BOUNDARY ELEMENT AND FINITE ELEMENT METHODS
FOR
MOVING BOUNDARY ELECTROCHEMICAL PROBLEMS

by
Edward Carl Hume, III
A.B., Harvard College
(1977)

Submitted to the Department of
Chemical Engineering
in Partial Fulfillment of the
Requirements of the
Degree of
DOCTOR OF PHILOSOPHY

at the
MASSACHUSETTS INSTITUTE OF TECHNOLOGY

August 1983

(C) Edward C. Hume, III 1983

The author hereby grants to M.I.T. permission to reproduce and
to distribute copies of this thesis document in whole or in part.

Signature of Author ____________________________
Department of Chemical Engineering
August 29, 1983

Certified by ________________________________ Thesis Supervisor

Certified by ________________________________ Thesis Supervisor

Accepted by ________________________________ Chairman PhD. Committee

MASSACHUSETTS INSTITUTE
OF TECHNOLOGY
Oct 24 1983 Archives
LIBRARIES
BOUNDARY ELEMENT AND FINITE ELEMENT METHODS FOR
MOVING BOUNDARY ELECTROCHEMICAL PROBLEMS

by

EDWARD CARL HUME, III

Submitted to the Department of Chemical Engineering
on August 29, 1983 in partial fulfillment of the
requirements for the Degree of Doctor of Philosophy in
Chemical Engineering

ABSTRACT

A tertiary current distribution model was developed for
predicting two-dimensional shape change during electro-
deposition through polymeric masks. The cathode geometry
consisted initially of a parallel array of microscopic,
rectangular trenches. Concentration polarization, activation
polarization, and a stagnant diffusion layer treatment of
convective mass transport were incorporated into the nonlinear
model equations which were discretized using the boundary
element method (BEM). Calculations of the the deposit shape
history were made for different values of the geometrical
parameters, the level of convection, and the degree of
polarization.

Three formulations of the BEM and one of the Galerkin finite
element method (FEM) were used for the spatial discretization
of a two-dimensional, moving boundary test problem based on
Laplace's equation. Euler-predictor, trapezoid-corrector
integration was used in time. The methods were applied to a
finite domain or a semi-infinite strip by the use of infinite
elements with reciprocal decay functions or a Green's function
that met particular boundary conditions.

The accuracy and computation costs of the calculations were
compared for three sets of linear boundary conditions, and
various initial conditions for the moving surface. The BEM
calculations using piecewise quadratic interpolating functions
were more accurate and more efficient than those using linear
interpolation. At high accuracy levels, they were also
preferred to the FEM calculations that used biquadratic
interpolation, although the FEM was found to be advantageous
when accuracy requirements were relaxed. For the semi-infinite
domain, the BEM based on a specialized Green's function (GBEM)
was the only method that reliably satisfied the far field
condition. The GBEM offered superior accuracy and efficiency
in a more compact representation.

Thesis Supervisors:
Dr. William M. Deen, Associate Professor of Chemical Engineering
Dr. Robert A. Brown, Associate Professor of Chemical Engineering
ACKNOWLEDGEMENTS

I wish to express my deep appreciation to Dr. William M. Deen and Dr. Robert A. Brown for their help in the course of my thesis study. I consider myself doubly fortunate to have had two concerned and insightful advisors to call upon. The reader will also note that they play a good game of softball. I also want to thank Dr. Herb Sawin for serving on my thesis committee.

During this time I have appreciated the patience of my family and friends, and in particular, I have appreciated the support of my fiancee, Judy.

I am also grateful for the financial support that has made the rapid fruition of my thesis effort possible. I have received fellowship support through the Department of Chemical Engineering with IBM providing the bulk of my support. I have also been honored to receive a Mobil Educational Assistance Grant and thank Dr. Wei for nominating me for this award.
# TABLE OF CONTENTS

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>ABSTRACT</td>
<td>2</td>
</tr>
<tr>
<td>ACKNOWLEDGEMENTS</td>
<td>3</td>
</tr>
<tr>
<td>LIST OF FIGURES</td>
<td>6</td>
</tr>
<tr>
<td>LIST OF TABLES</td>
<td>8</td>
</tr>
<tr>
<td>PREFACE</td>
<td>9</td>
</tr>
<tr>
<td>CHAPTER I, LITERATURE SURVEY, NUMERICAL METHODS FOR FINITE DOMAINS</td>
<td>13</td>
</tr>
<tr>
<td>Finite Difference Methods</td>
<td>13</td>
</tr>
<tr>
<td>Finite Element Methods</td>
<td>17</td>
</tr>
<tr>
<td>Boundary Element Methods</td>
<td>19</td>
</tr>
<tr>
<td>CHAPTER II, COMPARISON OF FINITE ELEMENT AND BOUNDARY ELEMENT METHODS FOR A FINITE DOMAIN</td>
<td>26</td>
</tr>
<tr>
<td>Model Problem Formulation</td>
<td>27</td>
</tr>
<tr>
<td>Numerical Discretization</td>
<td>31</td>
</tr>
<tr>
<td>Integration in time</td>
<td>31</td>
</tr>
<tr>
<td>Boundary element formulation</td>
<td>36</td>
</tr>
<tr>
<td>Finite element formulation</td>
<td>44</td>
</tr>
<tr>
<td>Results and Discussion</td>
<td>46</td>
</tr>
<tr>
<td>Convergence of the field solution</td>
<td>47</td>
</tr>
<tr>
<td>Convergence of the normal gradient</td>
<td>53</td>
</tr>
<tr>
<td>Storage requirement</td>
<td>54</td>
</tr>
<tr>
<td>Computation cost of the field solution</td>
<td>56</td>
</tr>
<tr>
<td>Integration in time</td>
<td>62</td>
</tr>
<tr>
<td>Choice of boundary condition</td>
<td>68</td>
</tr>
<tr>
<td>Choice of domain shape</td>
<td>72</td>
</tr>
<tr>
<td>Summary and Conclusions</td>
<td>75</td>
</tr>
<tr>
<td>List of Symbols</td>
<td>80</td>
</tr>
<tr>
<td>CHAPTER III, INFINITE DOMAINS</td>
<td>82</td>
</tr>
<tr>
<td>Model Problem Formulation</td>
<td>85</td>
</tr>
<tr>
<td>Numerical Discretization</td>
<td>89</td>
</tr>
<tr>
<td>BEM implementation</td>
<td>89</td>
</tr>
<tr>
<td>BEM infinite elements</td>
<td>93</td>
</tr>
<tr>
<td>FEM implementation</td>
<td>99</td>
</tr>
<tr>
<td>Results and Discussion</td>
<td>100</td>
</tr>
<tr>
<td>Convergence of the field solution</td>
<td>101</td>
</tr>
<tr>
<td>Section</td>
<td>Page</td>
</tr>
<tr>
<td>---------------------------------------------------------------</td>
<td>------</td>
</tr>
<tr>
<td>Choice of boundary condition</td>
<td>107</td>
</tr>
<tr>
<td>Computation cost of the field solution</td>
<td>110</td>
</tr>
<tr>
<td>Storage requirement</td>
<td>112</td>
</tr>
<tr>
<td>Integration in time</td>
<td>112</td>
</tr>
<tr>
<td>Summary and Conclusions</td>
<td>116</td>
</tr>
<tr>
<td>List of Symbols</td>
<td>118</td>
</tr>
<tr>
<td>CHAPTER IV, MASS TRANSFER ANALYSIS OF Electrodeposition Through Masks</td>
<td>121</td>
</tr>
<tr>
<td>Electrodeposition Through Masks</td>
<td>122</td>
</tr>
<tr>
<td>Formulation of the Model Problem</td>
<td>126</td>
</tr>
<tr>
<td>Calculation Algorithms</td>
<td>136</td>
</tr>
<tr>
<td>Results and Discussion</td>
<td>138</td>
</tr>
<tr>
<td>Summary and Conclusion</td>
<td>154</td>
</tr>
<tr>
<td>List of Symbols</td>
<td>156</td>
</tr>
<tr>
<td>CHAPTER V, POSTSCRIPT</td>
<td>159</td>
</tr>
<tr>
<td>REFERENCES</td>
<td>164</td>
</tr>
<tr>
<td>APPENDICES</td>
<td></td>
</tr>
<tr>
<td>A. Derivation of the Specialized Green's Function</td>
<td>171</td>
</tr>
<tr>
<td>B. Elements, Basis Functions, and Mapping</td>
<td>174</td>
</tr>
<tr>
<td>C. Analytical Expressions for the LBEM Formulation</td>
<td>183</td>
</tr>
<tr>
<td>D. Treatment of Logarithmic Singularities</td>
<td>189</td>
</tr>
<tr>
<td>E. GBEM Computer Code Listing</td>
<td>193</td>
</tr>
<tr>
<td>F. FEM Computer Code Listing</td>
<td>214</td>
</tr>
<tr>
<td>G. QBEM Computer Code Listing</td>
<td>237</td>
</tr>
<tr>
<td>H. LBEM Computer Code Listing</td>
<td>262</td>
</tr>
<tr>
<td>I. Tafel Kinetics Model Code Listing</td>
<td>281</td>
</tr>
<tr>
<td>J. CPU_TIMER Listing</td>
<td>309</td>
</tr>
<tr>
<td>Figure</td>
<td>Description</td>
</tr>
<tr>
<td>--------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Fig. 2.1</td>
<td>A schematic depiction of the model problem for the finite domain</td>
</tr>
<tr>
<td>Fig. 2.2</td>
<td>The FEM mesh with 6 active surface elements, drawn to scale for the reference comparison case</td>
</tr>
<tr>
<td>Fig. 2.3</td>
<td>Convergence of the methods to the field solution on the initial finite domain</td>
</tr>
<tr>
<td>Fig. 2.4</td>
<td>Convergence of the GBEM to the field solution on the initial finite domain using different numbers of quadrature points per element</td>
</tr>
<tr>
<td>Fig. 2.5</td>
<td>Accuracy versus CPU time comparison for solving the initial field solution on the finite domain</td>
</tr>
<tr>
<td>Fig. 2.6</td>
<td>CPU time for the field solution as a function of the number of active surface elements</td>
</tr>
<tr>
<td>Fig. 2.7</td>
<td>Error in the position of the moving surface as a function of integration time</td>
</tr>
<tr>
<td>Fig. 2.8</td>
<td>Accuracy versus CPU time comparison for the error in the position of the active surface at t=0.6 using a constant time step of 0.05</td>
</tr>
<tr>
<td>Fig. 2.9</td>
<td>Accuracy versus CPU time comparison for the error in the position of the active surface at t=0.6 as the element mesh and time step were refined in tandem</td>
</tr>
<tr>
<td>Fig. 2.10</td>
<td>Convergence of the methods for the initial field solution as a function of the form of the active surface boundary condition</td>
</tr>
<tr>
<td>Fig. 2.11</td>
<td>Rapid field solution convergence of the GBEM on the rectangular finite domain</td>
</tr>
<tr>
<td>Fig. 3.1</td>
<td>A schematic depiction of the model problem for the semi-infinite domain</td>
</tr>
<tr>
<td>Fig. 3.2</td>
<td>Convergence of the methods to the field solution on the initial semi-infinite domain</td>
</tr>
<tr>
<td>Fig. 3.3</td>
<td>RMS error of the initial field solution versus the infinite element decay origin</td>
</tr>
<tr>
<td>Fig. 3.4</td>
<td>Convergence of the methods for the initial field solution as a function of the active surface boundary condition choice</td>
</tr>
<tr>
<td>Fig. 3.5</td>
<td>Comparison of CPU time versus accuracy for</td>
</tr>
</tbody>
</table>
obtaining the initial field solution (Honeywell 9180 mainframe) .......................................................... 111

Fig. 3.6 Comparison of CPU time versus accuracy for the error in the position of the active surface as the element mesh and time step are refined in tandem........... 114

Fig. 4.1 Processing steps in electrodeposition through masks ..................................................................... 123

Fig. 4.2 Two-dimensional model geometry ............................................................................................ 128

Fig. 4.3 Convergence of the overall mass balance and of the concentration solution at the cathode surface..... 139

Fig. 4.4 Average initial current density as a function of the polarization parameter, for three diffusion film thicknesses.......................................................... 142

Fig. 4.5 Deposit profiles for two values of the polarization parameter, when the trench area is 35% or 70% full .......................................................... 144

Fig. 4.6 Effect of diffusion film thickness on the development of deposit nonuniformity ......................... 148

Fig. 4.7 Effect of the trench spacing on the development of deposit nonuniformity .................................. 149

Fig. 4.8 Effect of the trench depth on the development of deposit nonuniformity .................................... 150

Fig. 4.9 Nonuniformity of the current density on the initial flat cathode surface as a function of the average current density ............................................. 153

Fig. A.1 The infinite strip ............................................................................................................................. 171

Fig. A.2 Conformal map of the strip to the half-plane ............................................................................. 172

Fig. A.3 The intuitive method of images .................................................................................................. 172

Fig. B.1 Mapping of a quadratic element to the prototype element .................................................. 175

Fig. B.2 Quadratic basis functions on the prototype one-dimensional element ..................................... 176

Fig. B.3 Global interpolation over two contiguous quadratic elements ................................................ 178

Fig. B.4 Mapping of a bi-quadratic element to the prototype element .................................................. 179
<table>
<thead>
<tr>
<th>Table</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Table 2.1</td>
<td>Finite domain storage requirements</td>
<td>55</td>
</tr>
<tr>
<td>Table 2.2</td>
<td>Scaling of computation costs with the number of active surface elements</td>
<td>60</td>
</tr>
<tr>
<td>Table 2.3</td>
<td>Convergence slopes for different boundary conditions</td>
<td>70</td>
</tr>
<tr>
<td>Table 3.1</td>
<td>Reciprocal decay functions for quadratic elements</td>
<td>96</td>
</tr>
<tr>
<td>Table 3.2</td>
<td>RMS Field solution errors for different active surface boundary conditions and choices of $\theta$</td>
<td>105</td>
</tr>
<tr>
<td>Table 3.3</td>
<td>Infinite domain program storage requirements</td>
<td>113</td>
</tr>
</tbody>
</table>
PREFACE

Electrode shape change is important in a variety of electrochemical processes. Examples include the analysis of battery cycling (Choi et al., 1976), electrosmeoothing and polishing (Wagner, 1954), electromachining (Riggs, 1977), electrodeposition to fabricate microelectronic devices (Romankiw, 1979), and more conventional uses of electrodeposition such as fabricating printed circuit boards (Kessler and Alkire, 1976). The instantaneous rate of deformation of an electrode is determined by the reaction rate distribution along the electrode surface, which reflects the electrochemical current distribution throughout the cell. In turn, the cell’s current distribution is modified by the changing geometry of the electrode surface. Therefore, the successful prediction of electrode shape change requires consideration in the analysis of the changing current distribution.

Mathematical models for the evolving shape of the electrode are posed as nonlinear moving boundary problems which require repetitive solution of the electrochemical current distribution as the electrode surface is deformed from its initial position (Alkire et al., 1978; Riggs, 1977; Sautebin et al., 1980, 1982; Prentice and Tobias, 1982b). Solution of a moving boundary problem places stringent demands on a numerical method. The method must accommodate irregular deforming surfaces without undue complication. It also must
be able to incorporate the nonlinear equations that are used
to describe electrode reactions, and provide accurate
solutions of the current distribution. Because the current
distribution is solved repeatedly, the method must be fast.

Chapter I of this thesis is a survey of the methods
that are appropriate for moving boundary electrochemical
problems. Where possible, examples are drawn from
electrochemical applications. Because of the potential
utility of these kinds of calculations there is a need for
guidance in the selection of an appropriate numerical method.
The methods that are useful for these problems are in a state
of rapid development and there have been few direct
comparisons in the literature. This is not surprising in that
implementation of even a single method is a sizeable
commitment of effort. With a working method in hand, the
particular problem of interest can be solved; thus, there is
little motive to implement another method even though it might
be superior. Furthermore, for a valid comparison, an
analytical solution to the problem must be available so that
the respective accuracies can be assessed.

In Chapter II of this thesis three variations of the
boundary element method (BEM) are compared with a finite
element method (FEM) formulation for a two-dimensional moving
boundary problem in a finite domain. These methods are
particularly advantageous for moving boundary problems of the
kind encountered in electrochemical engineering, because they
accomodate irregular and deforming domains in a regular way
using an isoparametric mapping. To facilitate the comparison of the methods, fictitious boundary conditions were chosen so that the test problem would have an analytical solution.

When the dimensions of a region to be analyzed are very much smaller than those of the surrounding medium, it is common to idealize the system boundaries as being at infinity. Many such problems arise in the context of structure-fluid interactions, for example, flow around objects in the ocean (Bettes and Zienkiewicz, 1977) or the occurrence of a meniscus on a wetted object (Orr et al., 1977). The numerical analysis of infinite domains is a rapidly developing area of research. In Chapter III of this thesis, the comparison problem that is considered in Chapter II is extended to a semi-infinite strip. In this study the efficacy of introducing "infinite elements" into the BEM and the FEM is examined. Another alternative considered is to construct a BEM formulation based on using Green's functions that fulfill the conditions on the infinite boundaries. This method results in a particularly compact numerical problem, and is found to be accurate and reliable.

Electroforming is the fabrication of structures by the electrolytic deposition of metal on a conducting mold. Using photolithography techniques and materials that have been developed for the manufacture of integrated circuits, it has become possible to create electroforming molds whose characteristic dimensions are one micron or less. In the semiconductor industry these microscopic electroforming molds
are termed mask structures, so that the process of electroforming with them is called electroplating through polymeric masks. The motivation for this thesis research was to examine the mass transfer characteristics of electroplating through masks. This investigation is presented in Chapter IV, where predictions of the time dependent shape evolution of the electrodeposit are made for various operating conditions.

In the last chapter of this thesis some additional observations concerning the BEM and the FEM are made, including consideration of other problems.
CHAPTER I

LITERATURE SURVEY, NUMERICAL METHODS FOR FINITE DOMAINS

Many numerical approaches have been applied to electrochemical current distribution problems; these have been reviewed recently by Prentice and Tobias (1982a). However, only a handful of methods have been demonstrated that are generally applicable to the moving boundary problem of electrode shape change. Reviews that have been specifically focused on methods for moving boundary problems include Elliot and Ockendon (1982), Furzeland (1980), Lynch and Gray (1980), Meyer (1978), and Fox (1975).

FINITE DIFFERENCE METHODS

Riggs (1977) applied the finite difference method (FDM) to a free boundary study of electromachining. Riggs determined two-dimensional, radially symmetric material shapes that represented steady-state solutions for the constant advance of a cathodic toolpiece into a dissolving anodic workpiece. He divided the electrolyte region into ten different blocks where different grid point densities were used. Laplace's equation was solved for the potential with nonlinear boundary conditions. An iterative overrelaxation
scheme was used to solve the nonlinear algebraic equation set. Values of the potential field were updated block by block and linear interpolation was used to transfer potentials across block boundaries where the node density changed. With a converged potential field in hand, the shape of the electrolyte region was adjusted until a steady-state shape was found. An iterative procedure was used to locate grid points so that they would meet certain spacing criteria and fall exactly on the anode boundary, resulting in a nonuniform mesh.

Riggs found that predicted shapes would vary widely depending on whether an edge node point was treated as a cathode point, an insulator in the radial direction, or as an insulator in the azimuthal direction. He was able to obtain close agreement with experimental results by treating the edge node as approximately the average of the first two choices.

Two cumbersome aspects of the FDM are apparent. First, it is awkward to accommodate variable node spacing and care must be taken not to compromise the accuracy of the method where different mesh regions join. Second, the method is not well suited for curved arbitrary boundaries, where special provision has to be made to interpolate the field variable and normal fluxes. An additional concern is that the FDM may be more sensitive to the treatment of singular node points than methods that are formulated in terms of area or boundary integrals, since the FDM deals only with nodal values.

Riggs (1977) also used a FDM program to solve the
moving boundary problem of anodic smoothing of sinusoidal profiles with constant potential boundary conditions (i.e., constant Dirichlet conditions). The problem was discretized in time using explicit forward Euler integration; following solution of the potential field, the surface nodes were moved a small distance proportional to the local normal current. This first-order finite difference treatment of the time domain has been the predominant method in the electrochemical literature (Prentice and Tobias, 1982a).

Prentice and Tobias (Prentice, 1980; Prentice and Tobias, 1982b,c) used the FDM to simulate deposition or dissolution of two-dimensional electrode profiles. A fixed mesh was established in space with fine spacing in the region of the electrode surface movement. The location of the curved electrode surface was described with the aid of cubic splines. As the surface moved through the mesh, the number of nodes included in the electrolyte region either decreased or increased. This approach makes good use of the FDM's strength for regular meshes, but is not well suited for problems that involve large or irregular boundary motions. In that case, the use of a high resolution mesh for the entire region of the boundary motion burdens the method with unnecessary spatial resolution.

Like Riggs, Prentice and Tobias used the iterative approach of successive overrelaxation to solve for the potential field. Their model involved nonlinear boundary conditions and they found that a weighting algorithm was
needed to overcome convergence problems at certain parameter values. Most analyses with the FDM have relied upon an indirect iterative solution to the underlying matrix problem since the computation cost per iteration is small. Other iteration methods can be used with the FDM. Our results from finite element calculations suggest that Newton's method would offer better convergence.

An alternative approach with the FDM is to deform the numerical mesh in accordance with the boundary motion. This approach has been used by Boris et al. (1975) in a two-dimensional, free surface hydrodynamic study using a generalized triangular mesh. Other uses of this approach with the FDM have largely been confined to one-dimensional problems (Lynch and Gray, 1980).

Another FDM approach is to apply a coordinate transformation that maps the deforming domain to one that is fixed in time. The transformed equation set is invariably nonlinear, but the regular domain is easily discretized and a wide range of problems can be accommodated. Two-dimensional applications with implicit integration in time have been demonstrated by Forsyth and Rasmussen (1979), Saitoh (1978), and Duda et al. (1975). A distinguishing feature of Forsyth and Rasmussen's study was that they transformed a variational integral of Laplace's equation rather than Laplace's equation itself. This made it unnecessary to obtain the second derivative of the moving boundary or cross derivatives of the field variable. Forsyth and Rasmussen found that an irregular
mesh gave results that compared well with those of a regular mesh, and resulted in computational savings. This occurred despite the fact that first order errors were introduced at points where the mesh size changed abruptly. Since the coordinate transformation was a nontrivial operation, their conclusion that an irregular mesh was desirable in the transformed system is a strong argument for application of methods that are suited to irregular and deforming meshes in the original system.

FINITE ELEMENT METHODS

Alkire, Berg, and Sani (1978) demonstrated the applicability of the Galerkin finite element method (FEM) to moving boundary electrochemical problems. Alkire et al. examined deposit shapes on a regular array of metal strips in a problem with nonlinear Butler-Vollmer electrode kinetics. The FEM was used in the two-dimensional spatial domain in combination with forward Euler integration in time. The finite element mesh deformed to follow the motion of the electrode surface, which was approximated by a continuous, piecewise quadratic curve.

The FEM has advantages over the FDM in the following respects. There are relatively few restrictions on the node structure used to discretize the system domain and nodes may be concentrated in regions of large gradients without
complicating the computational algorithm. Also, basis functions that are used to interpolate the potential field are also used to interpolate the system boundaries—this is the isoparametric mapping principle. The ability to conveniently represent curved or deforming boundaries is therefore built into the method. A similar implementation of the FEM has also been used to analyze electrochemical leveling of triangular wave profiles (Sautebin, Froidevaux and Landolt, 1980; Sautebin and Landolt, 1982).

Other FEM formulations are possible. For a field equation containing time derivatives, the implicit time dependence of the deforming mesh contributes to the formulation (Lynch and Gray, 1980). Bonnerot and Jamet (1974, 1977) have used finite elements in space and time to solve one- and two-dimensional parabolic problems. The use of the FEM in time accounts directly for the effect of mesh deformation. Lynch and Gray (1980) show a general FEM approach that accounts for the effect of mesh deformation, and retains the usual finite difference discretization in time. Another approach is to keep the finite element mesh fixed in space, but this approach abandons the particular strength that the FEM has for irregular domains.
BOUNDARY ELEMENT METHODS

The boundary element method (BEM) is a recent generalization of boundary integral equation methods using elemental basis functions that have been associated with the FEM. Recent texts by Brebbia (1978) and Brebbia and Walker (1980) explain the method. Theoretical aspects are given broader coverage by Jawson and Symm (1977), who do not discuss higher-order numerical approximation akin to the FEM approach.

The BEM can be applied to electrochemical problems where Laplace’s equation is appropriate to describe the field variable. By using an integral equation form of Laplace’s equation, it is possible to reduce a problem from a form involving an entire region to a form involving data along the region’s boundary. This contrasts with the FDM or the FEM where nodal points must be defined throughout the problem domain. This reduction in dimensionality of the problem can result in computational savings, especially if only surface values of dependent variables are sought. Unlike the FDM or the FEM which generate banded symmetric matrix problems, the BEM results in a matrix problem that is full and asymmetric. The economy of the technique hinges on the reduction of the problem to only the system boundaries and the concomitant savings in the number of node points.

As with the FEM, isoparametric basis functions that are used to interpolate dependent variables are used to define the system boundary, so that the BEM method shares the
advantages that the FEM has for irregular or deforming domains. Unlike the FEM or the FDM, application of the method is limited to problems where at least a free space Green's function (fundamental solution) for the governing equation can be derived. For the most part this restricts the method to linear field equations. The free space Green's functions do not in general satisfy boundary conditions of the problem. The solution calculated using the BEM approximates these boundary conditions in a weighted residual sense.

Christian and Rasmussen (1976) demonstrated a boundary integral equation approach to a moving boundary electromachining model. Their method was similar to the BEM, but they did not use isoparametric, element-based interpolating functions. The model geometry was a two-dimensional annular region with the dissolving anode in the center. Constant potential electrode surfaces were assumed. The work neglected many of the electrochemical effects that Riggs (1977) found important in his analysis of electromachining, and should be viewed as a demonstration of the mathematical technique. Piecewise linear curves were used to represent the electrode surfaces, and the normal current was approximated as being constant on each linear segment. With this low order approximation, the formulation was done using analytical expressions for the integral terms. After the potential field was obtained, potential values at the edges of the linear segments were found by a fourth-order interpolation of neighboring segment center values. Forward
Euler integration was used in time to deform the anodic surface. Their results compared well with an analytical solution for the case of circular electrode surfaces, and also with the results of a perturbation analysis.

Liggett (1977) demonstrated application of the BEM in a study of free surface hydraulic problems. Liggett used isoparametric linear basis functions and gave analytical expressions for the integral terms needed to formulate the matrix problem. Three points are to made concerning the use of isoparametric basis functions. First, they lead to a consistent order of approximation for the dependent and independent variables. Second, they provide a rule for interpolation between nodal values that is consistent with the overall accuracy of the solution. Third, using the same functions for interpolating both dependent and independent variables is convenient, and leads to a simpler, less-arbitrary numerical algorithm.

Most applications of the BEM have used piecewise constant or linear interpolation of the dependent and independent variables so that explicit analytical expressions were used for the integral formulation terms (e.g., Liggett, 1977; Jawson and Symm, 1977; Christiansen and Rasmussen, 1976; Hess, 1975; Shaw, 1974; Cruse 1974, 1969). Higher-order interpolation requires numerical evaluation of singular integrals as well as integrals that are difficult to evaluate accurately (Liggett and Salmon, 1981; Fairweather et al., 1979; Inoue et al., 1979; Longuet-Higgins and Cokelet, 1976;
Lachat and Watson, 1976; Boissenot et al., 1974). There have not been many implementations of the BEM with higher-order basis functions reported in the literature and only a few studies have shown any comparison of interpolation order. Fairweather et al. (1979) compared using isoparametric quadratic interpolation with using piecewise constant interpolation of the dependent variable on two test problems based on Laplace’s equation. Their quadrature procedure involved using logarithmic quadrature points and weights for the singular integrals in the formulation, otherwise a 4 point Gauss-Legendre quadrature scheme was employed. Superior accuracy was obtained using quadratic interpolation but no mention was made of computation work requirements. The convergence of the calculations with mesh refinement was not examined closely. The reported error data suggests that the convergence rate of the quadratic method was variable and slowed as more node points were introduced. Our results in Chapter II show that a higher-order quadrature rule is needed to maintain a consistent convergence rate.

Liggett and Salmon (1981) compared cubic spline interpolation against linear interpolation on three sample problems. With the exception of a free-surface problem, linear interpolation was more accurate for a given computation expense. The cubic interpolation required numerical quadrature to implement and the accuracy and/or cost of the quadrature was thought to be pivotal in the comparison. Their integration scheme was similar to that of Fairweather et al.
except they reported using up to 16 Gauss-Legendre points per boundary element for the nonsingular integrals.

In a two-dimensional BEM analysis of stress (an elastostatic analysis) Boissenot, Lachat, and Watson (1974) used quadratic interpolation to define their domain geometry, and linear, quadratic, or cubic functional interpolation. They found in most cases that quadratic or cubic functional variation was preferred to linear variation. In addition, their results compared well with calculations using the FEM on triangular elements with quadratic interpolation. Analytical results were not available and the accuracy versus computation tradeoff was not determined. They stated that computation times with the BEM could be as much as three or four times less than with the FEM. This observation must be tempered somewhat, since it was apparent from drawings of the FEM and BEM meshes that less than half as many surface points were used with the BEM calculations as were on the surface with the FEM calculations. A greater effort had been made to selectively locate the BEM surface nodes where the solution varied.

Lachat and Watson (1976) have also applied linear, quadratic and cubic basis functions in a three-dimensional BEM elastostatic study. The stress distribution in a pressure vessel was modeled, and an analytical result was not available. Lachat and Watson used a complicated partitioning scheme to insure that their numerical integration error was bounded throughout the domain. The results from using
quadratic and cubic interpolation were in relatively close agreement and differed somewhat from the calculations using linear interpolation. Since the computation cost of a refined mesh of linear elements was comparable to the cost of a coarse mesh of quadratic elements, it appeared that the higher-order methods were more cost effective.

The BEM may also be applied to time-dependent field equations (Wrobel and Brebbia, 1981; Chang et al., 1973; Shaw, 1973). Shaw (1973) has demonstrated the solution of a transient diffusion problem in three-dimensions. The integral equation was based on a Green's function that is a fundamental solution to the full differential equation.

In summary, the FEM and the BEM easily accommodate irregular boundaries and nonuniform meshes as a consequence of the mapping principles used in their formulation. They are well suited for moving boundary problems in two- and three-dimensions where application of the FDM is cumbersome. There have been few applications of the BEM using higher-order interpolation because of the method's newness and the complexity of the formulation integrals. The existing studies have not adequately addressed the convergence and economy of BEM formulations using piecewise linear, quadratic, or higher-order interpolation. Similar BEM and FEM formulations have never been compared on a moving boundary problem, and the few comparisons on multi-dimensional static domains have been inadequate for resolving the accuracy and computation work
tradeoffs of the methods. This thesis addressed these deficiencies with the two-dimensional moving boundary comparison problem described in the next Chapter. Linear and quadratic BEM formulations were compared with a similar biquadratic FEM formulation, under conditions where accuracy and computation work were precisely measured.
CHAPTER II

COMPARISON OF FINITE ELEMENT AND BOUNDARY ELEMENT METHODS
FOR A FINITE DOMAIN

Three formulations of the boundary element method (BEM) and one of the Galerkin finite element method (FEM) were used for the spatial discretization of a prototype two-dimensional, moving boundary problem. Except for one artificial boundary condition chosen to provide an analytical solution, the problem resembles an electrode deformation problem based on Laplace’s equation. Euler-predictor, trapezoid corrector integration was used to discretize the time domain. Analytical solutions for the position of the moving boundary and the value of the field variable made it possible to assess the accuracy of the formulations for both static and deforming domains. The BEM formulations differed by using either piecewise linear or piecewise quadratic basis functions to interpolate dependent and independent variables, and by using either the free space Green’s function or a Green’s function that was chosen to satisfy particular conditions of the prototype problem. The FEM formulation used piecewise biquadratic basis functions.
MODEL PROBLEM FORMULATION

The model problem consists of solving Laplace's equation for the field variable, \( C = C(x,y) \), on a deforming, two-dimensional domain, \( \Omega \), with boundary surface, \( \Gamma \), described in a regular Cartesian coordinate system, (\( x,y \)). (A list of symbols appears at the end of the chapter.)

\[ \nabla^2 C = 0, \text{ in } \Omega \]  \hspace{1cm} (2.1)

The domain shape is depicted in Figure 2.1. The domain is confined between the two vertical surfaces at \( x=0 \) and \( x=1 \). A horizontal surface at \( y=H \) forms the top of the domain, and a deforming surface described by \( y=h(x,t) \) forms the bottom.

The homogeneous Dirichlet boundary condition, \( C=0 \), is imposed along the horizontal top surface, \( \Gamma \).

\[ C=0, \text{ on } \Gamma_1; \ y=H, \ 0 \leq x \leq 1 \]  \hspace{1cm} (2.2)

The straight vertical sidewalls, \( \Gamma_2 \), are taken to be symmetry planes.

\[ \mathbf{n} \cdot \nabla C = 0, \text{ on } \Gamma_2; \ x=0, \ x=1, \ h(x,t) \leq y \leq H \]  \hspace{1cm} (2.3)

A Robin boundary condition is applied to the active bottom surface, \( \Gamma_3 \).

\[ \mathbf{n} \cdot \nabla C = f_1 C + f_2, \text{ on } \Gamma_3; \ y=h(x,t), \ 0 \leq x \leq 1 \]  \hspace{1cm} (2.4)

The functions \( f_1 \) and \( f_2 \) are chosen so that the solution to the field equation (2.1) at any time is
Fig. 2.1 A schematic depiction of the model problem for the finite domain.
\[ C = \cos(\pi x) [\exp(-\pi y) - \exp(\pi(y-2H))] \quad (2.5) \]

The active surface, \( y = h(x,t) \), moves along the \( y \) axis with a rate equal to the local value of the field variable, \( C \).

\[ \frac{\partial h}{\partial t} = C(x, h(x,t)) \quad (2.6) \]

An initial configuration of the active surface must be specified, but the choice is arbitrary, i.e.,

\[ h(x,t=0) = h_0(x), \quad h_0(x) < H \quad (2.7) \]

Boundary condition (2.4) may be imposed using different pairs of functions \( f_1 \) and \( f_2 \) and still lead to the same field solution (2.5). We are interested in comparing the convergence of the methods for the following three choices:

**Type I - inhomogeneous**

\[ f_1 = 0 \quad (2.8a) \]

\[ f_2 = -\pi \{ n \sin(\pi x) [\exp(-\pi y) - \exp(\pi(y-2H))] \\
+ n \cos(\pi x) [\exp(-\pi y) + \exp(\pi(y-2H))] \} \]

**Type II - mixed**

\[ f_1 = -\pi n, \quad \frac{[\exp(-\pi y) + \exp(\pi(y-2H))] \} \\
[\exp(-\pi y) - \exp(\pi(y-2H)))] \quad (2.8b) \]

\[ f_2 = -\pi n \sin(\pi x) [\exp(-\pi y) - \exp(\pi(y-2H))] \]
Type III - mixed

\[ f_1 = -\pi n_\perp \tan(\pi x) \]
\[ f_2 = -\pi n_\perp \cos(\pi x) \left[ \exp(-\pi y) + \exp(\pi(y-2H)) \right] \]

\[ (2.8c) \]

In these expressions \( n_\perp \) and \( n_\parallel \) indicate components of the active surface normal, \( \mathbf{n} = n_\perp \mathbf{e}_\perp + n_\parallel \mathbf{e}_\parallel \), that points outward from the problem domain.

The correct position of the active surface at any time is found by integrating equation (2.6) using the field solution (2.5).

\[ \frac{\partial h}{\partial t} = \cos(\pi x) \left[ \exp(-\pi h) - \exp[\pi(h-2H)] \right] \]  \[ (2.9) \]

The substitution,

\[ \{ \exp(-\pi h) - \exp[\pi(h-2H)] \} = 2\sinh[\pi(H-h)] \exp(-\pi H), \]  \[ (2.10) \]

is made, and equation (2.9) is integrated by separation of variables. After rearrangement the result is,

\[ h(x,t) = H - \frac{2}{\pi} \tanh^{-1} \left( \tanh \left( \frac{\pi(H-h(x,0))}{2} \right) \right) \times \]
\[ \exp \left[ -2t \pi \cos(\pi x) \exp(-\pi H) \right] \]  \[ (2.11) \]

Two kinds of results are obtained from the numerical solution of the model problem. First, the ability of a numerical method to solve for the field solution at a particular time for a particular domain shape is tested. Second, given an initial shape of the domain, the accumulated
error in the calculated position of the surface as a function of elapsed time is measured.

NUMERICAL DISCRETIZATION

Integration in time

Implementation of a numerical solution requires discretization of the problem in two spatial dimensions and in time. The methods that are compared in this study vary substantially in their treatment of the spatial dimensions but the same approach is taken to the time integration. Equation (2.6) is regarded as an ordinary differential equation describing the motion of points on the active surface, \( y = h(x,t) \). At any instant considered in the time integration, the instantaneous rates of surface movement are obtained from numerical solution of the field equation (2.1).

An Euler-predictor, trapezoid-corrector time integration scheme is used to move the active surface. This constant time-step integration algorithm is also referred to as the improved Euler method (Dorn and McCracken, 1972). The essentials of the method are easily understood in a single equation context. Given the problem,

\[
\frac{dY}{dt} = f(Y),
\]

(2.12)

\( Y(t+\Delta t) \) is found from \( Y(t) \) by making the forward-Euler
prediction,

\[ Y_{pred} = Y(t) + \Delta t [ f(Y(t)) ] \]  \hspace{1cm} (2.13)

and then by using this value in a trapezoid-correction,

\[ Y(t+\Delta t) = Y(t) + (\Delta t/2) [ f(Y_{pred}) + f(Y(t)) ] \]  \hspace{1cm} (2.14)

The method is second order accurate in time, with local error being \( O(\Delta t^3) \), and therefore offers a similar level of approximation in time as the quadratic basis functions offer in the spatial representation. Also, the program structure is readily modified to implement the implicit trapezoid rule (Dahlquist and Bjorck, 1974), as might be desired for stability reasons when solving a different problem. For the comparison problem the right-hand portion of the active surface is accelerating towards \( y = -\infty \). The problem is mathematically unstable in the sense that error in the position of the active surface will increase with time irrespective of the integration scheme chosen. The electrodeposition process is similarly unstable, and this is discussed in Chapter IV.

The following algorithm demonstrates how the time integration scheme is applied to the two-dimensional model problem. The general form of the procedure used to find the field solution is also presented; however, details concerning the matrix equation formulation and the nodal structure will not be discussed until later, when the individual methods used to obtain the field solution are considered.
1. Initialization

A. The initial structure of nodes is defined. Nodes on or near the active surface move as the domain deforms, but other nodes are assigned to fixed locations.

B. The contributions to the matrix equation that arise from the fixed location nodes are calculated and stored. These results are used to initialize the matrix terms for each complete Euler-predictor, trapezoid-corrector time step.

2. Commence a new time step by a forward Euler prediction of the active surface position for the next time step.

A. Obtain the field solution and the rate of the active surface movement based on its current position.

B. Save the current active surface position.

C. Temporarily move the active surface
according to the Euler-predicted rates.

3. Trapezoid rule correction

A. Again obtain the field solution and the active surface movement rates.

B. Average the newly calculated rates with the surface movement rates found in the Euler prediction.

C. Move the active surface from its saved original position at this time step according to this composite rate, thus completing an entire integration time step.

4. Repeat steps 2 and 3 until the predetermined stop time has been reached.

The cost of solving the field equation at any instant can be reduced by keeping a portion of the node structure fixed and not recalculating matrix contributions from this portion. This economy is reflected in the following procedure that is used to obtain the field solution as required in steps 2 and 3 above.
1. Initialize the matrix terms for the current time step. The matrix equation contributions from the fixed nodes are read from storage.

2. Reposition the nodes in the vicinity of the active surface to accommodate its current position. This will not be necessary with the specialized Green's function BEM.

3. Calculate matrix contributions from the moving nodes and add them to the fixed contributions.

4. Solve the complete linear matrix problem directly by Gaussian-elimination with partial pivoting. The field solution and its values along the active surface are obtained.

A direct solution of the matrix problem is preferred to an iterative solution since the formulation of the matrix terms is relatively expensive. This contrasts with finite difference methods where formulation costs can be small relative to the cost of solving the equation set.
Boundary Element Formulation

The mathematical formulation of the BEM is derived from the requirement that weighted residual integrals of Laplace's equation vanish over the two-dimensional domain, $\Omega$.

$$
\int_{\Omega} w_i \nabla^2 C \, d\Omega = 0 
$$

(2.15)

The weighting function, $w_i = w_i(x_i, y_i; x, y)$, is defined with reference to a node point $(x_i, y_i)$ and an observation point $(x, y)$. Integrating by parts and using the divergence theorem reduces equation (2.15) to

$$
\int_{\Omega} \nabla w_i \cdot \nabla C \, d\Omega - \int_{\Gamma} w_i (n \cdot \nabla C) \, d\Gamma = 0
$$

(2.16)

where $d\Gamma$ implies integration over the system boundaries, $\Gamma$ (see Figure 2.1). This equation is the basis of the finite element method, as will be shown in the next section. A boundary element formulation is obtained by integrating by parts a second time, and using the divergence theorem to yield,

$$
- \int_{\Omega} \nabla^2 w_i \, d\Omega + \int_{\Gamma} C (n \cdot \nabla w_i) \, d\Gamma - \int_{\Gamma} w_i (n \cdot \nabla C) \, d\Gamma = 0
$$

(2.17)

The first integral over the domain is eliminated by choosing a Green's function (fundamental solution) for Laplace's equation as the weight function $w_i$. The general two-dimensional
boundary element method formulation is based on using the free
space Green's function

\[ w_i(x_i, y_i; x, y) \equiv -\frac{1}{4\pi} \ln \left[ \frac{(x-x_i)^2+(y-y_i)^2}{x^2+y^2} \right] \quad (2.18) \]

which satisfies, \( \nabla^2 w_i(x_i, y_i; x, y) + \delta(x-x_i)\delta(y-y_i) = 0 \). Here \( \delta \) is the Dirac delta function. Substituting equation (2.18) into equation (2.17) results in the integral form of Laplace's equation

\[ \Theta_i C_i - \frac{1}{2\pi} \int_{\Gamma} C(\mathbf{n} \cdot \nabla \ln(r)) d\Gamma + \frac{1}{2\pi} \int_{\Gamma} \ln(r) (\mathbf{n} \cdot \nabla C) d\Gamma = 0 \quad (2.19) \]

The form of equation (2.19) for any Green's function is,

\[ \Theta_i C_i + \int_{\Gamma} C(\mathbf{n} \cdot \nabla w_i) d\Gamma - \int_{\Gamma} w_i (\mathbf{n} \cdot \nabla C) d\Gamma = 0 \quad (2.20) \]

where \( r = \sqrt{(x-x_i)^2+(y-y_i)^2} \), and \( \Theta_i \) depends on the point, \((x_i, y_i)\), at which the equation is applied. When \((x_i, y_i)\) is outside of the domain, \( \Theta_i = 0 \); when \((x_i, y_i)\) is inside of the domain, \( \Theta = 1 \); and when \((x_i, y_i)\) is chosen to be on the domain boundary, as is the usual case, \( \Theta_i \) equals the interior angle of the domain divided by \( 2\pi \). For the usual smooth surface point, \( \Theta_i = 1/2 \).

Equation (2.19) is the usual starting point for the BEM. It is a general formulation that can be applied to solving Laplace's equation irrespective of the boundary conditions. However, when a Green's function is chosen that
fulfills selected homogeneous boundary conditions of the problem, all integrals along the surfaces where conditions have been met evaluate to zero and are eliminated from the formulation. There is no need to introduce node points to discretize these surfaces and a more compact computational problem results. This approach is applied to the model problem by choosing a Green's function that fulfills the zero gradient boundary condition along the vertical sidewalls. The necessary Green's function is derived by conformal mapping and the method of images, as described in Appendix A, and is,

$$w_1 = \frac{-1}{4\pi} \ln\left( \frac{\cosh[\pi(y-y_1)] - \cos[\pi(x+x_1)]}{\cosh[\pi(y-y_1)] - \cos[\pi(x-x_1)]} \right)$$

(2.21)

It satisfies $\partial w_1 / \partial x = 0$ along the vertical surfaces $x=0$ and $x=1$, for all values of $y$.

A matrix formulation of the BEM is constructed for this problem as follows. More general matrix approaches are shown by Brebbia (1978), Liggett (1977), or Brebbia and Walker (1980). A set of nodal points is chosen only along the boundary. The value of a dependent or independent variable at any location on the boundary is found from its value at the nodal points as interpolated by the basis functions:

$$C = \sum_i \bar{\varphi}_i(x,y) C_i, \quad \frac{dC}{dn} = \sum_i \bar{\varphi}_i(x,y) \left( \frac{dC}{dn} \right)_i$$

(2.22)

$$x = \sum_i \bar{\varphi}_i(x,y) x_i, \quad y = \sum_i \bar{\varphi}_i(x,y) y_i$$

Here $dC/dn$ symbolizes $\mathbf{n} \cdot \nabla C$. The basis function, $\bar{\varphi}_i$, is
associated with the $i$th node since it has the property of equaling unity at the node point $(x_i, y_i)$ and equaling zero at other node points. In this study linear and quadratic polynomial basis functions have been employed. In practice, groups of contiguous nodes are conceived as belonging to elements and the form of the basis functions reflects a local elemental coordinate system. The net effect is a continuous, piecewise linear or quadratic representation of the interpolated variable. This is discussed in Appendix B, where expressions for the basis functions are given. A typical boundary mesh for the model problem is shown as Figure 2.2, which is actually a drawing of the bi-quadratic finite element mesh. The quadratic boundary elements correspond to the walls of the individual finite elements that are on the outer surface of the domain. No node points in the interior of the domain are used in the BEM.

The boundary conditions of equations (2.2) - (2.4), are substituted directly into the integral equation (2.20) or they are approximated using a basis function representation and then substituted. The first approach was used throughout with one exception. When employing linear basis functions, applying a piecewise linear approximation to boundary condition (2.4) made it possible to formulate the problem in terms of integral contributions that were done analytically, avoiding the use of numerical quadrature. Closed form expressions for the necessary terms are listed in Appendix C. Direct substitution of the desired boundary conditions avoids
Fig. 2.2 The FEM mesh with 6 active surface elements, drawn to scale for the reference comparison case. Only the lower part of the domain deforms in time.
the need for special treatment of corner nodes where Neumann
and Dirichlet boundary conditions meet. Otherwise, provision
has to be made for double values at this kind of corner, since
the corner node is associated with an unknown value of C along
one surface, and an unknown value of dC/dn along the abutting
surface. The direct substitution approach is used in the
following development.

The boundary conditions for a well-posed problem
specify either C, dC/dn, or a relationship between them, at
every point along the surface. Therefore application of
equation (2.20) using each node as a base point for the
Green's function results in a set of simultaneous equations
for the unknown nodal values of C and dC/dn. The matrix
formulation

\[ AX = F \]

results from substituting the basis function representation
(2.22) and the boundary conditions (2.2-4,8) into the integral
equation (2.17). The matrix terms are defined as

\[ X_i \equiv \begin{cases} 
C_i, & \text{node } i \text{ is on } \Gamma_2 \text{ and } \Gamma_3 \\
(dC/dn)_i, & \text{node } i \text{ is on } \Gamma_1, \\
\text{including corners shared with } \Gamma_2 
\end{cases} \]

\[ F_i \equiv \int_{\Gamma_3} w_i f \, d\Gamma \]
When $X_i = C_i$:

$$A_{ij} = \theta_i \delta_{ij} + \int_{\Gamma_2, \Gamma_3} \bar{e}_i (n \cdot \nabla w_i) \, dr - \int_{\Gamma_1} w_j f_i \, dr \quad (2.26a)$$

When $X_i = (dC/dn)_i$:

$$A_{ij} = \int_{\Gamma_1} \bar{e}_i w_j \, dr \quad (2.26b)$$

For the general BEM, the integral contributions are nonzero only over the boundary segments indicated. Note that for the BEM using the Green's function (2.21) no integration is done over the surface $\Gamma_2$ since $(n \cdot \nabla w_i)$ is zero along this segment.

**Evaluation of BEM Integral Terms**

The integrals needed for the matrix formulation are calculated numerically using Gauss-Legendre quadrature, except for the formulation using linear basis functions, as previously noted. For the general boundary element method, the choice of 4 or 6 quadrature points per element was sufficient to insure that the overall accuracy of the method was independent of the integration error in evaluating the individual terms. With the specialized Green's functions, more quadrature points were required. This topic is discussed with the analysis of results.

Integrals containing the Green's function, $w_i(x_i, y_i; x, y)$, (equation 2.18 or 2.21) become singular when
integrating over the portion of the boundary containing the
node point \( (x_i, y_i) \). To accurately evaluate these integrals
using Gauss-Legendre quadrature requires more points and more
computation than using a quadrature rule that is specially
tailored for logarithmic singularities. A logarithmic
quadrature rule was applied as shown in Appendix D by
recasting the singular integrals as the sum of a singular
integral of appropriate form and a nonsingular integral
remainder. The necessary manipulations are described for both
the free space Green's function (equation 2.18) and the
specialized Green's function (equation 2.21).

The value of \( \theta_i(x_i, y_i) \) can be calculated from the
interior angle of the domain at \( (x_i, y_i) \) by using the basis
function representation. However, we have found that superior
accuracy is obtained if \( \theta_i \) is calculated by difference as
described by Brebbia (1978) and as shown in Chapter III.

Some of the above integrals require calculation of the
surface normal pointing outward from the domain. This is
easily obtained from the isoparametric mapping. In terms of
the local element coordinate, \( \xi \), explained in Appendix E, the
components of the surface normal are,

\[
n_x = \frac{dy/d\xi}{\sqrt{(dy/d\xi)^2 + (dx/d\xi)^2}} = \frac{dy/d\xi}{\sqrt{(dy/d\xi)^2 + (dx/d\xi)^2}} (2.27a)
\]

\[
n_y = \frac{dx/d\xi}{\sqrt{(dy/d\xi)^2 + (dx/d\xi)^2}} (2.27b)
\]
with \( \frac{dx}{d\xi} \) and \( \frac{dy}{d\xi} \) obtained from the expansions

\[
\frac{dx}{d\xi} = \sum_i d\tilde{u}_i(x, y)/d\xi, \quad \frac{dy}{d\xi} = \sum_i d\tilde{v}_i(x, y)/d\xi \quad (2.28a,b)
\]

**Finite Element Formulation**

Formulation of the finite element method follows directly from equation (2.16). The basis functions, \( \tilde{u}_i \), that are used to interpolate the field variable are also used as weighting functions in the residual equations.

\[
\int_{\Omega} \nabla \tilde{u}_i \cdot \nabla C \ d\Omega - \int_{\Gamma} \tilde{u}_i (n \cdot \nabla C) \ d\Gamma = 0 \quad (2.29)
\]

An algebraic equation set is formed from (2.29) by subdividing the domain into two-dimensional area elements in analogy to the partitioning of the domain surface in the application of the BEM. A typical mesh is shown as Figure 2.2. The details of this discretization are not critical and are treated in Appendix B along with an explanation of the basis functions and the isoparametric mapping procedure. The bi-quadratic basis functions employed in this study are obtained as the product of the one-dimensional quadratic basis functions that are used in the BEM formulations.

Application of the boundary conditions to equation (2.29) follows traditional methods for the Galerkin FEM (Zienkiewicz, 1977; Chung, 1978; Finlayson, 1980). On the vertical sidewalls where there is a zero normal gradient
boundary condition, the surface integral is set to zero, so that no calculation is needed. At the top boundary, the value of the field variable is set to zero. Since the field variable is specified at the top surface nodes, the residual equations based on these nodes are dropped from the overall matrix formulation so the system of algebraic equations is not overdetermined. Accordingly, the integral along the top surface no longer appears in the formulation. Along the bottom, active surface, the condition on the normal gradient of the field variable is set as \( \mathbf{n} \cdot \nabla C = f_1 C + f_2 \) (equations 2.4, 2.8). This expression is substituted directly into the surface integral of equation (2.29). The field variable, \( C(x,y) \), is replaced by its finite element representation,

\[
C = \sum_i \tilde{e}_i C_i \tag{2.30}
\]

and the resulting equations are written in matrix form as

\[
A C = F \tag{2.31}
\]

by defining

\[
A_{ij} = \int_{\Omega} \tilde{e}_i \cdot \nabla \tilde{e}_j d\Omega - \int_{\Gamma_3} \tilde{e}_i \tilde{e}_j f_1 d\Gamma \tag{2.32}
\]

\[
F_i = \int_{\Gamma_3} \tilde{e}_i f_2 d\Gamma, \quad C_i = \text{nodal value of } C \tag{2.33, 34}
\]

The integrals in (2.32) and (2.33) are evaluated by Gauss-Legendre integration. A fixed integration rule of 3x3
quadrature points per area element, with 3 quadrature points per element surface along \( \Gamma_i \), is sufficient to insure that the overall accuracy of the method is not diminished by integration error and that a definite solution is determined (Strang and Fix, 1973).

RESULTS AND DISCUSSION

The methods compared here are:

FEM - finite element method with biquadratic basis functions,

QBEM - boundary element method with quadratic basis functions using the free space Green's function (2.18),

LBEM - boundary element method with linear basis functions using the free space Green's function (2.18),

GBEM - boundary element method with quadratic basis functions using the specialized Green's function (2.21).

A particular test case of the model problem is constructed by specifying the domain dimensions and choosing a boundary condition. We now define a reference case to serve as a starting point for the comparison of these methods. To
this end, the top of the domain is arbitrarily set at \( y=4 \), and the initial shape of the active surface is specified as a portion of a cosine wave (see Figure 2.2).

\[
h(x,t=0) = 0.15 [1 - \cos(mx)], \quad 0 \leq x \leq 1 \quad (2.35)
\]

Also, the third boundary condition choice for the active surface is selected as the reference case (equations 2.4 and 2.8c). Unless it is specified otherwise, comparison results are based on these particular conditions.

*Convergence of the Field Solution*

Since solving the time-dependent moving boundary model problem will depend upon solving for the field variable twice at each time step, the accuracy and efficiency of the field solution are important considerations for the overall process. The error in the calculation of the field solution was examined using the initial domain shape, equation (2.35). The measure of error used throughout this study is the root-mean-square difference between the analytical solution and the numerical solution at the nodes on the active surface, \( \Gamma_3 \).

\[
\text{RMS ERROR} = \sqrt{\frac{N}{N}} \sum_{i=1}^{N} \frac{E_i^2}{N}
\]

\[
E_i = [\text{True Value at } h(x_i, t)] - [\text{Calculated Value at } h(x_i, t)] \quad (2.36)
\]
Equation (2.36) is used for both the field solution error (the exact solution given by equation (2.5)) and for the active surface position error (the exact position given by equation (2.11)).

The various methods were compared on the basis of an equivalent number of elements along the active surface. The total number of elements was in proportion to the number of active surface elements, keeping the mesh similar. Thus the number of active surface elements is proportional to the inverse of the distance between two adjacent nodes throughout the domain. The FEM mesh for 6 active surface elements is drawn to scale in Figure 2.2. When there were N active surface elements there are NxN total finite elements. With the QBEM, the boundary elements along the active surface and along the walls are exactly the FEM element boundaries along these surfaces. However, half as many elements were placed along the horizontal top surface, where there was little solution activity. With the FEM, the number of elements along the active surface established the number of element columns used throughout the domain. For the LBEM, each additional surface element was associated with only 1 additional node, instead of 2 additional nodes found with the QBEM. Therefore it required 2N linear elements to have the same number of nodes along the active surface as N quadratic elements.

Figure 2.3 shows the dependence of the field solution error for the various methods on the number of elements along the active surface, N. As N was increased, the calculations
Fig. 2.3 Convergence of the methods to the field solution on the initial domain. The number of other elements were kept in proportion to the number of active surface elements. The root-mean-square error is based only on the active surface nodes.
became increasingly accurate as evidenced by the negative sloping curves. The slopes indicated in the figure are based on the three points in each curve where the most elements were employed.

The theory of Strang and Fix (1973) gives assurance that the error due to the selected FEM quadrature rule (3x3 Gauss-Legendre quadrature) is negligible in comparison to that of the basis function approximation and the node spacing. Because of the complexity and variety of the formulation integrals, there is no similar theoretical result for the BEM and the overall accuracy of the method may be compromised by numerical integration error. With the FEM, the order of the polynomial basis functions and the node to node spacing limit the accuracy since other sources of error are negligible. Under these conditions the FEM solution error for solving Laplace's equation is expected to behave as

\[
\text{Solution Error} \propto N^{-(R+1)}
\]  \hspace{1cm} (2.37)

as the number of elements, \( N \), is made infinite (Strang and Fix, 1973). Here \( R \) is the order of the polynomial basis functions and \( N \) is inversely proportional to the node to node spacing. A similar theoretical convergence proof does not exist for the BEM, though it is reasonable to expect that the relation (2.37) applies if the integration error is made negligible.

Applying the convergence rule (2.37) to the biquadratic FEM, \( R=2 \), so a curve slope of \(-3\) is expected in
Figure 2.3. Reported slopes were based on the three points representing the most elements, which gave -3.63 for the FEM, in close agreement. The linear BEM formulation does not rely on numerical integration, and its observed convergence slope was -2.56. The non-rigorous application of (2.37) with R=1 gives an expected slope of -2, also in close agreement.

The case of the quadratic BEM is more complex. Informal application of (2.37) with R=2 implies that a convergence slope of -3 is attainable. However, this predicted rate of convergence is not obtained if the formulation integrals are not calculated with sufficient accuracy. If four Gauss-Legendre quadrature points were used per element, the solution convergence rate slowed when large numbers of elements were employed. Using six Gauss-Legendre points per element resulted in a rate of convergence of -3.48 which compares favorably with the FEM result.

For the specialized Green's function BEM, the integrands are more complex than for the general BEM and even higher-order quadrature rules are needed. The GBEM curve in Figure 2.3 was obtained by increasing the number of quadrature points per element from 4 to 32 as the number of surface elements was increased. Figure 2.4 shows different curves for the GBEM where each curve represents the convergence behavior for a particular quadrature rule. The number of quadrature points is varied between 4 and 32 per boundary element. It is apparent that even with 32 quadrature points per element the convergence rate of the solution slows down as large numbers
Fig. 2.4 Convergence of the GBEM to the field solution on the initial domain using different numbers of quadrature points per element.
of elements are introduced. The GBEM integrals contain rapidly varying hyperbolic cosine functions (equation 2.21) which are believed responsible for the diminished accuracy of numerical integration.

**Convergence of the Normal Gradient**

In the model problem, the boundary condition along the active surface is imposed on the normal gradient, and the error in the field solution along this surface is measured. An important distinction is to be made between the BEM and the FEM when the normal gradient is to be calculated. For the FEM, the normal gradient is obtained by differentiating the calculated values of the field variable. The convergence rate of the gradient is expected to be one order less than the convergence of the field variable (Strang and Fix, 1973). With the BEM, the normal gradient is obtained directly and the rate of its convergence is the same as the convergence of the field variable, as long as the formulation integrals are calculated with sufficient accuracy.

FEM solutions to the model problem were examined as if the gradient was unknown and could not be related analytically to the field variable data through the boundary condition. Thus, the normal flux was calculated by differentiating the field solution and comparing it to that given by the analytical solution. The results have been plotted in Figure 2.3 along with convergence curves for the field solution so
that the comparative difficulty that the FEM had in calculating the normal flux is apparent. The observed convergence rate of the flux calculation was found to be -1.80, which is close to the theoretical estimate of -2, based on the order of the basis functions.

**Storage Requirement**

Storage requirements for the various methods are shown in Table 2.1, which summarizes the matrix sizes needed for the methods as a function of the number of active surface elements. The figures show that 16 FEM active surface elements results in a larger storage requirement than 32 QBEM active surface elements or 64 GBEM active surface elements. With the linear BEM, there is only one new node for each new element. The compactness of the LBEM is the same as the QBEM if they are compared on a node to node basis. The most accurate result was obtained with the QBEM using 32 active surface elements (Figure 2.3). The computer storage required for the main formulation matrix scaled as $49N^2$ (Table 2.1) for the QBEM where $N$ was the number of active surface elements. The FEM required substantially more memory, scaling as $12N^3$, despite the fact that the FEM results in a banded symmetric matrix, and the boundary methods result in matrices that are full and asymmetric. The most accurate result was obtained with the QBEM as a consequence of being able to introduce more surface elements, while maintaining sufficient integration.
### TABLE 2.1

**FINITE DOMAIN PROGRAM STORAGE REQUIREMENTS**

<table>
<thead>
<tr>
<th>Active Surface Elements</th>
<th>LBEM</th>
<th>FEM</th>
<th>QBE</th>
<th>GBEM</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Total Nodes</td>
<td>Total Matrix Storage</td>
<td>Total Nodes</td>
<td>Total Matrix Storage</td>
</tr>
<tr>
<td>2</td>
<td>7</td>
<td>49</td>
<td>25</td>
<td>206</td>
</tr>
<tr>
<td>4</td>
<td>14</td>
<td>196</td>
<td>81</td>
<td>1162</td>
</tr>
<tr>
<td>6</td>
<td>21</td>
<td>441</td>
<td>169</td>
<td>3446</td>
</tr>
<tr>
<td>8</td>
<td>28</td>
<td>784</td>
<td>289</td>
<td>7634</td>
</tr>
<tr>
<td>16</td>
<td>56</td>
<td>3136</td>
<td>1089</td>
<td>54946</td>
</tr>
<tr>
<td>32</td>
<td>112</td>
<td>12544</td>
<td>4225</td>
<td>416066</td>
</tr>
<tr>
<td>64</td>
<td>224</td>
<td>50176</td>
<td>16641</td>
<td>3236482</td>
</tr>
</tbody>
</table>

The number of elements on other surfaces is kept in proportion to number of active surface elements.

**Matrix Storage Requirement Formulae:**

- **LBEM** = \( \frac{49N^2}{4} \)
- **FEM** = \( 12N^3 + 22N^2 + 10N + 2 \)
- **QBE** = \( 49N^2 \)
- **GBEM** = \( 9N^2 + 12N + 4 \)

\( N = \) the number of active surface elements
accuracy. The GBEM is even more economical of computer storage, scaling as $9N^2$.

*Computation Cost of the Field Solution*

Another dimension in the comparison of these methods is their relative cost in obtaining the field solution. The methods were implemented on the M.I.T. Information Processing Services' Honeywell model 6180 mainframe computer under the Multics time-sharing operating system. A measure of the computation work needed to solve for a solution is the elapsed virtual CPU time. This timing data is provided by the operating system and is corrected to remove effects of other concurrent processes.

The methods were implemented as FORTRAN programs (see Appendix) and compiled using an optimization option. With one exception the timing data reflects this optimization. In the case of the GBEM, the "optimized" program would generate incorrect results in 83% of the time that it took the unoptimized program to calculate the correct results. The GBEM data reported here is for the unoptimized program, with the understanding that there is potential for further savings. The computer codes were intended to be reasonably efficient and to be similar in the tradeoffs between execution time and program length or clarity. No attempt was made to rewrite the computer codes for enhanced performance or to use exceptional coding methods. Double precision floating point calculations
Fig. 2.5 Accuracy versus CPU time comparison for solving the initial field solution (Honeywell 9180 mainframe).
(72 bit) were used throughout.

In Figure 2.5 the CPU time needed to solve for the field solutions is plotted as a function of the accuracy attained. The figure illustrates clearly which method offered the least expensive field solution at a given level of error. The worst performer is the linear basis function BEM. The piecewise linear approximation of the LBEM was not expected to be as accurate as the QBEM. Conceivably the analytical formulation of the LBEM might have been so much less expensive that the LBEM would still have been an economical method. However, the analytical expressions for the formulation integrals required evaluation of arctangents and logarithms. Presumably the cost of evaluating these special functions and the lower accuracy of the method caused the overall poor economy. Focusing on the QBEM, using 4 quadrature points per element was most economical for less accurate results but using 6 quadrature points per element was preferable at higher accuracy. The curve for the GBEM represents using from 4 to 32 quadrature points as the mesh was refined. The GBEM was preferred to the 4 point QBEM over the entire range. The 6 point QBEM was again found to be the most economical method when the highest accuracy was sought. For a wide range of accuracies the FEM was able to deliver the least expensive results. It was only at the highest accuracy when the GBEM using 6 quadrature points was preferred.

In Figure 2.6 the CPU time data is plotted against the number of surface elements. For the GBEM a constant
Fig. 2.6 CPU time as a function of the number of active surface elements. Curve slopes are given in Table 2.2.
<table>
<thead>
<tr>
<th>Method</th>
<th>Using All of the Points</th>
<th>Using the 2 points with the most elements</th>
</tr>
</thead>
<tbody>
<tr>
<td>LBEM</td>
<td>1.96</td>
<td>2.31</td>
</tr>
<tr>
<td>QBEM (4 quad. pts.)</td>
<td>2.03</td>
<td>2.31</td>
</tr>
<tr>
<td>GBEM (4 quad. pts.)</td>
<td>1.75</td>
<td>2.16</td>
</tr>
<tr>
<td>FEM</td>
<td>2.65</td>
<td>3.22</td>
</tr>
</tbody>
</table>
quadrature rule of 4 points per element was maintained for this comparison. In formulating the BEM, integration was performed on every boundary element for every node point. The number of elements and the number of nodes was proportional to the number of active surface elements, N. Thus, the total formulation work scaled as N² for each version of the BEM. Direct solution of the matrix problem scaled as N³ for the BEM. The curve slopes in Figure 2.6, given in Table 2.2, therefore provide insight into the relative cost of formulation and matrix solution. The boundary methods were evidently dominated by formulation costs, since the overall cost grew as N². For the FEM, integration was performed on every element a single time so that the total formulation work scaled as N² as with the BEM. The number of FEM unknowns was proportional to N², and the bandwidth of the matrix was proportional to N, so the matrix solution work scaled as N⁴ (Dahlquist et al., 1974). With the FEM, the cost of solving the matrix problem had a much larger impact on the overall cost, as evidenced by the cost growth exponent of 3.22.

With each method there are improvements that can be made to reduce the computation expense, and the comparison times must be viewed as approximate. For the FEM, the bandwidth of the matrix problem can be reduced approximately 8% by using the node numbering scheme indicated in the subroutine GRIDSTART (see FEM listing in the Appendix). Another alternative is to use reduced quadratic elements that do not have center nodes, which would then make it desirable
to apply a 2x2 quadrature rule in the fixed region where the elements are rectangular (Bathe and Wilson, 1976). The 2x2 quadrature rule is not appropriate with biquadratic elements for stability reasons (Strang and Fix, 1973). For the BEM, savings can be realized by substituting a lower order quadrature rule when the base point for the Green’s function is widely separated from the element where integration is being performed. In this vein Lachat and Watson (1976) demonstrated an evaluation scheme for integrals that involved subdividing elements to meet error bounds according to the variation of the integrands. Also, greater effort can be made to preevaluate quantities that are needed in the different integrals and avoid their recalculation. A more basic but much longer range change is to do the formulation work for the BEM or the FEM in parallel using more advanced computer systems. Assuming that solving the simultaneous equation set would still be done serially, the economy of the BEM would be improved relative to the FEM.

Integration in Time

Integration in time requires the repetitive solution of the field variable as the domain is deformed. To evaluate the various methods in this context, the analytical expression for the location of the active surface (equation 2.11) was used to calculate an RMS position error (equation 2.36) at any time.
In Figure 2.7, the effect of the time step size on the development of position error is considered. Each successive curve in the figure represents halving the integration time step. The number of elements along the active surface was held constant at 6. The curves shown are based on application of the FEM but are virtually indistinguishable from results obtained using the other methods.

The curves in Figure 2.7 show that the position error was reduced by decreasing the time step. As smaller time steps were chosen, the marginal improvement in the accuracy diminished. Since each halving of the time step doubled the required computation work, the implication was that an appropriate time step existed representing the desired tradeoff between computation work and accuracy. It will be shown that the appropriate time step size depends on the fineness of the element mesh. As stated previously, the comparison problem is unstable. The rapid increase of error in the calculated position of the active surface after some time is a consequence of its unstable acceleration towards $y = -\infty$. The process of electrodeposition is also unstable and further consideration of this subject is found in Chapter IV.

The effect of varying the time step with a fixed element mesh has just been considered. If the time step size is fixed at 0.05 and the number of node points is varied, an analogous comparison emerges. Figure 2.8 shows the computation cost incurred as a function of the accuracy attained when the integration time reached $t = 0.6$. 

63
Fig. 2.7 Error in the position of the moving surface as a function of integration time. Effect of using different size time steps is shown for the FEM. Results for the other methods were similar.
Introduction of more node points was at first successful in reducing the error with a reasonable increase in computation expense. However as more node points were used, the computation costs rose much more dramatically than could be justified by the meager improvement in the accuracy of the solution, the curves in Figure 2.8 finally becoming vertical. This happened when the fixed time step became relatively coarse, and by itself, limited the accuracy obtained.

These results emphasize that to maintain a consistent convergence rate as more node points are introduced, the time step and the element mesh must be refined together. When this strategy was tried, it yielded results of higher accuracy with less computation. Figure 2.9 shows the effect of keeping the ratio of the time step to the nodal separation distance fixed as more node points were introduced. The ratio chosen gave a time step size of 0.05 for 8 quadratic elements on the active surface. The scales in Figures 2.8 and 2.9 are the same to facilitate comparison. At the highest accuracy, a three-fold reduction in the RMS error of the active surface position is achieved, for a given computation time.

The performance of the methods on the moving boundary problem was similar to their performance on the initial field solution problem. However, with the moving boundary, error that entered into the calculations accumulated. Therefore at each time step, the full accuracy of the field solution method was not justified. In particular, using 4 quadrature points with the GBEM and with the GBEM for the moving boundary
Fig. 2.8 Accuracy versus CPU time comparison for the error in the position of the active surface at t=0.6. A constant time step size of 0.05 was used and the element meshes were varied.
Fig. 2.9 Accuracy versus CPU time comparison for the error in the position of the active surface at \( t = 0.6 \) as the element mesh and time step were refined in tandem. The ratio of time step to node spacing was held constant.
problem was as effective as using higher accuracy quadrature rules. Figures 2.8 and 2.9 reflect this quadrature choice and the resulting computational savings.

The poor performance of the LBEM on the field solution problem (Figure 2.5) carried over directly to the moving boundary problem (Figure 2.9), since unlike the other BEM formulations there was no opportunity to use a more economical quadrature rule. The GBEM was found to be more economical than the 4-point QBEM, as found with the field solution results. The GBEM and the QBEM were initially more expensive than the FEM but as more nodes were used and a more accurate solution was obtained, they became more economical than the FEM. In Figure 2.9 it is apparent that the cost of accuracy rose faster with the FEM than with the boundary element methods.

*Choice of Boundary Condition*

In the formulation of the model problem, three choices of the boundary condition for the active surface were presented (equation 2.8), each of which is consistent with the same analytical solution. The results shown thus far have been based on the third choice of the boundary condition. In Figure 2.10, the convergence of the methods for all choices of the boundary condition is contrasted. Each part of the figure corresponds to a single method, showing the RMS error in the field solution for the initial domain. The curves for the FEM
Fig. 2.10 Convergence of the methods for the initial field solution as a function of the form of the active surface boundary condition (see text). Six quadrature points per element were used for the QBEM and the GBEM. Curve slopes are given in Table 2.3.
TABLE 2.3

CONVERGENCE SLOPES FOR DIFFERENT BOUNDARY CONDITIONS

Slopes of from Figure 2.10 (Log(RMS Error) versus Log(N)) using the three points with the highest number of elements

<table>
<thead>
<tr>
<th>Method</th>
<th>Boundary Condition Choice (I)</th>
<th>(II)</th>
<th>(III)</th>
</tr>
</thead>
<tbody>
<tr>
<td>LBEM</td>
<td>-2.09</td>
<td>-3.07</td>
<td>-2.56</td>
</tr>
<tr>
<td>GBEM (6 quad. pts.)</td>
<td>-3.51</td>
<td>-3.51</td>
<td>-3.48</td>
</tr>
<tr>
<td>FEM</td>
<td>-3.66</td>
<td>-3.81</td>
<td>-3.63</td>
</tr>
<tr>
<td>GBEM (6 quad. pts.)</td>
<td>-2.74</td>
<td>-0.98</td>
<td>-1.10</td>
</tr>
</tbody>
</table>
and the QBEM were very similar. The second choice of boundary condition yielded higher errors but when sufficient elements were used, all three boundary conditions lead to nearly the same convergence rate (Table 2.3). The results obtained with the first and third boundary conditions became almost indistinguishable.

The case of the LBEM was more complex. Because of the linear basis functions, more elements were needed to have error levels similar to the quadratic methods. Figure 2.10 shows that the second boundary condition choice yielded higher levels of error at first, but had a faster convergence rate as elements were added. The error levels, and the convergence rates were not too dissimilar though.

The choice of boundary condition has the largest impact on the GBEM. We have already seen in Figure 2.4 that up to 32 quadrature points were needed to maintain a rapid convergence rate with the third boundary condition choice. The second boundary condition choice had this property too. Also, the error levels with the second boundary condition choice were generally higher than with the third choice, as was the case with the FEM and the QBEM. Here the difference between the two choices was somewhat larger. There were two surprises with the first boundary condition choice. First, the level of error is very low. Using only two elements along the active surface resulted in an RMS error of $10^{-3}$. The second surprise was that 6 quadrature points per element was sufficient to yield cubic convergence of the method. One
possible reason for the improved convergence of this boundary condition involves subtle aspects of the matrix problem formulation. The inhomogeneous boundary condition terms were not added individually into the main matrix but were summed into the forcing righthand-side vector. This left the main matrix composed only of $H_{ii}$ terms that are defined in Chapter III (equation 3.18). Because of the evaluation method used to find $\Theta_i$ and $H_{ii}$ (also in Chapter III) the rows of this matrix sum to zero; presumably this is more forgiving of numerical error. The net result is that error levels with this boundary condition are from one to three orders of magnitude less than with the second and third boundary conditions.

Choice of Domain Shape

These results have been based on the particular comparison case domain shape stated in equation (2.35). The model problem is equally valid for other active surface shapes. Other shapes that have been examined include a cosine curve of twice the frequency and a flat active surface at variable inclinations (triangle wave). The range of possibilities vitiates any claim to a thorough investigation. With this caveat, the results that have been observed have been very similar to the results presented for the standard comparison case.

There was one noteworthy exception. When a flat, horizontal active surface was chosen, and the GBEM was used,
Fig. 2.11 Rapid field solution convergence of the GBEM on the rectangular domain (flat horizontal active surface) for boundary condition type I (type III is equivalent in this case).
an extraordinary convergence rate was obtained. The data plotted in Figure 2.11 was based on using 6 quadrature points for each boundary element and the first boundary condition choice. Note that for the flat horizontal shape, the first and third boundary condition choices are equivalent. The straight portion of the curve in Figure 2.11 has a slope of -4.97. Two main reasons are suggested for this fast convergence. First, the basis function approximation to the domain shape was exact, excluding roundoff error. The second reason for the fast convergence is that the solution was calculated only over two horizontal surfaces so that the method sidestepped the need to calculate the harder-to-approximate exponential decay along the vertical axis. In fact, the hyperbolic cosine terms in the specialized Green’s functions evaluated to only 4 different values. No other cases of unusually rapid convergence were noted, even when the other methods were applied to this rectangular domain. Note that the other methods still needed to represent the exponentially-decaying solution along the vertical sidewalls.
METHOD COMPARISON: SUMMARY AND CONCLUSIONS

We have formulated a moving boundary problem with an analytical solution that is valid for a variety of domain shapes and boundary conditions. With this comparison tool, we have been able to compare the convergence behavior of several boundary integral formulations against that of a biquadratic FEM formulation. Though we have focused on the differences between these methods, it is apparent that they are generally similar in their abilities to accurately solve Laplace's equation. It is particularly interesting to note that results obtained with small numbers of nodes and few time steps can be within a few percent of the exact solution. However, a relatively large difference was noted between the linear basis function BEM and the other methods, which are based on quadratic interpolation. The quadratic methods converge more quickly, as expected, and they are also computationally more efficient. It remains to be seen if using even higher-order basis functions would result in more accuracy at a given level of computation expense.

Liggett and Salmon (1981) compared using isoparametric cubic spline interpolation with using linear interpolation, and concluded that the analytically formulated linear method was more accurate for a given computation expense. Their comparison was based on two problems, for which error measurements were calculated at selected points on the boundaries. RMS error figures as used in our study are more
reliable measures of convergence because accuracy over an entire surface is assessed. Liggett and Salmon experienced convergence difficulties near the corners of their problem domains with the numerically integrated cubic BEM, and concluded that the accuracy of the quadrature diminished the overall accuracy, even when 16 quadrature points per boundary element were used. In Chapter III, the method of Brebbia (1978) for calculating \( \Theta \) in equation (2.19) is described. We have found this method to be instrumental in obtaining accurate solutions near the domain corners even if as many as 20 quadrature points per boundary element are used. Apparently Liggett and Salmon did not implement this evaluation method.

Other comparisons of boundary element methods using higher-order interpolation were performed by Boissenot, Lachat and Watson (1974), and Lachat and Watson (1976) as reviewed in Chapter I. Neither elastostatic study presented an analysis of error, since in both cases analytical solutions were not available. Based on the approximate agreement of the higher-order methods, quadratic and cubic interpolation appeared to be preferable to linear interpolation from an efficiency standpoint. Neither study lead to a more definite conclusion, and in particular, a quantitative comparison of cubic and quadratic interpolation was lacking. Lachat and Watson employed an indirect evaluation scheme for some of their terms, which is apparently the three-dimensional elastostatic counterpart of Brebbia's method.
The similar accuracy and convergence of the FEM and the BEM formulations with quadratic interpolation holds only for solutions of the field variable. We have shown that FEM solutions for gradients of the field variable were less accurate since they were obtained by differentiating the field solution. This is a fundamental difference of the finite element and boundary element methods. With the BEM, gradients of the field variable are obtained directly and are expected to have convergence rates that are similar to the field solution convergence results we have shown. This difference is particularly important for current distribution problems since movement of the electrode surfaces is in proportion to the local current, i.e., the field gradient.

We have compared the computation costs of these methods, especially the tradeoff between cost and accuracy. For both methods, formulation costs scale as $N^2$, where $N$ is the number of active surface elements. With the BEM, the cost for solving the simultaneous equation set grows as $N^3$ instead of $N^4$ for the FEM. Despite the scaling advantage, the FEM is more economical for solving the field solution at all but the highest accuracies. This is because the BEM formulations devote a much larger portion of their calculation time to formulation. Implementation of faster integration schemes for the BEM would be fruitful, particularly the use of lower order quadrature rules for integrals where the base node is widely separated from the integration contour (Lachat and Watson, 1976).
However, accuracy and economy in solving the field solution is not fully equivalent to accuracy and economy in solving the moving boundary problem. Because error inevitably enters into the calculations, the full accuracy of the field solution at each time step is not required. This was most evident with the boundary method that we formulated using Green's functions that automatically fulfilled selected boundary conditions. When solving for the field solution, the GBEM required up to 32 quadrature points per element to attain better accuracy as more nodes were added, and it stayed less economical than the FEM. However, when the GBEM was used to solve the moving boundary problem, a fixed 4 point quadrature rule was sufficient. With its advantageous cost scaling, the GBEM became more economical than the FEM above a certain level of accuracy. The QBEM method also became more economical than the FEM for the moving boundary problem, but since it required more node points than the GBEM, the accuracy level where it became preferred was somewhat higher.

Another aspect of the comparison is the computer storage requirement. In a fixed amount of computer memory we were able to introduce roughly twice as many surface nodes with the QBEM as compared to the FEM, and thus obtain a more accurate result. The GBEM is even more compact and approximately four times as many surface nodes could be used compared to the FEM. The boundary methods are expected to offer even greater storage advantages for three dimensional problems.
Additional observations concerning the application and particular strengths of these methods are found in Chapter V.
## LIST OF SYMBOLS FOR CHAPTER II

### English Characters

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$A$</td>
<td>formulation matrix</td>
</tr>
<tr>
<td>$C$</td>
<td>the field variable, a potential</td>
</tr>
<tr>
<td>$\mathbf{C}$</td>
<td>a vector of $C$ nodal values</td>
</tr>
<tr>
<td>$\mathbf{e}_x$</td>
<td>unit vector along the $x$ axis</td>
</tr>
<tr>
<td>$\mathbf{e}_y$</td>
<td>unit vector along the $y$ axis</td>
</tr>
<tr>
<td>$E_i$</td>
<td>solution or position error at the node point $(x_i, y_i)$</td>
</tr>
<tr>
<td>$f_1$</td>
<td>a function defined in equation (2.8)</td>
</tr>
<tr>
<td>$f_2$</td>
<td>a function defined in equation (2.8)</td>
</tr>
<tr>
<td>$\mathbf{E}$</td>
<td>a vector resulting from the matrix formulation</td>
</tr>
<tr>
<td>$h$</td>
<td>position of the active surface, $h=h(x,t)$</td>
</tr>
<tr>
<td>$h_0$</td>
<td>initial position of the active surface</td>
</tr>
<tr>
<td>$\mathbf{n}$</td>
<td>normal vector pointing out from $\Omega$</td>
</tr>
<tr>
<td>$n_x$</td>
<td>component of the normal vector along $\mathbf{e}_x$</td>
</tr>
<tr>
<td>$n_y$</td>
<td>component of the normal vector along $\mathbf{e}_y$</td>
</tr>
<tr>
<td>$N$</td>
<td>the number of elements along the active surface</td>
</tr>
<tr>
<td>$r$</td>
<td>radial distance from the point $(x_i, y_i)$</td>
</tr>
<tr>
<td>$t$</td>
<td>time</td>
</tr>
<tr>
<td>$w_i$</td>
<td>weighting function based at the point $(x_i, y_i)$</td>
</tr>
<tr>
<td>$x$</td>
<td>horizontal coordinate</td>
</tr>
<tr>
<td>$x_i$</td>
<td>horizontal coordinate of node $i$</td>
</tr>
<tr>
<td>$\mathbf{X}$</td>
<td>vector of unknown potentials and normal gradients</td>
</tr>
</tbody>
</table>
y  vertical coordinate

\( y_i \)  vertical coordinate of node \( i \)

**Greek characters**

\( \Gamma \)  the boundary surface of the model domain

\( \Gamma_1 \)  horizontal contour topping the finite domain

\( \Gamma_2 \)  symmetry plane surface contour

\( \Gamma_3 \)  active surface contour

\( \delta \)  Dirac delta function

\( \delta_{ij} \)  Kronecker delta function, \( \delta_{ij} = 1 \) if \( i=j \), otherwise \( \delta_{ij} = 0 \)

\( n \)  vertical coordinate on the prototype two-dimensional finite element

\( \Theta_i \)  constant related to the interior angle of the domain at the point \( (x_i, y_i) \)

\( \xi \)  horizontal coordinate on a prototype element

\( \xi_i \)  polynomial basis function centered at nodal point \( (x_i, y_i) \), an explicit function of \( (x, y) \)

\( \Omega \)  two-dimensional region where Laplace's equation applies
CHAPTER III
INFINITE DOMAINS

In this part of the thesis, the comparison problem that was developed in Chapter II is posed on the semi-infinite strip. The three methods to be compared are extensions of the methods that have already been applied to the finite domain. The biquadratic finite element method (FEM) and the general quadratic boundary element method (QBEM) are applied to this problem by the introduction of "infinite elements." The other comparison method is the quadratic boundary element method using a specific Green's function (GBEM) where the Green's function eliminates the need to discretize the infinite parallel sidewalls. Only the deforming surface is discretized; hence, the resulting numerical problem is unusually compact. To complete the comparison, the calculations in Chapter II are shown to be nearly identical to approximating the infinite domain problem by solving it on a truncated domain.

It is helpful to categorize the numerical techniques that have been found most useful for infinite domain problems as follows:

i. application of a finite numerical method on a finite domain

ii. extension of a finite numerical method to an infinite domain
iii. use of a numerical method that is intrinsically unbounded

In the first category, the most common and the simplest approach is to truncate the domain at some large distance. The distance is extended as required until subsequent solutions have converged to the desired level of accuracy. The disadvantage of this approach is that it may require a very large domain and consequently result in a very large numerical problem.

Also in the first category, the unbounded domain may be mapped on to a bounded one. This may be particularly advantageous for Laplace's equation since a conformal mapping will leave the equation unchanged. As an example, the study of periodic steep surface waves by Longuet-Higgins and Cokelet (1976) is naturally set in an infinite half strip defined by the wave surface and two planes of symmetry. The authors map the wave surface to a closed curve with points at infinite depth being mapped to the origin. The BEM was then applied so that the problem was discretized only over the closed curve.

In the second category, one approach is to match the numerical solution at a finite radius with an analytical far-field solution. As an example, Orr et al. (1977) have used this technique to calculate three-dimensional menisci shapes with the FEM. This approach is akin to scaling a problem as having an inner solution valid in the discretized region and an outer solution which is easily computed.
The introduction of "infinite elements" into the FEM as proposed by Ungless (1973) and by Bettess (1977) falls under the second category also. This method introduces into a conventional FEM formulation basis functions that extend to infinity. Despite its recent introduction, the method has been successfully applied in a number of variations (Bettess and Zienkiewicz, 1977; Bettess, 1980; Lynn and Hadid, 1981; Beer and Meek, 1981; Zienkiewicz, Emson and Bettess, 1983) which are explored in more detail in the comparison to follow.

Under the third category of numerical approaches, there is the boundary element method. BEM solutions are superpositions of fundamental solutions of the governing equation which are valid throughout the infinite problem domain. To apply the method, the problem must be posed in a way that eliminates contributions from infinite system boundaries. For example, an integral equation formulation different than that developed here is used to model potential flow around aircraft bodies (Hess, 1976). By the use of superposition, only the aircraft surface is discretized; the infinite medium is implicit. A reflected image of an object may be used to eliminate an infinite halfplane, as shown by Jawson and Symm (1977) in a thermal conduction study of buried cables. A more general approach is to use Green's functions that fulfill the required boundary condition on the infinite surface. Then the resulting integral equation will not require discretization of the infinite surface.

Since the BEM is capable of representing the far
field, coupling the FEM to it presents another means of extending the FEM to the infinite domain (McDonald and Wexler, 1972; Silvester and Hsieh, 1971; Cermak and Silvester, 1968). Zienkiewicz, Kelly, and Bettess (1977), have reviewed this area and discussed how this may be done in a conventional FEM context. Of particular interest is preserving the symmetric matrix formulation of the FEM, when it exists.

MODEL PROBLEM FORMULATION

As before, we are concerned with solving Laplace's equation for the field variable, \( C = C(x,y) \), on a deforming two-dimensional region, \( \Omega \), with boundary surface, \( \Gamma \), described in a rectangular Cartesian coordinate system, \( (x,y) \):

\[
\nabla^2 C = 0, \text{ in } \Omega \quad (3.1)
\]

The lower portion of the domain is unchanged from the finite version, as can be seen in the depiction in Figure 3.1. The region is confined between two vertical sidewalls, \( \Gamma_z \), where a symmetry condition is set.

\[
\mathbf{n} \cdot \nabla C = 0, \text{ on } \Gamma_z, \ x = 0, x = 1, h(x,t) \leq y \leq \infty \quad (3.2)
\]

The bottom deforming surface, \( y = h(x,t) \), still referred to as the active surface, is subject to the Robin condition,

\[
\mathbf{n} \cdot \nabla C = f_1 C + f_2, \text{ on } \Gamma_3, \ y = h(x,t), \ 0 \leq x \leq 1 \quad (3.3)
\]
Fig. 3.1 A schematic depiction of the model problem for the semi-infinite domain.
The functions $f_1$ and $f_2$ are similar to their finite domain counterparts and force the solution to the field equation (3.1) at any time to be

$$C = \cos(\pi x) \exp(-\pi y)$$  \hspace{1cm} (3.4)

The active surface moves in the vertical direction at a rate given by the local value of the field.

$$\frac{\partial h}{\partial t} = C(x, h(x, t))$$  \hspace{1cm} (3.5)

The initial configuration of the active surface is still arbitrary and must be specified.

$$h(x, t=0) = h_0(x)$$  \hspace{1cm} (3.6)

Whereas formerly the domain was closed at the top by a horizontal boundary, it now extends along the $y$ coordinate to positive infinity, forming a semi-infinite strip. The field variable is constrained to vanish as the ordinate becomes infinite.

$$C = 0; \quad y = \infty, \quad 0 \leq x \leq 1$$  \hspace{1cm} (3.7)

The three choices of the active surface boundary condition are preserved for the infinite domain problem. Their form reflects the simpler exponential form of the field solution. The selections of the functions $f_1$ and $f_2$ are:
Type I - inhomogeneous

\[ f_1 = 0 \]
\[ f_2 = -\pi [ n_x \sin(\pi x) \exp(-\pi y) + n_y \cos(\pi x) \exp(-\pi y) ] \quad (3.8a) \]

Type II - mixed

\[ f_1 = -\pi n_y \]
\[ f_2 = -\pi n_x \sin(\pi x) \exp(-\pi y) \quad (3.8b) \]

Type III - mixed

\[ f_1 = -\pi n_x \tan(\pi x) \]
\[ f_2 = -\pi n_y \cos(\pi x) \exp(-\pi y) \quad (3.8c) \]

In these expressions \( n_x \) and \( n_y \) indicate components of the active surface normal, \( \hat{n} = n_x \hat{e}_x + n_y \hat{e}_y \), that points outward from \( \Omega \).

The correct position of the active surface is obtained from integrating equation (3.5) using the field solution, equation (3.4).

\[ \frac{\partial h}{\partial t} = \cos(\pi x) \exp(-\pi h) \quad (3.9) \]

Upon integration the result is

\[ h(x,t) = -\frac{1}{\pi} \ln \left( \exp \left[ \pi h(x,0) \right] + (\pi t) \cos(\pi x) \right) \quad (3.10) \]

The accelerating unstable nature of the problem is apparent
from considering the position at $x=1$, $h(1,t)$. In the finite span of time, $t = \exp[nh(1,0)]/n$, the boundary reaches negative infinity.

NUMERICAL DISCRETIZATION

As in Chapter II, equation (3.5) is integrated for the position of the active surface using an Euler-predictor, trapezoid-corrector finite difference method. The thrust of the comparison study in this chapter is on the methods used to discretize the spatial domain, and their ramifications for the moving boundary problem.

**BEM Implementation**

The BEM matrix formulation,

$$A X = F$$

$$= - -$$

differs from Chapter II, reflecting the absence of the surface $\Gamma_1$. The terms become

$$X_i \equiv C_i$$

$$F_i \equiv \int_{\Gamma_3} \omega_i f_2 d\Gamma$$

(3.12)  
(3.13)
\[ A_{i,j} = \Theta_i(\delta_{i,j}) + \int_{\Gamma_2 \cup \Gamma_3} \mathbf{n} \cdot \nabla w_i \, dr - \int_{\Gamma_3} f_i \, dr \]  
\hspace{0.5cm} (3.14)

For the general QBEM case the appropriate fundamental solution is

\[ w_i = -\frac{1}{2\pi} \ln(r), \]  
\hspace{0.5cm} (3.15)

and for the GBEM case, is

\[ w_i = -\frac{1}{4\pi} \ln \left( \frac{\cosh[\pi(y-y_i)] - \cos[\pi(x+x_i)]}{\cosh[\pi(y-y_i)] - \cos[\pi(x-x_i)]} \right) \]  
\hspace{0.5cm} (3.16)

The value of \( \Theta_1 = \Theta_1(x_1, y_1) \) in equation (3.14) is equal to the interior angle of the domain at the point \((x_1, y_1)\). As described by Brebbia (1978), the direct calculation of \( \Theta_1 \) is avoided by calculating it by difference. Prior to the substitution of any boundary conditions, the BEM integral equation for the field variable, \( C \), is written as the matrix equation,

\[ \sum_{j} H_{i,j} C_j = \sum_{j} G_{i,j} (dC/dn)_j \]  
\hspace{0.5cm} (3.17)

where

\[ H_{i,j} \equiv (\delta_{i,j}) \Theta_i + \int_{\Gamma} \mathbf{n} \cdot \nabla w_i \, dr \]  
\hspace{0.5cm} (3.18)

and

\[ G_{i,j} \equiv \int_{\Gamma} w_i w_j \, dr \]  
\hspace{0.5cm} (3.19)
For the usual finite domain, if the set of $C_j$ is constant, then the $(dC/dn)_j$ should each be zero. This requires that the matrix rows of $H$ sum to zero:

$$
\sum_j H_{ij} = 0 ,
$$

(3.20)

and so $H_i$ can be found by difference. This evaluation method was used with all of the BEM formulations on the finite domain and with the QBEM on the infinite domain.

The natural alternative is to calculate $\theta_i$ directly, using the basis function representation of the surface. Surprisingly, direct calculation of $\theta_i$ resulted in less accurate results unless the $H_i$ integrals were done very accurately. Other alternatives for the evaluation of $\theta_i$ were studied in the development of the limiting current BEM model described in Chapter IV of this thesis. The direct evaluation of $\theta_i$ was especially simple because it was only needed on linear boundaries or square corners. The $H_i$ type integrals were computed analytically even though quadratic basis functions were used since straight elements with centered midnodes made analytical evaluation possible. If $\theta_i$ was evaluated directly and used with $H_i$ terms that were found by numerical integration, unrealistic variations in the current density resulted, even if as many as 20 quadrature points per element were used. The inaccuracy was most pronounced near the domain corners where the $H_{ij}$ integrands varied rapidly. However, directly evaluated $\theta_i$ terms used with analytically
evaluated \( H_i \) terms gave accurate results. If \( \theta_i \) is evaluated by difference, as shown above, accurate results are obtained irrespective of how the \( H_i \) terms are evaluated. Hence the indirect evaluation of \( \theta_i \) is a means of compensation for the error in evaluating the \( H_i \) terms.

A change is required to make the indirect calculation method work for the GBEM on the infinite domain. When applied to the GBEM calculation on the finite domain equation (3.20) results in

\[
H_i = -\sum_{j,j \neq i} \left[ \int_{\Gamma_1} \bar{x}_j (\mathbf{n} \cdot \nabla \mathbf{w}_i) \, d\Gamma + \int_{\Gamma_2} \bar{x}_j (\mathbf{n} \cdot \nabla \mathbf{w}_i) \, d\Gamma \right] \tag{3.21}
\]

For the GBEM on the infinite domain, the only discretized surface is the deforming curve, \( \Gamma_2 \), since \( (\mathbf{n} \cdot \nabla \mathbf{w}_i) \) is zero along the parallel sidewalls, \( \Gamma_1 \). The integral over \( \Gamma_1 \) has been moved to infinity and is "missing" from the calculation. To apply the indirect calculation procedure, the contribution from this missing surface has to be made up. To do this we imagine a virtual surface closing the domain. The missing contribution is,

\[
\sum_j \int_{\Gamma_\infty} \bar{x}_j (\mathbf{n} \cdot \nabla \mathbf{w}_i) \, d\Gamma = \int_{\Gamma_\infty} (\mathbf{n} \cdot \nabla \mathbf{w}_i) \, d\Gamma \tag{3.22}
\]

since \( \sum \bar{x}_j = 1 \) at any point. This integral is evaluated using the divergence theorem and realizing from symmetry considerations that the virtual surface receives half of the flux to a unit sink, \( \nabla^2 \mathbf{w}_i = -\delta(x-x_i)\delta(y-y_i) \).
\[ \int_{\Gamma_\infty} (\mathbf{n} \cdot \nabla w_i) d\mathbf{r} = \frac{1}{2} \int_{\Omega} \nabla^2 w_i \, d\Omega = -1/2 \]  

(3.23)

The rules for the infinite domain BEM calculation can now be stated.

When \( i \neq j \), \( H_{ij} \) is found by integration,

\[ H_{ij} = \int_{\Gamma} \mathbf{n} \cdot (\nabla w_i) \, d\mathbf{r} \]  

(3.24a)

When \( i = j \), \( H_{ii} \) is found by difference in the usual way for the GBEM,

\[ H_{ii} = -\Sigma_{j,j \neq i} H_{ij} \]  

(3.24b)

But for the GBEM, \( H_{ii} \) is found by difference as,

\[ H_{ii} = -\Sigma_{j,j \neq i} H_{ij} + 1/2 \]  

(3.24b)

**BEM Infinite Elements**

Application of the QBEM requires integration over the parallel infinite sidewalls, \( \Gamma_2 \). Usual finite boundary elements are defined along the active surface and along the parallel sidewalls up to an arbitrary height. The node structure corresponds to that used in the finite domain problem with the exception that no closing horizontal boundary is defined at the height of the topmost nodes (Figure 3.1).
The procedure shown by Bettess (1980) described below, is used to extend the topmost boundary elements on each of the sidewalls to infinity.

The quadratic polynomial basis functions are used to establish a mapping from an element in the global cartesian coordinate system to a prototype element with local coordinates that are well-suited for numerical quadrature. This has been discussed in Appendix B. Usually this mapping is only used inside of the original element. However, the mapping is defined everywhere, and in particular can be used to extend the element to infinity, subject to some restrictions which insure that the mapping is unique, i.e., that the element does not fold over on itself. For a quadratic boundary element, it suffices that the interior node be placed in the middle of the element (Bettess, 1980). Thus the usual finite basis functions can be used on the infinite element to obtain mapping related quantities as is done on the finite elements, eg.,

\[ y = \sum_j \bar{x}_j y_j \quad \text{and} \quad \frac{\partial y}{\partial \zeta} = \sum_j y_j \frac{\partial \bar{x}_j}{\partial \zeta}. \quad (3.25) \]

However the usual finite basis functions are not appropriate to interpolate the field variable as infinity is approached. An infinite basis function is constructed by multiplying the finite basis function by a decay function. The decay function is chosen so that the infinite basis function vanishes at infinity, and the integrals needed in the formulation are
finite.

The basis function notation, $\bar{\varepsilon}_i$, that has been used, refers to the global interpolating function that surrounds nodal point $i$. It is nonzero on the elements that intersect at nodal point $i$. Depending on the type of interpolation, it may be nonzero on more distant elements. The global basis function, $\bar{\varepsilon}_i$, is constructed from local basis functions that are defined on each element. Since on a particular element, a local basis function is identified with only one global basis function, the distinction between local and global is often left ambiguous. Let $\phi_i$ signify one of these elemental basis functions. A different subscript is chosen since the global indexing will not in general correspond to the element indexing, because each element has only a few nodes. On an infinite element, decay functions are defined for each local basis function. Thus on a particular infinite element, the global basis function that is used to interpolate the field variable is given as

$$
\bar{\varepsilon}_i = \phi_i D_i.
$$

(3.26)

By requiring the decay function, $D_i$, to evaluate to unity at the node point $i$, values of the field variable at the node point will only depend on the nodal value, as usual:

$$
\bar{\varepsilon}_i(x_i, y_i) = \delta_{ij},
$$

(3.27)

$$
C(x_i, y_i) = \sum_j \bar{\varepsilon}_j C_j = C_i
$$

(3.28)
TABLE 3.1

RECIPROCAL DECAY FUNCTIONS FOR QUADRATIC ELEMENTS

<table>
<thead>
<tr>
<th>Node</th>
<th>Quadratic Basis Function</th>
<th>Decay Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\xi = -1$</td>
<td>$\phi_1 = \xi(\xi - 1)/2$</td>
<td>$D_1 = \left(\frac{-1 - \xi^0}{\xi - \xi^0}\right)^3$</td>
</tr>
<tr>
<td>$\xi = 0$</td>
<td>$\phi_2 = 1 - \xi^2$</td>
<td>$D_2 = \left(\frac{-\xi^0}{\xi - \xi^0}\right)^3$</td>
</tr>
<tr>
<td>$\xi = +1$</td>
<td>$\phi_3 = \xi(\xi + 1)/2$</td>
<td>$D_3 = \left(\frac{+1 - \xi^0}{\xi - \xi^0}\right)^3$</td>
</tr>
</tbody>
</table>

$\xi$ is the local element coordinate as defined in Appendix B.
There are relatively few restrictions placed on the choice of decay function other than the need for certain terms to be bounded to insure the validity of Green's theorem (Bettess, 1980). Exponential type decay functions and reciprocal type decay functions have been used successfully. Here we choose to use the reciprocal type decay functions shown in Table 3.1. The form of the decay functions is appropriate to extend the field variable interpolation beyond the element end in the positive $\xi$ direction. The constant $\xi^0$ appearing in the decay functions (Table 3.1) determines where the decay originates in the local element coordinate system. With the decay in the positive $\xi$ direction, $\xi^0$ will be beyond the element edge in the opposite direction, i.e., $\xi^0 < -1$. The choice of $\xi^0$ is a degree of freedom that will be discussed with the other numerical results. If the element is to be extended in the direction of negative $\xi$, the functions shown must be modified accordingly.

Extending one side of the finite element to infinity requires that a suitable quadrature rule be employed. Standard Gauss-Legendre quadrature points, $P_i$, and weights, $W_i$, are based on the interval $[-1,1]$.

$$\int_{-1}^{+1} f(z) dz \approx \sum W_i f(P_i) \quad (3.29)$$

They can be made to apply to $[-1,\infty]$ by a suitable manipulation. Since
\[ \int_{-1}^{1} f(z) \, dz = \int_{-1}^{+1} f \left( \frac{2x}{1-x^2} \right) \frac{2 \, dx}{(1-x)^2}, \quad (3.30) \]

The new quadrature points and the new quadrature weights are related to the standard values by

\[ P_{\text{new}} = 2 \, P_{\text{old}} / (1 - P_{\text{old}}), \]

\[ W_{\text{new}} = 2 \, W_{\text{old}} / (1 - P_{\text{old}})^2 \quad (3.31) \]

The above provisions for infinite elements are readily incorporated into the finite domain computer code. In fact, the quadratic BEM program listing in the appendix will calculate results for the finite or infinite domain depending on the user’s input data. Similarly the GBEM computer program is designed to handle both the infinite and finite cases. Thus all other aspects of the calculations are handled as described in Chapter II, including the number of quadrature points used per element, and the treatment of logarithmic singularities (Appendix D).

As an aside, a new infinite element described by Bettess (1983), appears to be even easier to implement and provides a similar functional representation that decays with the reciprocal of distance. The basic idea is to map the infinite region to the standard prototype finite element, so that standard quadrature points and weights can be used. Computer program changes would primarily concern the introduction of new mapping functions.
FEM Implementation

Infinite elements are introduced into the FEM almost identically as described above; the usual basis functions are used to extend the mapping beyond the conventional finite element boundary, and decay functions are used to help interpolate the field variable. So that the infinite mapping is well-defined, the vertical sidewalls of the quadratic finite elements are constructed to be parallel to each other and parallel to the vertical sidewalls of the domain. Also, the interior node is centrally located and nodes interior to an element edge are centered on the edge.

Normally the bi-quadratic basis functions are obtained as the products of the one-dimensional quadratic functions identical to those used in the BEM formulation. This is still the case for the infinite elements. When interpolating the field variable, the one-dimensional decay functions, presented above, are associated with the basis functions along the vertical (infinite) coordinate. The usual quadratic functions are used to interpolate along the horizontal (finite) coordinate. Similarly the quadrature points and weights for integrating the two-dimensional element area are obtained as the matrix product of the appropriate one-dimensional quantities.

The residual equations are stated in matrix form as,

\[
A C = F . \quad (3.32)
\]
The individual terms appear the same as they are for the finite domain case, but now it is understood that the global basis functions can have local forms with decay functions:

\[
A_{ij} = \int_{\Omega} \nabla \bar{\psi}_i \cdot \nabla \bar{\psi}_j \, d\Omega - \int_{\Gamma_3} \bar{\psi}_i \bar{\psi}_j \, f_1 \, d\Gamma \quad (3.33)
\]

\[
F_i = \int_{\Gamma_3} \bar{\psi}_i \, f_2 \, d\Gamma, \quad C_i \equiv \text{nodal value of } C \quad (3.34,35)
\]

With the FEM, it is necessary to evaluate derivatives that are related to the field variable. When this is done on the infinite element, the chain rule is applied to equation (3.26), eg.,

\[
\frac{\partial \bar{\psi}_i}{\partial \zeta} = \frac{\partial \phi_k}{\partial \zeta} \frac{\partial D_k}{\partial \zeta} + \phi_k \frac{\partial D_k}{\partial \zeta} . \quad (3.36)
\]

As with the BEM methods, these changes are easily incorporated into the existing finite domain computer program, and other aspects of the calculation are handled as in Chapter II.

RESULTS AND DISCUSSION

A particular case of the model problem was defined to serve as a starting point for the method comparison. Hence unless specified otherwise the initial shape of the active surface was,

\[
h(x,t=0) = 0.15 [1 - \cos(\pi x)], \quad 0 \leq x \leq 1 \quad (3.37)
\]
the third boundary condition choice (equation 3.8c) was used, and the decay function origin, $\xi^*$ (Table 3.1), needed for the QBEM and FEM infinite elements, was set to $-1000$. These choices correspond directly to the model problem of Chapter II, where the domain was closed at $y=4$. The height $y=4$ was sufficiently large so that the finite domain calculations were nearly identical to approximation of the infinite domain problem by solving it on a truncated domain. The active surface boundary conditions imposed in the finite domain problem (equation 2.8a,b,c) differed from the corresponding infinite domain conditions (equation 3.8a,b,c) by no more than $10^{-8}$. Similarly, the field solution for the finite domain (equation 2.5) differed from the solution for the infinite domain (equation 3.4) by no more than $10^{-8}$ along the active surface, where RMS solution errors were calculated.

Convergence of the Field Solution

Figure 3.2 shows the dependence of the error in the field solution as the element mesh was refined. The measure of error was the root-mean-square difference (equation 2.36) in the computed field solution and equation (3.4) at the node points along the active surface. The QBEM results were obtained using 6 quadrature points per element, and $3 \times 3$ quadrature points per element were used for the FEM. As in Chapter II, the use of more quadrature points with the GBEM
Fig. 3.2 Convergence of the methods to the field solution on the initial domain. The number of other surface elements was kept in proportion to the number of active surface elements. Slopes are based on the three points using the most elements.
lead to better accuracy upon mesh refinement, so that from 4 to 32 quadrature points per element were used.

Figure 3.2 is to be contrasted with Figure 2.3, which shows the convergence curves for the finite domain problem. The accuracy of the FEM calculations was nearly the same for both problems. Consistent convergence with a slope of $-3.61$ was obtained. The GBEM calculations on the infinite domain resulted in substantially less error than on the finite domain. Consequently, for the infinite problem the GBEM calculations were almost always the most accurate for a specified number of active surface elements.

It is apparent from Figure 3.2 that the error in the QBEM calculations did not continue to decrease as more elements were introduced. For small numbers of elements, the QBEM convergence behavior was the same as it was on the finite domain. However, on the infinite domain, the use of 16 or 32 active surface elements gave solutions that were less accurate than solutions calculated with 4 active surface elements.

By examining the infinite element approximation, insight into the persistent error of the QBEM was obtained. Unlike with the FEM, the errors in the QBEM calculations were sensitive to $\xi^o$, the origin for the infinite element decay function (Table 3-1). Figure 3.3 shows the results of keeping either 4 or 8 active surface elements and varying $\xi^o$. Table 3.2 gives sample results for 6 active surface elements. The best accuracy was obtained when $\xi^o$ was large in magnitude. Increasing the magnitude of $\xi^o$ decreased the values of the
Fig. 3.3 RMS error of the initial field solution versus the infinite element decay origin.
### TABLE 3.2

RMS FIELD SOLUTION ERRORS WITH 6 ACTIVE SURFACE ELEMENTS FOR DIFFERENT ACTIVE SURFACE BOUNDARY CONDITIONS AND CHOICES OF $\xi^*$

#### Finite Element Method

<table>
<thead>
<tr>
<th>$\xi^*$</th>
<th>B.C. I</th>
<th>B.C. II</th>
<th>B.C. III</th>
</tr>
</thead>
<tbody>
<tr>
<td>-2</td>
<td>5.6945E-4</td>
<td>8.1592E-2</td>
<td>6.5008E-4</td>
</tr>
<tr>
<td>-5</td>
<td>7.3022E-4</td>
<td>5.1837E-3</td>
<td>6.4396E-4</td>
</tr>
<tr>
<td>-10</td>
<td>7.5817E-1</td>
<td>5.1830E-3</td>
<td>6.4394E-4</td>
</tr>
<tr>
<td>-20</td>
<td>1.8943E+3</td>
<td>&quot;</td>
<td>&quot;</td>
</tr>
<tr>
<td>-50</td>
<td>2.9721E+7</td>
<td>&quot;</td>
<td>&quot;</td>
</tr>
<tr>
<td>-100</td>
<td>2.5567E+9</td>
<td>&quot;</td>
<td>&quot;</td>
</tr>
<tr>
<td>-500</td>
<td>1.5050E+9</td>
<td>&quot;</td>
<td>&quot;</td>
</tr>
<tr>
<td>-1000</td>
<td>2.5197E+9</td>
<td>&quot;</td>
<td>&quot;</td>
</tr>
</tbody>
</table>

#### Quadratic Boundary Element Method

(6 quadrature points/element)

<table>
<thead>
<tr>
<th>$\xi^*$</th>
<th>B.C. I</th>
<th>B.C. II</th>
<th>B.C. III</th>
</tr>
</thead>
<tbody>
<tr>
<td>-2</td>
<td>singular</td>
<td>2.9177E-1</td>
<td>6.1266E-2</td>
</tr>
<tr>
<td>-5</td>
<td>&quot;</td>
<td>2.2575E-1</td>
<td>3.0619E-2</td>
</tr>
<tr>
<td>-10</td>
<td>&quot;</td>
<td>1.6530E-1</td>
<td>1.7160E-2</td>
</tr>
<tr>
<td>-20</td>
<td>&quot;</td>
<td>9.6021E-2</td>
<td>7.9678E-3</td>
</tr>
<tr>
<td>-50</td>
<td>&quot;</td>
<td>2.7602E-2</td>
<td>2.0734E-3</td>
</tr>
<tr>
<td>-100</td>
<td>&quot;</td>
<td>7.7415E-3</td>
<td>8.6100E-4</td>
</tr>
<tr>
<td>-500</td>
<td>&quot;</td>
<td>3.9360E-4</td>
<td>6.1262E-4</td>
</tr>
<tr>
<td>-1000</td>
<td>&quot;</td>
<td>5.3179E-4</td>
<td>6.0502E-4</td>
</tr>
</tbody>
</table>

#### Specialized Green’s Function Boundary Element Method

(6 quadrature points/element, there is no $\xi^*$)

<table>
<thead>
<tr>
<th></th>
<th>B.C. I</th>
<th>B.C. II</th>
<th>B.C. III</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1.3402E-5</td>
<td>3.1932E-3</td>
<td>3.2919E-4</td>
</tr>
</tbody>
</table>
decay functions and diminished the magnitude of the numerical contributions from the infinite elements. Thus, better accuracy was found under conditions where the contributions from the infinite elements were relatively less important. Results for the FEM calculation using 6 active surface elements are also shown in the same figure. The strong contrast in the sensitivities of the methods to $\xi^*$ parallels the contrast found in their convergence behavior.

With the FEM, integral terms from the infinite area elements only affected residual equations that involved nodal solution values from the surrounding elements (it is a banded matrix problem). With the QBEM, integral terms over the infinite boundary elements contributed to every term in the equation set that determined the field solution. Since the infinite element was only an approximate means of modeling the far field, and did not have the asymptotic behavior of the analytical solution (equation 3.4), its use necessarily introduced some error into the numerical methods. In view of the different ways in which the infinite element contributions are incorporated, it is not surprising that the QBEM calculations were more sensitive to $\xi^*$, and sometimes less accurate.
Choice of Boundary Condition

The convergence results shown in Figs. 3.2 and 3.3 were based on the third choice of boundary condition (equation 3.8c) for the active surface. Convergence results for all of the boundary condition choices (equation 3.8) are plotted in Figure 3.4. There are no curves for the first boundary condition with the QBEM or the FEM. The QBEM calculations were singular, and the FEM results at $\xi_0=-1000$ were so inaccurate that they were offscale with respect to Figure 3.4. The value $\xi_0=-1000$ was used because it gave better results for the QBEM with the other boundary conditions. The QBEM and GBEM results are for 6 quadrature points per element. This resulted in slower convergence for the GBEM than shown in Figure 3.2 where up to 32 quadrature points were used. As with the finite domain problem (Fig. 2.10), the GBEM calculations showed rapid convergence with the first boundary condition without use of additional quadrature points.

The second boundary condition gave results that were similar to the results obtained using the third boundary condition. As previously discussed, the accuracy of the QBEM calculations was sensitive to the choice of $\xi_0$ and eventually deteriorated as more elements were used, but the FEM and the GBEM worked reliably. Sample field solution results for all three boundary condition choices are shown in Table 3.2. Even though the QBEM solutions depended on $\xi_0$, the second and third boundary conditions always resulted in usable approximations.
Fig. 3.4 Convergence of the methods for the initial field solution as a function of the active surface boundary condition choice (see text). Six quadrature points per element are used for the QBEM and the GBEM.
of the true field solution. In contrast, use of the first boundary condition resulted in matrix problems that were singular with the QBEM and poorly conditioned with the FEM. The FEM solutions based on the first boundary condition were often grossly inaccurate.

The first boundary condition is an inhomogeneous condition on the normal gradient. With this choice, the Dirichlet condition at infinity \((C=0, y=\infty)\) is the only boundary condition involving the potential. This condition anchors the potential level - it determines the uniqueness of the solution. For a numerical method to work with the first boundary condition choice, it must incorporate the Dirichlet condition at infinity as reliably as it incorporates the other boundary conditions. Otherwise, a Neumann problem results where the solution is determined only to within an arbitrary constant (Jawson and Symm, 1977). The numerical difficulties with the QBEM and the FEM were from failure of the infinite element approximation to impose the Dirichlet condition. The singular matrices of the QBEM showed that the solution was not uniquely determined. With the FEM, examination of the pivots of the factored \(A\) matrix (equation 3.29) showed that the matrix was nearly singular. The FEM calculations were sensitive to the decay origin \(\xi^o\), and when erroneous, the calculated solutions would approximate the true solution (equation 3.4) displaced by an additive constant. The GBEM properly imposed the homogeneous Dirichlet condition at infinity, and accurate results were consistently obtained.
The second and third choices of boundary condition are Robin conditions that link the potential to the normal gradient. This resulted in better conditioned problems that did not depend as crucially on the infinite element approximation. In general, the convergence results resemble those obtained on the finite domain problem (Fig. 2.10) except for the problems with the infinite element approximation already noted.

Computation Cost of the Field Solution

The cost of computing the field solution is shown in Figure 3.5. The computation cost measure was the elapsed virtual CPU time on a Honeywell 6180 mainframe (explanation in Chapter II). Points on each curve in Figure 3.5 depict the cost versus accuracy tradeoff that was found as more active surface node points were introduced, and the number of other elements was kept in proportion. The GBEM yielded the most accurate field solution results for a given computation expense, over the entire accuracy range. This was true even though up to 32 quadrature points per element were used in the calculation of the integrals. The FEM calculations proved to be more economical than the QBEM calculations, especially at higher node point densities when the accuracy of the QBEM deteriorated. As in Chapter II, the QBEM and FEM computer programs were compiled with an optimization option, but the GBEM program was not, so a potential 17% reduction in the cost
Fig. 3.5 Comparison of CPU time versus accuracy for obtaining the initial field solution (Honeywell 9180 mainframe).
of the GBEM is estimated.

**Storage Requirement**

The greater economy of the GBEM is due in part to the relatively small matrix problem that resulted from discretizing only the deforming surface. The storage requirements for the methods are shown in Table 3.3. When 16 surface elements are used, the GBEM matrix is less than 2% of the size of the FEM matrix, and less than 12% of the size of the QBEM matrix.

**Integration in Time**

The infinite domain problem is virtually identical to the finite domain problem from the standpoint of choosing an appropriate time step size for the Euler-predictor, trapezoid-corrector integration algorithm. It was concluded in Chapter II that the appropriate time step size depended on the refinement of the element mesh. Keeping the time step size proportional to the node-to-node spacing was found to be computationally efficient for obtaining solutions over a broad range of accuracies. This rule was used to obtain the results shown in Figure 3.6, where the use of 8 active surface elements corresponded to the time step size of 0.05. The error in the position of the active surface at t=0.6 is plotted against the CPU time required for the calculation.
### TABLE 3.3
INFINITE DOMAIN PROGRAM STORAGE REQUIREMENTS

<table>
<thead>
<tr>
<th>Active Surface Elements</th>
<th>FEM Total Nodes</th>
<th>Matrix Storage</th>
<th>QBEM Total Nodes</th>
<th>Matrix Storage</th>
<th>GBEM Total Nodes</th>
<th>Matrix Storage</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>25</td>
<td>206</td>
<td>13</td>
<td>169</td>
<td>5</td>
<td>25</td>
</tr>
<tr>
<td>4</td>
<td>81</td>
<td>1162</td>
<td>25</td>
<td>625</td>
<td>9</td>
<td>81</td>
</tr>
<tr>
<td>6</td>
<td>169</td>
<td>3446</td>
<td>37</td>
<td>1369</td>
<td>13</td>
<td>169</td>
</tr>
<tr>
<td>8</td>
<td>289</td>
<td>7634</td>
<td>49</td>
<td>2401</td>
<td>17</td>
<td>289</td>
</tr>
<tr>
<td>16</td>
<td>1089</td>
<td>54946</td>
<td>97</td>
<td>9409</td>
<td>33</td>
<td>1089</td>
</tr>
<tr>
<td>32</td>
<td>4225</td>
<td>416066</td>
<td>193</td>
<td>37249</td>
<td>65</td>
<td>4225</td>
</tr>
<tr>
<td>64</td>
<td>16641</td>
<td>3236482</td>
<td>385</td>
<td>148225</td>
<td>129</td>
<td>16641</td>
</tr>
</tbody>
</table>

For the FEM and the QBEM, the number of other elements is kept in proportion to the number of active surface elements.

Matrix Storage Requirement Formulae:

- FEM = \(12N^3 + 22N^2 + 10N + 2\)
- QBEM = \(36N^2 + 12N + 1\)
- GBEM = \(4N^2 + 4N + 1\)

\(N = \text{the number of active surface elements}\)
Fig. 3.6 Comparison of CPU time versus accuracy for the error in the position of the active surface as the element mesh and time step are refined in tandem. The accumulated position error at $t=0.6$ is shown. The ratio of time step to node spacing is held constant.
The results in Figure 3.6 show that the GBEM was the most economical method for the moving boundary problem. It held a consistent advantage over the other methods, even at high accuracy, where its relative advantage narrowed in the field solution comparison (Figure 3.5). This improved performance of the GBEM was primarily the consequence of using only 4 quadrature points per boundary element for the moving problem, whereas additional quadrature points were used to obtain the field solution comparison. Two other contributing factors to the relative economy of the GBEM were that it required less node rearrangement at each time step, and that there were no matrix contributions from fixed elements to be read from storage. The results of Figure 3.6 may also be compared with the similar moving boundary problem results on the finite domain, Figure 2.9. The improved performance of the GBEM on the infinite domain was both because it was more accurate, and because it had fewer nodes and required less computation.

The performance of the GBEM on the moving boundary problem (Figure 3.6) was consistent with the convergence difficulty that was obvious in the field solution comparison (Figure 3.5). When few elements were used, the convergence of the method was similar to that found on the finite domain (Figure 2.9). However, as more elements were used, the accuracy of the method deteriorated, so that the best economy was achieved with 4 active surface elements. The FEM calculations showed consistent convergence and were nearly
identical to the results found on the similar finite domain problem (Figure 2.9).

SUMMARY AND CONCLUSIONS

The comparison problem that was developed in Chapter II has been extended to the semi-infinite strip. The use of infinite elements was found to be a convenient means of applying the Galerkin finite element method and the usual boundary element method based on the free space Green's function. The infinite elements worked reliably with the FEM as long as the Neumann choice of boundary condition for the active surface (equation 3.8a) was avoided. This choice made the calculations depend critically upon the boundary condition at infinity and lead to an ill-conditioned problem for the potential field. Because of the domain dimensions that were chosen, the problem solved in Chapter II was equivalent to approximation of the infinite problem by domain truncation. The infinite element calculations did not offer superior accuracy or economy than simply truncating the domain. Problems where infinite elements have been found superior to truncation include loading on an elastic half-space (Lynn and Hadid, 1981; Zienkiewicz et al., 1983) and viscous flow around a cylinder (Bettess, 1977). In both of those problems, decay of the field variable with distance was more gradual than in this problem.
Use of infinite elements with the BEM was less reliable. The accuracy was sensitive to the pole location for the reciprocal decay functions, and worsened as more elements were introduced. The full matrix structure of the BEM makes it less suitable for the use of infinite elements since their contributions affect every term in the residual equations.

The boundary element method based on a specialized Green's function was more accurate, required less computation work, and used less storage than the infinite element methods or the methods applied to the finite domain. It was the only method for the semi-infinite domain that satisfied the far field conditions and worked reliably with the Neumann boundary condition for the active surface. Because only the moving surface was discretized, the numerical representation was far more compact than with the other methods. The formulation developed here is applicable to solving Laplace's equation on two-dimensional, periodic (or reflectively symmetric) domains. The method can be extended to other two- and three-dimensional problems where suitable Green's functions can be derived in closed form. In addition to Laplace's equation, this includes the Helmholtz equation (Brebbia and Walker, 1980) and the transient diffusion equation (Wrobel and Brebbia, 1981; Shaw, 1974; Chang et al., 1973) with simple boundary conditions on the infinite surfaces.
LIST OF SYMBOLS FOR CHAPTER III

English Characters

A = formulation matrix

C the field variable, a potential

C a vector of C nodal values

Dₙ decay function defined in terms of the local element coordinate

gₓ unit vector along the x axis

gᵧ unit vector along the y axis

Eᵣ solution or position error at the node point (xᵣ,yᵣ)

f₁ a function defined in equation (3.8)

f₂ a function defined in equation (3.8)

E a vector resulting from the matrix formulation

Gₙ matrix terms defined by equation (3.19)

h position of the active surface, h=h(x,t)

h₀ initial position of the active surface

Hₙ matrix terms defined by equation (3.18)

n normal vector pointing out from Ω

nₓ component of the normal vector along eₓ

nᵧ component of the normal vector along eᵧ

N the number of elements along the active surface

Pᵢ coordinate of a Gauss-Legendre quadrature point (one-dimensional)
\( r \) radial distance from the point \((x_i, y_i)\)
\( t \) time
\( w_i \) weighting function based at the point \((x_i, y_i)\)
\( W^i \) Gauss-Legendre quadrature weight (one-dimensional)
\( x \) horizontal coordinate
\( x_i \) horizontal coordinate of node \( i \)
\( X \) vector of unknown potentials and normal gradients
\( y \) vertical coordinate
\( y_i \) vertical coordinate of node \( i \)

**Greek characters**

\( \Gamma \) the boundary surface of the model domain
\( \Gamma_h \) horizontal contour topping the finite domain
\( \Gamma_s \) symmetry plane surface contour
\( \Gamma_a \) active surface contour
\( \Gamma_\infty \) an imaginary surface contour analogous to \( \Gamma_h \)
\( \delta \) Dirac delta function
\( \delta_{ij} \) Kronecker delta function, \( \delta_{ij} = 1 \) if \( i=j \), otherwise \( \delta_{ij} = 0 \)
\( n \) vertical coordinate on the prototype two-dimensional finite element
\( \theta_a \) constant related to the interior angle of the domain at the point \((x_i, y_i)\)
\( \xi \) horizontal coordinate on a prototype element
\( \xi^a \) local coordinate of pole position for the decay functions
\( \phi_k \) polynomial basis function on the prototype element, an explicit function of the local variable(s) \( \xi \) or \( (\xi, \eta) \)

\( \phi_i \) polynomial basis function centered at nodal point \((x_i, y_i)\), an explicit function of \((x, y)\)

\( \Omega \) two-dimensional region where Laplace's equation applies
CHAPTER IV

MASS TRANSFER ANALYSIS OF ELECTRODEPOSITION THROUGH MASKS

In this chapter a tertiary current distribution model is developed for predicting two-dimensional shape change during electrodeposition through polymeric masks. The cathode geometry studied consists of a parallel array of microscopic, rectangular trenches, the trench bottoms being conductive metal, and the trench walls being formed from insulating mask material. Concentration polarization, activation polarization, and a stagnant diffusion layer treatment of convective mass transport are incorporated into the model equations. A quadratic BEM formulation and Euler-predictor, trapezoid-corrector time integration are used to calculate the deposit shape history as it depends on geometrical parameters, the level of convection, and the degree of polarization.
ELECTRODEPOSITION THROUGH MASKS

Electrodeposition through masks involves the following steps which are depicted in Figure 4.1 (Romankiw, 1979):

(i) metalization of the substrate

Substrate refers to the material upon which a deposited metal pattern is desired, e.g., a semiconductor wafer. For the electrodeposition to be successful, a conductive circuit path has to be established for the cathodic current. In addition, sufficient adhesion must exist between the substrate and the deposited metal. A thin layer of metal (circa 300A) applied to the substrate establishes a plating base that is both conductive and adhesive. The plating base or metalization layer can be applied by the thermal evaporation of metals under the bombardment of energetic ions (Colchaser, 1980).

(ii) establishment of a mask pattern

A radiation sensitive polymeric resist material is applied in a uniform layer on top of the plating base. Resist formulations typically contain azide or diazo compounds which initiate solubility changes in the material following exposure to UV light. A high resolution pattern is defined in the resist layer by the use of UV light, X-rays, or an electron beam. This exposure step is followed by a chemical development step to yield the channels and holes that metal will be
Fig. 4.1 Processing Steps in Electrodeposition Through Masks

Preparation

Polymeric Resist

Metallic Adhesion & Plating Base

Substrate

Exposure

UV, X-Ray, or Electron Beam

Template Mask

Resist

Substrate

Development

Resist

Substrate

Deposition

✓ Plated Metal

Resist

Substrate

Stripping

After Resist and Plating Base Removal
deposited in.

(iii) electrodeposition

The metalized patterned substrate is placed in a conventional electroplating bath and a potential difference is applied. Electrodeposition occurs from the ionic solution, forming the desired metal structure. This step is the subject of the model study.

(iv) mask removal

The mask is stripped with a chemical bath leaving the newly-formed deposit and exposing areas of the thin metalization layer that were covered.

(v) extraneous metalization removal

A variety of etching techniques can be used to remove the thin metalization layer where it has not been covered by the electrodeposit.

When metal is electrodeposited through a mask under the proper conditions, the deposit conforms faithfully to the dimensions of the mask pattern. Hence this process can be used to fabricate metal structures that preserve the pattern resolution obtainable with current lithography technology (Spiller et al., 1976).

Romankiw et al. (1979a) have compared this technique to other procedures used to obtain patterned microscopic metal structures. The techniques of wet chemical etching, reactive ion etching, sputter etching, and ion milling cannot etch
wholly in one direction. To varying extents they degrade the resolution of the mask pattern by lateral spread of the etching species. Lift-off techniques involving the evaporative deposition of a metal are more faithful to the mask pattern, but it is only with difficulty that features much taller than they are wide can be obtained. Electroplating through masks does not have this aspect ratio limitation. In view of its advantages, electroplating through masks is beginning to see use in the fabrication of magnetic bubble memories (Spiller et al., 1976), thin-film recording heads (Romankiw et al., 1979b), masks for X-ray lithography (Buckley et al., 1981), fresnel zone plates and submicron gratings for X-ray diffraction (Shaver et al., 1979), and interconnection networks for microelectronic circuitry (Tuxford and Romankiw, 1974).

This work is a theoretical investigation into the mass transfer aspects of the electrodeposition step of this metallization process. The masked electrode is distinguished by feature sizes that are small relative to the concentration boundary layer thickness, and by the separated, recessed cathodic areas. We derive a model which emphasizes the importance of diffusional limitations in influencing the evolving shape of the electrodeposit. This work is an example of a tertiary current distribution analysis since the effect of concentration variation on the cathode polarization is considered. Polarization refers to the dependence of the potential change at an electrode surface on the interfacial
current. When concentration variation is neglected, a primary or secondary current distribution problem results, depending on whether polarization is neglected or not. Other theoretical studies of electrode shape evolution either have not considered tertiary current distribution effects (Alkire, Bergh and Sani, 1978; Riggs, 1977; Sautebin et al., 1980; Prentice and Tobias, 1982b), have considered tertiary effects but do not apply to microprofiles (Prentice and Tobias, 1982c), or have only considered the limiting current case (Sautebin and Landolt, 1982). Among the studies just cited, only Alkire, Berg, and Sani (1978) have considered a discontinuous electrode geometry.

FORMULATION OF THE MODEL PROBLEM

A microprofile is defined as a feature of electrode topography whose size rivals or is smaller than the thickness of the concentration boundary layer adjacent to the electrode surface (Kardos and Foulke, 1962). Microprofiles of an electrode present an uneven surface to mass transfer by diffusion and convection because raised parts of the profile are more accessible to mass transport of the reacting species from the bulk electrolyte. In the process of electrodeposition through masks, the cathode layout is determined photolithographically, so that typical feature sizes are 1–100 microns. The thickness of the concentration boundary layer can be approximated from measurements or
calculations of the mass transfer coefficient. Under conditions of gentle stirring, an effective film thickness of 50-100 microns is typical. Hence deposited features will be within the size domain of the microprofile.

Kardos and Foulke (1962) have presented an extensive review of electrodeposition on microprofiles. They present theoretical estimations and experimental results to support the relatively large importance of diffusional and convective effects in determining the distribution of current density on a microprofile. For a microprofile, ohmic influences are negligible as compared to concentration and activation overpotentials in determining the local pattern of current distribution. Neglecting the role of plating bath additives, variation in the current density on a microprofile is primarily due to the variation in mass transfer by diffusion or convection of the depositing species.

We consider the case of a single depositing species in a supported electrolyte bath. The reduced importance of the potential field in the vicinity of a microprofile electrode feature leads to a simplification of the current distribution analysis. The portion of the electrode overpotential that is not attributed to changes in the concentration field of the depositing species is assumed constant. The contribution of migration to transport of the minority depositing species is negligible and solution of the potential field is unnecessary.

Convective effects are included in the analysis in an approximate manner by modeling the concentration boundary
layer as an equivalent stagnant film adjacent to the electrode surface. The thickness of this film is inversely proportional to the mass transfer coefficient. A thinner film corresponds to greater agitation, greater convective mass transport in the bulk, and a faster rate of deposition. Outside of the stagnant film, all species are assumed to be at their bulk concentrations.

The cell geometry considered is illustrated in Figure 4.2. At the cathode, where the deposition is occurring, a prototypic pattern of evenly spaced, identical rectangular trenches is considered. This might represent a mask for a grating or a group of circuit lines. Since the potential field only influences the deposit growth through its constant contribution to the cathodic overpotential, details of the anode geometry are not considered. Conceptually, the anode is parallel to the masked electrode plane and located well above the top of the stagnant diffusion film.

It is assumed that a single cathodic deposition reaction occurs and that the current density normal to the cathode surface is described by a concentration-dependent Tafel equation (Newman, 1973)

\[ i_{\text{catn}} = -i_0 (C_s/C_{\infty})^{\frac{1}{2}} \exp\left(-\alpha_c \eta F/RT\right) \quad (4.1) \]

The surface overpotential, \( \eta \), is found as the sum of the concentration overpotential, \( \eta_c \), and a constant

\[ \eta = \eta_c + \text{constant} \quad (4.2) \]

\[ \eta_c = -(RT/zF) \ln(C_{\infty}/C_s) \quad (4.3) \]
Fig. 4.2 Two-dimensional model geometry.
As discussed above, the constant corresponds to the level of applied potential, which influences the current level but is much less important than the concentration field in affecting current density uniformity. The expression (4.3) for the concentration overpotential is valid for a minority reacting species in a supported electrolyte, as shown by Newman (1973).

We assume that the bulk concentration and the diffusivity of the reacting species are constant during the deposition and that the rate of shape change of the deposit is slow compared to the establishment of the concentration field. The last assumption is based on an estimated Peclet number for the advance of the cathode of $10^{-5}$, using $10^{-6}$ cm/sec as a typical velocity for the cathode surface (Tuxford and Romankiwiw, 1974).

With these assumptions the deposit growth is modeled by solving Laplace's equation for the concentration field

$$\nabla^2 C = 0$$  \hspace{1cm} (4.4)

subject to the following boundary conditions. First, the cathode current density is proportional to the diffusive flux of the reacting species normal to the cathode

$$i_{\text{cath}} = n \cdot zFD\nabla C \text{, cathode surface}$$  \hspace{1cm} (4.5)

Here $n$ is the surface normal vector which points into the deposit surface from the electrolyte. Usually the current density is described by equation (4.1). We are also
interested in solving for the limiting current distribution for comparison. For this special case, the concentration of the limiting reactant at the cathode is set to zero

\[ C = 0, \text{ cathode surface (limiting current)} \quad (4.6) \]

The cathode current density also determines the rate of cathode movement. The position of the cathode surface is described parametrically as \( S(X, Y, t) \). Assuming a current efficiency of unity, its velocity in the direction of the surface normal is given as

\[
\frac{\partial S}{\partial t} = \frac{M}{zF} i_{\text{cat}} \quad (4.7)
\]

The cathode current density is negative, indicating that growth is into the electrolyte. At the bulk solution–diffusion layer interface, the reacting species is at its bulk concentration

\[ C = C_\infty, \quad Y = H \quad (4.8) \]

Finally, there is no flux across planes of symmetry or insulating surfaces

\[ \mathbf{n} \cdot \nabla C = 0, \text{ insulator, symmetry planes} \quad (4.9) \]

The equation set is recast in nondimensional variables by using the trench half-width as the characteristic length and defining
\[ c \equiv \frac{C}{C^0}, \quad s \equiv \frac{S}{L} \]

\[ x \equiv \frac{X}{L}, \quad y \equiv \frac{Y}{L} \quad (4.10) \]

\[ h \equiv \frac{H}{L}, \quad \tau \equiv \frac{DC_{w, t}}{\rho L^2} \]

\[ \nabla^* \equiv L \nabla \]

Upon substitution, equations (4.1-9) become

\[ \frac{\partial^2 c}{\partial x^2} + \frac{\partial^2 c}{\partial y^2} = 0 \quad (4.11) \]

\[ c = 1, \quad y = h \quad (4.12) \]

\[ \nabla^* c = -\xi c \quad , \text{cathode surface} \quad (4.13) \]

\[ c = 0 \quad , \text{cathode surface (limiting current)} \quad (4.14) \]

\[ \nabla^* c = 0, \quad \text{insulator, symmetry planes} \quad (4.15) \]

\[ s(x, y, \tau = 0) = \{ (x, y) : 0 \leq x \leq 1, \quad y = 0 \} \quad (4.16) \]

\[ \frac{\partial s}{\partial \tau} = \nabla^* c, \quad \tau > 0 \quad (4.17) \]

where the polarization parameter, \( \zeta \), is defined as

\[ \zeta \equiv \frac{L_i}{z \frac{F \eta}{C_{w, t}}} \exp[-\alpha \frac{F (\eta - \eta_c)}{RT}] \quad (4.18) \]

At any point in time, equations (4.11-15) are used to determine the concentration field along the cathode, which is then used to integrate equation (4.17). The result of the integration is a history of the deposit shape and its local
growth rate.

This model was originally implemented using a biquadratic FEM formulation similar to that described in Chapter II. The method converged reliably on the nonlinear, algebraic equation set at each time step for all parameter values, and conveniently accommodated the deforming domain. However, close examination of the solutions calculated with a moderate number of elements showed that nearly 25% of the current flow was through the insulating surfaces. The current was found by differentiation of the concentration field and as is discussed in Chapter II, this leads to a slower convergence rate than is found for the concentration field. This experience led to development of a quadratic BEM formulation which;

(i) satisfies the no current condition on the insulators directly,

(ii) has a faster rate of convergence for the normal gradient calculation compared to the FEM,

(iii) has the isoparametric mapping advantage of the FEM, and,

(iv) gives a simpler and compact spatial representation since only the domain boundary is discretized.

A GBEM formulation (Chapter II) could be applied to this problem, and would be computationally more efficient and more compact than the QBEM formulation shown here. It was not used in this study since it had not been developed at the time
this work was done.

The quadratic BEM matrix formulation for this problem follows. The starting point is the integral form of Laplace's equation (4.19) derived in Chapter II.

\[ \Theta_i C_i = \frac{1}{2\pi} \int \frac{C[n \cdot \nabla \ln(r)]}{r} dr + \frac{1}{2\pi} \int \frac{ln(r) (n \cdot \nabla C)}{r} dr = 0 \quad (4.19) \]

Substituting the basis functions and boundary conditions of equations (4.12-15) into equation (4.19) results in an algebraic equation for each node point \((x_i, y_i)\):

\[ \Theta_i C_i = \frac{1}{2\pi} \int \frac{n \cdot \nabla \ln(r)}{r} dr - \frac{1}{2\pi} \int \frac{(\Sigma \Xi_i C_i) (n \cdot \nabla \ln(r))}{r_i, r_c} dr 
+ \frac{1}{2\pi L} \int \left[ \ln(r) \right] \left[ \Sigma \Xi_i (dc/dn) \right] dr 
- \frac{\xi}{2\pi L} \int \left[ \ln(r) \right] (\Sigma \Xi_i C_i) \frac{\gamma - (\kappa_c/n)}{r_c} dr = 0 \quad (4.20) \]

The integrals in equation (4.20) are nonzero only along specified boundary segments as indicated by the notation. The segment types are \(r_c\) (cathode), \(r_i\) (insulator or symmetry plane), and \(r_s\) (top of diffusion layer).

The matrix formulation for this problem

\[ A X + F + Q = 0 \quad (4.21) \]

is obtained by defining
\[
A_{i,j} \equiv \begin{cases} 
\int_{\Gamma_i} \frac{\partial}{\partial r} \ln (r) \, d\Gamma - 2\pi \theta_i \delta_{ij}, & \text{for } X_i = c_i \\
- \frac{1}{L} \int_{\Gamma_s} \ln (r) \, \frac{\partial}{\partial \Gamma}, & \text{for } X_i = (dc/dn)_i 
\end{cases} 
\tag{4.22}
\]

\[
X_i \equiv \begin{cases} 
c_i & \text{for } \Gamma_i, \Gamma_c \\
(dc/dn)_i & \text{for } \Gamma_s 
\end{cases} 
\tag{4.23}
\]

\[
F_i \equiv \int_{\Gamma_s} n \cdot \nabla \ln (r) \, d\Gamma 
\tag{4.24}
\]

\[
Q_i \equiv - \frac{\xi}{L} \int_{\Gamma_c} \left[ \ln (r) \right] \left( \Sigma \mathbb{E}_{c_i} \right) \, d\Gamma 
\tag{4.25}
\]

In equation (4.25), the vector \( \mathbf{Q} \) has a nonlinear dependence on the unknown cathode surface concentrations. Solution to the matrix equations is accelerated using Newton-Raphson iteration, where the components of the Jacobian are found from the \( A \) matrix and from explicit differentiation of \( Q_i \) with respect to \( c_i \).

The integrals needed in the above terms were done numerically using a 10 point Gauss-Legendre quadrature rule, except for the integrals that contain \( \ln (r) \) singularities which occur when the base point coincides with the portion of
the boundary being integrated. In this case as shown in Appendix D, a 4 point quadrature formula that accounts for the singularity was applied. The study of Chapter II showed that these quadrature rules were conservative and that 4 Gauss-Legendre points with 2 logarithmic points would have sufficed. As discussed in Chapter III better numerical accuracy is obtained if $\theta_i$ is calculated by difference, as described by Brebbia (1978), rather than directly from the interior angle of the domain at $(x_i, y_i)$. The derivation summarized here for the Tafel polarization expressed in equation (4.1) also applies to the limiting current case, autatis autandis.

CALCULATION ALGORITHMS

Euler-predictor, trapezoid-corrector time integration was used to integrate equation (4.17). This algorithm is presented in Chapter II. Irrespective of the integration scheme used, the error in the calculated position of the deposit surface must increase with time. This is because the problem is physically and mathematically unstable, as will be discussed in the next section. As in the method comparison problem the cost of solving for the field solution at any instant can be reduced by keeping a portion of the node structure fixed and not recalculating matrix contributions from this portion. This economy is reflected in the following procedure that is used to solve the nonlinear algebraic
equation set for the concentration field as required in the
time-stepping.

1. Initialize the matrix terms for the current time step.
The matrix equation contributions from the fixed
nodes are read from storage.

2. Re-position the nodes in the trench region to
accommodate the current position of the cathode
nodes.

3. Calculate matrix contributions from the moving nodes,
add them with the fixed terms, and store the result.
The stored result at this step is a base for
iteration with the present node structure.

4. Begin (or continue) Newton-Raphson iteration. Read the
matrix terms stored in step 3.

5. Calculate nonlinear solution-dependent terms arising
from the boundary condition at the cathode and add
them to the matrix terms, thus completing
formulation of the residuals and the Jacobian.

6. Improve the solution approximation by solving the
linear equation set based on the Jacobian. Here the
Jacobian is full and asymmetric so that Gaussian
elimination with partial pivoting is used. Steps 4,
5, and 6 are repeated until a converged solution is
found. The convergence criterion was that no
particular unknown vary by more than $10^{-5}$ from one
solution to the next. Quadratic convergence rates
were observed for all parameters used and typically only three or four iterations were needed to meet the above convergence criterion.

7. Using equation (4.17), the concentrations along the cathode surface are used to calculate the rates of growth along the outward normal. The direction of a surface normal vector is derived from the basis function interpolation of the surface position.

RESULTS AND DISCUSSION

A definitive analysis of error in the numerical solutions is not possible since analytical solutions are not available. However, an estimate of the solution error was obtained by analyzing the convergence of the solution as the number of nodal points was increased. Figure 4.3 shows the root-mean-square (RMS) differences in the concentration values at the cathode nodes for solutions obtained with varying total nodes, relative to the solution obtained with 240 total nodes. The observed solution convergence rate was slightly less than quadratic for the calculation with the nonlinear Tafel kinetics and somewhat faster than quadratic for the linear limiting current calculation. There was close agreement between the solutions obtained with a small number of nodes and the solutions obtained using 240 nodes.

A strong indication that the solutions obtained were converging to the correct result was the rapid convergence of
Fig. 4.3 Convergence of the overall mass balance and of the concentration solution at the cathode surface. Imbalance of the mass flow over the boundaries is expressed as a fraction of the average cathode current. The concentration field error is the root-mean-square error at the cathode relative to the solution obtained using 240 total nodes. Slopes for the linear regression lines are as indicated.
the overall mass balance, also shown in Figure 4.3. The mass balance was obtained by integrating the net current over the surfaces of the domain. This imbalance was expressed as a fraction of the total current at the cathode boundary. The slopes of the mass balance curves in Figure 4.3 indicate that the mass balance error was proportional to the node spacing raised to the third power, and was comparable to solution error convergence rates for the Galerkin finite element method applied to Laplace's equation using biquadratic elements (Strang and Fix, 1973).

For each time step the concentration field had to be solved for twice, and for the deposition history of a profile, a sequence of time steps was needed. The results were insensitive to the size of the time step that was used. For most problems a simulation composed of 5 time steps agreed to within a few percent with simulations that used many smaller time steps. When the current distribution was nearly uniform, even fewer time steps were needed. Simulations that involved filling a deeper trench area, or that involved a less uniform current distribution were more sensitive to the time step and were handled accordingly.

The computation expense of these calculations may be estimated from the timing data presented in Chapter II (Figure 2.6). The limiting current calculation compares directly to the model problem solved in Chapter II; both resulted in linear equation sets to solve for the field variable. The Tafel kinetics model required approximately 3.2 times more
computation work per time step, which reflects the need for three or four iterations to find a converged solution. Since the field solution was obtained twice at every time step, the moving boundary problem cost is twice that shown in Figure 2.6, times the number of time steps, times 3.2 for the Tafel case.

Reaction kinetics, trench geometry, and deposition time are each important in determining the deposit shape history. The kinetic factors include, $\gamma$, $\alpha_c$, $z$, and $\xi$. The values of $\gamma$, $\alpha_c$ and $z$ were fixed throughout this study at 0.58, 0.36 and 2 respectively. These values are appropriate for a copper acid-sulfate bath and are based on the statistical study of Caban and Chapman (1977). The most important kinetic factor is the polarization parameter (equation 4.18), $\xi$. It is a measure of mass transfer limitations compared to electrode kinetic limitations. Increasing the Tafel exchange current density corresponds to a more active electrode surface and a higher value of $\xi$. On the other hand, larger diffusivity and bulk concentration of the depositing species decrease mass transfer limitations and thus decrease $\xi$. $\xi$ also incorporates the effect of applied cell potential on the electrode reaction; for a higher applied cell potential, $\xi$ will grow exponentially. As $\xi$ becomes infinite, the interfacial concentration becomes zero, and limiting current behavior results.

In Figure 4.4 the effect of $\xi$ on the average current density across the initial, flat cathode surface is portrayed.
Fig. 4.4 Average initial current density as a function of the polarization parameter, for three diffusion film thicknesses.
The three curves represent different levels of convection, or thicknesses of the diffusion film, $h$. Two regimes of electrochemical cell behavior are identified. When $\xi < 0.01$, the kinetic resistance of the electrode reaction controls the rate of reaction and the current is independent of $h$. The solution of the concentration field in this low current regime was nearly constant at the bulk value and the current density was nearly uniform over the cathode surface. At high levels of $\xi$, mass transfer controlled the electrodeposition. The total current was insensitive to the precise level of $\xi$; the cell geometry and the thickness of the diffusion film determined the deposition pattern. Only at intermediate values of $\xi$ were both mass transfer and kinetics important in setting the deposition history.

In Figure 4.4, the polarization scale is broken so that the current values from the limiting current calculations are shown on the extreme right. There was a smooth approach of the nonlinear (finite $\xi$) solutions to the separately calculated limiting current (infinite $\xi$) solutions. This confirmed the consistency of the underlying calculations.

The value of $\xi$ had a strong influence also on the thickness variations of the deposit shape. In Figure 4.5, deposit profiles are drawn to show the influence of $\xi$ after sufficient current has passed to fill either 35% or 70% of the trench area, denoted by $\Lambda=0.35$ and $\Lambda=0.70$. Figure 4.5 shows only the right half of a symmetrical trench deposit, being consistent with the model cell geometry of Figure 4.2. For
Fig. 4.5 Deposit profiles for two values of the polarization parameter, when the trench area is 35% or 70% full.
the cases depicted in Figure 4.5 the initial trench depth was 20% of the trench half-width. The pattern of nonuniformity depicted in Figure 4.5 is in accord with expectation. The right-hand side of the cathode received additional current from the area above the insulating resist material. The deposit on the left-hand side of the diagram corresponds to the center of the trench and only received current from the cell area above it. Therefore as ε was increased and mass transfer effects became more important, the deposit grew fastest next to the trench wall. The profile was flat as it approached the left side of the diagram. This was predictable from the symmetry condition that was imposed for the center of the trench.

Except near the limiting current the deposit profiles calculated for other conditions were similar to those depicted in Figure 4.5. The maximum deposit was at the trench wall and the minimum was at the trench centerline. A smooth curve connected the two extrema, and the curve became flat as it neared the centerline minimum. As the limiting current was approached and the current distribution became less uniform, it was possible to develop rippled deposit shapes. In this case the predicted node locations along the cathode surface were alternatively above and below some average curve.

This rippling is due to an underlying physical instability of the deposition process. A point on the cathode surface that rises above the neighboring surface receives more current because it is more accessible. The enhanced
deposition caused by the increased current accelerates the growth of the nonuniformity. The perturbation analysis of Fedkiw (1980), for a primary current distribution, indicates that shorter wavelength, sinusoidal electrode profiles will grow faster than longer wavelength ones. Our numerical simulations are consistent with this result in two ways. First, when rippling occurred, it had the periodicity of the node spacing, which is the lowest ripple wavelength that can be accommodated in the boundary element approximation. Second, refining the approximation by introducing more nodes caused rippled deposits to appear at lower currents, indicating that the shorter wavelength ripples had a faster rate of growth.

In addition to the node spacing, the formation of ripples was influenced to an unknown extent by the accumulation of calculation error. Fortunately, rippled surfaces were found only near the limiting current, and after a substantial degree of deposit growth. Prentice and Tobias (1982b) report similar instability phenomena which they suppressed, arbitrarily, by numerically smoothing their node locations. No smoothing procedure was employed in the present study.

An effective means of parameterizing the usual deposit shapes as they evolve in time is to express the difference in deposit thickness as a fraction of the maximum thickness. That is, the index of nonuniformity is defined as

\[ \Delta Y \equiv \frac{Y_{\text{max}} - Y_{\text{min}}}{Y_{\text{max}}}. \]  

(4.26)
The extent of the deposit is described by $\Delta Y$ (deposit area/trench area). Plots of $\Delta Y$ vs. $\Lambda$ are used to illustrate the effects of the various geometric parameters on the nonuniformity of the deposit. With reference to Figure 4.2, recall that the following geometric factors are defined as dimensionless ratios to the trench half width, $L$: $h$, the upper limit of the diffusion film; $b$, the depth of the trenches before deposition; and $w$, the overall symmetry width. To elicit the effect of these parameters on the deposit formation, it is convenient to define a standard set of parameter values as a reference case. For this purpose we choose $h=5$, $b=0.2$, and $w=2$.

Figure 4.6 displays the effect of varying the level of convection, as expressed by $h$, at three values of the polarization parameter, $\xi$. The other geometric parameters, $b$ and $w$, are held at their reference values. For a given value of $\xi$, increased convection (lower $h$) resulted in a more uniform deposit. With higher levels of convection, the diffusion film was thinner, mass transfer limitations were lessened, and current from above the insulating resist mask that caused uneven deposition made a smaller contribution to the total current. It is convenient to regard the area above the insulating resist mask as a reservoir of the depositing reactant. As mass transfer limitations become more pronounced, the additional mass flow from this reservoir promotes uneven deposition.

The curves in Figure 4.6 fall into groups showing the
Fig. 4.6 Effect of diffusion film thickness on the development of deposit nonuniformity.
Fig. 4.7 Effect of the trench spacing on the development of deposit nonuniformity.
Fig. 4.8 Effect of the trench depth on the development of deposit nonuniformity.
importance of $\xi$ in setting the nonuniformity of the deposit. At the low value, $\xi=0.2$, the influence of the trench geometry was only weakly felt and the profiles varied by only a few percent in thickness. One might suppose that the level of current at $\xi=0.2$ is only a small fraction of the limiting current, in view of the predicted uniformity of profile shapes. It is surprising to note that for the $h=5$ intermediate case, the current was at 39% of the limiting current level for the unfilled trench. The corresponding figures for $h=1$ and 10 are approximately 15% and 57% of the limiting current, respectively. When $\xi$ was 2.0, the current level for the $h=5$ case was at 91% of the limiting current.

In Figure 4.7 the effect of varying the overall symmetry width ($w$) is shown, with $h$ and $b$ held constant. As already noted, the dimensionless cathode half-width is always defined as 1.0. The results were again consistent with the idea of the insulator area functioning as a reservoir of the depositing reactant. When the overall symmetry width approached the cathode width ($w=1.1$), there was little mass flow available from the vanishing insulator area. If the insulator area were eliminated, a uniform one-dimensional current distribution would result. With a larger insulating area, mass flow to the cathode edge was augmented.

The effect of varying trench depth ($b$) is shown in Figure 4.8. As the initial trench was made deeper, the current density became more uniform, since the mass flow from above the insulator was able to diffuse laterally to a greater
extent before reaching the cathode surface. With progressively greater amounts of deposition, the deeper trenches became shallower and developed current distributions that were similar to those of trenches that were initially shallow. As any given deposit approached the top of the trench area, it encountered the least uniform current distribution of its history. For a deeper trench, this late growth had a smaller effect on the overall uniformity index ($\Delta Y$), because $\Delta Y$ was normalized by the maximum thickness of the deposit.

A word of explanation concerning the end points of the curves in Figs. 4.6-8 is in order. The calculations assumed that the deposit height along the insulator wall was always below the top of the insulator. Since the profiles can become sharply curved, the deposit can reach the top of the insulator when there is still a large portion of the trench area unfilled. The different curve lengths in Figs. 4.6-8 were caused by varied times for termination of the calculations, because of this trench overrun phenomenon.

The predicted degree of thickness nonuniformity increased very rapidly as the limiting current was approached. This is the theme of Figure 4.9, where we show the ratio of current density nonuniformity to the current density nonuniformity found at the limiting current, both for the initial flat cathode. The definition of current density nonuniformity is entirely analogous to equation (4.26) for deposit nonuniformity;
Fig. 4.9 Nonuniformity of the current density on the initial flat cathode surface as a function of the average current density. The nonuniformity of current density is expressed as a fraction of the nonuniformity found at the limiting current, and the average current is expressed as a fraction of the limiting current.
\[ \Delta I \equiv (I_{\text{max}} - I_{\text{min}})/I_{\text{max}} \] (4.27)

When the diffusion film thickness was large, the current from the area above the insulating resist became a larger fraction of the decreased total current. As the average current approached the average limiting current, the additional current from this region had a significant impact on the overall current distribution. Thus, the relative rise in current nonuniformity was more pronounced for a thicker diffusion film, as shown in Figure 4.9.

SUMMARY AND CONCLUSION

We have made theoretical predictions of deposit shapes resulting from the use of photolithographically prepared mask structures. Our analysis emphasizes the effect of concentration variations in determining the pattern of current nonuniformity on the exposed cathode, represented as a set of parallel, microscopic trenches. We have treated the applied cell potential as a constant contribution to the cathodic overpotential. This analysis is appropriate when ohmic influences on the local variation of current density are negligible compared to concentration and activation overpotentials.

Our modeling results show a strong dependence of deposit shape on the polarization parameter, \( \xi \), which expresses the importance of mass transfer limitations relative
to electrode kinetic limitations. For large values of $\xi$, where mass transfer limitations predominate, additional mass flow from the area above the inactive masking material to the cathodic surface causes enhanced deposition rates at the electrode edges adjacent to the mask. This edge effect can be diminished by reducing the fractional area of the substrate that is masked, by increasing the agitation of the bath, or by increasing the thickness of the mask pattern. Reducing the applied cell potential corresponds to reducing $\xi$. This diminishes mass transfer limitations and smooths the current distribution but slows the deposition rate. The tradeoff is such that deposit profiles formed at 40% of the limiting current show only a small degree of height variation, for the geometry considered.

The BEM was particularly well suited for the model problem since accurate evaluation of the normal concentration gradients was important. The present analysis can be readily extended to other mask geometries, including non-periodic and three-dimensional patterns. For the latter case, the relative compactness of the BEM is particularly advantageous. The BEM is also applicable to the unsteady diffusion equation which would allow this analysis to be extended to pulsed cell potentials. However, as discussed in Chapter V, the BEM is probably not the best method for the pulsed problem.
LIST OF SYMBOLS FOR CHAPTER IV

*English Characters*

\( A \) formulation matrix

\( b \) dimensionless depth of trench before deposition

\( B \) dimensional depth of trench before deposition (m)

\( c \) dimensionless concentration of the depositing species

\( c_i \) dimensionless concentration at the nodal point \( i \)

\( C \) concentration of the depositing species (g/mole/m³)

\( C_s \) concentration of the depositing species at the cathode surface (g/mole/m³)

\( C_\infty \) bulk concentration of the depositing species (g/mole/m³)

\( D \) diffusivity of the depositing species (m²/sec)

\( F \) Faraday's constant (96,500 C/g equiv.)

\( E \) vector resulting from matrix formulation

\( h \) dimensionless thickness of the diffusion film

\( H \) dimensional thickness of the diffusion film (m)

\( i_{\text{cath}} \) current density normal to the cathode surface (A/m²)

\( i_\infty \) exchange current density corresponding to the bulk concentration of the depositing species (A/m²)

\( \Delta I \) current nonuniformity index, Eq. (4.27)

\( L \) half-width of the cathode trench (m)

\( M \) molecular weight of the depositing species (g/gmole)
n  normal vector pointing from the electrolyte into the deposit
Q  vector of nonlinear contributions
r  radial distance from the point i
R  gas constant (8.32 J/gmole-°K)
s  dimensionless position of the cathode surface
S  position of the cathode surface, S=S(X,Y,t) (m)
t  time (sec)
T  absolute temperature (°K)
w  dimensionless symmetry width
w_i weighting function based at the ith node point
W  dimensional symmetry width (m)
x  horizontal coordinate, dimensionless
x_i  horizontal coordinate of node i
X  dimensional horizontal coordinate (m)
X  vector of unknown concentrations and fluxes
y  vertical coordinate, dimensionless
y_i  vertical coordinate of node i
Y  dimensional vertical coordinate (m)
\( \Delta Y \)  deposit nonuniformity index, Eq. (4.26)
z  valency of the depositing species and the number of electrons in the deposition reaction (g equiv./gmole)
Greek characters

\( \alpha_c \) cathodic transfer coefficient

\( \gamma \) concentration dependence exponent of the exchange current density

\( \Gamma \) the boundary surface of the model domain

\( \Gamma_c \) cathode surface contour

\( \Gamma_d \) diffusion film surface contour

\( \Gamma_i \) insulator or symmetry plane surface contour

\( \delta \) Dirac delta function

\( \delta_{ij} \) Kronecker delta function, \( \delta_{ij}=1 \) if \( i=j \), otherwise \( \delta_{ij}=0 \)

\( \eta \) surface overpotential

\( \eta_c \) concentration overpotential

\( \Theta_i \) constant related to position of the point \( i \)

\( \Lambda \) trench deposit area / total trench area

\( \xi \) polarization parameter

\( \rho \) density of the cathodic deposit (g/m\(^3\))

\( \tau \) dimensionless time

\( \bar{\varphi}_i \) quadratic polynomial basis function centered at nodal point \( i \)

\( \Omega \) two-dimensional region where Laplace's equation applies
CHAPTER V

POSTSCRIPT

The purpose of this chapter is to reflect upon the methods and problems examined in this thesis and project to other circumstances. We saw that the finite domain comparison problem examined in Chapter II was well suited for both the FEM and the BEM, and accurate results were obtained using either method. It is important to identify characteristics of problems that would not lead to so balanced a comparison. For example, the particular strength that the BEM has for infinite domains was made evident by the comparison study of Chapter III. Two other pertinent differences of the methods are highlighted by considering further changes to the comparison problem.

First, suppose that the field solution was needed at points in the interior of the domain. With the FEM, each point would require identification of the element that the point was on, and interpolation of the element node values using the basis functions. If the point happened to be a node point, the solution would be known without further calculation. For the BEM, an integral equation must be formulated and solved for each interior point. This is substantially more calculation than with the FEM.

Second, a change in favor of the BEM is to have the active surface move in proportion to the normal gradient, as
was the case with the electroplating study. As we saw in Chapter II, the FEM would require more nodes to obtain as accurate a solution since its convergence rate for the gradient is less. This principle would also apply to Stefan problems where motion of the melt-solid interface is in proportion to the heat flux (Elliott and Ockendon, 1982).

Since the BEM does not require discretization of the domain interior, it is particularly appropriate for applications that involve surfaces where drawing an element mesh would be unusually complicated. An example is modeling the deformation of steep surface waves by Longuet-Higgins and Cokelet (1976). Using a boundary formulation they were able to track the evolution of breaking waves beyond the point where the wave surface became vertical. The authors used different functions to interpolate the system boundaries and the independent variable. The methods demonstrated here use the same basis functions for both roles – they are isoparametric. This leads to a consistent level of approximation, and simplifies the formulation of the method.

The BEM can be applied to problems where at least free space Green’s functions are known. A list of free space Green’s functions for equations commonly encountered in engineering is given by Brebbia and Walker (1980). A more comprehensive listing including functions for specific boundary conditions is found in Butkovskiy (1982), and Greenberg’s text (1971) is instructive on the subject of deriving functions. In general, the BEM cannot be applied to
problems where the governing equation is nonlinear.

Application of the finite element method is more general. Fundamental solutions of the governing equation are not required and so it may be applied to more complicated or nonlinear field equations. A pertinent example application would be to a current distribution problem where transport by migration and diffusion are both important.

With Laplace's equation, the major advantage of the BEM is that only the domain boundary is discretized. Other field equations may require integration over the domain itself, in which case the FEM might be more suitable. An example is solving Poisson's equation. If the BEM is applied directly, an integral over the domain involving the source term and a Green's function results (Brebbia and Walker, 1980). However, for an uncomplicated source term, Poisson's equation can be reduced to Laplace's equation by the superposition of a particular solution (eg. Jawson and Symm, 1978).

Application of the BEM to the transient diffusion equation also requires integration of the initial conditions over the domain. On a static domain the integral formulation may be used to obtain results for a particular time without the usual finite difference step by step integration through prior time (Brebbia and Walker, 1980). For this purpose, the integration over the domain need only be done once, or not at all if the initial conditions are uniform (Shaw, 1973). However, for the case of a moving boundary problem, stepping
through time is required if the domain shape is not known as an explicit function of time. Even when not required, it is often the case that the time history is the raison d'être of the analysis, and time stepping is desired. Then the solution at the end of each time step becomes the initial condition for the next step, and integration over the domain is performed at each step (Wrobel and Brebbia, 1981). Based on our experience, this would appear to be much more expensive than application of the FEM. However, the BEM retains its particular strengths for the infinite domain and the direct calculation of surface gradients. As we have seen, there is hope of reducing the computation by applying a Green's function that meets selected boundary conditions.

A virtue of the FEM is that it is capable of providing diagnostic information as to the accuracy of the solution, with little additional calculation. For example, with Laplace's equation the mass balance can be examined element by element. The analyst (or computer code) can use this information as a guide to refine the element mesh in order to improve the accuracy of the solution. There is no comparable measure of local error with the BEM.

Another practical concern relates to the manner in which complicated models are developed. The computer code is easiest to develop in stages, e.g., starting with a linear field equation and linear boundary conditions, and progressing to incorporate the desired nonlinearities. The decision to use the BEM at the outset implies that the model will not
incorporate a nonlinear field equation. In a sense the BEM Tafel kinetics model of Chapter IV is at an evolutionary cul-de-sac; though it incorporates nonlinear boundary conditions and a moving boundary, it cannot be applied to larger size scales where explicit treatment of convection and migration would be needed. On the other hand a FEM model could be extended in this manner, and though substantial revision would be required, this would be preferred to starting from scratch with unfamiliar or untested algorithms.
REFERENCES


Soc., 128, 1116.


London.


APPENDIX A

DERIVATION OF THE SPECIALIZED GREEN'S FUNCTION

In this appendix the derivation of a Green's function satisfying the sidewall boundary condition of the comparison problems is given. The desired Green's function, $w_i = w_i(x_i, y_i; x, y)$, satisfies Laplace's equation for a unit "sink" at the point $(x_i, y_i)$; i.e., $\nabla^2 w_i + \delta(x-x_i)\delta(y-y_i) = 0$. It will also possess zero gradients across the boundaries of an infinite strip. Schematically we have:

![Figure A.1, The Infinite Strip](image)

To derive the Green's function in closed form, the strip is conformally mapped to a half-plane where a simple known result can be applied. The mapping is from the $(x, y)$ coordinate system on to the $(s, t)$ system and can be succinctly expressed using complex variables. If $z \equiv x + iy$ and $u \equiv s + it$ then
the mapping is \( u = \exp(i\pi z) \). In terms of the component functions, we have \( s = \cos(\pi x) \exp(-\pi y) \) and \( t = \sin(\pi x) \exp(-\pi y) \).

In the half-plane, the strip boundaries are collinear. A zero flux condition on this line can be satisfied for a point charge if an image charge is associated with it as a reflection across the half-plane.

The necessary Green's function in this coordinate system is
based on combining the free space Green's functions of the 
source and image points, resulting in (Greenberg, 1971),

\[ w_1(s_i, t_i; s, t) = \frac{-1}{4\pi} \ln \left[ \frac{(s-s_i)^2 + (t-t_i)^2}{(s-s_i)^2 + (t+t_i)^2} \right]. \quad (A.1) \]

Note that the Green's function is a function of two points: 
the base point \((s_i, t_i)\) and the observation point \((s, t)\). This 
result is expressed in the original \((x, y)\) coordinates by 
isubstitution of the mapping expressions. The procedure is 
straightforward though tedious. After some rearrangement the 
result is,

\[ w_1(x_1, y_1; x, y) = \frac{-1}{4\pi} \ln \left( \frac{\cosh[\pi(y-y_1)] - \cos[\pi(x-x_1)]}{\cosh[\pi(y-y_1)] - \cos[\pi(x+x_1)]} \right). \quad (A.2) \]

Derivatives of this function are needed for application of the 
BEM. They are,

\[ \frac{\partial w_1}{\partial x} = \frac{-1}{4} \left\{ \frac{\sin[\pi(x+x_1)]}{\cosh[\pi(y-y_1)] - \cos[\pi(x+x_1)]} \right\} + \frac{\sin[\pi(x-x_1)]}{\cosh[\pi(y-y_1)] - \cos[\pi(x-x_1)]} \quad (A.3) \]

\[ \frac{\partial w_1}{\partial y} = \frac{-1}{4} \left\{ \frac{\sinh[\pi(y-y_1)]}{\cosh[\pi(y-y_1)] - \cos[\pi(x+x_1)]} \right\} + \frac{\sinh[\pi(y-y_1)]}{\cosh[\pi(y-y_1)] - \cos[\pi(x-x_1)]} \quad (A.4) \]
APPENDIX B

ELEMENTS, BASIS FUNCTIONS, AND MAPPING

In this appendix, element-based operations that are used to formulate the FEM and the BEM are briefly explained. References for this material include texts on the FEM (Zienkiewicz, 1977; Chung, 1978; Bathe and Wilson, 1976; Finlayson, 1980) and texts on the BEM (Brebbia, 1978; Brebbia and Walker, 1980).

One-dimensional elements

The same elements that are used for application of the FEM in one spatial dimension are used for application of the BEM in two dimensions, since only the one-dimensional boundary contour is discretized. To apply the method a boundary contour is subdivided into segments called elements. On each element, the dependent and independent variables are interpolated using low order polynomial functions of the node locations. The order and kind of interpolation sets the number of nodes assigned to each element. For linear interpolation, only node points at the ends of the elements are defined, and for quadratic interpolation, a node is placed in the interior of the element in addition to the nodes at each element end. In both these cases, the interpolation is continuous from element to element but the spatial derivatives are discontinuous at the element boundaries.

Basis functions are the polynomial functions that are
used to interpolate between the node points. Let us examine the quadratic basis functions that were used in the QBEM and GBEM formulations. Calculations on an element are done by mapping the element to a prototype element with a local coordinate, $\xi$, where $-1 \leq \xi \leq +1$.

![Diagram](attachment:image.png)

**Figure B.1**
Mapping of a Quadratic Element to the Prototype Element

The basis functions are defined with reference to the local coordinate $\xi$ and the local node numbering (1, 2, 3) of the prototype element as sketched in Figure B.2.

\[
\phi_1 \equiv \xi(\xi-1)/2 \\
\phi_2 \equiv 1 - \xi^2 \\
\phi_3 \equiv \xi(\xi+1)/2
\]  

(B.1)  

(B.2)  

(B.3)
Interpolation on the prototype element is done by expanding the interpolated quantity in terms of its nodal values and the basis functions associated with each node. Expansions for the element coordinates are

\[ x(\xi) = \sum_{j=1}^{3} \phi_j(\xi) x_{i,j-1} \]  
\[ (B.4) \]

and

\[ y(\xi) = \sum_{j=1}^{3} \phi_j(\xi) y_{i,j-1} \]  
\[ (B.5) \]

where we have assumed that the global node numbering results in node \( i \) being mapped to the left-most node (node 1) of the prototype element. Equations (B.4) and (B.5) define the mapping of the element containing nodes \( i, i+1, \) and \( i+2 \) to the prototype element. The mapping is termed isoparametric since the same basis functions (equations B.1-3) are used to interpolate dependent variables, eg.
\[ C(\xi) = \sum_{j=1}^{N} \phi_j(\xi) C_{i,j-1} \]  \hspace{1cm} (B.6)

Equation (B.6) is written for \( C \) as an explicit function of \( \xi \) on the element containing nodes \( i, i+1, \) and \( i+2 \), but in principle, the dependent variable \( C \) is a function of \( (x,y) \). When \( C \) is sought at a particular point \( (x,y) \), the appropriate element is selected, and the proper value of \( \xi \) is determined using (B.4) and (B.5). Usually, only nodal values of the dependent variables are desired, so the calculation of \( \xi \) from (B.4) and (B.5) is unnecessary.

Integration along the boundary contour is done with respect to the differential arc length, \( d\Gamma \).

\[ I \equiv \int_{\Gamma} F(x,y) \, d\Gamma \]  \hspace{1cm} (B.7)

The contribution to this integral from a single element is given as

\[ I_\xi \equiv \int_{-1}^{-1} F[x(\xi), y(\xi)] (d\Gamma/d\xi) d\xi \]  \hspace{1cm} (B.8)

where the surface mapping Jacobian is

\[ \frac{d\Gamma}{d\xi} = \sqrt{\left(\frac{dx}{d\xi}\right)^2 + \left(\frac{dy}{d\xi}\right)^2} \]  \hspace{1cm} (B.9)

The Jacobian requires that \( dx/d\xi \) and \( dy/d\xi \) be evaluated. They are easily found by explicit differentiation of (B.4) and
(B.5). The numerical evaluation of (B.7) is accomplished by summing contributions of the form (B.8) over different elements.

In summary, global interpolation is done element by element using the local basis functions (B.1-3). As we have done in the main text, global basis functions, \( \bar{\phi}_i(x,y) \), may be defined as the union of local representations (B.1-3) and treated as functions of \((x,y)\) by reference to (B.4) and (B.5). Figure B.3 is a depiction of a global basis function, \( \bar{\phi}_i(x,y) \), in the \((x,y)\) coordinate system. The contribution of the \( \phi_1 \) function from the left element and the \( \phi_3 \) function from the right element are discernable.

![Diagram](image)

**Figure B.3**

Global Interpolation Over Two Contiguous Quadratic Elements
Two-dimensional Elements

Two dimensional elements are used for application of the FEM in two spatial dimensions and application of the BEM in three dimensions. Here we consider the bi-quadratic elements that were used to apply the FEM. As in one dimension, calculations on an element are done by mapping to a prototype element, in this case the square, $-1 \leq \xi \leq 1$, $-1 \leq \eta \leq 1$.

![Diagram of mapping from bi-quadratic element to prototype square]

Figure B.4
Mapping of a Bi-quadratic Element to the Prototype Element

The basis functions, $b_i$, on the prototype element, are obtained as the products of the one-dimensional quadratic functions, (B.1-3).
These basis functions are used in expansions that are completely analogous to (B.4-6), eg.,

\[ x(\xi, \eta) = \sum_{j=1}^9 b_j x_{k(j)} \]  
(B.19)

The subscript \( k(j) \) on \( x_{k(j)} \) implies that the global node numbering is unspecified but varies with the local node index, \( j \), of the prototype element.

Integrals on the two-dimensional domain will be of the form

\[ I = \int_{\Omega} \nabla b \cdot \nabla s \, d\Omega \]  
(B.20)

The contribution to (B.20) from an individual element is

\[ I = \int_{-1}^{1} \int_{-1}^{1} \frac{\partial b_i}{\partial x} \frac{\partial b_j}{\partial x} + \frac{\partial b_i}{\partial y} \frac{\partial b_j}{\partial y} \mid J \mid d\xi \, d\eta \]  
(B.21)
where the Jacobian, $J$, is equal to

$$J = \left( \frac{\partial x}{\partial \xi} \right) \left( \frac{\partial y}{\partial \eta} \right) - \left( \frac{\partial x}{\partial \eta} \right) \left( \frac{\partial y}{\partial \xi} \right) \quad (B.22)$$

The derivatives required to calculate $J$ are found from explicit differentiation of the basis function expansion. For example,

$$\frac{\partial x}{\partial \xi} = \sum_{i} x_{k_i} \left( \frac{\partial b_i}{\partial \xi} \right) \quad (B.23)$$

The derivatives of the local basis functions with respect to the global Cartesian coordinates (e.g., $\frac{\partial b_i}{\partial x}$) require some algebraic manipulation since the basis functions are known as explicit functions of $(\xi, \eta)$ and not of $(x, y)$. First the chain rule is applied to the quantities that can be evaluated explicitly:

$$\frac{\partial b_i}{\partial \xi} = \frac{\partial b_i}{\partial x} \frac{\partial x}{\partial \xi} + \frac{\partial b_i}{\partial y} \frac{\partial y}{\partial \xi} \quad (B.24)$$

$$\frac{\partial b_i}{\partial \eta} = \frac{\partial b_i}{\partial x} \frac{\partial x}{\partial \eta} + \frac{\partial b_i}{\partial y} \frac{\partial y}{\partial \eta} \quad (B.25)$$

Equations (B.24-25) are expressed in matrix form,

$$\begin{bmatrix} \frac{\partial b_i}{\partial \xi} \\ \frac{\partial b_i}{\partial \eta} \end{bmatrix} = \begin{bmatrix} \frac{\partial x}{\partial \xi} & \frac{\partial y}{\partial \xi} \\ \frac{\partial x}{\partial \eta} & \frac{\partial y}{\partial \eta} \end{bmatrix} \begin{bmatrix} \frac{\partial b_i}{\partial x} \\ \frac{\partial b_i}{\partial y} \end{bmatrix} \quad (B.26)$$

and the matrix is inverted to give the desired result.
\[
\begin{bmatrix}
\frac{\partial b_i}{\partial x} \\
\frac{\partial b_i}{\partial y}
\end{bmatrix} = \frac{1}{J} \begin{bmatrix}
\frac{\partial y}{\partial n} & -\frac{\partial y}{\partial \xi} \\
-\frac{\partial x}{\partial n} & \frac{\partial x}{\partial \xi}
\end{bmatrix} \begin{bmatrix}
\frac{\partial b_i}{\partial \xi} \\
\frac{\partial b_i}{\partial n}
\end{bmatrix} \tag{B.27}
\]

Here \( J \) is the Jacobian given by (B.22).
APPENDIX C

ANALYTICAL EXPRESSIONS FOR THE LBEM FORMULATION

In this appendix analytical expressions are developed for the integrals needed to implement the linear basis function BEM. Usually BEM elements are mapped to a prototype element where \(-1 \leq \xi \leq 1\) since this range is convenient for Gauss-Legendre quadrature (\(\xi\) is the local coordinate as explained in Appendix B). For analytical evaluation it is more convenient to map the element to \([0,1]\). In the following development the point that is mapped to \(\xi=0\) is \((x_1,y_1)\) and the \(\xi=1\) point is \((x_2,y_2)\). The elemental forms of the linear basis functions are

\[
\phi_1 \equiv 1 - \xi \quad \text{(C.1)} \\
\phi_2 \equiv \xi \quad \text{(C.2)}
\]

Hence

\[
\frac{dx}{d\xi} = x_2 - x_1, \quad \frac{dy}{d\xi} = y_2 - y_1 \quad \text{(C.3,4)}
\]

and the surface mapping Jacobian is found as

\[
\frac{dr}{d\xi} = \sqrt{\left(\frac{dx}{d\xi}\right)^2 + \left(\frac{dy}{d\xi}\right)^2} \quad \text{(C.5)}
\]

Formulation of the LBEM requires that integrals of the form

\[
H_{ij} \equiv -\frac{1}{2\pi} \int_{\Gamma} \frac{\mathbf{n}_i \cdot \nabla \ln(\mathbf{r})}{\mathbf{r}} \, d\Gamma \quad \text{(C.6)}
\]

183
be evaluated. This integral is over the boundary where \( \vec{\xi} \) is nonzero, and is composed of integrals over individual boundary elements that are of the form

\[
    h_{ik} \equiv -\frac{1}{2\pi} \int_0^1 \phi_k \left[ \vec{n} \cdot \nabla \ln(r) \right] (dr/d\xi) d\xi
\]  

(C.7)

The term \( \vec{n} \cdot \nabla \ln(r) \) is constructed as follows. The surface normal is found as

\[
    \vec{n} = \frac{\vec{e}_y dy/d\xi - \vec{e}_x dx/d\xi}{\sqrt{(dx/d\xi)^2 + (dy/d\xi)^2}}
\]  

(C.8)

By definition

\[
    r = \sqrt{(x-x_i)^2 + (y-y_i)^2},
\]  

(C.9)

so that

\[
    \nabla \ln(r) = [\vec{e}_x(x-x_i) + \vec{e}_y(y-y_i)] / r^2
\]  

(C.10)

and

\[
    \vec{n} \cdot \nabla \ln(r) = \frac{(x-x_i)dy/d\xi - (y-y_i)dx/d\xi}{r^2 \sqrt{(dx/d\xi)^2 + (dy/d\xi)^2}}
\]  

(C.11)

Upon substitution equation (C.7) becomes

\[
    h_{ik} = \frac{-1}{2\pi} \int_0^1 \frac{\phi_k d}{a\xi^2 + b\xi + c} d\xi
\]  

(C.12)

where the constants are defined as

\[
    a \equiv (y_2-y_i)^2 + (x_1-x_2)^2
\]  

(C.13)

\[
    b \equiv 2 \left[ (y_2-y_i)(y_1-y_i) + (x_2-x_i)(x_1-x_i) \right]
\]  

(C.14)
\[ c \equiv (y_i-y_i)^2 + (x_i-x_i)^2 \quad \text{(C.15)} \]

\[ d \equiv (y_i-y_i)(x_i-x_i) - (x_i-x_i)(y_i-y_i) \quad \text{(C.16)} \]

Substituting the first basis function (equation C.1) into equation (C.12) and rearranging gives
\[
h_{ii} = \frac{d}{4\pi a} \int_0^1 \frac{(2a\xi + b) - (2a + b)}{a\xi^2 + b\xi + c} \, d\xi \quad \text{(C.17)}
\]

The integral (C.17) is split as shown by the grouping in the numerator and integrated. The result is
\[
h_{ii} = \frac{d}{4\pi a} \left[ \ln[(a+b+c)/c] - \frac{(2a+b)}{|d|} \left[ \tan^{-1}\left(\frac{2a+b}{2|d|}\right) - \tan^{-1}\left(\frac{b}{2|d|}\right) \right] \right] \quad \text{(C.18)}
\]

Similar manipulation with the second basis function (equation C.2) results in
\[
h_{zz} = \frac{-d}{4\pi a} \left[ \ln[(a+b+c)/c] - \frac{b}{|d|} \left[ \tan^{-1}\left(\frac{2a+b}{2|d|}\right) - \tan^{-1}\left(\frac{b}{2|d|}\right) \right] \right] \quad \text{(C.19)}
\]

As explained in Chapter III, \( H_{ii} \) contributions are found by difference, and do not require any integration.

Integrals of the form
\[ G_{ij} \equiv \frac{-1}{2\pi} \int_{\Gamma} r \ln(r) \, dr \]  

(C.20)

must also be evaluated. On an individual boundary element, equation (C.20) has the form

\[ g_{ix} \equiv \frac{-1}{2\pi} \int_{0}^{1} \phi_x \ln(r) \left( \frac{dr}{d\xi} \right) d\xi \]  

(C.21)

Using the constants (C.13-16) the integral (C.21) becomes

\[ g_{ix} = \frac{-\sqrt{\alpha}}{4\pi} \int_{0}^{1} \phi_x \ln(a\xi^2 + b\xi + c) \, d\xi \]  

(C.22)

Substituting the first basis function (equation C.1) into equation (C.22) yields

\[ g_{i1} = \frac{1}{8\pi\sqrt{\alpha}} \int_{0}^{1} [(2a\xi + b) - (2a + b)] \ln(a\xi^2 + b\xi + c) \, d\xi \]  

(C.23)

where the integrand has been rearranged to show how it may be split into two integrals that are readily evaluated. Upon integration equation (C.23) becomes
\[ g_{11} = \frac{1}{8\pi\sqrt{a}} \left( (a+b+c)\ln(a+b+c) - (c)\ln(c) - (a+b) \\
- (2a+b) \left( 1 + b/2a \right)\ln(a+b+c) - 2 - (b/2a)\ln(c) \\
+ (2|d|/a) \left[ \tan^{-1}\left( \frac{2a+b}{2|d|} \right) - \tan^{-1}\left( \frac{b}{2|d|} \right) \right] \right) \] (C.24)

For the second basis function (equation C.2), equation (C.22) evaluates to

\[ g_{12} = \frac{1}{8\pi\sqrt{a}} \left( - (a+b+c)\ln(a+b+c) + (c)\ln(c) + (a+b) \\
+ (b) \left( 1 + b/2a \right)\ln(a+b+c) - 2 - (b/2a)\ln(c) \\
+ (2|d|/a) \left[ \tan^{-1}\left( \frac{2a+b}{2|d|} \right) - \tan^{-1}\left( \frac{b}{2|d|} \right) \right] \right) \] (C.25)

Equations (C.18, 19, 24, and 25) are the basis of the linear boundary element formulation. The numerical implementation of these equations is complicated by the terms that result in division by zero or singular function
evaluation. The LBEM code listing (Appendix H) circumvents these problems by appropriate tests in subroutines GIJ and INTEL.
APPENDIX D
TREATMENT OF LOGARITHMIC SINGULARITIES

In the numerical formulation of the boundary element method each node serves as a base point or reference point to formulate an integral equation. As integration is performed on the element containing the base node, log(r) singularities of the integrand result. A special quadrature procedure is used as described below in order to obtain fast convergence of the methods without resorting to the expense of using a higher order Gauss-Legendre quadrature formula.

For the general boundary element method, the integral will be of the form:

\[ I \equiv \int_{-1}^{1} F(\xi) \log(\sqrt{[x(\xi)-x_i]^2+[y(\xi)-y_i]^2}) d\xi \quad (D.1) \]

This example is for the case of the base node being the left node on the element. When \( \xi = -1 \), \((x, y)\) become equal to \((x_i, y_i)\). The integral can be written as the sum of a nonsingular and a singular integral.

\[ I = \int_{-1}^{1} F(\xi) \log\left(\frac{4[x(\xi)-x_i]^2 + [y(\xi)-y_i]^2}{(\xi + 1)^2}\right) d\xi \]
\[ + 4 \int_{0}^{1} F(2t-1) \log(t) \ dt \quad (D.2) \]

The nonsingular integral is evaluated numerically using the
usual Gauss-Legendre quadrature rules. The integrand divisor has been chosen so that the singular integral is in standard form for special log quadrature points and weights.

\[
\int_0^1 G(z) \log(z) \, dz \approx \sum_{i=1}^{n} w_i G(z_i) \quad (D.3)
\]

The quadrature points and weights are due to Stroud and Secrest (1966), and can be found in the computer program listings. Integrating with two logarithmic quadrature points yielded results that overlaid those obtained with more logarithmic points.

The case of the base node being the right node on the element is similar to the example shown except the logarithmic argument is multiplied by \(4/(1-\xi^2)\). When the base node is the center node of the quadratic surface element, the integral is decomposed into three integrals.

\[
I = \int_{-1}^{1} F(\xi) \log\left(\frac{[x(\xi)-x_i]^2 + [y(\xi)-y_i]^2}{\xi^2}\right) \, d\xi + \int_{0}^{1} F(-t) \log(t) \, dt + 2 \int_{0}^{1} F(t) \log(t) \, dt \quad (D.4)
\]

The singular integrals that occur when using specialized Green's functions are evaluated as follows. The integral is of the form,
\[ I \equiv \int_{-1}^{1} F(\xi) \log[ \frac{C1(\xi) \times C2(\xi)}{\xi^2} ] \, d\xi \quad (D.5) \]

\[ C1(\xi) \equiv \cosh[\pi(y-y_i)] - \cos[\pi(x-x_i)] \quad (D.6) \]

\[ C2(\xi) \equiv \cosh[\pi(y-y_i)] - \cos[\pi(x+x_i)] \quad (D.7) \]

The \( C1 \) function has a root at the point \((x_i, y_i)\). To see the nature of the singularity, the function arguments are expanded. The case of the center element node being the base node is shown.

\[ \cosh[\pi(y-y_i)] \approx 1 + [\pi(y-y_i)]^2/2 \]
\[ \approx 1 + [\pi(\xi(dy/d\xi))]^2/2 \quad (D.8) \]

\[ \cos[\pi(x-x_i)] \approx 1 - [\pi(x-x_i)]^2/2 \]
\[ \approx 1 - [\pi(\xi(dx/d\xi))]^2/2 \quad (D.9) \]

The difference of these quantities is proportional to \( \xi^2 \). Therefore the singularity can be treated as it was for the general Green's function case shown above. The argument of the logarithm is multiplied by \( 1/\xi^2 \) to decompose the integral into a nonsingular integral and two standard form singular integrals.

\[ I = \int_{-1}^{1} F(\xi) \log[ \frac{C1(\xi) \times C2(\xi)}{\xi^2} ] \, d\xi + \]

\[ + 2 \int_{0}^{1} F(-t) \log(t) \, dt + 2 \int_{0}^{1} F(t) \log(t) \, dt \quad (D.10) \]
The logarithmic argument has an additional root when \( x_i = 0 \) or \( x_i = 1 \) and \((x, y)\) approaches \((x_i, y_i)\). In this instance both the C1 and C2 functions have roots. To compensate, the factor used to divide the logarithmic argument is squared, and the remapped singular integrals are doubled. The cases of the left element node being the base point or the right element node being the base point are handled in a similar fashion. The logarithmic argument is multiplied by the same factors shown for the general boundary element method.
APPENDIX E

GBEM COMPUTER CODE LISTING

GBEM CALLING HIERARCHY

GBEM
QUAD
GRID
  SHAPE
MATFIX
ELMT
GRN
MATMOV
ELMT
BC
GRN
SGECO (LINPACK)
SGEFA (LINPACK)
  ISAMAX (LINPACK)
SSCAL (LINPACK)
SAXPY (LINPACK)
SSCAL (LINPACK)
SASUM (LINPACK)
SDOT (LINPACK)
SAXPY (LINPACK)
SGESL (LINPACK)
SAXPY (LINPACK)
SDOT (LINPACK)
ERRLIST
CPU_TIMER

MODULE NOTES

GBEM - main program
- reads the input data (file PTDATA)
- sets miscellaneous parameters
- writes report files (files REPORT, ELECTRODE)
- controls the time integration
- stores temporary matrix terms in file AFSAVE
- deforms the active surface

BC - subroutine
- computes boundary condition terms for the current quadrature point on the active surface.

CPU_TIMER - function (PL/1 programming language)
- calls system vclock() function
- returns elapsed virtual cputime (seconds)

ELMT - subroutine
- calculates mapping related quantities for the current
GBEM Listing Notes

quadrature point on the current element

ERRLIST - subroutine
-computes the error in the current active surface position and the error in the last field solution

GRID - subroutine
-draws initial boundary element mesh

GRN - subroutine
-computes terms related to the Green's function for the current quadrature point and the current base point.

MATFIX - subroutine
-computes matrix problem contributions for the boundary elements that are spatially fixed (not called for the infinite domain problem)

MATMOV - subroutine
-computes matrix problem contributions for the deforming boundary elements
-computes \( H_{ii} \) terms by difference

QUAD - subroutine
-sets up quadrature points and weights
-pre-evaluates basis functions at the quadrature points

SGECO - subroutine (LINPACK)
-factor the main matrix and estimate its condition

SGESL - subroutine (LINPACK)
-solve the linear matrix problem

SHAPE - function
-used to compute the initial active surface shape
-returns surface ordinate given the abscissa

Indicated routines are from the LINPACK library distributed by the Society for Industrial and Applied Mathematics and are documented by Dongarra et al (1980).

SIGNIFICANT COMMON AND INPUT VARIABLES

A(NXY,NXY) - the main matrix, it multiplies the unknowns
AMP - initial amplitude of the active surface
DELTIME - time step size
DPHI(3,NQ) - derivatives of the basis functions with respect to (w.r.t.) the local coordinate, pre-evaluated at the Gauss-Legendre quadrature points (d\( \xi \)/d\( \zeta \))
DSDXI - derivative, change in surface area w.r.t. the local coordinate (dS/d\( \zeta \))
DXDXI - derivative, change in global coordinate X w.r.
GBEM Listing Notes

t. ¥
derivative, dY/d¥
EPSILON - small tolerance for comparison
EVAL - the eigenvalue µ
F(NXY) - the righthand-side vector in the simultaneous
linear equation set, equivalent to SOLN(NXY),
(becomes overwritten with the matrix solution)
FINITE - logical, true for the finite domain problem
FXYN(IB) - rates of surface deformation at each active
           surface node
HDELT A - location of the top surface (finite domain)
HII(NXY) - sum of Hij contributions, i≠j
IB - rightmost node on the active surface, also,
    number of active surface nodes
IC - rightmost node along domain top (finite domain)
ID - leftmost node along domain top (finite domain)
IBCOND - boundary condition choice (1,2, or 3)
ICHOICE(5) - report control, see input data file
INFINITE - true for the infinite domain problem
ITCOUNT - count of completed time step iterations
LNPTR - offset of logarithmic quadrature points and
         weights in WT() and PT()
NSELECT - number of elements along the active surface
NELMTS - total number of elements
NDELT A - number of elements along the top boundary
NLN - number of logarithmic quadrature points
NNODES - total number of nodes
NODE(3) - node indexes on the current element
NO - number of Guass-Legendre quadrature points
NWRITE - active surface location is reported every
         NWRITE time steps
NXY - the size of the matrix problem, also the number
      of nodes
PHI(3,NQ) - the basis functions at the Gauss-Legendre
            quadrature points
PT(NQ+NLN) - the quadrature points
SOLN - the solution vector, equivalent to F
STOPTIME - integration stop time
TTIME - simulated integration time
TYPE - choice of the starting active surface shape
WT(NQ+NLN) - quadrature weights
X(NXY),Y(NXY) - node locations, global coordinates
X0(IB),Y0(IB) - constants related to initial active surface
                position, needed for error reporting
XQ,YQ - coordinates of a given quadrature points
YN(IB) - saved position of the active surface
INCLUDED FILES:

C implicit - specifies precision
IMPLICIT INTEGER (I-N), DOUBLE PRECISION (A-H,O-Z)

COMMON BLOCK

COMMON/BLK1/HDELTA, AMP, TYPE, NELECT, NDELTA, NNODES, NELMTS, IB,
   IC, ID, FINITE, INFINITE, NQ, NLN, WT(52), PT(52), LNPTR
3, PHI(3, 48), DPHI(3, 48), EVAL, EV2HB
   EXPEVHB, IBCOND, PI, PI4, NODE(3), EPSILON
   INTEGER TYPE
LOGICAL INFINITE, INFINITE
C

INPUT DATA FILE (Free format)

4, 0, 3, 0, 3, 1, 4.0, .. nlect, ndelta, ibond, amp, type, hdelta
0.0, 0.0, 1, 4.2, .. deltme, stoptime, nwrite, nq, nlh
1.0, 2.0, -3.0, -4.0, .. 4.5, .. 1choice
   if 1choice(1)>0 get initial report and 1choice(5) will work
   if 1choice(2)>0 get final report
   1choice(3) - not used
   if 1choice(4)>0 get condition record and verbose error report
   if 1choice(5)>0 only calculate field solution once and exit

196
%global card round auto_zero
INCLUDE "COMP"

6-25-83 r1 1.6

Main Program — — — Solution of Laplace's equation by Boundary Integral
Equation Methods on a Finite or Infinite Domain.

GBEM
BEM with Special
Green's Functions

Specialized Green's functions are used to impose
a no flux boundary condition on the parallel
sidewalls at x=0 and x=1.

A source boundary condition is imposed
in any three ways leading to a known solution
for the deformation of the active surface.

Quadratic boundary elements and Improved Euler
integration is used.

%include implicit
PARAMETER IXXYY=1600,IXY=40,IELECT=21
PARAMETER (IXY=140,IXXY=IXY,IELECT=129)

COMMON STATEMENTS ARE GOTTEN FROM INCLUDE FILES

%include blk1
   INCLUDE "BLK1"

A LARGER PROBLEM CAN BE TACKLED WHEN ARRAYS ARE IN COMMON
COMMON/BLKS/X(IXY),Y(IXY),A(IXXY),SOLN(IXY),HIU(IXY),IPVT(IXY)
DIMENSION F(IXY)
EQUIVALENCE (F(1),SOLN(1))
DIMENSION YN(IELECT),FXYN(IELECT),ICHICE(5),XO(IELECT),YO(IELECT)
LOGICAL REPORT,EPRED,ERROR,MFLAG

DIMENSION ITIME(3)

EXTERNAL date_time_(descriptors)
EXTERNAL clock_(descriptors)
CHARACTER*24 TDSTRING
EXTERNAL CPU TIMER (descriptors)
REAL CPUSTART, CPUTIME

-------------- INITIALIZE ----------------

OPEN FOR WRITE
OPEN(2,MODE="IN",FORM="FORMATTED",FILE="ptdata")
OPEN(3,MODE="INOUT",FORM="FORMATTED",FILE="electrode")
OPEN(4,MODE="INOUT",FORM="FORMATTED",FILE="report")
OPEN(20,MODE="INOUT",FORM="UNFORMATTED",FILE="rfsave")
CALL FOPEN (2,"PTDATA")
CALL FOPEN (3,"ELECTRODE")
CALL FOPEN (4,"REPORT")
CALL FOPEN (20,"RFSAVE")

-----READ PROBLEM PARAMETERS-----
READ (2,595) NLECT,NDELTA,IBCOND,AMP,TYPE,HDELTA
READ (2,595) NLECT,NDELTA,IBCOND,AMP,TYPE,HDELTA
READ (2,595) NLECT,NDELTA,IBCOND,AMP,TYPE,HDELTA

595

2ND ROW, INTEGRATION CONTROL
READ FREE(2) DELTIME,STOPTIME,NWRITE,NO,NLN
READ (2,595) DELTIME,STOPTIME,NWRITE,NO,NLN

3RD ROW, MISCELLANEOUS
READ FREE(2) ICHICE
READ (2,595) ICHICE

----------SET MISCELLANEOUS PARAMETERS-------

TOLERANCE FOR STOP TIME COMPARISON
EPSILON=1.0E-06
C SELECT INFINITE OR FINITE DOMAIN BASED ON ELEMENTS ALLOTTED FOR TOP
   FINITE=.TRUE.
   IF (NDELTA.EQ.0) FINITE=.FALSE.
   INFINITE=.TRUE.
   IF (FINITE) INFINITE=.FALSE.
C
C THE NUMBER OF ELEMENTS
   NELMTS=NELECT + NDELTA
C
C IB,IC,ID IDENTIFY CORNER NODE NUMBERS
   IB=NELECT + 1
   IC=1
   ID=IC + 2*NDELTA
C
C THE TOTAL NUMBER OF NODES
   NNODES=ID
   IF(INFINITE) NNODES=IB
C
C THE SIZE OF THE MATRIX PROBLEM
   NXY=NNODES
C
   NXY=NXY*NXY
C
C TEST FOR ADEQUATE STORAGE
   IF (NXY.LE.IXY.AND.IB.LE.IELECT) GOTO 5
   WRITE (4,600) NXY,IB,IXY,IELECT
600   FORMAT("SOLUTION REQUIRES ".I6," NODES WITH ".I6," CATHODE NODES"
   2 "," BUT ",I6," AND ",I5," ARE THE MAXIMUM.")
   STOP
   CONTINUE
C
C THE DOMAIN WIDTH OF 1 IS BUILT INTO THE GREEN'S FUNCTIONS AND
C IS IMPLICIT ELSEWHERE
C
C SET EIGENVALUE AND MISC. CONSTANTS
   PI=3.141592653589793D0
   EVAL=PI
   EV2MB=EVAL**2.0*HDELTA
   EXPVMB=EXP(EVAL*HDELTA)
   PI4 = 4.0*PI
C
C WILL THERE BE AN INITIAL REPORT?
   REPORT=.TRUE.
   IF (ICHOICE(1).LE.0) REPORT=.FALSE.
C
C ZERO INTEGRATION TIME
   TTIME=0.0
C
C ITCOUNT IS USED FOR PERIODIC POSITION REPORTS
   ITCOUNT=0
C
C MFLAG CONTROLS MATRIX CONTRIBUTION CALCULATION
   MFLAG=.TRUE.
C
C-------SET UP QUADRATURE POINTS AND WEIGHTS -------
   CALL QUAD
C
C---------------------------WRITE REPORT INFO TO REPORT FILE ---------------------------
C
C --- START CPU TIMING ----
   CALL CPU_TIMER(CPUSSTART)
   CALL date_time(clock(),TDSTRING)
   WRITE (4,700) TDSTRING
700   FORMAT(15("."),3X,A24,3X,15("."))
C
C CALL TIME(ITIME.IER)
C
C CALL FDAY(MONTH,IDAY,IYEAR)
C
C WRITE (4,700) MONTH,IDAY,IYEAR,ITIME
   WRITE (4,705)
705   FORMAT("Two Dimensional Moving Boundary Study Using",
   2 " Boundary Integrals")
   WRITE(4,706)
706   FORMAT(" Specialized Green's Functions - version 1.6,"
   2 " 72 bit Real Precision")
IF (.NOT.FINITE) WRITE(4,707)  
   FORMAT("SEMI-INFINITE Domain with Analytical Solution/")  
2   * COS(Pi*X)*EXP(-Pi*Y))  
IF (FINITE) WRITE(4,708)  
708   FORMAT("FINITE Domain with Analytical Solution/")  
2   * COS(Pi*X)*([EXP(-Pi*Y)-EXP(Pi*(Y-2*HDELTA))])  
WRITE (4,710) HDELTA,AMP,TYPE,IBCOND  
710   FORMAT("/DOMAIN GEOMETRY/)"  
2 5X,"HDELTA",5X,"AMPLITUDE TYPE Boundary Cond."/2F10.3,2I1O)  
WRITE (4,720) NLECT,NDELT,NELMTS,NNODES  
720   FORMAT("/DISTRIBUTION OF BOUNDARY ELEMENTS/")  
2 6X,"NLECT NDELT TOTAL ELEMENTS TOTAL NODES"  
3 /5I8,118)  
WRITE (4,730) DTIME,STOPTIME,NO,NLN  
730   FORMAT("/INTEGRATION PARAMETERS/")  
2 4X,"DELTIME STOPTIME NORMAL QD PTS"  
3 , " LOG QD PKS"/2F10.5,2I15)  
WRITE (4,738) NWRITE,ICHICE  
738   FORMAT("/PROGRAM CONTROL PARAMETERS/)"  
2 "NWRITE",".5," ICHICE=".5I4")  
C-------ALLOW FOR PERIODIC ELECTRODE LOCATION REPORTS  
C WRITE (3,740) MONTH, IDAY, IYEAR, ITIME  
C740   FORMAT("GBEM PROGRAM RUN ",.I2,"/",.I2,"/".I2,3X,.I2,":",.I2,":",.I2/  
C 2 "TIME, POSITION ERR, SOLN ERR")  
WRITE (3,740) TDSTRING  
740   FORMAT("Gbem Program Run ",A24,"/Time, Position Err, Soln Err")  
C  
C-------ESTABLISH FIXED ELEMENT NODE LOCATIONS--------  
CALL GRID(X,Y,NNODES)  
C WRITE (10,742) (I,X(I),Y(I),I=1,NNODES)  
C742  FORMAT(14,5X,2F10.3)  
C  
C-------SET UP XO, YO FOR PERIODIC ERROR REPORTS--------  
DO 13 I=1,IB  
C TEMP=EXP(EVAL*Y(I))  
C IF (INFINITE) GOTO 11  
C YO(I)= (TEMP+EXPVB)/(TEMP-EXPVB)  
XO(I)=2.0*EXP+COS(EVAL*X(I))/EXPVB  
GOTO 13  
C 11 YO(I)=TEMP  
XO(I)=EXP+COS(EVAL*X(I))  
13 CONTINUE  
C  
C-------CALC MATRIX CONTRIBUTIONS FROM FIXED NODES-------  
IF (FINITE) CALL MATIX(X,Y,NXY,A,HIJ,MFLAG)  
MFLAG=.FALSE.  
C  
C-------SAVE FIXED RESULTS AS A BASE FOR EACH TIME ITERATION-----  
IF (FINITE) WRITE (20) (A(I),I=1,NXXX), (HIJ(I),I=1,NNODES)  
C  
C-------END INITIALIZATION--------------------  
C  
C-------MAIN INTEGRATION LOOP ENTRY POINT------------------  
100 CONTINUE  
C  
C-------SET FLAG FOR EULER PREDICTION-------  
EUPRED=.TRUE.  
C-------SAVE ELECTRODE POSITION IN YN-----  
DO 109 I=1,IB  
109 YN(I)=Y(I)  
C-----110 CONTINUE IS ENTRY POINT AFTER EULER PREDICTION-----  
110 CONTINUE
C----- SOLVE FOR FLUX
C-----------REINITIALIZE MATRIX -----------
  IF (ITCOUNT.EQ.O.AND.EUPRED) GOTO 118
C CLEAR RHS VECTOR
  DO 112 I=1,NXY
    F(I)=0.0
  CONTINUE
  IF(INFINITE) GOTO 114
  REWIND 20
  READ (20)(A(I),I=1,NXXY),(HIJ(I),I=1,NNODES)
  GOTO 118
C INFINITE DOMAIN, THERE ARE NO FIXED CONTRIBUTIONS
C  DO 115 I=1,NXY
C  HIJ(I)=0.0
C  CONTINUE
C  DO 116 I=1,NXXY
C  A(I)=0.0
C  CONTINUE
C-----------ADD CONTRIBUTION FROM MOVING NODES AND FIXED ELEMENTS ----
  IF (FINITE) CALL MATFIX(X,Y,NXY,A,HIJ,MFLAG)
C-----------ADD CONTRIBUTION FROM DEFORMING ELEMENTS ----
  CALL MATMOV(X,Y,NXY,A,F,HIJ)
C WRITE(6,760) (HIJ(I),I=1,NNODES)
C760 FORMAT(G12.5)
C
C FACTOR MATRIX AND ESTIMATE CONDITION NUMBER
  CALL SGEQ (A,NXY,NXY,IPVT,RCOND,HIJ)
C (NOTE - HIJ IS USED AS A WORK VECTOR)
  IF (RCOND.NE.O) RCOND=1.0/RCOND
C NOW SOLVE EQUATION
  CALL SGESX (A,NXY,NXY,IPVT,F,O)
C
C IF(ICHOOSE(4).GT.0)WRITE (4,766) TTIME,RCOND
C766 FORMAT("T",F12.5," Matrix Condition=",G12.5)
C
C------ ON FIRST PASS REPORT THE SOLUTION AND NODAL LAYOUT-----
  IF (.NOT.REPORT) GOTO 127
  CALL ERRLIST(X,Y,F,NNODES,XO,YO,YN,ISELECT,TTIME,4,1)
  CALL ERRLIST(X,Y,F,NNODES,XO,YO,YN,ISELECT,TTIME,3,1)
C DEBUG-----EXIT
  IF(ICHOOSE(5).GT.0)GOTO 1000
C REPORT= .FALSE.
C
C----------------- EULER PREDICTION ------
  IF (.NOT.EUPRED) GOTO 150
    DO 135 I=1,IB
      FXYN(I)=SOLN(I)
    CONTINUE
    PREDICT NEW ELECTRODE POSITION:
      Y(I)=Y(I) + DELTIME*FXYN(I)
    CONTINUE
C ------ NOW FIND FLUX WITH THIS POSITION ----
    EUPRED=.FALSE.
    GO TO 110
C------ HERE FOR TRAPEZOID CORRECTOR ---------------
C150 CONTINUE
C CALCULATE NEW ELECTRODE MOVEMENT RATE AND AVERAGE IT WITH EULER
C PREDICTED RATE
  DO 165 I=1,IB
    FXYN(I)=FXYN(I) + SOLN(I)
  CONTINUE
C MOVE THE ELECTRODE
    SAVE=Y(I)
(I) = YN(I) + DELTIME * FXYN(I) / 2.0
YN(I) = SAVE
CONTINUE

C********** THIS COMPLETES AN INTEGRATION CYCLE **********
ITCOUNT = ITCOUNT + 1
TTIME = TTIME + DELTIME

C WRITE ELECTRODE POSITION EVERY NWRITE CYCLES
IF (MOD(ITCOUNT, NWRITE).NE.0) GO TO 200
CALL ERRLIST(X,Y,F,NNODES,XO,YO,YN,ISELECT,TTIME,3,ICHOICE(4))

C------- CHECK TIME AGAINST STOP TIME
200 IF (TTIME .GE. STOPTIME - EPSILON) GO TO 1000
C RETURN FOR MORE INTEGRATION
C GO TO 100

C ERROR RETURN
900 WRITE(4,781) TTIME
781 FORMAT("ERROR RETURN, SIML. TIME= ",F10.4)

C-------------------- HERE AT END ---------------------
1000 CONTINUE

C CALL OVERFL(I)
IF (I .NE. 2) WRITE (4,782) I
782 FORMAT("OVERFLOW CODE=",I2, " 3=UNDER, 1=OVER OR BOTH")

C FINAL REPORT ON LAYOUT AND SOLN
IF (ICHOICE(2).GT.0) WRITE (4,785)
785 FORMAT(/"****** FINAL NODE LAYOUT *****/
IF (ICHOICE(2).GT.0)
2 CALL ERRLIST(X,Y,F,NNODES,XO,YO,YN,ISELECT,TTIME,4,1)

C CALL date_time(clock(),TDSTRING)
 CALL CPU_TIMER(CPUTIME)
 CPUTIME = CPUTIME - CPUSTART
 WRITE (4,790) TDSTRING,CPUTIME
 WRITE (6,790) TDSTRING,CPUTIME
790 FORMAT(/" ALL DONE AT ",A24, " Elapsed CPU Seconds=",F12.4)
C DTIME = 3600 * ITIME(1) + 60 * ITIME(2) + ITIME(3)
C CALL TIME(ITIME, IER)
C DTIME = 3600 * ITIME(1) + 60 * ITIME(2) + ITIME(3) - DTIME
C WRITE (4,790) ITIME, DTIME
C CALL (10,790) ITIME, DTIME
C790 FORMAT(/"ALL DONE AT ",I2,:I2,":",I2.
C 2 " Elapsed time="F12.5," Seconds")
END
INCLUDE "COMP"
SUBROUTINE BC(QTERM,DQTERM,XQ,YQ,DDDXI,DDYXI)

%include implicit
C 4-19-83
C
COMPUTES BOUNDARY CONDITION TERMS FOR THE CURRENT SURFACE POINT
C
QTERM = INHOMOGENEOUS TERM
DQTERM = HOMOGENEOUS TERM
C BOTH TERMS CARRY INTEGRATION WEIGHTS THROUGH DDXI, DYXI
C BOTH TERMS CARRY DSDXI IMPLICITLY SINCE IT CANCELS OUT
C
INCLUDE "BLK1"
%include blk1

TEMP=EVAL*YQ
EX1=EXP(-TEMP)
EVX=EVAL*XQ
C
IF(FINITE) GOTO 134
C
GOTO (131,132,133), IBCOND
C
131
QTERM=EX1*((-DDXI*SIN(EVX)+DDXI*COS(EVX))
DQTERM=0.0
GOTO 138
132
QTERM=EX1*(-DDXI*SIN(EVX))
DQTERM=DDXI
GOTO 138
133
QTERM=EX1*DDXI*COS(EVX)
DQTERM=-DDXI*TAN(EVX)
GOTO 138
C
C FINITE DOMAIN TERMS
C
134
EX2=EXP(TEMP-EV2HB)
C
GOTO (135,136,137), IBCOND
C
135
QTERM=-DDXI*SIN(EVX)*(EX1-EX2) + DDXI*COS(EVX)*(EX1+EX2)
DQTERM=0.0
GOTO 138
136
QTERM=-DDXI*SIN(EVX)*(EX1-EX2)
DQTERM=DDXI*(EX1+EX2)/(EX1-EX2)
GOTO 138
137
QTERM=DDXI*COS(EVX)*(EX1+EX2)
DQTERM=-DDXI*TAN(EVX)
C
138
QTERM=QTERM=EVAL
DQTERM=DQTERM=EVAL
RETURN
END
C
C
C
INCLUDE "COMP"
SUBROUTINE ELMT(IQ,XQ,YQ,DXDXI,DDYXI,DSDXI,DXDII.X,Y,NXY)
%include implicit
C 4-20-83
C
CALCULATES MAPPING INFO FOR CURRENT ELEMENT, CURRENT QUADRATURE PT
C
(IQ,XQ,YQ) - INTEGRATION POINT, GLOBAL COORDINATES
C (DXDXI,DDYXI) - DERIVATIVES WRT LOCAL XI SURFACE VARIABLE
X
C
INCLUDE "BLK1"
%include blk1
C
XQ=0.0
YQ=0.0

202
DXDXI = 0.0
DYDXI = 0.0
DO 10 K = 1, 13
   XQ = XQ + PHI(K, IQ) * X(NODE(K))
   YQ = YQ + PHI(K, IQ) * Y(NODE(K))
   DXDXI = DXDXI + DPHI(K, IQ) * X(NODE(K))
   DYDXI = DYDXI + DPHI(K, IQ) * Y(NODE(K))
10 CONTINUE

SURFACE DERIVATIVES WILL CARRY THE INTEGRATION WEIGHTS THROUGH

DXDXI = DXDXI * WT(IQ)
DYDXI = DYDXI * WT(IQ)
DSXI = SQRT(DXDXI * DXDXI + DYDXI * DYDXI)
RETURN
END

INCLUDE 'COMP'
SUBROUTINE ERRLIST(X, Y, SOLN, NXY, XO, YO, YN, IE, TTIME, IOUT, ISWITCH)

DIMENSION X(NXY), Y(NXY), SOLN(NXY), XO(IE), YO(IE), YN(IE)

5-9-83
THIS ROUTINE COMPUTES THE ERROR IN THE CATHODE POSITION
THAT HAS ACCUMULATED WITH INTEGRATION TIME AS WELL AS
THE SOLUTION ERROR FOR THE ACTUAL CATHODE POSITION

YO - CONSTANT VECTOR RELATED TO INITIAL CATHODE POSITION
YO = EXP(EVAL*YO) (INFINITE DOMAIN)
YO = ([EXP(EVAL*YO) + EXP(EVAL*HDELTA)]/[EXP(EVAL*YO) -
      EXP(EVAL*HDELTA)]) (FINITE DOMAIN)

XO - CONSTANT VECTOR RELATED TO X POSITIONS
XO = EVAL*COS(EVAL*XO) (INFINITE DOMAIN)
XO = 2*EVAL*COS(EVAL*XO)*EXP(-EVAL*HDELTA)

TTIME - PRESENT TIME
POSITION ERROR IS FOR POSITION OF GRID LOCATION Y(I) AT SIMULATION
TIME TTIME.
SOLUTION ERROR IS FOR GRID POSITIONS IN YN() WHEN SOLN WAS PREVIOUSLY
CALCULATED

INCLUDE BLK1
INCLUDE "BLK1"

IF (ISWITCH.GT.0) WRITE(IOUT, 600) TTIME
600 FORMAT("/" POSITION AND PREVIOUS SOLUTION ERR. SIMUL. TIME=",
        2 F10.5, "X", 1X, "Y", 12X, "POS ERR", 8X, "SOLN", 8X, "SOLN ERR")
SOLRMS = 0.0
POSRMS = 0.0
DO 100 I = 1, IB
100 CONTINUE

SOLTRUE = TRUE SOLUTION
TEMP = EVAL*YN(I)
ETEMP = EXP(-TEMP)
IF (.NOT.INFINITE) ETEMP = ETEMP - EXP(TEMP-EV2HB)
SOLTRUE = COS(EVAL*X(I)) * ETEMP
SOLERR = SOLN(I) - SOLTRUE
SOLRMS = SOLRMS + SOLERR*SOLERR

IF (INFINITE) GOTO 50

CALCULATION OF TRUE CATHODE POSITION FOR FINITE DOMAIN
TEMP = YO(I) * EXP(TTIME * XO(I))
PTRUE = ALG((1.0 + TEMP) * EXPEVHB/(TEMP-1.0))/EVAL

203
GOTO 80

CALCULATION OF TRUE CATHODE POSITION FOR INFINITE DOMAIN
50  PTRUE=AALOG(TTIME*XO(I)+YO(I))/EVAL
C
80  PERR=y(i)-PTRUE
POS RMS=POS RMS+PERR*PERR
C
IF (ISWITCH.GT.0) WRITE(IOUT,700) X(I),Y(I),PERR,SOLN(I),SOLERR
700  FORMAT(5(G14.6,",",))
C
100  CONTINUE
C
SOL RMS=SQR T (SOL RMS/IB)
POS RMS=SQR T (POS RMS/IB)
IF (ISWITCH.GT.0) WRITE (IOUT,750) POS RMS,SOL RMS
750  FORMAT("RMS POSITION ERROR=",G12.5,"RMS SOLUTION ERROR=",
2  G12.5/)
IF (ISWITCH.LE.0) WRITE (IOUT,760) TTIME,POS RMS,SOL RMS
760  FORMAT(3(G12.5,",",))
RETURN
END
C
C
INCLUDE 'COMP'
SUBROUTINE GRID(X,Y,NXY)
%include implicit
C-- 4-18-83 -------------------------------
C Specialized Green's Function Version
C Sets up the boundary node structure for the comparison problem.
Nodes are only located along the deforming boundary described as
a function of X by SHAPE, and when the domain is finite, along a
horizontal strip at Y=HDELTA.
C
ACTIVE SURFACE
- RANGES FROM X=0 TO X=1
- FUNCTION SHAPE(X(I)) SPECIFIES Y AS A FUNCTION OF X
C
NODAL LAYOUT
- QUADRATIC BOUNDARY ELEMENTS
- NODE NUMBERING STARTS WITH X=0 ACTIVE ELECTRODE CORNER AND CONTINUES
  COUNTERCLOCKWISE AROUND THE CELL
- NLECT, # OF ELEMENTS ON ACTIVE SURFACE (*ELECTRODE)
- ND DELTA, # OF ELEMENTS ALONG CELL TOP. IF 0, FINITE IS FALSE
C
-------------------------------------------------------------
DIMENSION X(NXY),Y(NXY)
C
%include blk1
C  INCLUDE "BLK1"
C
C------------LOCATE ACTIVE SURFACE NODES----------------
 DO 20 I=1,IB
  X(I)=FLOAT(I-1)/FLOAT(IB-1)
  Y(I)=SHAPE(X(I))
  IF(Y(I).GE.HDELTA) GOTO 1000
20  CONTINUE
C
C-------- IF FINITE - TOP OF CELL ----------------------
 IF(INFINITE) RETURN
 DO 60 I=IC,ID
  X(I)= 1.0 - FLOAT(I-IC)/FLOAT(ID-IC)
  Y(I)=HDELTA
60  CONTINUE
C
RETURN
C ERROR RETURN
1000 CONTINUE
WRITE(4,1000) Y(1),Y(I),HDELTA
1000 FORMAT(//"Y(1)= ",F12.4," Y(I)= ",F12.4," HDELTA="F12.4/
2 "SHAPE FUNCTION HAS EXCEEDED TOP BOUNDARY"
3 "STOP IN GRIDSTART")
END

C C C
C INCLUDE "COMP"
C SUBROUTINE GRN(GREEN,DGDY,DGDX,CCC,XQ,YQ,X,Y,NXY,INB)
C %include implicit
C 4-29-83
C C CALCULATES TERMS RELATED TO THE GREENS FUNCTION WITH BASE POINT
C X(INB),Y(INB) AND OBSERVATION POINT XQ,YQ
C C DIMENSION X(NXY),Y(NXY)
C INCLUDE "BLK1"
C %include blk1
C
X1=PI*(XQ*X(INB))
X2=PI*(XQ-X(INB))
Y2=PI*(YQ-Y(INB))
C1=COS(X1)
C2=COS(X2)
CH2=COSH(Y2)
C12=-C1+CH2
C22=-C2+CH2
CCC=C12*C22
S1=SIN(X1)
S2=SIN(X2)
SH2=SINH(Y2)
C
GREEN = ALOG(CCCC)/PI4
DGDY = (SH2/C12 + SH2/C22)/4.0
DGDX = (S1/C12 + S2/C22)/4.0
RETURN
END

C C C C C C C
C INCLUDE 'COMP'
C SUBROUTINE MATFIX(X,Y,NXY,A,HIJ,MFLAG)
C 5-1-83
C %include implicit
C DIMENSION X(NXY),Y(NXY),A(NXY,NXY),HIJ(NXY),PHIZ(3)
C LOGICAL MFLAG
C
C %include blk1
C INCLUDE "BLK1"
C
-----------------------------------------------
C MATRIX CONTRIBUTIONS FOR FIXED ELEMENTS AND:
C - FIXED BASE NODES MFLAG=.TRUE.
C - MOVING BASE NODES MFLAG=.FALSE.
C
C ONLY CALLED FOR A FINITE DOMAIN WHERE THERE ARE FIXED ELEMENTS
C HIJ ARE CALCULATED ONLY TO HELP FIND H/I NEEDED ELSEWHERE
C **** ASSUMES FIXED TOP BOUNDARY IS HORIZONTAL WITH EVEN NODE SPACING
C
-----------------------------------------------

205
C-----OUTER LOOP FOR EVERY FIXED ELEMENT----------
  DO 5000 NEL = NELECT+1,NELMTS

C
  NODE(1)=2*NEL
  NODE(2)=NODE(1)+1
  NODE(3)=NODE(2)+1

C---- LOOP FOR NORMAL QUADRATURE POINTS
- USUAL INTEGRALS
- REMAPPED PORTIONS OF SINGULAR INTEGRALS

C  DO 3000 IQ = 1,NO

C  CALCULATE MAPPING DATA FOR THIS QUAD PT
  CALL ELMT(IQ,XQ,YQ,DXDXI,DYDXI,DSSIDX,X,Y,NXY)

C  -- LOOP FOR NODES USED AS A BASE POINT ------

C  ISTART=IC
  ISTOP=ID
  IF(MFLAG) GOTO 20

   ISTART=1
   ISTOP=IB

  20  DO 2000 IBN = ISTART,ISTOP

C--- BUILD COMPONENTS OF THE GREENS FN -------

C  CALL GRN(GREEN,DGDY,DGDY,CCCC,XQ,YQ,X,Y,NXY,IBN)

C  GGG = GREEN * DSDXI
  HHH = -DYDXI*DGGY

C  PLACE CONTRIBUTIONS IN Accord WITH BASIS FUNCTIONS
  DO 2000 J=1,3
  IF(IBN.EQ.NODE(J)) GOTO 1000
  NORMAL CASES:
  HIU(IBN) = HIU(IBN) + HHH*PHI(J,IQ)
  A(IBN,NODE(J)) = A(IBN,NODE(J)) - GGG*PHI(J,IQ)
  GOTO 2000

C  SINGULAR IQUERALS, IBN.EQ.NODE(J)

C  1000  XSING=1.0
  GOTO (1100,1200,1300), J

C  CONTRIBUTIONS TO GIU WITH SINGULARITY REMOVED

  1100  RGRE=4.0/((PT(IQ)+1.0)**2)
  IF(ABS(X(IBN)).LE.EPSILON) XSING=RGRE
  GOTO 1900

C  1200  RGRE=1.0/(PT(IQ)**2)
  GOTO 1900

C  1300  RGRE=4.0/((1.0-PT(IQ))**2)
  IF(ABS(X(IBN)-1.0).LE.EPSILON) XSING=RGRE

C  1900  RGRE=4.0/((CCCC-RGRE)*XSING)/PI4
  A(IBN,IBN) = A(IBN,IBN) -RGRE*DSDXI*PHI(J,IQ)

C  2000  CONTINUE
  3000  CONTINUE

C(CLOSE LOOPS ON BASIS FNS.NORMAL QUAD PTS. BASE NODES)

C(FOR CURRENT ELEMENT:
  LOOP FOR LOG QUADRATURE POINTS - GIU SINGULAR QUADRATURE
  DO 5000 IQ = 1,NLN

C  DXDXI=(X(NODE(3))-X(NODE(1)))/2.0
GBEM Code Listing

SURFACE DERIVATIVES WILL CARRY THE INTEGRATION WEIGHTS THROUGH

\[ \frac{D}{Dx} = \frac{D}{Dx} \times w(t + \text{LN PTR}) \]
\[ \text{DS} = \text{ABS}(\frac{D}{Dx}) \]

\[ \text{TEMP} = \text{DS} \times 4.0 / \pi 4 \]

LEFT NODE ON ELEMENT SINGULARITY - GIU CONTRIBUTION
\[ Z = 2.0 \times P(t + \text{LN PTR}) - 1.0 \]
\[ \text{PHIZ}(1) = Z \times (Z - 1.0) \times 0.5 \]

CENTER NODE ON ELEMENT SINGULARITY - MAP TO TWO INTEGRALS
THAT HAPPEN TO BE EQUAL WITH ASSUMED DOMAIN
\[ \text{PHIZ}(2) = 1.0 - P(t + \text{LN PTR} + 1) \times P(t + \text{LN PTR} + 1) \]

RIGHT NODE
\[ Z = 1.0 - 2.0 \times P(t + \text{LN PTR} + 1) \]
\[ \text{PHIZ}(3) = Z \times (Z + 1.0) \times 0.5 \]

DO 5000 K = 1,3
\[ X \times \text{SING} = 0.0 \]
IF((ABS(X(NODE(K))) - 1.0).LE.EPSILON).OR.(ABS(X(NODE(K)))
\[ \text{SING} = 1.0 \]
FACTR = TEMP * (1.0 + XSING)
\[ \text{A}(\text{NODE}(K), \text{NODE}(K)) = \text{A}(\text{NODE}(K), \text{NODE}(K)) - \text{PHIZ}(K) \times \text{FACTR} \]

5000 CONTINUE
RETURN
END

INCLUDE 'COMP'
SUBROUTINE MATMOV(X,Y,NXY,A,F,HIJ)
C 6-25-83
%include implicit
DIMENSION X(NXY),Y(NXY),A(NXY,NXY),HIJ(NXY),F(NXY)
DIMENSION PHIZ(3),DPHIZ(3)
LOGICAL SINGULAR
%include blk1
INCLUDE "BLK1"

---------------------------------------------------------------------
MATRIX CONTRIBUTIONS FOR DEFORMING ELEMENTS AND ALL BASE NODES.
LOOPING HIERARCHY:
ELEMENT
NORMAL QUAD POINTS
BASE NODES (ALL NODES)
BASIS FNS
SINGULAR QUAD PTS
BASE NODES - BASIS FNS (3 ON CURRENT ELEMENT)

---------------------------------------------------------------------
C-----OUTER LOOP FOR EVERY MOVING ELEMENT-----------------------------
DO 5000 NEL = 1,NELECT
C
\[ \text{NODE}(1) = 2 \times \text{NEL} - 1 \]
\[ \text{NODE}(2) = \text{NODE}(1) + 1 \]
\[ \text{NODE}(3) = \text{NODE}(2) + 1 \]

C----- LOOP FOR NORMAL QUADRATURE POINTS
- USUAL INTEGRALS
- REMAPPED PORTIONS OF SINGULAR INTEGRALS

DO 3000 IQ = 1, NQ

-------- MAPPING RELATED FOR THIS QUAD PT. THIS ELEMENT-----
CALL ELMT(IQ,XQ,YQ,DXDI,DYDI,DSDXI,X,Y,NXY)

-------- BOUNDARY CONDITION TERMS -------------------------------
CALL BC(QTERM, QTERM, XQ, YQ, DXDI, DYDI)

LOOP FOR EVERY NODE USED AS A BASE POINT

DO 2000 IBN = 1, NNODES

-------- BASE NODE SPECIFIC TERMS -------------------------------
BUILD COMPONENTS OF THE GREENS FN
CALL GRN(GREEN, DGDY, DGDX, CCCC, XQ, YQ, X, Y, NXY, IBN)

HHH = -DXDI*DGDY + DYDI*DGDX

DO = GREEN*QTERM

-------- PLACE CONTRIBUTIONS -------------------------------------

SINGULAR = (IBN.EQ.NODE(1)) .OR. (IBN.EQ.NODE(2)) .OR.
             (IBN.EQ.NODE(3))
IF(SINGULAR) GOTO 700
F(IBN) = F(IBN) + GREEN*QTERM

700  DO 2000 J = 1, 3
     HIJTEMP = HHH*PHI(J, IQ)
     IF(IBN.EQ.NODE(J)) GOTO 800
     NORMAL CASES:
     A(IBN,NODE(J)) = A(IBN,NODE(J)) + HIJTEMP - DO = PHI(J, IQ)
     HIJ(IBN) = HIJ(IBN) + HIJTEMP
     GOTO 2000

-- SINGULAR INTEGRALS, IBN.EQ.NODE(J) --------------------------

800  XSING = 1.0
      GOTO (1100, 1200, 1300), J

CONTRIBUTIONS TO GIJ WITH SINGULARITY REMOVED

1100  RGRE = 4.0/((PT(IQ)+1.0)**2)
      IF(ABS(X(IBN)).LE.EPSILON) XSING = RGRE
      GOTO 1900

1200  RGRE = 1.0/(PT(IQ)*PT(IQ))
      GOTO 1900

1300  RGRE = 4.0/((1.0-PT(IQ))**2)
      IF(ABS(X(IBN))-1.0).LE.EPSILON) XSING = RGRE

1900  RGRE = ALG(KCCC*RGRE*XSING)/PI4
      A(IBN, IBN) = A(IBN, IBN) - RGRE*QTERM*PHI(J, IQ)
      F(IBN) = F(IBN) + RGRE*QTERM

2000  CONTINUE

2000  - END LOOP ON BASIS FN, BASE NODE POINT

3000  CONTINUE

3000  - END LOOP ON NORMAL QUADRATURE POINTS

FOR CURRENT ELEMENT:

LOOP FOR LOG QUADRATURE POINTS - GIJ SINGULAR QUADRATURE

DO 5000 IQ = 1, NLN

LOOP FOR THE FOUR SINGULAR INTEGRALS ON THIS ELEMENT:

208
GBEM Code Listing

C 1= LEFT NODE AS BASE POINT, WHOLE ELEMENT MAPS TO 1 INTEGRAL
C 2= CENTER NODE AS BASE POINT - LEFT HALF (-1.0) INTEGRAL
C 3= RIGHT NODE AS BASE POINT, WHOLE ELEMENT BECOMES 1 INTEGRAL
C 4= CENTER NODE AS BASE POINT - RIGHT HALF (0,1) INTEGRAL
C
DO 4000 J=1,4
C
Z=PT(IQ+LNPTR)
GOTO(3100,3200,3300,3400), J
C
3100 Z=2.O*Z - 1.0
FACTR = 4.0
GOTO 3500
C
3200 Z=-Z
FACTR=2.0
GOTO 3500
C
3300 Z= 1.0 - 2.0*Z
FACTR = 4.0
GOTO 3500
C
3400 FACTR=2.0
C
3500 CONTINUE
C FIGURE OUT MAPPING DATA AT THIS POINT
XQ=O.0
YQ=O.0
DXDXI=O.0
DYDXI=O.0
PHIZ(I)= Z*(Z-1.0)*0.5
PHIZ(2)= 1.0-Z*Z
PHIZ(3)=Z*(Z+1.0)*0.5
DPIHIZ(I) = Z-O.5
DPIHIZ(2) = -2.0*Z
DPIHIZ(3) = Z + O.5
DO 3600 K=1,3
XQ=XQ+PHIZ(K)*X(NODE(K))
YQ=YQ+PHIZ(K)*Y(NODE(K))
DXDXI=DXDXI+DPIHIZ(K)*X(NODE(K))
DYDXI=DYDXI+DPIHIZ(K)*Y(NODE(K))
C
3600 CONTINUE
C THE SURFACE DERIVATIVES CARRY THE QUADRATURE WEIGHTS
C THE LOG R SINGULARITY IS IMPPLICIT IN THE POINTS AND WEIGHTS
DXDXI=DXDXI*WT(IQ+LNPTR)
DYDXI=DYDXI*WT(IQ+LNPTR)
C
C ----- COMPUTE BOUNDARY CONDITION TERMS AT THIS POINT
CALL BC(QTERM,DQTERM,XQ,YQ,DXDXI,DXDYI)
C
K=J
IF (J.EQ.4) K=2
XISING=O.0
IF((ABS(X(NODE(K)))-1.0).LE.EPSILON).OR.(ABS(X(NODE(K))
  .LE.EPSILON)) XISING=1.0
FACTR=FACTR*(1.0 + XISING )/PI4
F(NODE(K))=F(NODE(K)) + QTERM*FACTR
A(NODE(K),NODE(K))=A(NODE(K),NODE(K)) -PHIZ(K)*DQTERM*FACTR
CONTINUE
C
4000 CONTINUE
5000 CONTINUE
C
C Hl(J, I):
C NO Hl Contributions have been formed - they are
C calculated by difference, since sum over j, Hl(j) = 0
C
C INFINITE DOMAIN - A VIRTUAL SURFACE IS IMAGINED AT FINITE Y -
C TO CLOSE THE DOMAIN, AND MAKE THE SUM METHOD WORK. THE Hl(j)
C TERMS FROM THIS VIRTUAL SURFACE SUM TO 0.5
C
209
This sum method is important for reliable solutions near the domain corners.

```
FLUXINF = 0.0
IF (INFINITE) FLUXINF = 0.5
DO 7000 I = 1, IB
   A(I, I) = A(I, I) - HIJ(I) - FLUXINF
7000 CONTINUE
RETURN
END
```

INCLUDE 'COMP'
SUBROUTINE QUAD

ASSIGN QUADRATURE PTS AND WEIGHTS ACCORDING TO USER CHOICE GIVEN BY THE VALUES OF NO AND NLN
LOG QUADRATURE PTS AND WEIGHTS AND NORMAL PTS AND WEIGHTS ARE STORED IN THE SAME VECTORS
LNPTR IS AN OFFSET TO THE LOG VALUES

INCLUDE "BLK1"

```
48 IF (NO .NE. 48) GOTO 32
PT(1) = .03238 01709 62869 36203300
PT(3) = .09700 46992 09462 69893300
PT(5) = .16122 23560 68891 71805600
PT(7) = .22476 37903 94689 06122500
PT(9) = .28736 24873 55455 57673600
PT(11) = .34875 58862 92160 73816000
PT(13) = .40868 64819 90716 72916000
PT(15) = .46690 29047 50958 40454500
PT(17) = .52316 09747 22223 03367800
PT(19) = .57722 47260 83972 70381800
PT(21) = .62886 73967 76513 62399500
PT(23) = .67787 23796 32663 90521200
PT(25) = .72403 41309 23814 65467400
PT(27) = .76715 90325 15740 33925400
PT(29) = .80706 62040 29442 62708300
PT(31) = .84358 82616 24393 53071100
PT(33) = .87657 20202 74247 88590600
PT(35) = .90587 91367 15569 67282200
PT(37) = .93318 66907 06554 33114000
PT(39) = .95298 77031 60430 86072300
PT(41) = .97059 15925 46247 25046100
PT(43) = .98412 45837 22826 85774500
PT(45) = .99353 01722 66350 75754800
PT(47) = .99877 10072 52426 11860100
WT(1) = .06473 76968 12683 92250300
WT(3) = .06446 61644 35950 08220700
WT(5) = .06392 42385 84648 18662400
WT(7) = .06311 41922 86254 02565700
WT(9) = .06203 94231 59892 66390400
WT(11) = .06070 44391 65893 88003500
WT(13) = .05911 48396 93295 63574600
WT(15) = .05727 72821 00403 21970500
WT(17) = .05519 95036 99884 16298600
WT(19) = .05289 01894 85193 66709600
WT(21) = .05035 90355 52854 47495800
WT(23) = .04761 65584 92490 47482600
WT(25) = .04467 45608 56694 28041900
```
GBEM Code Listing

WT(27) = .04154 50829 43464 749214DO
WT(29) = .03824 13510 65830 706317DO
WT(31) = .03477 72225 64770 438893DO
WT(33) = .03116 72278 32798 088902DO
WT(35) = .02742 65097 08356 948200DO
WT(37) = .02357 07608 39324 379141DO
WT(39) = .01961 61604 57355 527814DO
WT(41) = .01557 93157 22943 848728DO
WT(43) = .01147 72345 79234 539490DO
WT(45) = .00732 75539 01276 262102DO
WT(47) = .00315 33460 52305 838633DO
GO TO 100

32 IF (NQ .NE. 32) GOTO 20
PT(1) = .04830 76656 87738DO
PT(3) = .14447 19615 82796DO
PT(5) = .23928 73622 52137DO
PT(7) = .33186 86022 8128DO
PT(9) = .42135 12761 30635DO
PT(11) = .50689 99068 32229DO
PT(13) = .58771 57572 40752DO
PT(15) = .66204 42669 30215DO
PT(17) = .73218 21187 40293DO
PT(19) = .79448 37959 67942DO
PT(21) = .84936 76137 32570DO
PT(23) = .89632 11557 66052DO
PT(25) = .93490 60759 37740DO
PT(27) = .96475 22555 87505DO
PT(29) = .98561 15115 45268DO
PT(31) = .99726 38618 49482DO
WT(1) = .09654 00885 14728DO
WT(3) = .08563 87200 79275DO
WT(5) = .09384 43990 80805DO
WT(7) = .09117 38786 95764DO
WT(9) = .08765 20930 04404DO
WT(11) = .08331 19242 26947DO
WT(13) = .07819 38957 87070DO
WT(15) = .07234 57941 08849DO
WT(17) = .06582 22227 76362DO
WT(19) = .05868 40834 78536DO
WT(21) = .05099 80592 82376DO
WT(23) = .04283 58980 22227DO
WT(25) = .03427 38629 13021DO
WT(27) = .02539 20653 09262DO
WT(29) = .01627 43947 30908DO
WT(31) = .00701 86100 09470 0970D
GO TO 100

20 IF (NQ .NE. 20) GOTO 16
PT(1) = .07652 65211 33497DO
PT(3) = .22778 58511 41645DO
PT(5) = .37370 60887 15420DO
PT(7) = .51086 70019 50827DO
PT(9) = .63605 36807 26515DO
PT(11) = .74633 19064 50151DO
PT(13) = .83911 69718 22219DO
PT(15) = .91223 44282 51326DO
PT(17) = .96397 19272 77914DO
PT(19) = .99312 85991 85095DO
WT(1) = .15275 33871 30726DO
WT(3) = .14917 29864 72604DO
WT(5) = .14208 61033 18382DO
WT(7) = .13168 86384 49177DO
WT(9) = .11119 45319 61518DO
WT(11) = .10193 01199 17240DO
WT(13) = .08327 67415 76705DO
WT(15) = .06287 20483 34108DO
WT(17) = .04610 14298 00387DO
WT(19) = .01761 40071 39152DO
GO TO 100

16 IF (NQ .NE. 16) GOTO 12
PT(1) = .09501 25098 37637DO

211
GBEM Code Listing

PT(3) = .28160 35507 79259DD
PT(5) = .45801 67776 57227DD
PT(7) = .61787 62444 02644DD
PT(9) = .75540 44083 55003DD
PT(11) = .86563 12023 87832DD
PT(13) = .94457 50230 73233DD
PT(15) = .98940 09349 91650DD
WT(1) = 18945 06104 55068DD
WT(3) = 18260 34150 44924DD
WT(5) = 16815 65193 95003DD
WT(7) = 14959 59888 16577DD
WT(9) = 12462 89712 55534DD
WT(11) = .09515 85116 82492DD
WT(13) = .06225 35239 38648DD
WT(15) = .02715 24594 11754DD
GOTO 100

12 IF(NQ.NE.12) GOTO 10
PT(1) = 12523 34085 11469DD
PT(3) = 36783 14989 98180DD
PT(5) = 58731 79642 86617DD
PT(7) = 76890 26741 94305DD
PT(9) = .90411 72563 70475DD
PT(11) = .98156 06342 46719DD
WT(1) = 24914 70458 13403DD
WT(3) = 23349 25365 38955DD
WT(5) = 20316 74267 23066DD
WT(7) = 16007 83285 43346DD
WT(9) = 10693 93259 95318DD
WT(11) = .04717 53363 86512DD
GOTO 100

10 IF(NQ.NE.10) GOTO 6
PT(1) = 1488743389181631DD
PT(3) = 43395394129247DD
PT(5) = 679405868298024DD
PT(7) = .865063366889885DD
PT(9) = .973906528517172DD
WT(1) = 295524224714753DD
WT(3) = 269266719308999DD
WT(5) = 219086362515992DD
WT(7) = 149451349150581DD
WT(9) = .066671344308688DD
GOTO 100

6 IF(NQ.NE.6) GOTO 4
PT(1) = 238619186031397DD
PT(3) = 661209386468265DD
PT(5) = 32468614203152DD
WT(1) = 467913934572691DD
WT(3) = 360761573048139DD
WT(5) = 171324492379170DD
GOTO 100

4 NQ=4
C (DEFAULT EVEN IF NQ.NE.4)
PT(1) = .339981042584856DD
PT(3) = .861136311594053DD
WT(1) = .652145154862546DD
WT(3) = .347854845137454DD
C Assign even pts and weights
100 DD 110 I=2,NQ.2
PT(I) = PT(I-1)
WT(I) = WT(I-1)
CONTINUE
110 IF( NLN.NE.4 ) GOTO 2
PT(LNPTR+1) = .04144848DD
PT(LNPTR+2) = .24527491DD
PT(LNPTR+3) = .55616545DD
PT(LNPTR+4) = .84898239DD
WT(LNPTR+1) = .38346406DD
WT(LNPTR+2) = .36687531DD
WT(LNPTR+3) = .19043512DD
WT(LNPTR+4) = .03922548DD
GO TO 200
NLN=2
C DEFAULT
PT(LNPR+1)=.11200820DO
PT(LNPR+2)=.60227690DO
WT(LNPR+1)=-.71853813DO
WT(LNPR+2)=-.28146187DO
200 CONTINUE
C PRE-EVALUATE BASIS FUNCTIONS AT THE QUADRATURE POINTS
DO 300 IQ=1,NQ
   Z=PT(IQ)
   PHI(1,IQ)= Z*(Z-1.0)*.5
   PHI(2,IQ)= 1.0-Z*Z
   PHI(3,IQ)= Z*(Z+1.0)*.5
   DPHI(1,IQ)= Z-.5
   DPHI(2,IQ)= -2.0*Z
   DPHI(3,IQ)= Z+.5
300 CONTINUE
C RETURN
END
C SGECO, SGEFA, SGESL, etc. - See Dongarra et al. 1979 (LINPACK).
C
C INCLUDE 'COMP'
FUNCTION SHAPE(Z)
  %include implicit
  %include "BLK1"
  %include blk1
  C THIS FUNCTION DETERMINES THE INITIAL CATHODE SHAPE BASED
  C ON THE INTEGER VARIABLE 'TYPE' AND THE FACTOR 'AMP' FOUND
  C IN THE COMMON BLOCK BLK1. THE FUNCTION SUPPLIES VALUES
  C OF Y WHEN CALLED WITH VALUES OF X AS ITS ARGUMENT. VALUES
  C OF X WILL RANGE BETWEEN 0 AND 1.
  C
  GO TO (10,20,30),TYPE
  C TYPE=1, COSINE WAVE, SCALED FROM (0,0) TO (1,AMP)
  10 SHAPE=AMP*0.5*(1.0-COS(Z*PI))
     RETURN
  C TYPE=2, TRIANGLE WAVE
  20 SHAPE=AMP*Z
     RETURN
  C TYPE=3, COSINE WAVE, OF PERIOD 1
  30 SHAPE=AMP*0.5*(1.0-COS(Z*2.0*PI))
     RETURN
END
APPENDIX F

FEM COMPUTER CODE LISTING

FEM CALLING HIERARCHY

FEM
  BHMAXA
  QUAD
  GRIDSTART
    SHAPE
  REGRID
  MATFORM
    QUMAP
    CHECKSIGN
    MAP
  MATNL
    MAP
  COLSOL
  ERRLIST
  CPU_TIMER

MODULE NOTES

FEM - main program
- reads the input data (file PTDATA)
- sets miscellaneous parameters
- writes report files (files REPORT, ELECTRODE)
- controls the time integration
- stores temporary matrix terms in file ASAVE
- deforms the active surface

BHMMAXA - subroutine
- sets up the vector MAXA which is used to index into the main A matrix, for "skyline" storage

CHECKSIGN - subroutine
- checks the sign of the Jacobian

COLSOL - subroutine
- Solves the banded, symmetric, simultaneous linear equation set

CPU_TIMER - function (PL/1 programming language)
- calls system vclock() function
- returns elapsed virtual cputime (seconds)

ERRLIST - subroutine
- computes the error in the current active surface position and the error in the last field solution

GRIDSTART - subroutine
FEM Listing Notes

-draws the initial finite element mesh

MAP - statement function
-indexing function for using the main matrix, relies on vector MAXA

MATFORM - subroutine
-computes matrix problem contributions (area integrals) for the 2-D elements.

MATNL - subroutine
-computes matrix problem contributions from the boundary condition on the active surface

QUAD - subroutine
-sets up quadrature points and weights
-pre-evaluates basis functions at the quadrature points

QUMAP - subroutine
-calculates mapping related quantities for the current quadrature point on the current biquadratic element

REGRID - subroutine
-adjusts node spacing in region close to active surface

SHAPE - function
-used to compute the initial active surface shape
-returns surface ordinate given the abscissa

SIGNIFICANT COMMON AND INPUT VARIABLES

A(NWK) -the main matrix, it multiplies the unknowns
AMP -initial amplitude of the active surface
DELTIME -time step size
DFDECAY(3,6) -derivatives of the basis fns. (times decay fns.) with respect to (w.r.t.) the local coordinate, pre-evaluated at the quadrature points
DPHI(3,6) -derivatives of the basis fns. the local coordinate, pre-evaluated at the quadrature points (dθi/dς)
DPHIDX(9) -derivatives of the basis fns. w.r.t. global coordinate, X
DPHIDY(9) -derivatives of the basis fns. w.r.t. global coordinate, Y
EPSILON -small tolerance for comparison
ETA0 -ς₀, decay function pole
EVAL -the eigenvalue μ (input data should be 1.0)
FINITE -logical, true for the finite domain problem
FXYN(18) -rates of surface deformation at each active surface node
HBULK -location of the top surface (finite domain) or of the topmost nodes (infinite domain)
HLOCAL -Top of deforming region
FEM Listing Notes

HIJ(NXY) - sum of H_{ij} contributions, i≠j
IB - rightmost node on the active surface, also, number of active surface nodes
IC - rightmost node along domain top (finite domain)
ID - leftmost node along domain top (finite domain)
IBCOND - boundary condition choice (1,2, or 3)
ICHOICE(5) - report control, see input data file
INFINITE - true for the infinite domain problem
ITCOUNT - count of completed time step iterations
IVMAX - total number of element rows
MAXA(NNM) - indexing vector for storage in A
NBULK - number of element rows in the fixed region
NELECT - number of elements along the active surface hence the number of element columns
NELMTS - total number of elements
NLOCAL - number of element rows in the deforming region
NNODES - total number of nodes
NODE(9) - node indexes on the current element
NQ - number of Gauss-Legendre quadrature points
NRWLOCAL - number of node rows in the deforming region
NWRITE - active surface location is reported every NWRITE time steps
PDECAY(3,6) - normal basis fns. times decay fns. at the quadrature points
PHI(3,6) - the basis functions at the quadrature points
PIF(3) - infinite element quadrature points
PT(3) - the quadrature points
R(NXY) - the righthand-side vector in the simultaneous linear equation set (becomes overwritten with the matrix solution)
SCALAR - the field solution is multiplied by this scalar (it was kept at 1.0)
STOPTIME - integration stop time
TTIME - simulated integration time
TYPE - choice of the starting active surface shape
VSBUK - vertical spacing factor, fixed region, 0.0 for uniform spacing
VSLOCAL - spacing factor, deforming region, 0.0 for uniform vertical node spacing
WSECT - domain width, always 1
WIF(3) - infinite element quadrature weights
WT(3) - quadrature weights
WTWT(3,6) - 2 sets of 3×3 quadrature weights for normal and infinite elements
X(NXY),Y(NXY) - node locations, global coordinates
XO(IB),YO(IB) - constants related to initial active surface position, needed for error reporting
XQ,YQ - coordinates of a given quadrature points
XX(3,3) - abscissae of the current element
YACOB - the Jacobian
YN(IB) - saved position of the active surface
YY(3,3) - ordinates of the current element

216
INCLUDED FILES:

C implicit - specifies precision
IMPLICIT INTEGER (I-N), DOUBLE PRECISION (A-H,O-Z)

COMMON BLOCKS

COMMON/BLK1/HLOCAL,HBULK,WELECT,VSLOCAL,VSBUtK,AMP,TYPE
INTEGER TYPE
COMMON/BLK2/NELECT,NLOCAL,NBULK,NRWLOCAL,IB,NNODES,IVMAX
COMMON/BLK3/NQ,WT(3),PT(3),WIF(3),PIF(3),ETAO,WTWT(3,6),PHI(3,6)
2,DPHI(3,6),DPDECAY(3,6),DPDECAY(3,6),INFINITE
LOGICAL INFINITE
COMMON/BLK4/XX(3,3),YY(3,3),DPHIDX(9),DPHIDY(9),NODE(9),YACOBI
COMMON/BLK5/EVAL,SEVAL,SCALAR,EV2HB,EXPEVHB,IBCND

INPUT DATA FILE (Free format)

1.0,4.0,0.0,0.0,+.3,1,; HLOCAL,HBULK,VSLOCAL,VSBUtK,AMplitude,TYPeOfshape
6,3,3,T,-2.0,; NELECT,NLOCAL,NBULK,INFINItE,ETAO
0.0,0.0 ,;DELTImE,STOPTImE,
1,+1,+2,-3,-4,+5 ,;NWriTE.ICHOICE(5)
1.0,+1.0,1 ; EIGENVALUE,SCALAR, IBCOND
IF ICHOICE(2)>0 GET FINAL REPORT
IF ICHOICE(3)>0 GET GRID LAYOUT
IF ICHOICE(4)>0 GET MATRIX CONDITION RECORD AND VERBOSE ERROR RECORD
IF ICHOICE(5)>0 GET EXIT AFTER 1ST SOLN (NO TIME INTEGRATION)

217
%global card round auto_zero
%include 'implicit'
PARAMETER (IWK=60000, IXY=1089, INM=IIY+1, IELECT=33)
C PARAMETER (IWK=149242, IXY=1681, INM=IIY+1, IELECT=49)
PARAMETER IWK=3772, IXY=169, INM=170, IELECT=17
INCLUDE 'COMP'
C 4-15-83 1.3 -------------------------------------------------------------
C
C MAIN PROGRAM -- -- FINITE ELEMENT SOLUTION OF LAPLACE'S EQUATION
C IN A FINITE OR INFINITE SYMMETRIC DOMAIN.
C A CHOICE OF BOUNDARY CONDITIONS IS IMPOSED AT THE ACTIVE
C "ELECTRODE" SURFACE, AND A KNOWN SOLUTION RESULTS AS
C THE SURFACE IS MADE TO DEFORM WITH TIME.
C BIQUADRATIC ELEMENTS AND IMPROVED EULER INTEGRATION ARE USED.
-------------------------------------------------------------
C
C COMMON STATEMENTS ARE GOTTEN FROM INCLUDE FILES
%include blk1
%include blk2
%include blk3
%include blk4
%include blk5
C
DIMENSION X(IXY),Y(IXY),A(IWK),R(IXY),MAXA(INM)
C A LARGER PROBLEM CAN BE TACKLED WHEN ARRAYS ARE PUT INTO COMMON
COMMON/BLKG/X(IXY),Y(IXY),A(IWK),R(IXY),MAXA(INM)
C
DIMENSION YN(IELECT),FXYN(IELECT),ICHICE(5),X0(IELECT),YO(IELECT)
LOGICAL REPORT,EUPEPRED,ERROR,FLAG
C
SYSTEM DEPENDENT TIMING RELATED
DIMENSION ITIME(3)
EXTERNAL datatime,(descriptors)
EXTERNAL clock,(descriptors)
CHARACTER*24 DETAILS
EXTERNAL CPU_TIME,(descriptors)
REAL CPUSTART,CPUTIME
C
------------------------------------------------------ INITIALIZE -------------------------------
C
C------OPEN INPUT FILE------
OPEN(2,MODE="IN",FORM="FORMATTED",FILE="ptdata")
CALL FOPEN (2,"PTDATA")
C
C------READ PROBLEM PARAMETERS------
C 1ST ROW GEOMETRICAL DATA
READ (2,595) HLOCAL,HBULK,VSLOCAL,VSBULK,AMP,TYPE
595 FORMAT(V)
C 2ND ROW ELEMENT LAYOUT DISTRIBUTION
READ (2,595) NELECT,NLOCAL,NBULK,INFINITE,ETAO
C 3RD ROW, INTEGRATION CONTROL
READ (2,595) DELTIME,STOPTIME
C 4TH ROW, REPORT CONTROL
READ (2,595) NWRITE,ICHICE
C 5TH ROW BOUNDARY CONDITION PARAMETERS
READ (2,595) EVAL,SCALAR,IBCND
C
C------ OPEN OUTPUT FILES ------
C
OPEN(3,MODE="INOUT",FORM="FORMATTED",FILE="electrode")
OPEN(4,MODE="INOUT",FORM="FORMATTED",FILE="report")

218
OPEN(20,MODE="INOUT",FORM="UNFORMATTED",FILE="asave")
CALL OPEN (3, "ELECTRODE")
CALL OPEN (4, "REPORT")
CALL OPEN (20, "ASAVE")

C-------------SET MISCELLANEOUS PARAMETERS------

C TOLERANCE FOR STOP TIME COMPARISON
EPSILON=1.0E-06

C IB IS THE NUMBER OF THE RIGHT SIDE CORNER ELECTRODE NODE
C IT ALSO IDENTIFIES THE NUMBER OF NODES PER NODE ROW
IB=2*NELECT + 1

C NNODES - THE TOTAL NUMBER OF NODES
C ALSO THE NUMBER OF Unknowns, GIVEN INFINITE BOUNDARY CONDITIONS
NNODES=(2*(NLOCAL+NBUCK)+1)*IB
NSOLN=NNODES

C FOR FINITE DOMAIN, TOP ROW IS SPECIFIED AND # OF Unknowns IS LESS
IF (.NOT.INFINITE) NSOLN=NNODES-IB
NNM=NNODES+1
NELMTS=(NLOCAL+NBUCK)*NELECT
IF (NNODES.LE.1) GOTO 2
WRITE (4,600) NNODES,IXY

600 FORMAT("SOLUTION REQUIRES ",I4," NODES, BUT ONLY ",I4,
2 " DIMENSIONED")
STOP

2 CONTINUE
IF (NELECT.LE.(IELECT-1)/2) GOTO 4
WRITE (4,610) NELECT,IELECT

610 FORMAT("TOO MANY ELECTRODE ELEMENTS"/"NELECT=",I4,"IELECT=",I4/)
STOP

4 CONTINUE

C NROWLOCAL - THE ROW NUMBER THAT IS THE TOP OF THE LOCAL DEFORMING
C REGION OF ELEMENTS
NROWLOCAL=2*NLOCAL +1

C IVMAX, THE TOTAL NUMBER OF ELEMENT ROWS (VERTICAL COUNT)
IVMAX=NLOCAL+NBUCK

C SET UP MAXA - INDEXING VECTOR FOR PACKED MATRIX A
CALL BIOMAXA(MAXA,NNM)

C NWK - THE SIZE OF THE MATRIX A THAT IS USED
NWK=MAXA(NNM)
IF(NWK.LE.IWK) GOTO 6
WRITE(4,620) NWK,IKW

620 FORMAT("MATRIX TOO LARGE"/"NWK=",I5,"IKW=",I5/)
STOP

6 CONTINUE

C SET ELECTRODE WIDTH - - IT IS THE REFERENCE LENGTH
WELECT=1.0

C SET EIGENVALUE
C SEVAL= SCALAR TIMES THE EIGENVALUE
SEVAL=FLOAT(INT(EVAL))*3.141592653589793D0/WELECT
SEVAL=SCALAR*EV

C EV2HB = CONSTANT USED IN FINITE DOMAIN ANALYTICAL SOLN
EV2HB=SEVAL*HBULK*2.0

C EXPnevH IS USED TO CALCULATE ANALYTICAL RESULTS IN ERRLIST
EXPnevH=EXP(SEVAL*HBULK)

C
C REPORT=.TRUE.
IF (Ichoice(1).LE.0) REPORT=.FALSE.
C TTIME - SIMULATION TIME
TTIME=0.0
ITCOUNT=0
C
C CHECK REFERENCE POINT IN LOCAL ELEMENT COORDINATES FOR 1/R^3 TYPE
C DECAY OF INFINITE ELEMENTS. REFERENCE POINT IS SELECTED SO THAT
C 3 BY 3 QUADRATURE WILL GIVE ACCURATE RESULTS.
IF (ETAO.GE.-1) ETAO=-8.5525
C
C
C C-----SET UP QUADRATURE POINTS AND WEIGHTS ----- CALL QUAD
C
C ------------------------WRITE REPORT INFO TO REPORT FILE ------------------------
CALL CPU_TIMER(CPUSTART)
CALL date_time(clock(),TDSTRING)
WRITE (4,700) TDSTRING
700 FORMAT('/15(**)*,3X,A24,3X,15(**)'/)
C CALL TIME(ITIME,IER)
C CALL FGDAY (MONTH,IDAY,IYEAR)
C WRITE (4,700) MONTH,IDAY,IYEAR,ITIME
C700 FORMAT('/15(**)*,3X,I2,"$/",I2,"$/",I2,3X,I2,:":",I2,:":",I2,3X,I2,
C 2 15(**)/)
WRITE (4,705)
705 FORMAT(" Two Dimensional Moving Boundary"
2 2 "Study using Biquadratic FINITE ELEMENTS (r1 1.3)"
2 IF (INFINITE) WRITE (4,707) SCALAR
707 FORMAT("SEMI-INFINITE Domain with Analytical Potential Solution"
2 F10.3,"Cos(Pi * X)*Exp(-Pi * Y)"
2 IF (.NOT.INFINITE) WRITE (4,709) SCALAR
709 FORMAT("FINITE DOMAIN with Analytical Potential Solution"
2 5X,F10.3,"Cos(Pi*X)*( Exp(-Pi*Y) - Exp(Pi*(Y-2*HBUK)) )"
2 WRITE (4,710) IBCOND
710 FORMAT("Boundary Condition type ",I2,", 72 bit Dbl. Prec.")
WRITE (4,715) HLOCAL,HBUK,WSELECT,VSLOCAL,VSBUK,AMP,TYP
715 FORMAT("DOMAIN GEOMETRY"
WRITE (4,720) NSELECT,NLOCAL,NBUK,NELMTS,NNODES,NWK
720 FORMAT("DISTRIBUTION OF ELEMENTS"
2 5X,"NSELECT",NLOCAL,NBUK,NODES,Matrix size"
3 /3I8,1I0,2I12)
WRITE (4,730) DELTIME,STOPTIME,NO
730 FORMAT("INTEGRATION PARAMETERS"
2 3X,"DELTIME",STOPTIME,QUADRATURE PTS"
3 2F10.5,1I5)
WRITE (4,735) NWRITE,ICHOICE
735 FORMAT("PROGRAM CONTROL PARAMETERS"
4"NWRITE ",ELECTRODE REPORT TIME STEP INTERVAL ",2X,I4/
6"ICHOICE ",REPORT CONTROL ",5I4/)
C
C C-----ALLOW FOR PERIODIC ELECTRODE LOCATION REPORTS
WRITE (3,740) TDSTRING
740 FORMAT("FEM PROGRAM RUN ",A24,"TIME, POSITION ERR, SOLN ERR"
2 ", "FLUX ERR"
C WRITE(3,740) MONTH,IDAY,IYEAR,ITIME
C740 FORMAT("FEM PROGRAM RUN ",I2,"/",I2,"/",I2,3X,I2,": ",I2,:": ",I2,/
C 2 "TIME, POSITION ERR, SOLN ERR")
C
C C C------ESTABLISH FIXED ELEMENT NODE LOCATIONS ------- CALL GRIDSTART(X,Y,NNODES)
C
C C------ SET UP XO, YO FOR PERIODIC ERROR REPORTS -----
DO 13 I=1,1B

220
FEM Code Listing

C

TEMP=EXP(EVAL*Y(I))
C
IF (INFINITE) GOTO 11
C
YO(I)=(*TEMP+EXPEVHB)/(TEMP-EXPEVHB)
XO(I)=2.0*EVAL*SCALE*COS(EVAL*X(I))/EXPEVHB
GOTO 13
C
11 YO(I)=TEMP
XO(I)=EVAL*SCALE*COS(EVAL*X(I))
C
C------------CALC MATRIX CONTRIBUTIONS FROM FIXED NODES --------
FLAG=.FALSE.
CALL MATFORM(A,NWK,X,Y,NNODES,MAXA,NNM,ERROR,FLAG)
IF (ERROR) GOTO 800
C
FLAG INDICATES FURTHER CALLS TO MATFORM ARE FOR DEFORMING ELEMENTS
FLAG=.TRUE.
C
C------ SAVE FIXED RESULTS AS A BASE FOR EACH TIME ITERATION ------
WRITE (20) (A(I),I=1,NWK)
C
C
C----------------------------- END INITIALIZATION -------------------
C
C
C--------------------------- MAIN INTEGRATION LOOP ENTRY POINT -------
100 CONTINUE
C
C------SET FLAG FOR EULER PREDICTION ------
EUPRED=.TRUE.
C------SAVE ELECTRODE POSITION IN YN ------
DO 109 I=1,1B
C
109 YN(I)=Y(I)
C------110 CONTINUE IS ENTRY POINT AFTER EULER PREDICTOR ----
110 CONTINUE
C
C Adjust deforming elements to reflect position of active surface
CALL REGRID(X,Y,NNODES,ERROR)
C
C---- SOLVE FOR FLUX
C REINITIALIZE MATRIX
IF (ITCOUNT.EQ.0.AND.EUPRED) GOTO 113
REWIND 20
READ (20) (A(I),I=1,NWK)
DO 112 I=1,NNODES
R(I)=0.0
112 CONTINUE
C
C--------ADD CONTRIBUTION FROM MOVING NODES ---------------
113 CALL MATFORM(A,NWK,X,Y,NNODES,MAXA,NNM,ERROR,FLAG)
IF (ERROR) GOTO 800
C
C------ A REFLECTS GEOMETRY AT THIS TIME ------
C
C ADD SURFACE INTEGRAL CONTRIBUTIONS INTO MATRIX EQUATIONS
C
CALL MATNL(A,NWK,X,Y,NNODES,MAXA,NNM,R)
C
C BANDED SYMMETRIC MATRIX SOLVER
C
CALL COLSOL (A,R,MAXA,NSOLN,NWK,NNM,4,PMAX,PMIN,DET,NDET)
C
C

221
C RATIO OF LARGEST PIVOT TO SMALLEST IS CONDITION ESTIMATE
CNDTN=ABS(PMAX/PMIN)
   IF (ICHOOSE(4).GT.0) WRITE (4,760) CNDTN,PMIN,PMAX,DET,NDET
760 FORMAT("Matrix Condition=",E11.5,5X,"Min. and Max. Pivots=
     2  ,2E12.5 /"Determinant",E12.5," * 10^",I4)
C
C-------- ON FIRST PASS REPORT THE SOLUTION AND NODAL LAYOUT------
IF (.NOT.REPORT) GOTO 127
   CALL ERRLIST(X,Y,R,NNODES,XO,YO,YN,IB,TTIME,4,1)
   CALL ERRLIST(X,Y,R,NNODES,XO,YO,YN,IB,TTIME,3,ICHOOSE(4))
C
C CHANCE TO EXIT WITH INITIAL SOLN
   IF (ICHOOSE(5).GT.0) GOTO 1000
C
C REPORT=.FALSE.
C
C---------------- EULER PREDICTION ------
127   IF (.NOT.EUPRED) GO TO 150
C
   DO 130 I=1,IB
      FXYN(I)=R(I)
C
   PREDICT NEW ELECTRODE POSITION:
      Y(I)=Y(I) + DELTIME*FXYN(I)
130   CONTINUE
C
C------- NOW FIND FLUX WITH THIS POSITION ----
EUPRED=.FALSE.
   GO TO 110
C
C---------- HERE FOR TRAPEZOID CORRECTOR  ----------
150   CONTINUE
C
C CALCULATE NEW ELECTRODE MOVEMENT RATE AND AVERAGE IT WITH EULER
C PREDICTED RATE
   DO 165 I=1,IB
      FXYN(I)=FXYN(I) + R(I)
C
   MOVE THE ELECTRODE
      SAVE=Y(I)
      Y(I)=YN(I) + DELTIME*FXYN(I)/2.0
      YN(I)=SAVE
165   CONTINUE
C
C----------------- THIS COMPLETES AN INTEGRATION CYCLE ---------------
   ITCOUNT=ITCOUNT +1
   TTIME=TTIME + DELTIME
C
C WRITE ELECTRODE POSITION EVERY NWRITE CYCLES
   IF (MOD(ITCOUNT,NWRITE).NE.0) GO TO 200
   CALL ERRLIST(X,Y,R,NNODES,XO,YO,YN,IB,TTIME,3,ICHOOSE(4))
C
C-------- CHECK TIME AGAINST STOP TIME
200   IF (TTIME.GE.STOPTIME-EPSILON) GO TO 1000
C
C RETURN FOR MORE INTEGRATION

   GO TO 100
C
C ERROR RETURN
B00   WRITE (4,775)
775   FORMAT ("Jacobian Sign Inversion")
900   WRITE(4,781) TTIME
FEM Code Listing

781  FORMAT("******ERROR RETURN. SIML. TIME= ".F10.4,"******")
C
C****************************************************************************
C
C*********** HERE AT END ***********
1000  CONTINUE  
C
C
C
C CALL OVERFL(I)
C IF (I.NE.2) WRITE (4,782) I
C782  FORMAT("OVERFLOW CODE=",I2," 3=UNDER,1=OVER OR BOTH")
C
C FINAL REPORT ON LAYOUT AND SOLN
IF (ICHOICE(2).GT.0) WRITE (4,785)
785  FORMAT(//"****** FINAL NODE LAYOUT ******")
IF (ICHOICE(2).GT.0)
  2   CALL ERRLIST(X,Y,R,NNODES,XO,YO,YN,IB,TTIME,4,1)
C
C PROVISION TO WRITE GRID FOR PLOTTING
IF (ICHOICE(3).LE.0) GOTO 1500
OPEN(21,MODE="INDUT",FORM="FORMATTED",FILE="xydata")
WRITE (21,786) HBULK,IB,NLOCAL,NBULK,NNODES,INFINITE
786  FORMAT (" -1.0",".F6.2",".4..015.F,",YMIN,YMAX,NFILL,SYMHT"
2  ",".HCOPY"/4(I3,".").L2","...IB,NLOCAL,NBULK,NNODES,INFINITE")
WRITE (21,787) (X(I),Y(I),I=1,NNODES)
787  FORMAT(F10.5,"","F10.5)
1500  CONTINUE  
C
C CALL CPU_TIMER(CPUTIME)
C CPUTIME=CPUTIME-CPUSTART
CALL date_time (clock(),TDSTRING)
WRITE (4,790) TDSTRING,CPUTIME
WRITE (6,790) TDSTRING,CPUTIME
790  FORMAT("/" ALL DONE AT ",A24." Elapsed CPU Seconds=".F12.5)
C
C
CDTIME=3600*ITIME(1)+60*ITIME(2)+ITIME(3)
C CALL TIME(ITIME.IER)
C DTIME=3600*ITIME(1)+60*ITIME(2)+ITIME(3) - DTIME
C WRITE (4,790) ITIME.DTIME
C WRITE(10,790) ITIME.DTIME
C790  FORMAT ("/"ALL DONE AT ",I2,"":".I2","":".I2,
C  2   " Elapsed Time (Seconds)=",F12.0)
END
SUBROUTINE BIQMAXA (MAXA,NNM)

%include implicit
%include b1k2
C
C FIGURES OUT INDEXING VECTOR MAXA FOR COMPACTED STORAGE OF MATRIX A
C """"BIQUADRATIC FINITE ELEMENTS"""
C PACKING SCHEME IS EXPLAINED IN BATHE AND WILSON, NUMERICAL METHODS
C IN FINITE ELEMENT ANALYSIS, 1976 ed, pp 210+,256+
C
C DIMENSION MAXA(NNM)
MAXA(1)=1
C LOOP FOR EVERY NODE
DO 100 N=1,NNODES
C NM REFERS TO THE LOWEST NUMBERED NODE=BASIS FUNCTION THAT THE
C CURRENT NODE=BASIS FUNCTION OVERLAPS WITH
NCOL = 1 + MOD(N-1,IB)
NROW = 1 + (N-1)/IB
IF (NCOL.EQ.1) NCOL=2
IF (NROW.EQ.1) NROW=2
NM= ((NCOL/2)-1)^2 + ((NROW/2)-1)^2*IB + 1
MAXA(N+1)=MAXA(N) + 1 + N -NM
C WRITE (10,600) N,NM,MAXA(N)
C600 FORMAT (I4,I4,1X,8X)
100 CONTINUE
C WRITE (10,600) N,NM,MAXA(N)
RETURN
END
C
C C
C SUBROUTINE CHECKSIGN (YACOBI,ERROR)
%include implicit
SAVE FLAG
LOGICAL ERROR
ERROR=.FALSE.
IF (FLAG.EQ.0.0) GO TO 10
IF (SIGN(FLAG,YACOBI).NE.FLAG) ERROR=.TRUE.
10 FLAG=YACOBI
RETURN
END
C
C C
C SUBROUTINE COLSOL(A,V,MAXA,NN,NNK,NNM,INUT,PMAX,PMIN,DET,NDET)
%include implicit
DIMENSION A(NWK),V(1),MAXA(NNM)
C---------------------------------------------------------------
C COLSOL SOLVES MATRIX PROBLEMS OF THE FORM AX=V WHERE
C MATRIX A IS Banded, Symmetric, AND USUALLY POSITIVE
C DEFINITE. THE PROBLEM IS SOLVED IN CORE
C MEMORY USING COMPACTED STORAGE OF A.
C
REFERENCE BATHE AND WILSON, NUMERICAL METHODS IN FINITE
ELEMENT ANALYSIS, 1976
ALGORITHM AND PROGRAM P.258 (PP.249-258)
Packing Scheme PP.210-212
C
C THE PROGRAM DOES AN LDL FACTORIZATION OF THE MATRIX
AND IMMEDIATELY SOLVES FOR THE RHS VECTOR
C
THIS PROGRAM IS MODIFIED FROM THE BATHE AND WILSON VERSION
WHICH SEPARATES FACTORIZATION AND REDUCTION OF THE RHS
AND WHICH STOPS ON NEGATIVE PIVOTS
PROVISION HAS BEEN MADE FOR REPORTING OF NEGATIVE PIVOTS, THE
MINIMUM AND MAXIMUM PIVOTS, AS WELL AS THE DETERMINANT.
C
-- INPUT VARIABLES --
A(NWK) = THE MATRIX IN COMPACTED FORM
FEM Code Listing

V(NN) = RHS VECTOR
MAXA(NNM) = VECTOR CONTAINING ADDRESSES OF DIAGONAL ELEMENTS
OF MATRIX A
NN = NUMBER OF EQUATIONS
NNK = NUMBER OF ELEMENTS BELOW SKYLINE OF MATRIX
NNM = NN+1
IOUT = UNIT # OF OUTPUT

- - OUTPUT - -
A(NNK) = D AND L FACTORS
V(NN) = SOLUTION VECTOR
PMIN = MINIMUM PIVOT
PMAX = MAXIMUM PIVOT
DET = DETERMINANT DIVIDED BY 10**NDET
NDET = DETERMINANT SCALE FACTOR

PERFORM FACTORIZATION OF STIFFNESS MATRIX

PMIN=A(MAXA(1))
PMAX=A(MAXA(1))
DET=1.0
NDET=0

DO 140 N=1,NN
KN=MAXA(N)
KL=KN+1
KU=MAXA(N+1)-1
KH=KU-KL
IF (KH) 110,90,50

50 K=N=KH
IC=0
KLT=KU
DO 80 J=1,KH
IC=IC+1
KLT=KLT-1
KI=MAXA(K)
ND=MAXA(K+1)-KI-1
IF (ND) 80,80,60
60 KK=MNO(IC,ND)
C=0.0
DO 70 L=1,KK
70 C=C+A(KI+L)*A(KLT+L)
A(KLT)=A(KLT)-C
K=K+1

90 K=N
B=0.0
DD=0.0
DO 100 KK=KL,KU
K=K-1
KI=MAXA(K)
C=A(KK)/A(KI)
B=B+ C*A(KK)
A(KK)=C
DD=DD+C*V(K)
100 CONTINUE

A(KN)=A(KN)-B
V(N)=V(N)-DD
DET=DET*A(KN)

TEMP=ABS(DET)
IF (TEMP.LE.1.0E-10) GOTO 107
DET=DET*1.0E-10
NDET=NDET+10
GOTO 109

107 IF(TEMP.GE.1.0E-10) GOTO 109

225
DET = DET * 1.0E+10
NDET = NDET - 10

C
109 TEMP = ABS(A(KN))
   IF (TEMP .LT. PMIN) PMIN = TEMP
   IF (TEMP .GT. PMAX) PMAX = TEMP
C
120 IF (A(KN)) 125, 120, 140
   FORMAT (1X, "***** SINGULAR MATRIX ***** EQN=", I4, ", ZERO PIVOT")
   STOP
125 CONTINUE
C
WRITE (IOUT, 2001) N, A(KN)
C2001 FORMAT (1X, "EQN=", I4, ", PIVOT=", E20.12)
140 CONTINUE
C
C
BACK SUBSTITUTE
C
DO 200 N = 1, NN
   K = MAXA(N)
200   V(N) = V(N)/A(K)
   IF (NN .EQ. 1) RETURN
   N = NN

DO 230 L = 2, NN
   KL = MAXA(N) - 1
   KU = MAXA(N + 1) - 1
   IF (KU - KL) 230, 210, 210
210   K = N
   DO 220 KK = KL, KU
   K = K - 1
220   V(K) = V(K) - A(KK) * V(N)
   N = N - 1
   RETURN
END

C
C
INCLUDE 'COMP'
SUBROUTINE ERRLIST(X,Y,SOLN,NXY,XO,YO,YN,IB,TTIME,IOUT,SWITCH)

INCLUDE 'implicit'
DIMENSION X(NXY), Y(NXY), SOLN(NXY), XO(IB), YO(IB), YN(IB)

C 3/19/83
C
C THIS ROUTINE COMPUTES THE ERROR IN THE CATHODE POSITION
C THAT HAS ACCUMULATED WITH INTEGRATION IN TIME AS WELL AS:
C * THE SOLUTION ERROR FOR THE ACTUAL CATHODE POSITION
C * THE ERROR IN THE SURFACE FLUX FOR THE ACTUAL CATHODE
C
C YO - CONSTANT VECTOR RELATED TO INITIAL CATHODE POSITION
C YO = EXP(EVAL*YO) (INFINITE DOMAIN)
C YO = (EXP(EVAL*YO) - EXP(EVAL*HBULK))/[EXP(EVAL*YO) - EXP(EVAL*HBULK)] (FINITE DOMAIN)
C
C XO - CONSTANT VECTOR RELATED TO X POSITIONS
C XO = EVAL*ALPHA*SCALAR*CDS(EVAL*XO) (INFINITE DOMAIN)
C XO = 2*EVAL*ALPHA*SCALAR*CDS(EVAL*XO)*EXP(-EVAL*HBULK)
C
C TTIME - PRESENT TIME

WHEN THIS ROUTINE IS CALLED, X() AND Y() CONTAIN THE LATEST
C POSITION OF THE ACTIVE SURFACE NODES, BUT THE DEFORMING ELEMENTS
C HAVE NOT BEEN ADJUSTED TO REFLECT THIS.
C
C POSITION ERROR IS FOR LATEST POSITION OF THE ACTIVE GRID FOUND IN
C X() AND Y() - THIS CORRESPONDS TO TIME TTIME
C
C SOLUTION ERROR AND FLUX ERROR ARE BASED ON THE ACTIVE SURFACE POSITION
C GIVEN IN YN() AND THE OTHER NODES THAT HAVE NOT BEEN REGRIDDED
C TO REFLECT THE CURRENT ACTIVE SURFACE POSITION

226
C BLK3 CONTAINS "INFINITE" - TELLS WHICH PROBLEM DOMAIN
%include blk3
C BLKS - BOUNDARY CONDITION PARAMETERS
%include blk5
C
C IF (ISWITCH.GT.0) WRITE(IOUT,600) TTIME
600 FORMAT(/" POSITION AND PREVIOUS SOLUTION ERR, SIMUL. TIME=" ,
3 "FLUX ERR"/
) SOLRMS=0.0 POSRMS=0.0 QRMS=0.0 DO 100 I=1,IB
C C SOLTRUE = TRUE SOLUTION
TEMP=EVAL*YN(I)
ETEMP=EXP(-TEMP)
DYE=ETEMP
IF (INFINITE) GOTO 10
TEM = EXP(TEMP-EV2HB)
ETEMP=ETEMP - TEM
DYE=ETEMP+TEM
C 10 EXI = EVAL*X(I)
CXI =COS(EXI)
SOLTRUE=SCALAR*CXI*ETEMP
TDDCDX=-SEVAL*SIN(EXI)*ETEMP
TDDCDY=-SEVAL*CXI*DYE
C C FIGURE OUT ACTUAL DCDY,DCDX AND SURFACE ORIENTATION:
CALL SURF(I1,B1,Y1,Y2,SOLN,NNODES,DCDX,DCDY,SNX,SNY)
QERR=SNX*(DCDX-TDDCDX) + SNY*(DCDY-TDDCDY)
QRMS=QRMS+QERR*QERR
C SOLERR=(SOLN(I)-SOLTRUE)/ABS(SCALAR)
SOLRMS=SOLRMS + SOLERR*SOLERR
C C IF (INFINITE) GOTO 50
C C CALCULATION OF TRUE CATHODE POSITION FOR FINITE DOMAIN
TEMP=YO(I)*EXP(TTIME*XO(I))
PTRUE=ALOG((1.0+TEMP)*EXPEVHB/(TEMP-1.0))/EVAL
GOTO 80
C C CALCULATION OF TRUE CATHODE POSITION FOR INFINITE DOMAIN
50 PTRUE=ALOG(TTIME*XO(I)+YO(I))/EVAL
C 80 PERR=Y(I)-PTRUE
POSRMS=POSRMS+PERR*PERR
C C IF (ISWITCH.GT.0) WRITE(IOUT,700) X(I),Y(I),PERR,SOLN(I),SOLERR
700 QERR FORMAT(6(G14.6,".","))
C 100 CONTINUE
C SOLRMS=SQRT(SOLRMS/IB)
POSRMS=SQRT(POSRMS/IB)
QRMS=SQRT(QRMS/IB)
C IF (ISWITCH.GT.0) WRITE (IOUT,750) POSRMS,SOLRMS,QRMS
750 FORMAT(/" RMS POSITION ERROR="G12.5," RMS SOLUTION ERROR=" ,
2 G12.5," RMS FLUX ERROR="G12.5/
) IF (ISWITCH.LE.0) WRITE (IOUT,760) TTIME,POSRMS,SOLRMS,QRMS
760 FORMAT(4(G12.5,".","))
C RETURN
END
INCLUDE 'COMP'
SUBROUTINE GRIDSTART (X,Y,NXY)

%include implicit

***** BICOQUADRATIC FINITE ELEMENT *****
SETS UP THE NODE STRUCTURE FOR A GENERAL ELECTROCHEMICAL CELL
-BOUND ON LEFT AND RIGHT BY PLANES OF SYMMETRY (INSULATORS)
-OPPOSING ELECTRODE IS AT INFINITY. SUPERPOSITION IS USED TO
SET A CONSTANT DEPOSITION OR DISSOLUTION CURRENT
-ACTIVE ELECTRODE INITIAL SHAPE IS GIVEN BY SHAPE() FUNCTION

ACTIVE ELECTRODE
-RANGES FROM X=0 TO X=1
-FUNCTION SHAPE(X(I)) SPECIFIES Y AS A FUNCTION OF X

NODAL LAYOUT
-NODE NUMBERING STARTS WITH X=0 ACTIVE ELECTRODE CORNER AND
INCREASES TO THE RIGHT.
-THE MATRIX PROBLEM COULD BE MADE SLIGHTLY SMALLER BY NUMBERING THE
NODES LIKE:

... N  N+2  N+4
... M+1 M+3 M+5
... M  M+2 M+4...

BUT FOR THIS VERSION THE SIMPLER NUMBERING SCHEME BELOW IS USED:

... N  N+1  N+2...
... M  M+1 M+2...
... K  K+1 K+2...

-NELECT, # OF ELEMENTS ALONG ELECTRODE
-NLOCAL, # OF ELEMENTS ALONG LEFT AND RIGHT LOCAL ELECTRODE REGION
-NBULK, # OF ELEMENTS ALONG LEFT AND RIGHT SYMMETRY PLANES.
-TOP ELEMENT ROW EXTENDS TO INFINITY

DIMENSION X(NXY),Y(NXY)

%include b1k1
%include b1k2

LOGICAL EVEN,EVENL
EVENL=(VSLOCAL.LE.0.0)
EVEN=(VSBUCK.LE.0.0)

SET UP CONSTANTS TO USE IN LOOPS
NROWS=2*NLOCAL+NBULK+1
IF (.NOT.EVEN) FACTB=(HBULK-HLOCAL)/(EXP(VSBUCK)-1.0)
IF (EVEN) FACTB=(HBULK-HLOCAL)/FLOAT(NROWS-NRWLOCAL)

LOOP FOR EVERY COLUMN OF NODES
DO 100 J=1,IB

ACTIVE ELECTRODE NODES - FIRST ROW OF NODES
X(J)=FLOAT(J-1)*WELECT/FLOAT(IB-1)
Y(J)=SHAPE(X(J))
IF (Y(J).GT.HLOCAL) GOTO 1000

FACTL IS THE LOCAL REGION SPACING FACTOR
IF (.NOT.EVENL) FACTL=(HLOCAL-Y(J))/(EXP(VSLOCAL)-1.0)
FACTL=1.0/FLOAT((NRWLOCAL-1)*IB)

LOOP FOR OTHER ROWS OF NODES
DO 100 I=3,NROWS,2

IJK=(I-1)*IB+J
X(IJK)=X(J)
X(IJK-IB)=X(J)

IF (I.GT.NRWLOCAL) GOTO 70
FEM Code Listing

C LOCAL REGION NODES
   IF (EVENL) Y(IJ)=Y(UJ) + FLOAT(IJ-J)*(HLOCAL-Y(UJ))*FACTL
   IF (.NOT.EVENL) Y(IJ)=Y(UJ)+(EXP(VSLOCAL*FLOAT(IJ-UJ)*)
   2 FACTL)-1.0)*FACTL
   Y(IJ-IB)=(Y(IJ)+Y(IJ-2*IB))/2.0
   GOTO 100

C BULK REGION NODES
  70 IF (EVEN) Y(IJ)=HLOCAL +FLOAT(I-NRWLOCAL)*FACTB
   IF (.NOT.EVEN) Y(IJ)=HLOCAL+(EXP(VSBULK*FLOAT(I-NRWLOCAL)/
   2 FLOAT(NROWS-NRWLOCAL))-1.0)*FACTB
   Y(IJ-IB)=(Y(IJ)+Y(IJ-2*IB))/2.0

C 100 CONTINUE

C RETURN

C ERROR RETURN
1000 CONTINUE
   WRITE(4,1100) Y(1),Y(J),Y(IB),HLOCAL
   1100 FORMAT(//Y(1)=','F12.4,' Y(J)=','F12.4,' Y(IB)=','F12.4.
   2 'HLOCAL=',F12.4/
   2 'SHAPE FUNCTION HAS EXCEEDED SPECIFIED LOCAL ELEMENT AREA'/
   3 'STOP IN GRIDSTART' )
   STOP
END

C SUBROUTINE MATFORM (A,NWK,X,Y,NXY,MAXA,NNM,ERROR,FLAG)
%include implicit
%include bk2
%include bk3
%include bk4

DIMENSION A(NWK),X(NXY,Y(NXY),MAXA(NNM)
LOGICAL ERROR,FLAG
MAP(I,J)=MAXA(J)+J-I

C-----------------------------------------------
C COMPUTES UPPER TRIANGULAR PART OF THE STIFFNESS MATRIX.
C IF FLAG IS TRUE, THE CALCULATIONS ARE LIMITED TO THE DEFORMING
C ELEMENTS, OTHERWISE THE CALCULATIONS ARE BASED ON THE FIXED
C ELEMENTS.
C THE BASIS FUNCTIONS HAVE BEEN PRE-EVALUATED AT THE QUADRATURE POINTS
C - SEE SUBROUTINE QUAD -
C BY SHIFTING THE VERTICAL QUADRATURE INDEX, JETA, FINITE OR SEMI-INFINITE
C QUADRATURE IS SELECTED.
C THE ISOPARAMETRIC MAPPING IS DONE IN QUMAP
C-----------------------------------------------

   IF (FLAG) GOTO 5
   IVSTART=NLOCAL+1
   IVSTOP=IVMAX
   GOTO 7
   5
   IVSTART=1
   IVSTOP=NLOCAL
C***** IVER INDEXES ELEMENTS VERTICALLY ****
   7 DO 100 IVER=IVSTART.IVSTOP
   C ** TOP ROW OF ELEMENTS IS HANDLED EXCEPTIONALLY FOR FINITE DOMAIN **
   NIMAX=9
   IF (IVER.EQ.IVMAX.AND(.NOT.INFINITE)) NIMAX=6
   C*** IHOR INDEXES ELEMENTS HORIZONTALLY **
   DO 100 IHOR=1.NELECT
   C** LOOP FOR THE NODES ON THIS ELEMENT **
   N=0
   DO 20 J=1,3

20 Continue
DO 20 I=1,3
   N=N+1
   NODE(N)=(IHOR-1)*2 + (IVER-1)*2*IB + (J-1)*IB + I
   XX(I,J)=X(NODE(N))
   YY(I,J)=Y(NODE(N))
20 CONTINUE

C** LOOP FOR THE QUADRATURE POINTS ON THIS ELEMENT **
   DO 40 J=1,NQ
      JETA=J
   C** PICK DIFFERENT QUADRATURE POINTS FOR INFINITE ELEMENTS *
      IF (IVER.EQ.IVMAX.AND.INFINITE) JETA=J+NQ
      DO 40 I=1,NQ
   C** FIND THE BASIS FUNCTION DERIVATIVES AT THIS QUADRATURE POINT **
      CALL QUAMAP (I,JETA)
      CALL CHECKSIGN (YACOBI,ERROR)
      IF (ERROR) RETURN
      DA=WWT(I,JETA)*ABS(YACOBI)
   C** COMPUTE CONTRIBUTIONS TO STIFFNESS INTEGRALS
      DO 30 NI=1,NIMAX
      DO 30 KJ=NI,NIMAX
         AA=DA*(DPHIDX(NI)*DPHIDX(KJ)+DPHIDY(NI)*DPHIDY(KJ))
         IJ=MAP(NODE(NI),NODE(KJ))
      30 CONTINUE
      A(IJ)=A(IJ)+AA
   40 CONTINUE
100 CONTINUE
RETURN
END

C INCLUDE 'COMP'
C 4-15-83
SUBROUTINE MATNL(A,NWK,X,Y,NXY,MAXA,NNM,R)
%include implicit
DIMENSION A(NWK),X(NXY),Y(NXY),MAXA(NNM),R(NXY)

%include b1k2
%include b1k3
%include b1k4
%include b1k5

MAP(I,J)=MAXA(J) + J - I

C COMPUTES CONTRIBUTIONS TO THE MATRIX PROBLEM ARISING FROM
C INTEGRALS OF THE NORMAL FLUX ON THE ACTIVE SURFACE
C THE BOUNDARY CONDITION CAN BE IMPOSED IN 3 WAYS
C THIS IS CONTROLLED BY IBCOND
C
C** LOOP FOR EVERY ELECTRODE ELEMENT **
   DO 100 IHOR=1,NELECT
   C
      DO 20 I=1,3
         NODE(I)=(IHOR-1)*2 + I
      20 CONTINUE
   C** LOOP FOR EVERY QUADRATURE POINT ON THIS ELEMENT **
   DO 80 IQ=1,NQ
   C** CALCULATE VALUES ALONG THE SURFACE AT THIS POINT **
      DXDXI=0.0
      DYDXI=0.0
      XQ=0.0
      YQ=0.0
      DO 30 NI=1,3
         DXDXI=DXDXI + DPHI(N.IQ)*X(NODE(NI))
         DYDXI=DYDXI + DPHI(N.IQ)*Y(NODE(NI))
         XQ=XQ + PHI(N.IQ)*X(NODE(NI))
         YQ=YQ + PHI(N.IQ)*Y(NODE(NI))
      30 CONTINUE
CONTINUE

C--------- BOUNDARY CONDITION MODULE--------------
C * NOTE DS=SQRT(DXDI*DQDXD + DYDI*DYDXI) CANCELS WITH THIS BC *
C CONTAINS QTERM = INHOMOGENEOUS TERM
C DOTERM = HOMOGENEOUS TERM
C AS FUNCTIONS OF XQ, YQ, DXDI, DYDXI, AND COMMON VARIABLES

TEMP=EVAL*YQ
EX1=EXP(-TEMP)
EVX=EVAL*XQ

C IF (.NOT.INFINITE) GOTO 134
C
GOTO (131, 132, 133), IBCOND

131 QTERM=EX1*(-DYDXI*SIN(EVX)+DXDI*COS(EVX))
DOTERM=0.0
GOTO 138

132 QTERM=EX1*(-DYDXI*SIN(EVX))
DOTERM=DXDI
GOTO 138

133 QTERM=EX1*DXDI*COS(EVX)
DOTERM=-DYDXI*TAN(EVX)
GOTO 138

C FINITE DOMAIN TERMS

134 EX2=EXP((TEMP-EV2HB)
C
GOTO (135, 136, 137), IBCOND

135 QTERM=-DYDXI*SIN(EVX)*((EX1-EX2) + DXDI*COS(EVX) = (EX1+EX2)
DOTERM=0.0
GOTO 138

136 QTERM=-DYDXI*SIN(EVX)*((EX1-EX2)
DOTERM=DXDI*(EX1+EX2)/(EX1-EX2)
GOTO 138

137 QTERM=DXDI*COS(EVX)*((EX1+EX2)
DOTERM=-DYDXI*TAN(EVX)
C
138 QTERM=QTERM*SEAL=WT(IQ)
DOTERM=DOTERM*EVAL=WT(IQ)

C----- MODULE END ----------------------------
C
C** LOOP FOR THE ELECTRODE SURFACE NODES ON THIS ELEMENT **
DO 70 I=1,3
R(NODE(I))=R(NODE(I)) = PHI(I,IQ)*QTERM
C** AND FOR THE DIFFERENT KINDS OF OVERLAP INTEGRAL QUANTITIES **
DO 50 K=1,3
A(IJ)=A(IJ) = PHI(I,IQ)*PH(I,K)*DOTERM
CONTINUE
70 CONTINUE
80 CONTINUE
100 CONTINUE
RETURN
END

C

SUBROUTINE QUAD
%include implicit
C
%include bik3
C
C ASSIGN QUADRATURE PTS AND WEIGHTS
C
NQ=3
WT(1)=5.0/9.0
WT(2)=8.0/9.0
WT(3)=WT(1)
PT(1)=-SORT(3.0/5.0)
PT(2)=0.0
PT(3)=PT(1)

C MODIFY GAUSS-LEGENDRE POINTS AND WEIGHTS SO THEY APPLY
C TO -1 TO +INFINITY
100 DO 250 I=1,NQ
   WIF(I)=2.0*WT(I)/((1.0-PT(I))**2)
   PIF(I)=2.0*PT(I)/(1.0-PT(I))
250 CONTINUE

C CALCULATE COMMONLY USED BASIS FUNCTION - QUADRATURE RELATED QUANTITIES
C
260 DO 500 I=1,NQ
   ISHIFT=I-NQ
   DO 260 J=1,NQ
      WTWT(I,J)=WT(I)*WT(J)
      WTWT(I,J+1)=WT(I)*WIF(J)
260 CONTINUE

C
C PHI(1,1)=PT(I)*(PT(I)-1.0)*.5
C DPHI(1,1)=PT(I)-.5
C PHI(2,1)=1.0-PT(I)*PT(I)
C DPHI(2,1)=-2.0*PT(I)
C PHI(3,1)=PT(I)*(PT(I)+1.0)*.5
C DPHI(3,1)=PT(I)+.5

C
C PHI(1,1,ISHIFT)=PIF(I)*(PIF(I)-1.0)*.5
C DPHI(1,1,ISHIFT)=PIF(I)
C PHI(2,1,ISHIFT)=1.0-PIF(I)*PIF(I)
C DPHI(2,1,ISHIFT)=PIF(I)-.5
C PHI(3,1,ISHIFT)=PIF(I)*(PIF(I)+1.0)*.5
C DPHI(3,1,ISHIFT)=PIF(I)+.5

C
C TEMP=1.0/(PIF(I)-ETAO)
C TEMP=TEMP*TEMP
D0 280 J=1,3
   PDECAY(J,1)=PHI(J,1)
   DPDECAY(J,1)=DPHI(J,1)
280 CONTINUE

C
C PDECAY(1,1,ISHIFT)=(-1.0-ETAO)*TEMP
C PDECAY(2,1,ISHIFT)=-ETAO*TEMP
C PDECAY(3,1,ISHIFT)=(1.0-ETAO)*TEMP

C
D0 290 J=1,3
   DPDECAY(J,1,ISHIFT)=DPHI(J,1,ISHIFT)*PDECAY(J,1,ISHIFT)
   PDECAY(J,1,ISHIFT)=PDECAY(J,1,ISHIFT)*PHI(J,1,ISHIFT)
   DPDECAY(J,1,ISHIFT)=DPDECAY(J,1,ISHIFT)-3.0*PDECAY(J,1,ISHIFT)/
   (PIF(I)-ETAO)
290 CONTINUE
500 CONTINUE
RETURN
END

C
C INCLUDE 'COMP'
C SUBROUTINE QUMAP(XI,JETA)
C %include implicit
C %include b1k3
C %include b1k4
C
C BIQUADRATIC FINITE ELEMENT MAPPING SUBROUTINE
C MAPS FROM (X,Y) TO (XI,JETA)
C GIVEN THE 9 (X,Y) PAIRS THAT DEFINE THE MAPPING OF THE QUADRILATERAL
C TO THE XI,ETA SQUARE, AND A PARTICULAR XI,ETA POINT OF INTEREST,
C THIS ROUTINE CALCULATES THE JACOBIAN OF THE MAPPING, THE BASIS FUNCTION
C DERIVATIVES DPHIDX AND DPHIDY AT THE XI,ETA POINT.

THE MAPPING:

<table>
<thead>
<tr>
<th>ETA</th>
<th>7</th>
<th>8</th>
<th>9</th>
</tr>
</thead>
<tbody>
<tr>
<td>(1,3)</td>
<td>(2,3)</td>
<td>(3,3)</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>5</td>
<td>6</td>
<td></td>
</tr>
<tr>
<td>(1,2)</td>
<td>(2,2)</td>
<td>(3,2)</td>
<td></td>
</tr>
<tr>
<td>-1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(1,1)</td>
<td>(2,1)</td>
<td>(3,1)</td>
<td></td>
</tr>
<tr>
<td>-1</td>
<td>0</td>
<td>1</td>
<td></td>
</tr>
</tbody>
</table>

X = SUM PHI(I)*XX(I)
Y = SUM PHI(I)*YY(I)

(TWOPARAMETRIC MAPPING)

TWO METHODS OF INDEXING
A NODE, (I,J) PAIRS OR
1-9.

( DPHI )   ( DY  -DY )   ( DPHI )
( DX )     ( DETA DXI ) ( DXI )
( ) = 1
( DPHI ) = JACOBIAN ( -DX DX ) ( DPHI )
( DY )     ( DETA DXI ) ( DETA )

JACOBIAN = (DX/DXI)(DY/DETA) - (DX/DETA)(DY/DXI)

BIQUADRATIC BASIS FUNCTIONS AND THEIR DERIVATIVES
OBTAINED AS PRODUCTS OF THE 1 D FUNCTIONS PHI( ), AND DPHI( ).

****** DERIVATIVES BASED ON THE MAPPING DEPEND ON EVERY BASIS FUNCTION *****
****** ZERO THESE DERIVATIVES AND THEN ADD CONTRIBUTIONS FROM EACH BASIS FCN. **

DXDXI=0.0
DXDETA=0.0
DYDXI=0.0
DYDETA=0.0
DO 10 JBF=1,3
   DO 10 IBF=1,3
   DXDXI=DXDXI+ PHI(JBF, JETA)*DPHI(IBF, XI)*XX(IBF, JBF)
   DXDETA=DXDETA+ DPHI(JBF, JETA)*PHI(IBF, XI)*XX(IBF, JBF)
   DYDXI=DYDXI+ PHI(JBF, JETA)*DPHI(IBF, XI)*YY(IBF, JBF)
   DYDETA=DYDETA+ DPHI(JBF, JETA)*PHI(IBF, XI)*YY(IBF, JBF)
10 CONTINUE

****** COMPUTE JACOBIAN OF THE MAPPING ******
YACOBI=DXDXI*DYDETA - DXDETA*DYDXI

C THE MAPPING CALCULATIONS HAVE BEEN COMPLETED -- NOW THE BASIS
C FUNCTIONS ARE USED TO INTERPOLATE THE FIELD VARIABLE
C
C------ THE BASIS FUNCTION DERIVATIVES IN GLOBAL COORDINATES AT THE
C------ POINT (XI, ETA) ------
N=0
DO 15 JBF=1,3
   DO 15 IBF=1,3
      N=N+1
15 CONTINUE
FEM Code Listing

BB=DBECAI(JFB,EJETA)*DBH(I FB,IXI)
CC=DBECAI(JFB,EJETA)*DBH(I FB,IXI)
DPIDX(N)=(DQDTEA-BB+DQDXI+CC/YACOBI
DPID(N)=(-DOQDAA-BB+DOQDA+CC)/YACOBI

C CONTINUE
C RETURN
C END
C
C
SUBROUTINE REGRIDX(X,Y,NXY,ERROR)
%include implicit
DIMENSION X(NXY),Y(NXY)
LOGICAL ERROR, EVENL
C
%include bkl1
%include bkl2
C
C
***** BIQUADRATIC FINITE ELEMENT VERSION *****
C USING THE POSITION OF THE FRESHLY MOVED ELECTRODE SURFACE,
C THIS SUBROUTINE ADJUSTS THE SPACING OF THE NODES IN THE
C LOCAL ELEMENT REGION. IF THE ELECTRODE HAS MOVED BEYOND
C THE LOCAL REGION, THE STRUCTURE OF THIS PROGRAM IS
C INAPPROPRIATE AND ERROR IS SET TRUE
C
C---------- CORNER NODES ARE INDICATED IN COMMON ----------
C IB= RIGHThAND SIDE ELECTRODE CORNER
C
C------------------------------------------------------------------
C C CLEAR TROUBLE FLAG
ERROR=.FALSE.
C------ USE ELECTRODE ROW AS A GUIDE TO REPOSITION LOCAL REGION ----
EVENL=(VSLOCAL.LE.0.0)
DO 200 J=1,IB
IF (Y(J).GE.HLOCAL) GOTO 1000
IF (.NOT.EVENL) FACTL=(HLOCAL-Y(J))/(EXP(VSLOCAL)-1.0)
FACTL=1.0/FACTL((NRWLOCAL-1)*IB)
C
C INNER LOOP TO CATCH MOST NODE ROWS IN LOCAL REGION
DO 150 I=3,NRWLOCAL-2.2
IJ=(I-1)*IB +J
IF (EVENL) Y(IJ)=Y(J)+FLOAT(IJ-J)*(HLOCAL-Y(J))*FACTL
IF (.NOT.EVENL) Y(IJ)=Y(J)+(EXP(VSLOCAL)*FLOAT(IJ-J)*
2 FACTL-1.0)*FACTL
Y(IJ-IB)=Y(IJ)+Y(IJ-2*IB))/2.0
150 CONTINUE
C
C ODD ROW JUST BELOW ROW NRWLOCAL
IJ=(NRWLOCAL-2)*IB +J
Y(IJ)=(Y(IJ+IB)+Y(IJ-IB))/2.0
C
200 CONTINUE
C
RETURN
1000 ERROR=. TRUE.
RETURN
END
C
C
FUNCTION SHAPE(Z)
%include implicit
%include bkl1
C
C THIS FUNCTION DETERMINES THE INITIAL CATHODE SHAPE BASED
C ON THE INTEGER VARIABLE 'TYPE' AND THE FACTOR 'AMP' FOUND
C IN THE COMMON BLOCK BLK1. THE FUNCTION SUPPLIES VALUES

234
OF Y WHEN CALLED WITH VALUES OF X AS ITS ARGUMENT. VALUES
OF X WILL RANGE BETWEEN 0 AND 1.

PI=3.141592653589793D0
GOTO (10,20,30),TYPE

C TYPE=1, COSINE WAVE, SCALED FROM (0,0) TO (1,AMP)
SHAPE=AMP*0.5*(1.0-COS(Z*PI))
RETURN

C TYPE=2, TRIANGLE WAVE
SHAPE=AMP*Z
RETURN

C TYPE=3, COSINE WAVE, OF PERIOD 1
SHAPE=AMP*0.5*(1.0-COS(Z*2.0*PI))
RETURN

END

INCLUDE 'COMP'

3-20-B3
SUBROUTINE SURF(I,IB,YN,X,Y,C,NNODES,DCDX,DCDY,SNX,SNY)
%include implicit
DIMENSION YN(IB),X(2,NNODES),Y(2,NNODES),C(2,NNODES)

COMPUTES THE NORMAL FLUX AT THE BOUNDARY NODE I ASSUMING
BIQUADRATIC FINITE ELEMENTS WITH NODE NUMBERING:

X,Y 2(IB)+1 ... 3(IB)
X,Y IB+1 ... 2(IB) 1 <= I <= IB
X,YN 1 2 3 ... IB

NOTE- FOR THE FIRST ROW OF NODES WE USE Y() VALUES CONTAINED IN YN()

IF (I.EQ.1) GOTO 100
IF (I.EQ.IB) GOTO 200

C NORMAL INTERIOR SURFACE NODE

IRIGHT=I+1
ILEFT=I-1
DCDXI=0.5*(C(IRIGHT)-C(ILEFT))
DYYXI=0.5*(YN(IRIGHT)-YN(ILEFT))
DXDXI=0.5*(X(IRIGHT)-X(ILEFT))

IMID=I+IB
ITOP=I+IB+IB
DCDETA=-1.5*C(I)+2.0*C(IMID)-0.5*C(ITOP)
DYYDETA=-1.5*YN(I)+2.0*YN(IMID)-0.5*YN(ITOP)
DXDETA=-1.5*X(I)+2.0*X(IMID)-0.5*X(ITOP)

C SNX, SNY ARE THE X AND Y COMPONENTS OF THE OUTWARD SURFACE NORMAL
DSDXI=SQR(DCX*DXXI+DYD*YDXI*DYYXI)
SNX=DXDI/DSDXI
SNY=DYDI/DSDXI
YACOB=DXDI*DYDETA-DYD*DXDETA*DYYXI
DCDX=(DYDETA*DCDXI-DYD*DXDETA*DYYXI)/YACOB
DCDY=((-DXDETA*DCDXI+DXDI*DXDETA)/YACOB
DCDN=SNX*DCDX + SNY*DCDY
RETURN

C LEFT CORNER I=1 DERIVATIVES

100 INR=I+1
IFR=I+2
DCDXI=-1.5*C(I)+2.0*C(INR)-0.5*C(IFR)
DXXI=-1.5*YN(I)+2.0*YN(INR)-0.5*YN(IFR)
DXDI=-1.5*X(I)+2.0*X(INR)-0.5*X(IFR)
GOTO 10
C RIGHT CORNER DERIVATIVE

200 INL=I-1
    IFL=I-2
    DCDXI=1.5*C(I)-2.0*C(INL)+0.5*C(IFL)
    DYDXI=1.5*YN(I)-2.0*YN(INL)+0.5*YN(IFL)
    DXDXI=1.5*X(I)-2.0*X(INL)+0.5*X(IFL)
    GOTO 10
END
APPENDIX G

QBEM COMPUTER CODE LISTING

QBEM CALLING HIERARCHY

QBEM
  QUAD
  GRIDSTART
  SHAPE
  REGRID
  MATRIFIED
    ELMT
      PHI
      DPHI
    INTEL
      PHI
      DPHI
      DECAY
  MATVAR
    ELMT
      PHI
      DPHI
    INTEL
      PHI
      DPHI
      DECAY
  MATNL
    ELMT
    NLINE
      PHI
      DPHI
  SGECO (LINPACK)
  SGEFA (LINPACK)
    ISAMAX (LINPACK)
  SSCAL (LINPACK)
  SAXPY (LINPACK)
  SSCAL (LINPACK)
  SASUM (LINPACK)
  SDOT (LINPACK)
  SAXPY (LINPACK)
  SGEGL (LINPACK)
    SAXPY (LINPACK)
    SDOT (LINPACK)
  ERRLIST
  CPU_TIMER
QBEM Listing Notes

MODULE NOTES

QBEM - main program
- reads the input data (file PTDATA)
- sets miscellaneous parameters
- writes report files (files REPORT, ELECTRODE)
- controls the time integration
- stores temporary matrix terms in file AFSAVE
- deforms the active surface

CPU_TIMER - function (PL/1 programming language)
- calls system vclock() function
- returns elapsed virtual cputime (seconds)

DECAY - function
- the decay functions

DPhi - function
- calculates derivatives of the basis functions

ELMT - subroutine
- sets up quantities needed to calculate integrals on the current element

ERRLIST - subroutine
- computes the error in the current active surface position and the error in the last field solution

INTEL - subroutine
- computes integrals on the current element

GRIDSTART - subroutine
- draws initial boundary element mesh

MATFIXED - subroutine
- computes matrix problem contributions for the boundary elements that are spatially fixed

MATNL - subroutine
- computes matrix terms from the boundary condition on the active surface

MATVAR - subroutine
- computes matrix problem contributions for the deforming boundary elements
- computes H_{ij} terms by difference

NLINTEL - subroutine
- computes the active surface boundary condition integrals

PHI - function
- calculates the basis functions

238
REGRID - subroutine
- adjusts the node spacing in the deforming region

QUAD - subroutine
- sets up quadrature related data

SGECO - subroutine (LINPACK)
- factor the main matrix and estimate its condition

SGESL - subroutine (LINPACK)
- solve the linear matrix problem

SHAPE - function
- used to compute the initial active surface shape
- returns surface ordinate given the abscissa

Indicated routines are from the LINPACK library distributed by the Society for Industrial and Applied Mathematics and are documented by Dongarra et al (1980).

SIGNIFICANT COMMON AND INFLT VARIABLES

A(NXY,NXY) - the main matrix, it multiplies the unknowns
AMP - initial amplitude of the active surface
DELTIE - time step size
DXDXI() - derivative, change in global coordinate X w.r. t. $\xi$, at the usual quadrature points
DYDXI() - derivative, $d^2/\partial \xi^2$
ELTYPE - type of current element, see ELMT subroutine
EPSILON - small tolerance for comparison
EVAL - the eigenvalue $\pi$
F(NXY) - the righthand-side vector in the simultaneous linear equation set, equivalent to SOLN(NXY), (becomes overwritten with the matrix solution)
FINITE - true for finite element
FXYN(IB) - rates of surface deformation at each active surface node
HBULK - location of the top surface (finite domain) or top nodes (infinite domain)
HIJ(NXY) - sum of $H_i$, contributions, $i \neq j$
HLOCAL - top of the deforming region
IB - rightmost node on the active surface, also, number of active surface nodes
IC - rightmost node along domain top (finite domain)
ICC - node on RHS where fixed and deforming regions meet
ID - leftmost node along domain top (finite domain)
IDD - node on LHS where fixed and deforming regions meet
IBCND - boundary condition choice (1, 2, or 3)
ICHOOSE(5) - report control, see input data file
INFINITE - true for the infinite domain problem

239
ITCOUNT - count of completed time step iterations
NBULK - number of elements along the fixed part of the symmetry walls
NELECT - number of elements along the active surface
NELMTS - total number of elements
NDELTA - number of elements along the top boundary
NLN - number of logarithmic quadrature points
NLOCAL - number of elements along the deforming part of the symmetry walls
NNODES - total number of nodes
NODE(3) - node indexes on the current element
NQ - number of Gauss-Legendre quadrature points
NWRITE - active surface location is saved every NWRITE time steps
PIF(NQ) - quadrature points for the infinite elements
PLN(NLN) - logarithmic quadrature points
PT(NQ) - the quadrature points
SCALAR - the field solution is multiplied by this scalar (it was kept at 1.0)
SINGULAR - true when current base node is on current element
SOLN(NXY) - the righthand-side vector in the simultaneous linear equation set, becomes the matrix solution
STOPTIME - integration stop time
TTIME - simulated integration time
TYPE - choice of the starting active surface shape
VSBULK - vertical spacing factor, fixed region, 0.0 for uniform spacing
VSLocal - spacing factor, deforming region, 0.0 for uniform vertical node spacing
WIF(NQ) - quadrature weights for infinite elements
WLNL(NLN) - logarithmic quadrature points
WT(NQ) - quadrature weights
X(NXY),Y(NXY) - node locations, global coordinates
XO(IB),YO(IB) - constants related to initial active surface position, needed for error reporting
XX(3),YY(3) - points on current element
YN(IB) - saved position of the active surface
INCLUDED FILES

C implicit - specifies precision
IMPLICIT INTEGER (I-N), DOUBLE PRECISION (A-H,O-Z)

COMMON BLOCKS

COMMON/BLK1/HLOCAL,HBULK,WELECT,VSLOCAL,VSBulk,AMP,TYPE
  INTEGER TYPE

COMMON/BLK2/NLECT,NLOCAL,NBULK,NDELA,NNODES,NELMTS
  2, IB, IC, ID, ICC, IDD

COMMON/BLK3/NQ,NLN,WT(20),PT(20),XQ(20),YQ(20)
  2, DDXI(20),DDYXI(20),WLN(7),PLN(7),WIF(20),PIF(20),INFINITE
  LOGICAL INFINITE

COMMON/BLK4/ELTYPE,SINGULAR,LEFT,CENTER,RIGHT,FINITE
  2, NODE(3), XX(3), YY(3), ETAO
  INTEGER ELTYPE
  LOGICAL SINGULAR, LEFT, CENTER, RIGHT, FINITE

COMMON/BLK5/EVAL,SCALAR,SEVAL,EV2HB,EXPEVHB,IBCOND

INPUT DATA FILE (free format)

   1.0,4.0,0.0,0.0,0.3,1., HLOCAL,HBULK,VSLOCAL,VSBulk,AMplitude,TYPEofshape
   6,3,3,0.,-1000.0.; NLECT, NLOCAL, NBULK,NDELA,ETAO
   0.0,0.0, .; DELTIME,STOPTIME
   1.6,2.,+1.,-3.,-4.,+5.; WRITE,NO,NLN,ICHOICE(5)
   1.0, +1.0,3.; EVAL,SCALAR, IBCOND
IF ICHOICE(1)>0 GET INITIAL REPORT
IF ICHOICE(2)>0 GET FINAL REPORT
IF ICHOICE(3) NOT USED
IF ICHOICE(4)>0 GET CONDITION RECORD AND VERBose ERROR REPORT
IF ICHOICE(5)>0 ONLY CALCULATE DIST. ONCE
QBEM Code Listing

%global card round auto_zero
C
INCLUDE "COMP"
C
4-15-83 rl 1.1 ----------------------------------------
C
Main Program -- - Solution of LaPlace's equation by Boundary Integrals
Equation Methods on a periodic Finite or Infinite domain. A source boundary condition is imposed
in any of these ways leading to a known solution
for the deformation of the active surface.
Quadratic boundary elements and Improved Euler
integration is used.
C
C
%include implicit
C
PARAMETER IXXYY=3025,IXY=55,ISELECT=21
PARAMETER (IXY=224, IXXYY=IXY*XIXY, ISELECT=65)
C
COMMON STATEMENTS ARE GOTTEN FROM INCLUDE FILES
C
%include blk1
%include blk2
%include blk3
%include blk4
%include blk5
C
A LARGER PROBLEM CAN BE TACKLED WHEN ARRAYS ARE IN COMMON
COMMON/BLKG/X(IXY),Y(IXY),A(IXXYY),HIJ(IXY),IPVT(IXY)
DIMENSION F(IXY),SOLN(IXY)
DIMENSION X(IXY),Y(IXY),A(IXXYY),F(IXY),SOLN(IXY),HIJ(IXY),IPVT(IXY)
EQUIVALENCE (F(1),SOLN(1))
DIMENSION YNIELECT,FYNYIELECT,ICHOICE(B),XOIELECT,YOIELECT
LOGICAL REPORT,EUPRED,ERROR
DIMENSION ITIME(3)
C
EXTERNAL date_time_(descriptors)
EXTERNAL clock_(descriptors)
CHARACTER*24 TDSTRING
EXTERNAL CPU_TIMER(descriptors)
REAL CPUSTART, CPUSUM
C
C----------------- INITIALIZE --------------------------
C
C---OPEN FILES-----
OPEN(2,MODE="IN",FORM="FORMATTED",FILE="ptdata")
OPEN(3,MODE="INOUT",FORM="FORMATTED",FILE="electrode")
OPEN(4,MODE="INOUT",FORM="FORMATTED",FILE="report")
OPEN(20,MODE="INOUT",FORM="UNFORMATTED",FILE="afsave")
C
CALL FOPEN (2, "PTDATA")
CALL FOPEN (3, "ELECTRODE")
CALL FOPEN (4, "REPORT")
CALL FOPEN (20, "AFSAVE")
C
C------READ PROBLEM PARAMETERS-----
C
1ST ROW GEOMETRICAL DATA
READ (2,595) HLOCAL,HBULK,VLOCAL,VSBULK,AMP,TYPE
595
C
2ND ROW ELEMENT LAYOUT DISTRIBUTION
READ (2,595) NIELECT,NLOCAL,NBULK,NDELTA,ETAO
C
3RD ROW, INTEGRATION CONTROL
READ (2,595) DELTIME,STOPTIME
C
4TH ROW, QUADRATURE AND MISCELLANEOUS
READ (2,595) NWRITE,NO,NLN,ICHOICE
C
5TH ROW ELECTRODE KINETIC PARAMETERS
READ (2,595) EVAI,SCALAR,IBCCOND
C
C-----------------SET MISCELLANEOUS PARAMETERS-----
C
C TOLERANCE FOR STOP TIME COMPARISON

C
242
EPSILON= 1.0E-06
C SELECT INFINITE OR FINITE DOMAIN BASED ON ELEMENTS ALLOTED FOR TOP
FINITE=.TRUE.
IF (NDELTA.EQ.0) FINITE=.FALSE.
INFINITE=.TRUE.
IF (FINITE) INFINITE=.FALSE.
C NELMTS=NELECT + 2*NLOCAL + 2*NBUCK + NDELTA
C ELEMENTS INTO LOCAL AND BULK REGIONS
C IB,IC,ICC,ID,IDD IDENTIFY CORNER NODE NUMBERS OR DIVISION OF THE
IB=2*NELECT + 1
ICC=IB=2*NLOCAL
IC=ICC+2*NBUCK
ID=IC+1
IF (FINITE) ID=IC + 2*NDELTA
IDD=ID+2*NBUCK
C NXY=2*NELMTS + 1
IF (FINITE) NXY=2*NELMTS
NNODES=NXY
NXXYY=NXY*NXY
C TEST FOR ADEQUATE STORAGE
C ' IF (NXY.LE.IXY.AND.IB.LE.IELECT) GOTO 5
WRITE (4,600) NXXYY,IB,IXY,IXELECT
600 FORMAT (*SOLUTION REQUIRES ",.16," NODES WITH ",.16," CATHODE NODES"
 2 ," BUT ",.15," AND ".15," ARE THE MAXIMUM.")
STOP
C CONTINUE
C C SET ELECTRODE WIDTH -- IT IS THE REFERENCE LENGTH
WSELECT=1.0
C C SET EIGENVALUE FOR ANALYTICAL SOLN
EVAL=FLOAT(INT(EVAL))=3.141592653589793D0/WSELECT
SEVAL=SCALAR_EVAL
EV2HB=EVAL*2.0*NBUCK
EXFEHB=EXP(EVAL*NBUCK)
C C CHECK ORIGIN FOR 1/R^3 DECAY OF INFINITE ELEMENTS, ETAO
C VALUE WAS GIVEN BY INPUT FILE
IF (ETAO.GE.-1.0) ETAO=-8.5525
C C REPORT=.TRUE.
IF (ITIM(1).LE.0) REPORT=.FALSE.
ITIME=0.0
ITCOUNT=0
C C C------SET UP QUADRATURE POINTS AND WEIGHTS ------
CALL QUAD
C C--------------WRITE REPORT INFO TO REPORT FILE --------------
C C --- START CPU TIMING -----
CALL CPU_TIMER(CPUSTART)
CALL date_time_clock_(), TDSTRING
WRITE (4,700) TDSTRING
700 FORMAT(/15(=""),3x,A24,3x,15(""))//
C C CALL TIME(ITIME,IER)
C C CALL FGDAY(MONTH, DAY, YEAR)
C C WRITE (4,700) MONTH, DAY, YEAR, ITIME
C C700 FORMAT(/25(""),15,"/",12,"/",12,15,"=":12,"12,3x,25(""))//
WRITE (4,700)
705 FORMAT("Two Dimensional Moving Boundary Study Using",
2 " Boundary Integrals (1.1.1. 72 bit Reals)")
2 IF (.NOT.FINITE) WRITE(4,707) SCALAR_EVAL,
707 FORMAT("SEMI-INFINITE Domain with Analytical Solution"
2 5x, 10.3."*COS(EV1*X)*EXP(-EV1*Y) (EV1=".F8.4")")
2 IF (FINITE) WRITE(4,708) SCALAR_EVAL
708 FORMAT("FINITE Domain with Analytical Solution")
OBEM Code Listing

2 5X,F10.3,"**COS(EV1*X)*(EXP(-EV1*Y)-EXP(EV1*(Y-2*HBULK)))".
3 " (EV1=".F8.4."))")
WRITE(4,709) IBCOND
709 FORMAT("Boundary Condition imposition type",I2)
WRITE (4,710) HLOCAL,HBULK,WELECT,VSLOCAL,VSBulk,ETAO,AMP,TYPE
710 FORMAT("Domain Geometry")
2 5X,"HLOCAL",4X,"HBULK",5X,"WIDTH",6X,"VSLOCAL" VSBulk,
3 6X,"ETAO",4X,"AMPLITUDE TYPE"/7F10.3,1I0)
WRITE (4,720) NLOCAL,NBULK,NDELTA,NELMTS,NXY
720 FORMAT (/"Distribution of Boundary Elements")
2 6X,"NELMTS NLOCAL HBULK NDELTA TOTAL ELEMENTS TOTAL NODES"
3/518,1I8)
WRITE (4,730) DELTIME,STOPTIME,QD.NLNN
730 FORMAT (/"Integration Parameters")
2 4X,"DELTIME STOPTIME NORMAL QD PTS"
3 ",LOG QD PTS"/2F10.5,2I15)
WRITE (4,735) NWRITE,ICHOICE
735 FORMAT(/"Program Control Parameters")
2 "NWRITE,ICHOICE",I15,I10,4I4/
C--------ALLOW FOR PERIODIC ELECTRODE LOCATION REPORTS
C WRITE(3,740) MONTH,DAY,YEAR,ITIME
C740 FORMAT("Bi EM Program run ",I2,"/",I2,"/",I2,3X,I2,":",I2,":",I2/
C 2 "TIME. POSITION ERR. SOLN ERR")
C WRITE(3,740) TDSTRING
C C
C------ESTABLISH FIXED ELEMENT NODE LOCATIONS -------
C CALL GRIDSTART(X,Y,NXY)
C WRITE (10,742) (1,X(I),Y(I),I=1,NXY)
C742 FORMAT(1I4,5X,2F10.3)
C C------ SET UP XO, YO FOR PERIODIC ERROR REPORTS ------
DO 13 I=1,1B
C TEMP=EXP(EVAL*Y(I))
C IF (INFINITE) GOTO 11
C YO(I)=EXP(EXP*VHBB)/(TEMP-EXP*VHBB)
XO(I)=2.0*EXP+SCALAR*COS(EVAL*X(I))/EXP-VHBB
GOTO 13
C 11 YO(I)=TEMP
XO(I)=EXP+SCALAR*COS(EVAL*X(I))
13 CONTINUE
C C------CALC MATRIX CONTRIBUTIONS FROM FIXED NODES ------
C CALL MATFIXED(X,Y,NXY,A,HIJ)
C C----- SAVE FIXED RESULTS AS A BASE FOR EACH TIME ITERATION ----
C WRITE (20) (A(I),I=1,NXXYY). (HIJ(I),I=1,NXY)
C C FINISH DEFINING MOVING REGION OF ELEMENTS
C CALL REGIDR(X,Y,NXY,ERROR)
C C-------------------------END INITIALIZATION -------------------
C C C------------------- MAIN INTEGRATION LOOP ENTRY POINT ------------
C CONTINUE
C
C C--SET FLAG FOR EULER PREDICTION ----
C EUPRED=.TRUE.
C C-----SAVE ELECTRODE POSITION IN YN ----
DO 109 I=1,1B
109 YN(I)=Y(I)
C C----110 CONTINUE IS ENTRY POINT AFTER EULER PREDICTION----
110 CONTINUE

244
C--- SOLVE FOR FLUX
C REINITIALIZE MATRIX
    IF (ITCOUNT.EQ.0.AND.EUPRED) GOTO 113
    REWIND 20
    READ (20) (A(I),I=1,NXXY),(HIJ(I),I=1,NXY)
C CLEAR RHS VECTOR
    DO 112 I=1,NXY
    F(I)=0.0
112 CONTINUE
C---------ADD CONTRIBUTION FROM MOVING NODES ---------
113 CALL MATVAR(X,Y,NXY,A,HIJ)
C CALL WRITE(6,760) (HIJ(I),I=1,NXY)
C760 FORMAT(G12.5)
C ADD FLUX RELATED INTEGRALS OVER ACTIVE SURFACE
C CALL MATNL(X,Y,NXY,A,F)
C
C FACTOR MATRIX AND ESTIMATE CONDITION NUMBER
C CALL SGECO(A,NXY,NXY,IPVT,RCOND,HIJ)
C (NOTE - HIJ IS USED AS A WORK VECTOR)
C IF (RCOND.NE.0) RCOND=1.0/RCOND
C NOW SOLVE EQUATION
C CALL SGESL(A,NXY,NXY,IPVT,F,0)
C C
C IF(ICHoice(4).GT.0)WRITE (4,766) TTIME,RCOND
C766 FORMAT("T="F12.5," Matrix Condition=*",G12.5)
C C
C------- ON FIRST PASS REPORT THE SOLUTION AND NODAL LAYOUT------
C IF (.NOT.REPORT) GOTO 127
C CALL ERRLIST(X,Y,F,NNODES,XO,YO,YN,IB,TTIME,4,1)
C CALL ERRLIST(X,Y,F,NNODES,XO,YO,YN,IB,TTIME,3,ICHoice(4))
C C DEBUG------EXIT
C IF(ICHoice(5).GT.0)GOTO 1000
C REPORT=.FALSE.
C C--------------------- EULER PREDICTION ---------------------
127 IF (.NOT.EUPRED) GO TO 150
    DO 135 I=1,IB
    FXYN(I)=SOLN(I)
C PREDICT NEW ELECTRODE POSITION:
    Y(I)=Y(I) + DELTIME*FXYN(I)
135 CONTINUE
C C READJUST TRENCH AREA ELEMENTS
C CALL REGRID(X,Y,NXY,ERROR)
C IF (ERROR) GOTO 900
C------ NOW FIND FLUX WITH THIS POSITION -----
C EUPRED=.FALSE.
C GO TO 110
C C----------- HERE FOR TRAPEZOID CORRECTOR ----------------------
150 CONTINUE
C C CALCULATE NEW ELECTRODE MOVEMENT RATE AND AVERAGE IT WITH EULER
C PREDICTED RATE
    DO 165 I=1,IB
    FXYN(I)=FXYN(I) + SOLN(I)
C MOVE THE ELECTRODE
    SAVE=Y(I)
    Y(I)=Y(I) + DELTIME*FXYN(I)/2.0
    YN(I)=SAVE
165 CONTINUE
C C READJUST ELEMENTS
C
C
CALL REGRID(X,Y,NXY,ERROR)
IF(ERROR) GOTO 900

C---------- THIS COMPLETES AN INTEGRATION CYCLE ----------
ITCOUNT=ITCOUNT +1
TIME=TIME + DELTIME

C WRITE ELECTRODE POSITION EVERY NWRITE CYCLES
IF (MOD(ITCOUNT,NWRITE).NE.0) GO TO 200
CALL ERRLIST(X,Y,F,NNODES,XO,YO,YN,IB,TTIME,3,ICHOICE(4))

C------- CHECK TIME AGAINST STOP TIME
200 IF (TTIME.GE.STOPTIME-EPSILON) GO TO 1000
C RETURN FOR MORE INTEGRATION
C GO TO 100
C ERROR RETURN
900 WRITE(4,781) TTIME
781 FORMAT("ERROR RETURN, SIML. TIME=",F10.4)
C-------------------------------
C************* HERE AT END *************
C 1000 CONTINUE

C CALL OVERFL(I)
C IF (I.NE.2) WRITE (4,782) I
782 FORMAT("OVERFLOW CODE=",I2," 3=UNDER,1=OVER OR BOTH")
C
C FINAL REPORT ON LAYOUT AND SOLN
IF (ICHOICE(2).GT.0) WRITE (4,785)
785 FORMAT (/"******* FINAL NODE LAYOUT ******/
IF (ICHOICE(2).GT.0)
   2 CALL ERRLIST(X,Y,F,NNODES,XO,YO,YN,IB,TTIME,4,1)

C CALL date_time(clock(),TDSTRING)
CALL CPU_TIMER(CPUTIME)
CPUTIME=CPUTIME-CPUSTART
WRITE (4,790) TDSTRING,CPUTIME
WRITE (6,790) TDSTRING,CPUTIME
790 FORMAT(/" ALL DONE AT ",A24," Elapsed CPU Seconds=",F12.4)
C DTIME=3600*ITIME(1)+60*ITIME(2)+ITIME(3)
C CALL TIME(ITIME,IER)
C DTIME=3600*ITIME(1)+60*ITIME(2)+ITIME(3) - DTIME
C WRITE (4,790) ITIME, DTIME
C WRITE (10,790) ITIME,DTIME
C790 FORMAT (/"ALL DONE AT ",&I2,:,&I2,:,&I2,
C 2 " Elapsed time="F12.5," Seconds")
END
C INCLUDE "COMP"
FUNCTION DECAY(K,Z)
%include implicit
C COMMON BLOCK TELLS ELEMENT TYPE AND BASE POINT FOR DECAY
%include b1k4
N=K
ETA=Z
IF (ELTYPE.EQ.4) GOTO 4
C HERE FOR ELTYPE 5, MUST INVERT LOCAL COORDINATE DIRECTION
ETA=-Z
GOTO (1,4,3),K
1 N=3
GOTO 4
3 N=1
4 GOTO (11,12,13),N
11 DECAY=(-1.0-ETA0)/(ETA-ETA0)
GOTO 20
12 DECAY=ETA0/(ETA-ETA0)
GOTO 20
13 DECAY=(1.0-ETA0)/(ETA-ETA0)
20 DECAY=DECAY*DECAY*DECAY
RETURN
END
C
C
C INCLUDE 'COMP'
FUNCTION DPHI(N,Z)
%include implicit
GO TO (1,2,3),N
1 DPHI=Z-0.5
RETURN
2 DPHI=-2.0*Z
RETURN
3 DPHI=Z+0.5
RETURN
END
C
C
C INCLUDE 'COMP'
SUBROUTINE ELMT(N,X,Y,NXY)
%include implicit
DIMENSION X(NXY),Y(NXY)
C
%include b1k2
%include b1k3
%include b1k4
C---------------------------------------------------------------
C DISCONTINUITY IN NODE NUMBERING BECAUSE OF BREAK FOR INFINITY
IDISC=0
IF(N.GT.NELECT +NLOCAL +NBULK.AND..NOT.Finite) IDISC=1
DO 10 M=1,3
NODE(M)=2*N + M + IDISC - 2
IF(N.EQ.NELMTS.AND.M.EQ.3) NODE(3)=1
XX(M)=X(NODE(M))
YY(M)=Y(NODE(M))
10 CONTINUE
C-------- SET UP VARIABLES TO DESCRIBE ELEMENT --------
C -- ELTYPE ---- 1=ELECTRODE, 2=INSUL, 3=TOP, 4=INFINITE RHS, 5=INFINITE LHS
ELTYPE=2
IF (ID.GT.NODE(2).AND.NODE(2).GT.IC) ELTYPE=3
IF (N.LE.NELECT) ELTYPE=1
IF (FINITE) GOTO 30
IF(NODE(3).EQ.IC) ELTYPE=4
IF(NODE(1).EQ.ID) ELTYPE=5
30 CONTINUE
C---- CALCULATE MAPPING INFO FOR THIS ELEMENT ----
247
THESE RESULTS ARE FOR THE NORMAL QUADRATURE POINTS THAT WILL BE
USED MOST OF THE TIME

DD 100 IQ=1,NQ
XI=PT(IQ)
IF (ELTYPE.EQ.4) XI=PIF(IQ)
IF (ELTYPE.EQ.5) XI=-PIF(IQ)
XQ(IQ)=0.0
YQ(IQ)=0.0
DQXI(IQ)=0.0
DYQXI(IQ)=0.0

DD 80 K=1,3
XQ(IQ)=XQ(IQ)+PHI(K,XI)*XX(K)
YQ(IQ)=YQ(IQ)+PHI(K,XI)*YY(K)
DQXI(IQ)=DQXI(IQ)+DPHI(K,XI)*XX(K)
DYQXI(IQ)=DYQXI(IQ)+DPHI(K,XI)*YY(K)
CONTINUE

80 CONTINUE
END

INCLUDE 'CMPL'
SUBROUTINE ERRLLST(X,Y,SOLN,NXY,XO,YO,YN,IB,TTIME,IOUT,ISWITCH)
DIMENSION X(NXY),Y(NXY),SOLN(NXY),XO(IB),YO(IB),YN(IB)

1/4/83
THIS ROUTINE COMPUTES THE ERROR IN THE CATHODE POSITION
THAT HAS ACCUMULATED WITH INTEGRATION IN TIME AS WELL AS
THE SOLUTION ERROR FOR THE ACTUAL CATHODE POSITION

YO - CONSTANT VECTOR RELATED TO INITIAL CATHODE POSITION
YO=EXP(EVAL*YO) (INFINITE DOMAIN)
YO={EXP(EVAL*YO)-EXP(EVAL*HBULK)}/(EXP(EVAL*YO)-
EXP(EVAL*HBULK)) (FINITE DOMAIN)

XO - CONSTANT VECTOR RELATED TO X POSITIONS
XO=EVAL*ALPHA*SCALAR*COS(EVAL*XO) (INFINITE DOMAIN)
XO=2*EVAL*ALPHA*SCALAR*COS(EVAL*XO)*EXP(-EVAL*HBULK)

TTIME - PRESENT TIME

POSITION ERROR IS FOR POSITION OF GRID LOCATION Y(I) AT SIMULATION
TIME TTIME.

SOLUTION ERROR IS FOR GRID POSITIONS IN YN() WHEN SOLN WAS PREVIOUSLY
CALCULATED.

BLK3 CONTAINS "INFINITE" - TELLS WHICH PROBLEM DOMAIN

INCLUDE blk3
BLK5 - BOUNDARY CONDITION PARAMETERS
INCLUDE blk5

IF (ISWITCH.GT.0) WRITE(IOUT,600) TTIME
600 FORMAT(13" POSITION AND PREVIOUS SOLUTION ERR. SIMUL. TIME="
,2 F10.5/"TX","Y",14X,"POS ERR",8X,"SOLN",9X,"SOLN ERR")/
SOLRMS=0.0
POSRMS=0.0
DD 100 I=1,IB

C
SOLTRUE = TRUE SOLUTION
TEMP=EVAL*YN(I)
ETEMP=EXP(-TEMP)
IF (.NOT.INFINITE) ETEMP=ETEMP-EXP(TEMP-EV2HB)
SOLTRUE=SCALAR*COS(EVAL*X(I))*ETEMP
SOLERR=(SOLN(I)-SOLTRUE)/ABS(SCALAR)
SOLRMS=SOLRMS + SOLERR

C
IF (INFINITE) GOTO 50
C CALCULATION OF TRUE CATHODE POSITION FOR FINITE DOMAIN
TEMP=Yo(I)*EXP(TTIME*Yo(I))
PTRUE=ALOG((1.0-TEMP)*EXP-EVHB/(TEMP-1.0))/EVAL
GOTO 80

C CALCULATION OF TRUE CATHODE POSITION FOR INFINITE DOMAIN
50 PTRUE=ALOG(TTIME*Yo(I)+Yo(I))/EVAL
80 PERR=Y(I)-PTRUE
POSRMS=POSRMS+PERR
C
IF (ISWITCH.GT.0) WRITE(IOUT,700) X(I),Y(I),PERR,SOILN(I),SOILERR
700 FORMAT(5(G14.6,".*"))
C
100 CONTINUE
C
SOLRMS=SQR(SOLRMS/IB)
POSRMS=SQR(POSRMS/IB)
IF (ISWITCH.GT.0) WRITE(IOUT,750) POSRMS,SOLRMS
750 FORMAT(10RMS POSITION ERROR=",G12.5,"/RMS SOLUTION ERROR=",G12.5/*
150 IF (ISWITCH.LE.0) WRITE(IOUT,760) TTIME,POSRMS,SOLRMS
760 FORMAT(3(G12.5,".*"))
RETURN
END
C
C INCLUDE 'COMP'
SUBROUTINE GRIDSTART (X,Y,NXY)
%include implicit
C
2/5/83 -----------------------------------------------
C SETS UP THE NODE STRUCTURE FOR A GENERAL ELECTROCHEMICAL CELL
- BOUNDED ON LEFT AND RIGHT BY PLANES OF SYMMETRY (INSULATORS)
- OPPOSING ELECTRODE CAN BE AT INFINITY OR CELL CAN BE CLOSED
- A HORIZONTAL SURFACE
- ACTIVE ELECTRODE INITIAL SHAPE IS GIVEN BY SHAPE() FUNCTION
C
ACTIVE ELECTRODE
- RANGE FROM X=O TO X=1
- FUNCTION SHAPE(X(I)) SPECIFIES Y AS A FUNCTION OF X
C
NODAL LAYOUT
- QUADRATIC BOUNDARY ELEMENTS
- NODE NUMBERING STARTS WITH X=O ACTIVE ELECTRODE CORNER AND CONTINUES
COUNTERCLOCKWISE AROUND THE CELL
- NLELECT, # OF ELEMENTS ON ELECTRODE
- NLOCAL, # OF ELEMENTS ON LEFT AND RIGHT PLANES BETWEEN ELECTRODE
AND TOP OF LOCAL REGION, HLOCAL
- NBULK, # OF ELEMENTS ON LEFT AND RIGHT SYMMETRY PLANES, LAST ONE
ON EACH SIDE CAN EXTEND TO INFINITY
- NDELTA, # OF ELEMENTS ALONG CELL TOP, IF 0, FINITE SHOULD BE FALSE
C
DIMENSION X(NXY),Y(NXY)
C
%include b1k1
%include b1k2
%include b1k4
C
LOGICAL EVEN,EVENL
EVENL=(VLOCAL.LE.0.0)
EVEN=(VSLOCAL.LE.0.0)
C
--------LOCATE ACTIVE ELECTRODE NODES---------
DO 20 I=1,IB
X(I)=FLOAT(I-1)*WELECT/FLOAT(IB-1)
Y(I)=SHAPE(X(I))
20 CONTINUE
C
---------RIGHT HAND SIDE, LOCAL ELEMENTS ABOVE ELECTRODE----
IF (Y(IB).GE.HLOCAL) GOTO 1000

249
IF(.NOT.EVENL) FACTR=(HLOCAL-Y(IB))/(EXP(VSLOCAL)-1.0)
DO 30 I=IB+2,ICC,2
X(I-1)=WELECT
X(I)=WELECT
IF (EVENL) Y(I)=Y(IB)+ FLOAT(I-IB)*(HLOCAL-Y(IB))/
2 FLOAT(ICC-IB)
IF (.NOT.EVENL) Y(I)=Y(IB) + (EXP(VSLOCAL*FLOAT(I-IB))/
2 FLOAT(ICC-IB))/-1.0)*FACTR
Y(I-1)=(Y(I)+Y(I-2))/2.0
30 CONTINUE
C------------------RIGHT HAND SIDE BULK -ABOVE LOCAL ----
IF (.NOT.EVEN) FACT=(HBU -HLOCAL)/(EXP(VSBULK)-1.0)
DO 50 I=ICC+2,IC,2
X(I-1)=WELECT
X(I)=WELECT
IF (EVEN) Y(I)=HLOCAL + FLOAT (I-ICC)*(HBU -HLOCAL)/
2 FLOAT(ICC-ICC)
IF (.NOT.EVENL) Y(I)=HLOCAL+(EXP(VSBULK*FLOAT(I-ICC))/
2 FLOAT(ICC-ICC))/-1.0)*FACTR
Y(I-1)=(Y(I)+Y(I-2))/2.0
50 CONTINUE
C------- IF FINITE - TOP OF CELL -----------------------
IF (.NOT.FINITE) GOTO 62
DO 80 I=IC+1,ID-1
X(I)=WELECT*(1.0-FLOAT(I-IC)/FLOAT(ID-IC))
Y(I)=HBU
60 CONTINUE
C-----------------LEFT-HAND SIDE NODES-BULK-------------
62 X(ID)=0.0
Y(ID)=HBU
DO 70 I=ID+2,IDD,2
X(I-1)=0.0
Y(I)=0.0
IF (EVENL) Y(I)=HBU - FLOAT(I-ID)*(HBU -HLOCAL)/
2 FLOAT(IDD-ID)
IF (.NOT.EVENL) Y(I)=HLOCAL+(EXP(VSBULK*1.0-FLOAT(I-ID))/
2 FLOAT(IDD-ID))/-1.0)*FACTR
Y(I-1)=(Y(I)+Y(I-2))/2.0
70 CONTINUE
C-----------------LEFT-HAND SIDE LOCAL ELEMENTS ABOVE ELECTRODE--
IF (Y(1).GE.HLOCAL) GOTO 1000
IF (IDD.EQ.NNODES-1) GOTO 85
IF (.NOT.EVENL) FACTL=(HLOCAL-Y(1))/(EXP(VSLOCAL)-1.0)
DO 80 I=IDD+2,NODES-1,2
X(I-1)=0.0
X(I)=0.0
IF (EVENL) Y(I)=HLOCAL-FLOAT(I-IDD)*(HLOCAL-Y(1))/
2 FLOAT(NNODES+1-IDD)
IF (.NOT.EVENL) Y(I)=Y(1)+(EXP(VSLOCAL*1.0-FLOAT(I-IDD))/
2 FLOAT(NNODES+1-IDD))/-1.0)*FACTL
Y(I-1)=(Y(I)+Y(I-2))/2.0
80 CONTINUE
85 X(NNODES)=0.0
Y(NNODES)=(Y(NNODES-1)+Y(1))/2.0
C C C C
C C C C
C C C C
C C C C
C C ERROR RETURN
1000 CONTINUE
WRITE(4,1100) Y(1),Y(IB),HLOCAL
1100 FORMAT('"Y(1)=",F12.4," Y(IB)=",F12.4," HLOCAL="',F12.4/
2 "SHAPE FUNCTION HAS EXCEEDED SPECIFIED LOCAL ELEMENT AREA"/
3 "STOP IN GRIDSTART")
STOP
END
C C C C
INCLUDE 'COMP'
SUBROUTINE INTEL(I,XXX,YYY,A,NXY,HIJ)
%include implicit
DIMENSION A(NXY,NXY),HHH(3),HIJ(NXY),GIJ(3)
LOGICAL PASS2
C
%include blk2
%include blk3
%include blk4
%include blk5
C
C THIS ROUTINE ACCOMPLISHES NUMERICAL INTEGRATION ON A GIVEN ELEMENT
C FOR A GIVEN BASE NODE POINT. IT HANDLES HIJ INTEGRALS FOR ALL
C ELEMENTS.
C SPECIALIZED OR POSSIBLY NONLINEAR ELECTRODE GIJ INTEGRALS
C ARE DONE IN NLINTELE
C
C FOR DEFORMING ELECTRODE ELEMENTS, SKIP TO NUMERICAL HIJ
C (COLLINEARITY TEST IS NOT APPLICABLE)
C IF (ELTYPE.EQ.1) GOTO 205
C
C TEST FOR COLLINEAR NODE AND ELEMENT
C IF HIU=0, GOTO GIJ OR EXIT IF INSULATOR
C
C IF(SINGULAR) GOTO 290
C IF(NODE(1)-ID) 2,1,1
C TEST FOR COLLINEAR LHS INSULATOR
1 IF( I.GE.ID.OR.I.EQ.1) RETURN
GOTO 9
C IF (NODE(1)-IC) 4,3,3
C TEST FOR COLLINEAR TOP SURFACE, EXIT FOR GIJ
3 IF( I.LE.ID.AND.I.GE.IC) GOTO 300
GOTO 9
C SINCE ELTYPE <> 1, WE KNOW ELEMENT IS RHS INSULATOR
4 IF(I.LE.IC.AND.I.GE.IB) RETURN
C CONTINUE
C
C NUMERICAL HIJ
C ZERO INTEGRAL TERM
205 DO 210 K=1,3
210 HHH(K)=0.0
C LOOP FOR EVERY QUADRATURE PT
GOTO (215,215,215,235,235),ELTYPE
215 DO 230 IQ=1,NQ
XI=PT(IQ)
DY=YQ(IQ)-YYY
DX=XQ(IQ)-XXX
R2=DY*DY + DX*DX
HH=WT(IQ)*(-DY*DXDI(IQ) + DX*DYDI(IQ))/R2
DO 225 K=1,3
225 HHH(K)=HHH(K) + HH*PHI(K,XI)
230 CONTINUE
GOTO 245
C
235 DO 240 IQ=1,NQ
XI=PFI(IQ)
IF (ELTYPE.EQ.5) XI=-XI
DY=YQ(IQ)-YYY
DX=XQ(IQ)-XXX
R2=DY*DY + DX*DX
HH=WIF(IQ)*(-DY*DXDI(IQ) + DX*DYDI(IQ))/R2
DO 240 K=1,3
240 HHH(K)=HHH(K)+HH*PHI(K,XI)*DECAY(K,XI)
C
C ADD INTEGRAL TERMS TO MATRIX A
QBEM Code Listing

C DIFFERENT RULE IS USED FOR FINITE DOMAIN AND SPECIAL CORNERS
C

245      IF (FINITE) GOTO 265
C
DO 260 K=1,3
IF (I.EQ.NODE(K)) GOTO 260
HIU(I)=HIU(I)+HHH(K)
A(I,NODE(K))=A(I,NODE(K))+HH(K)
260      CONTINUE
C
RETURN
C
C FINITE DOMAIN - POTENTIAL IS SPECIFIED AT TOP, DON'T HAVE MATRIX
C CONTRIBUTION FROM TOP SURFACE
C
265      DO 270 K=1,3
IF (I.EQ.NODE(K)) GOTO 270
HIU(I)=HIU(I)+HHH(K)
2 IF (ELTYPE,NE.3.AND.NODE(K).NE.IC.AND.NODE(K).NE.ID)
   2        A(I,NODE(K))=A(I,NODE(K))+HH(K)
270      CONTINUE
C
C 290      IF (ELTYPE,NE.3) RETURN
C
C ****** FINITE DOMAIN GIJ TYPE INTEGRALS FOR SURFACE OF UNKNOWN FLUX ******
C
C
C INITIALIZE TERMS
C
300      DO 305 K=1,3
305      GIU(K)=0.0
C
C ****** NORMAL QUADRATURE FOR USUAL CASES OR REMAPPED ******
C
C ****** INTEGRALS FOR SINGULAR CASES ******
C
DO 320 IQ=1,NQ
XI=PT(IQ)
R2=(YQ(IQ)-YQ)**2+(XQ(IQ)-XQ)**2
IF (LEFT) R2=4.0*R2/((XI+1.0)**2)
IF (RIGHT) R2=4.0*R2/((1.0-XI)**2)
IF (CENTER) R2=R2/(XI)**2
GG=W(T(IQ))*SQRT(DXDI(IQ)**2+DYXI(IQ)**2)*ALOG(R2)*0.5
310      GIU(K)=GIU(K)+GG*PHI(K,XI)
320      CONTINUE
C
C SINGULAR CASES REQUIRE FURTHER CALCULATION
IF (SINGULAR) GOTO 330
C
C ADD RESULTS INTO RESIDUAL AND JACOBIAN - ALL DONE
GOTO 1000
C
C ****** FROM HERE ON CODE FOR SINGULAR INTEGRALS, GIJ CONTRIBUTIONS ******
C
C
330      CONTINUE
C
PASS2=.FALSE.
400      DO 500 IQ=1,NL
XI=PLN(IQ)
IF (PASS2) XI=XI
IF (LEFT) XI=2.0*PLN(IQ)-1.0
IF (RIGHT) XI=1.0-2.0*PLN(IQ)
DXL=0.0
DYL=0.0
DO 410 K=1,3
DXL=DXL+DPHI(K,XI)*XX(K)
DYL=DYL+DPHI(K,XI)*YY(K)
410      CONTINUE
C------THE LOG R TERM IS BURIED IN THE QUADRATURE PTS AND WEIGHTS--
GG=WLN(IQ)*SQRT(DXL*DXL+DYL*DYL)

252
QBEM Code Listing

C LEFT AND RIGHT SINGULAR CASES REQUIRE FACTOR OF 2 FROM MAPPING
IF (.NOT.CENTER) GG=GG*GG
   DO 430 K=1,3
   GIJ(K)=GIJ(K) + GG*PHI(K,XI)
   CONTINUE
IF (.NOT.CENTER) GOTO 1000
IF (PASS2) GOTO 1000
PASS2=.TRUE.
GOTO 400

C CENTER NODE SINGULAR CASE IS BROKEN INTO 2 INTEGRALS, LEFT AND RIGHT
C OF CENTER - THAT'S WHY A SECOND PASS IS DONE ABOVE.
C
C - EXIT - ADD RESULTS
C
1000   DO 1020 K=1,3
1020   A(I,NODE(K))=A(I,NODE(K)) - GIJ(K)
   RETURN
   END

C
C
C
C
C
C
C
C INCLUDE 'COMP'
SUBROUTINE MATFIXED (X,Y,NXY,A,HIJ)
%include implicit
DIMENSION X(NXY),Y(NXY),A(NXY,NXY),HIJ(NXY)
C
%include blk2
%include blk4
C
C-----------------------------------------------------------------------------------
C SETS UP MATRIX CONTRIBUTIONS FOR FIXED ELEMENTS AND FIXED BASE NODES.
C-----------------------------------------------------------------------------------
C-----OUTER LOOP FOR EVERY FIXED ELEMENT-------
   NSTART=NELECT + NLOCAL + 1
   NSTOP=NELMTS - NLOCAL
   DO 5000 N=NSTART,NSTOP
C----- FOR ELEMENT N, FIGURE OUT THE MAPPING INFO AT THE NORMAL QUADRATURE POINTS
C
   CALL ELMT(N,X,Y,NXY)
C----- NOW LOOP FOR EVERY STATIONARY NODE-------
   DO 4000 I=ICC.IDD
      LEFT= I.EQ.NODE(1)
      CENTER= I.EQ.NODE(2)
      RIGHT= I.EQ.NODE(3)
      SINGULAR= LEFT.OR.CENTER.OR.RIGHT
      XXX=X(I)
      YYY=Y(I)
      CALL INTEL (I,XXX,YYY,A,NXY,HIJ)
   CONTINUE
4000   CONTINUE
5000   CONTINUE
C
   RETURN
   END

C
C
C
C
C INCLUDE 'COMP'
SUBROUTINE MATNL (X,Y,NXY,A,F)
%include implicit
DIMENSION X(NXY),Y(NXY),A(NXY,NXY),F(NXY)
C
%include blk2
%include blk4
C
C 253
C-----------------------------------------------------------------------------
C SUPervises CALCulation of NONlinear or specialZed CONTRIBUTions
C COMing FROM Electrode ELEMENTS and ALL BASE Nodes
C-----------------------------------------------------------------------------
C --- OUTER LOOP FOR Electrode ELEMENTS ---
C DO 5000 N=1,NELECT
C --- CALc MAPPING DATA FOR NORMAL QUADRATURE Pts
C CALL ELMT(N,X,Y,NXY)
C
C --- LOOP FOR EVERY BASE NODE ---
C DO 4000 I=1,NNODES
C LEFT= I.EQ.NODE(1)
C CENTER= I.EQ.NODE(2)
C RIGHT= I.EQ.NODE(3)
C SINGULAR= LEFT.OR.CENTER.OR.RIGHT
C XXX=X(I)
C YYY=Y(I)
C CALL NLINTEL (I,XXX,YYY,A,F,NXY)
C CONTINUE
C 4000 CONTINUE
C 5000 RETURN
C END
C
C INCLUDE 'COMP'
C SUBROUTINE MATVAR (X,Y,NXY,A,HIJ)
C% include implicit
C DIMENSION X(NXY),Y(NXY),A(NXY,NXY),HIJ(NXY)
C% include blk2
C% include blk4
C
C-----------------------------------------------------------------------------
C SETS UP MATRIX CONTRIBUTIONS for (All BOUNDARY elements and Moving
C BASE Nodes) and (MOVING elements with ALL BASE Nodes)
C-----------------------------------------------------------------------------
C LOGICAL ELMOVE,PASS2
C---- OUTER LOOP FOR EVERY ELEMENT------
C NFX=NELECT + NLOCAL
C NFX=NELMTS - NLOCAL
C DO 5000 N=1,NELMTS
C --- CALc MAPPING DATA FOR NORMAL QUADRATURE Pts
C CALL ELMT(N,X,Y,NXY)
C
C ELMOVE INDICATES A MOVING LOCAL REGION ELEMENT
C ELMOVE=(N.LE.NFX.OR.N.GT.NFXE)
C------- FOR MOVING ELEMENTS WE WILL LOOP ONCE FOR ALL
C BASE NODES. FOR OTHER ELEMENTS WE WILL LOOP TWICE, ONCE FOR
C LOW NUMBERED BASE NODES (ELECTRODE AND RIGHT LOCAL WALL), AND A
C SECOND TIME for HIGH NUMBERED NODES (LEFT LOCAL WALL).
C-------
C PASS2=.FALSE.
C ISTART=1
C ISTOP=NNODES
C IF (.NOT.ELMOVE) ISTOP=ICC-1
C 3000 DO 4000 I=ISTART,ISTOP
C LEFT= I.EQ.NODE(1)
C CENTER= I.EQ.NODE(2)
C RIGHT= I.EQ.NODE(3)
C SINGULAR= LEFT.OR.CENTER.OR.RIGHT
C XXX=X(I)
C YYY=Y(I)
C CALL INTEL (I,XXX,YYY,A,NXY,HIJ)
C CONTINUE
C 4000 IF (ELMOVE) GO to 5000
C IF (PASS2) GO to 5000
C 254
ISTART=IDD+1
ISTOP= NNODES
PASS2= TRUE.
GO TO 3000
CONTINUE
C C NOW ALL HIJ, J,NE,I, HAVE BEEN FIGURED --- FIGURE
C C OUT HIJ(I,I) BY SUMMATION
C IF(FINITE) GOTO 7000
DO 6000 I=1,2*NELMTS +1
6000 A(I,I)=-HIJ(I)
C RETURN
7000 DO 7500 I=1,IC-1
7500 A(I,I)=HIJ(I)
DO 8000 I=ID+1,NNODES
8000 A(I,I)=-HIJ(I)
RETURN
END
C C C INCLUDE 'COMP'
C 4-15-83
SUBROUTINE NLINTEL(I,XXX,YYY,A,F,NXY)
%include implicit
DIMENSION A(NXY,NXY),F(NXY),DQ(3),PHIXI(3)
C %include blk2
%include blk3
%include blk4
%include blk5
LOGICAL PASS2
C C THIS ROUTINE CALCULATES THE SURFACE INTEGRALS
C ASSOCIATED WITH THE ELECTRODE KINETIC BOUNDARY CONDITION.
C
C C C INITIALIZE TERMS
GIJ=0.0
DO IO=1,90
0 DQ(K)=0.0
C C-------NORMAL QUADRATURE FOR USUAL CASES OR REMAPPED
C C INTEGRALS FOR SINGULAR CASES
C C------- BOUNDARY CONDITION MODULE------------------------
C = NOTE DS=SQR(DX*L+DYL*DYL) CANCELS WITH THIS BC =
C CONTAINS TERM = INHOMOGENEOUS TERM
C DOTERM = HOMOGENEOUS TERM
C AS FUNCTIONS OF XXL, YYL, DXL, DYL, GG, AND COMMON VARIABLES
C TEMP=EVAL*YYL
EX1=EXP(-TEMP)
EVX=EVAL*XXL
C
IF(FINITE) GOTO 134
C
GOTO (131, 132, 133), IBCOND
C
131 QTERM=EX1*(-DYL*SIN(EVX)+DXL*COS(EVX))
DQTERM=0.0
GOTO 138
132 QTERM=EX1*(-DYL*SIN(EVX))
DQTERM=DXL
GOTO 138
133 QTERM=EX1*DXL*COS(EVX)
DQTERM=-DYL*TAN(EVX)
GOTO 138
C
C
FINITE DOMAIN TERMS
134 EX2=EXP(TEMP-EV2HB)
C
GOTO (135, 136, 137), IBCOND
C
135 QTERM=-DYL*SIN(EVX)*(EX1-EX2) + DXL*COS(EVX)*(EX1+EX2)
DQTERM=0.0
GOTO 138
136 QTERM=-DYL*SIN(EVX)*(EX1-EX2)
DQTERM=DXL*(EX1+EX2)/(EX1-EX2)
GOTO 138
137 QTERM=DXL*COS(EVX)*(EX1+EX2)
DQTERM=-DYL*TAN(EVX)
C
138 QTERM=QTERM*SEVAL*GG
DQTERM=DQTERM*EVAL*GG
C
C--------- MODULE END -----------------------------------------------------
C
GIJ=GIJ + QTERM
DO 30 K=1,3
30 DQ(K)=DQ(K) + DQTERM*PHI(K, XI)
C
C
SINGULAR CASES REQUIRE FURTHER CALCULATION
IF (SINGULAR) GOTO 200
C
C
ADD RESULTS INTO RESIDUAL AND JACOBIAN - ALL DONE
GOTO 1000
C
C
C--------- FROM HERE ON CODE FOR SINGULAR INTEGRALS, GIJ CONTRIBUTIONS-----
C
C
C
TERMS GIJ AND DQ(1 2 3) ALREADY HAVE VALUES
C FROM REMAPPED NONSINGULAR INTEGRALS---BUILD ON THEM
C
200 CONTINUE
PASS2=.FALSE.
C
300 DO 500 IQ=1,NLN
XI=PLN(IQ)
IF (PASS2) XI=XI
IF (LEFT) XI=2.0*PLN(IQ) - 1.0
IF (RIGHT) XI=1.0 - 2.0*PLN(IQ)
DXL=0.0
DYL=0.0
XXX=0.0
YVL=0.0
DO 410 K=1,3
DXL=DXL + DPHI(K, XI)*XX(K)
DYL=DYL + DPHI(K, XI)*YY(K)
PHIXI(K)=PHI(K, XI)
500 CONTINUE
410 CONTINUE
XXL = XXL + PHIXI(K) * XX(K)
YYL = YYL + PHIXI(K) * YY(K)

C---THE LOG R TERM IS BURIED IN THE QUADRATURE PTS AND WEIGHTS---
C * AGAIN, DS FACTOR CANCELS WITH THIS BC *
   GG = WLN(IQ)
   IF (.NOT.CENTER) GG = GG + GG
C
C------- BOUNDARY CONDITION MODULE-------------------------
C * NOTE DS = SQRT(DXL * DXL + DYL * DYL) CANCELS WITH THIS BC *
C CONTAINS QTERM = INHOMOGENEOUS TERM
C DOTERM = HOMOGENEOUS TERM
C AS FUNCTIONS OF XXL, YYL, DXL, DYL, GG, AND COMMON VARIABLES
C
   TEMP = EVAL * YYL
   EX1 = EXP(-TEMP)
   EVX = EVAL * XXL
C
   IF (FINITE) GOTO 234
   GOTO (231, 232, 233). IBCOND
C
231 QTERM = EX1 * (-DYL * SIN(EXV) + DXL * COS(EXV))
   DOTERM = 0.0
   GOTO 238
232 QTERM = EX1 * (-DYL * SIN(EXV))
   DOTERM = DXL
   GOTO 238
233 QTERM = EX1 * DXL * COS(EXV)
   DOTERM = -DYL * TAN(EXV)
   GOTO 238
C
C FINITE DOMAIN TERMS
234   EX2 = EXP(TEMP - EV2HB)
C
   GOTO (235, 236, 237). IBCOND
C
235 QTERM = -DYL * SIN(EXV) * (EX1 - EX2) + DXL * COS(EXV) * (EX1 + EX2)
   DOTERM = 0.0
   GOTO 238
236 QTERM = -DYL * SIN(EXV) * (EX1 - EX2)
   DOTERM = DXL * (EX1 + EX2) / (EX1 - EX2)
   GOTO 238
237 QTERM = DXL * COS(EXV) * (EX1 + EX2)
   DOTERM = -DYL * TAN(EXV)
C
238 QTERM = QTERM * SEVAL * GG
   DOTERM = DOTERM * SEV^2 * GG
C
C--- MODULE END -------------------------------------------
C
   GIJ = GIJ + QTERM
   DO 430 K = 1, 3
430   DQ(K) = DQ(K) + DOTERM * PHIXI(K)
C
500 CONTINUE
   IF (.NOT.CENTER) GOTO 1000
   IF (PASS2) GOTO 1000
   PASS2 = .TRUE.
   GOTO 300
C CENTER NODE SINGULAR CASE IS BROKEN INTO 2 INTEGRALS, LEFT AND RIGHT
C OF CENTER - THAT'S WHY A SECOND PASS IS DONE ABOVE.
C LEFT AND RIGHT SINGULAR CASES REQUIRE GG=2*GG BECAUSE OF MAPPING DIFFERENCE
C
C - EXIT - ADD RESULTS
C
1000 F(I) = F(I) + GIJ
   DO 1020 K = 1, 3
1020 A(I, NODE(K)) = A(I, NODE(K)) + DQ(K)
C
RETURN
END

C C
C INCLUDE 'COMP'
FUNCTION PHI(N,Z)
%include implicit
GOTO (1,2,3),N
1 PHI=Z*(Z-1.0)/2.0
RETURN
2 PHI=1.0-Z*Z
RETURN
3 PHI=Z*(Z+1.0)/2.0
RETURN
END

C C
C INCLUDE 'COMP'
C 2/2/83
SUBROUTINE QUAD
%include implicit
C
%include btk3
C C
C ASSIGN QUADRATURE PTS AND WEIGHTS ACCORDING TO USER CHOICE GIVEN
BY THE VALUES OF NQ AND NLN
C
10 IF (NQ.NE.10) GOTO 6
PT(1)=.148874389816310D0
PT(2)=PT(1)
PT(3)=.43395394129247D0
PT(4)=-PT(3)
PT(5)=.679409568299024D0
PT(6)=-PT(5)
PT(7)=.86506336688985D0
PT(8)=-PT(7)
PT(9)=.973906528517172D0
PT(10)=-PT(9)
WT(1)=.295524224714753D0
WT(2)=WT(1)
WT(3)=.268266719309996D0
WT(4)=WT(3)
WT(5)=.219086362515982D0
WT(6)=WT(5)
WT(7)=.149451349150581D0
WT(8)=WT(7)
WT(9)=.066671344308688D0
WT(10)=WT(9)
GO TO 100
6 IF (NQ.NE.6) GOTO 3
PT(1)=.238619186083197D0
PT(2)=PT(1)
PT(3)=.661209386466265D0
PT(4)=-PT(3)
PT(5)=.932469514203152D0
PT(6)=-PT(5)
WT(1)=.467913934572691D0
WT(2)=WT(1)
WT(3)=.360761573048139D0
WT(4)=WT(3)
WT(5)=.17324492379170D0
WT(6)=WT(5)
GO TO 100
3 IF (NQ.NE.3) GOTO 4
PT(1)=SORT(3.0/5.0)
PT(2)=0.0
PT(3)=-PT(1)
WT(1)=5.0/9.0
WT(2)=8.0/9.0
WT(3)=WT(1)
GOTO 100
4
NO=4
C (DEFAULT EVEN IF NO.NE.4)
PT(1)=.3398810435848560D0
PT(2)=-PT(1)
PT(3)=.8611363115940530D0
PT(4)=-PT(3)
WT(1)=.6521451548625460D0
WT(2)=WT(1)
WT(3)=.3478548451374540D0
WT(4)=WT(3)
100
IF (NLN.NE.4) GOTO 30
PLN(1)=.04144848D0
PLN(2)=.24527491D0
PLN(3)=.55616545D0
PLN(4)=.84888239D0
WLN(1)=-.38346406D0
WLN(2)=-.38687531D0
WLN(3)=-.19043512D0
WLN(4)=-.03922548D0
GOTO 200
30
IF(NLN.NE.3) GOTO 40
PLN(1)=.06389079D0
PLN(2)=.36899706D0
PLN(3)=.76688030D0
WLN(1)=-.51340455D0
WLN(2)=-.39198044D0
WLN(3)=-.09461540D0
GOTO 200
40
NLN=2
C DEFAULT
PLN(1)=.11208200D0
PLN(2)=.60227690D0
WLN(1)=-.71859331D0
WLN(2)=-.28146068D0
200
CONTINUE
C MODIFY GAUSS-LEGENDRE POINTS AND WEIGHTS SO THEY APPLY
C TO -1 TO +INFINITY
DO 250 I=1,NQ
WIFI(I)=2.0*WT(I)/((1.0-PT(I))**2)
PIFI(I)=2.0*PT(I)/((1.0-PT(I))
250
CONTINUE
RETURN
END
C C
C INCLUDE 'COMP'
SUBROUTINE REGRID(X,Y,NXY,IERR)
%include implicit
DIMENSION X(NXY),Y(NXY)
LOGICAL IERR,EVENL
C C
C include blk11
C include blk2
C
C--------------------------
C USING THE POSITION OF THE FRESHLY MOVED ELECTRODE SURFACE,
C THIS SUBROUTINE ADJUSTS THE SPACING OF THE NODES IN THE
C LOCAL ELEMENT REGION. IF THE ELECTRODE HAS MOVED BEYOND
C THE LOCAL REGION, THE STRUCTURE OF THIS PROGRAM IS
C INAPPROPRIATE AND IERR IS SET TRUE.
C--------------------------
C--CLEAR TROUBLE FLAG----
IERR=.FALSE.

259
QBEM Code Listing

C--------- CORNER NODES ARE INDICATED IN COMMON --------
C
C IC= RIGHTHAND SIDE ELECTRODE CORNER
C ICC= TOP NODE RIGHTSIDE (CORRESPONDING ELEMENT EXTENDS TO INFINITY)
C IC=B= TOP NODE LEFTSIDE (ELEMENT EXTENDS TO INFINITY)
C ICC= LEFTSIDE TOP OF LOCAL ELEMENTS
C ICC= CHECK FOR GROWTH ABOVE LOCAL AREA ON RHS-----------
C DELY=Y(ICC)-Y(IB)
C IF (DELY.LE.0.0) GO TO 1000
C
C-------ADJUST RHS LOCAL NODE SPACING----------
C IF (IB+2.EQ.ICC) GOTO 15
C EVENL=VLOCAL.LE.0.0
C FACT=DELY/FLOAT(ICC-IB)
C IF(.NOT.EVENL)FACTR=DELY/(EXP(VLOCAL)-1.0)
C DO 10 I=IB+2,ICC-2,2
C IF (EVENL) Y(I)=Y(IB) + FLOAT(I-IB)*FACT
C IF(.NOT.EVENL) Y(I)=Y(IB)+(EXP(VLOCAL)*FLOAT(I-IB)/
C 2 FLOAT(ICC-IB))-.1)*FACTR
C Y(I-1)=(Y(I)+Y(I-2))/2.0
C CONTINUE
C 15 Y(ICC-1)=(Y(ICC)+Y(ICC-2))/2.0
C
C--------CHECK FOR GROWTH ABOVE LOCAL REGION ON LHS-------
C DELY=Y(IDD)-Y(1)
C IF (DELY.LE.0.0) GO TO 1000
C
C-------ADJUST LHS LOCAL NODE SPACING----------
C IF (IDD.EQ.NNODES-1) GOTO 25
C FACT=DELY/FLOAT(NNODES+1-IDD)
C IF(.NOT.EVENL)FACTL=DELY/(EXP(VLOCAL)-1.0)
C DO 20 I=IDD+2,NNODES-1,2
C IF (EVENL) Y(I)=Y(IDD) + FLOAT(I-IDD)*FACT
C IF(.NOT.EVENL) Y(I)=Y(1)+FLOAT(1.0-FLOAT(I-IDD)/
C 2 FLOAT(NNODES+1-IDD))-.1)*FACTL
C Y(I-1)=(Y(I)+Y(I-2))/2.0
C CONTINUE
C 25 Y(NNODES)=(Y(1)+Y(NNODES-1))/2.0
C
C RETURN
C
C----------------------------------------------------
C 1000 IERR=.TRUE.
C
C-----HERE IF PROGRAM IS INVALIDATED BY ELECTRODE GROWTH ABOVE LOCAL REGION---
C----- OR BY GROWTH THROUGH SYMMETRY PLANE---
C RETURN
C END
C
C SGECO. SGEFA, SGESEL. etc. see Dongarra et al (1979) (LINPACK)

C
C INCLUDE 'COMP'
C FUNCTION SHAPE(Z)
C
C#include implicit
C#include blk1
C
C THIS FUNCTION DETERMINES THE INITIAL CATHODE SHAPE BASED
C ON THE INTEGER VARIABLE 'TYPE' AND THE FACTOR 'AMP' FOUND
C IN THE COMMON BLOCK BLK1. THE FUNCTION SUPPLIES VALUES
C OF Y WHEN CALLED WITH VALUES OF X AS ITS ARGUMENT. VALUES
C OF X WILL RANGE BETWEEN 0 AND 1.
C
C PI=3.14159265358979300
GOTO (10,20,30),TYPE
C
C TYPE=1. COSINE WAVE, SCALLED FROM (0,0) TO (1,AMP)
10 SHAPE=AMP*0.5*(1.0-COS(Z*PI))
RETURN
C

260
C TYPE=2, TRIANGLE WAVE
20    SHAPE=AMP*Z
    RETURN
C TYPE=3, COSINE WAVE, OF PERIOD 1
30    SHAPE=AMP*0.5*(1.0-COS(Z*2.0*PI))
    RETURN
    END
APPENDIX H

LBEM COMPUTER CODE LISTING

LBEM CALLING HIERARCHY

LBEM
  GRID
    SHAPE
  REGRID
  MATFIXED
    ELMT
    INTEL
      GIJ
  MATVAR
    ELMT
    INTEL
      GIJ
  MATNL
    ELMT
    GIJ
SGECD (LINPACK)
  SGEFA (LINPACK)
    ISAMAX (LINPACK)
  SSCAL (LINPACK)
  SAXPY (LINPACK)
  SSCAL (LINPACK)
  SASUM (LINPACK)
  SDOT (LINPACK)
  SAXPY (LINPACK)
SGESL (LINPACK)
  SAXPY (LINPACK)
  SDOT (LINPACK)
ERRLIST
CPU_TIMER

MODULE NOTES

LBEM - main program
  - reads the input data (file PTDATA)
  - sets miscellaneous parameters
  - writes report files (files REPORT, ELECTRODE)
  - controls the time integration
  - stores temporary matrix terms in file AFSAVE
  - deforms the active surface

CPU_TIMER - function (PL/1 programming language)
  - calls system vclock() function
  - returns elapsed virtual cpu time (seconds)

ELMT - subroutine
  - sets up quantities needed to calculate integrals on the
current element

ERRLIST - subroutine
  - computes the error in the current active surface position
    and the error in the last field solution

INTEL - subroutine
  - computes Hij integrals on the current element

GIJ - subroutine
  - calculates Gij integrals with log(r) kernels

GRID - subroutine
  - draws initial boundary element mesh

MATFIXED - subroutine
  - computes matrix problem contributions for the boundary
    elements that are spatially fixed

MATNL - subroutine
  - computes matrix terms from the boundary condition on the
    active surface

MATVAR - subroutine
  - computes matrix problem contributions for the deforming
    boundary elements
  - computes H_{ii} terms by difference

REGRID - subroutine
  - adjusts node spacing in deforming region

SGECO - subroutine (LINPACK)
  - factor the main matrix and estimate its condition

SGESL - subroutine (LINPACK)
  - solve the linear matrix problem

SHAPE - function
  - used to compute the initial active surface shape
  - returns surface ordinate given the abscissa

Indicated routines are from the LINPACK library distributed by
the Society for Industrial and Applied Mathematics and are
documented by Dongarra et al (1980).

SIGNIFICANT COMMON AND INPUT VARIABLES

A(NXY,NXY)  - the main matrix, it multiplies the unknowns
AMP         - initial amplitude of the active surface
DELTIME     - time step size
DX           - change in X over current element
DY           - change in Y over current element
ELTYPE
- type of current element, see ELMT subroutine
EPSILON
- small tolerance for comparison
EVAL
- the eigenvalue μ
FXYN(IB)
- rates of surface deformation at each active surface node
HDELTA
- location of the top surface (finite domain)
HIJ(NXY)
- sum of H_i contributions, i≠j
HLOCAL
- top of the deforming region
IB
- rightmost node on the active surface, also, number of active surface nodes
IC
- rightmost node along domain top (finite domain)
ICC
- node on RHS where fixed and deforming regions meet
ID
- leftmost node along domain top (finite domain)
IDD
- node on LHS where fixed and deforming regions meet
IBOUND
- boundary condition choice (1,2, or 3)
ICHOICE(5)
- report control, see input data file
INFINITE
- true for the infinite domain problem
ITCOUNT
- count of completed time step iterations
NBULK
- number of elements along the fixed part of the symmetry walls
NELECT
- number of elements along the active surface
NELMTS
- total number of elements
NDELTA
- number of elements along the top boundary
NLOCAL
- number of elements along the deforming parts of the symmetry walls
NNODES
- total number of nodes
NODE1,NODE2
- node indexes on the current element
NWRITE
- active surface location is saved every NWRITE time steps
SCALAR
- the field solution is multiplied by this scalar (it was kept at 1.0)
SINGULAR
- true when current base node is on current element
SOLN(NXY)
- the righthand-side vector in the simultaneous linear equation set, becomes the matrix solution
STOPTIME
- integration stop time
TTIME
- simulated integration time
TYPE
- choice of the starting active surface shape
VSBulk
- vertical spacing factor, fixed region, 0.0 for uniform spacing
VSLocal
- spacing factor, deforming region, 0.0 for uniform vertical node spacing
X(NXY),Y(NXY)
- node locations, global coordinates
XO(IB),YO(IB)
- constants related to initial active surface position, needed for error reporting
XX1,XX2
- X coordinates of current element
YY1,YY2
- Y coordinates of current element
YN(IB)
- saved position of the active surface

264
LBEM Code Listing

C implicit - specifies precision
IMPLICIT INTEGER (I-N), DOUBLE PRECISION (A-H,O-Z)

COMMON BLOCKS

COMMON/BLK1/HLOCAL,HBULK,WELECT,VSLOCAL,VSBulk,AMP,TYPe
INTEGER TYPE

COMMON/BLK2/NELECT,NLOCAL,NBULK,NDELTa,NNODES,NELMTS
2,IB,IC,ID,ICC,IDD

COMMON/BLK4/ELTYPe,SINGULAR,FINITE,NODE1,NODE2,XX1,XX2,YY1,YY2
2,DX,DY,DX2Y2,GIC,Etao
INTEGER ELTYPe
LOGICAL SINGULAR,FINITE

COMMON/BLK5/EVAL,SCALAR,SEVAL,EV2HB,EXPEVHB,PI,IBCOND,INFINITE
LOGICAL INFINITE

INPUT DATA FILE (free format)

1.0,4.0,0.0,0.0,0.0,3.1;; HLOCAL,HBULK,VSLOCAL,VSBulk,AMP1itude,TYPeofshape
15,8,8,5,-10.0;;NELECT,NLOCAL,NBULK,NDELTa,Etao (KEEP NELECT,NDELTa odd -BC 3)
0.0,0.0,;;DELTIME,STOPTIME
1,1,-2,-3,-4,+5 ;;NWriTE,CHOICE(5)
1.0, +1.0,1. ;;EVAL,SCALAR, IBCOND
IF CHOICE(1)>0 GET INITIAL REPORT
IF CHOICE(2)>0 GET FINAL REPORT
CHOICE(3) NOT USED
IF CHOICE(4)>0 GET CONDITION RECORD AND VERBOSE ERROR REPORT
IF CHOICE(5)>0 ONLY CALCULATE DIST. ONCE
LBEM Code Listing

%global card round auto_zero
INCLUDE "COMP"

SOLUTION of LaPlace's equation by Boundary Integral
Equation Methods on a periodic Finite or Infinite
domain. A source boundary condition is imposed
in any of three ways leading to a known solution
for the deformation of the active surface.
Linear boundary elements and Improved Euler
integration is used.

%include implicit
PARAMETER (INODES=224,IXYY=NODES*INODES,IELECT=65)

COMMON STATEMENTS ARE GOTTEN FROM INCLUDE FILES

%iinclude bk1
%iinclude bk2
%iinclude bk4
%iinclude bk5

DIMENSION X(INODES),Y(INODES),A(IXYY),SOLN(INODES),HIJ(INODES)
A BIGGER PROBLEM CAN BE DONE IF ARRAYS ARE IN COMMON
COMMON/BK6/X(INODES),Y(INODES),SOLN(INODES),HIJ(INODES),A(IXYY)
DIMENSION XO(IELECT),YO(IELECT),YN(IELECT),FXYN(IELECT)
DIMENSION IPVT(INODES),ICHOICE(S)
LOGICAL REPORT,EUPRED,ERROR

DIMENSION ITIME(3)
EXTERNAL date_time_(descriptors)
EXTERNAL clock_(descriptors)
CHARACTER*30 TDSTRING
EXTERNAL CPU_TIME(DESCRIPTIONS)
REAL CPUSTART,CPUSUM

-------------------------------- INITIALIZE -----------------------------

---OPEN FILES------
OPEN(2,MODE="IN",FORM="FORMATTED",FILE="ptdata")
OPEN(3,MODE="IN",FORM="FORMATTED",FILE="electrode")
OPEN(4,MODE="INPUT",FORM="FORMATTED",FILE="report")
OPEN(20,MODE="INPUT",FORM="UNFORMATTED",FILE="afsave")
CALL FOPEN (2,"PTDATA")
CALL FOPEN (3,"ELECTRODE")
CALL FOPEN (4,"REPORT")
CALL FOPEN (20,"AFSAVE")

-----READ PROBLEM PARAMETERS-----
1ST ROW GEOMETRICAL DATA
READ (2,590) HLOCAL,HBULK,VSLOCAL,VSBUK,AMP,TYPE
FORMAT(V)
2ND ROW ELEMENT LAYOUT DISTRIBUTION
READ (2,590) NELECT,NLOCAL,HBULK,NDELTA,ETAO
3RD ROW, INTEGRATION CONTROL
READ (2,590) DELTIME,STOPTIME
4TH ROW, QUADRATURE AND MISCELLANEOUS
READ (2,590) NWRITE,ICHOICE
5TH ROW ELECTRODE KINETIC PARAMETERS
READ (2,590) EVAL,SCALAR,IBCOND

------------------------SET MISCELLANEOUS PARAMETERS--------

PI*3.141592653589793D0
TOLERANCE FOR STOP TIME COMPARISON
EPSILON=1.0E-06
C SELECT INFINITE OR FINITE DOMAIN BASED ON ELEMENTS ALLOTED FOR TOP
FINITE=.TRUE.
IF (NDELTA.EQ.0) FINITE=.FALSE.
FINITE=.TRUE.
IF (FINITE) FINITE=.FALSE.
C NELMTS=NSELECT + 2*NLOCAL + 2*NBULK +NDELTA
C IB,IC,ICC,ID,IDDD IDENTIFY CORNER NODE NUMBERS OR DIVISION OF THE
C ELEMENTS INTO LOCAL AND BULK REGIONS
IB=NSELECT + 1
ICC=IB+NLOCAL
IC=ICC+NBULK
ID=IC+1
IF(FINITE) ID=IC+NDELTA
IDD=ID+NBULK
C NNODS=NELMTS + 1
IF (FINITE) NNODS=NELMTS
NXXY=NNODS*NNODS
C TEST FOR ADEQUATE STORAGE
IF (NNODS.LE.1) GOTO 5
WRITE (4,600) NNODS,IB,INODES,ISELECT
600 FORMAT("SOLUTION REQUIRES ",I6," NODES WITH ",I6," CATHODE NODES"
STOP
5 CONTINUE
C
C SET ELECTRODE WIDTH - - - IT IS THE REFERENCE LENGTH
WELECT=1.0
C SET EIGENVALUE FOR ANALYTICAL SOLN
EVAL=FLOAT(INT(EVAL))*PI/WELECT
SEVAL=SCALAR*EVAL
EV2HB=EV2*EXP(2.0*HBULK)
EXPVB=EXP(EV2HB)
C
C CHECK ORIGIN FOR 1/R^3 DECAY OF INFINITE ELEMENTS, ETAO
C VALUE WAS GIVEN BY INPUT FILE
IF(SETAO.GE.-1) ETAO=-8.5525
C
REPORT=.TRUE.
IF (ICHICE(1).LE.0) REPORT=.FALSE.
TTIME=0.0
ITCOUNT=0
C
C-----------------------------WRITE REPORT INFO TO REPORT FILE -----------------------------
CALL date_time(clock(),TSTRING)
CALL CPU_TIMER(CPUTIME)
WRITE (4,700) TSTRING
700 FORMAT(/************","3X,A24,3X,"************//)
C CALL TIME(ITIME,IER)
C CALL FGDAY (MONTH,IDAY,IYEAR)
C WRITE (4,700) MONTH,IDAY,IYEAR,ITIME
C700 FORMAT(/"25 Ankara",I5,"/",I2,"," ,I2, I5,":":" ,I2,"":",I2,3X,25("))
WRITE (4,705)
705 FORMAT("Two Dimensional Moving Boundary Study Using",
2 " Boundary Integrals "/
3 " * Linear Boundary Element Version 1.2. 72 bit Reals **/
IF (.NOT.FINITE) WRITE(4,707) SCALAR,EVAL
707 FORMAT("SEMI-INFINITE Domain with Analytical Solution/
2 5X,F10.3," *COS(EV1*X)*EXP(-EV1*Y) (EV1=" ,F8.4," )")
IF (FINITE) WRITE(4,708) SCALAR,EVAL
708 FORMAT("FINITE Domain with Analytical Solution /
2 5X,F10.3," *COS(EV1*X) *(EXP(-EV1*Y)-EXP(EV1*(Y-2*HBULK))))",
3 " (EV1=" ,F8.4," )")
WRITE(4,709) ISCOND
709 FORMAT(" Boundary Condition imposition type=" ,I2)
WRITE (4,710) HLOCAL,HBULK,WELECT,VSL,VSBUK,ETAO,AMP,TYPE
710 FORMAT("/"DOMAINE GEOMETRY")
LBEM Code Listing

2 5X."HLOCAL",4X."HBULK",5X."WIDTH",5X."VSLOCAL  VSBulk",
3 6X."ETAO",4X."AMPLITUDE TYPE"/7F10.3,11(0)  WRITE (4,720) NLECTL,NLOCAL,HBULK,NDELTA,NEMTS,NNODES
720  FORMAT (/*DISTRIBUTION OF BOUNDARY ELEMENTS*/
2 6X."NLECTL NLOCAL HBULK NDELTA TOTAL ELEMENTS TOTAL NODES"
3 /5(18)  WRITE (4,730) DELTIME,STOPTIME,NWRITE,ICHOICE
730  FORMAT (/*INTEGRATION PARAMETERS AND PROGRAM CONTROL*/
2 4X."DELTIME STOPTIME NWRITE ICHOICE"/2F10.5,110,514/)
C
C---------- ALLOC FOR PERIODIC ELECTRODE LOCATION REPORTS
C WRITE (3,740) MONTH,DAY,YEAR,ITIME
740  FORMAT("LINBEM PROGRAM RUN ",I2,"/",I2,"/",I2,3X,I2,":"I2,"":"I2/
2 "TIME, POSITION ERR. SOLN ERR"/
C WRITE (3,740) TSTRING
C
C---------- ESTABLISH FIXED ELEMENT NODE LOCATIONS -------
C CALL GRID(X,Y,NNODES)
C WRITE (10,742) (X(I),Y(I),I=1,NNODES)
742  FORMAT(I4,5X,2F10.3)
C
C---------- SET UP XO, YO FOR PERIODIC ERROR REPORTS ---------
DO 13 I=1,1B
C
C  TEMP=EXP(EVAL*X(I))
C IF (INFINITE) GOTO 11
C
C  YO(I)=(TEMP+XPVEH)/(TEMP-XPVEH)
10  XO(I)=2.0*EVAL+SCALAR*COS(EVAL*X(I))/XPVEH
C GOTO 12
11  YO(I)=TEMP
12  XO(I)=EVAL+SCALAR*COS(EVAL*X(I))
13  CONTINUE
14  CONTINUE
C
C---------- CALC MATRIX CONTRIBUTIONS FROM FIXED NODES -------
C CALL MATFIXED (X,Y,NNODES,A,H12)
C
C---------- SAVE FIXED RESULTS AS A BASE FOR EACH TIME ITERATION ----
C WRITE (20) (A(I),I=1,NXXYY),(H12(I),I=1,NNODES)
C
C FINISH DEFINING MOVING REGION OF ELEMENTS
C CALL REGRID(X,Y,NNODES,ERROR)
C
C---------- END INITIALIZATION --------------------

C
C---------- MAIN INTEGRATION LOOP ENTRY POINT ---------
100 CONTINUE
C
C--------- SET FLAG FOR EULER PREDICTION -------
EUPRED=TRUE.
C--------- SAVE ELECTRODE POSITION IN YN ----
DO 109 I=1,1B
109  YNI=Y(I)
C------110 CONTINUE IS ENTRY POINT AFTER EULER PREDICTION----
C
C--------- SOLVE FOR FLUX
C REINITIALIZE MATRIX
IF (ITCOUNT.EQ.0.AND.EUPRED) GOTO 113
REWIND 20
READ (20)(A(I),I=1,NXXYY),(H12(I),I=1,NNODES)
C CLEAR RHS VECTOR
DO 112 I=1,1B
112  CONTINUE
C
C--------- ADD CONTRIBUTION FROM MOVING NODES -----

263
CALL MATVAR(X,Y,NNODES,A,HIJ)
CALL MATNL(X,Y,NNODES,A,SOLN)
CALL SGECO(A,NNODES,NNODES,IPVT,RCOND,HIJ)
NOTE - HIJ IS USED AS A WORK VECTOR
IF (RCOND.NE.0) RCOND=1.0/RCOND
CALL SGESL(A,NNODES,NNODES,IPVT,SOLN,0)
IF(ICHOOSE(4).GT.0)WRITE(4,766)TTIME,RCOND
FORMAT(’T=’,F12.5,’ Matrix Condition=’,G12.5)
------ ON FIRST PASS REPORT THE SOLUTION AND NODAL LAYOUT------
IF(.NOT.REPORT) GOTO 127
CALL ERRLIST(X,Y,SOLN,NNODES,XO,YO,YN,IB,TTIME,4,1)
CALL ERRLIST(X,Y,SOLN,NNODES,XO,YO,YN,IB,TTIME,3,ICHOOSE(4))
Chance to exit with initial solution only
IF(ICHOOSE(5).GT.0) GOTO 1000
REPORT=.FALSE.
----------- EULER PREDICTION ---------
IF (.NOT.EUPRED) GOTO 150
DO 135 I=1,IB
FXYN(I) = SOLN(I)
Y(I) = Y(I) + DELTIME*FXYN(I)
135 CONTINUE
READJUST TRENCH AREA ELEMENTS
CALL REGRID(X,Y,NNODES,ERROR)
IF (ERROR) GOTO 900
------ NOW FIND FLUX WITH THIS POSITION ----
EUPRED=.FALSE.
GO TO 110
--------- HERE FOR TRAPEZOID CORRECTOR ---------------------
CONTINUE
CALCULATE NEW ELECTRODE MOVEMENT RATE AND AVERAGE IT WITH EULER
PREDICTED RATE
MOVE THE ELECTRODE
DO 165 I=1,IB
FXYN(I) = FXYN(I) - SOLN(I)
SAVE = Y(I)
Y(I) = YN(I) + DELTIME*FXYN(I)/2.0
YN(I) = SAVE
165 CONTINUE
READJUST ELEMENTS
CALL REGRID(X,Y,NNODES,ERROR)
IF (ERROR) GOTO 900
------------- THIS COMPLETES AN INTEGRATION CYCLE ------------------
ITCOUNT = ITCOUNT + 1
TTIME = TTIME + DELTIME
WRITE ELECTRODE POSITION EVERY NWRITE CYCLES
IF (MOD(ITCOUNT,NWRITE).NE.0) GOTO 200
CALL ERRLIST(X,Y,SOLN,NNODES,XO,YO,YN,IB,TTIME,3,ICHOOSE(4))
C------- CHECK TIME AGAINST STOP TIME
200     IF (TTIME.GE.STOPTIME-EPSILON) GO TO 1000
C
C RETURN FOR MORE INTEGRATION
C
GO TO 100
C
ERROR RETURN
900     WRITE(4,781) TTIME
781     FORMAT("ERROR RETURN, SIML. TIME= ",F10.4)
C-----------------------------------------------
C****** HERE AT END ******
C
1000    CONTINUE
C
C
C CALL OVERFL(I)
C IF (I.NE.2) WRITE (4,782) I
782     FORMAT("OVERFLOW CODE=",I2," 3=UNDER, 1=OVER OR BOTH")
C
C FINAL REPORT ON LAYOUT AND SOLN
IF (ICHOICE(2).GT.0) WRITE (4,785)
785     FORMAT ("***** FINAL NODE LAYOUT *****")
    IF (ICHOICE(2).GT.0) CALL ERRLIST(X,Y,SOLN,NNODES,XO,YO,YN,IB,
    2 TTIME,4,1)
C
CALL dat_time (clock(),TDSTRING)
CALL CPU_TIMER(CPUTIME)
CPUTIME=CPUTIME-CPUSTART
WRITE (4,790) TDSTRING,CPUTIME
790     FORMAT("Elapsed CPU Sec.=",F12.4)
C
CALL TIME(ITIME,IER)
C
DTIME=3600.0*ITIME(1)+60.0*ITIME(2)+ITIME(3)
C
WRITE(4,790) ITIME,DTIME
C
WRITE(10,790) ITIME,DTIME
C790     FORMAT ("ALL DONE AT ",I2,:",I2,:",I2,
C 2 " Elapsed Seconds=",F5.0)
END
LBEM Code Listing

C INCLUDE 'COMP'
C 2/15/83
SUBROUTINE ELMT(N,X,Y,NXY)

%include implicit
DIMENSION X(NXY), Y(NXY)

%include blk2
%include blk4

C THIS ROUTINE SETS UP VARIABLES THAT CHARACTERIZE BOUNDARY
C ELEMENT, N, FOR THE INTEGRATIONS TO BE PERFORMED ALONG ITS SURFACE
C ***** LINEAR BOUNDARY ELEMENT VERSION *****

C NODE NUMBERING HAS A DISCONTINUITY FOR INFINITE ELEMENTS
IDISC = 0
IF (N.GT.NELEST+NLOCAL+NBUKN) .AND. .NOT.FINITE) IDISC = 1
NODE1 = N + IDISC
NODE2 = N + 1 + IDISC
IF(N.EQ.NELMTS) NODE2=1
XX1 = X(NODE1)
YY1 = Y(NODE1)
XX2 = X(NODE2)
YY2 = Y(NODE2)
DX = XX2 - XX1
DY = YY2 - YY1
DXY = DX*DX + DY*DY
GIJC = 1.0/(4.0*SQRT(DXY))

C-------- SET UP VARIABLES TO DESCRIBE ELEMENT -------
C -- ETYPE --- 1=ELECTRODE, 2=INSUL, 3=TOP, 4=INFINITE RHS, 5=INFINITE LHS
ETYPE=2
IF (ID.GT.NODE1 .AND. NODE1.GE.IC) ETYPE=3
IF(N.LE.NELEST) ETYPE=4
IF (FINITE) GOTO 30
IF(NODE2.EQ.IC) ETYPE=5
IF(NODE1.EQ.ID) ETYPE=5
30 CONTINUE
RETURN
END

C
C
C INCLUDE 'COMP'
C SUBROUTINE ERRLIST(X,Y,SOLN,NXY, XO,YO,YN,IB, TTIME, IOUT, ISWITCH)

%include implicit
DIMENSION X(NXY), Y(NXY), SOLN(NXY), XO(IB), YO(IB), YN(IB)

C 1/26/83
C THIS ROUTINE COMPUTES THE ERROR IN THE CATHODE POSITION
C THAT HAS ACCUMULATED WITH INTEGRATION IN TIME AS WELL AS
C THE SOLUTION ERROR FOR THE ACTUAL CATHODE POSITION

YO - CONSTANT VECTOR RELATED TO INITIAL CATHODE POSITION
YO=EXP(EVAL*YO) (INFINITE DOMAIN)
YO={EXP(EVAL*YO)+EXP(EVAL*HBUKN)}/[EXP(EVAL*YO)-EXP(EVAL*HBUKN)] (FINITE DOMAIN)

XO - CONSTANT VECTOR RELATED TO X POSITIONS
XO=EVAP+ALPHA*SCALAR*COS(EVAL*xo) (INFINITE DOMAIN)
XO=2*EVAP+ALPHA*SCALAR*COS(EVAL*xo)*EXP(-EVAP*HBUKN)

TTIME - PRESENT TIME

POSITION ERROR IS FOR POSITION OF GRID LOCATION Y(I) AT SIMULATION
TIME TTIME.

SOLUTION ERROR IS FOR GRID POSITIONS IN YN() WHEN SOLN WAS PREVIOUSLY
CALCULATED

C BLKS - BOUNDARY CONDITION PARAMETERS AND "INFINITE" - TELLS WHICH DOMAIN
%include blk5

271
IF (ISWITCH.GT.0) WRITE(IOUT,600) TTIME
FORMAT(15x,"POSITION AND PREVIOUS SOLUTION ERR. SIMUL. TIME="),
2 F10.5/"%X,"%Y",14x,"%X","%Y",2x,"%SOLN","%SOLN","%SOLN ERR="/)
SOLRMS=0.0
POSRMS=0.0
DO 100 I=1,IB
C
C SOLTRUE = TRUE SOLUTION
C TEMP=EVAL*YN(I)
C ETTEMP=EXP(-TEMP)
C IF (.NOT.INFINITE) ETTEMP=ETEMP - EXP(TEMP-EV2HB)
C SOLTRUE=SCALAR*CDX(EVAL*X(I))*ETEMP
C SOLERR=(SOLN(I)-SOLTRUE)/ABS(SCALAR)
C SOLRMS=SOLRMS + SOLERR*SOLERR
C
C IF (INFINITE) GOTO 50
C C CALCULATION OF TRUE CATHODE POSITION FOR FINITE DOMAIN
C TEMP=YN(I)*EXP(TTIME*XO(I))
C PTRUE=ALOG((1.0+TEMP)*EXPVHB/(TEMP-1.0))/EVAL
C GOTO 80
C
C C CALCULATION OF TRUE CATHODE POSITION FOR INFINITE DOMAIN
C PTRUE=ALOG(TTIME*XO(I))/EVAL
C B80 PERR*Y(I)-PTRUE
C POSRMS=POSRMS+PERR*PERR
C C
C IF (ISWITCH.GT.0) WRITE(IOUT,700) X(I),Y(I),PERR,SOLN(I),SOLERR
C FORMAT(5(G14.6,".",*))
C 100 CONTINUE
C SOLRMS=SQR(SOLRMS/IB)
C POSRMS=SQR(POSRMS/IB)
C IF (ISWITCH.GT.0) WRITE (IOUT,750) POSRMS,SOLRMS
C FORMAT(15x,"RMS POSITION ERROR="),G12.5/"RMS SOLUTION ERROR="
C 2 G12.5/
C IF (ISWITCH.LE.0) WRITE (IOUT,760) TTIME,POSRMS,SOLRMS
C FORMAT(3(G12.5,".",*))
C RETURN
C END
C
C INCLUDE 'COMP'
C 2/15/83

SUBROUTINE GIJ(I,XBN,YBN,GIJ1,GIJ2)
%include implicit
%include blk4
C
C THIS ROUTINE CALCULATES NORMAL GIJ TYPE INTEGRALS FOR LINEAR
C BASIS FUNCTION BOUNDARY ELEMENTS
C [GIJ TYPE INTEGRALS HAVE LN(R) TERMS AND MULTIPLY FLUXES]
C--------------------------------------------------------------------------
C ASSUMED FROM COMMON AND STACK:
C CURRENT ELEMENT - NODE1,NODE2 (XX1,YY1) - (XX2,YY2), DX=XX2-XX1, DY
C 2X2Y2=DXX**2 + DYY**2, GIJC=A CONSTANT
C BASE NODE - I, XBN,YBN
C
C DXI= XXI - XBN
C DYL= YYI - YBN
C D2X= DXI*DYI + DXI
C BB = BB + BB
C CC = DYI*DYI + DXI*DXI
C B2A = BB/(DX2Y2+DX2Y2)
LBEM Code Listing

ABC = DX2Y2 + BB + CC
IF (NODE2.EQ.1) GOTO 200

C CODE WHEN ABC<>0 [WHEN NODE2.EQ.1, ABC=0, AND (1+B2A)=0]
ABCLOG = ALOG(ABC)
ABC = ABC * ABCLOG

200 GIJ1 = -ABC + DX2Y2 + BB
GIJ2 = ABC - (DX2Y2 + BB)
TEMP = (1.0 + B2A) * ABCLOG - 2.0
IF (NODE1.EQ.1) GOTO 220

C CODE FOR WHEN BB AND CC ARE NONZERO
CCLLOG = ALOG(CC)
TEMP = TEMP - B2A * CCLLOG
CCLLOG = CCLLOG + CC
GIJ1 = GIJ1 + CCLLOG
GIJ2 = GIJ2 - CCLLOG
ADD = ABS(DY*DXI - DX*DYI)
IF (ADD .LT. 1.0E-7) GOTO 220
C CODE WHEN I IS NOT ON ELEMENT, DD IS NONZERO
ADD = ABS(DY*DXI - DX*DYI)
ATA = BB/(ADD+ADD)
ATA = ATAN(ATA + DX2Y2/ADD) - ATAN(ATA)
TEM2 = ATA*ADD/DX2Y2
TEMP = TEMP + TEM2 + TEM2

220 GIJ1 = GIJ1 + (DX2Y2 + DX2Y2 + BB)*TEMP
GIJ2 = GIJ2 + (GIJ2 - BB)*TEMP

C GIJ1 AND GIJ2 ARE NORMAL GIJ TYPE INTEGRALS FOR BASIS FUNCTIONS
C 1 AND 2 RESPECTIVELY.
RETURN
END

C

1/18/83

INCLUDE 'COMP'

SUBROUTINE GRID (X,Y,NXY)
%include implicit

C SETS UP THE NODE STRUCTURE FOR A GENERAL ELECTROCHEMICAL CELL
C -BOUNDED ON LEFT AND RIGHT BY PLANES OF SYMMETRY (INSULATORS)
C -OPPOSING ELECTRODE CAN BE AT INFINITY OR CELL CAN BE CLOSED
C -BY A HORIZONTAL SURFACE
C -ACTIVE ELECTRODE INITIAL SHAPE IS GIVEN BY SHAPE() FUNCTION
C IT RANGES FROM X=0 TO X=1.0

C NODAL LAYOUT
C  **** LINEAR BOUNDARY ELEMENTS ****
C -NODE NUMBERING STARTS WITH X=0 ACTIVE ELECTRODE CORNER AND CONTINUES
C COUNTERCLOCKWISE AROUND THE CELL
C
C -NELECT, # OF ELEMENTS ON ELECTRODE
C -NLOCAL, # OF ELEMENTS ON LEFT AND RIGHT PLANES BETWEEN ELECTRODE
C AND TOP OF LOCAL REGION, HLOCAL
C -NBLK, # OF ELEMENTS ON LEFT AND RIGHT SYMMETRY PLANES, LAST ONE
C ON EACH SIDE CAN EXTEND TO INFINITY
C -NDELTA, # OF ELEMENTS ALONG CELL TOP, IF 0, FINITE SHOULD BE FALSE
C
C IB= RIGHHAND SIDE ELECTRODE CORNER
C IC= TOP NODE RIGHTSIDE (CORRESPONDING ELEMENT MAY EXTEND TO INFINITY)
C ICC= RIGHHAND SIDE TOP OF LOCAL ELEMENTS
C ID= TOP NODE LEFTSIDE (ELEMENT MAY EXTEND TO INFINITY)
C IDD= LEFTSIDE TOP OF LOCAL ELEMENTS
C
C DIMENSION X(NXY),Y(NXY)

C %include b1k1
%include b1k2
%include b1k4

C LOGICAL EVEN,EVENL

273
 EVENL=(VSLocal.LE.0.0) EVEN=(VSBulk.LE.0.0)
C-------------LOCATE ACTIVE ELECTRODE NODES-------------
DO 20 I=1,IB
   X(I)=FLOAT(I-1)*WELECT/FLOAT(IB-1)
   Y(I)=SHAPE(X(I))
20           CONTINUE
C-------------RIGHT HAND SIDE, LOCAL ELEMENTS ABOVE ELECTRODE----
   IF Y(IB).GE.HLOCAL) GOTO 1000
   IF (.NOT.EVENL) FACTR=(HLOCAL-Y(IB))/(EXP(VSLocal)-1.0)
   DO 30 I=IB+1,IC
      X(I)=WELECT
   IF (EVENL) Y(I)=Y(IB)+ FLOAT(I-IB)*(HLOCAL-Y(IB))/2
      FLOAT(IC-IB)
   IF (.NOT.EVENL) Y(I)=Y(IB)+(EXP(VSLocal-FLOAT(I-IB))/
      FLOAT(IC-IB)-1.0)*FACTR
30          CONTINUE
C-------------RIGHT HAND SIDE BULK - ABOVE LOCAL ----
   IF (.NOT.EVENL) FACTR=(HBulk-HLOCAL)/(EXP(VSLocal)-1.0)
   DO 50 I=IC+1,IC
      X(I)=WELECT
   IF (EVENL) Y(I)=HLOCAL + FLOAT (I-IC)*(HBulk-HLOCAL)/2
      FLOAT(IC-IC)
   IF (.NOT.EVENL) Y(I)=HLOCAL+(EXP(VSBulk-FLOAT(I-IC))/
      FLOAT(IC-IC)-1.0)*FACT
50          CONTINUE
C---------IF FINITE - TOP OF CELL -----------------------
   IF (.NOT.FINITE) GOTO 62
   DO 60 I=IC+1,IC
      X(I)=WELECT*(1.0-FLOAT(I-IC)/FLOAT(ID-IC))
   Y(I)=HBulk
60          CONTINUE
C--------------LEFT-HAND SIDE NODES-BULK--------------
62          DO 70 I=ID,IDD
   X(I)=0.0
   IF (EVENL) Y(I)=HBulk- FLOAT(I-ID)*(HBulk-HLOCAL)/2
      FLOAT(IDD-ID)
   IF (.NOT.EVENL) Y(I)=HLOCAL+(EXP(VSBulk)*FLOAT(I-ID)/
      FLOAT(IDD-ID)-1.0)*FACT
70          CONTINUE
C--------------LEFT-HAND SIDE LOCAL ELEMENTS ABOVE ELECTRODE----
   IF (Y(I).GE.HLOCAL) GOTO 1000
   IF (IDD.EQ.NNODES) RETURN
   IF (.NOT.EVENL) FACTL=(HLOCAL-Y(I))/(EXP(VSLocal)-1.0)
   DO 80 I=IDD+1,NNODES
      X(I)=0.0
   IF (EVENL) Y(I)=HLOCAL- FLOAT(I-IDD)*(HLOCAL-Y(I))/2
      FLOAT(NNODES-I-IDD)
   IF (.NOT.EVENL) Y(I)=Y(I)+(EXP(VSLocal-FLOAT(I-IDD))/
      FLOAT(NNODES-I-IDD))-1.0)*FACTL
80          CONTINUE
C ERROR RETURN
1000         CONTINUE
C WRITE(4,1100) Y(I),Y(IB),HLOCAL
1100         FORMAT//"Y(I)=" ,F12.4," Y(IB)=","F12.4," HLOCAL=" ,F12.4/
2         "SHAPE FUNCTION HAS EXCEEDED SPECIFIED LOCAL ELEMENT AREA="/3
3         "STOP IN GRID")
C STOP
END
C
C INCLUDE 'COMP'
C 5-24-83
C SUBROUTINE INTELI(XBN,YBN,A,NXY,HIJ)
%include 'implicit'
DIMENSION A(NXY,NXY),HIJ(NXY)

274
LBEM Code Listing

C
%include blk2
%include blk4
%include blk5
C
-------------------------------------------------------------------------------
C
LINEAR BASIS FUNCTION BOUNDARY INTEGRATION ROUTINE
C
THIS ROUTINE ACCOMPLISHES ANALYTICAL INTEGRATION ON A GIVEN ELEMENT
C
FOR A GIVEN BASE NODE POINT. IT HANDLES HIJ INTEGRALS FOR ALL
C
ELEMENTS.
C
IT CALLS GIJ TO DO GIJ TYPE INTEGRATION FOR THE SURFACE OF UNKNOWN
C
FLUX THAT EXISTS IN THE FINITE DOMAIN PROBLEM
C
-------------------------------------------------------------------------------
C
TEST FOR COLLINEAR NODE AND ELEMENT
C
IF SO HIJ=0, GOTO GIJ OR EXIT IF INSULATOR
C
IF(SINGULAR) GOTO 290
C
FOR DEFORMING ELECTRODE ELEMENTS, SKIP TO HIJ CALC
C
(COLLINEARITY TEST IS NOT APPLICABLE)
C
IF (ELTYPE.EQ.1) GOTO 215
C
IF(NODE1-ID) 2,1,1
C
TEST FOR COLLINEAR LHS INSULATOR
1  IF( I.GE.ID.OR.I.EQ.1) RETURN
   GOTO 9
2  IF (NODE1-IC) 4,3,3
C
TEST FOR COLLINEAR TOP SURFACE, EXIT FOR GIJ
3  IF(I.LE.ID.AND.I.GE.IC) GOTO 300
   GOTO 9
4  SINCE ELTYPE <> 1, WE KNOW ELEMENT IS RHS INSULATOR
5  IF(I.LE.IC.AND.I.GE.IB) RETURN
9  CONTINUE
C
C
TEST FOR analytical HIJ
C
GOTO (215,215,215,235,240), ELTYPE
C
NORMAL ELEMENTS - FINITE, INSULATOR, "ELECTRODE", OR TOP
215  DXI= XX1 - XBN
   Dyi= YY1 - YBN
   BB = DY*Dyi + DX*DXI
   BB = BB + BB
   CC = Dyi*Dyi + DXI*DXI
   DD = DY*DXI - DX*Dyi
   B2A = BB/(DX2Y2+DX2Y2)
   ADD = ABS(DD)
   IF (ADD.LE.1.OE-7) RETURN
      ATA = BB/(ADD+ADD)
      ATA = ATAN(ATA + DX2Y2/ADD) - ATAN(ATA)
      ATA = SIGN(ATA,DD)
      B2A = B2A*ATA
      HIJ1 = (DD/(DX2Y2+DX2Y2)) * ALOG(CC/(DX2Y2 + BB + CC))
      HIJ2 = -HIJ1 - B2A
      HIJ1 = HIJ1 + ATA + B2A
   GOTO 245
C
C
ELTYPE=4, INFINITE RHS ELEMENT
235  DXI= XX1 - XBN
   Dyi= YY1 - YBN
   DYY=DY
237  AAA= Dyi/Dyy
   AAA= AAA+AAA
   BBB=(Dyi*Dyi+DXI*DXI)/(DYY*DYY)
   DDD=-ETAO
   DDD2=DDD*DDD
   GGG=DDD*(DDD-AAA)+BBB
   GGG2=GGG*GGG
   GGG2=GGG2*GGG2

275
DEN = SQRT(4.0*BBB - AAA*AAA)
ATA = ATAN(AAA/DEN)
ATA = (PI - (ATA + ATA)) / (GGG2*DEN)
TEMP = DDD*(4.0*BBB - AAA*DDD) - AAA*BBB
EE = AAA - DDD
EE = BBB + BBB - EE
HIJ1 = (EE + TEMP)*ATA
HIJ2 = TEMP*ATA
HIJ1 = HIJ1 + (1.0 + DDD)/(GGG*DDD)
HIJ2 = HIJ2 - 1.0/GGG
TEMP = ALOG(BBB/DDD2)/GGG2
HIJ1 = HIJ1 + (BBB-DDD2 - AAA*(DDD+DDD)) + TEMP
HIJ2 = HIJ2 + (BBB-DDD2)*TEMP
HIJ1 = DDDD2*DI*HIJ1/DYY
HIJ2 = ((1.0-ETAQ)**2)*DI*HIJ2/DYY
IF (ELTYPE .NE. 5) GOTO 245

C REVERSE SIGN SINCE DY/DS WAS CALCULATED REVERSED
HIJ1 = -HIJ1
HIJ2 = -HIJ2
GOTO 245

C ELTYPE = 5, INFINITE LHS - ADJUST MAPPING WITH SIDE EFFECT OF REVERSING DY/DS
240 DIX = XX2*XBN
DY1 = YY2*YBN
DYY = DY
GOTO 237

C ADD UP HIJ, I.NE.J, TO EVENTUALLY CALCULATE HIJ, I=J
C DON'T WORRY ABOUT I.EQ.J BECAUSE OF SINGULARITY TEST
245 HIJ(I) = HIJ(I) + HIJ1 + HIJ2

C ADD INTEGRAL TERMS TO MATRIX A
C DIFFERENT RULE IS USED FOR FINITE DOMAIN AND SPECIAL CORNERS
IF (FINITE) GOTO 265
C
A(I,NODE1) = A(I,NODE1) + HIJ1
A(I,NODE2) = A(I,NODE2) + HIJ2
C
RETURN

C FINITE DOMAIN - POTENTIAL IS SPECIFIED AT TOP, DON'T HAVE MATRIX
C CONTRIBUTION FROM TOP SURFACE AND EXCLUDE TOP CORNERS
265 IF (ELTYPE.EQ.3) GOTO 300
IF (NODE1.NE.ID) A(I,NODE1) = A(I,NODE1) + HIJ1
IF (NODE2.NE.IC) A(I,NODE2) = A(I,NODE2) + HIJ2
RETURN

C ***** FINITE DOMAIN GIJ TYPE INTEGRALS FOR SURFACE OF UNKNOWN FLUX *****
290 IF (ELTYPE.NE.3) RETURN
C
300 CONTINUE
CALL GIJ(I,XBN,YBN,GiJ1,GiJ2)
C
- EXIT - ADD RESULTS
C
A(I,NODE1) = A(I,NODE1) - GiJ1
A(I,NODE2) = A(I,NODE2) - GiJ2
RETURN
END

C INCLUDE 'COMP'
C 1/11/83
SUBROUTINE MATFIXED (X,Y,NXY,A,HIIJ)
INCLUDE implicit
DIMENSION X(NXY),Y(NXY),A(NXY,NXY),HIJ(NXY)
C
INCLUDE blk2
%include blk4

--- SETS UP MATRIX CONTRIBUTIONS FOR FIXED ELEMENTS AND FIXED BASE NODES.

----- OUTTER LOOP FOR EVERY FIXED ELEMENT ------
DO 50 N=NELEC+1, NELMTS - NLOCAL

----- FOR ELEMENT N, CHARACTERIZE FOR INTEGRATION
CALL ELMT(N,X,Y,NXY)

----- NOW LOOP FOR EVERY STATIONARY NODE -------
DO 40 I=1, NDIS
      SINGULAR = I.EQ.NODE1. OR. I.EQ.NODE2
      CALL INTL(I,X(I),Y(I),A,NXY,HIJ)
      CONTINUE
40    CONTINUE

RETURN
END

--- INCLUDE 'COMP'

SUBROUTINE MATNL (X,Y,NXY,A,F)
%include implicit
DIMENSION X(NXY), Y(NXY), A(NXY,NXY), F(NXY)

%include blk2
%include blk4
%include blk5

--- SUPERVISES CALCULATION OF SOURCE TERM FLUX CONTRIBUTIONS
--- COMING FROM ELECTRODE ELEMENTS AND ALL BASE NODES
--- LINEAR BOUNDARY ELEMENT VERSION

--- OUTER LOOP FOR ELECTRODE ELEMENTS ---
DO 5000 N=1, NELEC

--- CHARACTERIZE THIS ELEMENT FOR INTEGRATION
CALL ELMT(N,X,Y,NXY)

--- CALCULATE CONSTANTS NEEDED FOR THE SOURCE BOUNDARY CONDITION ---
--- VALUES OF THE INTEGRAND AT THE ELEMENT ENDPONITS ---
EVX1 = EVAL*XX1
EVX2 = EVAL*XX2
EVY1=EVAL*YY1
EVY2=EVAL*YY2
EEVY1=EXP(-EVY1)
EEVY2=EXP(-EVY2)

IF(FINITE) GOTO 134

GOTO (131,132,133), ICOND

131 Q1 = EEVY1*(-DY*SIN(EVX1)+DX*COS(EVX1))
Q2 = EEVY2*(-DY*SIN(EVX2)+DX*COS(EVX2))
DQ1=0.0
DQ2=0.0
GOTO 138

132 Q1 = -EEVY1*DY*SIN(EVX1)
Q2 = -EEVY2*DY*SIN(EVX2)
DQ1=DX
DQ2=DX
GOTO 138

133 Q1 = EEVY1*DX*COS(EVX1)
Q2 = EEVY2*DX*COS(EVX2)
DQ1=-DY*TAN(EVX1)

277
LBEM Code Listing

DQ2=-DY*TAN(EVX2)
GOTO 138

C

FINITE DOMAIN

134 EX21=EXP(EVY1-EV2HB)
    EX22=EXP(EVY2-EV2HB)
    GOTO (135, 136, 137), I8COND

135 QT1=-DY*SIN(EVX1)*(EEVY1-EX21)+DX*COS(EVX1)*(EEVY1+EX21)
    QT2=-DY*SIN(EVX2)*(EEVY2-EX22)+DX*COS(EVX2)*(EEVY2+EX22)
    DO1=0.0
    DO2=0.0
    GOTO 138

136 QT1=-DY*SIN(EVX1)*(EEVY1-EX21)
    QT2=-DY*SIN(EVX2)*(EEVY2-EX22)
    DO1=DX*(EEVY1+EX21)/(EEVY1-EX21)
    DO2=DX*(EEVY2+EX22)/(EEVY2-EX22)
    GOTO 138

137 QT1=DX*COS(EVX1)*(EEVY1+EX21)
    QT2=DX*COS(EVX2)*(EEVY2+EX22)
    DO1=-DY*TAN(EVX1)
    DO2=-DY*TAN(EVX2)

138 DSR=1.0/SQRT(DX2Y2)
    QT1=QT1*SEVAL*DSR
    QT2=QT2*SEVAL*DSR
    DO1=DO1*EVAL*DSR
    DO2=DO2*EVAL*DSR

C --- LOOP FOR EVERY BASE NODE ---
DO 4000 I=1,NNODES
   SINGULAR= I.EQ.NODE1.OR.I.EQ.NODE2

4000 CONTINUE
5000 CONTINUE
RETURN
END

C

INCLUDE 'COMP'

11/11/83

SUBROUTINE MATVAR (X,Y,NXY, A,HIJ)
%include implicit
DIMENSION X(NXY), Y(NXY), A(NXY,NXY), HIJ(NXY)

%include blk2
%include blk4

--- SETS UP MATRIX CONTRIBUTIONS FOR (ALL BOUNDARY ELEMENTS AND MOVING BASE NODES) AND (MOVING ELEMENTS WITH ALL BASE NODES) ---

**** LINEAR BOUNDARY ELEMENT VERSION ****

--- OUTER LOOP FOR EVERY ELEMENT ---
NFX=NELECT + NLOCAL
NFXE=NELMTS - NLOCAL
DO 5000 N=1,NELMTS

5000 CONTINUE

C --- CALC MAPPING DATA FOR NORMAL QUADRAUTURE PTS
   CALL ELMT(N,X,Y,NXY)

C

ELMOVE INDICATES A MOVING LOCAL REGION ELEMENT

278
ELMOVE=(N.LE.NFX.OR.N.GT.NFXE)

FOR MOVING ELEMENTS WE WILL LOOP ONCE FOR ALL
BASE NODES. FOR OTHER ELEMENTS WE WILL LOOP TWICE. ONCE FOR
LOW NUMBERED BASE NODES (ELECTRODE AND RIGHT LOCAL WALL), AND A
SECOND TIME FOR HIGH NUMBERED NODES (LEFT LOCAL WALL).

PASS2=.FALSE.
ISTA=1
ISTP=NNODES
IF (.NOT.ELMOVE) ISTOP=ICC-1
   DO 4000 I=ISTA,ISTOP
      SINGULAR= I.EQ.NODE1.OR.I.EQ.NODE2
      CALL INTEL (I,X(I),Y(I),A,NXY,HIJ)
   4000 CONTINUE
IF (ELMOVE) GO TO 5000
IF (PASS2) GO TO 5000
ISTA=ID+1
ISTP=NNODES
PASS2=.TRUE.
IF (ISTA.LE.ISTOP) GO TO 3000
5000 CONTINUE
C
C NOW ALL HIJ, J,NE,I, HAVE BEEN FIGURED --- FIGURE
C OUT HIJ(I,I) BY SUMMATION
C
IF(FINITE) GOTO 7000
   DO 6000 I=1,NNODES
      A(I,I)=HIJ(I)
   6000 RETURN
7000 DO 7500 I=1,IC-1
   7500 A(I,I)=HIJ(I)
   DO 8000 I=ID+1,NNODES
   8000 A(I,I)=HIJ(I)
RETURN
END
C
C 1/1/83 INCLUDE 'COMP'
SUBROUTINE REGRID(X,Y,NXY,ERROR)
%include implicit
DIMENSION X(NXY),Y(NXY)
LOGICAL ERROR,EVENL

%include blk1
%include blk2
C
USING THE POSITION OF THE FREELY MOVED ELECTRODE SURFACE,
THIS SUBROUTINE ADJUSTS THE SPACING OF THE NODES IN THE
LOCAL ELEMENT REGION. IF THE ELECTRODE HAS MOVED BEYOND
THE LOCAL REGION, THE STRUCTURE OF THIS PROGRAM IS
INAPPROPRIATE AND ERROR IS SET TRUE.

***** LINEAR BOUNDARY ELEMENT VERSION *****
C
C CLEAR TROUBLE FLAG-----
ERROR=.FALSE.
C----- CORNER NODES ARE INDICATED IN COMMON -----
IB= RIGHHAND SIDE ELECTRODE CORNER
IC= TOP NODE RIGHTSIDE (CORRESPONDING ELEMENT EXTENDS TO INFINITY)
ICC=RIGHHAND SIDE TOP OF LOCAL ELEMENTS
ID=TOP NODE LEFT SIDE (ELEMENT EXTENDS TO INFINITY)
IDD=LEFTSIDE TOP OF LOCAL ELEMENTS
C
C------CHECK FOR GROWTH ABOVE LOCAL AREA ON RHS--------
DELY=Y(IC)-Y(IB)
C IF (DELY.LE.0) GO TO 1000
C
C------ADJUST RHS LOCAL NODE SPACING----------
LBEM Code Listing

IF (IB+1.EQ.ICC) GOTO 12

EVENL=VSLLOCAL.LE.0.0
FACT=DELY/FLOAT(IICC-IB)
IF(.NOT.EVENL)FACTR=DELY/(EXP(VSLLOCAL)-1.0)
DO 10 I=IB+1.IICC-1
IF (EVENL) Y(I)=Y(IB)+FLOAT(I-IB)*FACT
IF(.NOT.EVENL) Y(I)=Y(IB)+(EXP(VSLLOCAL*FLOAT(I-IB)/
2 FLOAT(IICC-IB)))-1.0)*FACTR
10 CONTINUE

C-------------------CHECK FOR GROWTH ABOVE LOCAL REGION ON LHS-------
12 DELY=Y(IDD)-Y(1)
IF (DELY.LE.0.0)GO TO 1000

C----------------ADJUST LHS LOCAL NODE SPACING-------
IF (IDD.EQ.NNODES) RETURN
FACT=DELY/FLOAT(NNODES+1-IDD)
IF(.NOT.EVENL)FACTL=DELY/(EXP(VSLLOCAL)-1.0)
DO 20 I=IDD+1.NNODES
IF (EVENL) Y(I)=Y(IDD)-FLOAT(I-IDD)*FACT
IF(.NOT.EVENL) Y(I)=Y(1)+(EXP(VSLLOCAL*(1.0-FLOAT(I-IDD)/
2 FLOAT(NNODES+1-IDD))))-1.0)*FACTL
20 CONTINUE
RETURN

C=======================================================================
C---------HERE IF PROGRAM IS INVALIDATED BY ELECTRODE GROWTH ---
C-- ABOVE LOCAL REGION---
1000 ERROR=TRUE.
RETURN
END

C
C SGECO, SGESL, SGefa, etc see Dongarra et al. 1979 (LINPACK)
C
C
C INCLUDE 'COMP'
FUNCTION SHAPE(Z)
%include implicit
%include blk1
C
THIS FUNCTION DETERMINES THE INITIAL CATHODE SHAPE BASED
ON THE INTEGER VARIABLE 'TYPE' AND THE FACTOR 'AMP' FOUND
IN THE COMMON BLOCK BLK1. THE FUNCTION SUPPLIES VALUES
OF Y WHEN CALLED WITH VALUES OF X AS ITS ARGUMENT. VALUES
OF X WILL RANGE BETWEEN 0 AND 1.
C
PI=3.14159265358979300
GOTO (10,20,30),TYPE
C
C TYPE=1, COSINE WAVE, SCALED FROM (0,0) TO (1,AMP)
10 SHAPE=AMP*0.5*(1.0-COS(Z*PI))
RETURN
C
C TYPE=2, TRIANGLE WAVE
20 SHAPE=AMP*Z
RETURN
C TYPE=3, COSINE WAVE, OF PERIOD 1
30 SHAPE=AMP*0.5*(1.0-COS(Z*2.0*PI))
RETURN
END
APPENDIX I

TAFEL KINETICS MODEL CODE LISTING

CALLING HIERARCHY

MAIN PROGRAM
  QUAD
  GRIDSTART
  REGRID
  MATFIXED
    ELMT
      PHI
      DPHI
    INTEL
      PHI
      DPHI
  MATVAR
    ELMT
      PHI
      DPHI
    INTEL
      PHI
      DPHI
  MATNL
    ELMT
    NLINTEL
      PHI
      DPHI
  SIMQ
  SURF
  CATH
  STAT
  LIST
    MASSBAL
      PHI
      DPHI
  CPU_TIMER

MODULE NOTES

MAIN PROGRAM
  - reads the input data (file ICDATA)
  - sets miscellaneous parameters
  - writes report files: REPORT, CATHODE, CONVERGENCE
  - controls the time integration
  - stores temporary matrix terms in files AFSAVE, AFCOPY
  - deforms cathode

CATH - subroutine
  - calculates the deposit area and nonuniformity of current
CPU_TIMER - function (PL/1 programming language)
- calls system vclock() function
- returns elapsed virtual cputime (seconds)

DPHI - function
- calculates derivatives of the basis functions

ELMT - subroutine
- sets up quantities needed to calculate integrals on the current element

INTEL - subroutine
- computes integrals on the current element

GRIDSTART - subroutine
- draws initial boundary element mesh

LIST - subroutine
- lists concentrations around the model boundary

MASSBAL - subroutine
- calculates the current imbalance

MATFIXED - subroutine
- computes matrix problem contributions for the boundary elements that are spatially fixed

MATNL - subroutine
- computes matrix terms from the nonlinear boundary condition on the active surface

MATVAR - subroutine
- computes matrix problem contributions for the deforming boundary elements
- computes $H_{ii}$ terms by difference

NLINTEL - subroutine
- computes the cathode boundary condition integrals

PHI - function
- calculates the basis functions

REGRID - subroutine
- adjusts the node spacing in the trench region

QUAD - subroutine
- sets up quadrature related data

SIMQ - subroutine
- solves the linear system of equations based on the Jacobian

STAT - subroutine
Tafel Kinetics Model Listing Notes

- computes statistics on the cathode surface shape

SURF - subroutine
- computes the cathode surface normal and the rates of movement along the axes

SIGNIFICANT COMMON AND INPUT VARIABLES

A(NXY,NXY) - the main matrix, it multiplies the unknowns
CORNER(3) - flags exceptional corners during integration
DELTIME - time step size
DXXI() - derivative, change in global coordinate X w.r. t. $\xi$, at the usual quadrature points
DYYI() - derivative, $dY/d\xi$
ELTYPE - type of current element, see ELMT subroutine
EPSILON - small tolerance for comparison
EXFACT - the polarization parameter
F(NXY) - righthand-side vector, the residuals
FXYN(1B,2) - rates of surface deformation at each cathode node
HDELTZA - location of the top surface
HIJ(NXY) - sum of $H_i$ contributions, $i \neq j$
HLOCAL - top of the deforming region
HTRENCH - height of the trench
IB - rightmost node on the cathode, also, number of cathode nodes
IBB - corner node where trench wall becomes the shelf
IC - rightmost node along domain top
ICC - node where resist shelf meets electrolyte
ID - leftmost node along domain top (finite domain)
IDO - node on LHS at level of trench top
IBCOND - boundary condition choice (1,2, or 3)
ICHICE(5) - report control, see input data file
INFINITE - true for the infinite domain problem
ITCOUNT - count of completed time step iterations
NBULK - number of elements along the symmetry walls
NCATH - number of elements along the cathode
NELMTS - total number of elements
NDELTA - number of elements along the top boundary
NLN - number of logarithmic quadrature points
NODE(3) - node indexes on the current element
NQ - number of Gauss-Legendre quadrature points
NTRENCH - number of elements along the trench wall and the midtrench symmetry line up to the trench top
NTRSTEP - number of elements along the horizontal trench top
NXY - total nodes
NWRITE - cathode location is reported every NWRITE time steps
PLN(NLN) - logarithmic quadrature points
POLAR - the polarization parameter when POTDROP=0.0
Tafel Kinetics Model Listing Notes

POTDROP  - applied potential - keep at 0.0 and use POLAR
PT(NQ)   - the quadrature points
SINGULAR - true when current base node is on current element
SOLN(NXY) - concentration and current densities solution
STOPTIME - integration stop time
TTIME    - simulated integration time
VSFACT   - vertical spacing factor 0.0 for uniform spacing
WLN(NLN) - logarithmic quadrature points
WT(NQ)   - quadrature weights
WTOTAL   - total symmetry width
WTRENCH  - trench width, always 1.0
X(NXY),Y(NXY) - node locations, global coordinates
XN(IB)   - save X coordinates of the cathode
XQ(NQ)   - abcissae of quadrature points on the current element
XX(3),YY(3) - points on current element
YN(IB)   - saved Y coordinates of the cathode
YQ(NQ)   - ordinates of quadrature points on the current element
INCLUDED FILES

C implicit - specifies precision
IMPLICIT INTEGER (I-N), REAL (A-H.O-Z)

COMMON BLOCKS

COMMON/BLK1/HTRENCH,HDELTA,WTERCH,WTOTAL,VSFAC T

COMMON/BLK2/NCATH,NTRENCH,NTRTOP,NBULK,NDELTA,NELMTS
2,IB,IC,ID,IBB,ICC,IDD

COMMON/BLK3/NQ,NLN,WT(20),PT(20),XQ(20),YQ(20)
2,DXDXY,DXDY(20),WLN(7),PLN(7)

COMMON/BLK4/ELTYPE,SINGULAR,LEFT,CENTER,RIGHT
2,CORNER(3),NODE(3),XX(3),YY(3)
INTEGER ELTYPE
LOGICAL SINGULAR,LEFT,CENTER,RIGHT,CORNER

COMMON/BLK5/EXFACT,CEXPO,DCEXPO

INPUT DATA FILE (free format)
0.2,5.0,1.0,2.0,0.0,;HTRENCH,HDELTA,WTERCH,WTOTAL,VSFAC T
1.0,4.4,8.1,;NCATH,NTRENCH,NTRTOP,NBULK,NDELTA (150 MAX)
0.0,0.0,1.0E-06,12,;DELTIME,STOPTIME,TOLERANCE,ITMAX
1,10,4,+1,-2,-3,-4,+5,;NWRITE,NLN,ICHICE(5)
0.06,58.2,0.0,1.0,;CALPHA,GAMMA,NELECTRON,POTDROP,POLAR
IF ICHOIE(1).EQ.O NO INITIAL REPORT
IF ICHOIE(1)>O GET INITIAL REPORT
IF ICHOIE(2)>O GET FINAL REPORT
IF ICHOIE(3)>O GET FINAL STATISTICAL ANALYSIS
IF ICHOIE(5) >O ONLY GET T=O SOLN

235
Tafel Kinetics Model Code Listing

%global ans177 card round auto_zero
%include implicit
C 4-10-83
C
C MAIN PROGRAM INTERMEDIATE CURRENT BOUNDARY ELEMENT ELECTRODEPOSITION
C INTO PARALLEL MICRORENSHES MODEL
C
C VERSION 3.0
C This version has large dimensions and uses file storage so that big
C problems may be tackled. It also incorporates CPU timing
C
C PARAMETER (IXY=300, IXXYY=IXY*IXY, ICATH=81)
C
C COMMON STATEMENTS ARE GOTTEN FROM INCLUDE FILES
C
C %include blkt1
C %include blkt2
C %include blkt3
C %include blkt4
C %include blkt5
C
C DIMENSION X(IXY), Y(IXY), A(IXXY), F(IXY), SOLN(IXY), HIJ(IXY)
C Larger Problem is possible by putting arrays in common
C COMMON/BK6/A(IXXY), X(IXY), Y(IXY), F(IXY), SOLN(IXY), HIJ(IXY)
C DIMENSION XN(ICATH), YN(ICATH), FXYN(ICATH, 2), ICHOICE(9)
C DIMENSION XNEW(ICATH), YNEW(ICATH)
C DIMENSION ASAVE(IXXY), FSAVE(IXY), HIJSAVE(IXY)
C DIMENSION ACOPY(IXXY), FCOPY(IXY)
C LOGICAL REPORT, EUPRED, ERROR
C EXTERNAL data_time (descriptors)
C EXTERNAL clock (descriptors)
C CHARACTER*24 TDSTRING
C EXTERNAL cpu_time (descriptors)
C REAL CPUSTART, CPUTE
C
C--------------------- INITIALIZE -------------------------------
C
C OPEN FILES------
C OPEN(2, MODE="IN", FORM="FORMATTED", FILE="idata")
C OPEN(3, MODE="INOUT", FORM="FORMATTED", FILE="cathode")
C OPEN(4, MODE="INOUT", FORM="FORMATTED", FILE="report")
C OPEN(20, MODE="INOUT", FORM="UNFORMATTED", FILE="afsave")
C OPEN(21, MODE="INOUT", FORM="UNFORMATTED", FILE="fscopy")
C OPEN(22, MODE="INOUT", FORM="FORMATTED", FILE="convergence")
C
C----READ PROBLEM PARAMETERS-----
C 1ST ROW GEOMETRICAL DATA
C READ (2, 600) HTRENCH, HDELTA, WTRENCH, WTOTAL, VSFAC
C 600 FORMAT(V)
C 2ND ROW ELEMENT LAYOUT DISTRIBUTION
C READ (2, 600) NCATH, NTRENCH, NTRTOP, NBULK, NDELTA
C 3RD ROW, INTEGRATION CONTROL
C READ (2, 600) DELTIME, STOPTIME, TOLERANCE, ITMAX
C 4TH ROW, QUADRATURE AND MISCELLANEOUS
C READ (2, 600) NWRITE, NOQ, NLN, ICHOICE
C 5TH ROW ELECTRODE KINETIC PARAMETERS
C READ (2, 600) CALPHA, GAMMA, NELECTRON, POTDROP, POLAR
C
C---------------------SET MISCELLANEOUS PARAMETERS-------
C
C TOLERANCE FOR COMPARISON
C EPSILON = 1.0E-06
C
C NELMNTS = NCATH + 2*NTRENCH + NTRTOP + 2*NBULK + NDELTA
C IB, IBB, IC, ICC, ID, IDD IDENTIFY CORNER NODE NUMBERS
C IB = 2*NCATH + 1
C IBB = IB + 2*NTRENCH
C ICC = IBB + 2*NTRTOP
C IC = ICC + 2*NBULK
C ID = IC + 2*NDELTA
C Tafel Kinetics Model Code Listing

C
IDD=ID+2*NBUlk
NXY=2*NELMTS
NXYY=NXY*NXY

C CORNER(2)=.FALSE.

C EXFACT=PREAD*EXP(-CALPHA* POTDROP)
CEXPO=GMMA+CALPHA/FLOAT(NELECTRON)
DCEXPO=CEXPO - 1.0

C REPORT=.TRUE.
IF (ICHICE(1).LE.0) REPORT=.FALSE.
TTIME=0.0
ITCOUNT=0

C C INITIALIZE SOLN
DO 10 I=1,NXY
10 SOLN(I)=0.5

C C-------SET UP QUADRATURE POINTS AND WEIGHTS-------

CALL QUAD

C C----------------------WRITE REPORT INFO TO REPORT FILE----------------------

CALL date_time (clock(),TSTRING)
WRITE (4,700) TSTRING

700 FORMAT(/'******************************','3X,A24,3X,'******************************')

C WRITE (4,710) NELMTS,HTRENCE,HDELTA,WTRENCE,WTOTAL,VSFACT

710 FORMAT("UNSTEADY, TWO-DIMENSIONAL INTERMEDIATE CURRENT",
2 "ELECTRODEPOSITION MODEL",
2 "SOLUTION BY BOUNDARY ELEMENT METHODS AND EULER PREDICTOR",
3 "TRAPEZOID CORRECTOR"/"TIME INTEGRATION. Version 3.0"
3 "BOUNDARY ELEMENTS, <=36 BIT PRECISION>"
4 "/GEOMETRIC PARAMETERS",
5 "/HTRENCE - DEPTH OF TRENCH .G12.5/
6 "/HDELTA - DIFFUSION LAYER THICKNESS .G12.5/
7 "/WTRENCE - CATHODE HALF WIDTH .G12.5/
8 "/WTOTAL - MIDTHRENCH TO SYMMETRY PLANE SEPARATION .G12.5/
9 "/VSFACT - VERTICAL SPACING FACTOR, BULK REGION .G12.5/

WRITE (4,720) NCATH,HTRENCE,HDELTA,WTRENCE,WTOTAL,VSFACT

720 FORMAT("DISTRIBUTION OF ELEMENTS"
3 "/NCATH - # OF CATHODE ELEMENTS .I4/
3 "/HTRENCE - TRENCH VERTICAL WALL .I4/
4 "/HDELTA - INSULATOR AT TRENCH TOP LEVEL .I4/
5 "/WTRENCE - EITHER SIDE BULK REGION .I4/
6 "/WTOTAL - ALONG DIFFUSION LAYER TOP .I4/

WRITE (4,730) DELTIME,STOPTIME

730 FORMAT("INTEGRATION PARAMETERS"
2 "/DELTIME - TIME INCREMENT .G12.5/
3 "/STOPTIME - FINAL STOPPING TIME .G12.5/

735 FORMAT("PROGRAM CONTROL PARAMETERS"
2 "/NUMBER OF QUADRATURE POINTS USED ON EACH ELEMENT"
3 4X,"NO",10X,"NORMAL INTEGRATION",16X.14
4 4X,"NO",10X,"LOG SINGULARITY",16X.14
5 "TOLERANCE FOR NEWTON-RAPHSON CONVERGENCE(Infinity Norm)"
6 4X,"TOLERANCE = ".33X,G12.5/
4 "/NWRITE - CATHODE REPORT TIME STEP INTERVAL ",4X,I4/
6 "/ICHICE - REPORT CONTROL ",5I4/

WRITE (4,737) CALPHA,GMMA,NELECTRON,POTDROP,POLAR

737 FORMAT("ELECTRODE KINETIC PARAMETERS"
3 "CALPHA",10X,"CATHODIC TRANSFER COEFF. ",4X,F10.4/
4 "GMMA",11X,"CONC. EXPONENT BUT-VLMR ",4X,F10.4/
5 "NELECTRON",7X,"# OF ELECTRONS IN RXN ",10X,14/
5 "POTDROP",9X,"-POTENTIAL IN TRENCH REGION",7X,G12.5/
6 "POLAR",11X,"-POLARIZATION FACTOR",10X,G12.5/

C WHEN POTDROP=0, POLAR=POLARIZATION PARAMETER
C C--------ALLOW FOR PERIODIC CATHODE LOCATION REPORTS
AA=DELTIME*FLOAT(NWRITE)

287
WRITE(3,740) AA,TOSTRING
740 FORMAT ("INTERMEDIATE CURRENT CATHODE GROWTH, TIME INC.= *,F6.3/
  2 "PROGRAM RUN ",A24/
  3 "PROFILES/*XMIN 0.0*/"SYM 200.0*/"END")
C
C----- START CPU TIMING
   CALL cpu_time(CPUTIME)
C----------ESTABLISH FIXED ELEMENT NODE LOCATIONS ---------
   CALL GRIDSTART(X,Y,NXY)
C----------CALC MATRIX CONTRIBUTIONS FROM FIXED NODES ------
   CALL MATFIXED(X,Y,NXY,A,F,HIJ)
C----- SAVE FIXED RESULTS AS A BASE FOR EACH TIME ITERATION -----
   WRITE (20) (A(I),I=1,NXXYY),(F(I),I=1,NXY),(H(I),I=1,NXY)
   DO 20 I=1,NXXYY
   20   ASAVE(I)=A(I)
   C   DO 22 I=1,NXY
   C   FSAVE(I)=F(I)
   C   HIUSAVE(I)=H(I)
   C
   C------------------------------------------------------------------ END INITIALIZATION -------------------------------
   C
   C------------------------------------------------------------------ MAIN INTEGRATION LOOP ENTRY POINT ----------------
   100  CONTINUE
   C
C-----SET FLAG FOR EUER PREDICTION -------
   EUPRED=.TRUE.
C-----SAVE CATHODE POSITION IN XN,YN -----
   DO 109 I=1,18
      XN(I)=X(I)
      YN(I)=Y(I)
   109   C----110 CONTINUE IS ENTRY POINT AFTER EUER PREDICTION----
   110   CONTINUE
   C
C--- SOLVE FOR FLUX
C REINITIALIZE MATRIX
   IF (ITCOUNT.EQ.0.AND.EUPRED) GOTO 112
   REWIND 20
   READ(20) (A(I),I=1,NXXYY),(F(I),I=1,NXY),(H(I),I=1,NXY)
   C   DO 116 I=1,NXXYY
   C116   A(I)=ASAVE(I)
   C   DO 117 I=1,NXY
   C   F(I)=FSAVE(I)
   C117   HI(I)=HIUSAVE(I)
   112  CONTINUE
C-------ADD CONTRIBUTION FROM MOVING NODES -----------
   CALL MATVAR(X,Y,NXY,A,F,HIJ)
   C
C--- SAVE A.F FOR INITIALIZING NEWTON ITERATIONS ----
C---- A. F REFLECT GEOMETRY AT THIS TIME ----
   C
   REWIND 21
   WRITE (21) (A(I),I=1,NXXYY),(F(I),I=1,NXY)
   C   DO 121 I=1,NXXYY
   C121   ACOPY(I)=A(I)
   C   DO 122 I=1,NXY
   C122   FCOPY(I)=F(I)
   C -- ADD IN SOLUTION DEPENDENT NONLINEAR TERMS TO FORM RESIDUALS
   C -- AND JACOBIAN ----------- BEGIN NEWTON ITERATION <---
   CNEWTON=0
   GOTO 120
   115  CONTINUE
   C-- READ FRESH A.F IF NOT FIRST NEWTON ITERATION --
   REWIND 21
   READ (21) (A(I),I=1,NXXYY),(F(I),I=1,NXY)
   C   DO 118 I=1,NXXYY
   C118   A(I)=ACOPY(I)
   C   DO 119 I=1,NXY
   C119
C119  F(I)=FCOPY(I)
120  CONTINUE
C------ BUILD RESIDUAL AND JACOBIAN FROM A,F, AND NEW TERMS ----
C  RESIDUAL= A*SOLN + F -(NONLINEAR VECTOR) ----
C  JACOBIAN= A - (RELATED NONLINEAR MATRIX) ----
C  RESIDUAL=F, JACOBIAN=A
C
N=0
DO 125 J=1,NXY
DO 125 I=1,NXY
N=N+1
125   F(I)=F(I) + A(N)*SOLN(J)
C
C  ADD NONLINEAR TERMS TO JACOBIAN AND RESIDUAL
CALL MATNL(X,Y,NXY,A,F,SOLN)
C
C  SOLVE  (JACOB)*((DELTA)=$(RESID)
C
CALL SIMO (A,F,NXY,KERR)
IF(KERR.EQ.1) WRITE (4,765) TTIME
765  FORMAT(12*(SINGULAR MATRIX AT TIME = "G10.3/)
IF (KERR.EQ.1) GOTO 1000
C
C  ADJUST SOLN VECTOR WHILE TAKING CONVERGENCE NORM
DELNORM=0.0
DO 127 I=1,NXY
   IF(ABS(F(I)).GT.DEVNORM)DELNORM=ABS(F(I))
127 CONTINUE
C  ABSOLUTE VALUE IS TAKEN SINCE NEAR LIMITING CURRENT, SOLUTION CAN OVERSHOOT
C  AND SMALL NEGATIVE CONCENTRATION VALUES WILL CAUSE EXPONETNATION ERROR
SOLN(I)=ABS(SOLN(I)-F(I))
127 CONTINUE
C
C  INCREASE IN NEWTON, DEL NORM
C  EXIT IF SUCCESSFULLY CONVERGED
C
1266 WRITE (22,766) INEWTON,DELNORM
766  FORMAT(12*(NEWTON=-",I4," DELNORM=",E12.5)
C
IF(DEVNORM.LE.TOLERANCE)GOTO 129
C  RETURN FOR MORE NEWTON ITERATION
IF(INEWTON.LT.ITMAX)GOTO 115
C  HERE IF FAILED CONVERGENCE
WRITE(4,767)
767  FORMAT("MAXIMUM ITERATIONS EXCEEDED")
GOTO 900
C
129 CONTINUE
C
C------ ON FIRST PASS REPORT THE SOLUTION AND NODAL LAYOUT------
IF (REPORT) CALL LIST(X,Y,NXY,SOLN)
C
C  DEBUG------EXIT
IF(ICHICE(5).GT.O)GOTO 1000
C
REPORT=.FALSE.
C
C-------- EULER PREDICTION --------
IF (.NOT.EUPRED) GO TO 150
DO 130 I=1,IB
   CALL SURF(I,X,Y,NXY,DXDT,DYDT,SOLN)
   FXYN(I,1)=DXDT
   FXYN(I,2)=DYDT
130 CONTINUE
C  PREDICT NEW CATHODE POSITION:
DO 135 I=1,IB
   Y(I)=Y(I) + DELTIME*DYDT
   IF(I.NE.1.AND.I.NE.IB) X(I)=X(I) + DELTIME*DXDT
135 CONTINUE
C
C  READJUST TRENCH AREA ELEMENTS
CALL REGRID(X,Y,NXY,XNEW,YNEW,ICATH,ERROR)

239
Tafel Kinetics Model Code Listing

IF (ERROR) GOTO 900
C------ NOW FIND FLUX WITH THIS POSITION ----
EUPRED=.FALSE.
GO TO 110
C
C------------ HERE FOR TRAPEZOID CORRECTOR -------------
150 CONTINUE
C
C CALCULATE NEW CATHODE MOVEMENT RATE AND AVERAGE IT WITH EULER
C PREDICTED RATE
DO 160 I=1,1B
CALL SURF (I,X,Y,XY,XD,DXD,DXDT,SOLN)
FXYN(I,1)=FXYN(I,1) + DxD
FXYN(I,2)=FXYN(I,2) + DXDT
160 CONTINUE
C
MOVE THE CATHODE
DO 165 I=1,1B
Y(I)=YN(I) + DELTIME*FXYN(I,2)/2.0
IF (I.NE.1.AND.I.NE.1B) X(I)=XN(I) + DELTIME*FXYN(I,1)/2.0
165 CONTINUE
C
READJUST ELEMENTS
CALL REGRID(X,Y,XY,XNEW,YNEW,ICATH,ERROR)
IF(ERROR) GOTO 900
C
C------------- THIS COMPLETES AN INTEGRATION CYCLE -------------
ITCOUNT=ITCOUNT + 1
TTIME=TTIME + DELTIME
C
C WRITE CATHODE POSITION EVERY NWRITE CYCLES
C
C IF (MOD(ITCOUNT,NWRITE).EQ.0) CALL CATH(X,Y,XY,3,TTIME)
C
C------ CHECK TIME AGAINST STOP TIME
C IF (TTIME.GE.STOPTIME-EPSILON) GO TO 1000
C
C RETURN FOR MORE INTEGRATION
C
GO TO 100
C
C ERROR RETURN
900 WRITE(4,781) TTIME
781 FORMAT("ERROR RETURN, SIML. TIME= ",F10.4)
C
C***********************************************
C*********** HERE AT END ***************
1000 CONTINUE
C
C CONVERT FINAL CATHODE POSITION TO EVENLY SPACED ABCISSA
CALL REGRID(X,Y,XY,XNEW,YNEW,ICATH,ERROR)
C
C CALL FOR STATISTICAL ANALYSIS OF FINAL CATHODE SHAPE
C
IF (ICHOICE(3).GT.0) CALL STAT (X,Y,XY)
C
C FINAL REPORT ON LAYOUT AND SOLN
C
IF (ICHOICE(2).GT.0) WRITE (4,785) TTIME
785 FORMAT (/*FINAL NODE LAYOUT AND PRIOR SOLN. TIME= ",G12.5/) IF (ICHOICE(2).GT.0) CALL LIST(X,Y,XY,SOLN)
C CALL date_time(clock(),TDSTRING)
CALL CPU_TIMER(CPUTIME)
CPUTIME=CPUTIME-CPUTIME
WRITE (4,780) TDSTRING,CPUTIME
WRITE (6,780) TDSTRING,CPUTIME
780 FORMAT(/" ALL DONE AT ",A24," Virtual CPU Seconds= ",F12.4 )
END
Tafel Kinetics Model Code Listing

C INCLUDE 'COMP'
SUBROUTINE CATH(X,Y,NXY,IUNIT,TTIME)

INCLUDE implicit
DIMENSION X(NXY),Y(NXY)

INCLUDE d1k2

C USED PERIODICALLY TO REPORT CATHODE DEPOSIT PROFILE, TOTAL AREA,
C NONUNIFORMITY OF DEPOSIT, AND CURRENT SIMULATION TIME

C
C FIND AREA OF DEPOSIT BY SIMPSON'S RULE
C -ABCissa IS EVENLY SPAcED, GROWTH IS PARABOLIC ON
C EACH ELEMENT
C
AREA=0.0
DO 100 I=1,NCATH
AREA=AREA+ 2.0*Y(2*I) + Y(2*I + 1)
100 CONTINUE
AREA=(2.0*AREA +Y(1) -Y(IB))*(X(IB)-X(1))/(6.0*NCATH)

C SCALE AREA BY AREA OF TRENCH
C -TRENCH WIDTH ASSUMED 1
AREA=AREA/Y(IB+2*NTRENCH)

C
C FIND NONUNIFORMITY OF CURRENT PROFILE
YMIN=Y(1)
YMAX=Y(1)
DO 200 I=2,IB
IF (Y(I).GT.YMAX) YMAX=Y(I)
IF (Y(I).LT.YMIN) YMIN=Y(I)
200 CONTINUE
C
IF (YMAX.NE.0.0) YDIFF=(YMAX-YMIN)/YMAX

C WRITE DATA IN DATA GENERAL PLOT READY FORMAT
WRITE(IUNIT,2000) AREA,YDIFF,TTIME
2000 FORMAT(F8.5,",",F8.5,",",AREA,YDIFF,TTIME=".F10.5"

C WRITE (IUNIT,2010) (X(I),Y(I),I=1,IB)
2010 FORMAT (F10.5,",",F10.5"

RETURN
END

C
C C
C
C%include dphi
FUNCTION DPHI(N,Z)

%include implicit
GO TO (1,2,3),N
1 DPHI=Z-0.5
RETURN
2 DPHI=2.0*Z
RETURN
3 DPHI=Z+0.5
RETURN
END

C
C C
C
C%include elmt
SUBROUTINE ELMT(N,X,Y,NXY)

%include implicit
DIMENSION X(NXY),Y(NXY)

291
%include blk2
%include blk3
%include blk4

C
C DD 10 M=1,3
NODE(M)=2*N + M - 2
C LAST NODE HAS INDEX 1
IF (N.EQ.NELMTS.AND.M.EQ.3) NODE(3)=1
XX(M)=X(NODE(M))
YY(M)=Y(NODE(M))
10 CONTINUE
C----- SET UP VARIABLES TO DESCRIBE ELEMENT ----- 
C -- ELTYPE ----1=CATHODE,2=INSUL,3=DELTA
ELTYPE=2
IF(N.EQ.NCATH)ELTYPE=1
IF(NODE(2).LT.ID.AND.NODE(2).GT.IC) ELTYPE=3
C WE NEED TO IDENTIFY THE TWO CORNER NODES IC AND ID
C WHEN THEY APPEAR IN INSULATOR ELEMENTS --- THEY ARE EXCEPTIONAL
C IN THAT THEY ARE NODES WHERE THE FLUX IS CALCULATED UNLIKE
C THE OTHER INSULATOR NODES
C CORNER(2)=.FALSE. IS SET IN MAIN PROGRAM
C CORNER(1)=NODE(1).EQ.ID
C CORNER(3)=NODE(3).EQ.IC
C----- CALCULATE MAPPING INFO FOR THIS ELEMENT ----
C THESE RESULTS ARE FOR THE NORMAL QUADRATURE POINTS THAT WILL BE
C USED MOST OF THE TIME
DO 100 IQ=1,NQ
XI=PT(IQ)
X0(IQ)=0.0
Y0(IQ)=0.0
DXDXI(IQ)=0.0
DYDXI(IQ)=0.0
20 CONTINUE
100 CONTINUE
C
C
C%include gridstart
SUBROUTINE GRIDSTART (X,Y,NXY)
%include implicit
C------------------------------------------------------------------------------------------------
C  DRAWING THE INITIAL NODE LAYOUT FOR THE L SHAPED PARALLEL TRENCH
C  REGIONS, USING QUADRATIC BOUNDARY ELEMENTS.
C
C  THE COMMON VARIABLE VSFACT IS USED TO CONTROL VERTICAL
C  NODE SPACING OUTSIDE OF THE TRENCH REGION.
C
C  THE NUMBER OF ELEMENTS ON EACH PORTION OF THE BOUNDARY IS GIVEN IN
C  THE COMMON VARIABLES NCATH,NTRENCH,NTROP,NBULK,NDELTA
C------------------------------------------------------------------------------------------------
C DIMENSION X(NXY),Y(NXY)
C
C%include blk1
%include blk2
C
C LOGICAL EVEN
EVEN=(VSFACT.EQ.O.O)
C---------LOCATE CATHODE NODES---------
DO 20 I=1,IB
X(I)=FLOAT(I-1)*WTRENCH/FLOAT(IB-1)
Y(I)=0.0
20 CONTINUE
C-----------------TRENCH VERTICAL WALL-----------------
DO 30 I=IBB+1,IBB
X(I)=WTRENCH
Y(I)=FLOAT(I-IBB)*HTRENCH/FLOAT(IBB-IB)
30 CONTINUE
C-----------------RHS TRENCH TOP LEVEL ----------------
DO 40 I=IBB+1,ICC
X(I)=WTRENCH+FLOAT(I-IBB)*(WTOTAL-WTRENCH)/FLOAT(ICC-IBB)
40 CONTINUE
C------------------RIGHT HAND SIDE ABOVE TRENCH TOP----
IF (.NOT.EVEN) FACT=(HDELTA-HTRENCH)/(EXP(VSFAC)-1.0)
DO 50 I=ICC+2,IC,2
X(I-1)=WTOTAL
X(I)=WTOTAL
IF (EVEN) Y(I)=HTRENCH+FLOAT(I-ICC)*(HDELTA-HTRENCH)/
2 FLOAT(IC-ICC)
IF (.NOT.EVEN) Y(I)=HTRENCH+(EXP(VSFAC)*FLOAT(I-ICC)/
2 FLOAT(IC-ICC))-1.0)*FACT
Y(I-1)=(Y(I)+Y(I-2))/2.0
50 CONTINUE
C----------------BOUNDARY LAYER NODES-----------------
DO 60 I=IC+1,ID
X(I)=WTOTAL FLOAT(I-IC)*WTOTAL/FLOAT(ID-IC)
60 CONTINUE
Y(I)=HDELTA
C-------------------LEFT-HAND SIDE NODES---------------
DO 70 I=ID+2,IDD,2
X(I-1)=0.0
X(I)=0.0
IF (EVEN) Y(I)=HDELTA- FLOAT(I-ID)*(HDELTA-HTRENCH)/
2 FLOAT(IDD-ID)
IF (.NOT.EVEN) Y(I)=HTRENCH+(EXP(VSFAC)+(1.0-FLOAT(I-ID))/
2 FLOAT(IDD-ID))-1.0)*FACT
Y(I-1)=(Y(I)+Y(I-2))/2.0
70 CONTINUE
C-----------------LEFT-HAND SIDE TRENCH REGION-------
DO 80 I=IDD+1,2*NELMTS
X(I)=0.0
80 Y(I)=HTRENCH- FLOAT(I-IDD)*HTRENCH/FLOAT(2*NELMTS+1-IDD)
RETURN
END

C%include intel
SUBROUTINE INTEL(I,XX*,YYY,A,F,NXY,HJ)
%include implicit
dimension A(NXY,NXY),F(NXY),HHH(3),GIJ(3),HIJ(NXY)
%include blk2
%include blk3
%include blk4
C
C THIS ROUTINE ACCOMPLISHES NUMERICAL INTEGRATION ON A GIVEN ELEMENT
C FOR A GIVEN BASE NODE POINT. IT CAN HANDLE ALL ELEMENTS AND ALL
C INTEGRALS, WHETHER THEY CONTAIN SINGULAR POINTS OR NOT.
C
C FOR DEFORMING CATHODE ELEMENTS, SKIP TO NUMERICAL HIJ
C (COLLINEARITY TEST IS NOT APPLICABLE)
C if (ELTYPE,EQ.1) goto 205
C
C TEST FOR COLLINEAR NODE AND ELEMENT
C if so HIJ=0, andSkip TO GIJ
C
293
IF(SINGULAR) GOTO 300
IF(NODE(1)-ID) 2,1,1
1  IF( I.GE.ID.OR.I.EQ.1) GOTO 300
   GOTO 9
2  IF(NODE(1)-IC) 4,3,3
3  IF(I.LE.ID.AND.I.GE.IC) GOTO 300
   GOTO 9
4  IF(NODE(1)-ICC) 6,5,5
5  IF(I.GE.ICC.AND.I.LE.IC) GOTO 300
   GOTO 9
6  IF(NODE(1)-IBB) 8,7,7
7  IF(I.GE.IBB.AND.I.LE.ICC) GOTO 300
   GOTO 9
8  IF(I.GE.IBB.AND.I.LE.IBB) GOTO 300
9  CONTINUE
C
C NUMERICAL HIJ
C ZERO INTEGRAL TERM
205   DO 210 K=1,3
210   HHH(K)=0.0
C LOOP FOR EVERY QUADRATURE PT
   DO 250 IQ=1,NQ
      XI=PT(IQ)
      DY=YQ(IQ)-YYY
      DX=XQ(IQ)-XXX
      R2=DY*DY + DX*DX
      HHH=WT(IQ)*(-DY*DXXI(IQ) + DX*DDXI(IQ))/R2
      DO 230 K=1,3
         HHH(K)=HHH(K) + HHH*PHI(K,XI)
      230 CONTINUE
C ADD INTEGRAL TERMS TO MATRIX A
   DO 260 K=1,3
      IF(I.EQ.NODE(K)) GOTO 260
      HIJ(I)+HIJ(K) = HHH(K)
      IF((ELTYPE.EQ.3).OR.CORNER(K)) GOTO 255
C "ELSE" INSULATOR, CATHODE NODE WHERE C IS NOT KNOWN
      A(I,NODE(K))=A(I,NODE(K)) + HHH(K)
   260 CONTINUE
C
C CALC GIJ NUMERICALLY FOR DELTA ELEMENTS
C
C 300 CONTINUE
C IF(ELTYPE.NE.3) RETURN
C
C ZERO GIJ INTEGRAL TERM
   DO 305 K=1,3
305   GIJ(K)=0.0
C IF SINGULAR GOTO SPECIAL CODE
   IF(SINGULAR) GOTO 450
C-----------LOOP FOR EACH NORMAL QUADRATURE POINT-----------
C
   DO 400 IQ=1,NQ
      XI=PT(IQ)
      R2=(YQ(IQ)-YYY)**2 + (XQ(IQ)-XXX)**2
      DS=SQR(DXX(IQ)**2 + DDX(IQ)**2)
      GG=WT(IQ)*DS*ALOG(R2)*0.5
      DO 310 K=1,3
         GIJ(K)=GIJ(K) + GG*PHI(K,XI)
      310 CONTINUE
   DO 410 K=1,3
410   A(I,NODE(K))=A(I,NODE(K)) - GIJ(K)
C
RETURN

C

C-----------------FROM HERE ON CODE FOR SINGULAR INTEGRALS, GIJ CONTRIBUTIONS--------

C

450 CONTINUE
C
FIRST LOOP FOR NONSINGULAR CONTRIBUTION OF REMAPPED SINGULAR INTEGRALS
C
(GIJ HAS BEEN ZERODED

C

DO 600 IQ=1,NQ
X=PT(IQ)
R2=(XQ(IQ) - XXX)**2 + (YQ(IQ) - YYY)**2
IF (LEFT) R2=4.0*R2/((XI+1.0)**2)
IF (RIGHT) R2=4.0*R2/((1.0 -XI)**2)
IF (CENTER) R2=R2/(XI**2)
GG=W(IQ)*SQRT(DXDI(IQ)**2 + DYDI(IQ)**2) * 0.5*ALDG(R2)
DO 500 K=1,3
GIJ(K)=GIJ(K) + GG*PHI(K,XI)

500 CONTINUE

600 CONTINUE

C
GIJ WILL BE ADDED TO BELOW BEFORE BEING ADDED INTO MATRIX A

C

C
IF (CENTER) GOTO 1500

C

C----HERE FOR LEFT OR RIGHT NODE SINGULARITY, LOOP FOR LOG QUAD. PTS--
DO 1100 IQ=1,NLN
IF (LEFT) XI=2.0*PLN(IQ) - 1.0
IF (RIGHT) XI=1.0 - 2.0*PLN(IQ)
DXL=0.0
DYL=0.0
DO 1010 K=1,3
DXL=DXL + DPHI(K,XI)*XX(K)
DYL=DYL + DPHI(K,XI)*YY(K)
1010 CONTINUE

THE LOG R TERM IS BURIED IN THE QUADRATURE PTS AND WEIGHTS--
GG=WLN(IQ)*2.0*SQRT(DXL**2 + DYL**2)
DO 1050 K=1,3
1050 GIJ(K)=GIJ(K) + GG*PHI(K,XI)
1100 CONTINUE

C

DO 1110 K=1,3
1110 A(I,NODE(K))=A(I,NODE(K)) - GIJ(K)
C

RETURN

C

C------HERE FOR CENTER NODE SINGULARITY, NOT INSULATOR ------
C
BREAK INTO 2 INTEGRALS, LEFT AND RIGHT OF SINGULARITY
1500 DO 1700 KK=1,2

C---LOOP FOR EACH QUADRATURE PT---
DO 1600 IQ=1,NLN
XI=PLN(IQ)
DD(KK.EQ.1) XI=-XI
DXL=0.0
DYL=0.0
DO 1510 K=1,3
DXL=DXL + DPHI(K,XI)*XX(K)
DYL=DYL + DPHI(K,XI)*YY(K)
1510 CONTINUE

GG=WLN(IQ)*SQRT(DXL**2 + DYL**2)
DO 1550 K=1,3
1550 GIJ(K)=GIJ(K) + GG*PHI(K,XI)
1600 CONTINUE

1700 CONTINUE
C

DD 1710 K=1,3
1710 A(I,NODE(K))=A(I,NODE(K)) - GIJ(K)
C

RETURN

295
Tafel Kinetics Model Code Listing

END

C

C

C

C

C%include list
SUBROUTINE LIST (X,Y,NXY,SOLN)
%include implicit
DIMENSION X(NXY),Y(NXY),SOLN(NXY)

C
%include blk1
%include blk2
%include blk5

C

C FIND MASS BALANCE BY SURFACE INTEGRATIONS
CALL MABBAL (X,Y,NXY,SOLN,DELFLUX,CATFLUX)
WRITE (4,660)
650 FORMAT ('/"NODAL LOCATION AND SOLUTION RESULTS"/"
2 "BOUNDARY LAYER AT Y=DELTA"/
3 "NODE X,Y COORDINATES DIMENSIONLESS FLUX"/
DO 10 K=0,1D-IC
I=ID-K
WRITE (4,660) I.X(I),Y(I),SOLN(I)
660 FORMAT (14,3X,3G12.5)
10 CONTINUE
WRITE (4,663) DELFLUX
663 FORMAT ('/"TOTAL FLUX ACROSS BOUNDARY LAYER SURFACE",13X,G12.5)
WRITE (4,670)
670 FORMAT ('/"CONCENTRATIONS IN THE BULK REGION"/
2 "LEFT SYMMETRY PLANE WALL",20X,"RIGHT SYMMETRY PLANE WALL"/
3 "NODE X,Y COORDINATES CONCENTRATIONS"/
DO 20 K=0,1D-BULK -2
IL=ID+1+K
IR=IC-1-K
WRITE (4,680) IL.X(IL),Y(IL),SOLN(IL),IR.X(IR),Y(IR),SOLN(IR)
680 FORMAT (2(14,3X,3G12.5))
20 CONTINUE
WRITE (4,690)
690 FORMAT ('/"CONCENTRATION ON THE RIGHHAND TRENCH TOP LEVEL"/
WRITE (4,660) (I.X(I),Y(I),SOLN(I),I=IB+2*NTRNCH+1,IB+2*(NTRNCH+
2 NTRTOP))
WRITE (4,700)
700 FORMAT ('/"CONCENTRATIONS ALONG VERTICAL TRENCH WALLS"/
DO 50 K=1,NTRNCH
IL=ID+2*D-BULK -1 + K
IR=IB + 2*NTRNCH + 1 - K
WRITE (4,710) IL.X(IL),Y(IL),SOLN(IL),IR.X(IR),Y(IR),SOLN(IR)
710 FORMAT (2(14,3X,3G12.5))
50 CONTINUE
WRITE (4,720)
720 FORMAT ('/"DIMENSIONLESS FLUX ACROSS CATHODE"/
FMN=ABS(EXFACT*(SOLN(1)**CEXP)))
FMAX=FMN
DO 60 I=1,IB
FLUX=EXFACT*(SOLN(I)**CEXP)
IF (ABS(FLUX).LT.FMIN) FMN=ABS(FLUX)
IF (ABS(FLUX).GT.FMAX) FMAX=ABS(FLUX)
WRITE (4,730) I.X(I),Y(I),SOLN(I),FLUX
60 CONTINUE
730 FORMAT (14,3X,4G14.7)
FPER=100.0*(FMAX-FMIN)/FMAX
WRITE (4,735) FPER
735 FORMAT ('/"MAXIMUM RELATIVE FLUX CHANGE ON CATHODE: ",F15.8," ")
DIFF=CATFLUX+DELFLUX
PERCENT=100.0*DIFF/CATFLUX
WRITE (4,740) CATFLUX,DIFF,PERCENT
740 FORMAT ('/"TOTAL MASS FLUX THROUGH CATHODE SURFACE",14X,G12.5/
2 /"OVERALL MASS FLUX IMBALANCE (+VE=CREATION)",16X,G12.5

296
Tafel Kinetics Model Code Listing

3 /*
 * MASS IMBALANCE AS PERCENTAGE OF CATHODE FLUX",.9X,.G12.5)
 RETURN
 END

C C C C C
C%include massbal
SUBROUTINE MASSBAL (X,Y,NXY,SOLN,DELFLUX,CATFLUX)
C%include implicit
 DIMENSION X(NXY),Y(NXY),SOLN(NXY)
 C
 C%include blk2
 C%include blk3
 C%include blk5
 C
 C PERFORMS OVERALL MASS BALANCE CALC. BY INTEGRATING NORMAL FLUX OVER
 C THE BOUNDARY LAYER AND OVER THE CATHODE SURFACE

 C DELFLUX=0.0
 CATFLUX=0.0
 DO 200 IO=1,NQ
 XI=PT(IO)
 DO 100 I=IC,ID-2,2
 DSDXI=(X(I)-X(I+2))/2.0
 CVALID FOR STRAIGHT, MIDSIDE NODE ELMT
 DCN=0.0
 DO 20 K=1,3
 20 DCN=DCN+PHI(K,XI)*SOLN(I+K-1)
 DELFLUX=DELFLUX+WT(IO)*DCN*DSDXI
 CONTINUE
 DO 150 I=1,IB-2,2
 DX=0.0
 DY=0.0
 CEE=0.0
 DO 120 K=1,3
 120 DX=DX +X(I+K-1)*DPHI(K,XI)
 150 DY=DY+Y(I+K-1)*DPHI(K,XI)
 CEE=CEE + SOLN(I+K-1)*PHI(K,XI)
 CONTINUE
 DCN=EFACT*(CEE**CEXPQ)
 CATFLUX=CATFLUX+WT(IO)*DCN*SQRT(DX**2 +DY**2)
 CONTINUE
 200 CONTINUE
 RETURN
 END

C C C C
C%include matfixed
SUBROUTINE MATFIXED (X,Y,NXY,A,F,HIJ)
C%include implicit
 DIMENSION X(NXY),Y(NXY),A(NXY,NXY),F(NXY),HIJ(NXY)
 C
 C%include blk2
 C%include blk4
 C
 C------------------------------------------------------------------------
 C SETS UP MATRIX CONTRIBUTIONS FOR FIXED ELEMENTS AND FIXED BASE NODES.
 C------------------------------------------------------------------------
 C------OUTER LOOP FOR EVERY FIXED ELEMENT------
 NSTART=NCATH + NTRENGH + 1
 NSTOP=NELMTS - NTRENGH
 DO 500 N=NSTART,NSTOP
 C---- FOR ELEMENT N, FIGURE OUT THE MAPPING INFO AT THE NORMAL
Tafel Kinetics Model Code Listing

C QUADRATURE POINTS
C
CALL ELMT(N,X,Y,NXY)
C----- NOW LOOP FOR EVERY STATIONARY NODE-------
ISTART=2*NSTART -1
ISTOP=2*NSTOP +1
DO 4000 I=ISTART,ISTOP
LEFT= I.EQ.NODE(1)
CENTER= I.EQ.NODE(2)
RIGHT= I.EQ.NODE(3)
SINGULAR= LEFT.OR.CENTER.OR.RIGHT
XXX=X(I)
YYY=Y(I)
CALL INTEL (I,XXX,YYY,A,F,NXY,HIJ)
4000 CONTINUE
5000 CONTINUE
C RETURN
END
C
C C%include matnl
SUBROUTINE MATNL (X,Y,NXY,A,F,SOLN)
%include implicit
DIMENSION X(NXY),Y(NXY),A(NXY,NXY),F(NXY),SOLN(NXY)
C
C%include blk2
C%include blk4
C
C----------------------------------------
C SUPERVISES CALCULATION OF NONLINEAR CONTRIBUTIONS COMING FROM
C CATHODE ELEMENTS AND ALL BASE NODES
C----------------------------------------
C --- OUTER LOOP FOR CATHODE ELEMENTS ---
DO 5000 N=1,NCATH
C --- CALC MAPPING DATA FOR NORMAL QUADRATURE PTS
CALL ELMT(N,X,Y,NXY)
C
C --- LOOP FOR EVERY BASE NODE ---
DO 4000 I=1,2*NELMTS
LEFT= I.EQ.NODE(1)
CENTER= I.EQ.NODE(2)
RIGHT= I.EQ.NODE(3)
SINGULAR= LEFT.OR.CENTER.OR.RIGHT
XXX=X(I)
YYY=Y(I)
CALL NLINTEL (I,XXX,YYY,A,F,NXY,SOLN)
4000 CONTINUE
5000 CONTINUE
RETURN
END
C
C C%include matvar
SUBROUTINE MATVAR (X,Y,NXY,A,F,HIJ)
%include implicit
DIMENSION X(NXY),Y(NXY),A(NXY,NXY),F(NXY),HIJ(NXY)
C
C%include blk2
C%include blk4
SETS UP MATRIX CONTRIBUTIONS FOR (ALL BOUNDARY ELEMENTS AND MOVING
BASE NODES) AND (MOVING ELEMENTS WITH ALL BASE NODES)

LOGICAL ELMOVE,PASS2

-----OUTER LOOP FOR EVERY ELEMENT---------
NFX=NCATH + NTRENCH
NFXE=NELMTS - NTRENCH
DO 5000 N=1,NELMTS
C --- CALC MAPPING DATA FOR NORMAL QUADRATURE PTS
CALL ELMT(N,X,Y,NXY)

C ELMOVE INDICATES A MOVING TRENCH REGION ELEMENT

ELMOVE=(N.LE.NFX.OR.N.GT.NFXE)

------- FOR MOVING ELEMENTS WE WILL LOOP ONCE FOR ALL
BASE NODES. FOR OTHER ELEMENTS WE WILL LOOP TWICE, ONCE FOR
LOW NUMBERED BASE NODES (CATHODE AND RIGHT TRENCH WALL), AND A
SECOND TIME FOR HIGH NUMBERED NODES (LEFT TRENCH WALL).

C C
C
C

PASS2=.FALSE.
ISTART=1
ISTOP=2*NELMTS

IF (.NOT.ELMOVE) ISTOP=2*(NCATH + NTRENCH)

3000 DO 4000 I=ISTART,ISTOP
LEFT= I.EQ.NODE(1)
CENTER= I.EQ.NODE(2)
RIGHT= I.EQ.NODE(3)
SINGULAR= LEFT.OR.CENTER.OR.RIGHT
XXX=X(I)
YYY=Y(I)
CALL INTEL (I,XXX,YYY,A,F,NXY,HIJ)

4000 CONTINUE
IF (ELMOVE) GO TO 5000
IF (PASS2) GO TO 5000
ISTART=2*(NELMTS - NTRENCH) + 2
ISTOP= 2*NELMTS
PASS2=.TRUE.
GO TO 3000

5000 CONTINUE
C
C NOW ALL HIJ, J.NE.I, HAVE BEEN FIGURED --- FIGURE
C OUT HIJ(I,I) BY SUMMATION
C
C CATHODE AND RHS INSULATOR
DD 6000 I=1,IC-1

6000 A(I,I)=HIJ(I)
C DELTA WHERE C=1
DD 6010 I=IC,ID

6010 F(I)=F(I) - HIJ(I)
C LHS INSULATOR
DD 6020 I=ID+1,2*NELMTS

6020 A(I,I)=HIJ(I)
C
RETURN
END
Tafel Kinetics Model Code Listing

C%include nlintel
SUBROUTINE Nlintel(I,XXX,YYY,A,F,NXY,SOLN)
%include implicit
DIMENSION A(NXY,NXY),F(NXY),SOLN(NXY),DQ(3)
C
%include blk2
%include blk3
%include blk4
%include blk5
C
---------------------------------------------------------------------
C THIS ROUTINE CALCULATES THE NONLINEAR SOLUTION DEPENDENT TERMS
C ASSOCIATED WITH THE CATHODE KINETIC BOUNDARY CONDITION.
C THE TERMS ARE ADDED INTO THE JACOBIAN AND THE RESIDUAL
C
---------------------------------------------------------------------
C
C INITIALIZE TERMS
GIJ=0.0
DO 10 K=1,3
10 DQ(K)=0.0
C
GOTO SPECIAL CODE IF SINGULAR
IF(SINGULAR) GOTO 200
C
C--------LOOP FOR EACH NORMAL QUADRATURE POINT--------
DO 100 IQ=1,NQ
XI=PT(IQ)
R2=(YQ(IQ)-YYY)**2 + (XQ(IQ)-XXX)**2
DS=SQRT(DXDXI(IQ)**2 + DYDXI(IQ)**2)
GG=WT(IQ)*DS*ALOG(R2)**0.5
CEE=0.0
DO 20 K=1,3
20 CEE=CEE + PHI(K,XI)*SOLN(NODE(K))
QTERM=GG*EXFACT*(CEE**CEXP0)
GIJ=GIJ + QTERM
DQTERM=CEXP0*QTERM/CEE
DO 30 K=1,3
30 DQ(K)=DQ(K) - DQTERM*PHI(K,XI)
100 CONTINUE
C
C ADD RESULTS INTO RESIDUAL AND JACOBIAN
F(I)=F(I) + GIJ
DO 120 K=1,3
120 A(I,NODE(K))=A(I,NODE(K)) + DQ(K)
RETURN
C
C--------FROM HERE ON CODE FOR SINGULAR INTEGRALS, GIJ CONTRIBUTIONS-------
C
C FIRST LOOP FOR NONSINGULAR CONTRIBUTION OF REMAPPED SINGULAR INTEGRALS
C (GIJ AND DQ() WERE ZEREOED ABOVE)
C
200 DO 300 IQ=1,NQ
XI=PT(IQ)
R2=(XQ(IQ) - XXX)**2 + (YQ(IQ) - YYY)**2
IF (LEFT) R2=4.0*R2/((XI+1.0)**2)
IF (RIGHT) R2=4.0*R2/((1.0 -XI)**2)
IF (CENTER) R2=R2/(XI**2)
GG=WT(IQ)*SQRT(DXDXI(IQ)**2 + DYDXI(IQ)**2) **0.5*ALOG(R2)
CEE=0.0
DO 220 K=1,3
220 CEE=CEE + PHI(K,XI)*SOLN(NODE(K))
QTERM=GG*EXFACT*(CEE**CEXP0)
GIJ=GIJ + QTERM
DQTERM=CEXP0*QTERM/CEE
DO 230 K=1,3
230 DQ(K)=DQ(K) - DQTERM*PHI(K,XI)
300 CONTINUE

300
Tafel Kinetics Model Code Listing

C GIJ AND DO(1 2 3) WILL BE ADDED TO BELOW
C AND THEN THEY WILL BE ADDED TO THE RESIDUAL AND THE
C JACOBIAN
C
C IF (CENTER) GOTO 600
C
C---HERE FOR LEFT OR RIGHT NODE SINGULARITY, LOOP FOR LOG QUAD. PTS--
C
C TERMS GIJ AND DO(1 2 3) ALREADY HAVE VALUES
C FROM REMAPPED NONSINGULAR INTEGRALS--- BUILD ON THEM
C
DD 500 IQ=1,NLN
IF (LEFT) XI=2.0*PLN(IQ) - 1.0
IF (RIGHT) XI=1.0 - 2.0*PLN(IQ)
DXL=0.0
DYL=0.0
CEE=0.0

DO 410 K=1,3
DXL=DXL + DPHI(K,XI)*XX(K)
DYL=DYL + DPHI(K,XI)*YY(K)
CEE=CEE + PHI(K,XI)*SOLN(NODE(K))
CONTINUE

410 THE LOG R TERM IS BURIED IN THE QUADRATURE PTS AND WEIGHTS--
DS=2.0 * SQRT(DXL**2 + DYL**2)
GG=WLN(IQ)*DS
QTERM=GG*EXFACT*(CEE**CEXP0)
GIJ=GIJ - QTERM
DQTERM=CEXP0*QTERM/CEE
DO 430 K=1,3

430 DQ(K)=DQ(K) - DQTERM*PHI(K,XI)

500 CONTINUE
C
F(I)=F(I) + GIJ
DO 520 K=1,3

520 A(I,NODE(K))=A(I,NODE(K)) + DQ(K)
C
RETURN
C
C------ HERE FOR CENTER NODE SINGULAR. , NOT INSULATOR ------
C BREAK INTO 2 INTEGRALS, LEFT AND RIGHT OF SINGULARITY
C
600 DO 700 KK=1,2
C
C----LOOP FOR EACH QUADRATURE PT----
DO 680 IQ=1,NLN
XI=PLN(IQ)
IF(KK.EQ.1) XI=-XI
DXL=0.0
DYL=0.0
CEE=0.0

DO 610 K=1,3
DXL= DXL + DPHI(K,XI)*XX(K)
DYL= DYL + DPHI(K,XI)*YY(K)
CEE=CEE + PHI(K,XI)*SOLN(NODE(K))
CONTINUE

610 GG=WLN(IQ)*SQRT(DXL**2 + DYL**2)
QTERM=GG*EXFACT*(CEE**CEXP0)
GIJ=GIJ - QTERM
DQTERM=CEXP0*QTERM/CEE
DO 630 K=1,3

630 DQ(K)=DQ(K) - DQTERM*PHI(K,XI)

680 CONTINUE

700 CONTINUE
C
F(I)=F(I) + GIJ
DO 720 K=1,3

720 A(I,NODE(K))=A(I,NODE(K)) + DQ(K)
C
RETURN
END
Tafel Kinetics Model Code Listing

C
C
C%include phi
FUNCTION PHI(N,Z)
%include implicit
GOTO (1,2,3),N
1 PHI=Z*(Z-1.0)/2.0
RETURN
2 PHI=1.0-Z*Z
RETURN
3 PHI=Z*(Z+1.0)/2.0
RETURN
END
C
C
C%include quad
SUBROUTINE QUAD
%include implicit
C
%include bik3
C
C
ASSIGN QUADRATURE PTS AND WEIGHTS ACCORDING TO USER CHOICE GIVEN
BY THE VALUES OF NQ AND NLN
C
20 IF (NQ.NE.20) GOTO 10
PT(1)=.07652 65211 33497d0
PT(2)=PT(1)
PT(3)=.22778 58511 41645d0
PT(4)=PT(3)
PT(5)=.37370 60887 15420d0
PT(6)=PT(5)
PT(7)=.51086 70019 50827d0
PT(8)=PT(7)
PT(9)=.63605 36807 26515d0
PT(10)=PT(9)
PT(11)=.74633 19064 60151d0
PT(12)=PT(11)
PT(13)=.83911 69718 22219d0
PT(14)=PT(13)
PT(15)=.91223 44282 51326d0
PT(16)=PT(15)
PT(17)=.96397 19272 77914d0
PT(18)=PT(17)
PT(19)=.99312 85991 85095d0
PT(20)=PT(19)
WT(1)=.15275 33871 30726d0
WT(2)=WT(1)
WT(3)=.14917 29864 72604d0
WT(4)=WT(3)
WT(5)=.14209 61093 18382d0
WT(6)=WT(5)
WT(7)=.13168 86384 49177d0
WT(8)=WT(7)
WT(9)=.11819 45319 61518d0
WT(10)=WT(9)
WT(11)=.10193 01198 17240d0
WT(12)=WT(11)
WT(13)=.08327 67415 76705d0
WT(14)=WT(13)
WT(15)=.06267 20483 34109d0
WT(16)=WT(15)
WT(17)=.04060 14298 00387d0
WT(18)=WT(17)
WT(19)=.01761 40071 39152d0
WT(20)=WT(19)
GO TO 100
10 IF (NQ.NE.10) GOTO 6
   PT(1) = 1.48874338981631d0
   PT(2) = PT(1)
   PT(3) = 4.33395394129247d0
   PT(4) = PT(3)
   PT(5) = 6.79409568299024d0
   PT(6) = PT(5)
   PT(7) = 8.65063366688985d0
   PT(8) = PT(7)
   PT(9) = 9.73906528517172d0
   PT(10) = PT(9)
   WT(1) = 295524224714753d0
   WT(2) = WT(1)
   WT(3) = 269266719309996d0
   WT(4) = WT(3)
   WT(5) = 2.19086362515982d0
   WT(6) = WT(5)
   WT(7) = 1.49451349150581d0
   WT(8) = WT(7)
   WT(9) = 0.66671344308688d0
   WT(10) = WT(9)
GO TO 100
6 IF (NQ.NE.6) GOTO 4
   PT(1) = 238619185083197d0
   PT(2) = PT(1)
   PT(3) = 6.61209386466265d0
   PT(4) = PT(3)
   PT(5) = 9.32469514203152d0
   PT(6) = PT(5)
   WT(1) = 467913934572691d0
   WT(2) = WT(1)
   WT(3) = 3.60761573048139d0
   WT(4) = WT(3)
   WT(5) = 1.71324492379170d0
   WT(6) = WT(5)
GO TO 100
4 C (DEFAULT EVEN IF NQ.NE.4)
   PT(1) = 3.39981043584856d0
   PT(2) = PT(1)
   PT(3) = 8.61136311594053d0
   PT(4) = PT(3)
   WT(1) = 5.52145154862546d0
   WT(2) = WT(1)
   WT(3) = 3.47854845137454d0
   WT(4) = WT(3)
100 IF (NLN.NE.7) GOTO 40
   PLN(1) = 0.1671935d0
   PLN(2) = 1.0018567d0
   PLN(3) = 2.4629424d0
   PLN(4) = 4.3346349d0
   PLN(5) = 6.3235098d0
   PLN(6) = 8.1111862d0
   PLN(7) = 9.4084816d0
   WLN(1) = 1.9616838d0
   WLN(2) = 2.7030264d0
   WLN(3) = 2.3986187d0
   WLN(4) = 1.6577577d0
   WLN(5) = 0.8894322d0
   WLN(6) = -0.3319430d0
   WLN(7) = -0.00593278d0
RETURN
40 C DEFAULT
   PLN(1) = 0.4144848d0
   PLN(2) = 2.4527481d0
   PLN(3) = 5.5616545d0
   PLN(4) = 8.4898239d0
   WLN(1) = -3.8346406d0

303
Taef Kinetics Model Code Listing

WL2(2)=-.38687531d0
WL2(3)=-.19043512d0
WL2(4)=-.03922548d0
RETURN
END

C C C
C
C%include regrid
SUBROUTINE REGRID(X,Y,XNY,XNEW,YNEW,ICATH,IERR)
C%include implicit
DIMENSION X(XNY),Y(XNY),XNEW(ICATH),YNEW(ICATH)
LOGICAL IERR
C C
%include b1k2

-----------------------------------------------
C USING POSITION OF FRESHLY MOVED CATHODE, THIS SUBROUTINE
C ADJUSTS THE SPACING OF THE NODES ON THE VERTICAL TRENCH
C WALLS. IF THE DEPOSIT HAS MOVED BEYOND THE TRENCH REGION
C THE STRUCTURE OF THIS PROGRAM IS INAPPROPRIATE AND IERR
C IS SET TO TRUE.
C
-----------------------------------------------
C-----CLEAR TROUBLE FLAG-----
IERR=.FALSE.
C-------- CORNER NODES ARE INDICATED IN COMMON ------
C IB= RIGHHAND SIDE TRENCH BOTTOM
C IBB= RIGHT SIDE TRENCH TOP
C IDD= LEFT TRENCH TOP
C-----------------------CHECK FOR GROWTH ABOVE TRENCH TOP ON RHS-------
C DELY=Y(IBB)-Y(IB)
C IF (DELY.LE.0) GO TO 1000
C-----------------------ADJUST RHS TRENCH WALL SPACING-------
C FACT=DELY/FLOAT(IBB-IB)
C DO 10 I=IB+1,IBB-1
C 10 Y(I)=Y(IB) + FLOAT(I-IB)*FACT
C
C-----------------------CHECK FOR GROWTH ABOVE TRENCH REGION ON LHS-------
C DELY=Y(IDD)-Y(1)
C IF (DELY.LE.0) GO TO 1000
C-----------------------ADJUST LHS TRENCH WALL SPACING-------
C FACT=DELY/FLOAT(2*NELMTS+1-IDD)
C DO 20 I=IDD+1,2*NELMTS
C 20 Y(I)=Y(IDD)-FLOAT(I-IDD)*FACT
C
C SHIFT CATHODE REPRESENTATION TO ONE THAT IS EVENLY SPACED ALONG X
C START WITH THE 2ND NODE AND THE FIRST CATHODE ELEMENT
C I=1
N=1
C
C CHOOSE A NEW NODE
30 I=I+1
C CHECK IF ALL DONE
IF(I.GT.2*NCACTH) GOTO 500
C XNODE IS POSITION OF EVENLY SPACED REPRESENTATION
C XNODE=FLOAT(I-1)*(X(IB)-X(1))/FLOAT(IB-1)
C IS CURRENT NODE ON THE CURRENT ELEMENT?
40 IF(XNODE.LE.X(2*N+1)) GOTO 50
C NO --- CHOOSE NEW ELEMENT
N=N+1
C IF WE'RE OUT OF ELEMENTS, ITS AN ERROR
IF(N.GT.NCATH) GOTO 1000
C GO SEE IF THIS ELEMENT IS ALRIGHT
GOTO 40
50 CONTINUE
C NODE HAS BEEN FOUND ON CURRENT ELEMENT
C Figure out xi coordinate from x value
C Fixed point iteration is used
C Initial guess
XI=0.0
TOLERANCE=1.0E-6
AA=(X(2*N+1) - 2.0*X(2*N) + X(2*N-1))/(X(2*N+1) - X(2*N+1))
BB=2.0*(X(2*N) - XNODE)/(X(2*N-1) - X(2*N+1))

60 XIOLD=XI
XI=AA*XIOLD**2 + BB
IF(ABS(XI-XIOLD).GT.TOLERANCE) GOTO 60
C Have converged on xi
XNEW(I)=XNODE
YNEW(I)=0.0
DO 70 K=1,3
   YNEW(I)=YNEW(I) + PHI(K,XI)*Y(2*N + K -2)
C Return for next node
GOTO 30

500 CONTINUE
C All done reassign x, y
DO 510 I=2,2*NCATh
   X(I)=XNEW(I)
C Return

1000 IERR=.TRUE.
C------ here if program is invalidated by cathode growth above trench level---
C------ or by growth through trench wall ---
RETURN
C
C
C%include simq

SUBROUTINE SIMQ(A,B,N,KS)
%include implicit

Solves simultaneous linear equation set A x = b

Description of parameters
A - Matrix of coefficients stored columnwise. These are destroyed in
the computation. The size of a is N x N
B - Vector of original constants (length N). These are replaced
by final solution values, vector X.
N - Number of eons= number of variables N .GE. 1
KS - Singularity flag returns 1 if singular else 0

Gaussian elimination with partial pivoting

Current pivot tolerance is 0.0

DIMENSION A(1),B(1)

FORWARD SOLN
TOL=0.0
KS=0
JU=N
DO 65 J=1,N
   JY=J+1
   JU=JU+N+1
   B1=G0.0
   IT=JU-J
   DO 30 I=J,U

SEARCH FOR MAXIMUM COEF. IN COLUMN
   IU=IT+I

65 CONTINUE
Tafel Kinetics Model Code Listing

IF(ABS(BIGA)-ABS(A(IJ))) 20,30,30
BIGA=A(IJ)
IMAX=I
CONTINUE

TEST FOR PIVOT LESS THAN TOLERANCE
IF(ABS(BIGA)-TOL) 35,35,40
KS=1
RETURN

INTERCHANGE ROWS IF NECESSARY
II=J+N*(J-2)
IT=IMAX-J
DO 50 K=J,N
   II=II+N
   I2=II+IT
   SAVE=A(I2)
   A(I2)=A(II)
   A(II)=SAVE
50

DIVIDE EQN BY LEADING COEFFICIENT
A(II)=A(II)/BIGA
SAVE=B(IMAX)
B(IMAX)=B(J)
B(J)=SAVE/BIGA

ELIMINATE NEXT VARIABLE
IF(J-N) 55,70,55
IOS=N*(J-1)
DO 65 IX=J,Y,N
   IXJ=IOS+IX
   IT=J-IX
   DO 60 JX=J,Y,N
      IXJX=N*(JX-1)+IX
      JUX=IXJX+IT
      A(IXJX)=A(IXJX)-(A(IXJ)*A(JUX))
60
   IXJ=IXJ-(B(J)*A(IXJ))
65

BACK SOLN
NY=N-1
IT=N=N
DO 80 K=1,NY
   IA=IT-J
   IB=N-J
   IC=N
   DO 80 J=1,J
      B(IB)*B(IB)-A(IA)*B(IC)
      IA=IA-N
80
IC=IC-1
RETURN
END

%include stat
SUBROUTINE STAT(X,Y,NXY)
%include implicit
DIMENSION X(NXY),Y(NXY)
%include d1k2

MEASURE DEPOSIT AREA  
ND AREA OF DEPOSIT BY SIMPSON'S RULE  
-ABCIISSA IS EVENLY SPACED, GROWTH IS PARABOLIC ON  
EACH ELEMENT
C
AREA=0.0
DO 100 I=1,NCATH
  AREA=AREA+2.0*Y(2*I)+Y(2*I+1)
100
CONTINUE
  AREA=2.0*AREA+Y(1)-Y(IB)*(X(IB)-X(1))/(6.0*NCATH)
C SCALE AREA BY AREA OF TRENCH
C -TRENCH WIDTH ASSUMED 1
  AREA=AREA/Y(IB+Z2*NTRENCH)
C
C-------------------------STATISTICAL ANALYSIS OF FINAL CATHODE SHAPE-------
SX=0.0
SY=0.0
SXX=0.0
SYY=0.0
SXY=0.0
N=0
DO 400 I=1,IB
  N=N+1
  SX=SX+X(I)
  SY=SY+Y(I)
  SXX=SXX+X(I)*X(I)
  SYY=SYY+Y(I)*Y(I)
  SXY=SXY+X(I)*Y(I)
400
CONTINUE
B1=(SX*X*SY/N)/(SXX-SX*SY/N)
BO=(SY-B1*SX)/N
SSTOTAL=SY-(SY*SY)/N
SSREG=B1*(SX*X*SY/N)
C SSRESID CAN BE NEGATIVE OF ORDER EPSILON WITH NEARLY LINEAR PROFILES
C BECAUSE OF PRECISION PROBLEMS, HENCE ABS()
  SSRESID=ABS(SSTOTAL-SSREG)
  RESDEV=SQRT(SSRESID/(N-2))
  YBAR=SY/N
  IF (YBAR.ABSENT1.0.E0) RIPPLE=RESDEV*100/YBAR
  WRITE(4,850) YBAR,AREA,B1,BO,SSTOTAL,SSREG,SSRESID,RESDEV,RIPPLE
850
  FORMAT ('/STATIONARY ANALYSIS OF FINAL CATHODE SHAPE'/
2 'AVERAGE Y VALUE',F10.4/
3 'Deposit Area/Trench Area',F10.6/
3 'LINEAR DECESSION OF CATHODE, SLOPE',F10.4/
4 'INTERCEPT=',F10.4/
5 'TOTAL SUM OF SQUARES ABOUT YBAR',E12.5/
6 'SS EXPLAINED BY THE LINEAR REGRESSION',E12.5/
7 'SS THAT REMAINS (RESIDUAL SS)',E12.5/
8 'RESIDUAL AS A DEVIATION',E12.5/
9 'RIPPLE (RESIDUAL DEVIATION/YBAR)*100',F10.4//)
C
C RETURN
END
C
C
C%include surf
SUBROUTINE SURF (I,X,Y,NXY,DXDT,DYDT,SOLN)
%include implicit
  DIMENSION SOLN(NXY),X(NXY),Y(NXY)
C
%include blk1
%include blk2
%include blk3
C IF (I.EQ.1.OR.I.EQ.IB) GO TO 100
  DX=X(I+1)-X(I-1)
  DY=Y(I+1)-Y(I-1)
100
  D=SQRT(DX**2+DY**2)
  DCNO=EXFACT*(SOLN(I)**CEXPO)
  DXDT=DX/DCNO/D
  DYDT=DY/DCNO/D
  RETURN

307
100 IF (I.NE.1) GO TO 200
   DX=-3.0*X(1) + 4.0*X(2) - X(3)
   DY=-3.0*Y(1) + 4.0*Y(2) - Y(3)
   GO TO 50
200  DX=X(IB-2) - 4.0* X(IB-1) + 3.0*X(IB)
     DY=Y(IB-2) - 4.0*Y(IB-1) + 3.0*Y(IB)
     GO TO 50
     END
APPENDIX J

CPU_TIMER LISTING

cpu_timer: procedure(cputime);
    declare cputime float bin(63);
    /* sets cputime to the number of virtual cpu seconds elapsed
       in the current session */
cputime=vclock()/1.0e+6;
    /* implicit conversion of elapsed microseconds

    To use in your fortran program:

    external cpu_timer(descriptors)
    real cpustart,cputime
    call cpu_timer(cpustart)
       (your algorithm = lines to be timed)
    call cpu_timer(cputime)
    cputime=cputime-cpustart
    write (4,600) cputime
    600 format ("Elapsed Virtual CPU seconds=",f12.5)

    */

    end cpu_timer: