("force[%d][%d] = ", i,j);
                scanf("%f", &force[i][j]);
                k = 2*(i-1) + j;
                node[k].forces = force[i][j];
            }
        }
        printf("");
        free_matrix(force,1,index,1,index);
    }
    else
    {
        i=1;
        printf("\nNode Direction Force");
        while( !feof( fp4 ) && ( i <= index))
        {
            fscanf(fp4,"%d %d %f",&nd,&dir,&value);
            printf("\n%2d      %5d          %5.2f",nd,dir,value);

```

```

        k = 2 * (nd-1) + dir;
        node[k].forces = value;
        i++;
    }
    if( i<index ) nerror("\nWarning! Not all forces read! Check FORCE.DAT");
    }
    fclose(fp4);
    printf("\n\n");
    return;
}

/* FORCE HISTORY */
/* ----- */
void history( float tt, float dur )
{
    float result;
    result = tt/(40*dur);
    if( result > 1 )
    {
        result = 1;
    }
    info[1].result = result;
    return ;
}

/* Newton Raphson method for multi-variable equations */
/* ===== */
void mnewt(float *u, float *extdp, int index, float ymod,
          float *uold, int ntrial, float dt)
{
    int i, k, p, q, *indx, *ivector(), nodes, icon;
    float *bet, **alpha, *vector(), **matrix();
    float errx, errf, tolx, tolf, d;
    float *xp, *yp, *x, *y;
    icon = info[1].icon;
    nodes = info[1].total;
    tolf = 0.010;
    tolx = 0.010;
    alpha = matrix(1,index,1,index);
    indx = ivector(1,index);
    bet = vector(1,index);
    xp = vector(1,index);
    yp = vector(1,index);
    x = vector(1,index);
    y = vector(1,index);
    for( i=1; i<=index; i++) /* initialize arrays */
    {
        bet[i] = 0.0;
        indx[i] = 0.0;
    }
    for( i=1; i<=index; i++) /* initiaize array */
    {
        for( k=1; k<=index; k++)
        {
            alpha[k][i] = 0.0;
        }
    }
    for( k=1; k<=ntrial; k++)
    {
        printf("After %d iteration(s)", k);
        callfunc(index, u, extdp, alpha, bet, ymod, uold, dt);
        errf = 0.0; /* check function convergence */
    }
}

```

```

for( i=1; i<=index-icon; i++ )
{
    errf += fabs(bet[i]);
}
if ( errf <= tolf) FREERETURN
ludcmp(alpha,index,indx,&d); /* solve linear equations */
lubksb(alpha,index,indx,bet); /* using LU decomposition */
errx = 0.0; /* check root convergence */
for( i=1; i<=index; i++ ) /* update solution */
{
    if( extdp[i] == 0)
    {
        bet[i] = 0;
    }
    u[i] += bet[i];
}
for( i=1; i<=nodes; i++ )
{
    p = 2*i-1;
    q = 2*i;
    x[i] = node[i].x; /* `import' coordinate values */
    y[i] = node[i].y;
/* modified values including disp. & the external supports.*/
/* increments due to iterations */
    xp[i] = x[i] + u[p];
    yp[i] = y[i] + u[q];
}
lent( xp, yp ); /* calculate length primed */
for( i=1; i<=index-icon; i++ ) /* check displacement convergence */
{
    errx += fabs(bet[i]);
}
if ( errx <= tolx) FREERETURN
printf("\nu[i] uold[i]");
for( i=1; i<=index; i++ )
{
    printf("\n%1.3f %1.3f",u[i],uold[i]);
}
printf("\n");
}
free_vector(xp,1,index);
free_vector(yp,1,index);
free_vector(x,1,index);
free_vector(y,1,index);
FREERETURN
}

/* A CALLING FUNCTION TO VW */
/* ----- */
void callfunc( int index, float *u, float *extdp, float ** alpha, float *bet,
float ymod, float *uold, float dt)
{
    int a, i, j, nod, bar, connect, icon, w, c;
    float *inithet, *newbet, *tempo, *pval, *pbet, modify;
    float *du, *value, *initval, *newval, a1, a2;
    float far *n1,*n2,*n3,*n4, *pan;
    float delta = 0.01;
    icon = info[1].icon;
    connect = info[1].connect; /* `import' value of connect */
    du = vector(1,index); /* allocate memory */
    pbet = vector(1,index);
    tempo = vector(1,index);

```

```

newbet = vector(1,index);
initbet = vector(1,index);
/* read in panel data */
if( icon != 0)
{
    n1 = vector(1,icon);
    n2 = vector(1,icon);
    n3 = vector(1,icon);
    n4 = vector(1,icon);
    pval = vector(1,index);
    value = vector(1,index);
    initval = vector(1,index);
    newval = vector(1,index);
    for( i=1; i<=icon; i++ )
    {
        n1[i] = node[i].n1;
        n2[i] = node[i].n2;
        n3[i] = node[i].n3;
        n4[i] = node[i].n4;
    }
/* ----- */
    for( i=1; i<=index; i++ )      /* initialize */
    {
        for( j=1; j<=index; j++ )
        {
            pval[i] = 0;
            value[i] = 0;
            newval[i] = 0;
            initval[i] = 0;
        }
    }
    for( j=1; j<=icon; j++ )      /* differentiate the constraint */
    {
        for( i=1; i<=index; i++ )
        {
            w = index - icon + j;
            g(u,value,index,j,n1,n2,n3,n4);
            initval[i] = value[j];
            bet[w] = -1*value[j];
            u[i] += delta;
            g(u,value,index,j,n1,n2,n3,n4);
            newval[i] = value[j];
            u[i] -= 2*delta;
            g(u,value,index,j,n1,n2,n3,n4);
            pval[i] = value[j];
            u[i] += delta;
            a1 = (newval[i]-initval[i])/delta;
            a2 = (initval[i]-pval[i])/delta;
            alpha[w][i] = (a1+a2)/2;
            alpha[i][w] = alpha[w][i];
        }
    }
    free_vector(n1,1,icon);
    free_vector(n2,1,icon);
    free_vector(n3,1,icon);
    free_vector(n4,1,icon);
    free_vector(value,1,index);
    free_vector(newval,1,index);
    free_vector(initval,1,index);
}
for( j=1; j<=index; j++ )
{

```

```

        du[j] = 0;          /* initialize all arbitrary displacements */
    }
    for( j=1; j<=(index-1con); j++ )
    {
        du[j] = 1;          /* arbitrary displacement */
        for( i=1; i<=(index-1con); i++ )
        {
            nod = i;          /* direction to be examined */
            bar = (i+1)/2;    /* node to be examined */
            vw(connect, u, bet, du, nod, bar, ymod, uold, dt, alpha);
            initbet[i] = bet[i];
            u[i] += delta;
            vw(connect, u, bet, du, nod, bar, ymod, uold, dt, alpha);
            newbet[i] = bet[i];
            u[i] -= 2*delta;
            vw(connect, u, bet, du, nod, bar, ymod, uold, dt, alpha);
            pbet[i] = bet[i];
            a1 = (newbet[i]-initbet[i])/delta;
            a2 = (initbet[i]-pbet[i])/delta;
            alpha[j][i] = (a1+a2)/2;
            u[i] += delta;
            if( i == j ) tempo[j] = initbet[i];
        }
        du[j] = 0;          /* reset displacement to zero */
    }
}
/* incorporate the external constraints */
for( i=1; i<=index; i++ )
{
    if( extdp[i] == 0 )
    {
        for( a=1; a<=index; a++ ) /* all elements in row = 0 */
        {
            alpha[i][a] = 0;
        }
        for( a=1; a<=index; a++ ) /* all elements in column = 0 */
        {
            alpha[a][i] = 0;
        }
        alpha[i][i] = 1;          /* diagonal element = 1 */
    }
}
for( i=1; i<=index-1con; i++ )
{
    if( extdp[i] == 0 )
    {
        bet[i] = 0;
    }
    else
    {
        bet[i] = -1*tempo[i];
    }
}
free_vector(du,1,index);
free_vector(pbet,1,index);
free_vector(tempo,1,index);
free_vector(newbet,1,index);
free_vector(initbet,1,index);
return;
}

/* VIRTUAL WORK ROUTINE */
/* ----- */

```

```

void vw( int connect, float *u, float *bet, float *du, int nod, int bar,
        float ymod, float *uold, float dt, float **alpha)
{
    int i, a, b, w, index, icon, c;
    float vwk, de, nn, evwk, eforce, ivw, modify;
    index = info[1].kc;
    icon = info[1].icon;
    eij( connect, u, du, uold, dt);          /* calculate strains */
    kij( connect, ymod );                   /* calculate internal forces */
    bet[nod] = 0.0; /* reset to zero */
    modify = 0.0;
    ivw = 0.0;
    nn = 0;
    de = 0;
    for( i=1; i<=connect; i++ ) /* check if there is any connection */
    {
        a = line[i].node1;
        b = line[i].node2;
        /* 'bar' specifies the NODE being considered */
        if( a == bar || b == bar ) /* Is there a connection ? */
        {
            nn = line[i].intf; /* Internal Force */
            de = line[i].dejk; /* Delta Strain */
            vwk = nn*de; /* internal virtual work */
        }
        else /* no connection */
        {
            vwk = 0;
        }
        ivw += vwk;
    }
    /* Lagrange Multipliers */
    for( c=1; c<=icon; c++ )
    {
        w = index-icon+c;
        modify += u[w]*alpha[w][nod]*du[nod];
    }
    ivw = ivw + modify;
    /* ----- */
    eforce = node[nod].forces;
    /* external virtual work */
    evwk = eforce*du[nod];
    bet[nod] = ivw - evwk; /* vw = ivw - evw */
    return;
}

/* CALCULATE THE STRAINS GIVEN THE DISPLACEMENTS */
/* ----- */
void eij(int connect, float *u, float *du, float *uold, float dt)
{
    int i, j, k, l, m, index;
    float far **uu, **uudot;
    float far **dduu, **uoldp;
    float dist[NODES], dpr[NODES], lgt[NODES], foo, dfoo;
    index = info[1].kc;
    uu = matrix(1,index,1,index);
    dduu = matrix(1,index,1,index);
    uudot = matrix(1,index,1,index);
    uoldp = matrix(1,index,1,index);
    for( i=1; i<=connect; i++ )
    {
        line[i].ejk = 0;

```

```

        line[i].dejk = 0;
        line[i].ejkdot = 0;
    }
    for( k=1; k<=connect; k++ )
    {
        i=2*k-1;
        j=2*k;
        dist[i] = line[k].dx1;
        dist[j] = line[k].dy1;
    }
    for( i=1; i<=connect; i++ )
    {
        lgt[i] = line[i].inlen;
        dpr[i] = line[i].length;
    }
    for( i=1; i<=index; i++ )
    {
        j = (i+1)/2;
        k = i+2-(2*j);
        uu[j][k] = u[i];
        dduu[j][k] = du[i];
        uoldp[j][k] = uold[i];
    }
    m=1;
    for( i=1; i<=connect; i++ )
    {
        j = line[i].node1;          /* first node of line i */
        k = line[i].node2;          /* second node of line i */
        for( l=1; l<=2; l++ )
        {
            foo= (2*dist[m]+(uu[k][l]-uu[j][l]))*(uu[k][l]-uu[j][l]);
            line[i].ejk += sqrt((lgt[i]*lgt[i]+foo)-lgt[i]);
            dfoo= 2*(dist[m]+uu[k][l]-uu[j][l])* (dduu[k][l]-dduu[j][l]);
            line[i].dejk += dfoo/(2*(lgt[i]+line[i].ejk));
            uudot[k][l] = (uu[k][l]-uoldp[k][l])/dt;
            uudot[j][l] = (uu[j][l]-uoldp[j][l])/dt;
            line[i].ejkdot += (dist[m]+uu[k][l]-uu[j][l])*(uudot[k][l]-uudot[j][l])/dpr[i];
            m+=1;
        }
    }
    free_matrix(uu,1,index,1,index);
    free_matrix(dduu,1,index,1,index);
    free_matrix(uoldp,1,index,1,index);
    free_matrix(uudot,1,index,1,index);
    return;
}

/* CALCULATE THE INTERNAL FORCES */
/* ----- */
void kij( int connect, float ymod )
{
    int i, nodes;
    float active, lamda, x[NODES], y[NODES], vel[NODES], vmax;
    float stiff[NODES], f1[NODES], f2[NODES], f3[NODES], ymod2;
    ymod2 = info[1].ymod2;
    active = info[1].active*info[1].result;
    /* calculate k from E,A, & length */
    for( i=1; i<=connect; i++ )
    {
        stiff[i] = 0;          /* reset stiffness value */
    }
}

```

```

for( i=1; i<=connect; i++ )
{
    if( line[i].type == 1 )
    {
        vel[i] = line[i].ejkdot;
        vmax = 8*line[i].fopt;
        lamda = line[i].length/line[i].fopt;
        if( lamda <= 0.6 || lamda >= 1.72 )
        {
            f1[i] = 0;
        }
        if( 0.6 < lamda < 0.8 )
        {
            f1[i] = (4.421*lamda)-2.653;
        }
        if( 0.8 <= lamda < 0.94 )
        {
            f1[i] = (1.067*lamda)-0.03;
        }
        if( 0.94 <= lamda < 1.06 )
        {
            f1[i] = 1;
        }
        if( 1.06 <= lamda < 1.72 )
        {
            f1[i] = (-1.515*lamda)+2.606;
        }
        if( (vel[i]/vmax) <= -1 )
        {
            f2[i] = 0;
        }
        if( -1 < (vel[i]/vmax) <= 0 )
        {
            f2[i] = (1+vel[i]/vmax)/(1-(4*vel[i]/vmax));
        }
        if( 0 < (vel[i]/vmax) <= 1 )
        {
            f2[i] = 2-((1-vel[i]/vmax)/(1+(4*vel[i]/vmax)));
        }
        if( (vel[i]/vmax) > 1 )
        {
            f2[i] = 2;
        }
    }
    if( lamda <= 0.6 )
    {
        f3[i] = -0.0276;
    }
    if( 0 < lamda < 1.72 )
    {
        f3[i] = 3.289*pow10(-6)*exp(9.037*lamda)-0.02766;
    }
    if( lamda >= 1.72 )
    {
        f3[i] = 18.43916933;
    }
    line[i].intf = ((active*f1[i])+f3[i])*0.4*line[i].area;
}
if( line[i].type == 2 )
{
    stiff[i] = ymod*line[i].diam*line[i].depth/line[i].inlen;    /* calculate bar stiffness */
    line[i].intf = (stiff[i]*line[i].ejk);
}
}

```

```

        if( line[i].type == 3 )
        {
            stiff[i] = ymod2*line[i].diam*line[i].depth/line[i].inlen;      /* calculate bar stiffness */
            line[i].intf = (stiff[i]*line[i].ejk);
        }
    }
    return;
}

/* CALCULATE OPTIMUM FIBRE LENGTHS */
/* ----- */
void foptimum( int connect )
{
    int i;
    float lamda;
    lamda = info[1].lambda;
    for( i=1; i<=connect; i++ )
    {
        if( line[i].type == 1 )
        {
            line[i].fopt = line[i].inlen/lamda;
            printf("\nlength[%d] = %1.3f          fopt[%d] = %1.3f",i,line[i].inlen,i,line[i].fopt);
        }
    }
    printf("\n");
    return;
}

/* CONSTRAINT EQUATION */
/* ----- */
void g( float *u, float *value, int index, int j,
        float *n1, float *n2, float *n3, float *n4 )
{
    int i, p, q, nodes;
    static far float x[NODES],y[NODES], w3;
    float far *f1, *f2, *f3, *f4, *alfa1, *alfa2, *alfa3, *alfa4, *store;
    nodes = info[1].total;
    f1 = vector(1,index);          /* allocate memory */
    f2 = vector(1,index);
    f3 = vector(1,index);
    f4 = vector(1,index);
    alfa1 = vector(1,index);
    alfa2 = vector(1,index);
    alfa3 = vector(1,index);
    alfa4 = vector(1,index);
    store = vector(1,index);
    for( i=1; i<=index; i++ )      /* initialize all variables */
    {
        f1[i]=0;
        f2[i]=0;
        f3[i]=0;
        f4[i]=0;
        alfa1[i]=0;
        alfa2[i]=0;
        alfa3[i]=0;
        alfa4[i]=0;
    }
    /* read node coordinates */
    for( i=1; i<=nodes; i++ )
    {
        p = 2*i-1;
        q = 2*i;
    }
}

```

```

        x[i] = node[i].x;          /* `import' coordinate values */
        y[i] = node[i].y;
        /* increments due to displacements */
        x[i] += u[p];
        y[i] += u[q];
    }
    /* calculate and store relevant parameters */
    flent(n1,n2,x,y,f1,j);
    flent(n3,n4,x,y,f2,j);
    flent(n1,n3,x,y,f3,j);
    flent(n2,n4,x,y,f4,j);
    angle(n1,n2,n3,x,y,alfa1,j,f1,f3);
    angle(n2,n1,n4,x,y,alfa2,j,f1,f4);
    alfa2[j] = -1*alfa2[j];
    angle(n3,n4,n1,x,y,alfa3,j,f2,f3);
    alfa3[j] = -1*alfa3[j];
    angle(n4,n3,n2,x,y,alfa4,j,f2,f4);
    w3 = 0.25*(alfa1[j]+alfa2[j]+alfa3[j]+alfa4[j]);
    line[j].angle = w3;
    store[j] = 0.125*(f1[j]+f2[j])*(f3[j]+f4[j])*(f3[j]+f4[j])*w3*w3;
    value[j] = store[j] - node[j].inval;
    free_vector(f1,1,index);
    free_vector(f2,1,index);
    free_vector(f3,1,index);
    free_vector(f4,1,index);
    free_vector(alfa1,1,index);
    free_vector(alfa2,1,index);
    free_vector(alfa3,1,index);
    free_vector(alfa4,1,index);
    free_vector(store,1,index);
return;
}

/* Relevant FUNCTIONS (--> CONSTRAINTS) */
/* ----- */

void flent(float *n1, float *n2, float x[NODES],
          float y[NODES], float *f, int j)
{
    float r,s,x1,x2,y1,y2;
    r = n1[j];
    s = n2[j];
    x1 = x[r];
    x2 = x[s];
    y1 = y[r];
    y2 = y[s];
    f[j] = sqrt(((x2-x1)*(x2-x1))+((y2-y1)*(y2-y1)));
return;
}

void angle(float *n1, float *n2, float *n3, float x[NODES],
          float y[NODES], float *alfa, int j, float *f1, float *f2)
{
    float r,s,t,ww,zz;
    r = n1[j];
    s = n2[j];
    t = n3[j];
    ww = ((x[t]-x[r])*(y[s]-y[r]))-((y[t]-y[r])*(x[s]-x[r]));
    zz = fabs(f1[j]*f2[j]);
    alfa[j] = (ww/zz);
return;
}
/* ----- */

```

```

/* OPEN OUT.DAT FILE */
/* ----- */
void out( int index, int icon )
{
    int i, count;
    fp5 = fopen("OUT.$$$","w");
    fprintf(fp5,"Time ");
    fprintf(fp5,"Activ      ");
    fprintf(fp5,"L/L0 ");
    for( i=1; i<=icon; i++ )
    {
        fprintf(fp5,"Angl%d ",i);
    }
    for( count=1; count<=icon; count++ )
    {
        i = index-icon+count;
        fprintf(fp5,"F%d ",i);
    }
    fprintf(fp5,"\n");
    fclose(fp5);
    return;
}

/* OUTPUT ROUTINE */
/* ----- */
void output( int index, int icon, float *u, float tt, float *pangle, float activation )
{
    int i, count;
    fp5 = fopen("OUT.$$$","a+");
    fprintf(fp5,"\n");
    fprintf(fp5,"%1.3f      ",tt);
    fprintf(fp5,"%1.3f      ",activation);
    fprintf(fp5,"%1.3f      ",info[1].lambda);
    for( i=1; i<=icon; i++ )
    {
        fprintf(fp5,"%1.2f      ",pangle[i]);
    }
    for( count=1; count<=icon; count++ )
    {
        i = index-icon+count;
        fprintf(fp5,"%1.2f      ",u[i]);
    }
    fclose(fp5);
    return;
}

/* EQUATION SOLVER */
/* ----- */
void lubksb(a,index,indx,b)
float **a,b[];
int index,*indx;
{
    int i,ii=0,ip,j;
    float sum;
    for( i=1; i<=index; i++ )
    {
        ip=indx[i];
        sum=b[ip];
        b[ip]=b[i];
        if( ii )
            for( j=ii;j<=i-1;j++) sum -= a[i][j]*b[j];
        else if (sum) ii=i;
    }
}

```

```

        b[i]=sum;
    }
    for( i=index; i>=1; i-- )
    {
        sum=b[i];
        for( j=i+1; j<=index; j++ ) sum -= a[i][j]*b[j];
        b[i]=sum/a[i][i];
    }
}

/* EQUATION SOLVER */
/* ----- */
void ludcmp(a,index,indx,d)
int index,*indx;
float **a,*d;
{
    int i,imax,j,k;
    float big,dum,sum,temp;
    float *vv,*vector();          /* vv stores the implicit scaling of each row */
    void nerror(),free_vector();
    vv=vector(1,index);
    *d=1.0;                       /* No row interchanges yet */
    for( i=1; i<=index; i++ )     /* Loop over rows to get the implicit scaling information */
    {
        big=0.0;
        for( j=1; j<=index; j++ )
            if ((temp=fabs(a[i][j])) > big) big=temp;
        if (big == 0.0) nerror("Singular matrix in routine LUDCMP");
        /* No nonzero largest element. */
        vv[i]=1.0/big;           /* Save the scaling */
    }
    for( j=1; j<=index; j++ )
    {
        for( i=1; i<j; i++ )
        {
            sum=a[i][j];
            for( k=1; k<i; k++ )
            {
                sum -= a[i][k]*a[k][j];
            }
            a[i][j]=sum;
        }
        big=0.0;                /* initialize the search for the largest pivot element */
        for( i=j; i<=index; i++ )
        {
            sum=a[i][j];
            for( k=1; k<j; k++ )
            {
                sum -= a[i][k]*a[k][j];
            }
            a[i][j]=sum;
            if ((dum=vv[i]*fabs(sum)) >= big)
            {
                big=dum;
                imax=i;
            }
        }
    }
    if (j != imax)              /* do we need to interchange rows ? */
    {
        for( k=1; k<=index; k++ ) /* Yes, do so ... */
        {
            dum=a[imax][k];

```

```
        a[imax][k]=a[j][k];
        a[j][k]=dum;
    }
    *d = -(*d);
    vv[imax]=vv[j];
}
indx[j]=imax;
if (a[j][j] == 0.0) a[j][j]=TINY;
/* If the pivot element is zero the matrix is singular
   ( at least to the precision of the algorithm). */
if (j != index)      /* Now, divide by the pivot element */
{
    dum=1.0/(a[j][j]);
    for( i=j+1; i<=index; i++) a[i][j] *= dum;
}
/* Go back to the next column in the reduction */
free_vector(vv,1,index);
}

#undef TINY
#undef FREEReturn
#undef NODES
```

## Appendix B:

### Sample Data Files

An example of the data files which are used in the single panel simulations of chapter 8.4 is given in this appendix. **Titles** are included in this listing to enhance the clarity, but they are **not part of the data files**.

#### N2.DAT

All the nodal coordinates are entered in this file.

<i>NODE #</i>	<i>X-coordinate</i>	<i>Y-coordinate</i>
1	0.000	0.000
2	10.686	3.288
3	19.728	-3.288
4	30.414	0.000

#### CONNECT.DAT

Information about the node connections, bar dimension and the material type is stored in this file. The material type 1 indicates a fibre, whereas 2 indicates a tendon sheath. Material type 3 indicates a tendon.

<i>Node 1</i>	<i>Node 2</i>	<i>Diameter (mm)</i>	<i>Material Type</i>	<i>Depth (mm)</i>
1	2	14.2	1	14.2
1	3	.1	2	14.2
2	4	.1	2	14.2
3	4	14.2	1	14.2

#### SUPPORT.DAT

The default setting is for all nodes to be unrestrained and free hinged. The user is prompted to input any nodal restraints in the absence of the input file "support.dat".

<i>Node #</i>	<i>Direction (x or y)</i>	<i>Support (0 -&gt; on)</i>
1	1	0
1	2	0
4	1	0
4	2	0

**FORCE.DAT**

The user is required to provide the magnitude of the applied force on each node in both the x- and y- directions. This can be done at run time in the absence of the input file "FORCE.DAT".

<i>Node #</i>	<i>X or Y direction ( 1 or 2 respectively )</i>	<i>Force Magnitude (N)</i>
1	1	0
1	2	0
2	1	0
2	2	0
3	1	0
3	2	0
4	1	0
4	2	0

**PANEL.DAT**

This file is required when an example is run which uses the volume constraint. The information provided in this file defines each panel in the structure. In the context of this thesis, the panels are always defined by quadrilaterals. At run time, the user is prompted for the number of panels.

<i>Panel #</i>	<i>Node 1</i>	<i>Node 2</i>	<i>Node 3</i>	<i>Node 4</i>
1	1	2	3	4

The order in which the nodes are specified is **very** important.