

Boundary Element Methods for Engineers: Part II

Plane Elastic Problems

Roger Fenner



Download free books at

bookboon.com

Roger Fenner

Boundary Element Methods for Engineers

Part II: Plane Elastic Problems



Boundary Element Methods for Engineers: Part II: Plane Elastic Problems

1st edition

© 2014 Roger Fenner & bookboon.com

ISBN 978-87-403-0733-7

Contents

Preface	Part I
Notation	Part I
Some Program Variable Names	Part I
1 Introduction	Part I
1.1 Continuum Mechanics Problems	Part I
1.2 Some Practical Engineering Problems	Part I
1.3 Methods for Solving Harmonic and Biharmonic Equations	Part I
2 Boundary Element Analysis of Potential Problems	Part I
2.1 Fundamental Solution	Part I
2.2 Boundary Integral Equation	Part I
2.3 Discretisation of the Boundary Integral Equation	Part I
2.4 Constant Boundary Elements	Part I

**YOU THINK.
YOU CAN WORK
AT RMB**

 **RAND
MERCHANT
BANK**
A division of FirstRand Bank Limited
Traditional values. Innovative ideas.

Rand Merchant Bank uses good business to create a better world, which is one of the reasons that the country's top talent chooses to work at RMB. For more information visit us at www.rmb.co.za

Thinking that can change your world

Rand Merchant Bank is an Authorised Financial Services Provider



Click on the ad to read more

2.5	Quadratic Boundary Elements	Part I
2.6	Scaling	Part I
2.7	Solving the Linear Equations	Part I
2.8	Solving Poisson's Equation	Part I
3	Constant Boundary Element Program for Potential Problems	Part I
3.1	Program BEM2PC	Part I
3.2	Some Test Problems for BEM2PC	Part I
3.3	An Example: Downstream Viscous Flow in a Rectangular Channel	Part I
4	Quadratic Boundary Element Program for Potential Problems	Part I
4.1	Program BEM2PQ	Part I
4.2	Some Test Problems for BEM2PQ	Part I
4.3	An Example: Downstream Viscous Flow in a Rectangular Channel	Part I
4.4	An Example: Heat Conduction in a Domain of Complex Shape	Part I
4.5	Discussion	Part I
Appendix A: Gaussian Quadrature		Part I
Appendix B: Gaussian Elimination		Part I



Discover the truth at www.deloitte.ca/careers

Deloitte.

© Deloitte & Touche LLP and affiliated entities.



Click on the ad to read more

Appendix C: Matlab Version of Constant Boundary Element Program for Potential Problems	Part I
Appendix D: Matlab Version of Quadratic Boundary Element Program for Potential Problems	Part I
Solutions to Problems – Part I	Part I
Preface	9
Notation	10
Some Program Variable Names	13
5 Boundary Element Analysis of Plane Elastic Problems	20
5.1 Fundamental Solution	21
5.2 Boundary Integral Equations	32
5.3 Discretisation of the Boundary Integral Equations	35
5.4 Boundary Conditions and Surface Stresses	37

**I WANT TO CHANGE DIRECTION,
AND THE WORLD.**

GOT-THE-ENERGY-TO-LEAD.COM

We believe that energy suppliers should be renewable, too. We are therefore looking for enthusiastic new colleagues with plenty of ideas who want to join RWE in changing the world. Visit us online to find out what we are offering and how we are working together to ensure the energy of the future.

RWE
The energy to lead

5.5	Quadratic Boundary Elements	41
5.6	Scaling	51
5.7	Solving the Linear Equations	51
5.8	Body Forces	51
6	Quadratic Boundary Element Program for Plane Elastic Problems	53
6.1	Program BEM2EQ	54
6.2	Some Test Problems for BEM2EQ	100
6.3	An Example: Confined Compression of a Rubber Block	116
6.4	An Example: Stress Concentration at a Hole in a Flat Plate	118
6.5	Discussion	122
7	Further Applications	128
7.1	Axi-symmetric Problems	128
7.2	Higher-Order Boundary Elements	128
7.3	Three-dimensional Problems	128
7.4	Non-Linear Problems	130
7.5	Comparison with Other Methods	130

bookboon.com

Corporate eLibrary

See our Business Solutions for employee learning

[Click here](#)



Appendix A: Gaussian Quadrature	132
Appendix B: Gaussian Elimination	135
Appendix E: Matlab Version of Quadratic Boundary Element Program for Plane Elastic Problems	141
Solutions to Problems – Part II	188

An advertisement for SKF. It features a woman with long dark hair smiling in the foreground, with a blurred wind turbine in the background. The text 'Brain power' is written in large white letters. To the right, there is a paragraph about wind energy and SKF's role. At the bottom left, it says 'Plug into The Power of Knowledge Engineering. Visit us at www.skf.com/knowledge'. The SKF logo is in the bottom right corner.

Brain power

By 2020, wind could provide one-tenth of our planet's electricity needs. Already today, SKF's innovative know-how is crucial to running a large proportion of the world's wind turbines.

Up to 25 % of the generating costs relate to maintenance. These can be reduced dramatically thanks to our systems for on-line condition monitoring and automatic lubrication. We help make it more economical to create cleaner, cheaper energy out of thin air.

By sharing our experience, expertise, and creativity, industries can boost performance beyond expectations. Therefore we need the best employees who can meet this challenge!

The Power of Knowledge Engineering

Plug into The Power of Knowledge Engineering.
Visit us at www.skf.com/knowledge

SKF

Preface

A few decades ago, the advent of high-speed electronic digital computers gave tremendous impetus to all numerical methods for solving engineering problems, and made it possible to solve with good accuracy many problems which previously could only be solved approximately. Finite difference methods, applied manually, gave way to finite element methods, which are still one of the most versatile and widely used, particularly in structural and solid mechanics. In thermofluids, methods of the finite volume type tend to be preferred.

Slower to develop have been boundary element methods, based on boundary integral equations. Initial development was largely in the hands of mathematicians, as the underlying mathematics are relatively sophisticated. It was engineers, however, who turned boundary element methods into practically useful and powerful techniques.

The purpose of this book is to serve as a deliberately simple introduction to boundary element methods applicable to a wide range of engineering problems. The mathematics are kept as simple as reasonably possible. Computer programs form an integral part of the boundary element approach and they are treated as such in the text. Several programs suitable for use on desktops or laptops are presented and described in detail and their uses are illustrated with the aid of a number of practical examples. Problems, with solutions, are provided at the ends of the chapters, for readers to solve for themselves.

The programming language used in the main text is Fortran. Although it is somewhat unfashionable these days for general programming purposes, Fortran is still very widely used in engineering computation. Matlab versions of the programs are also provided in Appendices. Full listings of all the programs, both Fortran and Matlab, are available for download [here](#).

A prior knowledge of either Fortran or Matlab is desirable. The level of continuum mechanics, numerical analysis, matrix algebra, vector analysis and other mathematics employed is that normally taught in undergraduate engineering courses. The book is therefore suitable for engineering undergraduates and other students at an equivalent level. Postgraduates and practising engineers may also find it useful if they are comparatively new to boundary element methods.

The book is presented in two Parts. Part I started with a brief review of the problems encountered in engineering, showing that they of two broad types. It then described boundary element treatments of problems of the potential type, using both constant and quadratic boundary elements. This Part II is concerned with elastic stress analysis problems of the plane strain and plane stress types.

Imperial College London

Professor Roger Fenner

Notation

The mathematical symbols commonly used in the main text are defined in the following list. In some cases particular symbols have more than one meaning in different parts of the book, although this should not cause any serious ambiguity.

A	area of a solution domain
$[A]$	square matrix
A_{ij}	coefficient of matrix $[A]$
a_1, a_2, a_3	constants in general boundary condition Equation 1.83
$[B]$	square matrix
B_{ij}	coefficient of matrix $[B]$
$[b]$	column vector of known coefficients
C	torsional couple
C_p	specific heat
$C_{xx}, C_{xy}, C_{yx}, C_{yy}$	free term constants
D	flexural rigidity of a flat plate
E	Young's modulus
e	strain
F_D, F_P	drag and pressure flow shape factors for downstream flow
f_1	function of position in Poisson's equation
f_2	function of position in biharmonic equation
G	shear modulus
g	acceleration due to gravity
g	heat generated per unit volume
H	height of a channel or solution domain in general
H	total rate of heat conduction
h	surface heat transfer coefficient
i	nodal point number
\mathbf{i}	unit vector in the x co-ordinate direction
J	Jacobian of transformation (from global to intrinsic co-ordinates)
j	nodal point number
\mathbf{j}	unit vector in the y co-ordinate direction
K	ratio of outer to inner radius for a cylinder
k	thermal conductivity
k	nodal point number
\mathbf{k}	unit vector in the z co-ordinate direction
L	maximum dimension of the solution domain
M	total number of boundary elements in a mesh
m	element number
N	total number of nodes in a mesh
N_c	shape function
n	direction of outward normal to the boundary of a solution domain
\mathbf{n}	outward normal vector to boundary

n_x, n_y	components of \mathbf{n} in the x and y directions
$\hat{\mathbf{n}}$	unit outward normal vector to boundary
\hat{n}_x, \hat{n}_y	components of $\hat{\mathbf{n}}$ in the x and y directions
P	source/force point on a boundary
P_z	pressure gradient in the z -co-ordinate direction
p	pressure
p	source/force point
Q	volumetric flow rate
Q	field point on a boundary
q	field point
R	a function of intrinsic co-ordinate ξ
r	radial co-ordinate
r	distance between a source/force point and a field point
\mathbf{r}	vector distance between points
$\hat{\mathbf{r}}$	unitvector in the direction between points
S	boundary of a domain
S	distance along a boundary
\mathbf{S}	vector tangent to a boundary
S	ratio between lengths of successive elements on a boundary segment
S_m	part of a boundary forming element m
s	distance along a solution domain boundary
T	temperature
T	traction kernel function
T_∞	remote temperature of surroundings in thermal convection
t	time
t	traction
U	displacement kernel function
u	displacement or velocity in the x -direction
u_r	displacement in the radial direction
u_θ	displacement in the θ -direction
\bar{u}	mean value of u
V_z	velocity component of a boundary in the z co-ordinate direction
v	displacement or velocity in the y -direction
\bar{v}	mean value of v
W	width of a channel or solution domain in general
w	displacement or velocity in the z -direction
X, Y	global Cartesian co-ordinates
$\bar{X}, \bar{Y}, \bar{Z}$	components of body forces per unit volume in the Cartesian co-ordinate directions
x, y, z	Cartesian co-ordinates
$[x]$	column vector of unknown quantities
α	coefficient of linear thermal expansion
α	coefficient in mixed boundary condition, Equation 2.32
β	coefficient in mixed boundary condition, Equation 2.32

γ	an angle
ΔT	change in temperature
ε	a small distance
η	local intrinsic co-ordinate within an element
θ	angle of rotation per unit length of a bar in torsion
θ	angular co-ordinate
θ	an angle
κ	permeability of a porous medium
μ	viscosity
ν	Poisson's ratio
ξ	local intrinsic co-ordinate within an element
π_P	dimensionless pressure gradient
π_Q	dimensionless flow rate
ρ	density
σ	stress
ϕ	velocity potential
ϕ	fundamental solution for potential
χ	stress function
ψ	stream function
ψ	potential
∇	grad operator
∇^2	harmonic (Laplacean) operator
∇^4	biharmonic operator
<i>Subscripts</i>	
c	counter for nodes within an element
e	von Mises equivalent (stress)
i, j, k	nodal point numbers
L	solution to Laplace's equation
m	element number
n	direction of the outward normal to a boundary
PI	particular integral solution satisfying Poisson's equation
r	radial direction in polar co-ordinates
s	direction along a boundary
s	segment
T	total solution (Laplace plus particular integral)
x, y, z	Cartesian co-ordinate directions
θ	angular direction in polar co-ordinates
1, 2	inner and outer of two concentric circular boundaries
1, 2, 3	nodes of a quadratic element
<i>Superscripts</i>	
*	effective value under plane stress conditions
*	modified quantity

Some Program Variable Names

The Fortran computer program variable names widely used in the programs and main text are defined in alphabetical order in the following list.

A	Coefficients of matrix $[A]$
AII	Matrix diagonal coefficient A_{ii} (potential problems)
AIIXX, AIIXY, AIIYX, AIIYY	Coefficients at the diagonal of matrix (elastic problems)
AIJ	Matrix coefficient A_{ij}
AK	First kernel function contributing to the matrix $[A]$ (potential problems)
AKXX, AKXY, AKYX, AKYY	First kernel functions contributing to the $[A]$ matrix (elastic problems)
ALPERP	Perpendicular distance from centre of curvature to mid point of a segment chord
ALPERP2	Square of ALPERP
ALPHA	Element values of constants in mixed boundary conditions
ALPHAN	Nodal point values of constants in mixed boundary conditions (quadratic elements)
ALPHASEG	Boundary segment values of constants in mixed boundary conditions
ALSEG	Length of a segment chord measured between its end points
ANG	Angular position of current end point on a curved boundary segment
ANGFIR	Angular position of first end point on a curved boundary segment
ANGSEG	Angle subtended at centre of curvature by a curved boundary segment
ANGSTORE	Angular positions of end points on curved boundary segments
AROW	Array storing element node contributions to the $[A]$ matrix (potential problems)
AROWX	Array storing element node contributions to the $[A]$ matrix (elastic problems)
AROWY	Array storing element node contributions to the $[A]$ matrix (elastic problems)
BDPSI	Coefficient of right hand side vector (matrix $[B]$ times vector of knowns)
BETA	Element values of constants in mixed boundary conditions
BETAN	Nodal point values of constants in mixed boundary conditions (quadratic elements)
BETASEG	Boundary segment values of constants in mixed boundary conditions
BIJ	Matrix coefficient B_{ij}
BK	Second kernel function contributing to the $[B]$ matrix (potential problems)
BKXX, BKXY, BKYX, BKYY	Second kernel functions contributing to the $[B]$ matrix (elastic problems)
BK2	Non-singular part of second kernel function when P is the current element node (potential problems)
BK2XX, BK2XY, BK2YX, BK2YY	Non-singular parts of second kernel functions when P is the current element node (elastic problems)

BROW	Array storing element node contributions to the $[B]$ matrix (potential problems)
BROWX	Array storing element node contributions to the $[B]$ matrix (elastic problems)
BROWY	Array storing element node contributions to the $[B]$ matrix (elastic problems)
BTX, BTY	Coefficients of right hand side column vector (matrix $[B]$ times the vector of knowns)
CASE	Alphanumeric plane stress or strain problem type
D	Perpendicular distance from P to the element containing node Q
DPSI	Nodal point values of the potential gradient solution to Laplace's equation
DPSIPI	Nodal point values of the particular integral potential gradient function
DPSIPIIM	Values of the particular integral potential gradient at the nodes of each element
DPSISEG	Values of potential gradient applied as boundary conditions to the boundary segments
DPSISTORE	Temporary store for potential gradient
DPSIT	Nodal point values of total potential gradient (Laplace plus particular integral)
DRDN	Rate of change of radius with distance along normal to boundary
DUDZ	Rate of change of displacement u with ξ along element
DVDZ	Rate of change of displacement v with ξ along element
DZDE	Jacobian of transformation from intrinsic co-ordinate ξ to η
E	Young's modulus
EGL	Values of the intrinsic co-ordinate at the Gauss points (logarithmic quadrature)
ELENGTH	Lengths of the elements
ESS	Direct strain along boundary

With us you can
shape the future.
Every single day.

For more information go to:
www.eon-career.com

Your energy shapes the future.

e-on



ESTORE	Stored value of Young's modulus
ETA	Intrinsic co-ordinate η
EVX	x component of the vector along an element
EVY	y component of the vector along an element
F1	Constant function f_1 in Poisson's equation
FLOWELEM	Potential flow across an element
FLOWIN	Total potential flow into the domain
FLOWOUT	Total potential flow out of the domain
FLOWSEG	Flows of potential across the boundary segments
FXELEM	Force on an element in x direction
FXSEG	Total force on a boundary segment in x direction
FYELEM	Force on an element in y direction
FYSEG	Total force on a boundary segment in y direction
HX	Interval between points in the x direction used in domain integration
HY	Interval between points in the y direction used in domain integration
I	Node counter
I1, I2, I3	Numbers of the three nodes of a quadratic element
IBC	Type number of boundary conditions applied to the (constant) elements
IBCD	Counter for segments subject to applied potential gradient boundary conditions
IBCE	Type number of boundary conditions applied to the (quadratic) elements
IBCM	Counter for segments subject to applied mixed boundary conditions
IBCN	Type number of boundary conditions applied to the nodes (of quadratic elements)
IBCP	Counter for segments subject to applied potential boundary conditions
IBCPC	Counter for point displacement constraints
IBCS	Counter for segments subject to applied stress boundary conditions
IBCU	Counter for segments subject to applied displacement boundary conditions
IBOUND	Counter for boundaries
IC	Case number for logarithmic Gaussian quadrature
IDIRPC	Direction numbers of point displacement constraints
IEEND	Counter for element end points
IEP1	Counter for first end point of an element
IEP2	Counter for second end point of an element
IFIRST	Numbers of first nodes on the segments
IFLAG	Flag for ill-conditioning of the $[A]$ matrix
IGAUSS	Counter for Gauss points
IINT	Counter for internal points
ILAST	Numbers of last nodes on the segments
IN	Counter for nodes within an element
IROW	Number of row in the $[A]$ matrix

ISEG	Segment counter
ISEGBC	Segment numbers for a particular type of boundary condition
ISEGELEM	Segment numbers for elements
ISEGEND	Segment numbers for element end points
ISEGMAX	Number of last segment on current boundary
ISEGMIN	Number of first segment on current boundary
ISEND	Counter for boundary segment end points
IT	Number indicating type of Gaussian quadrature (normal or logarithmic)
IX	Counter for points in the x direction used in domain integration
IXMAX	Maximum value of IX
IY	Counter for points in the y direction used in domain integration
IYMAX	Maximum value of IY
J	Node counter
JACOB	Jacobian of transformation from global to local intrinsic ξ co-ordinate
JMAX	Maximum number of columns in the extended $[A]$ matrix
M	Element counter
M1	Numbers of the elements adjacent to the first node of each element
M3	Numbers of the elements adjacent to the third node of each element
MAXL	Maximum dimension of the solution domain
MAXNB	Maximum number of boundaries allowed by the array dimensions
MAXNEL	Maximum number of elements
MAXNEQN	Maximum number of equations
MAXNNP	Maximum number of nodal points allowed by the array dimensions
MAXNPC	Maximum number of point displacement constraints
MFIRST	Numbers of the first elements on the segments
MLAST	Numbers of the last elements on the segments
MMAX	Number of last element on current boundary
MMIN	Number of first element on current boundary
NBCD	Number of segments subject to applied potential gradient boundary conditions
NBCM	Number of segments subject to applied mixed boundary conditions
NBCP	Number of segments subject to applied potential boundary conditions
NBCPC	Number of point displacement constraints
NBCS	Number of segments subject to applied stress boundary conditions
NBCT	Total number of segments subject to applied boundary conditions
NBCU	Number of segments subject to applied displacement boundary conditions
NBOUND	Number of boundaries
NEEND	Number of element end points
NEL	Number of elements
NELB	Numbers of elements on the boundaries

NELSEG	Number of elements on current boundary segment
NEP1	Numbers of the first end points of the elements
NEP2	Numbers of the second end points of the elements
NEQN	Number of equations
NGAUSS	Number of Gauss points
NINT	Number of internal points
NNP	Number of nodal points
NNPB	Numbers of nodal points on each of the boundaries
NODE	Numbers of the nodes of the elements
NODEPC	Numbers of nodes subjected to point displacement constraints
NSEGB	Numbers of boundary segments on each of the boundaries
NSEGTOT	Total number of boundary segments
NU	Poisson's ratio
NX	Number of internal points in the x direction used in domain integration
NY	Number of internal points in the y direction used in domain integration
PI	π
PSI	Nodal point values of the potential solution to Laplace's equation
PSIBOT	Value of potential on the bottom edge of a rectangular domain
PSIIP	Laplace equation potential at an internal point
PSIPT	Total potential at an internal point (Laplace plus particular integral)

be > your degree

Bring your talent and passion to a global organization at the forefront of business, technology and innovation. Discover how great you can be. Visit accenture.com/bookboon

Be greater than.
consulting | technology | outsourcing

accenture
High performance. Delivered.

© 2013 Accenture. All rights reserved.



PSILEFT	Value of potential on the left hand edge of a rectangular domain
PSIPI	Nodal point values of the particular integral potential function
PSIRIGHT	Value of potential on the right hand edge of a rectangular domain
PSISEG	Values of potential applied as boundary conditions to the boundary segments
PSIT	Nodal point values of total potential (Laplace plus particular integral)
PSITOP	Value of potential on the top edge of a rectangular domain
PSIVAL	Values of potential stored for domain integration
R1	Distance from point P to the first end of element containing node Q
R1X	x component of radius vector from P to the first end of element containing Q
R1Y	y component of radius vector from P to the first end of element containing Q
R2	Distance from point P to the second end of element containing node Q
R2X	x component of radius vector from P to the second end of element containing Q
R2Y	y component of radius vector from P to the second end of element containing Q
RATSEG	Ratio between successive element lengths on current boundary segment
RFN	Value of function $R(\xi)$
RSEG	Radius of curvature of current boundary segment
RX	Component in x direction of unit radius vector from P to Q
RY	Component in y direction of unit radius vector from P to Q
SD	Shape function derivatives for quadratic elements (normal quadrature)
SDL	Shape function derivative values for quadratic elements (logarithmic quadrature)
SF	Shape function values for quadratic elements (normal quadrature)
SFL	Shape function values for quadratic elements (logarithmic quadrature)
SFN	Shape function value at a Gauss point
SIGE	Nodal point values of von Mises equivalent stress
SIGNN	Nodal point values of direct stress normal to boundary
SIGNNSEG	Boundary segment values of direct stress normal to boundary
SIGSN	Nodal point values of shear stress along boundary
SIGSNSEG	Boundary segment values of shear stress along boundary
SIGSS	Nodal point values of direct stress along boundary
STORE	Stored values in the boundary condition application process
TITLE	Alphanumeric title for the problem (maximum 80 characters)
TMX	Element nodal point values of traction in x direction
TX	Nodal point values of traction in x direction
TMY	Element nodal point values of traction in y direction
TY	Nodal point values of traction in y direction
U	Nodal point values of displacement in x direction
UELEM	Element values of displacement in x direction
UNGX	x component of the unit normal at a Gauss point
UNGY	y component of the unit normal at a Gauss point
UNMX	x components of the unit normals at the nodes of each element

UNMY	y components of the unit normals at the nodes of each element
UNX	x components of the unit normals at the nodes
UNY	y components of the unit normals at the nodes
USEG	Boundary segment values of displacement in x direction
UV	Nodal point values of computed displacements (or tractions)
V	Nodal point values of displacement in y direction
VELEM	Element values of displacement in y direction
VSEG	Boundary segment values of displacement in y direction
WG	Values of the Gaussian weighting factors (normal quadrature)
WGL	Values of the Gaussian weighting factors (logarithmic quadrature)
XC	x co-ordinate of the origin for the particular integral function
XCENT	x co-ordinate of the centre of curvature of a curved boundary segment
XEEND	x co-ordinates of the element end points
XFIRST	x co-ordinate of first end point of current boundary segment
XINT	x co-ordinate of an internal point
XLAST	x co-ordinate of last end point of current boundary segment
XMID	x co-ordinate of the mid point between the ends of a curved segment
XNODE	x co-ordinates of the nodes
XP	x co-ordinate of point
XPOINT	Global x co-ordinate of an internal point
XQ	x co-ordinate of Gauss point
XSEND	x co-ordinates of the boundary segment end points
XX	x co-ordinate relative to the origin for the particular integral
YC	y co-ordinate of the origin for the particular integral function
YCENT	y co-ordinate of the centre of curvature of a curved boundary segment
YEEND	y co-ordinates of the element end points
YFIRST	y co-ordinate of first end point of current boundary segment
YINT	y co-ordinate of an internal point
YINTGL	Values of y direction integrals stored for domain integration
YLAST	y co-ordinate of last end point of current boundary segment
YMID	y co-ordinate of the mid point between the ends of a curved segment
YNODE	y co-ordinates of the nodes
YP	y co-ordinate of point P
YPOINT	Global co-ordinate of an internal point
YQ	y co-ordinate of Gauss point Q
YSEND	y co-ordinates of the boundary segment end points
YY	y co-ordinate relative to the origin for the particular integral
ZETA	Intrinsic co-ordinate ξ
ZG	Values of the intrinsic co-ordinate ξ at the Gauss points (normal quadrature)

5 Boundary Element Analysis of Plane Elastic Problems

In this chapter a form of boundary element analysis for two-dimensional elastic stress analysis problems such as those outlined in Chapter 1 is presented. Only quadratic elements are considered. Three-dimensional problems are discussed briefly in Section 7.3.

In Sections 1.2.5 and 1.2.6 it was shown that both plane strain and plane stress problems can be described in terms of a fourth-order biharmonic differential equation for Airy's stress function. The relevant Equations are 1.71 and 1.77, both of which are special cases of Equation 1.85. Clearly, such problems are fundamentally different from potential problems, which are governed by the second-order Equation 1.84.

Using an Airy stress function approach to solving plane elastic problems can be very appropriate when the boundary conditions are defined in terms of stresses. In more general problems of practical engineering interest, however, boundary conditions are typically defined in terms of a mixture of stresses and displacements, and a different approach is to be preferred.



"I studied English for 16 years but...
...I finally learned to speak it in just six lessons"

Jane, Chinese architect

ENGLISH OUT THERE

Click to hear me talking before and after my unique course download

Following the methodology developed in Chapter 2 for potential problems, what are required are a fundamental solution, equivalent to Equation 2.13, and a relationship equivalent to Green's symmetric identity, expressed in Equation 2.25.

5.1 Fundamental Solution

In the case of potential problems, the fundamental solution (Section 2.1) was in practical terms that due to a source of potential concentrated at a point in a solution domain of infinite extent in all directions. A mathematical requirement of a fundamental solution is that its value is singular (goes to infinity) at a point – the point where the source is located.

In elasticity problems the corresponding fundamental solution is that due to a force concentrated at a point in an infinite domain, which again has the property of singularity at the point concerned. Immediately there is a substantial difference between the two classes of problems. The fundamental solution for potential problems involves only a scalar quantity: the source strength, and the effects of the point source in terms of potential distribution will be the same in all directions moving away from the point. In contrast, the fundamental solution for elasticity problems involves a vector quantity: the point force has both magnitude and direction. The effect of the force in terms of stresses and displacements is certainly not the same in all directions moving away from the point.

In three dimensions, the fundamental solution is truly that due to a concentrated point force. In two dimensions, however, the problem is either one of plane stress, when the solution domain is very thin in the third dimension normal to the domain (Section 1.2.6), or plane strain, when the solution domain is very thick in the third dimension (Section 1.2.5). In either case, the fundamental solution needs to be thought of as that due to a force uniformly distributed along a line through the domain thickness in the third dimension. The force is a line force per unit thickness, which is applicable to either plane stress or plane strain. In the plane of the solution domain it still appears as a force acting at a point.

Rather than deriving the fundamental solution from first principles, the approach adopted here is to state the result, and then show that this satisfies all the relevant requirements. Figure 5.1 shows a plane with both Cartesian (x, y) co-ordinates and polar (r, θ) co-ordinates, both with the same origin, and the angular co-ordinate measured anti-clockwise from the r direction. Stress components σ_{rr} (radial direct stress), $\sigma_{\theta\theta}$ (hoop direct stress) and $\sigma_{r\theta}$ (shear stress) are shown acting on a small region of the domain. As in Equation 1.1, shear stresses are complementary, so that $\sigma_{r\theta} = \sigma_{\theta r}$, and $\sigma_{r\theta}$ is shown acting on all four faces of the region.

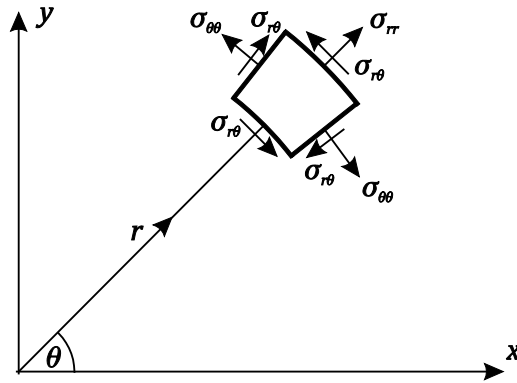


Figure 5.1 Stress components in polar co-ordinates

Due to a line force of unit magnitude (per unit length) acting in the x direction at the origin, the three stress components at distance r from the origin and angular location θ are given by

$$\sigma_{rr} = -\frac{(3+\nu^*) \cos \theta}{4\pi r} \quad (5.1)$$

$$\sigma_{\theta\theta} = \frac{(1-\nu^*) \cos \theta}{4\pi r} \quad (5.2)$$

$$\sigma_{r\theta} = \frac{(1-\nu^*) \sin \theta}{4\pi r} \quad (5.3)$$

The displacement components, u_r in the radial direction and u_θ in the hoop direction, are given by

$$u_r = \frac{(1+\nu^*)(3-\nu^*) \cos \theta}{4\pi E^*} \ln \left(\frac{d}{r} \right) \quad (5.4)$$

$$u_\theta = \frac{\sin \theta}{4\pi E^*} \left[(1+\nu^*)^2 - (1+\nu^*)(3-\nu^*) \ln \left(\frac{d}{r} \right) \right] \quad (5.5)$$

where the length d is arbitrary, and remains to be chosen. It appears because the problem (of a concentrated force in an infinite medium) has had no displacement boundary conditions defined.

The parameters E^* and ν^* in these equations are the effective Young's modulus and Poisson's ratio for the elastic material of the domain. Their definitions depend on whether the problem is one of plane stress or of plane strain. For plane stress

$$E^* = E \quad \text{and} \quad \nu^* = \nu \quad (5.6)$$

while for plane strain

$$E^* = \frac{E}{(1-\nu^2)} \quad \text{and} \quad \nu^* = \frac{\nu}{(1-\nu)} \quad (5.7)$$

In other words, under plane stress the effective properties are the actual properties, while under plane strain they are modified according to Equations 5.7. A plane strain problem can be treated as plane stress, provided the modified properties are used.

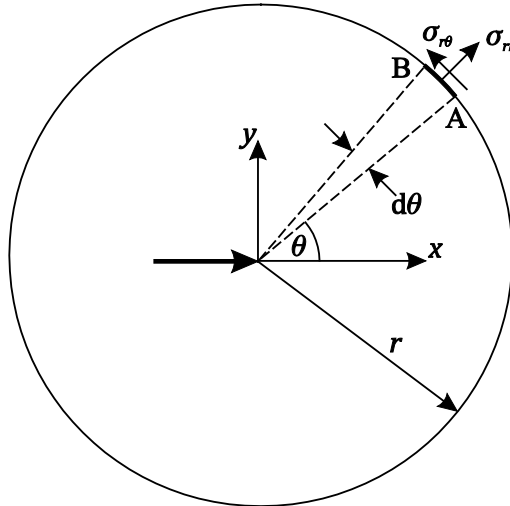


Figure 5.2 Stresses acting on a circle about the point of force application



Figure 5.2 shows a circular region of radius r of the domain centred at the origin. The stresses acting on a small arc AB of the outer surface of the region of angular extent $d\theta$ are σ_{rr} and $\sigma_{r\theta}$ as shown. For unit thickness of domain in the direction normal to the plane shown, the forces on AB in the radial and tangential directions are $\sigma_{rr} \times r d\theta$ and $\sigma_{r\theta} \times r d\theta$, respectively, and the total force on the region in the negative x direction, which should be equal to the applied line force, is

$$Force = \int_0^{2\pi} (-\sigma_{rr} \cos \theta + \sigma_{r\theta} \sin \theta) r d\theta \quad (5.8)$$

$$= \int_0^{2\pi} \frac{1}{4\pi} [(3 + \nu^*) \cos^2 \theta + (1 - \nu^*) \sin^2 \theta] d\theta \quad (5.9)$$

Now

$$\int_0^{2\pi} \cos^2 \theta d\theta = \int_0^{2\pi} \sin^2 \theta d\theta = \pi \quad (5.10)$$

so that Equation 5.9 becomes

$$Force = \frac{1}{4\pi} [(3 + \nu^*)\pi + (1 - \nu^*)\pi] = \frac{1}{4}(3 + 1) = 1 \quad (5.11)$$

The force is of unit magnitude, as required. An integral expression similar to Equation 5.8 can be used to show that the total force in the y direction is zero. Note that the requirement that the total forces on the circular region be constant, independent of radius, means that all the stresses are proportional to r^{-1} .

In plane polar co-ordinates, the strains defined in terms of displacements (equivalent to Equations 1.2 and 1.3) are

$$e_{rr} = \frac{\partial u_r}{\partial r} = -\frac{(1+\nu^*)(3-\nu^*) \cos \theta}{4\pi E^* r} \quad (5.12)$$

$$e_{\theta\theta} = \frac{u_r}{r} + \frac{1}{r} \frac{\partial u_\theta}{\partial r} = \frac{(1+\nu^*)^2 \cos \theta}{4\pi E^* r} \quad (5.13)$$

$$e_{r\theta} = \frac{1}{r} \frac{\partial u_r}{\partial \theta} + r \frac{\partial}{\partial r} \left(\frac{u_\theta}{r} \right) = \frac{2(1+\nu^*)(1-\nu^*) \sin \theta}{4\pi E^* r} \quad (5.14)$$

Identical results should be obtained by applying the elastic constitutive equations (equivalent to Equations 1.20 to 1.25) to the stresses defined in Equations 5.1 to 5.3.

Plane stress

Under plane stress conditions and in the absence of temperature changes, the direct stress σ_{zz} normal to the plane of the domain is zero, and

$$e_{rr} = \frac{1}{E} (\sigma_{rr} - \nu \sigma_{\theta\theta}) = \frac{\cos \theta}{4\pi E r} [-(3 + \nu) - \nu(1 - \nu)] = -\frac{(1+\nu)(3-\nu) \cos \theta}{4\pi E r} \quad (5.15)$$

which, in view of Equations 5.6, is identical to Equation 5.12. Similarly

$$e_{\theta\theta} = \frac{1}{E}(\sigma_{\theta\theta} - \nu\sigma_{rr}) = \frac{\cos \theta}{4\pi E r} [(1 - \nu) + \nu(3 + \nu)] = \frac{(1+\nu)^2 \cos \theta}{4\pi E r} \quad (5.16)$$

which is identical to Equation 5.13. Finally

$$e_{r\theta} = \frac{\sigma_{r\theta}}{G} = \frac{2(1+\nu)}{E} \sigma_{r\theta} = \frac{2(1+\nu)(1-\nu) \sin \theta}{4\pi E r} \quad (5.17)$$

which is identical to Equation 5.14.

Plane strain

Under plane strain conditions, and again in the absence of temperature changes, the direct strain e_{zz} normal to the plane of the domain is zero, from which

$$e_{zz} = \frac{1}{E}[\sigma_{zz} - \nu(\sigma_{rr} + \sigma_{\theta\theta})] = 0 \quad (5.18)$$

$$\sigma_{zz} = \nu(\sigma_{rr} + \sigma_{\theta\theta}) \quad (5.19)$$

Then, using Equations 5.7

$$\begin{aligned} e_{rr} &= \frac{1}{E}[\sigma_{rr} - \nu(\sigma_{\theta\theta} + \sigma_{zz})] = \frac{1}{E}[\sigma_{rr} - \nu(\sigma_{\theta\theta} + \nu\sigma_{rr} + \nu\sigma_{\theta\theta})] \\ &= \frac{1}{E}[(1 - \nu^2)\sigma_{rr} - \nu(1 + \nu)\sigma_{\theta\theta}] \\ &= \frac{1}{E^*} \left[\sigma_{rr} - \frac{\nu}{1-\nu} \sigma_{\theta\theta} \right] = \frac{1}{E^*} [\sigma_{rr} - \nu^* \sigma_{\theta\theta}] = -\frac{(1+\nu^*)(3-\nu^*) \cos \theta}{4\pi E^* r} \end{aligned} \quad (5.20)$$

as in Equation 5.12. Also

$$\begin{aligned} e_{\theta\theta} &= \frac{1}{E}[\sigma_{\theta\theta} - \nu(\sigma_{zz} + \sigma_{rr})] = \frac{1}{E}[\sigma_{\theta\theta} - \nu(\nu\sigma_{rr} + \nu\sigma_{\theta\theta} + \sigma_{rr})] \\ &= \frac{1}{E}[(1 - \nu^2)\sigma_{\theta\theta} - \nu(1 + \nu)\sigma_{rr}] \\ &= \frac{1}{E^*} \left[\sigma_{\theta\theta} - \frac{\nu}{1-\nu} \sigma_{rr} \right] = \frac{1}{E^*} [\sigma_{\theta\theta} - \nu^* \sigma_{rr}] = \frac{(1+\nu^*)^2 \cos \theta}{4\pi E^* r} \end{aligned} \quad (5.21)$$

as in Equation 5.13. Finally

$$e_{r\theta} = \frac{\sigma_{r\theta}}{G} = \frac{2(1+\nu)}{E} \sigma_{r\theta} = \frac{2(1+\nu^*)(1-\nu^*) \sin \theta}{4\pi E^* r} \quad (5.22)$$

as in Equation 5.14.

Equations 5.15 to 5.17 and 5.20 to 5.22 show that the expressions for strains, under either plane stress or plane strain conditions, obtained from the stresses via the constitutive equations, are identical to those obtained from the displacements. With the stresses also satisfying equilibrium with the applied line force, this means that Equations 5.1 to 5.5 represent the true solution for a line force at a point in an infinite plane, in other words the fundamental solution for plane elastic problems.

5.1.1 Displacement kernel functions

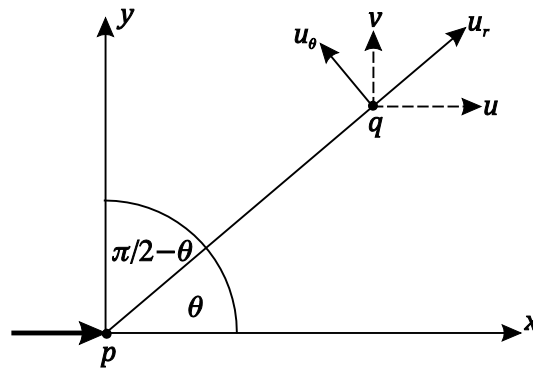


Figure 5.3 Displacements at the field point

These results now need to be generalised for line forces applied in both the x and y directions, to give the displacements and tractions in the same pair of directions. The term traction will be explained shortly. Figure 5.3 shows the displacements at a field point q produced by a line force of unit magnitude in the x direction at force point p . Displacements u_r and u_θ are given by Equations 5.4 and 5.5. The corresponding displacement in the x direction at point q is

$$\begin{aligned}
 u &= u_r \cos \theta - u_\theta \sin \theta \\
 &= \frac{(1 + \nu^*)(3 - \nu^*) \cos^2 \theta}{4\pi E^*} \ln\left(\frac{d}{r}\right) - \frac{\sin^2 \theta}{4\pi E^*} \left[(1 + \nu^*)^2 - (1 + \nu^*)(3 - \nu^*) \ln\left(\frac{d}{r}\right) \right] \\
 &= \frac{(1 + \nu^*)(3 - \nu^*)}{4\pi E^*} \ln\left(\frac{d}{r}\right) - \frac{(1 + \nu^*)^2 \sin^2 \theta}{4\pi E^*} \\
 &= \frac{(1 + \nu^*)^2}{4\pi E^*} \left[\frac{(3 - \nu^*)}{(1 + \nu^*)} \ln\left(\frac{d}{r}\right) - 1 + \cos^2 \theta \right] \quad (5.23)
 \end{aligned}$$

It is convenient to choose the arbitrary length d such that

$$u = \frac{(1 + \nu^*)^2}{4\pi E^*} \left[\frac{(3 - \nu^*)}{(1 + \nu^*)} \ln\left(\frac{1}{r}\right) + \cos^2 \theta \right] \quad (5.24)$$

$$\text{Now } \cos \theta = \hat{r}_x, \quad \sin \theta = \hat{r}_y \quad (5.25)$$

where \hat{r}_x and \hat{r}_y are the components in the x and y directions of the radius vector of unit length in the direction from point p to point q . Equation 5.24 can be written as

$$u = U_{xx}(p, q) = \frac{(1+\nu^*)^2}{4\pi E^*} \left[\frac{(3-\nu^*)}{(1+\nu^*)} \ln\left(\frac{1}{r}\right) + \hat{r}_x \hat{r}_x \right] r(p, q) \neq 0 \quad (5.26)$$

$U_{xx}(p, q)$ is the displacement kernel function defining the displacement in the x direction at q due to a unit line force in the x direction at p .

Again from Figure 5.3, the displacement in the y direction at point p is

$$\begin{aligned} v &= u_r \sin \theta + u_\theta \cos \theta \\ &= \frac{(1+\nu^*)(3-\nu^*) \cos \theta \sin \theta}{4\pi E^*} \ln\left(\frac{d}{r}\right) + \frac{\cos \theta \sin \theta}{4\pi E^*} \left[(1+\nu^*)^2 - (1+\nu^*)(3-\nu^*) \ln\left(\frac{d}{r}\right) \right] \\ v &= U_{xy}(p, q) = \frac{(1+\nu^*)^2}{4\pi E^*} \cos \theta \sin \theta = \frac{(1+\nu^*)^2}{4\pi E^*} \hat{r}_x \hat{r}_y \end{aligned} \quad (5.27)$$

$U_{xy}(p, q)$ is the displacement kernel function defining the displacement in the y direction at q due to a unit line force in the x direction at p .

Join American online LIGS University!

Interactive Online programs
BBA, MBA, MSc, DBA and PhD

Special Christmas offer:

- ▶ enroll **by December 18th, 2014**
- ▶ **start studying and paying only in 2015**
- ▶ **save up to \$ 1,200** on the tuition!
- ▶ Interactive Online education
- ▶ visit ligsuniversity.com to find out more!

Note: LIGS University is not accredited by any nationally recognized accrediting agency listed by the US Secretary of Education. More info [here](http://ligsuniversity.com).



For a line force of unit magnitude in the y direction at point p (Figure 5.3), similar results can be obtained. The angular position of field point q relative to the line of action of the force is now not θ but $-\left(\frac{\pi}{2} - \theta\right)$, so $\sin \theta$ becomes $-\cos \theta$, and $\cos \theta$ becomes $\sin \theta$. Using Equation 5.24, the displacement in the direction of the applied force at q is

$$v = U_{yy}(p, q) = \frac{(1+\nu^*)^2}{4\pi E^*} \left[\frac{(3-\nu^*)}{(1+\nu^*)} \ln\left(\frac{1}{r}\right) + \sin^2 \theta \right] = \frac{(1+\nu^*)^2}{4\pi E^*} \left[\frac{(3-\nu^*)}{(1+\nu^*)} \ln\left(\frac{1}{r}\right) + \hat{r}_y \hat{r}_y \right] \quad (5.28)$$

$$r(p, q) \neq 0$$

Using Equation 5.27, the displacement in the direction at $\pi/2$ anti-clockwise from the direction of the applied force at q is

$$-u = -\frac{(1+\nu^*)^2}{4\pi E^*} \sin \theta \cos \theta$$

and
$$u = U_{yx}(p, q) = \frac{(1+\nu^*)^2}{4\pi E^*} \hat{r}_y \hat{r}_x \quad (5.29)$$

The four equations, Equations 5.26 to 5.29, define the four displacement kernels. They can be expressed elegantly in a single equation if tensor notation is introduced. For present purposes, however, it is more convenient to have them in their more explicit, if more lengthy, forms. The first subscript of U indicates the direction of the unit line force at p , while the second subscript indicates the direction of the resulting displacement at q .

5.1.2 Traction kernel functions

The displacement kernel functions are expressed in terms of displacements at the field point in the global co-ordinate directions. Similar results are required for stresses. For this purpose, the concept of tractions is introduced. A traction is the resultant stress, or force per unit area, on a surface in a particular direction.

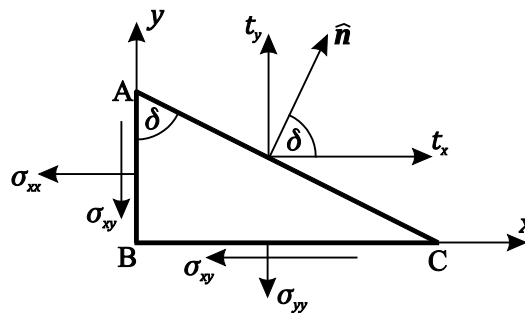


Figure 5.4 Tractions and stresses at a surface within a domain

Figure 5.4 shows both the Cartesian stress components and the equivalent tractions, t_x and t_y in the x and y co-ordinate directions, acting on a surface of a small triangular piece of material within a domain. The unit outward normal vector to the surface, \hat{n} , is inclined at angle δ to the x direction. For equilibrium of forces in the x direction (bearing in mind that the domain is of unit thickness)

$$t_x \times (AC) = \sigma_{xx} \times (AB) + \sigma_{xy} \times (BC)$$

$$t_x = \sigma_{xx} \cos \delta + \sigma_{xy} \sin \delta = \sigma_{xx} \hat{n}_x + \sigma_{xy} \hat{n}_y \quad (5.30)$$

where \hat{n}_x and \hat{n}_y are the components in the co-ordinate directions of the unit normal vector, and AC, AB and BC are the lengths of the sides of the small triangle. Similarly, for equilibrium of forces in the y direction

$$t_y \times (AC) = \sigma_{yy} \times (BC) + \sigma_{xy} \times (AB)$$

$$t_y = \sigma_{yy} \sin \delta + \sigma_{xy} \cos \delta = \sigma_{yy} \hat{n}_y + \sigma_{xy} \hat{n}_x \quad (5.31)$$

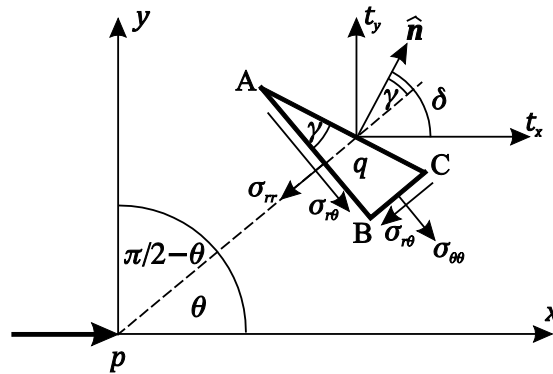


Figure 5.5 Stress components in polar co-ordinates

Figure 5.5 shows the polar co-ordinate stress components at field point q due to a line force in the x direction at p . Also the equivalent tractions in the global co-ordinate directions, t_x and t_y , acting on a surface of a small triangular piece of domain whose outward normal is inclined at angle γ to radius r , and angle δ to the x direction. For equilibrium of forces on the right-angled triangular piece in the x direction

$$t_x \times (AC) = \sigma_{rr} \times (AB) \cos \theta - \sigma_{r\theta} \times (AB) \sin \theta - \sigma_{\theta\theta} \times (BC) \sin \theta + \sigma_{r\theta} \times (BC) \cos \theta$$

$$t_x = \sigma_{rr} \cos \gamma \cos \theta - \sigma_{r\theta} \cos \gamma \sin \theta - \sigma_{\theta\theta} \sin \gamma \sin \theta + \sigma_{r\theta} \sin \gamma \cos \theta$$

With the stresses defined by Equations 5.1 to 5.3, this becomes

$$\begin{aligned}
 t_x &= -\frac{(3 + \nu^*) \cos^2 \theta \cos \gamma}{4\pi r} - \frac{(1 - \nu^*) \sin^2 \theta \cos \gamma}{4\pi r} \\
 &= -\frac{1}{4\pi r} [(1 - \nu^*) + 2(1 + \nu^*) \cos^2 \theta] \cos \gamma \\
 t_x &= -\frac{1}{4\pi r} [(1 - \nu^*) + 2(1 + \nu^*) \cos^2 \theta] \frac{dr}{dn}
 \end{aligned} \tag{5.32}$$

where $\frac{dr}{dn} = \cos \gamma$. The traction kernel function for defining the traction in the x direction at q due to a unit line force in the x direction at p is

$$T_{xx}(p, q) = -\frac{1}{4\pi r} [(1 - \nu^*) + 2(1 + \nu^*) \hat{r}_x \hat{r}_x] \frac{dr}{dn} \quad r(p, q) \neq 0 \tag{5.33}$$

Similarly, for equilibrium of forces in the y direction

$$\begin{aligned}
 t_y \times (AC) &= \sigma_{rr} \times (AB) \sin \theta + \sigma_{r\theta} \times (AB) \cos \theta + \sigma_{\theta\theta} \times (BC) \cos \theta + \sigma_{r\theta} \times (BC) \sin \theta \\
 t_y &= \sigma_{rr} \cos \gamma \sin \theta + \sigma_{r\theta} \cos \gamma \cos \theta + \sigma_{\theta\theta} \sin \gamma \cos \theta + \sigma_{r\theta} \sin \gamma \sin \theta \\
 &= -\frac{1}{4\pi r} [(3 + \nu^*) - (1 - \nu^*)] \cos \theta \sin \theta \cos \gamma + \frac{(1 - \nu^*)}{4\pi r} \sin \gamma
 \end{aligned} \tag{5.34}$$



ie business school

#1 EUROPEAN BUSINESS SCHOOL
FINANCIAL TIMES 2013

#gobeyond

MASTER IN MANAGEMENT

Because achieving your dreams is your greatest challenge. IE Business School's Master in Management taught in English, Spanish or bilingually, trains young high performance professionals at the beginning of their career through an innovative and stimulating program that will help them reach their full potential.

- Choose your area of specialization.
- Customize your master through the different options offered.
- Global Immersion Weeks in locations such as Rio de Janeiro, Shanghai or San Francisco.

Because you change, we change with you.

www.ie.edu/master-management | mim.admissions@ie.edu | f t in YouTube

Now $\sin \gamma = \sin(\delta - \theta) = \sin \delta \cos \theta - \cos \delta \sin \theta$

$$= \hat{n}_y \hat{r}_x - \hat{n}_x \hat{r}_y \quad (5.35)$$

where \hat{n}_x and \hat{n}_y are the components in the x and y directions of the unit outward normal to surface AC at point q . The kernel function for defining the traction in the y direction at q due to a unit line force in the x direction at p is

$$T_{xy}(p, q) = -\frac{1}{4\pi r} \left[2(1 + \nu^*) \hat{r}_x \hat{r}_y \frac{dr}{dn} - (1 - \nu^*) (\hat{r}_x \hat{n}_y - \hat{r}_y \hat{n}_x) \right] \quad (5.36)$$

$$r(p, q) \neq 0$$

For a line force of unit magnitude in the y direction at point p (Figure 5.5), similar results can be obtained. The angular position of field point q relative to the line of action of the force is now not θ but $-\left(\frac{\pi}{2} - \theta\right)$, so $\sin \theta$ becomes $-\cos \theta$, and $\cos \theta$ becomes $\sin \theta$. Also, \hat{n}_x becomes \hat{n}_y and \hat{n}_y becomes $-\hat{n}_x$. Using Equation 5.32, the traction in the direction of the applied force at q is

$$t_y = -\frac{1}{4\pi r} [(1 - \nu^*) + 2(1 + \nu^*) \sin^2 \theta] \frac{dr}{dn} \quad (5.37)$$

$$\text{and} \quad T_{yy}(p, q) = -\frac{1}{4\pi r} [(1 - \nu^*) + 2(1 + \nu^*) \hat{r}_y \hat{r}_y] \frac{dr}{dn} \quad r(p, q) \neq 0 \quad (5.38)$$

Using Equation 5.34, the traction in the direction at $\pi/2$ anti-clockwise from the direction of the applied force at q is

$$\begin{aligned} -t_x &= -\frac{1}{4\pi r} \left[2(1 + \nu^*) (-\sin \theta) \cos \theta \frac{dr}{dn} - (1 - \nu^*) (-\hat{n}_x \sin \theta - \hat{n}_y (-\cos \theta)) \right] \\ t_x &= -\frac{1}{4\pi r} \left[2(1 + \nu^*) \sin \theta \cos \theta \frac{dr}{dn} - (1 - \nu^*) (\hat{n}_x \sin \theta - \hat{n}_y \cos \theta) \right] \end{aligned} \quad (5.39)$$

$$\text{and} \quad T_{yx}(p, q) = -\frac{1}{4\pi r} \left[2(1 + \nu^*) \hat{r}_y \hat{r}_x \frac{dr}{dn} - (1 - \nu^*) (\hat{r}_y \hat{n}_x - \hat{r}_x \hat{n}_y) \right] \quad (5.40)$$

$$r(p, q) \neq 0$$

Again Equations 5.33, 5.36, 5.38 and 5.40 for the traction kernel functions can be generalised in a single formula using tensor notation.

The displacement and traction kernel functions defined here for elastic stress analysis are equivalent to those defined in Equations 2.13 and 2.34 for potential problems. There is a considerable difference in complexity, reflecting the vector nature of a concentrated force against the scalar nature of a point source. But, there are important similarities in that both pairs are functions of either $\ln(r^{-1})$ (displacements and potential) or r^{-1} (tractions and potential gradient). The techniques of dealing with such functions developed for potential problems will be applicable to elastic problems.

5.2 Boundary Integral Equations

The relationship equivalent to Green's symmetric identity for potential problems required to develop the boundary integral equation for elastic stress analysis problems is Betti's reciprocal theorem. Suppose that an elastic body is subject to a number of forces, F_k , at particular points and in particular directions on its surface, and that the corresponding displacements at the same points and in the same directions as the forces are u_k . Suppose also that the same body is separately subject to another independent set of forces, F_k^* , at the same points and in the same directions, producing corresponding displacements u_k^* . Betti's theorem states that

$$\sum_k F_k u_k^* = \sum_k F_k^* u_k \quad (5.41)$$

It is a form of virtual work principle: the total virtual work done by the first set of forces moving through the second set of displacements is equal to the virtual work done by the second set of forces moving through the first set of displacements.

For present purposes, the first set of forces and displacements can be those associated with the particular problem to be solved, and the second set with the fundamental solution. Also, a force F , which must in reality act over a finite area, is equivalent to a traction, t , and the summation in Equation 5.41 becomes an integral over the boundary surface, S

$$\int_S t_k u_k^* dS = \int_S t_k^* u_k dS \quad (5.42)$$

Suppose that some point within the solution domain serves as the origin and point of application of the concentrated forces (in the two co-ordinate directions) for the fundamental solution, as shown in Figure 5.6

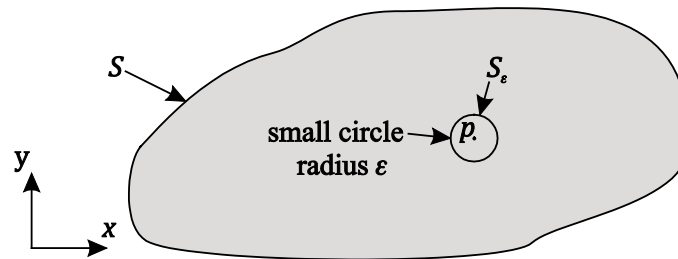


Figure 5.6 A solution domain, including a small region of radius ϵ surrounding the point p

Now, the fundamental solution is not valid at the point p itself. Consider therefore a small internal boundary, S_ϵ , of radius ϵ centred at p . Equation 5.42 can be written for the region between S and S_ϵ , which excludes p where both displacements and tractions are singular

$$\int_S t_k u_k^* dS + \int_{S_\epsilon} t_k u_k^* dS = \int_S t_k^* u_k dS + \int_{S_\epsilon} t_k^* u_k dS \quad (5.43)$$

The next step is to consider what happens as the radius ε shrinks to zero. On the circle, a displacement kernel u_k^* (Equations 5.26 to 5.29) takes the general form of

$$u_k^* = f_1(\theta) \ln\left(\frac{1}{\varepsilon}\right) + f_2(\theta) \quad (5.44)$$

where $f_1(\theta)$ is zero for U_{xy} and U_{yx} . If ε is small the value of traction t_k is effectively constant over the boundary S_ε , and equal to its value at point p . Therefore

$$\begin{aligned} \int_{S_\varepsilon} t_k u_k^* dS &= t_k(p) \int_0^{2\pi} f_1(\theta) \ln\left(\frac{1}{\varepsilon}\right) \varepsilon d\theta + t_k(p) \int_0^{2\pi} f_2(\theta) \varepsilon d\theta \\ &= t_k(p) \varepsilon \ln\left(\frac{1}{\varepsilon}\right) C_1 + t_k(p) \varepsilon C_2 \end{aligned}$$

where C_1 and C_2 are constants. The second term obviously goes to zero as $\varepsilon \rightarrow 0$. Similarly, $\lim_{\varepsilon \rightarrow 0} \left[\varepsilon \ln\left(\frac{1}{\varepsilon}\right) \right] = 0$, and consequently

$$\int_{S_\varepsilon} t_k u_k^* dS = 0 \quad (5.45)$$

Also, a traction kernel t_k^* must be such that

$$\int_{S_\varepsilon} t_k^* dS = 1 \quad (5.46)$$

SMS from your computer

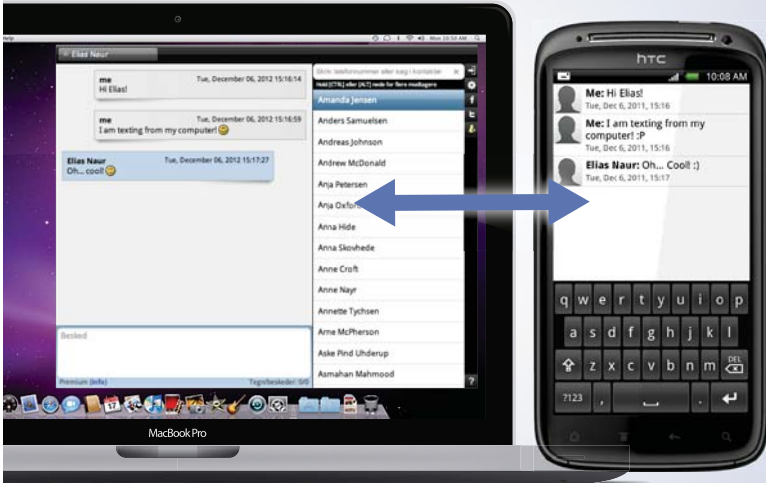
...Sync'd with your Android phone & number


FREE
30 days trial!

Go to

BrowserTexting.com

and start texting from
your computer!




BrowserTexting

if the traction is in the direction of the unit concentrated force at p (and zero if it is at right angles to the force). Therefore, if the force is in the x direction

$$\int_{S_\epsilon} t_k^* u \, dS = u(p) \quad (5.47)$$

and as the boundary S_ϵ shrinks to zero size at the point p , Equation 5.43 becomes

$$u(p) = \int_S t_k u_k^* \, dS - \int_S t_k^* u_k \, dS \quad (5.48)$$

The concentrated force in the x direction at p creates displacements and tractions in both the x and y directions at all points on the boundary of the domain, as defined by the kernel functions. Therefore

$$\begin{aligned} u(p) = & \int_S [t_x(Q)U_{xx}(p, Q) + t_y(Q)U_{xy}(p, Q)] \, dS(Q) \\ & - \int_S [u(Q)T_{xx}(p, Q) + v(Q)T_{xy}(p, Q)] \, dS(Q) \end{aligned} \quad (5.49)$$

Similarly, if a unit line force is applied in the y direction at p

$$\begin{aligned} v(p) = & \int_S [t_x(Q)U_{yx}(p, Q) + t_y(Q)U_{yy}(p, Q)] \, dS(Q) \\ & - \int_S [u(Q)T_{yx}(p, Q) + v(Q)T_{yy}(p, Q)] \, dS(Q) \end{aligned} \quad (5.50)$$

The point Q is a field point on the boundary which moves along the boundary as the integrations proceed.

The displacement and traction kernel functions are known, so provided the values of both the displacements and tractions are known at every point along the boundary, Equations 5.49 and 5.50 provide a means of calculating the solution displacements at any point within the solution domain. They can be differentiated to find strains at p , and hence stresses via the elastic constitutive equations.

In engineering practice, stresses and strains at points within an elastic solution domain are rarely required. In all but a few classes of problems the largest individual stress components and the most intense states of stress are located on a boundary. So for present purposes, data at internal points are not computed.

Let point p be taken to the boundary at a point P , and again consider a small region of exclusion with boundary S_ϵ of radius centred at P . While Equation 5.45 still holds, Equation 5.46 becomes

$$\int_{S_\epsilon} t_k^* \, dS = \text{constant} \quad (5.51)$$

if the traction is in the direction of the concentrated force at P , where the constant is no longer unity. Indeed, if the boundary at P is smooth, the value of the constant is $\frac{1}{2}$, because exactly half of the complete circle is within the solution domain. If the boundary is not smooth, however, the value of the constant is difficult to evaluate directly. Similarly, if the traction is in the direction at right angles to the direction of the concentrated force at P , Equation 5.51 also applies, where the constant is no longer zero. Equations 5.49 and 5.50 become

$$C_{xx}(P)u(P) + C_{xy}(P)v(P) = \int_S [t_x(Q)U_{xx}(P, Q) + t_y(Q)U_{xy}(P, Q)]dS(Q) - \int_S [u(Q)T_{xx}(P, Q) + v(Q)T_{xy}(P, Q)]dS(Q) \quad (5.52)$$

$$C_{yx}(P)u(P) + C_{yy}(P)v(P) = \int_S [t_x(Q)U_{yx}(P, Q) + t_y(Q)U_{yy}(P, Q)]dS(Q) - \int_S [u(Q)T_{yx}(P, Q) + v(Q)T_{yy}(P, Q)]dS(Q) \quad (5.53)$$

where $C_{xx}(P)$, $C_{xy}(P)$, $C_{yx}(P)$ and $C_{yy}(P)$ are constants. The free terms containing these constants contribute only to the coefficients at the diagonal of the relevant matrix in the numerical implementation of the method, and these coefficients can always be evaluated indirectly.

Equations 5.52 and 5.53 are now a pair of boundary integral equations for point P as the point at which line forces of the fundamental solution are applied, the first for force in the x direction and the second for force in the y direction. They can be used to determine the distributions of the displacements and tractions over the boundary.

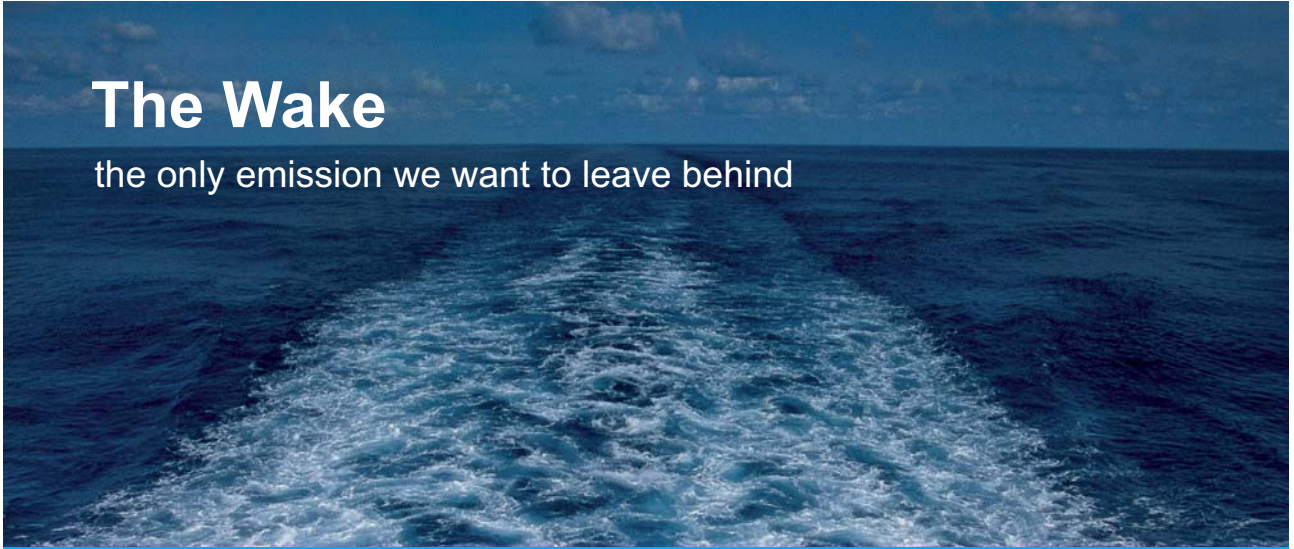
5.3 Discretisation of the Boundary Integral Equations

In general, Equations 5.52 and 5.53 cannot be solved analytically, and some form of numerical method must be employed. Boundary displacements and tractions are found not as a continuous algebraic functions of position along the boundary, but as numerical values at a finite number of discrete points on the boundary. The boundary may be subdivided into small pieces or boundary elements. Associated with each element are one or more of these points: the nodes or nodal points. The distributions of displacements and tractions over the elements are defined in terms of nodal point values by suitable interpolation functions.

Boundary integral Equations 5.52 and 5.53 are applied to each of the N nodal points P in turn, in the discretised form

$$\begin{aligned}
 c_{xx}(P)u(P) + c_{xy}(P)v(P) + \sum_{m=1}^M \int_{S_m} [u(Q)T_{xx}(P, Q) + v(Q)T_{xy}(P, Q)] dS(Q) \\
 = \sum_{m=1}^M \int_{S_m} [t_x(Q)U_{xx}(P, Q) + t_y(Q)U_{xy}(P, Q)] dS(Q)
 \end{aligned}
 \tag{5.54}$$

$$\begin{aligned}
 c_{yx}(P)u(P) + c_{yy}(P)v(P) + \sum_{m=1}^M \int_{S_m} [u(Q)T_{yx}(P, Q) + v(Q)T_{yy}(P, Q)] dS(Q) \\
 = \sum_{m=1}^M \int_{S_m} [t_x(Q)U_{yx}(P, Q) + t_y(Q)U_{yy}(P, Q)] dS(Q)
 \end{aligned}
 \tag{5.55}$$



The Wake


the only emission we want to leave behind

Low-speed Engines Medium-speed Engines Turbochargers Propellers **Propulsion Packages** PrimeServ

The design of eco-friendly marine power and propulsion solutions is crucial for MAN Diesel & Turbo. Power competencies are offered with the world's largest engine programme – having outputs spanning from 450 to 87,220 kW per engine. Get up front! Find out more at www.mandieselturbo.com

Engineering the Future – since 1758.

MAN Diesel & Turbo





Click on the ad to read more

The total number of boundary elements is M , m is an element counter, and S_m is the piece of the boundary occupied by element number m . The details of how integration is carried out over an individual element depend on the type of element involved. Equations 5.54 and 5.55 represent a set of $2N$ linear equations, where N is the number of nodal points ($N = M$ for constant or linear boundary elements, $N = 2M$ for quadratic elements). It is convenient to arrange them in the order of the node numbers, with for each node the equation corresponding to the x direction force first, followed by that for the y direction force

$$[A][u] = [B][t] \quad (5.56)$$

Matrices $[A]$ and $[B]$ are square and contain known constant coefficients, in general all of which will be non-zero. The column vectors $[u]$ and $[t]$ contain the nodal point values of displacements and tractions, two components for each at each node. At each node in each direction either the displacement or traction will be unknown and the other known, or there will be a linear relationship between them. The equations can be rearranged, taking all unknown quantities to the left hand side, known quantities to the right hand side, giving a set of linear equations in a familiar form

$$[A^*][x] = [B^*][y] = [b] \quad (5.57)$$

where $[A^*]$ and $[B^*]$ are modified coefficient matrices and $[b]$ is a column vector of known coefficients. This set can be solved for the $2N$ unknowns x at the nodes, meaning that the displacements and tractions are known at every nodal point on the boundary. Given this information, stresses at the nodes can also be found.

5.4 Boundary Conditions and Surface Stresses

The most straightforward type of boundary condition met in practice is where tractions are prescribed. In other words, for a particular node the traction components in both of the global co-ordinate directions are known. These values contribute to the right hand side of Equations 5.57, the solution of which gives the displacement components at the node.

Traction boundary conditions are most likely to be prescribed in the form of stresses on the boundary, in particular the direct stress normal to the boundary and the shear stress along it.

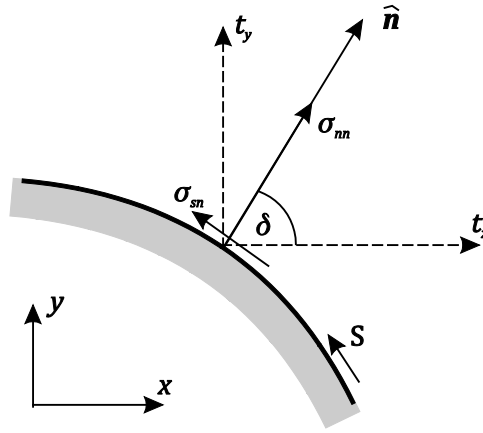


Figure 5.7 Stresses and tractions at a point on the boundary

Figure 5.7 shows the normal stress σ_{nn} and shear stress σ_{sn} acting at a point on the boundary. Note the sign convention for shear stresses: a positive shear stress acts in the positive S direction, along the boundary keeping the domain to the left. The outward unit normal vector \hat{n} at the point concerned is inclined at angle γ to the x global co-ordinate direction, and has components \hat{n}_x and \hat{n}_y in the x and y directions. Because both stresses and both tractions (resultant stresses) act on the same surface, they can be resolved like forces

$$t_x = \sigma_{nn} \cos \gamma - \sigma_{sn} \sin \gamma = \sigma_{nn} \hat{n}_x - \sigma_{sn} \hat{n}_y \quad (5.58)$$

$$t_y = \sigma_{nn} \sin \gamma + \sigma_{sn} \cos \gamma = \sigma_{nn} \hat{n}_y + \sigma_{sn} \hat{n}_x \quad (5.59)$$

With displacement boundary conditions, for a particular node the displacement components in both of the global co-ordinate directions are known. These values must be taken from the left hand to the right hand side of Equations 5.57 to contribute to $[b]$. The solution of the equations gives the traction components at the node.

It should be noted that tractions are often discontinuous at a corner, and indeed at any point where either the direction of the outward normal to the boundary or the magnitudes of the stresses, change abruptly. Figure 5.8 shows a right-angled corner of a domain with an applied normal stress on one side of the corner, and stress free on the other side, a situation that occurs commonly in practice. Label A indicates the point of the corner. Consider the two points B_1 and B_2 on the two sides of A. At B_1 the tractions are $t_x = 0, t_y = 0$, whereas at B_2 they are $t_x = \sigma_0, t_y = 0$. These traction values remain unchanged as B_1 and B_2 approach A. Clearly, there is a discontinuity in traction t_x at A, which raises the question as to what is meant by the tractions at such a point.

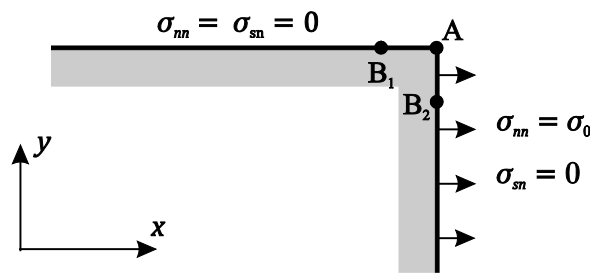


Figure 5.8 Traction at a corner

In practice, there is no difficulty if stresses are prescribed on both sides of a corner, provided the integrals which must be evaluated over both sides of the corner employ the relevant traction values, without reference to any (undefined) corner value. The displacement components at the corner, which are not discontinuous, are the unknowns to be found. The case of displacements prescribed on both sides of a corner is not of great practical interest, corresponding to a form of rigid body displacement. At least for uniform displacements the tractions would be zero. If one side of a corner has displacements prescribed, and the other stresses, then it is convenient to regard the corner as being part of the prescribed displacement side, and treat the unknown tractions as the tractions applicable to this side (the tractions on the other side being known).

TURN TO THE EXPERTS FOR SUBSCRIPTION CONSULTANCY

Subscribe is one of the leading companies in Europe when it comes to innovation and business development within subscription businesses.

We innovate new subscription business models or improve existing ones. We do business reviews of existing subscription businesses and we develop acquisition and retention strategies.

Learn more at [linkedin.com/company/subscribe](https://www.linkedin.com/company/subscribe) or contact
Managing Director Morten Suhr Hansen at mha@subscribe.dk

SUBSCR✓**BE** - to the future



Other types of boundary conditions create greater complexity. For example, a line of symmetry has fixed (usually zero) displacement normal to the line, zero traction along it. Such a mixture of displacements and tractions can be dealt with, but corners where a line of symmetry meets a part of the boundary with either prescribed displacements, prescribed stresses or indeed another line of symmetry, require special care. Further possible types of mixed boundary condition include, for example, flexible supports where tractions are related to displacements.

It is important to note that in boundary element methods for elastic stress analysis problems it is not possible to apply point force boundary conditions, which would give rise to infinite stresses. This is in contrast to other numerical methods such as finite elements. It does, however, reflect physical reality in which even concentrated forces on a body are applied over small but finite areas. Similarly, point displacement constraints are not permitted if they would require point forces to maintain them. On the other hand, such constraints applied to a domain which is already in force equilibrium, simply in order to prevent movement of the domain as a rigid body, are permitted. Indeed, they are necessary to prevent such a problem giving rise to a singular $[A^*]$ matrix in Equations 5.57. This can be a very useful facility, and will be demonstrated in the problems considered in Chapter 6.

Point constraints are straightforward to apply. If, say, a particular node needs to be constrained in the x direction, the first of the two equations corresponding to that node is modified. All the coefficients in that equation, including that in the right hand side vector, but excluding the coefficient on the diagonal of matrix $[A^*]$, are set to zero. The diagonal coefficient is set to one.

The primary results of the analysis are the displacements and tractions at the nodes. Stresses are likely to be of much greater interest in practice than tractions. Referring to Figure 5.7, the normal and shear stresses can be obtained from the tractions as

$$\sigma_{nn} = t_x \cos \gamma + t_y \sin \gamma = t_x \hat{n}_x + t_y \hat{n}_y \quad (5.60)$$

$$\sigma_{sn} = -t_x \sin \gamma + t_y \cos \gamma = -t_x \hat{n}_y + t_y \hat{n}_x \quad (5.61)$$

The other stress component of interest is the direct stress in the direction along the surface, σ_{ss} . This cannot be found from the tractions, but can be found from the displacements. If u_s is the displacement along the boundary in the S direction, the direct strain there is given by

$$e_{ss} = \frac{\partial u_s}{\partial S} \quad (5.62)$$

But this strain can also be found from the stresses and elastic properties

$$e_{ss} = \frac{1}{E^*} (\sigma_{ss} - \nu^* \sigma_{nn})$$

from which
$$\sigma_{ss} = E^* e_{ss} + \nu^* \sigma_{nn} \quad (5.63)$$

Given the three stress components, σ_{nn} , σ_{ss} and σ_{sn} , other stress parameters can also be found, such as the principal stresses at the point. Often of greater practical use is a single stress, reflecting the overall intensity of the state of stress and hence the likelihood of yielding or failure. The choice of this single or equivalent stress therefore depends on the criterion adopted for yielding or failure. At least for ductile materials, the most commonly used is the von Mises equivalent stress, which in the present situation is defined as

$$\sigma_e = \sqrt{\sigma_{nn}^2 + \sigma_{ss}^2 - \sigma_{nn}\sigma_{ss} + 3\sigma_{sn}^2} \quad (5.64)$$

To test for yield or failure, this equivalent stress is compared with the yield or failure stress for the material concerned.

5.5 Quadratic Boundary Elements

Experience of potential problems in Chapters 3 and 4 showed that, although constant boundary elements are easier to implement, quadratic elements give much more accurate results for the same fineness of discretisation. The only exceptions to this were when results were required at points within a solution domain close to a boundary, or when the domain was very narrow so that opposite sides of the boundary were close together. For elastic stress analysis problems attention is confined here to quadratic boundary elements. This is particularly because such problems are often concerned with rapid stress variations in local regions such as stress concentrations or cracks. Results at internal points are rarely required, but care will need to be taken with long slender domains.

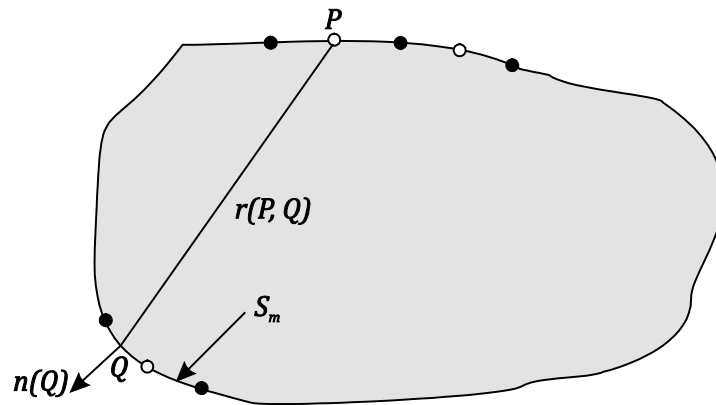


Figure 5.9 Quadratic element discretisation of a two-dimensional solution domain boundary

Figure 5.9 shows a typical arrangement of quadratic line elements on parts of the boundary of a two-dimensional solution domain. Each element has three nodes, one at each end and one at its centre. While the end nodes are shown as solid circles, those at the centres of elements are shown as open circles. The relevant boundary integral equations are Equations 5.54 and 5.55. The total number of nodal points is twice the total number of elements, and there are two equations per node, hence $4M$ equations are generated, M being the number of elements. The integrations involved in the evaluation of the equation coefficients must in general be carried out numerically.

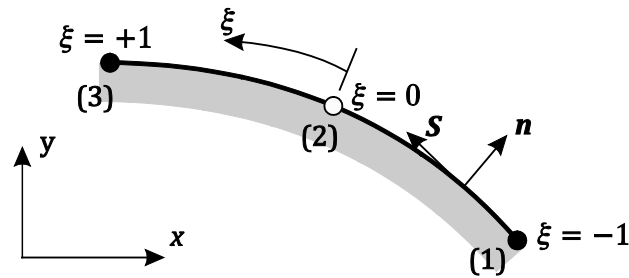
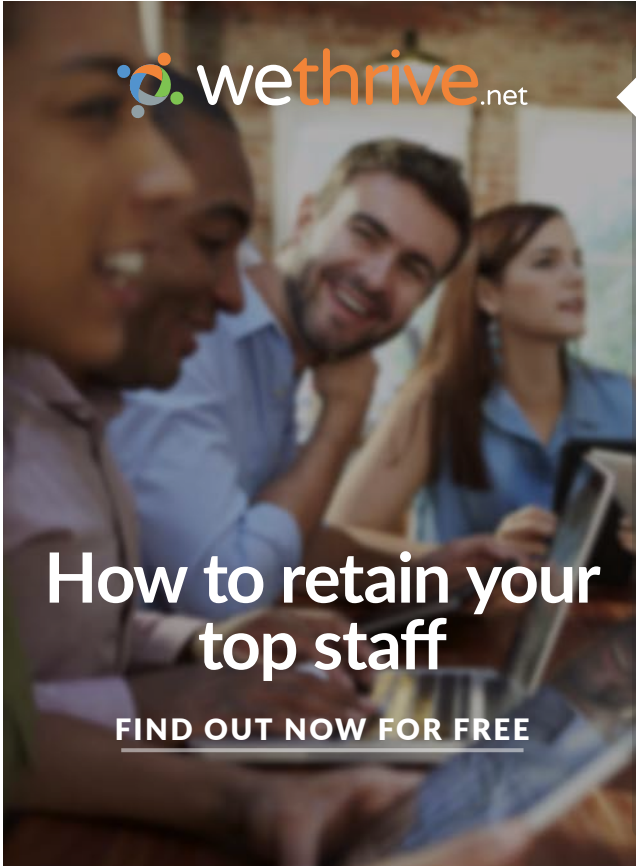


Figure 5.10 A typical quadratic line element






wethrive.net

How to retain your top staff

FIND OUT NOW FOR FREE

DO YOU WANT TO KNOW:

- 
What your staff really want?
- 
The top issues troubling them?
- 
How to make staff assessments work for you & them, painlessly?

Get your free trial

Because happy staff get more done

Figure 5.10 shows a typical curved element, exactly as used for potential problems. Its three nodes are numbered in the direction of integration from 1 to 3, these numbers being local to the particular element. The second node is at the centre of the element as measured along its curved length. A local intrinsic co-ordinate, ξ , follows the curved shape of the element. It has its origin at the centre node: that is, $\xi = 0$ at node 2, and takes the values -1 and $+1$ at nodes 1 and 3, respectively.

The distributions of displacements and tractions are approximated as quadratic functions of position along the element in terms of values at the three nodes. For example

$$u(\xi) = N_1(\xi)u_1 + N_2(\xi)u_2 + N_3(\xi)u_3 = \sum_{c=1}^3 N_c(\xi) u_c \quad (5.65)$$

Shape functions are as defined in Equations 2.50, 2.51 and 2.52

$$N_1(\xi) = \frac{1}{2}\xi(\xi - 1) \quad (5.66)$$

$$N_2(\xi) = 1 - \xi^2 \quad (5.67)$$

$$N_3(\xi) = \frac{1}{2}\xi(\xi + 1) \quad (5.68)$$

Variations of the global co-ordinates are similarly defined

$$x(\xi) = N_1(\xi)x_1 + N_2(\xi)x_2 + N_3(\xi)x_3 = \sum_{c=1}^3 N_c(\xi)x_c \quad (5.69)$$

$$y(\xi) = N_1(\xi)y_1 + N_2(\xi)y_2 + N_3(\xi)y_3 = \sum_{c=1}^3 N_c(\xi)y_c \quad (5.70)$$

The rates of change of the global co-ordinates with respect to ξ are

$$\frac{dx}{d\xi} = \sum_{c=1}^3 \frac{dN_c(\xi)}{d\xi} x_c, \quad \frac{dy}{d\xi} = \sum_{c=1}^3 \frac{dN_c(\xi)}{d\xi} y_c \quad (5.71)$$

and the derivatives of the shape functions are

$$\frac{dN_1(\xi)}{d\xi} = \xi - \frac{1}{2}, \quad \frac{dN_2(\xi)}{d\xi} = -2\xi, \quad \frac{dN_3(\xi)}{d\xi} = \xi + \frac{1}{2} \quad (5.72)$$

As in Equation 2.59 the vector normal to the boundary can be found as

$$\mathbf{n} = n_x \mathbf{i} + n_y \mathbf{j} = \mathbf{S} \wedge \mathbf{k} = \frac{dy}{d\xi} \mathbf{i} - \frac{dx}{d\xi} \mathbf{j} \quad (5.73)$$

where n_x and n_y are the components in the x and y directions of the vector normal \mathbf{n} .

In order to integrate round the boundary it is necessary to change from global co-ordinate to local intrinsic co-ordinate ξ within each element, by means of the Jacobian of transformation

$$J(\xi) = \frac{dS}{d\xi} = \frac{\sqrt{(dx)^2 + (dy)^2}}{d\xi} = \sqrt{\left(\frac{dx}{d\xi}\right)^2 + \left(\frac{dy}{d\xi}\right)^2} = \sqrt{n_x^2 + n_y^2} \quad (5.74)$$

which can also be used to define the components of the unit normal

$$\hat{\mathbf{n}} = \frac{\mathbf{n}}{\sqrt{n_x^2 + n_y^2}} = \frac{n_x}{J(\xi)} \mathbf{i} + \frac{n_y}{J(\xi)} \mathbf{j} = \hat{n}_x \mathbf{i} + \hat{n}_y \mathbf{j} \quad (5.75)$$

Introducing the transformation from global to intrinsic co-ordinate, the boundary integral equations, Equations 5.54 and 5.55, become

$$\begin{aligned} C_{xx}(P)u(P) + C_{xy}(P)v(P) + \sum_{m=1}^M \int_{-1}^{+1} [u(Q)T_{xx}(P, Q) + v(Q)T_{xy}(P, Q)] J(\xi) d\xi \\ = \sum_{m=1}^M \int_{-1}^{+1} [t_x(Q)U_{xx}(P, Q) + t_y(Q)U_{xy}(P, Q)] J(\xi) d\xi \end{aligned} \quad (5.76)$$

$$\begin{aligned} C_{yx}(P)u(P) + C_{yy}(P)v(P) + \sum_{m=1}^M \int_{-1}^{+1} [u(Q)T_{yx}(P, Q) + v(Q)T_{yy}(P, Q)] J(\xi) d\xi \\ = \sum_{m=1}^M \int_{-1}^{+1} [t_x(Q)U_{yx}(P, Q) + t_y(Q)U_{yy}(P, Q)] J(\xi) d\xi \end{aligned} \quad (5.77)$$

Then introducing the parametric representation of Equation 5.65 for the displacement and traction distributions

$$\begin{aligned} C_{xx}(P)u(P) + C_{xy}(P)v(P) + \sum_{m=1}^M \sum_{c=1}^3 \int_{-1}^{+1} [u_c T_{xx}(P, Q) + v_c T_{xy}(P, Q)] N_c(\xi) J(\xi) d\xi \\ = \sum_{m=1}^M \sum_{c=1}^3 \int_{-1}^{+1} [\{t_x\}_c U_{xx}(P, Q) + \{t_y\}_c U_{xy}(P, Q)] N_c(\xi) J(\xi) d\xi \end{aligned} \quad (5.78)$$

$$\begin{aligned} C_{yx}(P)u(P) + C_{yy}(P)v(P) + \sum_{m=1}^M \sum_{c=1}^3 \int_{-1}^{+1} [u_c T_{yx}(P, Q) + v_c T_{yy}(P, Q)] N_c(\xi) J(\xi) d\xi \\ = \sum_{m=1}^M \sum_{c=1}^3 \int_{-1}^{+1} [\{t_x\}_c U_{yx}(P, Q) + \{t_y\}_c U_{yy}(P, Q)] N_c(\xi) J(\xi) d\xi \end{aligned} \quad (5.79)$$

where c is the number (1, 2 or 3) of the node in element number m . The first (traction) kernel functions on the left hand sides of the equations and the second (displacement) kernels on the right hand sides are given by Equations 5.33, 5.36, 5.38, 5.40 and 5.26 to 5.29.

5.5.1 Points P and Q not in the same element

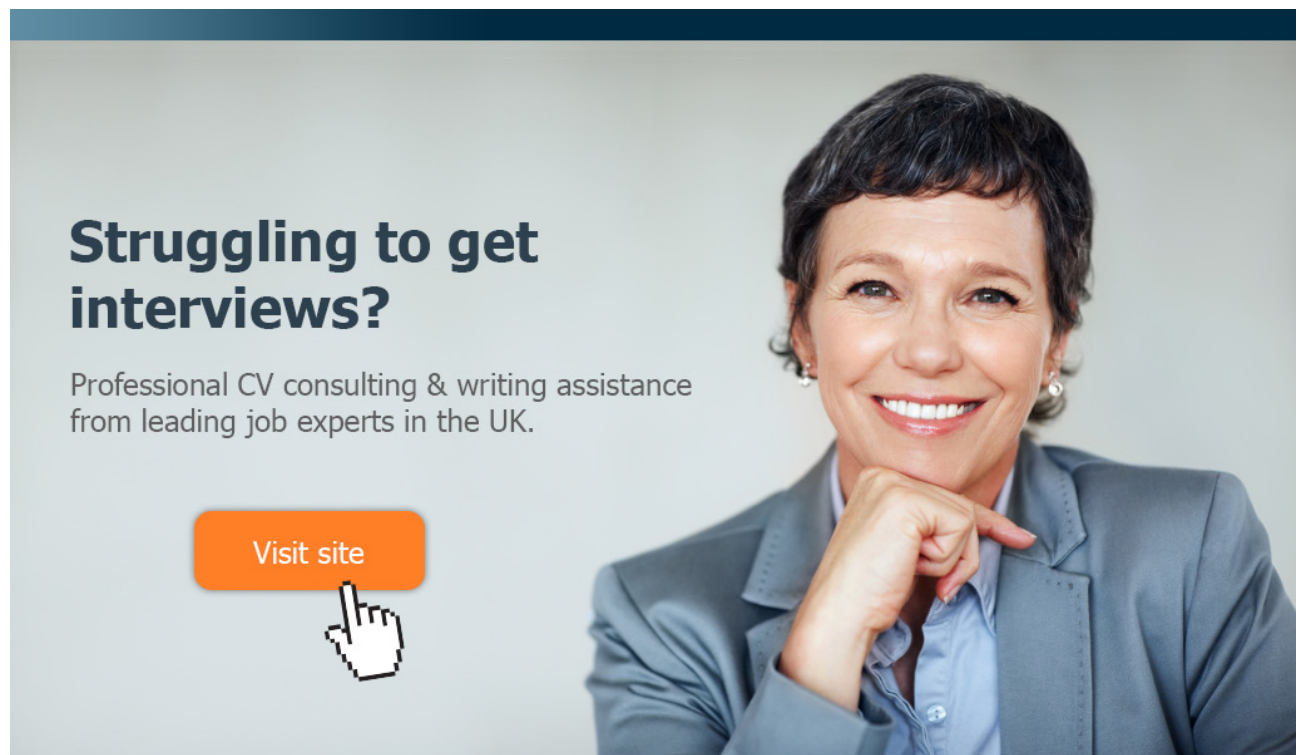
The required integrations are performed using Gaussian quadrature, described in Appendix A. Provided point P is not in the element over which integration is required the process is straightforward.

5.5.2 Points P and Q in the same element

The case where P is in the relevant element requires some care, because when P and Q are coincident the kernel functions are singular. The orders of the singularities are r^{-1} and $\ln(r^{-1})$. Provided that P is not the c th node of the element, however, the shape function $N_c(\xi)$ goes to zero at P , and the products of kernels and shape functions are not singular at P , and may be integrated normally.

If P is the c th node of the element then the terms at the diagonal of matrix $[A]$, involving both the first (traction) kernel functions and the free term constants $C_{xx}(P)$, $C_{xy}(P)$, $C_{yx}(P)$, and $C_{yy}(P)$, are difficult to evaluate directly. They can, however, be evaluated indirectly. If there are no tractions applied to the boundary of a domain, Equations 5.56 become

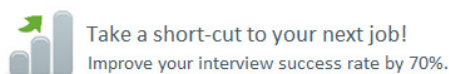
$$[A][u] = [0] \quad (5.80)$$



Struggling to get interviews?

Professional CV consulting & writing assistance from leading job experts in the UK.

[Visit site](#)



Click on the ad to read more

Possible outcomes of this lack of loading include rigid body displacement of the entire domain in the x direction by a unit distance (all x -direction displacements take the value 1, all y -direction displacements take the value 0). Also, rigid body displacement of the entire domain in the y direction by a unit distance (all y -direction displacements take the value 1, all x -direction displacements take the value 0). This means that the sum of all the coefficients multiplying x -direction displacements in any row of $[A]$ should be zero, and, separately, the sum of all the coefficients multiplying y -direction displacements in any row of $[A]$ should also be zero. These conditions allow the coefficients of displacements at point P (four coefficients in total, two in each of the two equations) to be determined by summations equivalent to Equations 2.47 and 2.66.

Still considering P to be the c th node of the element in Equations 5.78 and 5.79, the second (displacement) kernels require special treatment to deal with the $\ln(r^{-1})$ singularity, using a modified form of Gaussian quadrature. The procedures follow closely those described in Section 2.5.2, but are repeated here for convenience. The radial distance from P to a point Q at (x, y) can be arranged in the form

$$r(P, Q) = \eta R(\xi) \quad (5.81)$$

where $R(\xi)$ is a known function and η is a modified intrinsic co-ordinate with its origin at P . The form of $R(\xi)$ depends on which of the three element nodes P is located at. Co-ordinate η is chosen in each case to conform to the requirements of Gaussian quadrature involving a logarithmic function (Appendix A).

P at the first node

If P is at the first node of the element

$$r(P, Q) = [(x - x_1)^2 + (y - y_1)^2]^{\frac{1}{2}} \quad (5.82)$$

and x and y are defined in terms of intrinsic co-ordinate ξ by Equations 5.69 and 5.70. So

$$x - x_1 = [N_1(\xi) - 1]x_1 + N_2(\xi)x_2 + N_3(\xi)x_3 \quad (5.83)$$

$$y - y_1 = [N_1(\xi) - 1]y_1 + N_2(\xi)y_2 + N_3(\xi)y_3 \quad (5.84)$$

Intrinsic co-ordinate η is chosen to range from 0 at the first node ($\xi = -1$) to 1 at the third node ($\xi = +1$), so that

$$\eta = \frac{1}{2}(\xi + 1) \quad \left| \frac{d\xi}{d\eta} \right| = 2 \quad (5.85)$$

$$[N_1(\xi) - 1] = \frac{1}{2}(\xi^2 - \xi - 2) = \frac{1}{2}(\xi + 1)(\xi - 2) = \eta(\xi - 2) \quad (5.86)$$

$$N_2(\xi) = 1 - \xi^2 = 2\eta(1 - \xi) \quad (5.87)$$

$$N_3(\xi) = \frac{1}{2}\xi(\xi + 1) = \eta\xi \quad (5.88)$$

Therefore

$$x - x_1 = \eta[(\xi - 2)x_1 + 2(1 - \xi)x_2 + \xi x_3] \quad (5.89)$$

$$y - y_1 = \eta[(\xi - 2)y_1 + 2(1 - \xi)y_2 + \xi y_3] \quad (5.90)$$

$$R(\xi) = \{[(\xi - 2)x_1 + 2(1 - \xi)x_2 + \xi x_3]^2 + [(\xi - 2)y_1 + 2(1 - \xi)y_2 + \xi y_3]^2\}^{\frac{1}{2}} \quad (5.91)$$

The integrals of the second kernels in Equations 5.78 and 5.79 can be expressed in the form

$$\begin{aligned} \int_{-1}^{+1} U_{xx}(P, Q) N_c(\xi) J(\xi) d\xi &= \int_{-1}^{+1} \frac{(1 + \nu^*)^2}{4\pi E^*} \left[\frac{(3 - \nu^*)}{(1 + \nu^*)} \ln \left(\frac{1}{r(P, Q)} \right) + \hat{r}_x \hat{r}_x \right] N_c(\xi) J(\xi) d\xi \\ &= K_1 K_2 \int_0^1 \ln \left(\frac{1}{\eta} \right) N_c(\xi) J(\xi) \left| \frac{d\xi}{d\eta} \right| d\eta + K_1 K_2 \int_{-1}^{+1} \ln \left(\frac{1}{R(\xi)} \right) N_c(\xi) J(\xi) d\xi + K_1 \int_{-1}^{+1} \hat{r}_x \hat{r}_x N_c(\xi) J(\xi) d\xi \end{aligned} \quad (5.92)$$

$$\int_{-1}^{+1} U_{xy}(P, Q) N_c(\xi) J(\xi) d\xi = \int_{-1}^{+1} \frac{(1 + \nu^*)^2}{4\pi E^*} \hat{r}_x \hat{r}_y N_c(\xi) J(\xi) d\xi = K_1 \int_{-1}^{+1} \hat{r}_x \hat{r}_y N_c(\xi) J(\xi) d\xi \quad (5.93)$$

$$\int_{-1}^{+1} U_{yx}(P, Q) N_c(\xi) J(\xi) d\xi = \int_{-1}^{+1} \frac{(1 + \nu^*)^2}{4\pi E^*} \hat{r}_y \hat{r}_x N_c(\xi) J(\xi) d\xi = K_1 \int_{-1}^{+1} \hat{r}_y \hat{r}_x N_c(\xi) J(\xi) d\xi \quad (5.94)$$

$$\begin{aligned} \int_{-1}^{+1} U_{yy}(P, Q) N_c(\xi) J(\xi) d\xi &= \int_{-1}^{+1} \frac{(1 + \nu^*)^2}{4\pi E^*} \left[\frac{(3 - \nu^*)}{(1 + \nu^*)} \ln \left(\frac{1}{r(P, Q)} \right) + \hat{r}_y \hat{r}_y \right] N_c(\xi) J(\xi) d\xi \\ &= K_1 K_2 \int_0^1 \ln \left(\frac{1}{\eta} \right) N_c(\xi) J(\xi) \left| \frac{d\xi}{d\eta} \right| d\eta + K_1 K_2 \int_{-1}^{+1} \ln \left(\frac{1}{R(\xi)} \right) N_c(\xi) J(\xi) d\xi + K_1 \int_{-1}^{+1} \hat{r}_y \hat{r}_y N_c(\xi) J(\xi) d\xi \end{aligned} \quad (5.95)$$

$$\text{where } K_1 = \frac{(1 + \nu^*)^2}{4\pi E^*} \text{ and } K_2 = \frac{(3 - \nu^*)}{(1 + \nu^*)} \quad (5.96)$$

are constants.

Because $R(\xi)$ is not zero within the range of integration, all of the integrals except those involving $\ln \left(\frac{1}{\eta} \right)$ can be evaluated by normal Gaussian quadrature. Those involving the singular logarithmic function $\ln \left(\frac{1}{\eta} \right)$ can be evaluated using the appropriate quadrature formula described in Appendix A.

P at the second node

If P is at the second node of the element

$$r(P, Q) = [(x - x_2)^2 + (y - y_2)^2]^{\frac{1}{2}} \quad (5.97)$$

$$x - x_2 = N_1(\xi)x_1 + [N_2(\xi) - 1]x_2 + N_3(\xi)x_3 \quad (5.98)$$

$$y - y_2 = N_1(\xi)y_1 + [N_2(\xi) - 1]y_2 + N_3(\xi)y_3 \quad (5.99)$$

For integration purposes, the element needs to be divided into two regions, from the second node to the third node ($\xi = 0$ to 1) and from the second node to the first node ($\xi = 0$ to -1). Between the second and third nodes the intrinsic co-ordinate is chosen as

$$\eta_1 = \xi \quad \left| \frac{d\xi}{d\eta_1} \right| = 1 \quad (5.100)$$

$$\text{and } N_1(\xi) = \frac{1}{2}\xi(\xi - 1) = \frac{1}{2}\eta_1(\xi - 1) \quad (5.101)$$

$$[N_2(\xi) - 1] = -\xi^2 = -\eta_1\xi \quad (5.102)$$

$$N_3(\xi) = \frac{1}{2}\xi(\xi + 1) = \frac{1}{2}\eta_1(\xi + 1) \quad (5.103)$$



gaiTEYE®
Challenge the way we run

**EXPERIENCE THE POWER OF
FULL ENGAGEMENT...**

.....

**RUN FASTER.
RUN LONGER..
RUN EASIER...**

**READ MORE & PRE-ORDER TODAY
WWW.GAITEYE.COM**

Therefore

$$x - x_2 = \eta_1 \left[\frac{1}{2}(\xi - 1)x_1 - \xi x_2 + \frac{1}{2}(\xi + 1)x_3 \right] \quad (5.104)$$

$$y - y_2 = \eta_1 \left[\frac{1}{2}(\xi - 1)y_1 - \xi y_2 + \frac{1}{2}(\xi + 1)y_3 \right] \quad (5.105)$$

$$R(\xi) = \left\{ \left[\frac{1}{2}(\xi - 1)x_1 - \xi x_2 + \frac{1}{2}(\xi + 1)x_3 \right]^2 + \left[\frac{1}{2}(\xi - 1)y_1 - \xi y_2 + \frac{1}{2}(\xi + 1)y_3 \right]^2 \right\}^{\frac{1}{2}} \quad (5.106)$$

Between the second and first nodes the intrinsic co-ordinate η_2 is chosen as

$$\eta_2 = -\xi \quad \left| \frac{d\xi}{d\eta_2} \right| = 1 \quad (5.107)$$

and $R(\xi)$ is as defined in Equation 5.106.

The integrals of the second kernels $U_{xy}(P, Q)$ and $U_{yx}(P, Q)$ in Equations 5.78 and 5.79 are as in Equations 5.93 and 5.94. The integrals of the other two can be expressed in the form

$$\begin{aligned} \int_{-1}^{+1} U_{xx}(P, Q) N_c(\xi) J(\xi) d\xi &= \int_{-1}^{+1} \frac{(1 + \nu^*)^2}{4\pi E^*} \left[\frac{(3 - \nu^*)}{(1 + \nu^*)} \ln \left(\frac{1}{r(P, Q)} \right) + \hat{r}_x \hat{r}_x \right] N_c(\xi) J(\xi) d\xi \\ &= K_1 K_2 \int_0^1 \ln \left(\frac{1}{\eta_1} \right) N_c(\xi) J(\xi) \left| \frac{d\xi}{d\eta_1} \right| d\eta_1 + K_1 K_2 \int_0^1 \ln \left(\frac{1}{\eta_2} \right) N_c(\xi) J(\xi) \left| \frac{d\xi}{d\eta_2} \right| d\eta_2 \\ &+ K_1 K_2 \int_{-1}^{+1} \ln \left(\frac{1}{R(\xi)} \right) N_c(\xi) J(\xi) d\xi + K_1 \int_{-1}^{+1} \hat{r}_x \hat{r}_x N_c(\xi) J(\xi) d\xi \end{aligned} \quad (5.108)$$

$$\begin{aligned} \int_{-1}^{+1} U_{yy}(P, Q) N_c(\xi) J(\xi) d\xi &= \int_{-1}^{+1} \frac{(1 + \nu^*)^2}{4\pi E^*} \left[\frac{(3 - \nu^*)}{(1 + \nu^*)} \ln \left(\frac{1}{r(P, Q)} \right) + \hat{r}_y \hat{r}_y \right] N_c(\xi) J(\xi) d\xi \\ &= K_1 K_2 \int_0^1 \ln \left(\frac{1}{\eta_1} \right) N_c(\xi) J(\xi) \left| \frac{d\xi}{d\eta_1} \right| d\eta_1 + K_1 K_2 \int_0^1 \ln \left(\frac{1}{\eta_2} \right) N_c(\xi) J(\xi) \left| \frac{d\xi}{d\eta_2} \right| d\eta_2 \\ &+ K_1 K_2 \int_{-1}^{+1} \ln \left(\frac{1}{R(\xi)} \right) N_c(\xi) J(\xi) d\xi + K_1 \int_{-1}^{+1} \hat{r}_y \hat{r}_y N_c(\xi) J(\xi) d\xi \end{aligned} \quad (5.109)$$

The third and fourth integrals on the right hand sides of Equations 5.108 and 5.109 can be evaluated by normal Gaussian quadrature, while the first and second involve the singular logarithmic function and must be evaluated using the appropriate quadrature formula.

P at the third node

If P is at the third node of the element

$$r(P, Q) = [(x - x_3)^2 + (y - y_3)^2]^{\frac{1}{2}} \quad (5.110)$$

$$x - x_3 = N_1(\xi)x_1 + N_2(\xi)x_2 + [N_3(\xi) - 1]x_3 \quad (5.111)$$

$$y - y_3 = N_1(\xi)y_1 + N_2(\xi)y_2 + [N_3(\xi) - 1]y_3 \quad (5.112)$$

Intrinsic co-ordinate η is chosen to range from 0 at the third node ($\xi = +1$) to 1 at the first node ($\xi = -1$), so that

$$\eta = \frac{1}{2}(1 - \xi) \quad \left| \frac{d\xi}{d\eta} \right| = 2 \quad (5.113)$$

$$N_1(\xi) = \frac{1}{2}\xi(\xi - 1) = -\eta\xi \quad (5.114)$$

$$N_2(\xi) = 1 - \xi^2 = 2\eta(1 + \xi) \quad (5.115)$$

$$[N_3(\xi) - 1] = \frac{1}{2}(\xi^2 + \xi - 2) = \frac{1}{2}(\xi - 1)(\xi + 2) = -\eta(\xi + 2) \quad (5.116)$$

Therefore

$$x - x_3 = \eta[-\xi x_1 + 2(1 + \xi)x_2 - (\xi + 2)x_3] \quad (5.117)$$

$$y - y_3 = \eta[-\xi y_1 + 2(1 + \xi)y_2 - (\xi + 2)y_3] \quad (5.118)$$

$$R(\xi) = \{[-\xi x_1 + 2(1 + \xi)x_2 - (\xi + 2)x_3]^2 + [-\xi y_1 + 2(1 + \xi)y_2 - (\xi + 2)y_3]^2\}^{\frac{1}{2}} \quad (5.119)$$

The integrals of the second kernels $U_{xy}(P, Q)$ and $U_{yx}(P, Q)$ in Equations 5.78 and 5.79 are as in Equations 5.93 and 5.94. The integrals of the other two can be expressed in the form

$$\begin{aligned} \int_{-1}^{+1} U_{xx}(P, Q) N_c(\xi) J(\xi) d\xi &= \int_{-1}^{+1} \frac{(1 + \nu^*)^2}{4\pi E^*} \left[\frac{(3 - \nu^*)}{(1 + \nu^*)} \ln\left(\frac{1}{r(P, Q)}\right) + \hat{r}_x \hat{r}_x \right] N_c(\xi) J(\xi) d\xi \\ &= K_1 K_2 \int_0^1 \ln\left(\frac{1}{\eta}\right) N_c(\xi) J(\xi) \left| \frac{d\xi}{d\eta} \right| d\eta + K_1 K_2 \int_{-1}^{+1} \ln\left(\frac{1}{R(\xi)}\right) N_c(\xi) J(\xi) d\xi + K_1 \int_{-1}^{+1} \hat{r}_x \hat{r}_x N_c(\xi) J(\xi) d\xi \end{aligned} \quad (5.120)$$

$$\begin{aligned} \int_{-1}^{+1} U_{yy}(P, Q) N_c(\xi) J(\xi) d\xi &= \int_{-1}^{+1} \frac{(1+\nu^*)^2}{4\pi E^*} \left[\frac{(3-\nu^*)}{(1+\nu^*)} \ln\left(\frac{1}{r(P, Q)}\right) + \hat{r}_y \hat{r}_y \right] N_c(\xi) J(\xi) d\xi \\ &= K_1 K_2 \int_0^1 \ln\left(\frac{1}{\eta}\right) N_c(\xi) J(\xi) \left| \frac{d\xi}{d\eta} \right| d\eta + K_1 K_2 \int_{-1}^{+1} \ln\left(\frac{1}{R(\xi)}\right) N_c(\xi) J(\xi) d\xi + K_1 \int_{-1}^{+1} \hat{r}_y \hat{r}_y N_c(\xi) J(\xi) d\xi \quad (5.121) \end{aligned}$$

The second and third integrals on the right hand sides can be evaluated by normal Gaussian quadrature, while the first involves the singular logarithmic function and must be evaluated using the appropriate quadrature formula.

5.6 Scaling

As in the case of two-dimensional potential problems (Section 2.6), it is necessary to scale all distances between nodes to be less than unity. This is done by dividing by the maximum dimension of the problem (the maximum distance between any pair of nodes). Displacements have the units of length, so these are also scaled by dividing by the maximum dimension. Similarly, stresses and tractions are scaled by dividing by the value of Young's modulus. Once the solution to the scaled problem has been obtained, the scaling is removed.

5.7 Solving the Linear Equations

As for potential problems (Section 2.7), the linear equations arising from the boundary element analysis are most appropriately solved by direct methods such as Gaussian elimination. The method is described in detail, including an appropriate computer subprogram, in Appendix B.

5.8 Body Forces

Many potential problems involve the Poisson form of governing differential equation rather than the simpler Laplace form. As a result, a particular integral has to be added, which can be done by modifying the boundary conditions for the Laplace problem, as explained in Section 2.8. In elastic stress analysis problems the equivalent situation arises if there are significant body forces. The most common of these is gravity, but other examples include centrifugal loading. Gravitational effects only become significant in physically very large components and structures, such as bridges and dams. In the large majority of problems body forces can be neglected, and they are not considered in detail here.

Problems

- 5.1 Under what conditions are the states of plane stress and plane strain indistinguishable? Is this of any practical significance?
- 5.2 Find the particular forms of the typical displacement and traction kernel functions, $U_{xx}(p, q)$, $U_{xy}(p, q)$, $T_{xx}(p, q)$ and $T_{xy}(p, q)$ for an incompressible material under plane strain conditions.

- 5.3 The typical point on a boundary shown in Figure 5.7 is subject to flexible boundary conditions

$$\sigma_{nn} = -k_n u_n \quad \sigma_{sn} = -k_s u_s$$

where u_n and u_s are the displacements in the n and S directions, and k_n and k_s are normal and shear stiffnesses. Define the boundary conditions for the displacements and tractions in the global co-ordinate directions.

- 5.4 Assuming that gravity acts in the negative y direction, show that the following distributions of stresses and displacements provide a possible particular integral for this body force

$$\begin{aligned} \sigma_{yy} &= \rho g y, & \sigma_{xx} &= \nu^* \sigma_{yy}, & \sigma_{xy} &= 0 \\ u &= 0, & v &= \frac{\rho g y^2}{2E^*} (1 - \nu^{*2}) \end{aligned}$$

where ρ is the material density, and g is the acceleration due to gravity.

- 5.5 If a body whose material has a density of ρ is rotated at an angular velocity of ω about the y axis, the centrifugal body force per unit volume generated is $\rho \omega^2 x$ in the x direction. Show that the following distributions of stresses and displacements provide a possible particular integral for this body force

$$\begin{aligned} \sigma_{xx} &= -\frac{\rho \omega^2 x^2}{2}, & \sigma_{yy} &= \nu^* \sigma_{xx}, & \sigma_{xy} &= 0 \\ u &= -\frac{\rho \omega^2 x^3}{6E^*} (1 - \nu^{*2}), & v &= 0 \end{aligned}$$

6 Quadratic Boundary Element Program for Plane Elastic Problems

In this chapter a Fortran computer program to implement the quadratic boundary element formulation for two-dimensional elastic stress analysis problems developed in Chapter 5 is presented and described in detail. It is then used to solve a range of problems to demonstrate the capabilities of the method.

For readers who prefer to use Matlab, a translation is provided in Appendix E.

As far as possible the program structure, file names, variable names, and actual coding follow those used in the programs for potential problems, particularly the quadratic element program described in Chapter 4. As in the case of that program, the principle adopted in programming is to try to make the coding straightforward to understand and follow, rather than necessarily the most efficient in terms of computation.

This e-book
is made with
SetaPDF



PDF components for PHP developers

www.setasign.com





6.1 Program BEM2EQ

The program name indicates that it is for **boundary element method** analysis of **two-dimensional elastic** problems using **quadratic elements**. Each of the subprogram units which make up the whole are described in turn. The Preface explains how the full program can be accessed as a single file.

As with programs for potential problems, much of the coding is concerned with the definition of the arrangement of elements on the boundary (or boundaries) of the solution domain, and the application of boundary conditions to them. Again, entering the co-ordinates of the nodes of each and every element, followed by the type and magnitude of the relevant boundary condition applied to each is an option, but tedious. Instead, each boundary can be divided up into a series of boundary segments, which are either straight lines or circular arcs. The number of elements within a segment can then be chosen, and varied easily, and from which the program generates all the element geometric data. The elements on a segment do not have to be uniform in size, but can be varied in length by a constant ratio between successive elements. In the present version of the program, each segment is subject to only one uniform boundary condition: defined displacements or defined stresses. The program distributes this condition to all the elements involved. Consequently, the ends of segments are conveniently defined as points where there is a significant change in either shape (a corner, for example) or boundary condition. Use of this facility is demonstrated later in this chapter.

6.1.1 Main program

At the beginning of the program is a storage module named SHARED DATA2EQ which allows stored data to be accessed and shared by all those subprograms that require them (by means of a USE statement). The dimensioned array sizes in the module allow for up to 250 quadratic boundary elements with 500 nodes, with up to 10 different boundaries forming the solution domain, and a maximum number of point (node) displacement constraints of 20. With two equations per node, this means that up to 1000 equations can be solved. A dictionary of the variable names used is provided at the beginning of the book, and at the beginning of Part II.

The main program named BEM2EQ is designed mainly to call each of the other subprograms in turn. It does, however, also serve to name the files with which the program communicates via OPEN statements. File DATA, which is addressed as file number 5 in the program, serves to supply the input data which defines the current problem. The main output of results is to file RESULTS, addressed as file number 6. Element mesh data, on the other hand, are output to file MESHRES (mesh results) and numbered 7.

```
MODULE SHARED DATA2EQ
!
!  MODULE STORING SHARED DATA.
!
```

```

REAL :: XEEND(260), YEEND(260), XNODE(500), YNODE(500)
REAL :: XSEND(250), YSEND(250)
REAL :: UNMX(250,3), UNMY(250,3), ANGSTORE(260), MAXL
REAL :: A(1000,1001), UV(1000), U(500), V(500), USEG(250), VSEG(250)
REAL :: AROWX(250,6), AROWY(250,6), BROWX(250,6), BROWY(250,6)
REAL :: ZG(8), WG(8), EGL(8), WGL(8), JACOB, UNGX, UNGY
REAL :: SIGNNSEG(250), SIGSNSEG(250), UELEM(250), VELEM(250)
REAL :: SIGNN(250,3), SIGSS(250,3), SIGSN(250,3), SIGE(250,3)
REAL :: TX(500), TY(500), TMX(250,3), TMY(250,3)
REAL :: FXSEG(250), FYSEG(250)
REAL :: SF(3,8), SD(3,13), SFL(4,3,8), SDL(4,3,8)
REAL :: PI, E, NU, ESTORE
REAL :: AKXX, AKXY, AKYX, AKYY, BKXX, BKXY, BKYX, BKYY
REAL :: BK2XX, BK2XY, BK2YX, BK2YY
INTEGER :: NEL, NNP, MAXNEL, MAXNNP, MAXNB, NEEND, NGAUSS
INTEGER :: NODE(250,3), M1(250), M3(250)
INTEGER :: NBOUND, NSEGTOT, NELB(10), NSEGB(10)
INTEGER :: NBCU, NBCS, NBCM, NBCT, IBCE(250), IBCN(500), ISEGBC(250)
INTEGER :: ISEGEND(250), ISEGELEM(250), MFIRST(250), MLAST(250)
INTEGER :: NBCPC, MAXNPC, NODEPC(20), IDIRPC(20)

```



**YOU THINK.
YOU CAN WORK
AT RMB**

 **RAND
MERCHANT
BANK**
A division of FirstRand Bank Limited
Traditional values. Innovative ideas.

Rand Merchant Bank uses good business to create a better world, which is one of the reasons that the country's top talent chooses to work at RMB. For more information visit us at www.rmb.co.za

Thinking that can change your world

Rand Merchant Bank is an Authorised Financial Services Provider



Click on the ad to read more

```
END MODULE SHARED DATA2EQ

PROGRAM BEM2EQ
!
! PROGRAM FOR SOLVING TWO DIMENSIONAL ELASTIC STRESS ANALYSIS PROBLEMS
! BY THE BOUNDARY ELEMENT METHOD USING QUADRATIC ELEMENTS.
!
USE SHARED DATA2EQ
OPEN(5, FILE="DATA")
OPEN(6, FILE="RESULTS")
OPEN(7, FILE="MESHRES")
PI=4.0*ATAN(1.)
!
! DEFINE THE MAXIMUM PROBLEM SIZE PERMITTED BY THE ARRAY DIMENSIONS.
MAXNEL=250
MAXNNP=500
MAXNB=10
MAXNPC=20
!
! INPUT THE PROBLEM TITLE AND TYPE, ALSO MATERIAL PROPERTIES.
CALL INTITLE
!
! INPUT AND GENERATE THE MESH DATA.
CALL MESHQ
!
! OUTPUT THE MESH DATA.
CALL MSHOUT
!
! EVALUATE AND STORE VALUES OF THE SHAPE FUNCTIONS AND THEIR DERIVATIVES
! AT THE GAUSS POINTS AND NODES.
CALL SHAPE
!
! INPUT, PROCESS AND OUTPUT THE BOUNDARY CONDITIONS.
CALL BCS
!
! FORM THE COEFFICIENT MATRIX AND APPLY THE BOUNDARY CONDITIONS.
CALL FRMTRX
!
```

```

! SOLVE THE LINEAR EQUATIONS.
    NEQN=2*NNP
    MAXNEQN=2*MAXNNP
    MAXNEQNP1=MAXNEQN+1
    CALL ELIMIN(A,UV,NEQN,MAXNEQN,MAXNEQNP1,IFLAG)
    IF(IFLAG == 1) THEN
        WRITE(6,61)
61    FORMAT(/ "MATRIX ILL-CONDITIONING DETECTED IN EQUATION SOLVER")
        STOP
    END IF

!
! OUTPUT THE NODAL DISPLACEMENTS, ELEMENT STRESSES AND FORCES ON THE
! BOUNDARY SEGMENTS.
    CALL OUTPUT

!

STOP
END PROGRAM BEM2EQ

```

After defining the maximum numbers of boundary elements (MAXNEL), nodes (MAXNNP), boundaries (MAXNB) and point constraints (MAXNPC) permitted by the array dimensions, the main program calls the following subprograms in turn:

- INTITLE for the problem title,
- MESHQ to input and create the mesh data,
- MSHOUT to write out the mesh data,
- SHAPE for defining the element shape functions,
- BCS for the boundary conditions,
- FRMTRX to define the $[A]$ and $[B]$ coefficient matrices,
- ELIMIN to solve the equations, and finally
- OUTPUT to write out the results.

As in the potential programs, the matrix ill-conditioning warning is most likely to be triggered by trying to solve a poorly defined problem, such as one with only prescribed stress boundary conditions, and displacement nowhere defined. Computed results from ELIMIN are contained in array UV: typically nodal point displacements, but tractions for nodes subject to prescribed displacements.

6.1.2 Subprogram INTITLE

An alphanumeric title for the problem is first read into TITLE. Next a six character message, either "STRESS" or "STRAIN" is read into CASE, to define whether the domain is in a state of plane stress or plane strain. The default is plane stress: any message other than STRAIN will result in a state of plane stress being assumed. Finally, the values of Young's modulus and Poisson's ratio are read into E and NU. If plane strain is to be assumed, the effective values of these properties are defined according to Equations 5.7.

```
SUBROUTINE  INTITLE
!
!  SUBPROGRAM TO INPUT PROBLEM TITLE AND WHETHER PLANE
!  STRESS OR PLANE STRAIN. ALSO THE MATERIAL PROPERTIES.
!
      USE SHARED DATA2EQ
      CHARACTER(80) :: TITLE
      CHARACTER(6)  :: CASE
!
!  INPUT THE PROBLEM TITLE.
      READ(5,FMT="(A80)") TITLE
      WRITE(6,61) TITLE
```



Discover the truth at www.deloitte.ca/careers

Deloitte.

© Deloitte & Touche LLP and affiliated entities.



Click on the ad to read more

```

61  FORMAT("QUADRATIC BOUNDARY ELEMENT SOLUTION FOR",
      &      " TWO DIMENSIONAL ELASTIC PROBLEM" // A)
!
!  INPUT THE PROBLEM CASE TYPE:
!    "STRESS" DEFINES PLANE STRESS
!    "STRAIN" DEFINES PLANE STRAIN
!  THE DEFAULT IS PLANE STRESS.
      READ(5,FMT="(A6)") CASE
      WRITE(6,62) CASE
62  FORMAT(/"UNDER PLANE ",A," CONDITIONS")
!
!  INPUT AND OUTPUT YOUNG'S MODULUS AND POISSON'S RATIO.
      READ(5,*) E,NU
      WRITE(6,63) E,NU
63  FORMAT(/"YOUNG'S MODULUS = ",E12.4,10X,"POISSON'S RATIO = ",F5.3)
!
!  MODIFY PROPERTIES IF CASE IS ONE OF PLANE STRAIN.
      IF(CASE == "STRAIN") THEN
          E=E/(1.-NU**2)
          NU=NU/(1.-NU)
      END IF
!
      RETURN
      END SUBROUTINE INTITLE

```

6.1.3 Subprogram to input and generate the element mesh data

Subprogram MESHQ is identical to the subprogram described in Section 4.1.3, with the exception that the USE statement is for module SHARED DATA2EQ rather than SHARED DATA2PQ.

```

      SUBROUTINE MESHQ
!
!  SUBPROGRAM TO READ IN AND GENERATE THE GEOMETRIC DATA FOR A MESH OF
!  QUADRATIC ELEMENTS.
!
      USE SHARED DATA2EQ
!
!  INPUT THE NUMBER OF SEPARATE BOUNDARIES.
      READ(5,*) NBOUND
!

```

```
! TEST THE NUMBER OF BOUNDARIES.
      IF(NBOUND < 1 .OR. NBOUND > MAXNB) THEN
        WRITE(6,61) NBOUND,MAXNB
61      FORMAT(/ "NBOUND =",I4,2X,"OUTSIDE PERMITTED RANGE 1 TO",I4)
        STOP
      END IF
!
! FOR EACH BOUNDARY IN TURN INPUT THE NUMBER OF SEGMENTS.
      NEL=0
      IEEND=0
      NSEGTOT=0
      MMAXB=0
      Each boundary in turn: DO IBOUND=1,NBOUND
        NELB(IBOUND)=0
        MMINB=MMAXB+1
        READ(5,*) NSEGB(IBOUND)
        NSEGTOT=NSEGTOT+NSEGB(IBOUND)
!
! TEST THE NUMBER OF SEGMENTS.
      IF(NSEGTOT < 1 .OR. NSEGTOT > MAXNEL) THEN
        WRITE(6,62) NSEGTOT,MAXNEL
62      FORMAT(/ "NSEGTOT =",I6,2X,"OUTSIDE PERMITTED RANGE 1 TO",I6)
        STOP
      END IF
!
! INPUT THE CARTESIAN GLOBAL COORDINATES OF THE END POINTS OF THE
! SEGMENTS. TAKE THE END POINTS CONSECUTIVELY, KEEPING THE DOMAIN
! TO THE LEFT OF THE DIRECTION OF NUMBERING.
      READ(5,*) (XSEND(ISEND),YSEND(ISEND),ISEND=1,NSEGB(IBOUND))
!
! DEFINE THE FIRST END POINT ON THE CURRENT BOUNDARY.
      IEEND=IEEND+1
      XEEND(IEEND)=XSEND(1)
      YEEND(IEEND)=YSEND(1)
!
! FOR EACH OF THE SEGMENTS (BETWEEN ENDS 1 AND 2, 2 AND 3, ETC.)
! INPUT THE RADIUS OF CURVATURE (+VE FOR CONVEX WITH CENTRE OF
! CURVATURE INSIDE DOMAIN, -VE FOR CONCAVE), THE NUMBER OF
! ELEMENTS IN THE SEGMENT, AND THE LENGTH RATIO BETWEEN SUCCESSIVE
```



```

!  ELEMENTS IN THE DIRECTION OF NUMBERING.
      ISEGMAX=NSEGTOT
      ISEGMIN=ISEGMAX-NSEGB(IBOUND)+1
      Each segment in turn: DO ISEG=ISEGMIN,ISEGMAX
      READ(5,*) RSEG,NELSEG,RATSEG
!
!  FIND AND TEST THE NUMBER OF ELEMENTS SO FAR.
      NEL=NEL+NELSEG
      NELB(IBOUND)=NELB(IBOUND)+NELSEG
      IF(NEL < 1 .OR. NEL > MAXNEL) THEN
        WRITE(6,63) NEL,MAXNEL
63      FORMAT(/ "NEL =",I6,2X,"OUTSIDE PERMITTED RANGE 1 TO",I6)
        STOP
      END IF
!
!  FIRST AND LAST ELEMENTS ON CURRENT SEGMENT.
      MLAST(ISEG)=NEL
      MFIRST(ISEG)=NEL-NELSEG+1
      MMAXB=MMAXB+NELSEG
!

```



**I WANT TO CHANGE DIRECTION,
AND THE WORLD.**

GOT-THE-ENERGY-TO-LEAD.COM

We believe that energy suppliers should be renewable, too. We are therefore looking for enthusiastic new colleagues with plenty of ideas who want to join RWE in changing the world. Visit us online to find out what we are offering and how we are working together to ensure the energy of the future.

RWE
The energy to lead



```

! COORDINATES OF THE FIRST END POINT OF THE SEGMENT.
    ISEND=ISEG-ISEGMIN+1
    XFIRST=XSEND (ISEND)
    YFIRST=YSEND (ISEND)
!
! COORDINATES OF THE LAST END POINT OF THE SEGMENT.
    ISEND=ISEND+1
    IF (ISEG == ISEGMAX) ISEND=1
    XLAST=XSEND (ISEND)
    YLAST=YSEND (ISEND)
!
! GENERATE ELEMENT DATA FOR A STRAIGHT SEGMENT.
    IF (RSEG == 0.) THEN
!
! DEFINE THE ELEMENT END POINT COORDINATES ON THE SEGMENT.
    Each element in turn: DO M=1,NELSEG
        IEEND=IEEND+1
        ISEGEND (IEEND)=ISEG
        IF (RATSEG == 1.) THEN
            XEEND (IEEND)=XFIRST+ (XLAST-XFIRST) *FLOAT (M) /FLOAT (NELSEG)
            YEEND (IEEND)=YFIRST+ (YLAST-YFIRST) *FLOAT (M) /FLOAT (NELSEG)
        ENDIF
        IF (RATSEG /= 1.) THEN
            XEEND (IEEND)=XFIRST+ (XLAST-XFIRST) * (1.-RATSEG**M)
&                / (1.-RATSEG**NELSEG)
            YEEND (IEEND)=YFIRST+ (YLAST-YFIRST) * (1.-RATSEG**M)
&                / (1.-RATSEG**NELSEG)
        END IF
    END DO Each element in turn
!
! DEFINE THE ELEMENT NODES AND COORDINATES.
    IEEND=IEEND-NELSEG
    Each element in turn: DO IELSEG=1,NELSEG
        IEEND=IEEND+1
        M=MFIRST (ISEG)+IELSEG-1
        I1=2*M-1
        I2=I1+1
        I3=I1+2
        IF (ISEG == ISEGMAX .AND. IELSEG == NELSEG) I3=NODE (MMINB,1)

```

```

        NODE (M, 1) = I1
        NODE (M, 2) = I2
        NODE (M, 3) = I3
        ISEGELEM (M) = ISEG
        IF (ISEG == ISEGMIN .AND. IELSEG == 1) THEN
            XNODE (I1) = XEEND (IEEND-1)
            YNODE (I1) = YEEND (IEEND-1)
        END IF
        XNODE (I3) = XEEND (IEEND)
        YNODE (I3) = YEEND (IEEND)
        XNODE (I2) = 0.5 * (XNODE (I1) + XNODE (I3))
        YNODE (I2) = 0.5 * (YNODE (I1) + YNODE (I3))
!
!  STORE THE NUMBERS OF THE ADJACENT ELEMENTS.
        M1 (M) = M-1
        M3 (M) = M+1
        END DO Each element in turn
    END IF
!
!  GENERATE ELEMENT DATA FOR A SEGMENT IN THE FORM OF A CIRCULAR ARC.
        IF (RSEG /= 0.) THEN
!
!  LOCATE THE CENTRE OF THE ARC.
            XMID = (XFIRST + XLAST) / 2.
            YMID = (YFIRST + YLAST) / 2.
            ALSEG = SQRT ((XLAST - XFIRST) ** 2 + (YLAST - YFIRST) ** 2)
            ALPERP2 = RSEG ** 2 - (ALSEG / 2.) ** 2
            IF (ABS (ALPERP2) < 1.E-6 * RSEG ** 2) ALPERP2 = 0.
            IF (ALPERP2 < -1.E-6 * RSEG ** 2) THEN
                WRITE (6, 64) ISEG
64      FORMAT (/ "DATA ERROR FOR SEGMENT NUMBER", I6,
&              / "NOT POSSIBLE TO CREATE A CIRCULAR ARC")
                STOP
            END IF
            ALPERP = SQRT (ALPERP2)
            UVFLX = (XLAST - XFIRST) / ALSEG
            UVFLY = (YLAST - YFIRST) / ALSEG
            FACT = 1.
            IF (RSEG < 0.) FACT = -1.

```

```
XCENT=XMID-ALPERP*UVFLY*FACT
YCENT=YMID+ALPERP*UVFLX*FACT
!
!  FIND THE ANGLE SUBTENDED THERE BY THE SEGMENT.
    IF (ALPERP /= 0.) ANGSEG=2.*ATAN (ALSEG*0.5/ALPERP)
    IF (ALPERP == 0.) ANGSEG=PI
!
!  DEFINE THE ELEMENT END POINT COORDINATES ON THE SEGMENT.
    ANGFI=ATAN2 (YFIRST-YCENT,XFIRST-XCENT)
    ANGSTORE (IEEND)=0.
    Each element in turn: DO M=1,NELSEG
    IEEND=IEEND+1
    ISEGEN (IEEND)=ISEG
    IF (RATSEG == 1.) ANG=ANGSEG*FLOAT (M) /FLOAT (NELSEG)
    IF (RATSEG /= 1.) ANG=ANGSEG* (1.-RATSEG**M) / (1.-RATSEG**NELSEG)
    IF (RSEG < 0.) ANG=-ANG
    XEEND (IEEND)=XCENT+ABS (RSEG) *COS (ANGFI+ANG)
    YEEND (IEEND)=YCENT+ABS (RSEG) *SIN (ANGFI+ANG)
    ANGSTORE (IEEND)=ANG
    END DO Each element in turn
```

bookboon.com

Corporate eLibrary

See our Business Solutions for employee learning

[Click here](#)



[Click on the ad to read more](#)

```

!
!  DEFINE THE ELEMENT NODES AND COORDINATES.
      IEEND=IEEND-NELSEG
      Each element in turn: DO IELSEG=1,NELSEG
      IEEND=IEEND+1
      M=MFIRST(ISEG)+IELSEG-1
      I1=2*M-1
      I2=I1+1
      I3=I1+2
      IF( ISEG == ISEGMAX .AND. IELSEG == NELSEG) I3=NODE(MMINB,1)
      NODE(M,1)=I1
      NODE(M,2)=I2
      NODE(M,3)=I3
      ISEGELEM(M)=ISEG
      IF( ISEG == ISEGMIN .AND. IELSEG == 1) THEN
        XNODE(I1)=XEEND(IEEND-1)
        YNODE(I1)=YEEND(IEEND-1)
      END IF
      XNODE(I3)=XEEND(IEEND)
      YNODE(I3)=YEEND(IEEND)
      ANG=0.5*(ANGSTORE(IEEND-1)+ANGSTORE(IEEND))
      XNODE(I2)=XCENT+ABS(RSEG)*COS(ANGFIR+ANG)
      YNODE(I2)=YCENT+ABS(RSEG)*SIN(ANGFIR+ANG)
!
!  STORE THE NUMBERS OF THE ADJACENT ELEMENTS.
      M1(M)=M-1
      M3(M)=M+1
      END DO Each element in turn
    END IF
!
    END DO Each segment in turn
!
!  ADJACENT ELEMENTS FOR END ELEMENTS OF THE BOUNDARY.
      M1(MMINB)=MMAXB
      M3(MMAXB)=MMINB
      END DO Each boundary in turn
      NEEND=IEEND
      NNP=NEL*2
!

```

```

!   DETERMINE THE MAXIMUM DIMENSION OF THE SOLUTION DOMAIN.
      MAXL=0.
      Each node in turn: DO I=1,NNP
      Each other node in turn: DO J=1,NNP
      DIST=SQRT((XNODE(I)-XNODE(J))**2+(YNODE(I)-YNODE(J))**2)
      IF(DIST > MAXL) MAXL=DIST
      END DO Each other node in turn
      END DO Each node in turn
!
      RETURN
      END SUBROUTINE MESHQ

```

6.1.4 Mesh data output subprogram

Subprogram MSHOUT is identical to the subprogram described in Section 4.1.4, with the exception that the USE statement is for module SHARED DATA2EQ rather than SHARED DATA2PQ.

```

      SUBROUTINE MSHOUT
!
!   SUBPROGRAM TO WRITE OUT THE MESH DATA.
!
      USE SHARED DATA2EQ
!
!   OUTPUT THE NUMBERS OF ELEMENTS AND NODES, ALSO THE NODAL
!   COORDINATES.
      WRITE(7,71) NEL,NNP,(I,XNODE(I),YNODE(I),I=1,NNP)
71  FORMAT(/ "GEOMETRIC DATA FOR THE MESH" //
& 10X,"NUMBER OF ELEMENTS =",I6 //
& 10X,"NUMBER OF NODAL POINTS =",I6 //
& "COORDINATES OF THE NODAL POINTS"//
& 2(" I X Y ") /
& 2(I6,2E12.4))
!
!   OUTPUT THE ELEMENT NODE NUMBERS.
      WRITE(7,72) (M,(NODE(M,IN),IN=1,3),M=1,NEL)
72  FORMAT(/ 'ELEMENT NODE NUMBERS' //
& 1X,2(10X,'M ND1 ND2 ND3')/ (2(7X,4I5)))
!
!   SCALE THE NODAL POINT COORDINATES.
      Each node in turn: DO I=1,NNP
      XNODE(I)=XNODE(I)/MAXL

```

```
YNODE(I)=YNODE(I)/MAXL  
END DO Each node in turn  
  
!  
  
RETURN  
  
END SUBROUTINE MSHOUT
```

6.1.5 Subprogram for defining shape functions

Subprogram SHAPE is identical to the subprogram described in Section 4.1.6, with the exception that the USE statement is for module SHARED DATA2EQ rather than SHARED DATA2PQ.

```
SUBROUTINE SHAPE  
  
!  
! SUBPROGRAM TO EVALUATE AND STORE VALUES OF THE SHAPE FUNCTIONS AND  
! THEIR DERIVATIVES, AT THE GAUSS POINTS AND NODES.  
!  
  
USE SHARED DATA2EQ  
  
!  
! STORE APPROPRIATE COORDINATES AND WEIGHT FACTORS FOR NORMAL GAUSSIAN  
! QUADRATURE IN ARRAYS XG AND CG, ALSO THOSE FOR LOGARITHMIC FUNCTION  
! INTEGRATION IN XGL AND CGL.
```



Brain power

By 2020, wind could provide one-tenth of our planet's electricity needs. Already today, SKF's innovative know-how is crucial to running a large proportion of the world's wind turbines.

Up to 25 % of the generating costs relate to maintenance. These can be reduced dramatically thanks to our systems for on-line condition monitoring and automatic lubrication. We help make it more economical to create cleaner, cheaper energy out of thin air.

By sharing our experience, expertise, and creativity, industries can boost performance beyond expectations. Therefore we need the best employees who can meet this challenge!

The Power of Knowledge Engineering

Plug into The Power of Knowledge Engineering.
Visit us at www.skf.com/knowledge

SKF


```
NGAUSS=8
ZG(1)=-0.9602898564
ZG(2)=-0.7966664774
ZG(3)=-0.5255324099
ZG(4)=-0.1834346424
ZG(5)=-ZG(4)
ZG(6)=-ZG(3)
ZG(7)=-ZG(2)
ZG(8)=-ZG(1)
WG(1)=0.1012285362
WG(2)=0.2223810344
WG(3)=0.3137066458
WG(4)=0.3626837833
WG(5)=WG(4)
WG(6)=WG(3)
WG(7)=WG(2)
WG(8)=WG(1)
EGL(1)=0.013320244
EGL(2)=0.079750429
EGL(3)=0.197871029
EGL(4)=0.354153994
EGL(5)=0.529458575
EGL(6)=0.701814530
EGL(7)=0.849379320
EGL(8)=0.953326450
WGL(1)=0.164416605
WGL(2)=0.237525610
WGL(3)=0.226841984
WGL(4)=0.175754079
WGL(5)=0.112924030
WGL(6)=0.057872211
WGL(7)=0.020979074
WGL(8)=0.003686407
!
!  NORMAL GAUSSIAN QUADRATURE.
  For each Gauss point in turn: DO IGAUSS=1,NGAUSS
    ZETA=ZG(IGAUSS)
    SF(1,IGAUSS)=0.5*ZETA*(ZETA-1.)
    SF(2,IGAUSS)=1.-ZETA**2
    SF(3,IGAUSS)=0.5*ZETA*(ZETA+1.)
```

```

SD(1,IGAUSS)=ZETA-0.5
SD(2,IGAUSS)=-2.*ZETA
SD(3,IGAUSS)=ZETA+0.5
END DO For each Gauss point in turn
!
! FOUR CASES OF LOGARITHMIC GAUSSIAN QUADRATURE TO CONSIDER.
! IC=1 - INTEGRATION OVER WHOLE ELEMENT FROM FIRST TO THIRD NODE.
! IC=2 - INTEGRATION OVER HALF ELEMENT FROM SECOND TO THIRD NODE.
! IC=3 - INTEGRATION OVER HALF ELEMENT FROM SECOND TO FIRST NODE.
! IC=4 - INTEGRATION OVER WHOLE ELEMENT FROM THIRD TO FIRST NODE.
!
For each case in turn: DO IC=1,4
For each Gauss point in turn: DO IGAUSS=1,NGAUSS
ETA=EGL(IGAUSS)
IF(IC == 1) ZETA=2.*ETA-1.
IF(IC == 2) ZETA=ETA
IF(IC == 3) ZETA=-ETA
IF(IC == 4) ZETA=1.-2.*ETA
SFL(IC,1,IGAUSS)=0.5*ZETA*(ZETA-1.)
SFL(IC,2,IGAUSS)=1.-ZETA**2
SFL(IC,3,IGAUSS)=0.5*ZETA*(ZETA+1.)
SDL(IC,1,IGAUSS)=ZETA-0.5
SDL(IC,2,IGAUSS)=-2.*ZETA
SDL(IC,3,IGAUSS)=ZETA+0.5
END DO For each Gauss point in turn
END DO For each case in turn
!
! SHAPE FUNCTION DERIVATIVES AT THE NODES, STORED AS THOUGH THEY
! ARE FOR GAUSS POINTS NUMBERED 11, 12 AND 13.
Each element node in turn: DO IGAUSS=11,13
IF(IGAUSS == 11) ZETA=-1.
IF(IGAUSS == 12) ZETA=0.
IF(IGAUSS == 13) ZETA=1.
SD(1,IGAUSS)=ZETA-0.5
SD(2,IGAUSS)=-2.*ZETA
SD(3,IGAUSS)=ZETA+0.5
END DO Each element node in turn
!
RETURN
END SUBROUTINE SHAPE

```

6.1.6 Subprogram for applying the boundary conditions

The subprogram BCS serves to apply boundary conditions of either the prescribed displacement or prescribed stress types. As already indicated, it is assumed that each segment of elements has a uniform boundary condition applied to it, which is an important consideration when defining the segments. Also applied are any point displacement constraints. The total numbers of segments subject to the first two types of condition, and the number of point constraints, are first read into variables NBCU, NBCS and NBCPC, respectively. In the case of prescribed stresses it is only necessary to include those segments subject to non-zero stresses.

Prior to reading in the boundary conditions, the components in the global co-ordinate directions of the unit normals at every node are calculated with the aid of subprogram JACOBI. Because the direction of the normal to the boundary may be discontinuous at an element end node, particularly at a corner of the domain, the calculation is performed for every node of every element, and stored in arrays UNMX and UNMY.

With us you can
shape the future.
Every single day.

For more information go to:
www.eon-career.com

Your energy shapes the future.

e-on



Storage arrays for normal and shear stresses, SIGNN and SIGSN, and tractions in the global co-ordinate directions, TMY and TMY, for every node of every element are set to zero. Array IBCE stores the type of boundary condition (1 or 2 for the two types, prescribed displacement or prescribed stress) for each element. In the absence of other information, the condition at a node is assumed to be zero stresses. The default boundary condition type is therefore set as 2, with the corresponding zero values of stresses for the elements being stored in arrays TMX and TMY.

```

      SUBROUTINE  BCS
!
!  SUBPROGRAM TO INPUT, PROCESS AND OUTPUT THE BOUNDARY CONDITIONS.
!
      USE SHARED DATA2EQ
!
!  FIRST FIND THE UNIT NORMAL COMPONENTS AT THE ELEMENT NODES.
      Each element in turn: DO M=1,NEL
      Each element node in turn: DO IN=1,3
      IGAUSS=IN+10
      IT=1
      IC=1
      CALL JACOBI(M, IGAUSS, IT, IC)
      UNMX(M, IN)=UNGX
      UNMY(M, IN)=UNGY
      END DO Each element node in turn
      END DO Each element in turn
!
!  INPUT THE NUMBERS OF SEGMENTS SUBJECT TO EACH TYPE OF BOUNDARY
!  CONDITION. ALSO THE NUMBER OF POINT CONSTRAINTS.
!      NBCU - PRESCRIBED DISPLACEMENTS.
!      NBCS - NON-ZERO PRESCRIBED STRESSES.
!  ANY SEGMENT NOT INCLUDED IS ASSUMED TO BE SUBJECT TO ZERO
!  STRESSES.
!      NBCPC - NODAL POINT DISPLACEMENT CONSTRAINTS.
!
      READ(5,*) NBCU,NBCS,NBCPC
!
!  TEST THESE BOUNDARY CONDITION NUMBERS.
      NBCT=NBCU+NBCS
      IF(NBCU < 0 .OR. NBCU > MAXNEL .OR. NBCS < 0 .OR. NBCS > MAXNEL
&          .OR. NBCT < 0 .OR. NBCT > MAXNEL) THEN
          WRITE(6,61) NBCU,NBCS,NBCT,MAXNEL

```

```

61      FORMAT(/ "NBCU =", I6, 3X, "NBCS =", I6, 3X, / "NBCT =", I6, 3X,
&          "OUTSIDE PERMITTED RANGE 0 TO", I6)
      STOP
    END IF
    IF (NBCPC < 0 .OR. NBCPC > MAXNPC) THEN
      WRITE(6, 62) NBCPC, MAXNPC
62      FORMAT(/ "NBCPC =", I4, 3X, "OUTSIDE PERMITTED RANGE 0 TO", I3)
      STOP
    END IF
!
!  INITIALISE THE BOUNDARY CONDITION STORAGE ARRAYS.
    Each element in turn: DO M=1, NEL
      IBCE(M)=2
      Each element node in turn: DO IN=1, 3
        SIGNN(M, IN)=0.
        SIGSN(M, IN)=0.
        TMX(M, IN)=0.
        TMY(M, IN)=0.
      END DO Each element node in turn
    END DO Each element in turn
!
!  INPUT, STORE AND OUTPUT THE PRESCRIBED DISPLACEMENT CONDITIONS.
    IF (NBCU > 0) THEN
      READ(5, *) (ISEGBC(IBCUC), USEG(IBCUC), VSEG(IBCUC), IBCUC=1, NBCU)
      WRITE(6, 63)
63      FORMAT(/ "PRESCRIBED DISPLACEMENT BOUNDARY CONDITIONS")
      Each segment with prescribed displacements: DO IBCUC=1, NBCU
        ISEG=ISEGBC(IBCUC)
        IF (ISEG < 1 .OR. ISEG > NSEGTOT) THEN
          WRITE(6, 64) ISEG, NSEGTOT
64      FORMAT(/ "ISEG = ", I6, 2X, "OUTSIDE PERMITTED RANGE 1 TO", I6)
          STOP
        END IF
        Each element on current segment: DO M=MFIRST(ISEG), MLAST(ISEG)
          IBCE(M)=1
          UELEM(M)=USEG(IBCUC)
          VELEM(M)=VSEG(IBCUC)
        END DO Each element on current segment
!
      WRITE(6, 65) USEG(IBCUC), VSEG(IBCUC), MFIRST(ISEG), MLAST(ISEG)

```

```

65      FORMAT(/ "U =",E12.4,3X,"V =",E12.4,3X,"ON ELEMENTS",I4,3X,
&          "TO",I4)
      END DO Each segment with prescribed displacements
END IF

!
! INPUT, STORE AND OUTPUT THE PRESCRIBED STRESS CONDITIONS.
      IF(NBCS > 0) THEN
          READ(5,*) (ISEGBC(IBCS),SIGNNSEG(IBCS),SIGSNSEG(IBCS),
&          IBCS=1,NBCS)
          WRITE(6,66)
66      FORMAT(/ "PRESCRIBED STRESS BOUNDARY CONDITIONS")
          Each segment with prescribed stresses: DO IBCS=1,NBCS
              ISEG=ISEGBC(IBCS)
              IF(ISEG < 1 .OR. ISEG > NSEGTOT) THEN
                  WRITE(6,64) ISEG,NSEGTOT
                  STOP
              END IF
              Each element on current segment: DO M=MFIRST(ISEG),MLAST(ISEG)
                  IBCE(M)=2
!

```



© 2013 Accenture. All rights reserved.

be > your degree

Bring your talent and passion to a global organization at the forefront of business, technology and innovation. Discover how great you can be.

Visit accenture.com/bookboon

Be greater than.
consulting | technology | outsourcing

accenture
High performance. Delivered.



```

! FIND THE TRACTIONS AT THE ELEMENT NODES FROM THE PRESCRIBED STRESSES.
! ALSO STORE THE STRESSES AT THE ELEMENT NODES.
    Each element node in turn: DO IN=1,3
    TMX(M,IN)=SIGNNSEG(IBC5)*UNMX(M,IN)-SIGSNSEG(IBC5)*UNMY(M,IN)
    TMY(M,IN)=SIGNNSEG(IBC5)*UNMY(M,IN)+SIGSNSEG(IBC5)*UNMX(M,IN)
    SIGNN(M,IN)=SIGNNSEG(IBC5)
    SIGSN(M,IN)=SIGSNSEG(IBC5)
    END DO each element node in turn
    END DO Each element on current segment
!
    WRITE(6,67) SIGNNSEG(IBC5),SIGSNSEG(IBC5),MFIRST(ISEG),
&
    MLAST(ISEG)
67    FORMAT(/ "SIGNN =",E12.4,3X,"SIGSN =",E12.4,3X,"ON ELEMENTS",
&
    I4,3X,"TO",I4)
    END DO Each segment with prescribed stresses
    END IF
!
! ASSEMBLE THE VECTOR OF KNOWN VARIABLES, APPLYING SCALING.
! FIRST INITIALISE THE NODAL DISPLACEMENTS AND TRACTIONS.
    Each node in turn: DO I=1,NNP
    U(I)=0.
    V(I)=0.
    TX(I)=0.
    TY(I)=0.
    END DO Each node in turn
    Each element in turn: DO M=1,NEL
    Each element node in turn: DO IN=1,3
    I=NODE(M,IN)
    IBCN(I)=IBCE(M)
!
! CHECK THE CONDITION APPLIED TO THE ADJACENT ELEMENT FOR AN ELEMENT
! END NODE. IF THE CONDITION IS PRESCRIBED DISPLACEMENTS BUT THE
! EXISTING CONDITION AT THE NODE IS NOT, THEN IMPOSE PRESCRIBED
! DISPLACEMENTS.
    MADJ=M
    IF(IN == 1) MADJ=M1(M)
    IF(IN == 3) MADJ=M3(M)
    IF(IBCE(MADJ) == 1 .AND. IBCN(I) == 2) IBCN(I)=1
!

```



```

!   STORE KNOWN VARIABLES.
      IF(IBCNC(I) == 1) THEN
        IF(IBCCE(M) == 1) THEN
          IF(U(I) == 0.) U(I)=UELEM(M)/MAXL
          IF(V(I) == 0.) V(I)=VELEM(M)/MAXL
          IF(U(I) /= 0.) U(I)=0.5*(U(I)+UELEM(M)/MAXL)
          IF(V(I) /= 0.) V(I)=0.5*(V(I)+VELEM(M)/MAXL)
        END IF
      END IF
END DO Each element node in turn
END DO Each element in turn

!
!   SCALE YOUNG'S MODULUS.
      ESTORE=E
      E=1.

!
!   INPUT, STORE AND OUTPUT NODAL POINT DISPLACEMENT CONSTRAINTS.
!   IDIRPC STORES DIRECTIONS OF THE CONSTRAINTS - 1 FOR X, 2 FOR Y.
!   SUCH CONSTRAINTS SHOULD NOT REQUIRE POINT FORCES TO MAINTAIN THEM.
      IF(NBCU == 0 .AND. NBCPC < 3) THEN
        WRITE(6,68)
68      FORMAT(/ "INSUFFICIENT DISPLACEMENT BOUNDARY CONDITIONS",
&           " OR POINT CONSTRAINTS")
        STOP
      END IF
      IF(NBCPC > 0) THEN
        READ(5,*) (NODEPC(IBCPC),IDIRPC(IBCPC),IBCPC=1,NBCPC)
        Each point constraint in turn: DO IBCPC=1,NBCPC
          IF(NODEPC(IBCPC) < 0 .OR. NODEPC(IBCPC) > NNP) THEN
            WRITE(6,69) NODEPC(IBCPC),NNP
69      FORMAT(/ "POINT CONSTRAINT NODE",I4," OUTSIDE RANGE 0 TO ",I4)
            STOP
          END IF
          IF(IDIRPC(IBCPC) /= 1 .AND. IDIRPC(IBCPC) /= 2) THEN

```

```

WRITE(6,70) IDIRPC(IBCPC),NODEPC(IBCPC)
70  FORMAT(/ "POINT CONSTRAINT DIRECTION",I3," FOR NODE",I4,
&          " OUTSIDE RANGE 1 TO 2")
      STOP
      END IF
      IF(IDIRPC(IBCPC) == 1) WRITE(6,71) NODEPC(IBCPC)
71  FORMAT(/ "NODE",I4," CONSTRAINED WITH U = 0")
      IF(IDIRPC(IBCPC) == 2) WRITE(6,72) NODEPC(IBCPC)
72  FORMAT(/ "NODE",I4," CONSTRAINED WITH V = 0")
      END DO Each point constraint in turn
      END IF
!
      RETURN
      END SUBROUTINE BCS

```

For each segment over which displacements are prescribed, the segment number and values of displacements are read into ISEGBC, USEG and VSEG, respectively. Taking each of these segments in turn, the elements have their boundary condition type set to 1 (in array IBCE), and the values of displacements are stored in arrays UELEM and VELEM. The prescribed displacements and numbers of the elements to which they are applied are then written out.

For each segment over which stresses are prescribed, the segment number and values of normal and shear stresses are read into ISEGBC, SIGNNSEG and SIGSNSEG, respectively. Taking each of these segments in turn, the elements have their boundary condition type set to 2 (in array IBCE). The values of the stresses are both stored as nodal point stresses in arrays SIGNN and SIGNS and converted to tractions at the nodes of the elements according to Equations 5.58 and 5.59 and stored in arrays TMX and TMY. The prescribed stresses and numbers of the elements to which they are applied are then written out.

For each node in turn, the known variable values are stored: displacements in arrays U and V, or tractions in arrays TX and TY. Displacements are scaled by dividing by MAXL, the maximum domain dimension. Tractions are scaled by dividing by Young's modulus. The type of boundary condition at the node is stored in array IBCN, obtained from that for the corresponding element. At the ends of segments, which may correspond to physical corners in the problem or may correspond only to changes in boundary conditions, this can give rise to an ambiguity in type at a node. At a node at which there is a change from displacement to stress boundary condition the former is chosen, which is why in the program there is a test for displacement boundary condition on the current element, but stress boundary condition at the current node (which can arise from the way the boundary conditions have been applied to the elements). The displacement boundary condition type is imposed by setting the relevant value in array IBCN to 1.

If at a node linking two segments there is a change, not in boundary condition type but in the values of either displacements or stresses, it is appropriate to assign average values to the node and store these as the known variables for that node. This is achieved in the program by testing every node to see whether both the node and element boundary condition is of the same type. If it is, the known variables are taken as the stored values for the element if the known variable values are zero. This will apply to all element midside nodes, and to all element end nodes which have not been tested in previous elements. If the known variable values are not zero, which can only occur if prescribed values are applied to the immediately adjacent element and that conditions at the nodes of the element have already been tested, then the average of the present element values and the stored values are re-stored. This works both for adjacent elements with different values of displacement or tractions and for the much more common case of adjacent elements with the same value prescribed: the common value is re-stored.

This averaging process is really a detail to give sensible values of the variables in the output of the final results. What is really important is that the prescribed displacements and tractions are applied element by element in forming the $[b]$ column vector in Equations 5.57, so that each of the adjacent elements contributes according to its own prescribed values. This will be explained further in connection with subprogram FRMTRX.

In preparation for using scaled values of the displacement kernels, the value of Young's modulus is set to 1, having stored the actual value in the variable ESTORE.



"I studied English for 16 years but...
...I finally learned to speak it in just six lessons"

Jane, Chinese architect

ENGLISH OUT THERE

Click to hear me talking before and after my unique course download



Finally in subprogram BCS the nodal point displacement constraints are read in: the node number into NODEPC and the direction of the constraint into IDIRPC. Direction 1 is the x direction and direction 2 is the y direction. A value of 1 means that the node is constrained not to move in the x direction, 2 that it cannot move in the y direction. In the absence of any displacement boundary conditions, at least 3 (and often only 3) point constraints are required to prevent movement of the domain as a rigid body, without at the same time unnaturally restricting deformation of the elastic component under consideration. Typically, one node is constrained in both co-ordinate directions, while another which is suitably located is constrained to move only in the direction which is radial to the fully constrained node. Before the point constraint data are read in, a test is applied to the numbers of displacement boundary conditions and point constraints in the subprogram, followed by a warning message where necessary.

6.1.7 Subprogram to form the coefficient matrix

Subprogram FRMTRX forms the coefficient matrix and right hand side vector of Equations 5.57, and stores them as an extended matrix in array A. The extended matrix is simply $[A^*]$ with an extra column to contain column vector $[b]$. Each node in turn is treated as point P , the force point for the fundamental solution, then integration is carried out over every element in turn. This integration involves the knowns and unknowns at each of the three nodes of each element. Node counters I and J are used for P and the current element node, respectively, so that $2*I-1$ and $2*I$ are the numbers of the two rows in $[A^*]$ corresponding to P , while $2*J-1$ and $2*J$ are the numbers of the two columns in $[A^*]$ corresponding to the element node. The actual integrations of the kernel function products are carried out in subprogram INTKER which is called by FRMTRX. What INTKER does is to compute the integrals of the traction and displacement kernel functions required in Equations 5.78 and 5.79. If the knowns and unknowns at the th node of the current element are tractions and displacements, then these integral quantities will contribute to the $[A]$ and $[B]$ matrices, respectively. Initially they are stored (in subprogram INTKER) in the two-dimensional arrays AROWX and AROWY as the coefficients that will form the current pair of rows of $[A]$, and BROWX and BROWY as the coefficients that will form the current pair of rows of $[B]$, but with the contributions from every node of every element kept distinct. The two rows forming the pair for each node are for concentrated force applied at the force point P in the x direction (AROWX and BROWX) and y direction (AROWY and BROWY).

```

      SUBROUTINE  FRMTRX
      !
      !  SUBPROGRAM TO FORM THE COEFFICIENT MATRIX AND RIGHT HAND SIDE VECTOR,
      !  MODIFIED TO SUIT THE BOUNDARY CONDITIONS.
      !
      USE SHARED DATA2EQ
      !
      !  DEFINE THE NUMBER OF COLUMNS IN THE EXTENDED COEFFICIENT MATRIX A.
      JMAX=2*NNP+1
      !

```

```

! FORM THE COEFFICIENT MATRIX A, AND THE RIGHT HAND SIDE VECTOR B*T.
    Take each node in turn as P: DO I=1,NNP
!
! INITIALISE THE TWO ROWS OF MATRIX A CORRESPONDING TO THE NODE.
    Each pair of coefficients in turn: DO J=1,JMAX
        A(2*I-1,J)=0.
        A(2*I,J)=0.
    END DO Each pair of coefficients in turn
!
! INITIALISE THE ELEMENT CONTRIBUTIONS TO THE CURRENT ROWS OF THE
! A AND B MATRICES.
    Each element in turn: DO M=1,NEL
        Each element node in turn: DO IN=1,3
            AROWX(M,2*IN-1)=0.
            AROWX(M,2*IN)=0.
            AROWY(M,2*IN-1)=0.
            AROWY(M,2*IN)=0.
            BROWX(M,2*IN-1)=0.
            BROWX(M,2*IN)=0.
            BROWY(M,2*IN-1)=0.
            BROWY(M,2*IN)=0.
        END DO Each element node in turn
    END DO Each element in turn
!
! SET UP THE LOOP TO INTEGRATE OVER EACH ELEMENT IN TURN.
    Each element in turn: DO M=1,NEL
!
! INTEGRATE THE KERNEL PRODUCTS OVER THE CURRENT ELEMENT.
        CALL INTKER(I,M)
    END DO Each element in turn
!
! EVALUATE THE COEFFICIENTS AT THE DIAGONAL OF MATRIX A.
    AIIXX=0.
    AIIXY=0.
    AIIYX=0.
    AIIYY=0.
    Each element in turn: DO M=1,NEL
        Each element node in turn: DO IN=1,3
            AIIXX=AIIXX-AROWX(M,2*IN-1)

```

```

      AIIXY=AIIXY-AROWX (M,2*IN)
      AIIYX=AIIYX-AROWY (M,2*IN-1)
      AIIYY=AIIYY-AROWY (M,2*IN)
      END DO Each element node in turn
      END DO Each element in turn
      IF (IBCN(I) == 2) THEN
        A(2*I-1,2*I-1)=AIIXX
        A(2*I-1,2*I)=AIIXY
        A(2*I,2*I-1)=AIIYX
        A(2*I,2*I)=AIIYY
      END IF
!
!  INITIALISE THE B*T VECTOR COEFFICIENTS.
      BTX=0.
      BTY=0.
      IF (IBCN(I) == 1) THEN
        BTX=-AIIXX*U(I)-AIIXY*V(I)
        BTY=-AIIYX*U(I)-AIIYY*V(I)
      END IF
!

```

DUKE
THE FUQUA
SCHOOL
OF BUSINESS

BUSINESS HAPPENS

www.fuqua.duke.edu/globalmba

Learn More >

HERE.



Click on the ad to read more

```

!   APPLY THE BOUNDARY CONDITIONS TO THE CURRENT ROWS OF A AND B, BY
!   CONSIDERING EACH ELEMENT IN TURN.
      Each element in turn: DO M=1,NEL
      Each element node in turn: DO IN=1,3
      J=NODE(M,IN)

!
!   IF DISPLACEMENTS ARE PRESCRIBED OVER THE ELEMENT,
!   INTERCHANGE THE A AND B COEFFICIENTS.
      IF (IBCE(M) == 1) THEN
        A(2*I-1,2*J-1)=A(2*I-1,2*J-1)-BROWX(M,2*IN-1)
        A(2*I-1,2*J)=A(2*I-1,2*J)-BROWX(M,2*IN)
        A(2*I,2*J-1)=A(2*I,2*J-1)-BROWY(M,2*IN-1)
        A(2*I,2*J)=A(2*I,2*J)-BROWY(M,2*IN)
        BTX=BTX-AROWX(M,2*IN-1)*U(J)-AROWX(M,2*IN)*V(J)
        BTY=BTY-AROWY(M,2*IN-1)*U(J)-AROWY(M,2*IN)*V(J)
      END IF

!
!   IF STRESSES ARE PRESCRIBED OVER THE ELEMENT, STORE THE A MATRIX
!   COEFFICIENTS, EXCEPT AT A CORNER NODE WHERE PRESCRIBED DISPLACEMENTS
!   HAVE BEEN IMPOSED.
      IF (IBCE(M) == 2) THEN
        BTX=BTX+(BROWX(M,2*IN-1)*TMX(M,IN)+BROWX(M,2*IN)*TMY(M,IN))
        &      /ESTORE
        BTY=BTY+(BROWY(M,2*IN-1)*TMX(M,IN)+BROWY(M,2*IN)*TMY(M,IN))
        &      /ESTORE
      IF (IBCN(J) == 2) THEN
        A(2*I-1,2*J-1)=A(2*I-1,2*J-1)+AROWX(M,2*IN-1)
        A(2*I-1,2*J)=A(2*I-1,2*J)+AROWX(M,2*IN)
        A(2*I,2*J-1)=A(2*I,2*J-1)+AROWY(M,2*IN-1)
        A(2*I,2*J)=A(2*I,2*J)+AROWY(M,2*IN)
      END IF
      IF (IBCN(J) == 1) THEN
        BTX=BTX-AROWX(M,2*IN-1)*U(J)-AROWX(M,2*IN)*V(J)
        BTY=BTY-AROWY(M,2*IN-1)*U(J)-AROWY(M,2*IN)*V(J)
      END IF
    END IF
  END DO Each element node in turn
END DO Each element in turn
!

```



```

!   STORE THE B*T COEFFICIENTS AS EXTENSIONS OF MATRIX A.
      A(2*I-1,JMAX)=BTX
      A(2*I,JMAX)=BTY
      END DO Take each node in turn as P
!
!   APPLY NODAL POINT DISPLACEMENT CONSTRAINTS.
      IF(NBCPC > 0) THEN
        Each point constraint in turn: DO IBCPC=1,NBCPC
          IROW=2*(NODEPC(IBCPC)-1)+IDIRPC(IBCPC)
          Each column in turn: DO J=1,JMAX
            A(IROW,J)=0.
          END DO Each column in turn
          A(IROW,IROW)=1.
        END DO Each point constraint in turn
      END IF
!
      RETURN
END SUBROUTINE FRMTRX

```

At the beginning of the subprogram the current rows of array A and all the coefficients of the $AROW$ and $BROW$ arrays are set to zero in anticipation of the assembly process. Then for each boundary element in turn subprogram $INTKER$ is called to carry out the integrations. Before any boundary conditions are applied, the coefficients of the 2×2 submatrix at the diagonal of the $[A]$ matrix, which include the free terms, are computed as explained in connection with Equation 5.80. For each of the four coefficients, the sum of corresponding values along the current row of matrix $[A]$ must be zero, which allows the value at the diagonal to be equated to the sum of the other values with its sign reversed.

Off-diagonal coefficients for any element end node have two contributions, from the two elements which share it, stored in the relevant locations in $AROWX$ and $AROWY$. The coefficients on the current pair of rows of the product of matrix $[B^*]$ with the vector of known variables are to be accumulated in BTX and BTY . These are set to zero initially in anticipation of the element contributions to be added to them. If the boundary condition at point P is of the displacement type, the products of the coefficients at the diagonal of $[A]$ with the prescribed displacements there are known and must be moved from the left hand side to the right hand side of the equation, and stored in BTX and BTY with their signs changed.

The boundary conditions applied to each of the elements in turn are now used to determine how the terms stored in the $AROW$ and $BROW$ arrays must be added to $[A]$ and $[B]$. It is perhaps helpful to consider the relevant fragments of the equations represented by the current rows of the $[A]$ and $[B]$ matrices, the fragments being for the $(2 \cdot IN - 1)^{th}$ and the $(2 \cdot IN)^{th}$ equations for the element, corresponding to the IN^{th} node of the element numbered M , as a result of the integration over the element

$$\begin{aligned} & \dots + \text{AROWX}(M, 2 \cdot \text{IN} - 1) \times u_j + \text{AROWX}(M, 2 \cdot \text{IN}) \times v_j \dots = \\ & \dots + \text{BROWX}(M, 2 \cdot \text{IN} - 1) \times \{t_x\}_j + \text{BROWX}(M, 2 \cdot \text{IN}) \times \{t_y\}_j + \dots \end{aligned} \quad (6.1)$$

$$\begin{aligned} & \dots + \text{AROWY}(M, 2 \cdot \text{IN} - 1) \times u_j + \text{AROWY}(M, 2 \cdot \text{IN}) \times v_j \dots = \\ & \dots + \text{BROWY}(M, 2 \cdot \text{IN} - 1) \times \{t_x\}_j + \text{BROWY}(M, 2 \cdot \text{IN}) \times \{t_y\}_j + \dots \end{aligned} \quad (6.2)$$

where j is the number of the IN^{th} node of the element ($J = \text{NODE}(M, \text{IN})$ in programming terms). These parts of a pair of equations are presented in a mixture of program and physical variables in an attempt to clarify what is going on.

Tractions known over the element

If the tractions over the element are known and the displacements unknown then the above products involving BROWX and BROWY, which are known quantities, are added to the right hand side vector whose coefficients for the current equation are BTX and BTY. Note that because the traction values stored in arrays TMX and TMY have not already been scaled, they are divided by the value of Young's modulus stored in ESTORE. With the displacements at the particular node unknown, on the left hand side of the equations the computed AROWX and AROWY values are simply added to the relevant $[A]$ matrix coefficients stored in $A(I, J)$.

Join American online LIGS University!

Interactive Online programs
BBA, MBA, MSc, DBA and PhD

Special Christmas offer:

- ▶ enroll **by December 18th, 2014**
- ▶ **start studying and paying only in 2015**
- ▶ **save up to \$ 1,200** on the tuition!
- ▶ Interactive Online education
- ▶ visit ligsuniversity.com to find out more!

Note: LIGS University is not accredited by any nationally recognized accrediting agency listed by the US Secretary of Education. More info [here](http://ligsuniversity.com).



If the displacements at the particular node are known, which implies that the node is an end node at a point where the boundary condition changes from displacement type to traction type, then the products of the AROWX and AROWY terms with known displacements in Equations 6.1 and 6.2, must be taken from the left hand side of the equation to the right hand side and subtracted from BTX and BTY.

Displacements known over the element

If the displacements over the element (and hence at all three nodes) are known and the tractions are unknown then the terms in Equations 6.1 and 6.2 have to be moved to the opposite sides

$$\begin{aligned} & \dots - \text{BROWX}(M, 2*IN-1) \times \{t_x\}_j - \text{BROWX}(M, 2*IN) \times \{t_y\}_j + \dots = \\ & \dots - \text{AROWX}(M, 2*IN-1) \times u_j - \text{AROWX}(M, 2*IN) \times v_j \dots \end{aligned} \quad (6.3)$$

$$\begin{aligned} & \dots - \text{BROWY}(M, 2*IN-1) \times \{t_x\}_j - \text{BROWY}(M, 2*IN) \times \{t_y\}_j + \dots = \\ & \dots - \text{AROWY}(M, 2*IN-1) \times u_j - \text{AROWY}(M, 2*IN) \times v_j \dots \end{aligned} \quad (6.4)$$

The products of the AROWX and AROWY terms with the known displacements, which are known quantities, are subtracted from the right hand side vector whose coefficients for the current equations are BTX and BTY. On the left hand sides of the equations the computed BROWX and BROWY values are simply subtracted from the relevant $[A]$ matrix coefficients stored in $A(I, J)$.

Once assembly of the linear equations for the current point P is complete, the right hand side vector coefficients accumulated in BTX and BTY are stored in the last column of extended matrix $[A]$. The assembly process is repeated for all nodes as force point P .

Finally, any nodal point constraints are applied. As already discussed in Section 5.4, these can only be used in positions where they would not give rise to concentrated forces at the points concerned. In other words they are used to anchor a domain which is in force equilibrium but would otherwise be free to move as rigid body. Point constraints are applied as follows. The relevant equation number to be modified is the $[2(i-1) + k]^{\text{th}}$, where i is the node number and k is the direction number of the constraint (1 for x and 2 for y). All the coefficients in that equation, including the one in the right hand side vector, are set to zero. The coefficient on the diagonal of matrix $[A]$ is then set to one. These changes have the effect of defining the relevant displacement at the node to be zero.

6.1.8 Subprogram for integrating kernel function products over an element

As indicated in the previous section, the purpose of subprogram INTKER is to compute the integrals involving the kernel functions for a particular source point P (node numbered I) and element m , and which are required in Equations 5.78 and 5.79. The global co-ordinates of P are first stored in XP and YP . Then, for each element node in turn (numbered c in the these equations, IN in the subprogram) the node number is stored as J .

```

      SUBROUTINE  INTKER(I,M)
!
!  SUBPROGRAM TO INTEGRATE THE KERNEL PRODUCTS FOR A PARTICULAR FORCE
!  POINT P (INDICATED BY NODE NUMBER I) OVER A PARTICULAR ELEMENT
!  (INDICATED BY ARGUMENT M) BY GAUSSIAN QUADRATURE.
!
      USE SHARED DATA2EQ
!
!  CONSTANT IN DISPLACEMENT KERNELS MULTIPLYING THE LOGARITHMIC TERM.
      CL=(1.+NU)*(3.-NU)/(4.*PI*E)
!
!  COORDINATES OF POINT P.
      XP=XNODE(I)
      YP=YNODE(I)
!
!  SET UP THE ELEMENT NODE LOOP.
      Each element node in turn: DO IN=1,3
      J=NODE(M,IN)
!
!  IF P IS NOT THE CURRENT ELEMENT NODE, USE NORMAL GAUSSIAN QUADRATURE.
      IF(I /= J) THEN
      Each Gauss point in turn: DO IGAUSS=1,NGAUSS
!
!  EVALUATE JACOBIAN AND UNIT NORMAL VECTOR COMPONENTS, ALSO THE KERNELS
!  AT THE PARTICULAR GAUSS POINT FOR NORMAL QUADRATURE OVER THE WHOLE
!  ELEMENT.
      IT=1
      IC=1
      CALL JACOBI(M, IGAUSS, IT, IC)
      CALL KERNEL(XP, YP, M, IGAUSS, UNGX, UNGY)
!
!  ACCUMULATE THE INTEGRALS.
      SFN=SF(IN, IGAUSS)

```

```

      AROWX (M, 2*IN-1) = AROWX (M, 2*IN-1) + WG (IGAUSS) * AKXX * SFN * JACOB
      AROWX (M, 2*IN) = AROWX (M, 2*IN) + WG (IGAUSS) * AKXY * SFN * JACOB
      AROWY (M, 2*IN-1) = AROWY (M, 2*IN-1) + WG (IGAUSS) * AKYX * SFN * JACOB
      AROWY (M, 2*IN) = AROWY (M, 2*IN) + WG (IGAUSS) * AKYY * SFN * JACOB
      BROWX (M, 2*IN-1) = BROWX (M, 2*IN-1) + WG (IGAUSS) * BKXX * SFN * JACOB
      BROWX (M, 2*IN) = BROWX (M, 2*IN) + WG (IGAUSS) * BKXY * SFN * JACOB
      BROWY (M, 2*IN-1) = BROWY (M, 2*IN-1) + WG (IGAUSS) * BKYX * SFN * JACOB
      BROWY (M, 2*IN) = BROWY (M, 2*IN) + WG (IGAUSS) * BKYY * SFN * JACOB
      END DO Each Gauss point in turn
    END IF

!
! IF P IS THE CURRENT ELEMENT NODE, SOME LOGARITHMIC QUADRATURE
! IS REQUIRED.
      IF (I == J) THEN
!
! P AT THE FIRST NODE OF THE ELEMENT.
      IF (IN == 1) THEN
!
! TERMS INVOLVING NORMAL QUADRATURE.
      Each Gauss point in turn: DO IGAUSS=1, NGAUSS

```



ie business school

#1 EUROPEAN BUSINESS SCHOOL
FINANCIAL TIMES 2013

#gobeyond

MASTER IN MANAGEMENT

Because achieving your dreams is your greatest challenge. IE Business School's Master in Management taught in English, Spanish or bilingually, trains young high performance professionals at the beginning of their career through an innovative and stimulating program that will help them reach their full potential.

- Choose your area of specialization.
- Customize your master through the different options offered.
- Global Immersion Weeks in locations such as Rio de Janeiro, Shanghai or San Francisco.

Because you change, we change with you.

www.ie.edu/master-management | mim.admissions@ie.edu | [f](#) [t](#) [in](#) [You Tube](#) [u](#)

```

      IT=1
      IC=1
      CALL JACOBI (M, IGAUSS, IT, IC)
      CALL KERN2 (M, IN, IGAUSS)
      SFN=SF (IN, IGAUSS)
      BROWX (M, 2*IN-1)=BROWX (M, 2*IN-1) +WG (IGAUSS) *BK2XX*SFN*JACOB
      BROWX (M, 2*IN)=BROWX (M, 2*IN) +WG (IGAUSS) *BK2XY*SFN*JACOB
      BROWY (M, 2*IN-1)=BROWY (M, 2*IN-1) +WG (IGAUSS) *BK2YX*SFN*JACOB
      BROWY (M, 2*IN)=BROWY (M, 2*IN) +WG (IGAUSS) *BK2YY*SFN*JACOB
      END DO Each Gauss point in turn

!
!  TERMS INVOLVING LOGARITHMIC QUADRATURE.
      Each Gauss point in turn: DO IGAUSS=1,NGAUSS
      IT=2
      IC=1
      CALL JACOBI (M, IGAUSS, IT, IC)
      SFN=SFL (IC, IN, IGAUSS)
      DZDE=2.
      BROWX (M, 2*IN-1)=BROWX (M, 2*IN-1) +WGL (IGAUSS) *CL*SFN*JACOB*DZDE
      BROWY (M, 2*IN)=BROWY (M, 2*IN) +WGL (IGAUSS) *CL*SFN*JACOB*DZDE
      END DO Each Gauss point in turn
      END IF

!
!  P AT THE SECOND NODE OF THE ELEMENT.
      IF (IN == 2) THEN

!
!  TERMS INVOLVING NORMAL QUADRATURE.
      Each Gauss point in turn: DO IGAUSS=1,NGAUSS
      IT=1
      IC=1
      CALL JACOBI (M, IGAUSS, IT, IC)
      CALL KERN2 (M, IN, IGAUSS)
      SFN=SF (IN, IGAUSS)
      BROWX (M, 2*IN-1)=BROWX (M, 2*IN-1) +WG (IGAUSS) *BK2XX*SFN*JACOB
      BROWX (M, 2*IN)=BROWX (M, 2*IN) +WG (IGAUSS) *BK2XY*SFN*JACOB
      BROWY (M, 2*IN-1)=BROWY (M, 2*IN-1) +WG (IGAUSS) *BK2YX*SFN*JACOB
      BROWY (M, 2*IN)=BROWY (M, 2*IN) +WG (IGAUSS) *BK2YY*SFN*JACOB
      END DO Each Gauss point in turn

!

```



```

!   TERMS INVOLVING LOGARITHMIC QUADRATURE.
      Each Gauss point in turn: DO IGAUSS=1,NGAUSS
      IT=2
      IC=2
      CALL  JACOBI (M, IGAUSS, IT, IC)
      SFN=SFL(IC, IN, IGAUSS)
      DZDE=1.
      BROWX (M, 2*IN-1)=BROWX (M, 2*IN-1)+WGL (IGAUSS) *CL*SFN*JACOB*DZDE
      BROWY (M, 2*IN)=BROWY (M, 2*IN)+WGL (IGAUSS) *CL*SFN*JACOB*DZDE
      IC=3
      CALL  JACOBI (M, IGAUSS, IT, IC)
      SFN=SFL(IC, IN, IGAUSS)
      DZDE=1.
      BROWX (M, 2*IN-1)=BROWX (M, 2*IN-1)+WGL (IGAUSS) *CL*SFN*JACOB*DZDE
      BROWY (M, 2*IN)=BROWY (M, 2*IN)+WGL (IGAUSS) *CL*SFN*JACOB*DZDE
      END DO Each Gauss point in turn
    END IF

!
!   P AT THE THIRD NODE OF THE ELEMENT.
      IF(IN == 3) THEN

!
!   TERMS INVOLVING NORMAL QUADRATURE.
      Each Gauss point in turn: DO IGAUSS=1,NGAUSS
      IT=1
      IC=1
      CALL  JACOBI (M, IGAUSS, IT, IC)
      CALL  KERN2 (M, IN, IGAUSS)
      SFN=SF(IN, IGAUSS)
      BROWX (M, 2*IN-1)=BROWX (M, 2*IN-1)+WG (IGAUSS) *BK2XX*SFN*JACOB
      BROWX (M, 2*IN)=BROWX (M, 2*IN)+WG (IGAUSS) *BK2XY*SFN*JACOB
      BROWY (M, 2*IN-1)=BROWY (M, 2*IN-1)+WG (IGAUSS) *BK2YX*SFN*JACOB
      BROWY (M, 2*IN)=BROWY (M, 2*IN)+WG (IGAUSS) *BK2YY*SFN*JACOB
      END DO Each Gauss point in turn

!
!   TERMS INVOLVING LOGARITHMIC QUADRATURE.
      Each Gauss point in turn: DO IGAUSS=1,NGAUSS
      IT=2
      IC=4
      CALL  JACOBI (M, IGAUSS, IT, IC)
      SFN=SFL(IC, IN, IGAUSS)

```



```

DZDE=2.
BROWX (M, 2*IN-1) =BROWX (M, 2*IN-1) +WGL ( IGAUSS) *CL*SFN*JACOB*DZDE
BROWY (M, 2*IN) =BROWY (M, 2*IN) +WGL ( IGAUSS) *CL*SFN*JACOB*DZDE
END DO Each Gauss point in turn
END IF
END IF
END DO Each element node in turn
!
RETURN
END SUBROUTINE INTKER

```

SMS from your computer

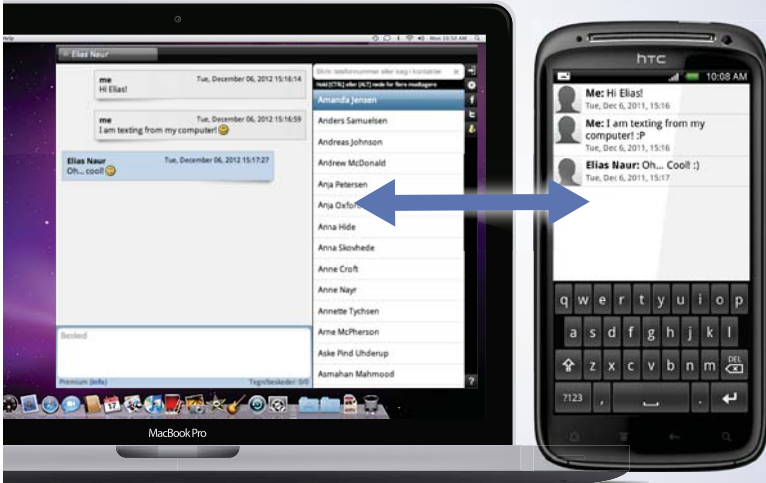
...Sync'd with your Android phone & number

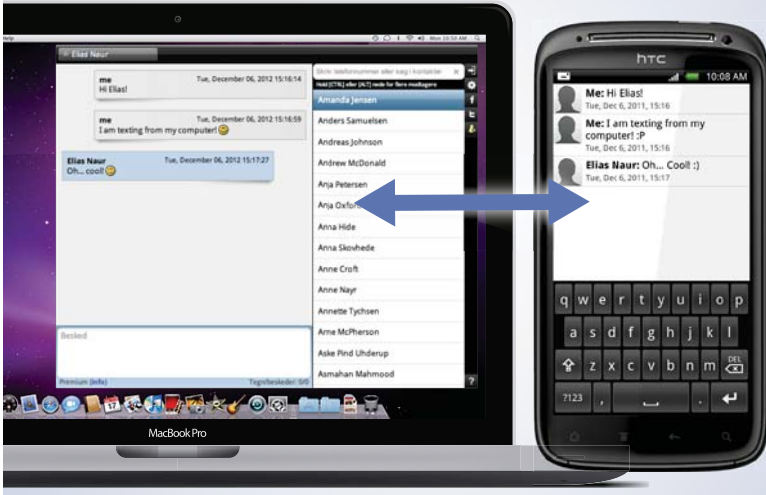
FREE
30 days trial!!

Go to

BrowserTexting.com

and start texting from
your computer!




BrowserTexting

P not at the current node of the element containing Q

If P and Q are not in the same element, or P is not at the current node of the element containing Q (that is, $I \neq J$) then normal Gaussian quadrature (Appendix A) over the whole element is carried out to evaluate the integrals. The required shape functions at the Gauss points have already been computed in subprogram SHAPE and stored in array SF. The Jacobians at the Gauss points are found in subprogram JACOBI, and the kernel functions in subprogram KERNEL. The last of these subprograms stores the kernel functions in variables AKXX, AKXY, AKYX, AKYY and BKXX, BKXY, BKYX, BKYY, respectively. These correspond to T_{xx} , T_{xy} , T_{yx} , T_{yy} and U_{xx} , U_{xy} , U_{yx} , U_{yy} in the equations. For each Gauss point the products of Gaussian weighting factor (WG), kernel functions, shape function (SFN) and Jacobian (JACOB) are then added to either arrays AROWX, AROWY, BROWX or BROWY to complete the numerical integration process.

P at the current node of the element containing Q

If P and Q are in the same element and P is at the current node of the element (that is, $I=J$) then, because of the singular nature of the second kernel function, some logarithmic Gaussian quadrature is required, as described in Section 5.5.2. The first kernel functions do not have to be evaluated because the relevant coefficients at the diagonal of the $[A]$ matrix have already been found indirectly by summing the off-diagonal terms.

If P is the first node of the element the integration process follows Equations 5.82 to 5.96. The integrals of the second kernel function are split into singular and non-singular parts. To the latter are applied Gaussian quadrature in the usual way, the non-singular parts of the second kernel functions being defined in subprogram KERN2 and returned in variables BK2XX, BK2XY, BK2YX and BK2YY. The singular parts are integrated using logarithmic Gaussian quadrature (Appendix A) applied to the whole element, with the origin of the modified intrinsic co-ordinate η being at the first node. The derivative of ξ with respect to η , $|d\xi/d\eta|$, is stored in variable DZDE.

If P is the second node of the element the integration process follows Equations 5.97 to 5.109. The integrals of the second kernel functions are again split into singular and non-singular parts. To the latter are applied Gaussian quadrature in the usual way, the non-singular parts of the second kernel function being defined in subprogram KERN2 and returned in variables BK2XX, BK2XY, BK2YX and BK2YY. The singular parts are integrated using logarithmic Gaussian quadrature applied to the two halves of the element separately, in each case with the origin of the modified intrinsic co-ordinate η at the second node.

If P is the third node of the element the integration process follows Equations 5.110 to 5.121, and is very similar to the case of P at the first node of the element.

6.1.9 Subprogram for computing the Jacobian at a point on an element

Subprogram JACOBI is identical to the subprogram described in Section 4.1.10, with the exception that the USE statement is for module SHARED DATA2EQ rather than SHARED DATA2PQ.

```
      SUBROUTINE  JACOBI (M, IGAUSS, IT, IC)
!
!  SUBPROGRAM TO EVALUATE THE JACOBIAN AND THE COMPONENTS OF THE UNIT
!  NORMAL VECTOR AT A PARTICULAR GAUSS POINT.
!  M INDICATES THE ELEMENT NUMBER.
!  IGAUSS INDICATES THE GAUSS POINT NUMBER.
!  IT INDICATES THE TYPE OF QUADRATURE.
!    IT=1 - NORMAL GAUSSIAN QUADRATURE.
!    IT=2 - LOGARITHMIC GAUSSIAN QUADRATURE.
!  IC INDICATES THE CASE NUMBER FOR LOGARITHMIC GAUSSIAN QUADRATURE,
!  AS DEFINED IN SUBROUTINE SHAPE.
!
      USE SHARED DATA2EQ
!
!  CALCULATE THE DERIVATIVES OF THE GLOBAL COORDINATES WITH RESPECT TO
!  THE LOCAL INTRINSIC COORDINATE.
      DX=0.
      DY=0.
      Each element node in turn: DO IN=1,3
      IF (IT == 1) SFND=SD(IN, IGAUSS)
      IF (IT == 2) SFND=SDL(IC, IN, IGAUSS)
      J=NODE (M, IN)
      DX=DX+SFND*XNODE (J)
      DY=DY+SFND*YNODE (J)
      END DO Each element node in turn
!
!  COMPONENTS OF THE LOCAL OUTWARD NORMAL VECTOR AT THE GAUSS POINT.
      UNGX=DY
      UNGY=-DX
!
!  JACOBIAN OF THE COORDINATE TRANSFORMATION.
      JACOB=SQRT (UNGX**2+UNGY**2)
!
!  SCALE THE VECTOR COMPONENTS TO GIVE THE UNIT NORMAL VECTOR.
      UNGX=UNGX/JACOB
      UNGY=UNGY/JACOB
!
      RETURN
      END SUBROUTINE JACOBI
```

By products of the calculation for a Jacobian (JACOB) are the components of the outward normal vector to the boundary at the chosen Gauss point. These are converted to components of the unit normal and stored in variables UNGX and UNGY.


6.1.10 Subprogram for computing the kernel functions when P is not at the current node of the element containing Q

Subprogram KERNEL computes the kernel functions at a particular Gauss point Q . The co-ordinates XQ and YQ of the point are first found using Equations 5.69 and 5.70, followed by the components r_x and r_y (RX and RY) of the radius vector of length r (R) joining P and Q . The rate of change of radius r with the normal n to the boundary (DRDN) is found as the cosine of the angle between them, or the scalar product of the two unit vectors. The first kernel functions are computed using Equations 5.33, 5.36, 5.38, 5.40, and the second kernels from Equations 5.26 to 5.29.

```

SUBROUTINE  KERNEL (XP, YP, M, IGAUSS, UNX, UNY)
!
!  SUBPROGRAM TO EVALUATE THE KERNELS WHEN P IS NOT THE CURRENT
!  ELEMENT NODE.
!  XP, YP INDICATE THE GLOBAL COORDINATES OF POINT P.
!  M INDICATES THE ELEMENT NUMBER.
!  IGAUSS INDICATES THE NUMBER OF THE GAUSS POINT, Q.

```



The Wake


the only emission we want to leave behind

Low-speed Engines Medium-speed Engines Turbochargers Propellers Propulsion Packages PrimeServ

The design of eco-friendly marine power and propulsion solutions is crucial for MAN Diesel & Turbo. Power competencies are offered with the world's largest engine programme – having outputs spanning from 450 to 87,220 kW per engine. Get up front! Find out more at www.mandieselturbo.com

Engineering the Future – since 1758.

MAN Diesel & Turbo




```

!
      USE SHARED DATA2EQ
!
! COORDINATES OF GAUSS POINT Q.
      XQ=0.
      YQ=0.
      Each element node in turn: DO IN=1,3
      SFN=SF(IN, IGAUSS)
      J=NODE(M, IN)
      XQ=XQ+SFN*XNODE(J)
      YQ=YQ+SFN*YNODE(J)
      END DO Each element node in turn
!
! COMPONENTS AND MAGNITUDE OF THE RADIUS VECTOR FROM P TO Q.
      RX=XQ-XP
      RY=YQ-YP
      RSQ=RX**2+RY**2
      R=SQRT(RSQ)
      RX=RX/R
      RY=RY/R
!
! RATE OF CHANGE OF R WITH THE NORMAL TO THE BOUNDARY AT Q.
      DRDN=RX*UNX+RY*UNY
!
! PARAMETERS IN THE KERNEL FUNCTIONS.
      C1=-1./(4.*PI*R)
      C2=1.-NU
      C3=2.*(1.+NU)
      C4=(1.+NU)**2/(4.*PI*E)
      C5=(3.-NU)*ALOG(1./R)/(1.+NU)
!
! EVALUATE THE KERNELS.
      AKXX=C1*(C2+C3*RX*RX)*DRDN
      TERM1=C3*RX*RY*DRDN
      TERM2=C2*(RX*UNY-RY*UNX)
      AKXY=C1*(TERM1-TERM2)
      AKYX=C1*(TERM1+TERM2)
      AKYY=C1*(C2+C3*RY*RY)*DRDN
      BKXX=C4*(C5+RX*RX)
      BKXY=C4*RX*RY

```

```

BKXX=BKYY
BKYY=C4*(C5+RY*RY)
!
RETURN
END SUBROUTINE KERNEL

```

6.1.11 Subprogram for computing the second kernel function when P is at the current node of the element containing Q

Subprogram KERN2 computes, for a particular Gauss point Q , the non-singular components of the second kernel functions when force point is at the current node of the element containing Q . In other words, all but the $\ln\left(\frac{1}{\eta}\right)$, $\ln\left(\frac{1}{\eta_1}\right)$, and $\ln\left(\frac{1}{\eta_2}\right)$ terms in Equations 5.92 to 5.96, 5.108 and 5.109, 5.120 and 5.121. These are stored in BK2XX, BK2XY, BK2YX and BK2YY in the subprogram. The value of function $R(\xi)$ (RFN in the subprogram) is defined by either Equation 5.91, 5.106 or 5.119, according to whether P is at the first, second or third node of the element. The relevant value of the intrinsic co-ordinate ξ (variable ZETA in the program) is obtained from array ZG for normal Gaussian quadrature.

```

SUBROUTINE KERN2 (M, IN, IGAUSS)
!
! SUBPROGRAM TO EVALUATE THE NON-SINGULAR LOGARITHMIC TERM IN THE
! SECOND KERNEL WHEN P IS THE CURRENT ELEMENT NODE.
! M INDICATES THE ELEMENT NUMBER.
! IN INDICATES THE NUMBER OF THE ELEMENT NODE FORMING P.
! IGAUSS INDICATES THE GAUSS POINT NUMBER.
!
USE SHARED DATA2EQ
!
! COORDINATES OF GAUSS POINT Q.
XQ=0.
YQ=0.
Each element node in turn: DO IC=1,3
SFN=SF(IC, IGAUSS)
J=NODE(M, IC)
XQ=XQ+SFN*XNODE(J)
YQ=YQ+SFN*YNODE(J)
END DO Each element node in turn
!
! ELEMENT NODE NUMBERS.
I1=NODE(M, 1)
I2=NODE(M, 2)
I3=NODE(M, 3)
!

```

```
!  EVALUATE THE INTRINSIC COORDINATE.
      ZETA=ZG(IGAUSS)
!
!  P AT THE FIRST NODE OF THE ELEMENT.
      IF(IN == 1) THEN
        XCOMP=(ZETA-2.)*XNODE(I1)+2.*(1.-ZETA)*XNODE(I2)+ZETA*XNODE(I3)
        YCOMP=(ZETA-2.)*YNODE(I1)+2.*(1.-ZETA)*YNODE(I2)+ZETA*YNODE(I3)
        RFN=SQRT(XCOMP**2+YCOMP**2)
        I=I1
      END IF
!
!  P AT THE SECOND NODE OF THE ELEMENT.
      IF(IN == 2) THEN
        XCOMP=-0.5*(ZETA-1.)*XNODE(I1)+ZETA*XNODE(I2)
        &      -0.5*(ZETA+1.)*XNODE(I3)
        YCOMP=-0.5*(ZETA-1.)*YNODE(I1)+ZETA*YNODE(I2)
        &      -0.5*(ZETA+1.)*YNODE(I3)
        RFN=SQRT(XCOMP**2+YCOMP**2)
        I=I2
      END IF
```

TURN TO THE EXPERTS FOR **SUBSCRIPTION** CONSULTANCY

Subscribe is one of the leading companies in Europe when it comes to innovation and business development within subscription businesses.

We innovate new subscription business models or improve existing ones. We do business reviews of existing subscription businesses and we develop acquisition and retention strategies.

Learn more at [linkedin.com/company/subscribe](https://www.linkedin.com/company/subscribe) or contact
Managing Director Morten Suhr Hansen at mha@subscribe.dk

SUBSCR✓**BE** - to the future



```

!
!  P AT THE THIRD NODE OF THE ELEMENT.
      IF (IN == 3) THEN
          XCOMP=-ZETA*XNODE(I1)+2.*(1.+ZETA)*XNODE(I2)-(ZETA+2.)*XNODE(I3)
          YCOMP=-ZETA*YNODE(I1)+2.*(1.+ZETA)*YNODE(I2)-(ZETA+2.)*YNODE(I3)
          RFN=SQRT(XCOMP**2+YCOMP**2)
          I=I3
      END IF
!
!  COMPONENTS AND MAGNITUDE OF THE RADIUS VECTOR FROM P TO Q.
      XP=XNODE(I)
      YP=YNODE(I)
      RX=XQ-XP
      RY=YQ-YP
      RSQ=RX**2+RY**2
      R=SQRT(RSQ)
      RX=RX/R
      RY=RY/R
!
!  PARAMETERS IN THE KERNEL FUNCTIONS.
      C4=(1.+NU)**2/(4.*PI*E)
      C5=(3.-NU)*ALOG(1./RFN)/(1.+NU)
!
!  EVALUATE THE KERNELS.
      BK2XX=C4*(C5+RX*RX)
      BK2XY=C4*RX*RY
      BK2YX=BK2XY
      BK2YY=C4*(C5+RY*RY)
!
      RETURN
END SUBROUTINE KERN2

```

6.1.12 Results output subprogram

Subprogram OUTPUT organises and writes out the primary results of the computation. Firstly, displacements and tractions are stored in their proper arrays.

```

SUBROUTINE OUTPUT
!
!  SUBPROGRAM TO WRITE OUT THE NODAL POINT VALUES OF DISPLACEMENTS
!  AND ELEMENT STRESSES AND COMPUTE THE FORCES ON THE BOUNDARY SEGMENTS.
!

```



```

        USE SHARED DATA2EQ
!
!  ARRANGE FOR U, V AND TX, TY TO CONTAIN THE NODAL DISPLACEMENTS
!  AND TRACTIONS, RESPECTIVELY.
      Each node in turn: DO I=1,NNP
        IF(IBC(N(I)) == 1) THEN
          TX(I)=UV(2*I-1)
          TY(I)=UV(2*I)
        END IF
        IF(IBC(N(I)) == 2) THEN
          U(I)=UV(2*I-1)
          V(I)=UV(2*I)
        END IF
      END DO Each node in turn
!
!  HEADING FOR OUTPUT OF NODAL DISPLACEMENTS AND TRACTIONS.
      WRITE(6,61)
61  FORMAT(/ "NODAL POINT DISPLACEMENTS AND TRACTIONS" //
& "      I          U          V          TX          TY")
      E=ESTORE
      Each node in turn: DO I=1,NNP
!
!  REMOVE THE SCALING.
        U(I)=U(I)*MAXL
        V(I)=V(I)*MAXL
        TX(I)=TX(I)*ESTORE
        TY(I)=TY(I)*ESTORE
!
!  OUTPUT THE NODAL VALUES OF DISPLACEMENTS AND TRACTIONS.
        WRITE(6,62) I,U(I),V(I),TX(I),TY(I)
62  FORMAT(I4,4E15.6)
      END DO Each node in turn
!
!  HEADING FOR OUTPUT OF ELEMENT STRESSES.
      WRITE(6,63)
63  FORMAT(/ "STRESSES AT THE NODES OF THE ELEMENTS" //
& "      M IN          SIGNN          SIGSS          SIGSN",
& "          SIGE")
!

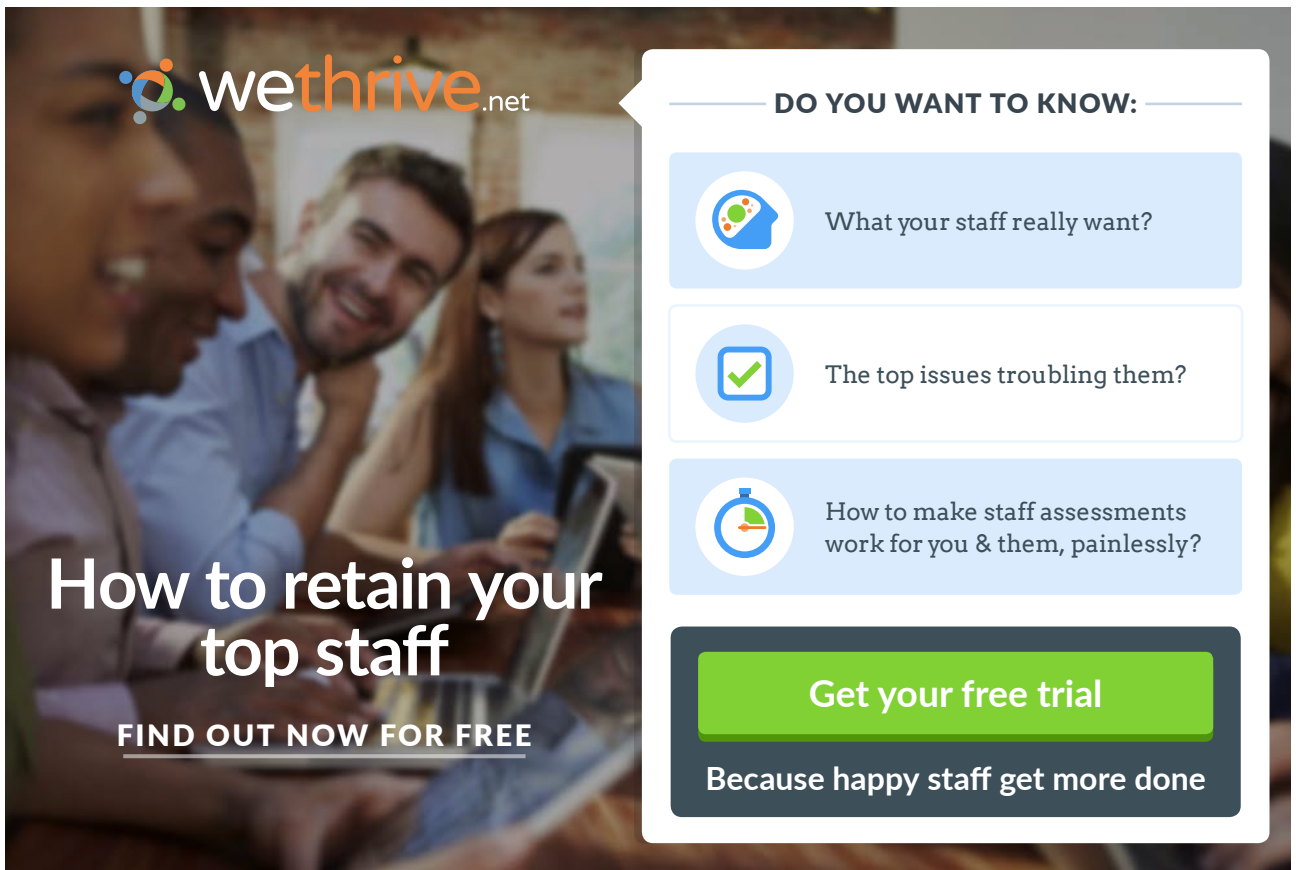
```

```

!  NORMAL AND SHEAR STRESSES AT THE NODES OF THE ELEMENTS.

      Each element in turn: DO M=1,NEL
      Each element node in turn: DO IN=1,3
      IF (IBCE(M) /= 2) THEN
        J=NODE(M,IN)
        TMX(M,IN)=TX(J)
        TMY(M,IN)=TY(J)
        SIGNN(M,IN)=TMX(M,IN)*UNMX(M,IN)+TMY(M,IN)*UNMY(M,IN)
        SIGSN(M,IN)=-TMX(M,IN)*UNMY(M,IN)+TMY(M,IN)*UNMX(M,IN)
      END IF
!
!  DIRECT STRESS ALONG THE BOUNDARY SURFACE.
      IC1=NODE(M,1)
      IC2=NODE(M,2)
      IC3=NODE(M,3)
      IGAUSS=IN+10
      DUDZ=SD(1,IGAUSS)*U(IC1)+SD(2,IGAUSS)*U(IC2)+SD(3,IGAUSS)*U(IC3)
      DVDZ=SD(1,IGAUSS)*V(IC1)+SD(2,IGAUSS)*V(IC2)+SD(3,IGAUSS)*V(IC3)
      IT=1
      IC=1

```



wethrive.net

How to retain your top staff

FIND OUT NOW FOR FREE

DO YOU WANT TO KNOW:

- What your staff really want?
- The top issues troubling them?
- How to make staff assessments work for you & them, painlessly?

Get your free trial

Because happy staff get more done

```

        CALL  JACOBI (M,10+IN,IT,IC)
        ESS=(-DUDZ*UNMY (M,IN)+DVDZ*UNMX (M,IN)) / (JACOB*MAXL)
        SIGSS (M,IN)=E*ESS+NU*SIGNN (M,IN)
!
!  VON MISES EQUIVALENT STRESS.
        SIGE (M,IN)=SQRT (SIGNN (M,IN)**2+SIGSS (M,IN)**2
&          -SIGNN (M,IN)*SIGSS (M,IN)+3.*SIGSN (M,IN)**2)
!
!  OUTPUT THE STRESSES AT THE NODES OF THE ELEMENTS.
        WRITE (6,64) M,IN,SIGNN (M,IN),SIGSS (M,IN),SIGSN (M,IN),
&          SIGE (M,IN)
64  FORMAT (2I4,4E15.6)
        END DO Each element node in turn
        END DO Each element in turn
!
!  COMPUTE THE FORCES ON THE BOUNDARY SEGMENTS.
        Each segment in turn: DO ISEG=1,NSEGTOT
        FXSEG (ISEG)=0.
        FYSEG (ISEG)=0.
        END DO Each segment in turn
        Each element in turn: DO M=1,NEL
        ISEG=ISEGELEM (M)
!
!  APPLY GAUSSIAN QUADRATURE.
        FXELEM=0.
        FYELEM=0.
        Each element node in turn: DO IN=1,3
        Each Gauss point in turn: DO IGAUSS=1,NGAUSS
        SFN=SF (IN,IGAUSS)
        IT=1
        IC=1
        CALL  JACOBI (M,IGAUSS,IT,IC)
        FXELEM=FXELEM+WG (IGAUSS)*SFN*JACOB*TMX (M,IN)*MAXL
        FYELEM=FYELEM+WG (IGAUSS)*SFN*JACOB*TMY (M,IN)*MAXL
        END DO Each Gauss point in turn
        END DO Each element node in turn
!
!  ACCUMULATE THE FORCES ON THE SEGMENT.
        FXSEG (ISEG)=FXSEG (ISEG)+FXELEM

```

```

        FYSEG(ISEG)=FYSEG(ISEG)+FYELEM
    END DO Each element in turn
!
!   OUTPUT THE SEGMENT FORCE RESULTS.
        WRITE(6,65) (ISEG,FXSEG(ISEG),FYSEG(ISEG),ISEG=1,NSEGTOT)
65    FORMAT(/ "FORCES ACTING ON THE BOUNDARY SEGMENTS"
&          // "SEGMENT      FX              FY" / (I5,2E14.5))
!
        RETURN
    END SUBROUTINE OUTPUT

```

If the boundary condition at a particular point is of the first type (prescribed displacements) the solution array UV contains the computed tractions, which are stored in arrays TX and TY. If the boundary condition is of the second type then displacements were the unknowns and are now stored in arrays U and V. Then all displacements and tractions have the scaling removed, and nodal point values of displacements and tractions are written out.

Stresses at each of the three nodes of each element are then computed, unless they were given as input data. In many cases this will provide duplicate information for end nodes of adjacent elements, but not if such nodes are at corners of the domain where tractions are discontinuous. Normal and shear stresses are first found using Equations 5.60 and 5.61, followed by the direct stresses along the boundary from Equations 5.62 and 5.63. The von Mises equivalent stress is found from Equation 5.64, and all four stresses are written out.

In many problems of practical engineering interest, it is useful to have the global co-ordinate components of the forces applied to each of the boundary segments. These can be found by integrating the tractions element by element (forces stored in FXELEM and FYELEM), summing to give total segment forces, FXSEG and FYSEG, which are then written out.

6.1.13 Subprogram for solving the linear equations

Both the Gaussian elimination algorithm and subprogram ELIMIN for solving the linear equations are described in Appendix B. The matrix $[A^*]$ extended to include the right hand side vector $[b]$ (in Equation 5.57) is entered in array A, and the solutions returned in array X (UV in the calling main program).

6.2 Some Test Problems for BEM2EQ

Before attempting to solve real problems for which the solutions are not known, it is essential to test any computer program as extensively as possible on problems for which exact analytical solutions are available for comparison. Test problems serve not only to verify that the program is working correctly, but also to examine the level of discretisation (the fineness of the boundary element mesh in this case) necessary to obtain accurate results.

6.2.1 Simple tension

One of the most basic test problems available is simple tension. Figure 6.1 shows a thin strip of material which has dimensions of 4 units by 2 units. Both the left and right hand edges of the strip (AD and BC) are subject to uniform normal tensile stresses of magnitude 3 units. The top and bottom edges (DC and AB) are free of applied stresses. Young's modulus for the material of the strip is 2000 units, and its Poisson's ratio is 0.3. Under plane stress conditions, with a uniaxial stress of $\sigma_{xx} = 3$ the direct strain in the x direction is $e_{xx} = \frac{\sigma_{xx}}{E} = 0.0015$ and in the y direction is $e_{yy} = -\nu e_{xx} = -0.00045$. Hence the length of the strip increases by $0.0015 \times 4 = 0.006$ and the width reduces by $0.00045 \times 2 = 0.0009$. The total forces acting on the left hand and right hand edges are both 6 (stress times edge length, the strip thickness being treated as unity).

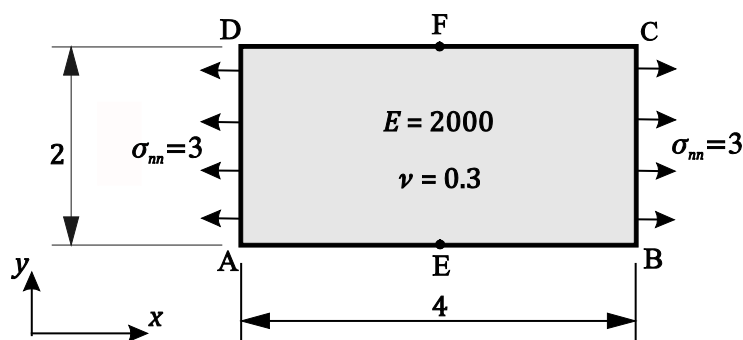


Figure 6.1 Thin strip in simple tension

Struggling to get interviews?

Professional CV consulting & writing assistance from leading job experts in the UK.

Visit site



Take a short-cut to your next job!
Improve your interview success rate by 70%.



TheCVagency
Visit theagency.co.uk for more info.



Click on the ad to read more

Although the strip is in force equilibrium, the problem cannot yet be solved using boundary elements because the boundary conditions are given purely in terms of stresses, with no displacements prescribed. This difficulty can be overcome by applying suitable point constraints. The constraints adopted here are to fix the midpoint E of side AB in both the x and y directions, while constraining midpoint F of side DC not to move in the x direction (but to remain free to move in the y direction). The point constraints prevent rigid body movement of the strip as a whole, without giving rise to any point forces at either point E or point F.

To obtain a quadratic boundary element solution to the problem, the boundary has first to be divided into segments. Four segments along the four edges of the domain are appropriate, because they are straight lines between corners and the boundary condition over each one is uniform. The ends of the four segments are the corner points labelled A, B, C and D (in that order) in Figure 6.1. The choice of starting point is arbitrary, but then the direction of numbering must keep the domain to the left. The data file DATA to solve the problem using one element (three nodes) per side of the domain is as follows

SIMPLE TENSION	
STRESS	(Plane stress selected)
2000. 0.3	(Young's modulus and Poisson's ratio)
1	(Number of boundaries)
4	(Number of segments on the boundary)
0. 0. 4. 0. 4. 2. 0. 2.	(Co-ordinates of the segment ends)
0. 1 1.	(Radii of curvature,
0. 1 1.	number of elements
0. 1 1.	and element length ratio
0. 1 1.	for each of the 4 segments)
0 2 3	(Two prescribed stresses, three point constraints)
2 3. 0.	(Normal stress = 3 on segment 2)
4 3. 0.	(Normal stress = 3 on segment 4)
2 1	(Node 2 constrained in x direction)
2 2	(Node 2 constrained in y direction)
6 1	(Node 6 constrained in x direction)

The annotations on the right have been added for explanation, and are not part of the data file. Note that the prescribed zero values of stresses (on DC and AB) do not have to be entered explicitly.

The mesh data output file MESHRES is as follows

```

GEOMETRIC DATA FOR THE MESH
    NUMBER OF ELEMENTS = 4
    NUMBER OF NODAL POINTS = 8
    COORDINATES OF THE NODAL POINTS

I      X      Y      I      X      Y
1  0.0000E+00  0.0000E+00  2  0.2000E+01  0.0000E+00
3  0.4000E+01  0.0000E+00  4  0.4000E+01  0.1000E+01
5  0.4000E+01  0.2000E+01  6  0.2000E+01  0.2000E+01
7  0.0000E+00  0.2000E+01  8  0.0000E+00  0.1000E+01

ELEMENT NODE NUMBERS

M  ND1  ND2  ND3      M  ND1  ND2  ND3
1   1   2   3          2   3   4   5
3   5   6   7          4   7   8   1

```

With one element per segment, 4 elements and 8 nodes are created. Both elements and nodes are numbered as for the segments, anticlockwise from the origin, as shown in Figure 6.2. Element numbers are shown circled. Element end nodes are shown as small solid circles, mid-side nodes as open circles.

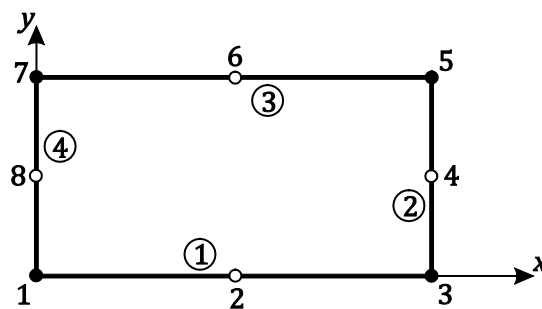


Figure 6.2 Discretisation of simple tension test problem

The RESULTS data file output by the program is

```

QUADRATIC BOUNDARY ELEMENT SOLUTION FOR TWO DIMENSIONAL ELASTIC PROBLEM
SIMPLE TENSION
UNDER PLANE STRESS CONDITIONS
YOUNG'S MODULUS = 0.2000E+04 POISSON'S RATIO = 0.300
PRESCRIBED STRESS BOUNDARY CONDITIONS
SIGNN = 0.3000E+01 SIGSN = 0.0000E+00 ON ELEMENTS 2 TO 2
SIGNN = 0.3000E+01 SIGSN = 0.0000E+00 ON ELEMENTS 4 TO 4
NODE 2 CONSTRAINED WITH U = 0

```

NODE 2 CONSTRAINED WITH $V = 0$
 NODE 6 CONSTRAINED WITH $U = 0$
 NODAL POINT DISPLACEMENTS AND TRACTIONS

I	U	V	TX	TY
1	-0.299965E-02	-0.101365E-05	-0.300000E+01	0.000000E+00
2	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00
3	0.299965E-02	-0.101577E-05	0.300000E+01	-0.799680E-06
4	0.300182E-02	-0.450233E-03	0.300000E+01	0.000000E+00
5	0.299965E-02	-0.899451E-03	0.150000E+01	-0.399840E-06
6	0.000000E+00	-0.900459E-03	0.000000E+00	0.000000E+00
7	-0.299965E-02	-0.899450E-03	-0.300000E+01	0.000000E+00
8	-0.300182E-02	-0.450231E-03	-0.300000E+01	0.000000E+00

STRESSES AT THE NODES OF THE ELEMENTS

M	IN	SIGNN	SIGSS	SIGSN	SIGE
1	1	0.000000E+00	0.299965E+01	0.000000E+00	0.299965E+01
1	2	0.000000E+00	0.299965E+01	0.000000E+00	0.299965E+01
1	3	0.000000E+00	0.299965E+01	0.000000E+00	0.299965E+01
2	1	0.300000E+01	0.156770E-02	0.000000E+00	0.299922E+01
2	2	0.300000E+01	0.156531E-02	0.000000E+00	0.299922E+01
2	3	0.300000E+01	0.156287E-02	0.000000E+00	0.299922E+01
3	1	0.000000E+00	0.299965E+01	0.000000E+00	0.299965E+01
3	2	0.000000E+00	0.299965E+01	0.000000E+00	0.299965E+01
3	3	0.000000E+00	0.299965E+01	0.000000E+00	0.299965E+01
4	1	0.300000E+01	0.156229E-02	0.000000E+00	0.299922E+01
4	2	0.300000E+01	0.156415E-02	0.000000E+00	0.299922E+01
4	3	0.300000E+01	0.156595E-02	0.000000E+00	0.299922E+01

FORCES ACTING ON THE BOUNDARY SEGMENTS

SEGMENT	FX	FY
1	0.00000E+00	0.00000E+00
2	0.60000E+01	-0.53312E-06
3	0.00000E+00	0.00000E+00
4	-0.60000E+01	0.00000E+00

The prescribed stress conditions are a normal value of 3 and shear value of zero over element 2 (nodes 3, 4 and 5) on the right end edge (segment 2), and a normal value of 3 and shear value of zero over element 4 (nodes 7, 8 and 1) on the left hand edge (segment 4). Segments (and elements) 1 and 3 defining the top and bottom edges had no boundary condition prescribed by the DATA file, so zero stress conditions are assumed.

The computed displacements show that at nodes 3, 4 and 5 (on the right hand edge) the displacements in the direction are 0.0029996, 0.0030018 and 0.0029996, respectively, while at nodes 7, 8 and 1 (on the left hand edge) they are -0.0029996, -0.0030018 and -0.0029996. The extensions of the strip between node pairs 3 and 1, 4 and 8 and 5 and 7 are therefore 0.0059992, 0.0060036 and 0.0059992, respectively. These compare very favourably with the exact value of 0.006. Note the slight differences between element end nodes and element midside nodes, which were also experienced in the quadratic element analysis of potential problems. The displacement in the y direction of node 6 (which is vertically above constrained node 2) is -0.00090046, which compares with the exact value of -0.0009.

The value of σ_{xx} (SIGSS) computed at all the nodal points on the top and bottom edges is 2.9996, compared with the exact value of 3. Also, small tensile values of σ_{yy} (SIGSS) of about 0.00156 are computed at all the nodes on the right and left hand side edges, where they should be zero. The forces on the left and right hand edges are exact at 6.0000, which is to be expected because they were prescribed in the boundary conditions.

The advertisement for Gaiteye features a background image of a person running on a path during a sunrise or sunset. The Gaiteye logo, consisting of a stylized yellow 'G' and the brand name, is in the top left. Below it is the tagline 'Challenge the way we run'. The central text reads 'EXPERIENCE THE POWER OF FULL ENGAGEMENT...' followed by a horizontal dotted line. To the left, the text 'RUN FASTER. RUN LONGER.. RUN EASIER...' is displayed. On the right, there is a technical diagram showing a runner's foot with concentric circles and lines indicating movement or pressure. Below this diagram is a yellow button with the text 'READ MORE & PRE-ORDER TODAY' and the website 'WWW.GAITEYE.COM'. A hand cursor icon is positioned over the button.

gaiteye®
Challenge the way we run

**EXPERIENCE THE POWER OF
FULL ENGAGEMENT...**

**RUN FASTER.
RUN LONGER..
RUN EASIER...**

**READ MORE & PRE-ORDER TODAY
WWW.GAITEYE.COM**

The results are highly accurate for this very simple problem. Using just one quadratic element and three nodes on each of the four edges of the boundary gives results which are almost correct to four significant figures. For this very simple problem it is not necessary to refine the mesh because the accuracy of the results is already adequate for most practical engineering purposes.

Using displacement boundary conditions

An alternative approach to the simple tension problem shown in Figure 6.1 is to apply displacement boundary conditions to one end of the strip, say edge AD. Unfortunately, this normally changes the nature of the problem because contraction along AD is prevented. The only exception is if the value of Poisson's ratio is set to zero, rendering the problem one-dimensional.

The RESULTS data file output for this case is

```

QUADRATIC BOUNDARY ELEMENT SOLUTION FOR TWO DIMENSIONAL ELASTIC PROBLEM
SIMPLE TENSION
UNDER PLANE STRESS      CONDITIONS
YOUNG'S MODULUS =      0.2000E+04      POISSON'S RATIO = 0.000
PRESCRIBED DISPLACEMENT BOUNDARY CONDITIONS
U = 0.00000E+00 V = 0.00000E+00 ON ELEMENTS 4 TO 4
PRESCRIBED STRESS BOUNDARY CONDITIONS
SIGNN = 0.3000E+01 SIGSN = 0.0000E+00 ON ELEMENTS 2 TO 2
NODAL POINT DISPLACEMENTS AND TRACTIONS

```

	I	U	V	TX	TY
1	0.000000E+00	0.000000E+00	-0.300536E+01	0.254296E-03	
2	0.300082E-02	0.376321E-07	0.000000E+00	0.000000E+00	
3	0.600076E-02	-0.847336E-06	0.300000E+01	-0.799680E-06	
4	0.600260E-02	-0.344766E-08	0.300000E+01	0.000000E+00	
5	0.600076E-02	0.840583E-06	0.150000E+01	-0.399840E-06	
6	0.300083E-02	-0.407261E-07	0.000000E+00	0.000000E+00	
7	0.000000E+00	0.000000E+00	-0.300536E+01	-0.251318E-03	
8	0.000000E+00	0.000000E+00	-0.299728E+01	-0.181283E-06	

```

STRESSES AT THE NODES OF THE ELEMENTS
M IN      SIGNN      SIGSS      SIGSN      SIGE
1 1 0.000000E+00 0.300127E+01 0.000000E+00 0.300127E+01
1 2 0.000000E+00 0.300038E+01 0.000000E+00 0.300038E+01
1 3 0.000000E+00 0.299949E+01 0.000000E+00 0.299949E+01
2 1 0.300000E+01 0.168764E-02 0.000000E+00 0.299916E+01
2 2 0.300000E+01 0.168792E-02 0.000000E+00 0.299916E+01

```

```

2 3 0.300000E+01 0.168820E-02 0.000000E+00 0.299916E+01
3 1 0.000000E+00 0.299949E+01 0.000000E+00 0.299949E+01
3 2 0.000000E+00 0.300038E+01 0.000000E+00 0.300038E+01
3 3 0.000000E+00 0.300127E+01 0.000000E+00 0.300127E+01
4 1 0.300536E+01 0.000000E+00 0.251318E-03 0.300536E+01
4 2 0.299728E+01 0.000000E+00 0.181283E-06 0.299728E+01
4 3 0.300536E+01 0.000000E+00 -0.254296E-03 0.300536E+01

```

FORCES ACTING ON THE BOUNDARY SEGMENTS

SEGMENT	FX	FY
1	0.00000E+00	0.00000E+00
2	0.60000E+01	-0.53312E-06
3	0.00000E+00	0.00000E+00
4	-0.60000E+01	0.75079E-06

The extensions of the strip computed at nodes 3, 4 and 5 on edge BC are 0.0060007, 0.0060026 and 0.0060007 (exact 0.006), the σ_{xx} stresses at nodes on the top and bottom edges of the strip are 3.0013, 3.0004 and 2.9995 (exact 3), and the force on edge AD is exact at 6.0000. The accuracy remains good when displacements are prescribed rather than stresses.

6.2.2 Pure shear

In order to explore shear loading, Figure 6.3 shows the cross-section of a rectangular block of material which is long in the direction normal to the cross-section. The material concerned again has a Young's modulus of 2000 and a Poisson's ratio of 0.3. The base of the block, AB, is rigidly constrained with prescribed zero values of displacements. The other edges have shear stresses of magnitude 3 applied in such a way as to give rise to a state of pure shear. In other words, on sides BC and DA $\sigma_{sn} = +3$, and on side CD $\sigma_{sn} = -3$, bearing in mind that co-ordinate S is measured along the boundary keeping the domain always to the left.

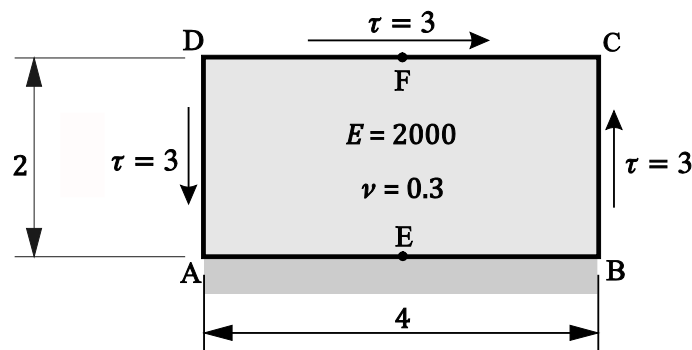


Figure 6.3 Block under pure shear

The shear modulus for the material is

$$G = \frac{E}{2(1 + \nu)} = \frac{2000}{2(1 + 0.3)} = 769.23$$

and the shear strain due to a shear stress of 3 units is $3/769.23 = 0.0039$. With a block height of 2, the displacement of top surface DC is $u = 2 \times 0.0039 = 0.0078$.

The RESULTS data file output for this case is

```
QUADRATIC BOUNDARY ELEMENT SOLUTION FOR TWO DIMENSIONAL ELASTIC PROBLEM
PURE SHEAR
UNDER PLANE STRAIN CONDITIONS
YOUNG'S MODULUS = 0.2000E+04 POISSON'S RATIO = 0.300
PRESCRIBED DISPLACEMENT BOUNDARY CONDITIONS
U = 0.0000E+00 V = 0.0000E+00 ON ELEMENTS 1 TO 1
PRESCRIBED STRESS BOUNDARY CONDITIONS
SIGNN = 0.0000E+00 SIGSN = 0.3000E+01 ON ELEMENTS 2 TO 2
SIGNN = 0.0000E+00 SIGSN = -0.3000E+01 ON ELEMENTS 3 TO 3
SIGNN = 0.0000E+00 SIGSN = 0.3000E+01 ON ELEMENTS 4 TO 4
```

This e-book
is made with
SetaPDF



PDF components for PHP developers

www.setasign.com



NODAL POINT DISPLACEMENTS AND TRACTIONS

I	U	V	TX	TY
1	0.000000E+00	0.000000E+00	-0.300535E+01	-0.184857E-02
2	0.000000E+00	0.000000E+00	-0.299735E+01	0.206969E-06
3	0.000000E+00	0.000000E+00	-0.300535E+01	0.184803E-02
4	0.390155E-02	0.144327E-05	0.000000E+00	0.300000E+01
5	0.780091E-02	0.117580E-06	0.150000E+01	0.150000E+01
6	0.780417E-02	0.283357E-09	0.300000E+01	0.000000E+00
7	0.780091E-02	-0.118911E-06	0.150000E+01	-0.150000E+01
8	0.390155E-02	-0.144314E-05	0.000000E+00	-0.300000E+01

STRESSES AT THE NODES OF THE ELEMENTS

M	IN	SIGNN	SIGSS	SIGSN	SIGE
1	1	0.184857E-02	0.792242E-03	-0.300535E+01	0.520542E+01
1	2	-0.206969E-06	-0.887008E-07	-0.299735E+01	0.519156E+01
1	3	-0.184803E-02	-0.792012E-03	-0.300535E+01	0.520542E+01
2	1	0.000000E+00	0.621710E-02	0.300000E+01	0.519616E+01
2	2	0.000000E+00	0.129209E-03	0.300000E+01	0.519615E+01
2	3	0.000000E+00	-0.595411E-02	0.300000E+01	0.519616E+01
3	1	0.000000E+00	-0.716504E-02	-0.300000E+01	0.519616E+01
3	2	0.000000E+00	0.000000E+00	-0.300000E+01	0.519615E+01
3	3	0.000000E+00	0.716504E-02	-0.300000E+01	0.519616E+01
4	1	0.000000E+00	0.595146E-02	0.300000E+01	0.519616E+01
4	2	0.000000E+00	-0.130672E-03	0.300000E+01	0.519615E+01
4	3	0.000000E+00	-0.621280E-02	0.300000E+01	0.519616E+01

FORCES ACTING ON THE BOUNDARY SEGMENTS

SEGMENT	FX	FY
1	-0.12000E+02	0.19397E-06
2	0.53312E-06	0.60000E+01
3	0.12000E+02	-0.26656E-06
4	0.00000E+00	-0.60000E+01

The computed displacements at nodes 5, 6 and 7 on the top surface DC are 0.0078009, 0.0078042 and 0.0078009 (exact 0.0078). The computed shear stresses at nodes 1, 2 and 3 on the bottom surface AB are -3.00535, -2.9973 and -3.0053 (exact 3), and the total shear force on this surface is exact at -12.000.

It is worth noting that had the shear stresses on sides BC and DA been omitted in Figure 6.3, although the block might appear to be simply sheared, the actual state of stress would have been much more complex. In particular, at corners A and B there is a conflict between finite shear stress along the constrained base and zero shear stress on the side of the block. Equation 1.1 requires shear stresses to be complementary.

6.2.3 Steel component

Figure 6.4 shows a thin steel component with elastic properties of Young's modulus 207 GN/m^2 , Poisson's ratio 0.30. The bottom edge is subject to a uniform normal stress of 100 MN/m^2 , and the top edges to uniform normal stresses of 75 MN/m^2 (giving force equilibrium in the vertical direction). The top edges are also subject to outwardly directed shear stresses of 30 MN/m^2 , giving force equilibrium in the horizontal direction.

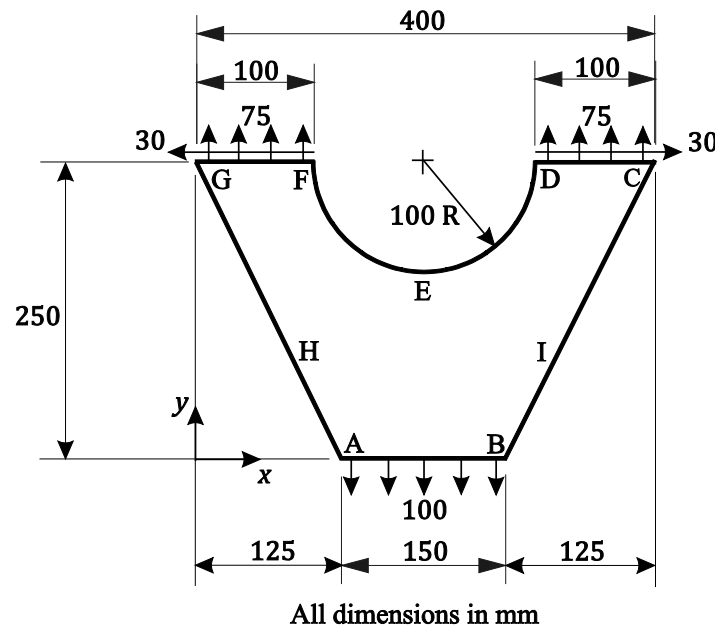


Figure 6.4 A steel component under load

This problem involves somewhat more complex geometry and varying stresses, and is therefore likely to require rather more mesh refinement than the very simple problems treated so far. The object is to find the direct stress in the surface at point E (mid way between points D and F on the semi-circular arc), and the maximum von Mises equivalent stress anywhere on the boundary (and hence anywhere in the component). There is no exact analytical solution for comparison.

The boundary of the domain is conveniently divided into six segments: AB, BC, CD, DEF, FG and GA. For convenience, the same number of boundary elements are used on each segment. Displacement constraints are required to prevent rigid body movement: for example, point A fixed in both directions, and point B constrained not to move in the direction. Table 6.1 shows the effects of varying the number of nodes per segment from 4 to 40. Less than 4 gives poor geometric representation of the semi-circle DEF (with each element then representing an arc of more than 45°).

Elements per segment	σ_{ss} at E MN/m ²	max. σ_e MN/m ²
4	-111.3	152.2 (140.2)
6	-119.9	151.5 (148.7)
10	-123.8	153.7 (153.7)
20	-125.1	152.9 (152.9)
40	-125.3	152.7 (152.7)

Table 6.1 Stress results for varying levels of mesh refinement

In each case point E, which is on the vertical line of symmetry for the problem, is located at the common node of two elements, and the value of σ_{ss} obtained from each element is the same (to at least four significant figures). On the other hand, the maximum value of equivalent stress, which occurs in the regions of both points H and I in Figure 6.4, does not necessarily occur at a node at all. This means that a reasonable degree of mesh refinement is needed to capture a good approximation to the maximum. In each case the maximum is detected at an element end node, and the two values of equivalent stress from the two elements meeting at that node are not necessarily the same. In the table the values in brackets are those from the adjacent element.

There are two criteria to keep in mind when testing for convergence with mesh refinement. Firstly, whether a value of stress changes with increasing number of elements. Secondly, whether stress values at an element end node (in this case the maximum equivalent stress) are the same for both elements meeting at that node. With these in mind, the second criterion in this problem is satisfied at 10 elements per segment or more, whereas the first is only satisfied (to three significant figures) at 20 elements per segment or more. The computed direct stress in the surface at point E is 125 MN/m² (compressive) and the maximum von Mises equivalent stress 153 MN/m² (actually the tensile direct stresses in the surfaces in the regions of points H and I).

6.2.4 A thick-walled cylinder test problem

The next test is designed to demonstrate the ability of the program to solve problems with curved boundaries and those with more than one boundary. Figure 6.5 shows the cross-section of a long thick-walled circular cylinder with internal and external radii of $r_1 = 4$ and $r_2 = 8$. The cylinder is subject to an internal pressure of $p = 3$ and no pressure on the outside. The Young's modulus and Poisson's ratio of the material of the cylinder are 2000 and 0.3, respectively. Cylinders are widely used in engineering practice to contain fluids under pressure: from pipes for relatively low pressures to thick-walled vessels for high-pressure chemical reactors.

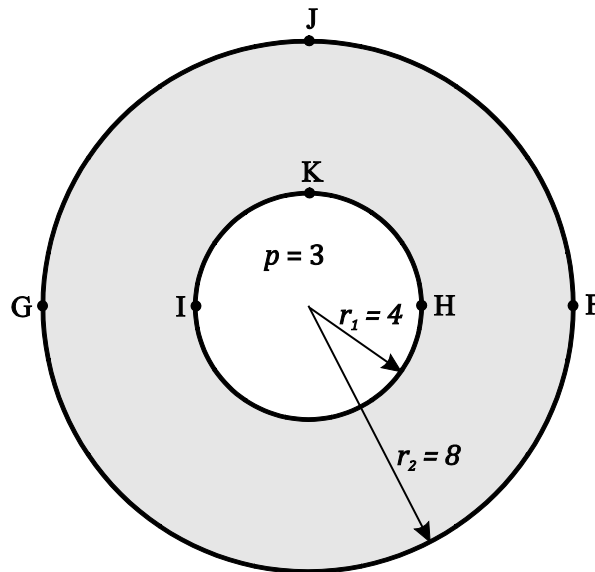


Figure 6.5 Thick-walled cylinder problem

The analytical solution to the thick-walled cylinder problem, under either plane stress or plane strain conditions, gives a general result for the radial and hoop stresses as

$$\sigma_{rr} = A - \frac{B}{r^2}, \quad \sigma_{\theta\theta} = A + \frac{B}{r^2} \quad (6.5)$$

YOU THINK. YOU CAN WORK AT RMB

**RAND
MERCHANT
BANK**
A division of FirstRand Bank Limited
Traditional values. Innovative ideas.

Rand Merchant Bank uses good business to create a better world, which is one of the reasons that the country's top talent chooses to work at RMB. For more information visit us at www.rmb.co.za

Thinking that can change your world

Rand Merchant Bank is an Authorised Financial Services Provider



Click on the ad to read more

where r is radial distance from the centre of the cylinder, and A and B are constants. Stress σ_{rr} is the direct radial stress in the r direction while hoop stress $\sigma_{\theta\theta}$ is, as its name suggests, the direct stress in the circumferential direction at any given radial position (for example, along the inner or outer surfaces of the cylinder). The values of the constants depend on the boundary conditions applied to the cylinder. In the present problem, these are

$$\sigma_{rr} = -p \text{ at } r = r_1 \quad \text{and} \quad \sigma_{rr} = 0 \text{ at } r = r_2 \quad (6.6)$$

$$\text{Hence } A = \frac{p}{(K^2-1)}; \quad \text{and} \quad B = \frac{pr_2^2}{(K^2-1)} \quad \text{where } K = \frac{r_2}{r_1} \quad (6.7)$$

and the hoop stresses at the inner and outer cylinder surfaces are

$$\sigma_{\theta\theta} = \frac{p(K^2+1)}{(K^2-1)} \quad \text{at} \quad r = r_1 \quad \text{and} \quad \sigma_{\theta\theta} = \frac{2p}{(K^2-1)} \quad \text{at} \quad r = r_2 \quad (6.8)$$

In this problem $K = 2$ and $p = 3$, and these hoop stresses are 5 and 2, respectively.

These results for stresses do not depend on the values of either Young's modulus or Poisson's ratio. Radial displacement, u_r , can be found from the hoop strain

$$e_{\theta\theta} = \frac{u_r}{r} = \frac{1}{E^*}(\sigma_{\theta\theta} - \nu^* \sigma_r) \quad (6.9)$$

$$\text{At the inner cylinder surface } u_r = \frac{r_1}{E^*}(5 + 3\nu^*) \quad (6.10)$$

$$\text{and at the outer surface } u_r = \frac{2r_2}{E^*} \quad (6.11)$$

From Equations 5.7

$$E^* = \frac{E}{(1-\nu^2)} = 2197.80 \quad \text{and} \quad \nu^* = \frac{\nu}{(1-\nu)} = 0.428571 \quad (6.12)$$

and the displacements at the inner and outer cylinder surfaces are

$$u_r = 0.011440 \quad \text{and} \quad u_r = 0.007280$$

Bearing in mind that program BEM2EQ cannot accept boundary segment specifications with angles greater than π , the simplest mesh to represent the geometry shown in Figure 6.5 requires four segments, two on the outer boundary with end points arbitrarily chosen at (8, 0) and (-8, 0), and two on the inner boundary with end points at (4, 0) and (-4, 0). These end points are the points F, G, H and I in Figure 6.5. The origin for the global Cartesian co-ordinates is the centre of the cylinder. The following is an annotated DATA input file for this problem, for the case of four elements (9 nodes) per semi-circular segment, two per quadrant. This means that each element represents an arc of $\pi/4$ by means of a parabolic (quadratic) curve passing through both the ends and midpoint of the arc.

```
THICK-WALLED CYLINDER

STRAIN                                (Plane strain selected)

2000. 0.3                             (Young's modulus and Poisson's ratio)

2                                      (Number of boundaries)

2                                      (Number of segments on first boundary)

8. 0. -8. 0.                          (Co-ordinates of the segment ends)

8. 4 1.                               (Radii of curvature, numbers of elements
and length ratios)

2                                      (Number of segments on second boundary)

4. 0. -4. 0.                          (Co-ordinates of the segment ends)

-4. 4 1.                              (Radii of curvature, numbers of elements
and length ratios)

0 2 3                                 (2 prescribed stresses and 3 point constraints)

3 -3. 0.                              (Normal stress = -3 on segment 3)

4 -3. 0.                              (Normal stress = -3 on segment 4)

1 1                                   (Node 1, point F, constrained in x direction)

1 2                                   (Node 1, point F, constrained in y direction)

9 2                                   (Node 9, point G, constrained in y direction)
```

As in the case of the simple tension problem of Section 6.2.1, sufficient displacements constraints must be prescribed to prevent rigid body movement of the whole cylinder. Those chosen are no movement in either the x or y direction for point F, and no movement in the x direction for point G. In terms of node numbers, for the present mesh of 4 elements per segment these are 1 and 9.

Table 6.2 shows the results obtained for meshes ranging from one element per quadrant up to four elements per quadrant. For the computed stresses, two values are shown, the first for element end nodes, and the second for midside nodes. The computed displacements are for the highest points on surfaces, J and K in the figure, in the direction.

Elements per quadrant	At inner surface		At outer surface	
	$\sigma_{\theta\theta}$	u_r	$\sigma_{\theta\theta}$	u_r
Exact	5	0.011440	2	0.007280
1	4.8985	0.011029	1.9653	0.007030
	4.7743		1.9313	
2	4.9940	0.011417	1.9967	0.007213
	4.9875		1.9965	
3	5.0006	0.011437	1.9996	0.007279
	4.9984		1.9996	
4	5.0014	0.011440	2.0001	0.007280
	5.0001		2.0001	

Table 6.2 Results for the thick-walled cylinder problem



Discover the truth at www.deloitte.ca/careers

Deloitte.

© Deloitte & Touche LLP and affiliated entities.



Click on the ad to read more

Since the problem is axi-symmetric it is to be expected that the computed stresses at all the nodes on the inner boundary would be identical, and similarly for all the nodes on the outer boundary. This proves to be the case, but for expected fluctuations between the values at element end nodes and midside nodes. At four elements per quadrant the results are almost correct to four significant figures, which is adequate for most practical purposes. In this case each element covers a circular arc of 22.5° , which demonstrates the ability of quadratic elements to accurately model curved boundaries. Thirty two elements having 64 nodes are sufficient to model this multiply-connected solution domain.

6.3 An Example: Confined Compression of a Rubber Block

A practical problem for solution by program BEM2EQ is the compression of a rubber block which is confined between rigid surfaces. It has important engineering applications in the bearings for bridges. Bridges are often supported on rubber pads. These are relatively flexible in shear, allowing the bridge to expand or contract in length with changes in ambient temperature and the load it is carrying. But they are much stiffer in compression. The reason for this is that rubber, which is much more flexible than metals, is incompressible (or very nearly incompressible), having a Poisson's ratio of 0.5. It is worth noting that many finite element methods are incapable of analysing incompressible material problems under plane strain conditions.

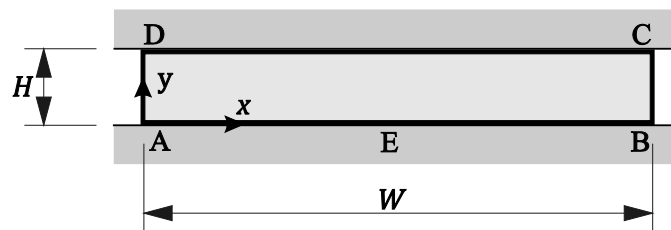


Figure 6.6 Rubber block between rigid surfaces

Figure 6.6 shows a rubber block sandwiched between rigid surfaces, to which it is bonded. The height and width of the block are H and W , respectively. Consider the case where $W/H = 10$. The dimension of the block in the direction normal to the plane shown is large, so that a state of plane strain may be assumed. With the bottom surface AB fixed in both global co-ordinate directions, the top surface is to be lowered by 0.1% of the thickness H . The (maximum) compressive stress at point E is to be computed, together with the total compressive force on the surface AB, both to be expressed as multiples of the stress and force which would exist if the block were in a state of simple compression.

If the Young's modulus is taken as 1000, a 0.1% simple compressive strain would give a stress of $0.001 \times 1000 = 1$. If block width is taken as 1, then the total forces on the upper and lower surfaces would also be 1 under the same conditions. So in the boundary element model it is convenient to take $W = 1, H = 0.1$ and an upper surface vertical displacement of $v = -0.0001$ (0.1% of 0.1). The computed stresses and forces will be the required multiples.

The boundary of the domain is conveniently divided into four segments: AB, BC, CD and DA. With a wide disparity in the lengths of the segments, it is no longer appropriate to use the same number of elements on each segment. Best results are normally obtained when the lengths of adjacent elements are reasonably similar (not differing by more than a factor of about 2 or 3). In this problem it is appropriate to have ten times as many elements on AB and CD as on BC and DA. This means that the coarsest mesh which can be used has 10 elements each on AB and CD, and only one each on BC and DA. This is also desirable from the point of view that the domain is slender and it is important that opposite sides are not too close together in relation to element sizes (for the reasons discussed in Section 4.2.1 for potential problems). With 10 elements on each of the horizontal edges, their distance apart is equal to the common lengths of the elements. Mesh refinement (keeping the 10:1 element number ratio between the horizontal and vertical segments) will only improve the situation.

Table 6.3 shows the effects of varying the number of elements on the horizontal segments from 10 to 50 (with the numbers of elements on the vertical segments always one tenth of these).

Elements per segment	σ_{yy} at E	Total force on AB
10	-50.43	-34.51
20	-50.15	-34.03
30	-50.18	-33.95
40	-50.19	-33.92
50	-50.19	-33.91

Table 6.3 Stress and force results for varying levels of mesh refinement

There is no change in the maximum stress of 50.2 beyond 20 elements per horizontal segment and the total force has settled to 33.9 by 30 elements per segment. As multiples of the values expected for simple compression these are large numbers, and would have been much higher if larger ratios had been considered. The explanation is that, because the rubber material is assumed to be completely incompressible, the reduction in the height of the block caused by bringing the confining surfaces closer together can only be accommodated by material being forced out horizontally at the side edges, BC and DA. Because there are no displacements at the corners A, B, C and D the rubber is forced out into curved bulges. In this case the maximum horizontal displacements at the centres of BC and DA are each computed as 0.00072, 7.2 times the relative displacement of the confining surfaces. The driving forces required to produce this deformation are provided by the stress gradients in the horizontal direction moving away from the centre of the block. From a maximum of 50.2 at point E, both σ_{xx} and σ_{yy} (the state of stress in the block is hydrostatic, with these stresses equal) follow a roughly parabolic horizontal distribution to finish at close to one at sides BC and DA. The average stress is 33.9, numerically equal to the total applied force.

6.4 An Example: Stress Concentration at a Hole in a Flat Plate

As a final example, consider the classic problem of the stress concentration created at a small circular hole at the centre of a thin flat square plate under tension. This is often used as a test of any numerical method of stress analysis. Figure 6.7 shows the geometry and loading. The radius of the hole is a , and the width and height of the plate are $2b$: the ratio of hole diameter to plate width is a/b . For present purposes this ratio is taken as $1/20$. A uniform tensile stress is applied to both the top and bottom edges of the plate.

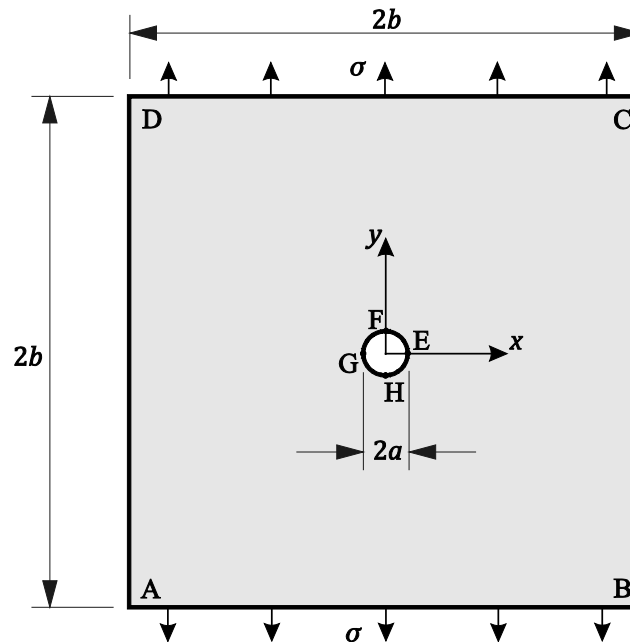


Figure 6.7 Square plate with a small central hole

There is no exact analytical solution for the case of a plate of finite width and height, although one does exist for an infinitely large plate. This gives a maximum (tensile) hoop stress at both points E and G of 3σ , together with a (compressive) hoop stress at both points F and H of $-\sigma$. It also indicates that there are rapid changes in stresses and deformations close to the hole, which make this a relatively challenging application for any numerical method. The present problem is for a plate of finite width, although the width to diameter ratio is relatively high. The maximum stress concentration factor (ratio of local stress to remote applied stress) can be expected to be a little higher than 3.

The outer boundary of the domain is conveniently divided into four segments: AB, BC, CD and DA, and the inner boundary into two: EHG and GFE. On the outer boundary the stresses and displacements do not vary much and only a few boundary elements are required: four per segment is sufficient. Rather more elements are likely to be required on the inner boundary. Point constraints are required to prevent rigid body movement: point A is constrained in both directions, while point B is constrained not to move in the y direction.

The DATA file for this problem with four elements per semi-circular segment (two per quadrant) is

STRESS CONCENTRATION	
STRESS	(Plane stress selected)
2000. 0.3	(Young's modulus and Poisson's ratio)
2	(Number of boundaries)
4	(Number of segments on first boundary)
-0.5 -0.5 0.5 -0.5 0.5 0.5 -0.5 0.5	(Co-ordinates of the segment ends)
0. 4 1.	(Radii of curvature, numbers of elements
0. 4 1.	and length ratios)
0. 4 1.	
0. 4 1.	
2	(Number of segments on second boundary)
0.025 0. -0.025 0.	(Co-ordinates of the segment ends)
-0.025 4 1.	(Radii of curvature, numbers of elements
-0.025 4 1.	and length ratios)
0 2 3	(2 prescribed stresses and 3 point constraints)
1 1. 0.	(Normal stress = 1 on segment 1)
3 1. 0.	(Normal stress = 1 on segment 3)
1 1	(Node 1 constrained in x direction)
1 2	(Node 1 constrained in y direction)
9 2	(Node 9 constrained in y direction)

**I WANT TO CHANGE DIRECTION,
AND THE WORLD.**

GOT-THE-ENERGY-TO-LEAD.COM

We believe that energy suppliers should be renewable, too. We are therefore looking for enthusiastic new colleagues with plenty of ideas who want to join RWE in changing the world. Visit us online to find out what we are offering and how we are working together to ensure the energy of the future.

RWE
The energy to lead



Click on the ad to read more

The overall dimensions of the plate are taken as 1, so that $a=0.025$ and $b=0.5$. The Young's modulus and Poisson's ratio are arbitrarily chosen as 2000 and 0.3, respectively, although they do not affect the results for stress concentration.

An edited version of the RESULTS data file output produced is

QUADRATIC BOUNDARY ELEMENT SOLUTION FOR TWO DIMENSIONAL ELASTIC PROBLEM
STRESS CONCENTRATION

UNDER PLANE STRESS CONDITIONS

YOUNG'S MODULUS = 0.2000E+04 POISSON'S RATIO = 0.300

PRESCRIBED STRESS BOUNDARY CONDITIONS

SIGNN = 0.1000E+01 SIGSN = 0.0000E+00 ON ELEMENTS 1 TO 4

SIGNN = 0.1000E+01 SIGSN = 0.0000E+00 ON ELEMENTS 9 TO 12

NODE 1 CONSTRAINED WITH U = 0

NODE 1 CONSTRAINED WITH V = 0

NODE 9 CONSTRAINED WITH V = 0

STRESSES AT THE NODES OF THE ELEMENTS

M IN	SIGNN	SIGSS	SIGSN	SIGE
17 1	0.000000E+00	0.301684E+01	0.000000E+00	0.301684E+01
17 2	0.000000E+00	0.242750E+01	0.000000E+00	0.242750E+01
17 3	0.000000E+00	0.105800E+01	0.000000E+00	0.105800E+01
18 1	0.000000E+00	0.945216E+00	0.000000E+00	0.945216E+00
18 2	0.000000E+00	-0.424362E+00	0.000000E+00	0.424362E+00
18 3	0.000000E+00	-0.101370E+01	0.000000E+00	0.101370E+01
19 1	0.000000E+00	-0.101370E+01	0.000000E+00	0.101370E+01
19 2	0.000000E+00	-0.424375E+00	0.000000E+00	0.424375E+00
19 3	0.000000E+00	0.945128E+00	0.000000E+00	0.945128E+00
20 1	0.000000E+00	0.105804E+01	0.000000E+00	0.105804E+01
20 2	0.000000E+00	0.242751E+01	0.000000E+00	0.242751E+01
20 3	0.000000E+00	0.301677E+01	0.000000E+00	0.301677E+01
21 1	0.000000E+00	0.301691E+01	0.000000E+00	0.301691E+01
21 2	0.000000E+00	0.242751E+01	0.000000E+00	0.242751E+01
21 3	0.000000E+00	0.105795E+01	0.000000E+00	0.105795E+01
22 1	0.000000E+00	0.945229E+00	0.000000E+00	0.945229E+00
22 2	0.000000E+00	-0.424373E+00	0.000000E+00	0.424373E+00
22 3	0.000000E+00	-0.101371E+01	0.000000E+00	0.101371E+01
23 1	0.000000E+00	-0.101371E+01	0.000000E+00	0.101371E+01
23 2	0.000000E+00	-0.424386E+00	0.000000E+00	0.424386E+00
23 3	0.000000E+00	0.945244E+00	0.000000E+00	0.945244E+00
24 1	0.000000E+00	0.105804E+01	0.000000E+00	0.105804E+01
24 2	0.000000E+00	0.242751E+01	0.000000E+00	0.242751E+01
24 3	0.000000E+00	0.301683E+01	0.000000E+00	0.301683E+01

FORCES ACTING ON THE BOUNDARY SEGMENTS

SEGMENT	FX	FY
1	-0.56196E-07	-0.10000E+01
2	0.00000E+00	0.00000E+00
3	0.56196E-07	0.10000E+01
4	0.00000E+00	0.00000E+00
5	0.00000E+00	0.00000E+00
6	0.00000E+00	0.00000E+00

Results for nodal point displacements and tractions have been omitted, together with stresses on the outer boundary (elements 1 to 16). The hoop stress (SIGSS) at point E is either that for the first node of element 17 (3.01684) or the third node of element 24 (3.01683). At point G it is either that for the third node of element 20 (3.01677) or the first node of element 21 (3.01691). The four values are consistent, and give an average to five significant figures of 3.0168. The hoop stress at point F is either that for the third node of element 18 (-1.01370) or the first node of element 19 (-1.01370). At point H it is either that for the third node of element 22 (-1.01371) or the first node of element 23 (-1.01371). The four values are consistent, and give an average to five significant figures of -1.01371. With only two elements per quadrant of the hole in the plate the stress values are very close to what is expected.



Corporate eLibrary

See our Business Solutions for employee learning

[Click here](#)

Management

Time Management

Problem solving

Self-Confidence

Effectiveness

Project Management

Goal setting

Motivation

Coaching

121

[Click on the ad to read more](#)

Download free eBooks at bookboon.com

Elements per quadrant	σ_{yy} at E and G	σ_{xx} at F and H
2	3.0168	-1.0137
3	3.0206	-1.0163
4	3.0211	-1.0167
6	3.0215	-1.0168

Table 6.4 Stress concentration results for varying levels of mesh refinement at the hole

Table 6.4 shows the effects of refining the mesh around the hole. Both stress values increase in magnitude a little and converge to 3.021 at the sides of the hole and -1.017 at the top and bottom of the hole. Converged results are achieved with only a modest number of boundary elements: at 6 elements per quadrant the total used is 40. With rapid variations of stresses and deformations near the hole, this is the type of problem where the ability to vary the element distributions along segments (to give small elements at the stress concentrations) might have been expected to offer significant benefits. The quality of the results from using elements of constant size suggests, however, that this is not necessary. The situation would have been different if symmetry had been invoked to model only one quarter of the plate (say, the first quadrant in the plane). Element size variation on the lines of symmetry would have been required to accommodate both small elements near the hole and much larger elements at the outer boundary.

6.5 Discussion

It is clear that quadratic boundary elements applied to elastic stress analysis problems offer good accuracy with relatively small numbers of elements. The scope of the program described in this chapter is deliberately limited, to keep the coding reasonably straightforward to understand. Many other features can be incorporated: for example, additional types of boundary conditions, displacement and stress calculations at internal points, and inclusion of body forces such as gravity and centrifugal effects.

Another useful facility is the ability to be able to divide a domain into several distinct regions, with elements along the boundaries between them. The different regions can be of different materials, having different elastic properties. This makes possible the treatment of not only multi-material problems, but also multiple bodies in contact with each other. Another class of problems that can be catered for is that of crack problems in linear elastic fracture mechanics. One of the boundary element end nodes can be located at the crack tip and the midside nodes in the adjacent elements shifted to force the required stress singularity at the crack tip (just as is done in finite element methods). If both faces of the crack have to be modelled, then the multi-region facility is very useful. This is because coincident nodal points in a single boundary element region are not permitted, and nearly coincident nodes give rise to inaccurate analysis. Coincident and nearly coincident nodes in different regions, on the two sides of the crack, present no difficulties.

Problems

- 6.1** In the stress concentration problem illustrated in Figure 6.7 (with $a/b = 1/20$), instead of the tensile loading applied to the top and bottom edges of the plate, the hole itself is subjected to a uniform internal pressure. Use program BEM2EQ to find the maximum stress concentration factor (relative to the applied pressure). Compare this value with the available analytical solutions.
- 6.2** Consider the long thick-walled cylinder whose cross-section is shown in Figure 6.5, but with its outer surface rigidly contained. An internal pressure of $p = 1$ is applied. The Young's modulus and Poisson's ratio of the material are 1000 and 0.3, respectively. Use program BEM2EQ to find both the stresses in the cylinder at the inner and outer surfaces, and the radial displacement of the inner surface, and compare with analytical results.
- 6.3** Figure 6.8 shows a thin rectangular plate with a small semi-circular notch at the centre of one horizontal edge. The radius of the notch is $1/20^{\text{th}}$ of the vertical height of the plate. The plate is subjected to uniform tensile stresses on its vertical edges. Use program BEM2EQ to find the maximum stress concentration at the notch. Take Poisson's ratio to be 0.4.

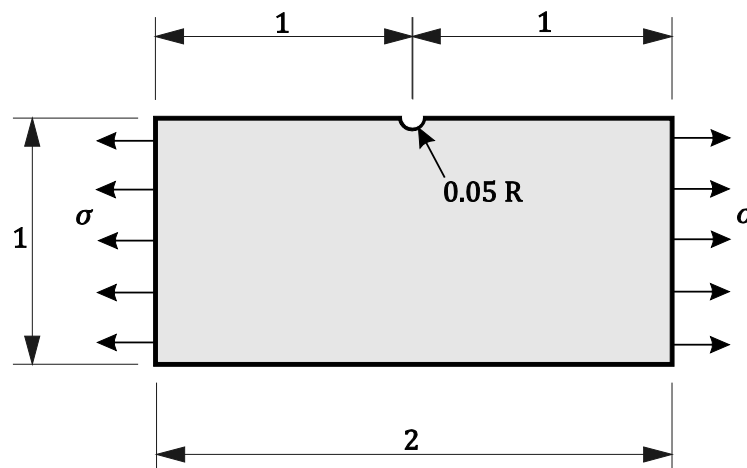


Figure 6.8 Notched plate in tension

- 6.4** Figure 6.9 shows a thin square plate with a square hole at its centre. The corners of the hole are rounded, with radii of curvature 10% of the hole dimensions, 5% of the overall plate dimensions. Use program BEM2EQ to find the maximum stress concentration at the hole. Take Poisson's ratio to be 0.25.

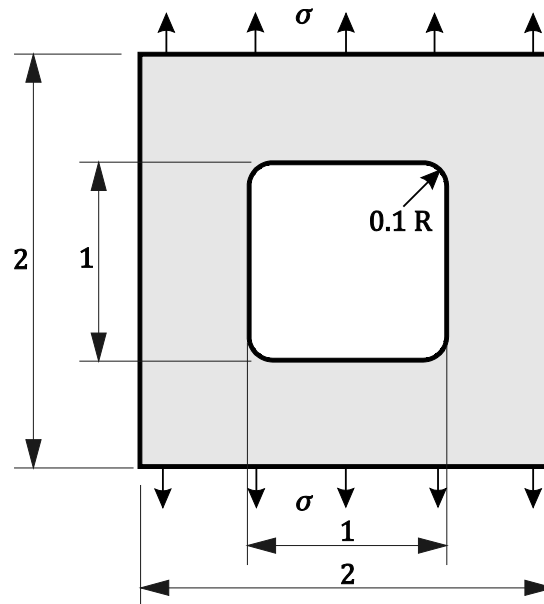


Figure 6.9 Square plate with a square central hole

- 6.5** A cast iron pedestal is trapezoidal in cross-section, with a horizontal base 0.3 m wide, horizontal top 0.1 m wide and height 0.1 m, and is symmetrical about the vertical centre line. Plane strain conditions may be assumed. The Young's modulus is 70 GN/m² and Poisson's ratio 0.25. The top surface of the pedestal is uniformly loaded in compression and the base rests on a rigid surface. The purpose of the pedestal is to spread the load over the supporting surface. Use program BEM2EQ to find the ratio between the maximum compressive stress applied to the supporting surface and the stress applied to the top of the pedestal. The surface is rough enough to prevent any slipping. Is a coefficient of friction of 0.5 large enough to achieve this?

- 6.6** Figure 6.10 shows a steel tension test piece, which is 3 mm thick. The central section is 20 mm wide, increasing to 30 mm at the ends where it is gripped. The transition in width is via quadrant fillets with radii of 5 mm. Young's modulus and Poisson's ratio for steel may be taken as 207 GN/m² and 0.3, respectively. The ends of the test piece are subjected to uniform stresses of 60 MN/m². Use program BEM2EQ to find the stiffness of the test piece and the maximum stress induced. Find also the maximum and minimum axial stresses over the central 30 mm, the region where an extensometer is to be attached and where it is intended that the axial stress is uniform.

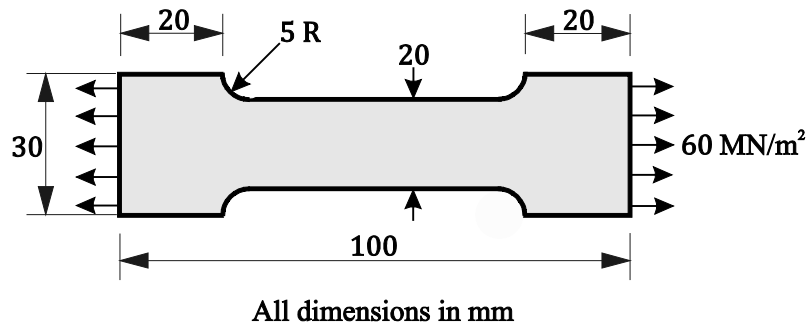


Figure 6.10 Tension test piece

Brain power



Plug into The Power of Knowledge Engineering.
Visit us at www.skf.com/knowledge

By 2020, wind could provide one-tenth of our planet's electricity needs. Already today, SKF's innovative know-how is crucial to running a large proportion of the world's wind turbines.

Up to 25 % of the generating costs relate to maintenance. These can be reduced dramatically thanks to our systems for on-line condition monitoring and automatic lubrication. We help make it more economical to create cleaner, cheaper energy out of thin air.

By sharing our experience, expertise, and creativity, industries can boost performance beyond expectations. Therefore we need the best employees who can meet this challenge!

The Power of Knowledge Engineering





Click on the ad to read more

- 6.7** If the edges of the plate in Figure 6.7 (with $a/b = 1/20$) are subject to pure shear rather than simple tension, use program BEM2EQ to find the maximum stress concentration at the hole, expressed as a multiple of the applied shear stress.
- 6.8** Consider the stress concentration problem shown in Figure 6.7 not with a circular hole at the centre but with an elliptical hole. The major axis of the ellipse lies along the x axis and is $1/20^{\text{th}}$ of the plate width, while the minor lies along the axis and is half the major axis. Use program BEM2EQ to find the maximum stress concentration factor. Take Poisson's ratio to be 0.3.
- 6.9** Consider the stress concentration problem shown in Figure 6.7 not with a single circular hole at the centre but with two holes of equal size ($a/b = 1/20$), side by side. Their centres are on the x axis and equidistant from the centre of the plate. The distance between the centres is four times their common radius. Use program BEM2EQ to find the stress concentration factors at the sides of the holes. Take Poisson's ratio to be 0.3. Consider both (a) when the applied uniform tensile stresses are in the y direction as shown, at right angles to the line of hole centres, and (b) when they are in the x direction, parallel to the line of centres.
- 6.10** Figure 6.11 shows the cross-section of a long circular steel cylinder, internal and external radii 100 and 250 mm, respectively. Running the full length of the cylinder is a straight groove which is semi-circular in shape with a radius of 5 mm. Young's modulus and Poisson's ratio for steel may be taken as 207 GN/m^2 and 0.3, respectively. Use program BEM2EQ to find the maximum von Mises equivalent stress in the cylinder when an internal pressure of 120 MN/m^2 is applied.

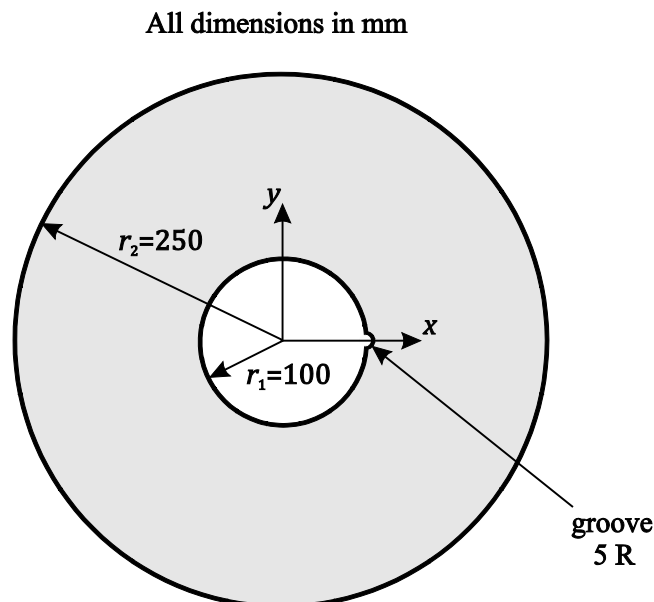


Figure 6.11 Thick-walled cylinder with a groove

- 6.11** The long thick-walled internally pressurised cylinder described in Section 6.2.4 has been incorrectly manufactured so that it is slightly elliptical in cross-section, with major axis 1% greater than minor axis. Use program BEM2EQ to find the ratio between the maximum hoop stress (at the internal surface) and the applied pressure, and compare this to the value for the intended circular cylinder.
- 6.12** Part of the suspension for a motor vehicle involves a torsional spring formed by a circular cylindrical rubber bush which is 75 mm long and has inner and outer diameters of 20 mm and 50 mm, respectively. The outer surface is bonded to a rigid housing, and the inner surface to a metal shaft. The rubber has a Young's modulus of 80 MN/m² and a Poisson's ratio of 0.5. Use program BEM2EQ to find the torque which must be applied to the shaft to rotate it by 0.1 radians.

With us you can
shape the future.
Every single day.

For more information go to:
www.eon-career.com

Your energy shapes the future.

e-on



7 Further Applications

In Chapter 1 many continuum mechanics problems in a wide range of engineering subjects were categorised according to their governing differential equations into either the harmonic or biharmonic types. Problems governed by harmonic equations are, in engineering terms, more often referred to as potential problems, such as heat conduction, ideal fluid flow, porous medium flow, torsion and at least one form of fluid flow. Part I of the book was devoted to such potential problems, and boundary element methods were developed. Part II has been concerned with biharmonic problems in elastic stress analysis, particularly two-dimensional plane strain and plane stress. The purpose of this final chapter is to review briefly some further applications of boundary element methods.

7.1 Axi-symmetric Problems

There are many problems in engineering, both harmonic and biharmonic, which are symmetrical in terms of both geometry and boundary conditions about some axis. For example, the thick-walled cylinder test problems considered in Sections 3.2.2, 4.2.2 and 6.2.4 were symmetrical about the axis of the cylinder. In each case a cross-section of the cylinder at right angles to the axis was treated as the solution domain. It would also have been possible to treat as the solution domain a radial cross-sectional plane through the wall of the cylinder, which also passed through the cylinder axis. Boundary elements could be used to model the boundary of the domain. Interestingly, if a solid rather than hollow axi-symmetric component is modelled, so that the axis of symmetry forms part of the boundary of the solution domain, then it is not necessary to have boundary elements on the axis. The level of mathematics required to formulate axi-symmetric analyses is, however, rather more challenging than for two-dimensional problems.

7.2 Higher-Order Boundary Elements

It is of course possible to use boundary elements with shape functions of orders higher than quadratic. For example, some work has been done with cubic elements, including some which provide a smooth transition (continuity of slope) between one element and the next. The high quality of the results that can be obtained using quadratic elements, however, means that they are generally the elements of choice.

7.3 Three-dimensional Problems

Three-dimensional problems are amenable to solution by a boundary element approach. Again, only the boundary of the relevant solution domain is modelled. The surface is covered with two-dimensional boundary elements. The equivalent in three-dimensional analysis of the constant element for two-dimensional problems is the constant triangle or quadrilateral. Over either such element the potential and potential gradient normal to the surface (in a potential problem) or the displacements and tractions (in a stress analysis problem) are assumed to be constant over the element, and equal to the value at the node at the middle of the element. For potential problems there remains only one unknown at each node. For stress analysis there are now three, typically either displacements or tractions in each of the three global co-ordinate directions, and three equations are required at each node.

Progressing from constant to quadratic elements, the latter can again be either triangular or quadrilateral in shape, though not necessarily flat or with straight edges (in the same way that quadratic line elements are not necessarily straight lines). Isoparametric elements are generally preferred, with the geometric shape being defined in terms of quadratic functions of the three spatial co-ordinates. Triangular elements typically have six nodes, quadrilateral eight, one at each corner and one at the centre of each side. Quadrilateral elements are often used to model most of a surface, with triangles being useful to deal with local geometric features, particularly where a mesh refinement is required to suit a particular problem. Indeed, there are situations where a mesh refinement cannot be effected without using at least a few triangles.

While three-dimensional meshes of (curved) triangles and quadrilaterals are significantly more difficult to create than two-dimensional meshes of line elements, they still only involve the discretisation of the boundary rather than the interior of a domain as well. Using boundary elements, the dimensionality of the computational problem is reduced by one. Nevertheless, the amount of computation involved in a three-dimensional boundary element analysis is very substantial, particularly because the coefficient matrix in the final set of algebraic equations is essentially fully populated with non-zero values.

be > your degree

Bring your talent and passion to a global organization at the forefront of business, technology and innovation. Discover how great you can be. Visit accenture.com/bookboon

Be greater than.
consulting | technology | outsourcing

accenture
High performance. Delivered.

© 2013 Accenture. All rights reserved.



7.4 Non-Linear Problems

All the problems so far considered are linear in the sense that they involve the solution of sets of linear algebraic equations. Non-linearities can be introduced by either geometric, material property or boundary condition effects. Geometric non-linearities arise, for example, in problems involving solid media in which the strains are sufficiently large to significantly affect the shape of the solution domain. Such problems can be solved by an incremental approach in which the non-linear analysis is replaced by a series of linear analyses for progressively increasing external loads, after each of which the boundary element mesh geometry is recomputed. Non-linear behaviour due to boundary conditions arises in, for example, heat transfer problems with radiation boundary conditions, where the rate of heat transfer to or from a boundary depends on the fourth power of the absolute temperature there, rather than the first power in a convection boundary condition. In elastic stress analysis problems, contact between two or more bodies gives rise to non-linear behaviour: increased contact force between them changes the region of contact and hence modifies the boundary conditions. Again incremental approaches are possible.

Examples of material non-linearities include non-Newtonian fluids and non-linearly elastic or elastoplastic solids, whose properties are functions of the local state of deformation or stress. For example, the viscosity of a fluid may be a function of the local strain rates (velocity gradients), or the elastic modulus of a solid may be a function of the local state of stress or strain. The form of dependence in each case would be determined from experimental data. It is also possible for material properties, such as viscosity, thermal conductivity, or elastic modulus to be significantly dependent on temperature, and therefore not known in advance. Such problems cannot be solved by boundary element method which discretise only the boundary: some internal discretisation and analysis is required adequately to treat the non-linearity. At least some of the benefits of the boundary element approach may thereby be lost.

7.5 Comparison with Other Methods

Other methods for solving engineering continuum mechanics problems include finite element, finite difference and finite volume methods. They are what are often referred to as domain methods: the entire two- or three-dimensional domain of the problem has to be discretised, rather than just the boundary. The principal advantage of boundary element methods is the effective reduction in dimensionality of the numerical problem. At least for linear situations, a two-dimensional physical problem requires only a one-dimensional mesh of boundary line elements, while a three-dimensional physical problem requires a two-dimensional mesh of typically triangular or quadrilateral elements. Benefits of this simplification can be seen in the problems addressed in this book: meshes of line elements for two-dimensional problems are easily generated using only a small number of geometric parameters, leading to very concise data input files. Analyses of even quite complicated problems can be set up quickly and easily.

Although it has not been demonstrated in this book, boundary element meshes, particularly of quadratic elements, are not merely domain method meshes with the internal points removed. For a given level of boundary discretisation, boundary element methods normally give substantially greater accuracy. But, although far fewer algebraic equations have to be solved, they have to be treated as being fully populated with non-zero coefficients. The computational benefits, although still present, are less clearcut.

Features of boundary element methods which are perhaps deterrents to their wider use by engineers are the mathematics involved. Not only are they arguably more complex than for domain methods, but also tend to be of unfamiliar types: for example, integral equations, singular fundamental solutions, and analyses often expressed in tensor notation. An aim of this book has been to try to simplify and clarify the mathematics involved, even when this is sometimes at the expense of more bulky and less mathematically elegant equations. A further aim has been to demonstrate that boundary element methods are worthy of serious consideration for engineering analyses, and are particularly straightforward to use.



"I studied English for 16 years but...
...I finally learned to speak it in just six lessons"

Jane, Chinese architect

ENGLISH OUT THERE

Click to hear me talking before and after my unique course download



Appendix A: Gaussian Quadrature

The aim of this appendix is to explain the Gaussian quadrature method of numerical integration, particularly as applied to boundary elements.

When a function cannot be integrated analytically, some form of numerical integration or quadrature must be used. Two of the simplest methods are the familiar trapezium and Simpson's rules. Given a function $f(x)$, values of the function are obtained for a number of values of the independent variable x over the required range of integration, and an approximate value of the integral obtained from them. An important feature (and limitation) of these simple methods is that the values of the independent variable at which the function is sampled are often taken at constant intervals, and certainly little attention is paid to the optimum choice of sampling points.

If there is no restriction on the positions at which the function can be evaluated, then there is a class of numerical integration procedures referred to as Gaussian quadrature, which is to be preferred. The advantage over the trapezium and Simpson's rules is that greater accuracy for a given number of function evaluations, and therefore of overall computational effort, can be obtained.

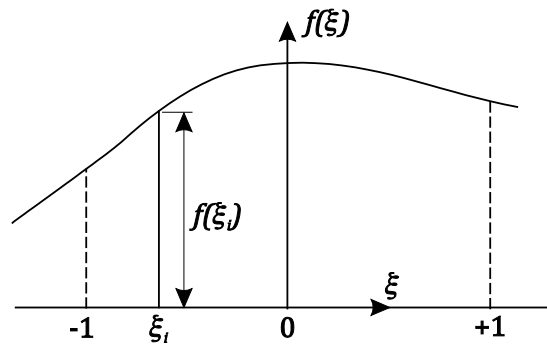


Figure A.1 Typical function to be integrated numerically

Figure A.1 shows a typical function to be integrated: the area between the curve $f(\xi)$ and the ξ axis between the limits $\xi = -1$ and $\xi = +1$ is required. It is convenient to transform the actual independent variable, such as x , into a dimensionless variable with these limits. This is the reason for introducing the local intrinsic co-ordinate ξ for quadratic boundary elements. Assuming that the function is evaluated at n points, known as the Gauss points, the required integral can be expressed approximately as

$$\int_{-1}^{+1} f(\xi) d\xi \approx \sum_{p=1}^n \omega_p f(\xi_p) \quad (\text{A.1})$$

where ω_p the are weighting factors. Simpson's rule (defined in Equation 3.49) is actually a particular form of this result in which, for a single application of the rule

$$\begin{aligned} n &= 3 \\ \xi_1 &= -1 & \xi_2 &= 0 & \xi_3 &= +1 \\ \omega_1 &= \omega_3 = \frac{1}{3} & \omega_2 &= \frac{4}{3} \end{aligned} \quad (\text{A.2})$$

Given the freedom to choose the ξ_p , however, the values of these and the corresponding weighting factors can be optimised for the integration of polynomial functions to give better accuracy. These are now well established and are available for a wide range of n , the number of Gauss points (for example, Stroud & Secrest 1966).

For any numerical integration, choosing the number of Gauss points involves a compromise between accuracy and the amount of computation involved. In boundary element methods the number of points is normally at least $n = 4$. In this book a higher value of $n = 8$ is preferred, for which the values of ξ and ω are

$$\begin{aligned} \xi_8 &= -\xi_1 = 0.9602898564 & \omega_1 &= \omega_8 = 0.1012285362 \\ \xi_7 &= -\xi_2 = 0.7966664774 & \omega_2 &= \omega_7 = 0.2223810344 \\ \xi_6 &= -\xi_3 = 0.5255324099 & \omega_3 &= \omega_6 = 0.3137066458 \\ \xi_5 &= -\xi_4 = 0.1834346424 & \omega_4 &= \omega_5 = 0.3626837833 \end{aligned} \quad (\text{A.3})$$

to ten decimal places.

Gaussian quadrature formulae have also been worked out for functions which involve particular forms of functions as multipliers, including cases of multipliers which are singular within the range of integration. One such case which is very relevant to boundary element methods is

$$\int_0^1 \ln\left(\frac{1}{\eta}\right) f(\eta) d\eta \approx \sum_{p=1}^n \omega_p^* f(\eta_p^*) \quad (\text{A.4})$$

the numerical values for $n = 8$ being

$$\begin{aligned}
 \eta_1^* &= 0.013320244 & \omega_1^* &= 0.164416605 \\
 \eta_2^* &= 0.079750429 & \omega_2^* &= 0.237525610 \\
 \eta_3^* &= 0.197871029 & \omega_3^* &= 0.226841984 \\
 \eta_4^* &= 0.354153994 & \omega_4^* &= 0.175754079 \\
 \eta_5^* &= 0.529458575 & \omega_5^* &= 0.112924030 \\
 \eta_6^* &= 0.701814530 & \omega_6^* &= 0.057872211 \\
 \eta_7^* &= 0.849379320 & \omega_7^* &= 0.020979074 \\
 \eta_8^* &= 0.953326450 & \omega_8^* &= 0.003686407
 \end{aligned}
 \tag{A.5}$$

Reference

Stroud, A.H. & Secrest, D. 1966, *Gaussian Quadrature Formulas*, Prentice-Hall.

DUKE
THE FUQUA
SCHOOL
OF BUSINESS

BUSINESS HAPPENS

HERE.

www.fuqua.duke.edu/globalmba

Learn More >



Click on the ad to read more

The aim of this appendix is to explain Gaussian elimination, which is a direct method for solving sets of simultaneous linear algebraic equations of the general form

where x_1, x_2, \dots, x_n are the unknowns, and the coefficients a_{ij} and b_i are all known constants. This set of equations can be expressed in matrix form as follows

$$[A][x] = [b] \quad (\text{B.3})$$

The unknowns are successively eliminated by algebraic manipulation. The first equation can be used to eliminate x_1 from the remaining $n - 1$ equations. The modified second equation is then used to eliminate x_2 from the remaining $n - 2$ equations, and so on until the last equation contains only x_n . Thus may be found, followed by all the other unknowns, by back substitution. Let the coefficients of $[A]$ and $[b]$ shown in Equations B.1 and B.2 be given the notation $a_{ij}^{(1)}$ and $b_i^{(1)}$. After the n th elimination, the modified coefficients are

$$b_i^{(k+1)} = b_i^{(k)} - \phi b_k^{(k)}$$

where $\emptyset = a_{ik}^{(k)}/a_{kk}^{(k)}$; $i = k + 1, k + 2, \dots, n$ and $j = k, k + 1, \dots, n$. Note that the column vector $[b]$ is treated just like a column of $[A]$, and advantage can be taken of this fact to simplify the computer programming of the process. The final set of equations is

$$\begin{aligned} a_{11}^{(1)}x_1 + a_{12}^{(1)}x_2 + \dots + a_{1n}^{(1)}x_n &= b_1^{(1)} \\ a_{22}^{(2)}x_2 + \dots + a_{2n}^{(2)}x_n &= b_2^{(2)} \\ &\dots \\ a_{nn}^{(n)}x_n &= b_n^{(n)} \end{aligned} \quad (\text{B.5})$$

Expressed in matrix terminology, the elimination process triangularises $[A]$. The unknowns are obtained in reverse order

$$\begin{aligned} x_n &= b_n^{(n)}/a_{nn}^{(n)} \\ x_i &= \left(b_i^{(i)} - \sum_{j=i+1}^n a_{ij}^{(i)}x_j \right) / a_{ii}^{(i)} \end{aligned} \quad (\text{B.6})$$

where $i = n - 1, n - 2, \dots, 1$.

A great many arithmetic operations are involved in solving large sets of equations by elimination. Any errors introduced, such as roundoff errors caused by representing numbers with only a finite number of significant figures, tend to be magnified and may become unacceptably large. Equations B.4 show that the elimination process involves many multiplications by the factors \emptyset . In order to minimize the effects of roundoff errors, these factors should be made as small as possible, and certainly less than one. Therefore, the pivotal coefficient $a_{kk}^{(k)}$ should be the largest coefficient in the leading column of the remaining submatrix

$$|a_{kk}^{(k)}| > |a_{ik}^{(k)}| \quad \text{for } i = k + 1, k + 2, \dots, n \quad (\text{B.7})$$

This condition also helps to avoid division by zero in Equations B.6, and can be achieved by a process known as partial pivoting. Immediately before each elimination, the leading column is searched for the largest coefficient. By interchanging equations, this can be made the pivotal coefficient to satisfy Equation B.7. The idea of partial pivoting can be extended to searching the whole of the remaining submatrix for the largest coefficient. Such complete pivoting involves interchanging both rows and columns, and is more difficult to program. Since it offers only modest advantages in terms of accuracy over partial pivoting, it is rarely used.

Another refinement which helps to minimize the effects of roundoff errors is to scale the equations to make their coefficients similar in magnitude. One way to do this is to normalize each equation so that the largest coefficient in each row of $[A]$ is of magnitude one. Scaling can be particularly important when corresponding coefficients in different equations differ by several orders of magnitude.

Any given set of equations may provide either duplicate (for example, two equations identical) or inconsistent information, and cannot be solved. In either case, the matrix of coefficients $[A]$ is said to be singular, and its determinant is zero. Such a condition can be detected during partial pivoting if it is impossible to find a non-zero pivotal value at some stage of the elimination process or at the start of the back substitution process. What is more difficult to detect, however, is when a set of equations is nearly singular or ill-conditioned. This arises when, for example, two equations provide very nearly the same information about the unknowns. Alternatively, the effect of roundoff errors may be to make what is a singular set of equations apparently ill-conditioned, in that rather than a zero pivotal coefficient, a very small value is obtained. Indeed, the detection of a very small pivotal coefficient can be used as a convenient test for either a singular or very ill-conditioned set of equations, but without being able to distinguish between them. By very small pivotal coefficient is meant a value very small in relation to the magnitudes of the coefficients of the matrix at the start of the elimination process, which are of order unity if scaling has been applied.

The following subprogram ELIMIN provides a Fortran implementation of Gaussian elimination.

```

      SUBROUTINE ELIMIN(A,X,NEQN,NROW,NCOL,IFLAG)
      !
      !  SUBROUTINE FOR SOLVING SIMULTANEOUS LINEAR EQUATIONS BY GAUSSIAN
      !  ELIMINATION WITH PARTIAL PIVOTING.
      !
      REAL :: A(NROW,NCOL),X(NROW)
      !
      !  INITIALIZE ILL-CONDITIONING FLAG.
      IFLAG=0
      !
      !  SCALE EACH EQUATION TO HAVE A MAXIMUM COEFFICIENT MAGNITUDE OF UNITY.
      JMAX=NEQN+1
      Each equation in turn: DO I=1,NEQN
      AMAX=0.
      Search for maximum: DO J=1,NEQN
      ABSA=ABS(A(I,J))
      IF(ABSA > AMAX) AMAX=ABSA
      END DO Search for maximum

```

```
!  
    Scale coefficients: DO J=1,JMAX  
    A(I,J)=A(I,J)/AMAX  
    END DO Scale coefficients  
!  
    END DO Each equation in turn  
!  
!  
! COMMENCE ELIMINATION PROCESS.  
    Eliminate each variable in turn: DO K=1,NEQN-1  
!  
! SEARCH LEADING COLUMN OF THE COEFFICIENT MATRIX FROM THE DIAGONAL  
! DOWNWARDS FOR THE LARGEST VALUE.  
    IMAX=K  
    Search for largest value: DO I=K+1,NEQN  
    IF (ABS (A(I,K)) > ABS (A(IMAX,K))) IMAX=I  
    END DO Search for largest value  
!  
! IF NECESSARY, INTERCHANGE EQUATIONS TO MAKE THE LARGEST COEFFICIENT  
! BECOME THE PIVOTAL COEFFICIENT.  
    IF (IMAX /= K) THEN
```

Join American online LIGS University!

Interactive Online programs
BBA, MBA, MSc, DBA and PhD

Special Christmas offer:

- ▶ enroll **by December 18th, 2014**
- ▶ **start studying and paying only in 2015**
- ▶ **save up to \$ 1,200** on the tuition!
- ▶ Interactive Online education
- ▶ visit ligsuniversity.com to find out more!

Note: LIGS University is not accredited by any nationally recognized accrediting agency listed by the US Secretary of Education. More info [here](http://ligsuniversity.com).



```

        Interchange coefficients: DO J=K,JMAX
        ATEMP=A(K,J)
        A(K,J)=A(IMAX,J)
        A(IMAX,J)=ATEMP
    END DO Interchange coefficients
END IF

!
! ELIMINATE X(K) FROM EQUATIONS (K+1) TO NEQN, FIRST TESTING FOR
! EXCESSIVELY SMALL PIVOTAL COEFFICIENT (ASSOCIATED WITH A SINGULAR
! OR VERY ILL-CONDITIONED MATRIX).
    IF (ABS(A(K,K)) < 1.E-5) THEN
        IFLAG=1
        RETURN
    END IF
    Each of remaining equations: DO I=K+1,NEQN
    FACT=A(I,K)/A(K,K)
    Modify coefficients: DO J=K,JMAX
    A(I,J)=A(I,J)-FACT*A(K,J)
    END DO Modify coefficients
    END DO Each of remaining equations
!
    END DO Eliminate each variable in turn
!
! SOLVE THE EQUATIONS BY BACK SUBSTITUTION, FIRST TESTING
! FOR AN EXCESSIVELY SMALL LAST DIAGONAL COEFFICIENT.
    IF (ABS(A(NEQN,NEQN)) < 1.E-5) THEN
        IFLAG=1
        RETURN
    END IF
    X(NEQN)=A(NEQN,JMAX)/A(NEQN,NEQN)
    Then each unknown in turn backwards: DO I=NEQN-1,1,-1
    SUM=A(I,JMAX)
    Sum products: DO J=I+1,NEQN
    SUM=SUM-A(I,J)*X(J)
    END DO Sum products
    X(I)=SUM/A(I,I)
    END DO Then each unknown in turn backwards
    RETURN
END SUBROUTINE ELIMIN

```

The arguments of ELIMIN include the array of equation coefficients, A , which incorporates the vector of constants $[b]$ as its last column, and the solution vector, X . The argument NEQN enters the number of equations to be solved, while IFLAG returns an integer flag value to the calling program to warn of a singular or very ill-conditioned coefficient matrix.

The first action of the program is to scale the coefficients of the equations to give a maximum coefficient magnitude of unity in each row of $[A]$. The elimination process is then started: each equation, with the exception of the last, is used in turn to eliminate the corresponding unknown from the subsequent equations. The current elimination number is given by K , and is equivalent to k in Equations B.4. Before performing the necessary eliminations with a particular equation, however, a search is made down the leading column of the remaining submatrix to find the coefficient of greatest magnitude, as defined by Equation B.7. The search technique locates the row number of the largest coefficient, IMAX, by first assuming that it corresponds to the coefficient on the diagonal, and only changing this if a larger coefficient is found. If the pivotal coefficient is not the largest, the relevant equations are interchanged by interchanging all their coefficients. In computing terms, this movement of data between storage locations is inefficient. While it could be avoided by keeping a record of the revised order in which the equations are to be considered, this makes the program significantly more difficult to understand. For present purposes, some computational efficiency is sacrificed in the interests of clarity.

Despite the search for the largest coefficient, it is still possible for the pivotal coefficient to be extremely small or zero if the coefficient matrix is singular or very ill-conditioned. Bearing in mind that the equations were scaled initially, an appropriate test of magnitude is $|a_{kk}| < 10^{-5}$. If this condition is satisfied at any stage of the elimination process, the problem is rejected. Rejection is indicated by setting the value of IFLAG to one, which may be detected by the calling program. If the partial pivoting is successful, however, the eliminations defined by Equations B.4 are performed, with the variable FACT being used to store the values of the factor ϕ . After testing the magnitude of the last diagonal coefficient ($a_{nn}^{(n)}$ in Equations B.5), the back substitutions defined in Equations B.6 are performed to find the required solutions.

Appendix E: Matlab Version of Quadratic Boundary Element Program for Plane Elastic Problems

The aim of this appendix is to present a Matlab version of the program BEM2EQ which is described in detail in its Fortran form in Chapter 6. Matlab is now very widely used by engineers for many purposes.

The Matlab version is as far as possible a literal translation of the Fortran. This means that the detailed explanations provided in Chapter 6 are also applicable to the Matlab script. On the other hand, no attempt is made to take advantage of special built-in features of Matlab such as simplified handling of matrices and vectors, and the solution of sets of linear equations.

Fortran is a compiler which converts the source code into machine code before computation starts. In contrast, Matlab is an interpreter which goes through the code line-by-line, translating and executing each line as it goes. Strictly speaking therefore, Matlab works with scripts rather than programs, although scripts are often loosely referred to as programs. A consequence of this difference between a compiler and an interpreter is that for large computationally intensive problems Fortran generally executes noticeably more quickly than Matlab.



ie business school

#1 EUROPEAN BUSINESS SCHOOL
FINANCIAL TIMES 2013

#gobeyond

MASTER IN MANAGEMENT

Because achieving your dreams is your greatest challenge. IE Business School's Master in Management taught in English, Spanish or bilingually, trains young high performance professionals at the beginning of their career through an innovative and stimulating program that will help them reach their full potential.

- Choose your area of specialization.
- Customize your master through the different options offered.
- Global Immersion Weeks in locations such as Rio de Janeiro, Shanghai or San Francisco.

Because you change, we change with you.

www.ie.edu/master-management | mim.admissions@ie.edu | f t in YouTube

The same function and variable names are used in the Fortran and Matlab programs: upper case names in Fortran become lower case in Matlab. The definitions of the names given in Some Program Variable Names at the beginnings of both Parts of the book are equally valid for both. DO loops in Fortran become “for” loops in Matlab, and functions are called in somewhat different ways. Also, STOP in Fortran is replaced by an error message to the computer screen in Matlab. The main difference, however, is to be seen in the details of the handling of input and output, although the same approach of inputting data from one pre-defined file and outputting to other files is adopted. In particular, the required input DATA files used in Fortran are unchanged, and the output files are effectively identical.

The Matlab version of BEM2EQ, stored as the file BEM2EQ.m, is as follows.

```
function BEM2EQ(varargin)
%
% PROGRAM FOR SOLVING TWO DIMENSIONAL ELASTIC STRESS ANALYSIS PROBLEMS
% BY THE BOUNDARY ELEMENT METHOD USING QUADRATIC ELEMENTS.
%
clear global; clear functions;
global fid5 fid6 fid7
%
shareddata2eq
fid5=fopen('DATA','r');
fid6=fopen('RESULTS','w+');
fid7=fopen('MESHRES','w+');
%
% DEFINE THE MAXIMUM PROBLEM SIZE PERMITTED BY THE ARRAY DIMENSIONS.
maxnel=250;
maxnnp=500;
maxnb=10;
maxnpc=20;
%
% INPUT THE PROBLEM TITLE AND TYPE, ALSO MATERIAL PROPERTIES.
intitle;
%
% INPUT AND GENERATE THE MESH DATA.
meshq;
%
% OUTPUT THE MESH DATA.
mshout;
%
```



```
% EVALUATE AND STORE VALUES OF THE SHAPE FUNCTIONS AND THEIR DERIVATIVES
% AT THE GAUSS POINTS AND NODES.

shape;

%

% INPUT, PROCESS AND OUTPUT THE BOUNDARY CONDITIONS.

bcs;

%

% FORM THE COEFFICIENT MATRIX AND APPLY THE BOUNDARY CONDITIONS.

frmtrx;

%

% SOLVE THE LINEAR EQUATIONS.

neqn=2*nnp;
[uv,iflag]=elimin(a,neqn);
if(iflag == 1)
    fprintf(fid6,['\n', ...
        'MATRIX ILL-CONDITIONING DETECTED IN EQUATION SOLVER','\n']);
    error('MATRIX ILL-CONDITIONING DETECTED IN EQUATION SOLVER');
end;

%

% OUTPUT THE NODAL DISPLACEMENTS, ELEMENT STRESSES AND FORCES ON THE
```

SMS from your computer

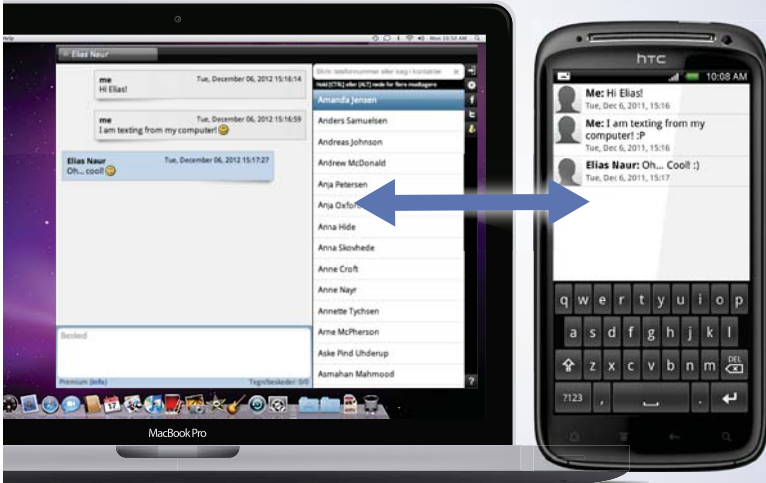
...Sync'd with your Android phone & number


FREE
30 days trial!

Go to

BrowserTexting.com

and start texting from
your computer!




BrowserTexting



```
% BOUNDARY SEGMENTS.
output;
%
end %program BEM2PQ

function intitle(varargin)
%
% SUBPROGRAM TO INPUT PROBLEM TITLE AND WHETHER PLANE
% STRESS OR PLANE STRAIN. ALSO THE MATERIAL PROPERTIES.
%
global nu e fid5 fid6
%
% INPUT THE PROBLEM TITLE.
title=fgetl(fid5);
fprintf(fid6,['QUADRATIC BOUNDARY ELEMENT SOLUTION FOR', ...
    ' TWO DIMENSIONAL ELASTIC PROBLEM','\n','\n','%s','\n'], title);
%
% INPUT THE PROBLEM CASE TYPE:
% 'STRESS' DEFINES PLANE STRESS
% 'STRAIN' DEFINES PLANE STRAIN
% THE DEFAULT IS PLANE STRESS.
pcase=fgetl(fid5);
fprintf(fid6,['\n','UNDER PLANE ','%s',' CONDITIONS','\n'],pcase);
%
% INPUT AND OUTPUT YOUNG'S MODULUS AND POISSON'S RATIO.
line=fgetl(fid5);
vec=str2num(line);
e=vec(1); nu=vec(2);
fprintf(fid6,['\n','YOUNGS MODULUS = ','%12.4e', ...
    ' POISSONS RATIO = ','%5.3f','\n'],e,nu);
%
% MODIFY PROPERTIES IF CASE IS ONE OF PLANE STRAIN.
strain='STRAIN';
if(strcmpi(pcase,strain) == 1)
    e=e/(1.-nu^2);
    nu=nu/(1.-nu);
end;
%
```



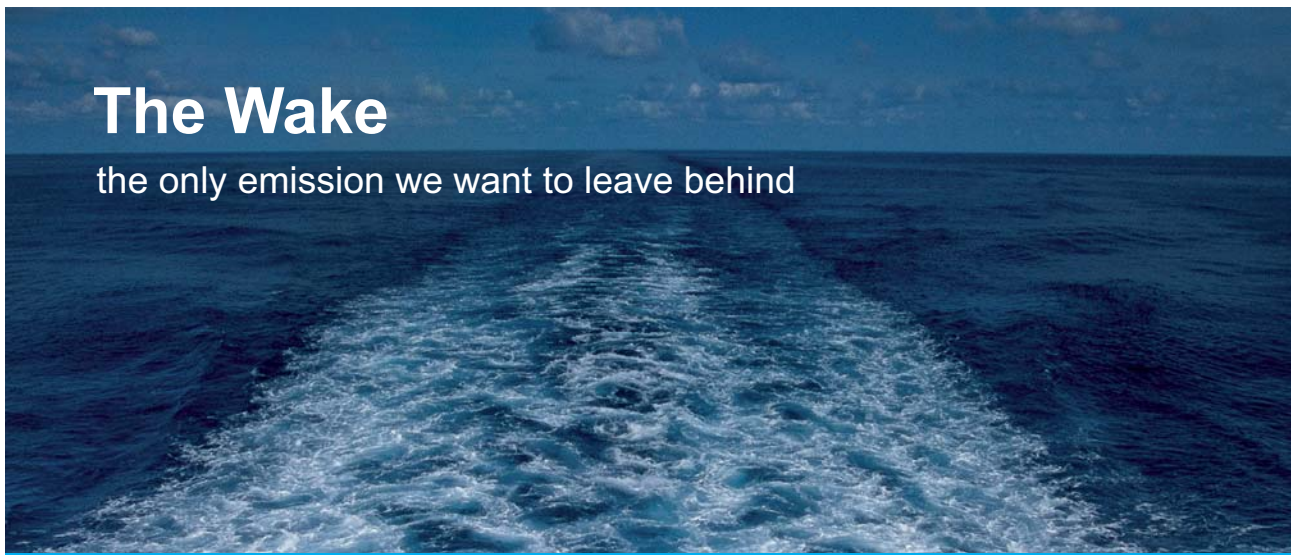
```

return;
end %function intitle


function meshq(varargin)
%
% SUBPROGRAM TO READ IN AND GENERATE THE GEOMETRIC DATA FOR A MESH OF
% QUADRATIC ELEMENTS.
%
global node maxl ynode xnode nnp nel neend m3 m1 angstore yeend xeend ...
isegelem mfirst isegend ysend xsend mlast maxnel nelb nsegb ...
nsegtot nbound maxnb fid5 fid6
%
% INPUT THE NUMBER OF SEPARATE BOUNDARIES.
line=fgetl(fid5);
nbound=str2num(line);
%
% TEST THE NUMBER OF BOUNDARIES.
if(nbound < 1 || nbound > maxnb)
    fprintf(fid6,['\n','NBOUND =','%4i', ...
        ' OUTSIDE PERMITTED RANGE 1 TO','%4i','\n'],nbound,maxnb);
    error('nbound error');
end;
%
% FOR EACH BOUNDARY IN TURN INPUT THE NUMBER OF SEGMENTS.
nel=0;
ieend=0;
nsegtot=0;
mmaxb=0;
for ibound=1:nbound; % each boundary in turn
    nelb(ibound)=0;
    mminb=mmaxb+1;
    line=fgetl(fid5);
    nsegb(ibound)=str2num(line);
    nsegtot=fix(nsegtot+nsegb(ibound));
%
% TEST THE NUMBER OF SEGMENTS.
if(nsegtot < 1 || nsegtot > maxnel)
    fprintf(fid6,['\n','NSEGTOT =','%6i', ...
        ' OUTSIDE PERMITTED RANGE 1 TO','%6i','\n'],nsegtot,maxnel);

```

```
error('nsegtot error');  
end;  
%  
% INPUT THE CARTESIAN GLOBAL COORDINATES OF THE END POINTS OF THE  
% SEGMENTS. TAKE THE END POINTS CONSECUTIVELY, KEEPING THE DOMAIN  
% TO THE LEFT OF THE DIRECTION OF NUMBERING.  
% Each line of input data is assumed to contain an arbitrary number  
% of pairs of x,y coordinates.  
isend=0;  
while isend < nsegb(ibound);  
    line=fgetl(fid5);  
    vec=str2num(line);  
    veclength=length(vec);  
    np=veclength/2;  
    for ip=1:np;  
        isend=isend+1;  
        xsend(isend)=vec(2*ip-1);  
        ysend(isend)=vec(2*ip);  
    end;  
end;
```




The Wake

the only emission we want to leave behind

Low-speed Engines Medium-speed Engines Turbochargers Propellers Propulsion Packages PrimeServ

The design of eco-friendly marine power and propulsion solutions is crucial for MAN Diesel & Turbo. Power competencies are offered with the world's largest engine programme – having outputs spanning from 450 to 87,220 kW per engine. Get up front! Find out more at www.mandieselturbo.com

Engineering the Future – since 1758.
MAN Diesel & Turbo



Click on the ad to read more

```
%
% DEFINE THE FIRST END POINT ON THE CURRENT BOUNDARY.
ieend=ieend+1;
xeend(ieend)=xsend(1);
yeend(ieend)=ysend(1);

%
% FOR EACH OF THE SEGMENTS (BETWEEN ENDS 1 AND 2, 2 AND 3, ETC.)
% INPUT THE RADIUS OF CURVATURE (+VE FOR CONVEX WITH CENTRE OF
% CURVATURE INSIDE DOMAIN, -VE FOR CONCAVE), THE NUMBER OF
% ELEMENTS IN THE SEGMENT, AND THE LENGTH RATIO BETWEEN SUCCESSIVE
% ELEMENTS IN THE DIRECTION OF NUMBERING.
isegmax=nsegtot;
isegmin=isegmax-nsegb(ibound)+1;
for iseg=isegmin:isegmax; % each segment in turn
    line=fgetl(fid5);
    vec=str2num(line);
    rseg=vec(1); nelseg=vec(2); ratseg=vec(3);

%
% FIND AND TEST THE NUMBER OF ELEMENTS SO FAR.
nel=fix(nel+nelseg);
nelb(ibound)=fix(nelb(ibound)+nelseg);
if(nel < 1 || nel > maxnel)
    fprintf(fid6,['\n','NEL =','%6i',' OUTSIDE PERMITTED' ...
        ' RANGE 1 TO','%6i','\n'],nel,maxnel);
    error('nel error');
end;

%
% FIRST AND LAST ELEMENTS ON CURRENT SEGMENT.
mlast(iseg)=fix(nel);
mfirst(iseg)=fix(nel-nelseg+1);
mmaxb=mmaxb+nelseg;

%
% COORDINATES OF THE FIRST END POINT OF THE SEGMENT.
isend=iseg-isegmin+1;
xfirst=xsend(isend);
yfirst=ysend(isend);

%
% COORDINATES OF THE LAST END POINT OF THE SEGMENT.
isend=isend+1;
```

```

    if(iseq == isegmax)
        isend=1;
    end;
    xlast=xsend(isend);
    ylast=ysend(isend);
%
% GENERATE ELEMENT DATA FOR A STRAIGHT SEGMENT.
    if(rseg == 0.)
%
% DEFINE THE ELEMENT END POINT COORDINATES ON THE SEGMENT.
        for m=1:nelseg; % each element in turn
            ieend=ieend+1;
            isegend(ieend)=fix(iseq);
            if(ratseg == 1.)
                xeend(ieend)=xfirst+(xlast-xfirst)*(m)/(nelseg);
                yeend(ieend)=yfirst+(ylast-yfirst)*(m)/(nelseg);
            end;
            if(ratseg ~= 1.)
                xeend(ieend)=xfirst+(xlast-xfirst)*(1.-ratseg^m)/ ...
                    (1.-ratseg^nelseg);
                yeend(ieend)=yfirst+(ylast-yfirst)*(1.-ratseg^m)/ ...
                    (1.-ratseg^nelseg);
            end;
        end; % each element in turn
%
% DEFINE THE ELEMENT NODES AND COORDINATES.
        ieend=ieend-nelseg;
        for ielseg=1:nelseg; % each element in turn
            ieend=ieend+1;
            m=mfirst(iseq)+ielseg-1;
            i1=2*m-1;
            i2=i1+1;
            i3=i1+2;
            if(iseq == isegmax && ielseg == nelseg)
                i3=node(mminb,1);
            end;
            node(m,1)=fix(i1);
            node(m,2)=fix(i2);
            node(m,3)=fix(i3);
            isegelem(m)=fix(iseq);
        end;
    end;
end;

```

```
if (iseg == isegmin && ielseg == 1)
    xnode(i1) = xeend(ieend-1);
    ynode(i1) = yeend(ieend-1);
end;
xnode(i3) = xeend(ieend);
ynode(i3) = yeend(ieend);
xnode(i2) = 0.5 * (xnode(i1) + xnode(i3));
ynode(i2) = 0.5 * (ynode(i1) + ynode(i3));

%
% STORE THE NUMBERS OF THE ADJACENT ELEMENTS.
    m1(m) = fix(m-1);
    m3(m) = fix(m+1);
end; % each element in turn
end;

%
% GENERATE ELEMENT DATA FOR A SEGMENT IN THE FORM OF A CIRCULAR ARC.
    if (rseg ~= 0.)
%
% LOCATE THE CENTRE OF THE ARC.
        xmid = (xfirst + xlast) / 2.;
```

TURN TO THE EXPERTS FOR **SUBSCRIBE** CONSULTANCY

Subscribe is one of the leading companies in Europe when it comes to innovation and business development within subscription businesses.

We innovate new subscription business models or improve existing ones. We do business reviews of existing subscription businesses and we develop acquisition and retention strategies.

Learn more at [linkedin.com/company/subscribe](https://www.linkedin.com/company/subscribe) or contact
Managing Director Morten Suhr Hansen at mha@subscribe.dk

SUBSCR✓**BE** - to the future



```

ymid=(yfirst+ylast)/2.;
alseg=sqrt((xlast-xfirst)^2+(ylast-yfirst)^2);
alperp2=rseg^2-(alseg/2.)^2;
if(abs(alperp2) < 1.0e-6*rseg^2)
    alperp2=0.;
end;
if(alperp2 < -1.0e-6*rseg^2)
    fprintf(fid6,['\n','DATA ERROR FOR SEGMENT NUMBER','%6i', ...
        '\n','NOT POSSIBLE TO CREATE A CIRCULAR ARC','\n'],iseg);
    error('circular arc error');
end;
alperp=sqrt(alperp2);
uvflx=(xlast-xfirst)/alseg;
uvfly=(ylast-yfirst)/alseg;
fact=1.;
if(rseg < 0.)
    fact=-1.;
end;
xcen=xmid-alperp*uvfly*fact;
ycen=ymid+alperp*uvflx*fact;

%
% FIND THE ANGLE SUBTENDED THERE BY THE SEGMENT.
if(alperp ~= 0.)
    angseg=2.*atan(alseg*0.5/alperp);
end;
if(alperp == 0.)
    angseg=pi;
end;

%
% DEFINE THE ELEMENT END POINT COORDINATES ON THE SEGMENT.
angfir=atan2(yfirst-ycen,xfirst-xcen);
angstore(ieend)=0.;
for m=1:nelseg; % each element in turn
    ieend=ieend+1;
    isegend(ieend)=fix(iseg);
    if(ratseg == 1.)
        ang=angseg*(m)/(nelseg);
    end;
    if(ratseg ~= 1.)
        ang=angseg*(1.-ratseg^m)/(1.-ratseg^nelseg);
    end;
end;

```

```

end;
if(rseg < 0.)
    ang=-ang;
end;
xeend(ieend)=xcent+abs(rseg)*cos(angfir+ang);
yeend(ieend)=ycent+abs(rseg)*sin(angfir+ang);
angstore(ieend)=ang;
end; % each element in turn

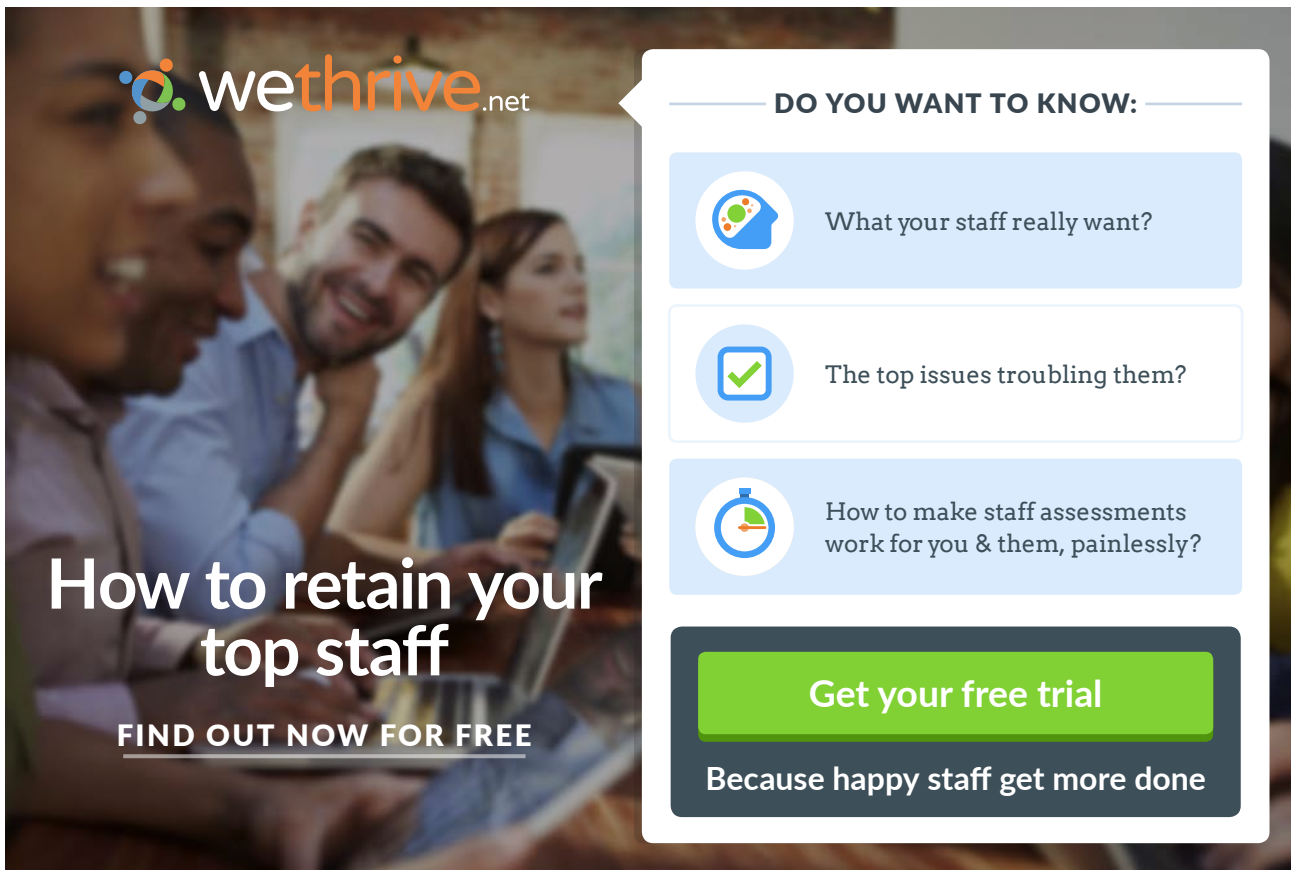
%
% DEFINE THE ELEMENT NODES AND COORDINATES.
ieend=ieend-nelseg;
for ielseg=1:nelseg; % each element in turn
    ieend=ieend+1;
    m=mfirst(iseg)+ielseg-1;
    i1=2*m-1;
    i2=i1+1;
    i3=i1+2;
    if(iseg == isegmax && ielseg == nelseg)
        i3=node(mminb,1);
    end;
    node(m,1)=fix(i1);
    node(m,2)=fix(i2);
    node(m,3)=fix(i3);
    isegelem(m)=fix(iseg);
    if(iseg == isegmin && ielseg == 1)
        xnode(i1)=xeend(ieend-1);
        ynode(i1)=yeend(ieend-1);
    end;
    xnode(i3)=xeend(ieend);
    ynode(i3)=yeend(ieend);
    ang=0.5*(angstore(ieend-1)+angstore(ieend));
    xnode(i2)=xcent+abs(rseg)*cos(angfir+ang);
    ynode(i2)=ycent+abs(rseg)*sin(angfir+ang);

%
% STORE THE NUMBERS OF THE ADJACENT ELEMENTS.
m1(m)=fix(m-1);
m3(m)=fix(m+1);
end; % each element in turn
end;

%
```

```

end; % each segment in turn
%
% ADJACENT ELEMENTS FOR END ELEMENTS OF THE BOUNDARY.
m1(mminb)=fix(mmaxb);
m3(mmaxb)=fix(mminb);
end; % each boundary in turn
neend=fix(ieend);
nnp=fix(nel*2);
%
% DETERMINE THE MAXIMUM DIMENSION OF THE SOLUTION DOMAIN.
maxl=0.;
for i=1:nnp; % each node in turn
    for j=1:nnp; % each other node in turn
        dist=sqrt((xnode(i)-xnode(j))^2+(ynode(i)-ynode(j))^2);
        if(dist > maxl)
            maxl=dist;
        end;
    end; % each other node in turn
end; % each node in turn
%
```



wethrive.net

How to retain your top staff

FIND OUT NOW FOR FREE

DO YOU WANT TO KNOW:

- What your staff really want?
- The top issues troubling them?
- How to make staff assessments work for you & them, painlessly?

Get your free trial

Because happy staff get more done


```

return;
end %function meshq

function mshout(varargin)
%
% SUBPROGRAM TO WRITE OUT THE MESH DATA.
%
global node maxl ynode xnode nnp nel fid7
%
% OUTPUT THE NUMBERS OF ELEMENTS AND NODES, ALSO THE NODAL
% COORDINATES.
fprintf(fid7,['GEOMETRIC DATA FOR THE MESH','\n','\n', ...
    '          NUMBER OF ELEMENTS =','%6i','\n','\n', ...
    '          NUMBER OF NODAL POINTS =','%6i','\n','\n', ...
    'COORDINATES OF THE NODAL POINTS','\n','\n', ...
    '      I      X      Y      ', ...
    '      I      X      Y      '],nel, nnp);
for i=1:nnp;
    if(mod(i,2) == 1);
        fprintf(fid7,['\n','%6i','%12.4e','%12.4e'], ...
            i,xnode(i),ynode(i));
    end;
    if(mod(i,2) == 0);
        fprintf(fid7,['%6i','%12.4e','%12.4e'], ...
            i,xnode(i),ynode(i));
    end;
end;
%
% OUTPUT THE ELEMENT NODE NUMBERS.
fprintf(fid7,['\n','\n','ELEMENT NODE NUMBERS','\n','\n', ...
    '      M    ND1  ND2  ND3', ...
    '      M    ND1  ND2  ND3']);
for m=1:nel;
    if(mod(m,2) == 1)
        fprintf(fid7,['\n','      ','%5i'],m);
    end;
    if(mod(m,2) == 0)
        fprintf(fid7,['      ','%5i'],m);
    end;
end;

```


```

    for in=1:3;
        fprintf(fid7,'%5i',node(m,in));
    end;
end;
%
% SCALE THE NODAL POINT COORDINATES.
for i=1:nnp; % each node in turn
    xnode(i)=xnode(i)/max1;
    ynode(i)=ynode(i)/max1;
end; % each node in turn
%
return;
end %function mshout

function bcs(varargin)
%
% SUBPROGRAM TO INPUT, PROCESS AND OUTPUT THE BOUNDARY CONDITIONS.
%
global nodepc idirpc nnp nbcpc nbcu e estore node tmy ty tmx tx ...
    ibcn ibce max1 velem v uelem u m3 m1 nel mlast mfirst signsseg ...
    signnseg signn signn unmx unmy nsegtot isegbc nbcs vseg useg ...
    maxnpc maxnel nbct ungy ungx fid5 fid6
%
% FIRST FIND THE UNIT NORMAL COMPONENTS AT THE ELEMENT NODES.
for m=1:nel; % each element in turn
    for in=1:3; % each element node in turn
        igauss=in+10;
        it=1;
        ic=1;
        jacobi(m,igauss,it,ic);
        unmx(m,in)=ungx;
        unmy(m,in)=ungy;
    end; % each element node in turn
end; % each element in turn
%
% INPUT THE NUMBERS OF SEGMENTS SUBJECT TO EACH TYPE OF BOUNDARY
% CONDITION. ALSO THE NUMBER OF POINT CONSTRAINTS.
%     NBCU - PRESCRIBED DISPLACEMENTS.
%     NBKS - NON-ZERO PRESCRIBED STRESSES.

```

```
% ANY SEGMENT NOT INCLUDED IS ASSUMED TO BE SUBJECT TO ZERO
% STRESSES.
%     NBCPC - NODAL POINT DISPLACEMENT CONSTRAINTS.
%
line=fgetl(fid5);
vec=str2num(line);
nbcu=vec(1); nbcs=vec(2); nbcpc=vec(3);
%
% TEST THESE BOUNDARY CONDITION NUMBERS.
nbct=fix(nbcu+nbcs);
if(nbcu < 0 || nbcu > maxnel || nbcs < 0 || nbcs > maxnel ...
    || nbct < 0 || nbct > maxnel)
    fprintf(fid6,['\n','NBCU =','%6i','    NBCS =','%6i','\n', ...
        '    NBCT =','%6i','    OUTSIDE PERMITTED RANGE 0 TO', ...
        '%6i','\n'],nbcu,nbcs,nbct,maxnel);
error('numbers of boundary conditions error');
end;
if(nbcpc < 0 || nbcpc > maxnpc)
    fprintf(fid6,['\n','NBCPC =','%4i', ...
        '    OUTSIDE PERMITTED RANGE 0 TO','%3i','\n'],nbcpc,maxnpc);
```



Struggling to get interviews?

Professional CV consulting & writing assistance from leading job experts in the UK.

[Visit site](#)



Click on the ad to read more

```

error('number of point constraints error');
end;
%
% INITIALISE THE BOUNDARY CONDITION STORAGE ARRAYS.
for m=1:nel; % each element in turn
    ibce(m)=2;
    for in=1:3; % each element node in turn
        signn(m,in)=0.;
        sigsn(m,in)=0.;
        tmx(m,in)=0.;
        tmy(m,in)=0.;
    end; % each element node in turn
end; % each element in turn
%
% INPUT, STORE AND OUTPUT THE PRESCRIBED DISPLACEMENT CONDITIONS.
if(nbcu > 0)
    for ibcu=1:nbcu;
        line=fgetl(fid5);
        vec=str2num(line);
        isegbc(ibcu)=vec(1); useg(ibcu)=vec(2); vseg(ibcu)=vec(3);
    end;
    fprintf(fid6,['\n','PRESCRIBED DISPLACEMENT BOUNDARY CONDITIONS','\n']);
    for ibcu=1:nbcu; % each segment with prescribed displacements
        iseg=isegbc(ibcu);
        if(iseg < 1 || iseg > nsegtot)
            fprintf(fid6,['\n','ISEG = ','%6i',' OUTSIDE PERMITTED RANGE' ...
                ' 1 TO','%6i','\n'],iseg,nsegtot);
            error('iseg error');
        end;
        for m=mfirst(iseg):mlast(iseg); % each element on current segment
            ibce(m)=1;
            uelem(m)=useg(ibcu);
            velem(m)=vseg(ibcu);
        end; % each element on current segment
    end;
    fprintf(fid6,['\n','U = ','%12.4e',' V = ','%12.4e', ...
        ' ON ELEMENTS','%4i',' TO ','%4i','\n'], ...
        useg(ibcu),vseg(ibcu),mfirst(iseg),mlast(iseg));
    end; % each segment with prescribed displacements;
end;
%
```

```
% INPUT, STORE AND OUTPUT THE PRESCRIBED STRESS CONDITIONS.
if(nbcs > 0)
    for ibcs=1:nbcs;
        line=fgetl(fid5);
        vec=str2num(line);
        isegbc(ibcs)=vec(1); signnseg(ibcs)=vec(2); sigsnseg(ibcs)=vec(3);
    end;
    fprintf(fid6,['\n','PRESCRIBED STRESS BOUNDARY CONDITIONS','\n']);
    for ibcs=1:nbcs; % each segment with prescribed stresses
        iseg=isegbc(ibcs);
        if(iseg < 1 || iseg > nsegtot)
            fprintf(fid6,['\n','ISEG = ','%6i',' OUTSIDE PERMITTED RANGE' ...
                ' 1 TO','%6i','\n'],iseg,nsegtot);
            error('iseg error');
        end;
        for m=mfirst(iseg):mlast(iseg); % each element on current segment
            ibce(m)=2;
        %
        % FIND THE TRACTIONS AT THE ELEMENT NODES FROM THE PRESCRIBED STRESSES.
        % ALSO STORE THE STRESSES AT THE ELEMENT NODES.
            for in=1:3; % each element node in turn
                tmx(m,in)=signnseg(ibcs)*unmx(m,in)-sigsnseg(ibcs)*unmy(m,in);
                tmy(m,in)=signnseg(ibcs)*unmy(m,in)+sigsnseg(ibcs)*unmx(m,in);
                signn(m,in)=signnseg(ibcs);
                sigsn(m,in)=sigsnseg(ibcs);
            end; % each element node in turn
        end; % each element on current segment
        %
        fprintf(fid6,['\n','SIGNN = ','%12.4e',' SIGSN = ','%12.4e', ...
            ' ON ELEMENTS','%4i',' TO','%4i','\n'], ...
            signnseg(ibcs),sigsnseg(ibcs),mfirst(iseg),mlast(iseg));
    end; % each segment with prescribed stresses
end;

%
% ASSEMBLE THE VECTOR OF KNOWN VARIABLES, APPLYING SCALING.
% FIRST INITIALISE THE NODAL DISPLACEMENTS AND TRACTIONS.
for i=1:nnp; % each node in turn
    u(i)=0.;
    v(i)=0.;
    tx(i)=0.;
    ty(i)=0.;
```

```
end; % each node in turn
for m=1:nel; % each element in turn
    for in=1:3; % each element node in turn
        i=node(m,in);
        ibcn(i)=fix(ibce(m));
    %
    % CHECK THE CONDITION APPLIED TO THE ADJACENT ELEMENT FOR AN ELEMENT
    % end NODE. IF THE CONDITION IS PRESCRIBED DISPLACEMENTS BUT THE
    % EXISTING CONDITION AT THE NODE IS NOT, THEN IMPOSE PRESCRIBED
    % DISPLACEMENTS.
    madj=m;
    if(in == 1)
        madj=m1(m);
    end;
    if(in == 3)
        madj=m3(m);
    end;
    if(ibce(madj) == 1 && ibcn(i) == 2)
        ibcn(i)=1;
    end;
```

The advertisement features a background image of a person running on a path, overlaid with technical diagrams of a shoe's sole and foot mechanics. The GaiTEYE logo is in the top left, and a yellow call-to-action button is in the bottom right.

gaiteye®
Challenge the way we run

**EXPERIENCE THE POWER OF
FULL ENGAGEMENT...**

**RUN FASTER.
RUN LONGER..
RUN EASIER...**

**READ MORE & PRE-ORDER TODAY
WWW.GAITEYE.COM**

```
%
% STORE KNOWN VARIABLES.
    if(ibcn(i) == 1)
        if(ibce(m) == 1)
            if(u(i) == 0.)
                u(i)=uelem(m)/maxl;
            end;
            if(v(i) == 0.)
                v(i)=velem(m)/maxl;
            end;
            if(u(i) ~= 0.)
                u(i)=0.5*(u(i)+uelem(m)/maxl);
            end;
            if(v(i) ~= 0.)
                v(i)=0.5*(v(i)+velem(m)/maxl);
            end;
        end;
    end;
    if(ibce(m) == 2 && ibcn(i) == 2)
        if(tx(i) == 0.)
            tx(i)=tmx(m,in)/e;
        end;
        if(ty(i) == 0.)
            ty(i)=tmy(m,in)/e;
        end;
        if(tx(i) ~= 0.)
            tx(i)=0.5*(tx(i)+tmx(m,in)/e);
        end;
        if(ty(i) ~= 0.)
            ty(i)=0.5*(ty(i)+tmy(m,in)/e);
        end;
    end;
end; % each element node in turn
end; % each element in turn
%
% SCALE YOUNG'S MODULUS.
estore=e;
e=1.;
%
```

```
% INPUT, STORE AND OUTPUT NODAL POINT DISPLACEMENT CONSTRAINTS.
% IDIRPC STORES DIRECTIONS OF THE CONSTRAINTS - 1 FOR X, 2 FOR Y.
% SUCH CONSTRAINTS SHOULD NOT REQUIRE POINT FORCES TO MAINTAIN THEM.
if(nbcu == 0 && nbcpc < 3)
    fprintf(fid6,['\n','INSUFFICIENT DISPLACEMENT BOUNDARY CONDITIONS', ...
        ' OR POINT CONSTRAINTS','\n']);
    error('displacement constraint error');
end;
if(nbcpc > 0)
    for ibcpc=1:nbcpc;
        line=fgetl(fid5);
        vec=str2num(line);
        nodepc(IBCPC)=vec(1); idirpc(IBCPC)=vec(2);
    end;
    for ibcpc=1:nbcpc; % each point constraint in turn
        if(nodepc(IBCPC) < 0 || nodepc(IBCPC) > nnp)
            fprintf(fid6,['\n','POINT CONSTRAINT NODE','%4i', ...
                ' OUTSIDE RANGE 0 TO ', '%4i', '\n'],nodepc(IBCPC),nnp);
            error('point constraint node error');
        end;
        if(idirpc(IBCPC) ~= 1 && idirpc(IBCPC) ~= 2)
            fprintf(fid6,['\n','POINT CONSTRAINT DIRECTION','%3i', ...
                ' FOR NODE','%4i', ' OUTSIDE RANGE 1 TO 2', '\n'], ...
                idirpc(IBCPC),nodepc(IBCPC));
            error('point constraint direction error');
        end;
        if(idirpc(IBCPC) == 1)
            fprintf(fid6,['\n','NODE','%4i', ' CONSTRAINED WITH U = 0', ...
                '\n'],nodepc(IBCPC));
        end;
        if(idirpc(IBCPC) == 2)
            fprintf(fid6,['\n','NODE','%4i', ' CONSTRAINED WITH V = 0', ...
                '\n'],nodepc(IBCPC));
        end;
    end; % each point constraint in turn
end;
%
return;
end %function bcs
```



```
function frmtrx(varargin)
%
% SUBPROGRAM TO FORM THE COEFFICIENT MATRIX AND RIGHT HAND SIDE VECTOR,
% MODIFIED TO SUIT THE BOUNDARY CONDITIONS.
%
global a idirpc nodepc nbcpc node v arowy u arowx ibcn estore tmy ...
    browy tmx browx ibce nel nnp
%
% DEFINE THE NUMBER OF COLUMNS IN THE EXTENDED COEFFICIENT MATRIX A.
jmax=2*nnp+1;
%
% FORM THE COEFFICIENT MATRIX A, AND THE RIGHT HAND SIDE VECTOR B*T.
for i=1:nnp; % take each node in turn as P
%
% INITIALISE THE TWO ROWS OF MATRIX A CORRESPONDING TO THE NODE.
for j=1:jmax; % each pair of coefficients in turn
    a(2*i-1,j)=0.;
    a(2*i,j)=0.;
end; % each pair of coefficients in turn
%
```

This e-book
is made with
SetaPDF



PDF components for PHP developers

www.setasign.com



```
% INITIALISE THE ELEMENT CONTRIBUTIONS TO THE CURRENT ROWS OF THE
% A AND B MATRICES.
for m=1:nel; % each element in turn
    for in=1:3; % each element node in turn
        arowx(m,2*in-1)=0.;
        arowx(m,2*in)=0.;
        arowy(m,2*in-1)=0.;
        arowy(m,2*in)=0.;
        browx(m,2*in-1)=0.;
        browx(m,2*in)=0.;
        browy(m,2*in-1)=0.;
        browy(m,2*in)=0.;
    end; % each element node in turn
end; % each element in turn

%
% SET UP THE LOOP TO INTEGRATE OVER EACH ELEMENT IN TURN.
for m=1:nel; % each element in turn
%
% INTEGRATE THE KERNEL PRODUCTS OVER THE CURRENT ELEMENT.
    intker(i,m);
end; % each element in turn
%
% EVALUATE THE COEFFICIENTS AT THE DIAGONAL OF MATRIX A.
aiixx=0.;
aiixy=0.;
aiiyx=0.;
aiiyy=0.;
for m=1:nel; % each element in turn
    for in=1:3; % each element node in turn
        aiixx=aiixx-arowx(m,2*in-1);
        aiixy=aiixy-arowx(m,2*in);
        aiiyx=aiiyx-arowy(m,2*in-1);
        aiiyy=aiiyy-arowy(m,2*in);
    end; % each element node in turn
end; % each element in turn
if(ibcn(i) == 2)
    a(2*i-1,2*i-1)=aiixx;
    a(2*i-1,2*i)=aiixy;
    a(2*i,2*i-1)=aiiyx;
    a(2*i,2*i)=aiiyy;
end;
```

```
%
% INITIALISE THE B*T VECTOR COEFFICIENTS.
btx=0.;
bty=0.;
if(ibcn(i) == 1)
    btx=-aiixx*u(i)-aiixy*v(i);
    bty=-aiiyx*u(i)-aiiyy*v(i);
end;

%
% APPLY THE BOUNDARY CONDITIONS TO THE CURRENT ROWS OF A AND B, BY
% CONSIDERING EACH ELEMENT IN TURN.
for m=1:nel; % each element in turn
    for in=1:3; % each element node in turn
        j=node(m,in);

%
% IF DISPLACEMENTS ARE PRESCRIBED OVER THE ELEMENT,
% INTERCHANGE THE A AND B COEFFICIENTS.
        if(ibce(m) == 1)
            a(2*i-1,2*j-1)=a(2*i-1,2*j-1)-browx(m,2*in-1);
            a(2*i-1,2*j)=a(2*i-1,2*j)-browx(m,2*in);
            a(2*i,2*j-1)=a(2*i,2*j-1)-browy(m,2*in-1);
            a(2*i,2*j)=a(2*i,2*j)-browy(m,2*in);
            btx=btx-arowx(m,2*in-1)*u(j)-arowx(m,2*in)*v(j);
            bty=bty-arowy(m,2*in-1)*u(j)-arowy(m,2*in)*v(j);
        end;

%
% IF STRESSES ARE PRESCRIBED OVER THE ELEMENT, STORE THE A MATRIX
% COEFFICIENTS, EXCEPT AT A CORNER NODE WHERE PRESCRIBED DISPLACEMENTS
% HAVE BEEN IMPOSED.
        if(ibce(m) == 2)
            btx=btx+(browx(m,2*in-1)*tmx(m,in)+browx(m,2*in)*tmy(m,in))/
estore;
            bty=bty+(browy(m,2*in-1)*tmx(m,in)+browy(m,2*in)*tmy(m,in))/
estore;

        if(ibcn(j) == 2)
            a(2*i-1,2*j-1)=a(2*i-1,2*j-1)+arowx(m,2*in-1);
            a(2*i-1,2*j)=a(2*i-1,2*j)+arowx(m,2*in);
            a(2*i,2*j-1)=a(2*i,2*j-1)+arowy(m,2*in-1);
            a(2*i,2*j)=a(2*i,2*j)+arowy(m,2*in);
```

```

end;
if(ibcn(j) == 1)
    btx=btx-arowx(m,2*in-1)*u(j)-arowx(m,2*in)*v(j);
    bty=bty-arowy(m,2*in-1)*u(j)-arowy(m,2*in)*v(j);
end;
end;
%
    end; % each element node in turn
end; % each element in turn
%
% STORE THE B*T COEFFICIENTS AS EXTENSIONS OF MATRIX A.
a(2*i-1,jmax)=btx;
a(2*i,jmax)=bty;
end; % take each node in turn as P
%
% APPLY NODAL POINT DISPLACEMENT CONSTRAINTS.
if(nbcpc > 0)
    for ibcpc=1:nbcpc; % each point constraint in turn
        irow=2*(nodepc(ibcpc)-1)+idirpc(ibcpc);
        for j=1:jmax; % each column in turn

```



**YOU THINK.
YOU CAN WORK
AT RMB**

 **RAND
MERCHANT
BANK**
A division of FirstRand Bank Limited
Traditional values. Innovative ideas.

Rand Merchant Bank uses good business to create a better world, which is one of the reasons that the country's top talent chooses to work at RMB. For more information visit us at www.rmb.co.za

Thinking that can change your world

Rand Merchant Bank is an Authorised Financial Services Provider



Click on the ad to read more

```
        a(irow,j)=0.;
    end; % each column in turn
    a(irow,irow)=1.;
    end; % each point constraint in turn
end;
%
return;
end %function frmtrx
```

```
function shape(varargin)
%
% SUBPROGRAM TO EVALUATE AND STORE VALUES OF THE SHAPE FUNCTIONS AND
% THEIR DERIVATIVES, AT THE GAUSS POINTS AND NODES.
%
global sd sdl sfl egl ngauss sf zg wgl wg
%
% STORE APPROPRIATE COORDINATES AND WEIGHT FACTORS FOR NORMAL GAUSSIAN
% QUADRATURE IN ARRAYS XG AND CG, ALSO THOSE FOR LOGARITHMIC FUNCTION
% INTEGRATION IN XGL AND CGL.
ngauss=8;
zg(1)=-0.9602898564;
zg(2)=-0.7966664774;
zg(3)=-0.5255324099;
zg(4)=-0.1834346424;
zg(5)=-zg(4);
zg(6)=-zg(3);
zg(7)=-zg(2);
zg(8)=-zg(1);
wg(1)=0.1012285362;
wg(2)=0.2223810344;
wg(3)=0.3137066458;
wg(4)=0.3626837833;
wg(5)=wg(4);
wg(6)=wg(3);
wg(7)=wg(2);
wg(8)=wg(1);
egl(1)=0.013320244;
egl(2)=0.079750429;
```

```

egl(3)=0.197871029;
egl(4)=0.354153994;
egl(5)=0.529458575;
egl(6)=0.701814530;
egl(7)=0.849379320;
egl(8)=0.953326450;
wgl(1)=0.164416605;
wgl(2)=0.237525610;
wgl(3)=0.226841984;
wgl(4)=0.175754079;
wgl(5)=0.112924030;
wgl(6)=0.057872211;
wgl(7)=0.020979074;
wgl(8)=0.003686407;
%
%  NORMAL GAUSSIAN QUADRATURE.
for igauss=1:ngauss; % each gauss point in turn
    zeta=zg(igauss);
    sf(1,igauss)=0.5*zeta*(zeta-1.);
    sf(2,igauss)=1.-zeta^2;
    sf(3,igauss)=0.5*zeta*(zeta+1.);
    sd(1,igauss)=zeta-0.5;
    sd(2,igauss)=-2.*zeta;
    sd(3,igauss)=zeta+0.5;
end; % each gauss point in turn
%
%  FOUR CASES OF LOGARITHMIC GAUSSIAN QUADRATURE TO CONSIDER.
%      IC=1 - INTEGRATION OVER WHOLE ELEMENT FROM FIRST TO THIRD NODE.
%      IC=2 - INTEGRATION OVER HALF ELEMENT FROM SECOND TO THIRD NODE.
%      IC=3 - INTEGRATION OVER HALF ELEMENT FROM SECOND TO FIRST NODE.
%      IC=4 - INTEGRATION OVER WHOLE ELEMENT FROM THIRD TO FIRST NODE.
%
for ic=1:4; % each case in turn
    for igauss=1:ngauss; % each gauss point in turn
        eta=egl(igauss);
        if(ic == 1)
            zeta=2.*eta-1.;
        end;
        if(ic == 2)
            zeta=eta;

```

```
end;
if(ic == 3)
    zeta=-eta;
end;
if(ic == 4)
    zeta=1.-2.*eta;
end;
sfl(ic,1,igauss)=0.5*zeta*(zeta-1.);
sfl(ic,2,igauss)=1.-zeta^2;
sfl(ic,3,igauss)=0.5*zeta*(zeta+1.);
sdl(ic,1,igauss)=zeta-0.5;
sdl(ic,2,igauss)=-2.*zeta;
sdl(ic,3,igauss)=zeta+0.5;
end; % each gauss point in turn
end; % each case in turn
%
% SHAPE FUNCTION DERIVATIVES AT THE NODES, STORED AS THOUGH THEY
% ARE FOR GAUSS POINTS NUMBERED 11, 12 AND 13.
for igauss=11:13; % each element node in turn
    if(igauss == 11)
```



Discover the truth at www.deloitte.ca/careers

Deloitte.

© Deloitte & Touche LLP and affiliated entities.



Click on the ad to read more

```

        zeta=-1.;
    end;
    if(igauss == 12)
        zeta=0.;
    end;
    if(igauss == 13)
        zeta=1.;
    end;
    sd(1,igauss)=zeta-0.5;
    sd(2,igauss)=-2.*zeta;
    sd(3,igauss)=zeta+0.5;
end; % each element node in turn
%
return;
end %function shape

```

```

function intker(i,m)
%
% SUBPROGRAM TO INTEGRATE THE KERNEL PRODUCTS FOR A PARTICULAR FORCE
% POINT P (INDICATED BY NODE NUMBER I) OVER A PARTICULAR ELEMENT
% (INDICATED BY ARGUMENT M) BY GAUSSIAN QUADRATURE.
%
global node jacob wgl browy browx sfl ngauss bk2yy wg bk2yx bk2xy ...
    bk2xx sf bkyy bkxy bkxx akyy arowy akyx akxy arowx akxx ...
    ungy ungx ynode xnode e nu
%
% CONSTANT IN DISPLACEMENT KERNELS MULTIPLYING THE LOGARITHMIC TERM.
cl=(1.+nu)*(3.-nu)/(4.*pi*e);
%
% COORDINATES OF POINT P.
xp=xnode(i);
yp=ynode(i);
%
% SET UP THE ELEMENT NODE LOOP.
for in=1:3; % each element node in turn
    j=node(m,in);
%

```



```
% IF P IS NOT THE CURRENT ELEMENT NODE, USE NORMAL GAUSSIAN QUADRATURE.
if(i ~= j)
    for igauss=1:ngauss; % each gauss point in turn
    %
    % EVALUATE JACOBIAN AND UNIT NORMAL VECTOR COMPONENTS, ALSO THE KERNELS
    % AT THE PARTICULAR GAUSS POINT FOR NORMAL QUADRATURE OVER THE WHOLE
    % ELEMENT.
        it=1;
        ic=1;
        jacobi(m,igauss,it,ic);
        kernel(xp,yp,m,igauss,ungx,ungy);
    %
    % ACCUMULATE THE INTEGRALS.
        sfn=sf(in,igauss);
        arowx(m,2*in-1)=arowx(m,2*in-1)+wg(igauss)*akxx*sfn*jacob;
        arowx(m,2*in)=arowx(m,2*in)+wg(igauss)*akxy*sfn*jacob;
        arowy(m,2*in-1)=arowy(m,2*in-1)+wg(igauss)*akyx*sfn*jacob;
        arowy(m,2*in)=arowy(m,2*in)+wg(igauss)*akyy*sfn*jacob;
        browx(m,2*in-1)=browx(m,2*in-1)+wg(igauss)*bkxx*sfn*jacob;
        browx(m,2*in)=browx(m,2*in)+wg(igauss)*bkxy*sfn*jacob;
        browy(m,2*in-1)=browy(m,2*in-1)+wg(igauss)*bkxy*sfn*jacob;
        browy(m,2*in)=browy(m,2*in)+wg(igauss)*bkyy*sfn*jacob;
    end; % each gauss point in turn
end;
%
% IF P IS THE CURRENT ELEMENT NODE, SOME LOGARITHMIC QUADRATURE
% IS REQUIRED.
if(i == j)
    %
    % P AT THE FIRST NODE OF THE ELEMENT.
    if(in == 1)
    %
    % TERMS INVOLVING NORMAL QUADRATURE.
        for igauss=1:ngauss; % each gauss point in turn
            it=1;
            ic=1;
            jacobi(m,igauss,it,ic);
            kern2(m,in,igauss);
            sfn=sf(in,igauss);
            browx(m,2*in-1)=browx(m,2*in-1)+wg(igauss)*bk2xx*sfn*jacob;
```

```

    browx(m,2*in)=browx(m,2*in)+wg(igauss)*bk2xy*sfn*jacob;
    browy(m,2*in-1)=browy(m,2*in-1)+wg(igauss)*bk2yx*sfn*jacob;
    browy(m,2*in)=browy(m,2*in)+wg(igauss)*bk2yy*sfn*jacob;
end; % each gauss point in turn

%
% TERMS INVOLVING LOGARITHMIC QUADRATURE.
for igauss=1:ngauss; % each gauss point in turn
    it=2;
    ic=1;
    jacobi(m,igauss,it,ic);
    sfn=sfl(ic,in,igauss);
    dzde=2.;
    browx(m,2*in-1)=browx(m,2*in-1)+wgl(igauss)*cl*sfn*jacob*dzde;
    browy(m,2*in)=browy(m,2*in)+wgl(igauss)*cl*sfn*jacob*dzde;
end; % each gauss point in turn
end;

%
% P AT THE SECOND NODE OF THE ELEMENT.
if(in == 2)

%
```

**I WANT TO CHANGE DIRECTION,
AND THE WORLD.**

GOT-THE-ENERGY-TO-LEAD.COM

We believe that energy suppliers should be renewable, too. We are therefore looking for enthusiastic new colleagues with plenty of ideas who want to join RWE in changing the world. Visit us online to find out what we are offering and how we are working together to ensure the energy of the future.

RWE
The energy to lead...



Click on the ad to read more

```
% TERMS INVOLVING NORMAL QUADRATURE.
    for igauss=1:ngauss; % each gauss point in turn
        it=1;
        ic=1;
        jacobi(m,igauss,it,ic);
        kern2(m,in,igauss);
        sfm=sf(in,igauss);
        browx(m,2*in-1)=browx(m,2*in-1)+wg(igauss)*bk2xx*sfn*jacob;
        browx(m,2*in)=browx(m,2*in)+wg(igauss)*bk2xy*sfn*jacob;
        browy(m,2*in-1)=browy(m,2*in-1)+wg(igauss)*bk2yx*sfn*jacob;
        browy(m,2*in)=browy(m,2*in)+wg(igauss)*bk2yy*sfn*jacob;
    end; % each gauss point in turn

%
% TERMS INVOLVING LOGARITHMIC QUADRATURE.
    for igauss=1:ngauss; % each gauss point in turn
        it=2;
        ic=2;
        jacobi(m,igauss,it,ic);
        sfm=sf(ic,in,igauss);
        dzde=1.;
        browx(m,2*in-1)=browx(m,2*in-1)+wgl(igauss)*cl*sfn*jacob*dzde;
        browy(m,2*in)=browy(m,2*in)+wgl(igauss)*cl*sfn*jacob*dzde;
        ic=3;
        jacobi(m,igauss,it,ic);
        sfm=sf(ic,in,igauss);
        dzde=1.;
        browx(m,2*in-1)=browx(m,2*in-1)+wgl(igauss)*cl*sfn*jacob*dzde;
        browy(m,2*in)=browy(m,2*in)+wgl(igauss)*cl*sfn*jacob*dzde;
    end; % each gauss point in turn
end;

%
% P AT THE THIRD NODE OF THE ELEMENT.
    if(in == 3)

%
% TERMS INVOLVING NORMAL QUADRATURE.
    for igauss=1:ngauss; % each gauss point in turn
        it=1;
        ic=1;
        jacobi(m,igauss,it,ic);
        kern2(m,in,igauss);
```

```

        sfn=sf(in,igauss);
        browx(m,2*in-1)=browx(m,2*in-1)+wg(igauss)*bk2xx*sfn*jacob;
        browx(m,2*in)=browx(m,2*in)+wg(igauss)*bk2xy*sfn*jacob;
        browy(m,2*in-1)=browy(m,2*in-1)+wg(igauss)*bk2yx*sfn*jacob;
        browy(m,2*in)=browy(m,2*in)+wg(igauss)*bk2yy*sfn*jacob;
    end; % each gauss point in turn
%
% TERMS INVOLVING LOGARITHMIC QUADRATURE.
    for igauss=1:ngauss; % each gauss point in turn
        it=2;
        ic=4;
        jacobi(m,igauss,it,ic);
        sfn=sfl(ic,in,igauss);
        dzde=2.;
        browx(m,2*in-1)=browx(m,2*in-1)+wgl(igauss)*cl*sfn*jacob*dzde;
        browy(m,2*in)=browy(m,2*in)+wgl(igauss)*cl*sfn*jacob*dzde;
    end; % each gauss point in turn;
end;
end; % each element node in turn
%
return;
end %function intker

```

```

function jacobi(m,igauss,it,ic)
%
% SUBPROGRAM TO EVALUATE THE JACOBIAN AND THE COMPONENTS OF THE UNIT
% NORMAL VECTOR AT A PARTICULAR GAUSS POINT.
% M INDICATES THE ELEMENT NUMBER.
% IGAUSS INDICATES THE GAUSS POINT NUMBER.
% IT INDICATES THE TYPE OF QUADRATURE.
% IT=1 - NORMAL GAUSSIAN QUADRATURE.
% IT=2 - LOGARITHMIC GAUSSIAN QUADRATURE.
% IC INDICATES THE CASE NUMBER FOR LOGARITHMIC GAUSSIAN QUADRATURE,
% AS DEFINED IN SUBROUTINE SHAPE.
%
global jacob ungy ungx node ynode xnode sdl sd

```

```
%  
% CALCULATE THE DERIVATIVES OF THE GLOBAL COORDINATES WITH RESPECT TO  
% THE LOCAL INTRINSIC COORDINATE.  
dx=0.;  
dy=0.;  
for in=1:3; % each element node in turn  
    if(it == 1)  
        sfnd=sd(in,igauss);  
    end;  
    if(it == 2)  
        sfnd=sdl(ic,in,igauss);  
    end;  
    j=node(m,in);  
    dx=dx+sfnd*xnode(j);  
    dy=dy+sfnd*yndode(j);  
end; % each element node in turn  
%  
% COMPONENTS OF THE LOCAL OUTWARD NORMAL VECTOR AT THE GAUSS POINT.  
ungx=dy;  
ungy=-dx;
```

bookboon.com

Corporate eLibrary

See our Business Solutions for employee learning

[Click here](#)



Click on the ad to read more

```
%
% JACOBIAN OF THE COORDINATE TRANSFORMATION.
jacob=sqrt(ungx^2+ungy^2);
%
% SCALE THE VECTOR COMPONENTS TO GIVE THE UNIT NORMAL VECTOR.
ungx=ungx/jacob;
ungy=ungy/jacob;
%
return;
end %function jacob

function kernel(xp,yp,m,igauss,unx,uny)
%
% SUBPROGRAM TO EVALUATE THE KERNELS WHEN P IS NOT THE CURRENT
% ELEMENT NODE.
% XP, YP INDICATE THE GLOBAL COORDINATES OF POINT P.
% M INDICATES THE ELEMENT NUMBER.
% IGAUSS INDICATES THE NUMBER OF THE GAUSS POINT, Q.
%
global bky bkyx bkxx akyy akyx akxy akxx nu e node ynode xnode sf
%
% COORDINATES OF GAUSS POINT Q.
xq=0.;
yq=0.;
for in=1:3; % each element node in turn
    sfn=sf(in,igauss);
    j=node(m,in);
    xq=xq+sfn*xnode(j);
    yq=yq+sfn*ynode(j);
end; % each element node in turn
%
% COMPONENTS AND MAGNITUDE OF THE RADIUS VECTOR FROM P TO Q.
rx=xq-xp;
ry=yq-yp;
rsq=rx^2+ry^2;
r=sqrt(rsq);
rx=rx/r;
ry=ry/r;
```

```
%
% RATE OF CHANGE OF R WITH THE NORMAL TO THE BOUNDARY AT Q.
drdn=rx*unx+ry*uny;
%
% PARAMETERS IN THE KERNEL FUNCTIONS.
c1=-1./(4.*pi*r);
c2=1.-nu;
c3=2.*(1.+nu);
c4=(1.+nu)^2/(4.*pi*e);
c5=(3.-nu)*log(1./r)/(1.+nu);
%
% EVALUATE THE KERNELS.
akxx=c1*(c2+c3*rx*rx)*drdn;
term1=c3*rx*ry*drdn;
term2=c2*(rx*uny-ry*unx);
akxy=c1*(term1-term2);
akyx=c1*(term1+term2);
akyy=c1*(c2+c3*ry*ry)*drdn;
bkxx=c4*(c5+rx*rx);
bkxy=c4*rx*ry;
bkyx=bkxy;
bkyy=c4*(c5+ry*ry);
%
return;
end %function kernel

function kern2(m,in,igauss)
%
% SUBPROGRAM TO EVALUATE THE NON-SINGULAR LOGARITHMIC TERM IN THE
% SECOND KERNEL WHEN P IS THE CURRENT ELEMENT NODE.
% M INDICATES THE ELEMENT NUMBER.
% IN INDICATES THE NUMBER OF THE ELEMENT NODE FORMING P.
% IGAUSS INDICATES THE GAUSS POINT NUMBER.
%
global bk2yy bk2xy bk2yx bk2xx nu e ynode xnode zg node sf
%
% COORDINATES OF GAUSS POINT Q.
```

```
xq=0.;
yq=0.;
for ic=1:3; % each element node in turn
    sfn=sf(ic,igauss);
    j=node(m,ic);
    xq=xq+sfn*xnode(j);
    yq=yq+sfn*yndoe(j);
end; % each element node in turn
%
% ELEMENT NODE NUMBERS.
i1=node(m,1);
i2=node(m,2);
i3=node(m,3);
%
% EVALUATE THE INTRINSIC COORDINATE.
zeta=zg(igauss);
%
% P AT THE FIRST NODE OF THE ELEMENT.
if(in == 1)
    xcomp=(zeta-2.)*xnode(i1)+2.*(1.-zeta)*xnode(i2)+zeta*xnode(i3);
```



Brain power

By 2020, wind could provide one-tenth of our planet's electricity needs. Already today, SKF's innovative know-how is crucial to running a large proportion of the world's wind turbines.

Up to 25 % of the generating costs relate to maintenance. These can be reduced dramatically thanks to our systems for on-line condition monitoring and automatic lubrication. We help make it more economical to create cleaner, cheaper energy out of thin air.

By sharing our experience, expertise, and creativity, industries can boost performance beyond expectations. Therefore we need the best employees who can meet this challenge!

The Power of Knowledge Engineering

Plug into The Power of Knowledge Engineering.
Visit us at www.skf.com/knowledge

SKF


```

    ycomp=(zeta-2.)*ynode(i1)+2.*(1.-zeta)*ynode(i2)+zeta*ynode(i3);
    rfn=sqrt(xcomp^2+ycomp^2);
    i=i1;
end;
%
% P AT THE SECOND NODE OF THE ELEMENT.
if(in == 2)
    xcomp=-0.5*(zeta-1.)*xnode(i1)+zeta*xnode(i2)-0.5*(zeta+1.)*xnode(i3);
    ycomp=-0.5*(zeta-1.)*ynode(i1)+zeta*ynode(i2)-0.5*(zeta+1.)*ynode(i3);
    rfn=sqrt(xcomp^2+ycomp^2);
    i=i2;
end;
%
% P AT THE THIRD NODE OF THE ELEMENT.
if(in == 3)
    xcomp=-zeta*xnode(i1)+2.*(1.+zeta)*xnode(i2)-(zeta+2.)*xnode(i3);
    ycomp=-zeta*ynode(i1)+2.*(1.+zeta)*ynode(i2)-(zeta+2.)*ynode(i3);
    rfn=sqrt(xcomp^2+ycomp^2);
    i=i3;
end;
%
% COMPONENTS AND MAGNITUDE OF THE RADIUS VECTOR FROM P TO Q.
xp=xnode(i);
yp=ynode(i);
rx=xq-xp;
ry=yq-yp;
rsq=rx^2+ry^2;
r=sqrt(rsq);
rx=rx/r;
ry=ry/r;
%
% PARAMETERS IN THE KERNEL FUNCTIONS.
c4=(1.+nu)^2/(4.*pi*e);
c5=(3.-nu)*log(1./rfn)/(1.+nu);
%
% EVALUATE THE KERNELS.
bk2xx=c4*(c5+rx*rx);
bk2xy=c4*rx*ry;
bk2yx=bk2xy;
bk2yy=c4*(c5+ry*ry);
%
```

```

return;
end %function kern2


function output(varargin)
%
% SUBPROGRAM TO WRITE OUT THE NODAL POINT VALUES OF DISPLACEMENTS
% AND ELEMENT STRESSES AND COMPUTE THE FORCES ON THE BOUNDARY SEGMENTS.
%
global nsegtot fyseg fxseg node maxl tmy jacob wg tmx sf ngauss ...
    isegelem nel sige sigsn sigss signn nu e unmx unmy v sd u ty tx ...
    ibce estore nnp uv ibcn fid6
%
% ARRANGE FOR U, V AND TX, TY TO CONTAIN THE NODAL DISPLACEMENTS
% AND TRACTIONS, RESPECTIVELY.
for i=1:nnp; % each node in turn
    if(ibcn(i) == 1)
        tx(i)=uv(2*i-1);
        ty(i)=uv(2*i);
    end;
    if(ibcn(i) == 2)
        u(i)=uv(2*i-1);
        v(i)=uv(2*i);
    end;
end; % each node in turn
%
% HEADING FOR OUTPUT OF NODAL DISPLACEMENTS AND TRACTIONS.
fprintf(fid6,['\n','NODAL POINT DISPLACEMENTS AND TRACTIONS','\n','\n',
...
    '      I      U      V      TX      TY','\n']);
e=estore;
for i=1:nnp; % each node in turn
%
% REMOVE THE SCALING.
u(i)=u(i)*maxl;
v(i)=v(i)*maxl;
tx(i)=tx(i)*estore;
ty(i)=ty(i)*estore;
%

```

```
% OUTPUT THE NODAL VALUES OF DISPLACEMENTS AND TRACTIONS.
fprintf(fid6,['%4i','%15.6e','%15.6e','%15.6e','%15.6e', ...
    '\n'],i,u(i),v(i),tx(i),ty(i));
end; % each node in turn
%
% HEADING FOR OUTPUT OF ELEMENT STRESSES.
fprintf(fid6,['\n','STRESSES AT THE NODES OF THE ELEMENTS', ...
    '\n','\n',' M IN SIGNN SIGSS SIGSN', ...
    ' SIGE','\n']);
%
% NORMAL AND SHEAR STRESSES AT THE NODES OF THE ELEMENTS.
for m=1:nel; % each element in turn
    for in=1:3; % each element node in turn
        if(ibce(m) ~= 2)
            j=node(m,in);
            tmx(m,in)=tx(j);
            tmy(m,in)=ty(j);
            signn(m,in)=tmx(m,in)*unmx(m,in)+tmy(m,in)*unmy(m,in);
            sigsn(m,in)=-tmx(m,in)*unmy(m,in)+tmy(m,in)*unmx(m,in);
        end;
    end;
end;
```

With us you can
shape the future.
Every single day.

For more information go to:
www.eon-career.com

Your energy shapes the future.

e-on



Click on the ad to read more

```
%
% DIRECT STRESS ALONG THE BOUNDARY SURFACE.
    ic1=node(m,1);
    ic2=node(m,2);
    ic3=node(m,3);
    igauss=in+10;
    dudz=sd(1,igauss)*u(ic1)+sd(2,igauss)*u(ic2)+sd(3,igauss)*u(ic3);
    dvdz=sd(1,igauss)*v(ic1)+sd(2,igauss)*v(ic2)+sd(3,igauss)*v(ic3);
    it=1;
    ic=1;
    jacobi(m,10+in,it,ic);
    ess=(-dudz*unmy(m,in)+dvdz*unmx(m,in))/(jacob*maxl);
    sigss(m,in)=e*ess+nu*signn(m,in);

%
% VON MISES EQUIVALENT STRESS.
    sige(m,in)=sqrt(signn(m,in)^2+sigss(m,in)^2 ...
        -signn(m,in)*sigss(m,in)+3.*signn(m,in)^2);

%
% OUTPUT THE STRESSES AT THE NODES OF THE ELEMENTS.
    fprintf(fid6,['%4i','%4i','%15.6e','%15.6e','%15.6e','%15.6e', ...
        '\n'],m,in,signn(m,in),sigss(m,in),signn(m,in),sige(m,in));
    end; % each element node in turn
end; % each element in turn

%
% COMPUTE THE FORCES ON THE BOUNDARY SEGMENTS.
for iseg=1:nsegtot; % each segment in turn
    fxseg(iseg)=0.;
    fyseg(iseg)=0.;
end; % each segment in turn
for m=1:nel; % each element in turn
    iseg=isegelem(m);

%
% APPLY GAUSSIAN QUADRATURE.
    fxelem=0.;
    fyelem=0.;
    for in=1:3; % each element node in turn
        for igauss=1:ngauss; % each gauss point in turn
            sfn=sf(in,igauss);
            it=1;
            ic=1;
```

```

        jacobi(m, igauss, it, ic);
        fxelem=fxelem+wg(igauss)*sfn*jacob*tmx(m,in)*maxl;
        fyelem=fyelem+wg(igauss)*sfn*jacob*tmy(m,in)*maxl;
    end; % each gauss point in turn
end; % each element node in turn
%
% ACCUMULATE THE FORCES ON THE SEGMENT.
    fxseg(iseg)=fxseg(iseg)+fxelem;
    fyseg(iseg)=fyseg(iseg)+fyelem;
end; % each element in turn
%
% OUTPUT THE SEGMENT FORCE RESULTS.
fprintf(fid6,['\n','FORCES ACTING ON THE BOUNDARY SEGMENTS', ...
            '\n','\n','SEGMENT      FX              FY','\n']);
for iseg=1:nsegtot;
    fprintf(fid6,['%5i','%14.5e','%14.5e','\n'], ...
            iseg,fxseg(iseg),fyseg(iseg));
end;
%
return;
end %function output

function [x,iflag]=elimin(a,neqn)
%
% SUBROUTINE FOR SOLVING SIMULTANEOUS LINEAR EQUATIONS BY GAUSSIAN
% ELIMINATION WITH PARTIAL PIVOTING.
%
% INITIALIZE ILL-CONDITIONING FLAG.
iflag=0;
for i=1:neqn; x(i)=0.; end;
%
% SCALE EACH EQUATION TO HAVE A MAXIMUM COEFFICIENT MAGNITUDE OF UNITY.
jmax=neqn+1;
for i=1:neqn; % each equation in turn
    amax=0.;
    for j=1:neqn; % search for maximum
        absa=abs(a(i,j));

```

```
    if(absa > amax)
        amax=absa;
    end;
end; % search for maximum
%
for j=1:jmax; % scale coefficients
    a(i,j)=a(i,j)/amax;
end; % scale coefficients
%
end; % each equation in turn
%
% COMMENCE ELIMINATION PROCESS.
for k=1:neqn-1; % eliminate each variable in turn
%
% SEARCH LEADING COLUMN OF THE COEFFICIENT MATRIX FROM THE DIAGONAL
% DOWNWARDS FOR THE LARGEST VALUE.
imax=k;
for i=k+1:neqn; % search for largest value
    if(abs(a(i,k)) > abs(a(imax,k)))
        imax=i;
    end;
end;
end;
```

© 2013 Accenture. All rights reserved.

be > your degree

Bring your talent and passion to a global organization at the forefront of business, technology and innovation. Discover how great you can be.

Visit accenture.com/bookboon

Be greater than.
consulting | technology | outsourcing

accenture
High performance. Delivered.



```

    end;
    end; % search for largest value
%
% IF NECESSARY, INTERCHANGE EQUATIONS TO MAKE THE LARGEST COEFFICIENT
% BECOME THE PIVOTAL COEFFICIENT.
if(imax ~= k)
    for j=k:jmax; % interchange coefficients
        atemp=a(k,j);
        a(k,j)=a(imax,j);
        a(imax,j)=atemp;
    end; % interchange coefficients
end;
%
% ELIMINATE X(K) FROM EQUATIONS (K+1) TO NEQN, FIRST TESTING FOR
% EXCESSIVELY SMALL PIVOTAL COEFFICIENT (ASSOCIATED WITH A SINGULAR
% OR VERY ILL-CONDITIONED MATRIX).
if(abs(a(k,k)) < 1.0e-5)
    iflag=1;
    return;
end;
for i=k+1:neqn; % each of remaining equations
    fact=a(i,k)/a(k,k);
    for j=k:jmax; % modify coefficients
        a(i,j)=a(i,j)-fact*a(k,j);
    end; % modify coefficient
end; % each of remaining equations
%
end; % eliminate each variable in turn
%
% SOLVE THE EQUATIONS BY BACK SUBSTITUTION, FIRST TESTING
% FOR AN EXCESSIVELY SMALL LAST DIAGONAL COEFFICIENT.
if(abs(a(neqn,neqn)) < 1.0e-5)
    iflag=1;
    return;
end;
x(neqn)=a(neqn,jmax)/a(neqn,neqn);
for i=neqn-1:-1:1; % then each unknown in turn backwards
    sum=a(i,jmax);
    for j=i+1:neqn; % sum products
        sum=sum-a(i,j)*x(j);
    end;
end;

```

```

    end; % sum products
    x(i)=sum/a(i,i);
end; % then each unknown in turn backwards
return;
end %function elimin

```

The equivalent of the Fortran SHARED DATA2EQ is shareddata2eq, stored as the file shareddata2eq.m, is as follows.

```

%%%--- module shareddata2eq;
%
% module STORING SHARED DATA.
%
global xeend; if isempty(xeend), xeend=zeros(1,260); end;
global yeend; if isempty(yeend), yeend=zeros(1,260); end;
global xnode; if isempty(xnode), xnode=zeros(1,500); end;
global ynode; if isempty(ynode), ynode=zeros(1,500); end;
global xsend; if isempty(xsend), xsend=zeros(1,250); end;
global ysend; if isempty(ysend), ysend=zeros(1,250); end;
global unmx; if isempty(unmx), unmx=zeros(250,3); end;
global unmy; if isempty(unmy), unmy=zeros(250,3); end;
global angstore; if isempty(angstore), angstore=zeros(1,260); end;
global maxl; if isempty(maxl), maxl=0; end;
global a; if isempty(a), a=zeros(1000,1001); end;
global uv; if isempty(uv), uv=zeros(1,1000); end;
global u; if isempty(u), u=zeros(1,500); end;
global v; if isempty(v), v=zeros(1,500); end;
global useg; if isempty(useg), useg=zeros(1,250); end;
global vseg; if isempty(vseg), vseg=zeros(1,250); end;
global arowx; if isempty(arowx), arowx=zeros(250,6); end;
global arowy; if isempty(arowy), arowy=zeros(250,6); end;
global browx; if isempty(browx), browx=zeros(250,6); end;
global browy; if isempty(browy), browy=zeros(250,6); end;
global zg; if isempty(zg), zg=zeros(1,8); end;
global wg; if isempty(wg), wg=zeros(1,8); end;
global egl; if isempty(egl), egl=zeros(1,8); end;
global wgl; if isempty(wgl), wgl=zeros(1,8); end;
global jacob; if isempty(jacob), jacob=0; end;
global ungx; if isempty(ungx), ungx=0; end;

```



```
global ungy; if isempty(ungy), ungy=0; end;
global signnseg; if isempty(signnseg), signnseg=zeros(1,250); end;
global sigsnseg; if isempty(sigsnsseg), sigsnseg=zeros(1,250); end;
global uelem; if isempty(uelem), uelem=zeros(1,250); end;
global velem; if isempty(velem), velem=zeros(1,250); end;
global signn; if isempty(signn), signn=zeros(250,3); end;
global sigss; if isempty(sigss), sigss=zeros(250,3); end;
global sigsn; if isempty(sigsns), sigsn=zeros(250,3); end;
global sige; if isempty(sige), sige=zeros(250,3); end;
global tx; if isempty(tx), tx=zeros(1,500); end;
global ty; if isempty(ty), ty=zeros(1,500); end;
global tmx; if isempty(tmx), tmx=zeros(250,3); end;
global tmy; if isempty(tmy), tmy=zeros(250,3); end;
global fxseg; if isempty(fxseg), fxseg=zeros(1,250); end;
global fyseg; if isempty(fyseg), fyseg=zeros(1,250); end;
global sf; if isempty(sf), sf=zeros(3,8); end;
global sd; if isempty(sd), sd=zeros(3,13); end;
global sfl; if isempty(sfl), sfl=zeros(4,3,8); end;
global sdl; if isempty(sdl), sdl=zeros(4,3,8); end;
global e; if isempty(e), e=0; end;
```



"I studied English for 16 years but...
...I finally learned to speak it in just six lessons"

Jane, Chinese architect

ENGLISH OUT THERE

Click to hear me talking before and after my unique course download



```

global nu; if isempty(nu), nu=0; end;
global estore; if isempty(estore), estore=0; end;
global akxx; if isempty(akxx), akxx=0; end;
global akxy; if isempty(akxy), akxy=0; end;
global akyx; if isempty(akyx), akyx=0; end;
global akyy; if isempty(akyy), akyy=0; end;
global bkxx; if isempty(bkxx), bkxx=0; end;
global bkxy; if isempty(bkxy), bkxy=0; end;
global bkyx; if isempty(bkyx), bkyx=0; end;
global bkyy; if isempty(bkyy), bkyy=0; end;
global bk2xx; if isempty(bk2xx), bk2xx=0; end;
global bk2xy; if isempty(bk2xy), bk2xy=0; end;
global bk2yx; if isempty(bk2yx), bk2yx=0; end;
global bk2yy; if isempty(bk2yy), bk2yy=0; end;
global nel; if isempty(nel), nel=0; end;
global nnp; if isempty(nnp), nnp=0; end;
global maxnel; if isempty(maxnel), maxnel=0; end;
global maxnnp; if isempty(maxnnp), maxnnp=0; end;
global maxnb; if isempty(maxnb), maxnb=0; end;
global neend; if isempty(neend), neend=0; end;
global ngauss; if isempty(ngauss), ngauss=0; end;
global node; if isempty(node), node=zeros(250,3); end;
global m1; if isempty(m1), m1=zeros(1,250); end;
global m3; if isempty(m3), m3=zeros(1,250); end;
global nbound; if isempty(nbound), nbound=0; end;
global nsegtot; if isempty(nsegtot), nsegtot=0; end;
global nelb; if isempty(nelb), nelb=zeros(1,10); end;
global nsegb; if isempty(nsegb), nsegb=zeros(1,10); end;
global nbcu; if isempty(nbcu), nbcu=0; end;
global nbcs; if isempty(nbcs), nbcs=0; end;
global nbcm; if isempty(nbcm), nbcm=0; end;
global nbct; if isempty(nbct), nbct=0; end;
global ibce; if isempty(ibce), ibce=zeros(1,250); end;
global ibcn; if isempty(ibcn), ibcn=zeros(1,500); end;
global isegbc; if isempty(isegbc), isegbc=zeros(1,250); end;
global isegend; if isempty(isegend), isegend=zeros(1,250); end;
global isegelem; if isempty(isegelem), isegelem=zeros(1,250); end;
global mfirst; if isempty(mfirst), mfirst=zeros(1,250); end;
global mlast; if isempty(mlast), mlast=zeros(1,250); end;

```

```
global nbcpc; if isempty(nbcpc), nbcpc=0; end;  
global maxnpc; if isempty(maxnpc), maxnpc=0; end;  
global nodepc; if isempty(nodepc), nodepc=zeros(1,20); end;  
global idirpc; if isempty(idirpc), idirpc=zeros(1,20); end;
```



The advertisement features a grey background with a faint world map. In the top left is the Duke University logo. The text 'BUSINESS HAPPENS' is prominently displayed in the center. Below it is the website URL 'www.fuqua.duke.edu/globalmba'. An orange button with the text 'Learn More >' is at the bottom. On the right side, there is a circular collage of six diverse individuals' faces, with the word 'HERE.' in bold black letters in the center of the collage.

DUKE
THE FUQUA
SCHOOL
OF BUSINESS

BUSINESS HAPPENS

www.fuqua.duke.edu/globalmba

Learn More >

HERE.



Solutions to Problems – Part II

Numerical answers are given, followed where appropriate by detailed notes on the solution procedure and comments on the results obtained.

Chapter 5

5.1 Indistinguishable when $\nu = 0$ ($\nu^* = \nu$ and $E^* = E$). Of little practical significance because no known materials have Poisson's ratios of zero, although some have values close to zero.

5.2 An incompressible material has a Poisson's ratio of $\nu = 1/2$. In plane strain $\nu^* = 1$ and $E^* = 4E/3$. The typical displacement and traction kernel functions become

$$U_{xx}(p, q) = \frac{3}{4\pi E} \left[\ln\left(\frac{1}{r}\right) + \hat{r}_x \hat{r}_x \right]$$

$$U_{xy}(p, q) = \frac{3}{4\pi E} \hat{r}_x \hat{r}_y$$

$$T_{xx}(p, q) = -\frac{1}{\pi r} \hat{r}_x \hat{r}_x \frac{dr}{dn}$$

$$T_{xy}(p, q) = -\frac{1}{\pi r} \hat{r}_x \hat{r}_y \frac{dr}{dn}$$

Unlike displacement-based finite element methods, incompressible plane strain presents no difficulties to the present boundary element formulation.

5.3
$$t_x \hat{n}_x + t_y \hat{n}_y = -k_n (u \hat{n}_x + v \hat{n}_y)$$

$$t_y \hat{n}_x - t_x \hat{n}_y = -k_s (v \hat{n}_x - u \hat{n}_y)$$

5.4 The body forces per unit volume in Equations 1.6 and 1.7 are $\bar{X} = 0$ and $\bar{Y} = -\rho g$, and the expressions for stresses satisfy these equations. Strains e_{xx} and e_{xy} are zero, from both the strain definitions of Equations 1.2 and 1.3, and in terms of stresses from Equations 1.20 and 1.23. From Equations 1.2

$$e_{yy} = \frac{\rho g y}{E^*} (1 - \nu^{*2})$$

and from Equation 1.21

$$e_{yy} = \frac{1}{E^*} (\sigma_{yy} - \nu^* \sigma_{xx}) = \frac{\rho g y}{E^*} (1 - \nu^{*2})$$

The expressions for e_{yy} are identical, and all the relevant equations are satisfied.

- 5.5 The body forces per unit volume in Equations 1.6 and 1.7 are $\bar{X} = \rho\omega^2x$ and $\bar{Y} = 0$, and the expressions for stresses satisfy these equations. Strains e_{yy} and e_{xy} are zero, from both the strain definitions of Equations 1.2 and 1.3, and in terms of stresses from Equations 1.21 and 1.23. From Equations 1.2

$$e_{xx} = -\frac{\rho\omega^2x^2}{2E^*}(1-\nu^{*2})$$

and from Equation 1.20

$$e_{xx} = \frac{1}{E^*}(\sigma_{yy} - \nu^*\sigma_{xx}) = -\frac{\rho\omega^2x^2}{2E^*}(1-\nu^{*2})$$

The expressions for e_{xx} are identical, and all the relevant equations are satisfied.

Chapter 6

- 6.1 Answer: maximum hoop stress concentration factor 1.005. The analytical solution for a hole in an infinite plate ($K \rightarrow \infty$) is 1 (Equations 6.8). For a plate width 20 times the hole diameter ($K = 20$), which ignores the square corners of the plate, it is 1.005, identical to four significant figures to that computed. But, with compressive radial stress (the pressure) at the surface of the hole, the hoop stress is not the largest stress to be considered there. The maximum von Mises equivalent stress concentration factor (relative to the applied pressure) is 1.74. Results were obtained using 4 elements per side of the outer boundary, 6 elements per quadrant of the hole.
- 6.2 Answers: at the inner surface, hoop stress 0.231, radial displacement 0.00240. At the outer surface, radial stress -0.539, hoop stress -0.231. Stress values are averaged over element end nodes and midside nodes: for example, typical hoop stress values at the inner surface are 0.2313 at end nodes, 0.2309 at midside nodes (a difference which is not significantly reduced by mesh refinement). Results were obtained using 4 elements per quadrant on both boundaries.

The general analytical solutions for stresses in a thick-walled cylinder are given by Equations 6.5. In this problem the boundary conditions are $\sigma_{rr} = -p$ at $r = r_1$ and $e_{\theta\theta} = 0$ at $r = r_2$, from which

$$A = \frac{-p(1+\nu^*)r_1^2}{(1-\nu^*)r_2^2 + (1+\nu^*)r_1^2}, \quad B = \frac{p(1-\nu^*)r_1^2r_2^2}{(1-\nu^*)r_2^2 + (1+\nu^*)r_1^2}$$

At the inner surface, $r = r_1$ and the hoop stress is

$$\sigma_{\theta\theta} = \frac{p[(1-\nu^*)K^2 - (1+\nu^*)]}{(1-\nu^*)K^2 + (1+\nu^*)}$$

where, $K = \frac{r_2}{r_1} = 2$, $p = 1$ and $\nu^* = \frac{\nu}{1-\nu} = 0.42857$, giving $\sigma_{\theta\theta} = 0.2308$.

At the outer surface, $r = r_2$ and the radial and hoop stresses are

$$\sigma_{rr} = \frac{-2p}{(1-\nu^*)K^2 + (1+\nu^*)} = -0.5384, \quad \sigma_{\theta\theta} = \nu^* \sigma_{rr} = -0.2308$$

Radial displacement at the inner surface can be found from the hoop strain with the aid of Equation 6.9

$$u_r = \frac{r}{E^*} (\sigma_{\theta\theta} - \nu^* \sigma_{rr})$$

Where $\sigma_{\theta\theta} = 0.2308$, $\sigma_r = -1$, and $E^* = \frac{E}{(1-\nu^2)} = 1098.9$, giving $u_r = 0.002402$.

- 6.3 Answer: 3.12. Result obtained using 4 elements per edge of the plate on edges without the notch, 4 elements per quadrant in the notch, and 10 elements on each of the segments adjacent to the notch. No significant change in the result was found when element sizes on these two segments were graded to give smaller elements near the notch.
- 6.4 Answer: 7.81. Result obtained using 4 elements per outer edge of the plate, 20 per straight side of the square hole, and 8 per quadrant of the rounded corners.

Join American online LIGS University!

Interactive Online programs
BBA, MBA, MSc, DBA and PhD

Special Christmas offer:

- ▶ enroll **by December 18th, 2014**
- ▶ **start studying and paying only in 2015**
- ▶ **save up to \$ 1,200** on the tuition!
- ▶ Interactive Online education
- ▶ visit ligsuniversity.com to find out more!

Note: LIGS University is not accredited by any nationally recognized accrediting agency listed by the US Secretary of Education. More info [here](http://ligsuniversity.com).



- 6.5** Answer: 0.681, a friction coefficient of more than 0.5 would be required to prevent slipping. The friction requirement is investigated by finding the ratio between shear and normal stresses (SIGSN and SIGNN) at nodes on the base of the pedestal, giving the minimum coefficient to prevent slipping. In this case the value of 0.5 is exceeded at points more than about 0.07 m from the central line of symmetry (less than about 0.08 m from the edges of the pedestal base). Results were obtained using 4 elements on each of the top and sloping sides of the pedestal, and 8 on the base. The boundary conditions are zero displacements on the base, and a normal stress of -1 on the top of the pedestal. Such contact problems, involving slippage with friction and possible local loss of contact, can be solved by boundary element methods, but are non-linear in the sense that the boundary conditions change with the loading, and require more sophisticated programming.
- 6.6** Answers: stiffness 142 MN/m, maximum stress (SIGSS) 190 MN/m² at the fillets, stress (SIGSS) variation over the central 30 mm of length is from 89.3 to 89.7 MN/m², compared with the nominal stress of 90 MN/m². A state of plane stress may be assumed. Results were obtained using 4 elements per straight edge at the ends of the test piece, 4 elements on each of the fillets, and 10 elements along each side of the central section (to provide more detail there, and to have element nodes at the ends of the central 30 mm). Point constraints were applied to the corners at the left hand end of the test piece. The stiffness is estimated from the computed relative displacement between the nodes at the centres of the two ends: 0.380×10^{-4} m. The tensile force in the test piece is $60 \times 10^6 \times 0.03 \times 0.003 = 5400$ N (stress times width times thickness), giving a stiffness of $5400/0.380 \times 10^{-4}$ N/m or 142 MN/m.
- 6.7** To apply pure shear, balancing shear stresses must be applied to all four edges of the plate: for example, $\sigma_{sn} = +1$ on BC and DA, $\sigma_{sn} = -1$ on CD and AB (Figure 6.7). The maximum computed hoop stresses at the hole are 4.03, occurring at four angular positions at 45° to the x and y axes. The analytical solution for a hole at the centre of an infinite plate in pure shear gives a value of 4 times the applied shear stress. Results were obtained using 4 elements per side of the outer boundary, 6 elements per quadrant of the hole.
- 6.8** Answer: 5.03 (hoop stress concentration factor at both sides of the hole). Rather than create the mesh data for an elliptical hole boundary, the same geometric data as in Problems 6.1 and 6.7 can be used, but with the y coordinates of the corners of the outer boundary doubled in magnitude. Then in subprogram MESHQ all YNODE values can be scaled by a factor of 0.5 immediately before finding the maximum dimension of the domain. This creates the elliptical shape of hole and brings the outer boundary shape back to square. Results were obtained using 4 elements per side of the outer boundary, 6 elements per quadrant of the hole. The exact solution for an elliptical hole with a ratio of major to minor axes of 2, under tension in an infinite plate, is 5.

6.9 Answers: (a) 3.11 at the outer sides of the holes ($y = 0, x = \pm 0.075$), 3.06 at the inner sides ($y = 0, x = \pm 0.025$). (b) 2.74 (at $x = \pm 0.05, y = \pm 0.025$). Results were obtained using 4 elements per side of the outer boundary, 6 elements per quadrant of each of the holes. An example of a solution domain with three boundaries.

6.10 Answer: 745 MN/m² at the bottom of the groove. The position of the maximum equivalent stress is as expected. With a radius ratio of $K = 2.5$, the hoop stresses at the inner and outer surface of the cylinder in the absence of the groove are

$$\sigma_{\theta\theta} = p \frac{(K^2+1)}{(K^2-1)} = 166 \text{ MN/m}^2 \quad \text{and} \quad \sigma_{\theta\theta} = \frac{2p}{(K^2-1)} = 45.7 \text{ MN/m}^2$$

These figures are useful for checking the computed results: hoop stresses at the inner surface remote from the groove should be reasonably close to these (within about 1–2%, say). The choice of mesh in this problem requires some care. Experience shows that 4 elements per quadrant should be sufficient on the outer boundary, and perhaps 6 elements per quadrant in the semi-circular groove. On the inner cylindrical surface it is important to have elements near the groove which are similar in length to the very small elements in the groove. Using elements of uniform size to satisfy this requirement leads to an unnecessarily large number of elements. A reasonable compromise was found to be 16 elements on each of two nearly semi-circular segments (that is, about 8 per quadrant), with an element length ratio (as explained in Sections 3.1.3 and 4.1.3) of 1.3. In other words, the constant ratio between the lengths of successive elements moving away from the groove was 1.3. The total number of elements used was 60. In addition to checking inner surface hoop stresses remote from the groove against the analytical values above, another useful test is for the tangential stresses (SIGSS) in the elements adjacent to the corner between the groove and the inside surface of the cylinder: at the common node they should both be close to the applied pressure. The computed hoop stress at the bottom of the groove is 678 MN/m², 4.08 times the hoop stress at the inner surface in the absence of the groove. Combined with a radial stress of -120 MN/m², this gives a von Mises equivalent stress of 745 MN/m², some 6.2 times the applied pressure in magnitude, also some 3.00 times the maximum equivalent stress in the absence of the groove.

- 6.11** Answer: 5.23. The problem does not state whether the fault in manufacture caused the cylinder to distort after machining, in which case the wall thickness would be constant around the circumference, or both inner and outer surface were made slightly elliptical. The effect on the result is unlikely to be significant at such a low level of eccentricity, and for convenience the latter is assumed. The change in geometry is effected by modifying the original mesh to scale all the YNODE values by a factor of 0.99 in subprogram MESHQ immediately before finding the maximum dimension of the domain. Results were obtained using 4 elements per quadrant on both boundaries. The result for maximum hoop stress in the circular cylinder is 5 times the internal pressure. So, a 1% eccentricity of the cylinder gives a 4.6% increase in the maximum hoop stress.
- 6.12** Answer: 299 Nm. The problem did not specify whether plane stress or plane strain is to be assumed. Because the state of deformation in the rubber bush is one of shearing, plane stress and plane strain give the same results. The problem calls for torque to be computed (via shear stress on the inner boundary) for a given rotation of this boundary. This rotation displacement boundary condition cannot be applied using the present program. On the other hand, a uniform shear stress can be applied to the inner boundary: a value of 1 N/m² was used. Results were obtained using 4 elements per quadrant on both the inner and outer circular boundaries. The computed value of tangential displacement at the inner boundary was 0.15750×10^{-9} m. At a radius of 0.01 m, this corresponds to an angular displacement of 0.15750×10^{-7} radians. The shear stress required to produce the required rotation of 0.1 radians is therefore

$$\text{shear stress} = 1 \times \frac{0.1}{0.15750 \times 10^{-7}} = 6.3492 \times 10^6 \text{ N/m}^2$$

This acts over a surface area of radius 0.01 m and length 0.075 m at a distance of 0.01 m from the axis of the shaft. Hence, the torque is

$$\text{torque} = 6.3492 \times 10^6 \times 2 \times \pi \times 0.01^2 \times 0.075 = 299 \text{ Nm}$$

The problem can also be solved analytically, giving a result of 299.2 Nm.