Lawrence Berkeley Laboratory
UNIVERSITY OF CALIFORNIA

Materials & Molecular Research Division

DESIGN OF THIN-GAP CHANNEL FLOW CELLS

V. Edwards
(Ph.D. Thesis)

July 1986

Prepared for the U.S. Department of Energy under Contract DE-AC03-76SF00098

DISCLAIMER

LBL-21848

Design of Thin-Gap Channel Flow Cells

Victoria Edwards

Ph. D. Thesis

Department of Chemical Engineering

University of California

and

Lawrence Berkeley Laboratory

Berkeley, CA 94704

July, 1986

Design of Thin-Gap Channel Flow Cells

Victoria Edwards

Abstract

This dissertation is an investigation of thin-gap channel electrochemical reactors. A thin-gap flow cell consists of parallel plates, each with an embedded electrode. Electrolyte flows between the plates, and electrochemical reactions occur on the electrode surfaces. The term "thin-gap" applies when the spacing between the plates is thin compared to the diffusion-boundary-layer thickness.

The first three chapters of the dissertation contain a theoretical investigation of channel flow cells. A mathematical model is presented that includes the effects of interacting boundary layers, simultaneous electrochemical reactions, and ohmic potential drop. Two techniques are developed for calculating the steady-state current, concentration, and potential distribution in the flow cell.

Experimental verification of the mathematical model is presented in the last two chapters. The experiments were carried out with a $K_3Fe(CN)_6/K_4Fe(CN)_6/NaOH$ electrolyte in a channel flow cell with nickel electrodes. The current-voltage curves showed excellent agreement with the model over a wide range of boundary-layer thicknesses.

Acknowledgment

I would like to thank Professor Newman for everything he has taught me. His insight and methodology for tackling difficult problems are resources I have only begun to tap. In addition to having a vast wealth of knowledge and expertise, Professor Newman is a generous man who is dedicated to his students. Professor Tobias' advice — both technical and nontechnical — is gratefully acknowledged. I also thank him for his friendship and for reviewing the manuscript. I thank Professor DeJonghe for his review of the manuscript.

The machine-shop personnel are to be commended, particularly Fred Wolff and Andy Anderson, for their helpful suggestions, and Clay Taylor, for building the flow cell.

I am happy to have made several high-quality friends at Berkeley, particularly Gina Whitney and Mike Matlosz. My friends have provided me with an enjoyable and stimulating work environment and encouragement when it was needed.

Finally, I would like to thank Dad for his technical advice and training and both parents for their constant support.

This work was supported by the Assistant Secretary for Conservation and Renewable Energy, Office of Energy Systems Research, Energy Storage Division of the U.S. Department of Energy under Contract No. DE-AC03-76SF00098.

Table of Contents

Abstract 1

Acknowledgments i

Chapter 1. Analysis of Thin-Gap Channel Flow Cells 1

1. Introduction 1
1.1. Literature Review 4
1.2. A Mathematical Model of Thin-Gap Channel Flow Cells 7
1.2.1. Assumptions 7
1.2.2. Governing Equations 7
1.3. Dimensionless Formulation 13
1.4. Solution Techniques 15
1.4.1. Advantages and Disadvantages of the Two Solution Techniques 24
1.4.2. Mesh Spacing 25
1.4.3. Integration Methods 26
1.5. Summary 29

Nomenclature for Chapter 1 30

Chapter 2. The Asymmetric Graetz Problem in Channel Flow 36

2.1. Introduction 36
2.2. Problem Statement 36
2.3. The Lévêque Approach 37
2.4. The Graetz Approach 38
2.5. Solution of the Eigenvalue Problem 38
2.6. Combining the Graetz and Lévêque Solutions 39
2.7. Asymptotic Forms for Large Eigenvalues 39
2.8. Empirical Approach 42
2.9. Summary of Results 43

Nomenclature for Chapter 2 44

Chapter 3. Thin-Gap-Model Results 47

3. Introduction 47
3.1. Limiting Cases of Current Distribution 48
3.1.1. Results for Thin Diffusion Boundary Layers 50
3.1.2. Results for Straight Current Lines 50
3.2. Thin-Gap-Model Results for a Plating Reaction 52
3.3. Results for a Redox Reaction 62

Nomenclature for Chapter 3 71

Chapter 4. Experimental Apparatus and Procedure 73

4.1. Introduction 73
4.2. Selection of the Electrochemical System 74
4.3. Electrolyte Properties and Preparation 78
4.4. Channel-Flow-Cell Apparatus and Procedure 79
4.4.1. Experimental Apparatus 79
4.4.2. Experimental Procedure 86
4.5. Rotating-Disk Experiments 87
4.5.1. Experimental Procedure 87

Nomenclature for Chapter 4 90

Chapter 5. Comparison of Experimental and Theoretical Results 92

5.1. Introduction 92
5.2. Results for Thin Boundary Layers 93
5.3. Discussion 105
5.4. Measured and Simulated Polarization Curves 108

Nomenclature for Chapter 5 124

Appendices

Appendix A. The Lévêque Series 126

A.1. Introduction 126
A.2. Mathematical Formulation 126

Appendix B. Numerical Solution of the Asymmetric Graetz Problem 130

Appendix C. Program EIGEN 134

Appendix D. Iterative Methods 143

D.1. Introduction 143
D.2. Comparison of Iterative Methods 144
D.3. Application of Some Iterative Methods to the Rotating-Disk Problem 150
D.3.1. Cutting the Loop 151
D.3.2. Calculating the Jacobian 152
D.4. Iterative Methods for the Channel Problem 155
D.4.1. Successive Substitution 155
D.4.2. Newton-Raphson with a Global Approximation Method 156
D.5. Collocation Applied to the Channel Problem 157
D.5.1. Introduction 157
D.5.2. Choice of Trial Functions 159
D.5.3. Basic Iteration Scheme 160
D.5.4. Limiting-Current Problems 170
D.5.5. First-Order Continuation 172

D.6. Results at the Limiting Current 173
D.7. Summary and Suggestions for Future Work 179

Appendix E. Programs for Investigating Iterative Methods for the Rotating-Disk Problem 181

E.1. Program CURDB 181
E.2. Program CURDBN 194
E.3. Program CURDE 206
E.4. Program CURDEN 218

Appendix F. Program CHANNEL 227

F.1. Program Description 227
F.2. Major Subroutines 231
F.3. Program Listing 234

Appendix G. Program NEWCHAN 319

G.1. Program Description 319
G.2. Major Subroutines 323
G.3. Program Listing 326

Appendix H. Tables of Experimental Parameters 379

References 393

1. Analysis of Thin-Gap Channel Flow Cells

1. Introduction

A channel flow cell consists of two parallel plates, which serve as the anode and the cathode for electrochemical reactions. The electrolyte flows between the electrodes, and the current flow is roughly perpendicular to the fluid flow. Figure 1-1 shows a schematic of a channel flow cell. When the gap between the two plates is thin compared to the diffusion-boundary-layer thickness, the cell is called a thin-gap channel flow cell. The advantage of a thin-gap cell is its low resistance.

Channel cells are used in many industrial electrochemical processes, such as copper refining, some zinc-halogen energy storage cells, and the Monsanto process for conversion of acrylonitrile to adiponitrile in the preparation of nylon.[1-8] Channel flow cells are attractive for such applications because they provide continuous production, are simple to operate and maintain, and do not require a high capital investment.[9]

Some cells have a membrane separator at the midplane of the channel; others are undivided. Separators are used to prevent "chemical shorting," where a product from one electrode diffuses to the other electrode and reacts back to the starting material. With a separator, it is possible for the anolyte and catholyte to have different compositions and flow rates,[10] allowing greater control over the selectivity of the reaction. The problems with separators, however, are that they are expensive and they increase significantly the cell resistance, which is a critical factor in electroorganic syntheses because the electrolytes typically have low conductivities. In some cases, the separator must be eliminated because an intermediate produced on one electrode is needed for the reaction on the other electrode. Reactions of this type, called "paired" syntheses, are attractive because both electrode reactions contribute directly to useful products. Some

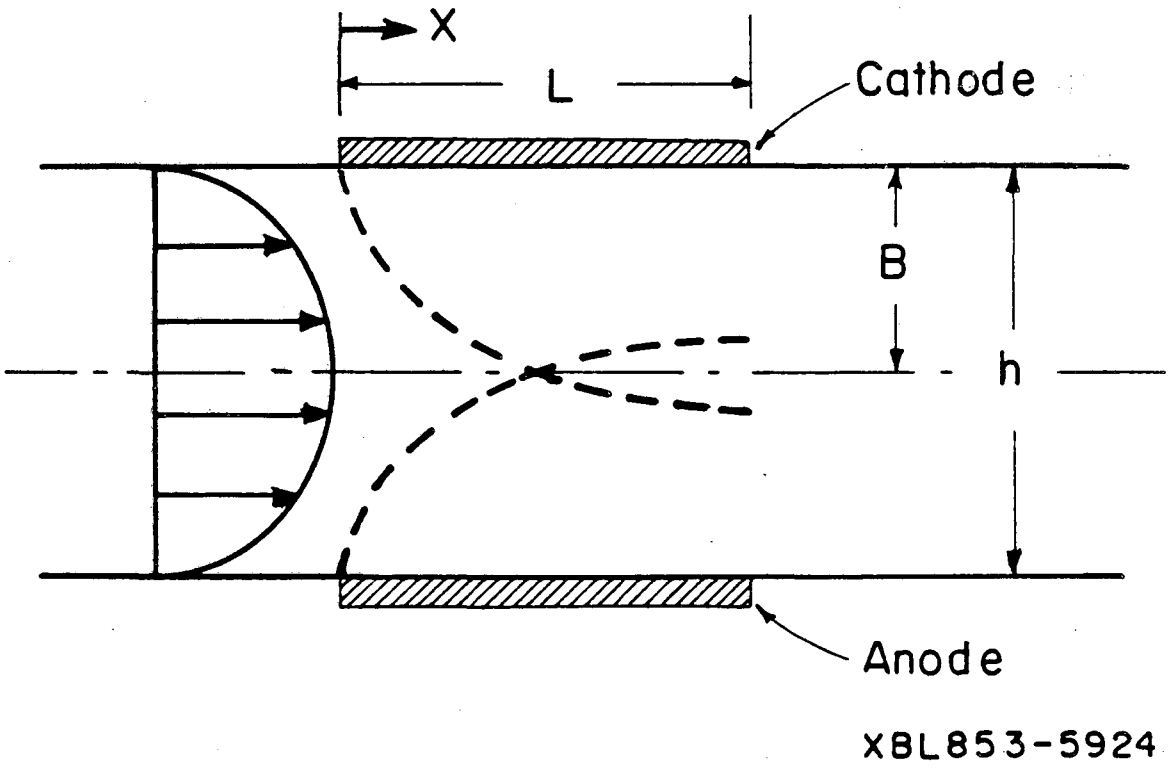


Figure 1-1. Channel flow cell geometry.

examples of paired syntheses[1] include the epoxidation of propylene to form propylene oxide, [1,2,11-15] cathodic hydrodimerization of acrylonitrile to adiponitrile with anodic oxidation of 2-propanol to acetone,[16] and production of sorbitol and gluconic acid from glucose.[17] Undivided thin-gap cells may be useful for such applications because the gap is thinner than the diffusion boundary layer. If the gap is too thin, however, there may be electrical shorts caused by impurities bridging the gap, or bubbles from gas-evolving side reactions may become trapped in the cell. Also, as the gap spacing is decreased, the heat generation and pressure drop increase. Thus, the design of a thin-gap cell requires an analysis of the economic tradeoff between decreased cell resistance, increased pressure drop, and increased diffusion of products across the gap.

The analysis of thin-gap channel flow cells is complex because several coupled phenomena must be taken into account, such as mass transfer, thermodynamics, and the kinetics of multiple reactions. Although channel-flow-cell models have been developed previously,[18-27] each model contains restrictive assumptions. In this study, a laminar-flow model is developed that allows for multiple electrochemical reactions in cells with interacting diffusion boundary layers (thin-gap cells). The model predicts current, concentration, and potential distributions along the electrode surfaces. The major limitation of the model is that it does not treat homogeneous reactions or gas-evolving reactions.

Chapter 1 presents, first, a brief review of previous laminar-flow models, and, second, a discussion of the new thin-gap model. Chapter 2 develops a solution that is needed in Duhamel's superposition integral to construct the solution to the channel-flow problem with arbitrary axially-varying boundary conditions. Theoretical results are presented in chapter 3. Chapter 4 describes an experiment made to verify the model, and chapter 5 discusses the comparison between the theoretical and experimental

results.

1.1. Literature Review

Modeling electrochemical systems, such as channel flow cells, requires equations describing mass transfer, current flow, and electroneutrality. Mass transfer is generally described by material balances and flux expressions that contain terms representing diffusion, convection, and migration in the electric field. The current flow is caused by the motion of charged species and is therefore related to the fluxes of the ions. Electroneutrality is observed in all solutions, except in the thin ($\approx$ 1 to 10 nm) double charge layer near electrodes and other boundaries. These equations are discussed in chapter 11 of Newman's book.[28]

In this work, we focus on mathematical models that assume laminar flow between the plates. This assumption is valid if the Reynolds number is less than about 2100. The choice between laminar and turbulent flow is dictated by the economic tradeoff between enhanced mass transport and increased pressure drop. Acosta *et al.*[29] have examined this tradeoff in an experimental investigation of thin-gap flow cells. Another incentive for using turbulent flow is to ensure that the flow is fast enough to sweep out bubbles from any gas-evolving side reactions.

Another assumption sometimes made in mathematical models of flow cells is that the diffusion boundary layers are thin, which will be true when

$$L < 0.005\, Re\, Sc\, d_e. \tag{1-1}$$

This assumption simplifies the analysis of the mass transfer because not only can the electrodes be treated separately, but also the parabolic velocity profile can be taken to be linear throughout the boundary layer, and the Lévêque approximation[28,30,31] may be used.[32] If

$$L > 0.05\, Re\, Sc\, d_e, \tag{1-2}$$

then the mass transfer is fully developed,[33] and the electrodes must be treated simultaneously.

A third assumption is that migration of the reacting ions may be neglected in the flux equations. This assumption is valid when there is an excess of nonreacting ("supporting") electrolyte.[28]

The assumptions discussed above — thin boundary layers and excess supporting electrolyte — have been used both by Parrish and Newman[20,21] and by Lee and Selman.[10] Parrish and Newman obtained the current and concentration distributions in an undivided channel flow cell with a single metal deposition and dissolution reaction. Lee and Selman included the effects of a separator and of finite electrode resistance. Both models account for mass transfer, reaction kinetics, thermodynamics, and ohmic potential drop.†

White *et al.*[18,19] have developed two models that include migration and multiple reactions, without invoking the thin-diffusion-layer assumption. Their first model[18] assumes that the aspect ratio, h/L, is small, and that the current lines are therefore straight. That model, therefore, does not account for the two-dimensional structure of the potential distribution. This approach is similar to that taken by Alkire and Ng for modeling flow-by porous electrodes.[34,35] Fedkiw[36] and Risch[37] discuss the two-dimensional nature of the potential distribution in flow-by porous electrodes. Recently,[38] Mader *et al.* published a simplified version of this model that has a single mesh point in the axial direction. This simplification cuts the computer time by a factor of 40, but the results are significantly less accurate. They suggest using the simple model for screening sets of parameters to use in the more expensive model. The model

†Ohmic potential drop is the potential difference arising from the flow of current through the resistive solution.

of Nguyen *et al.*[19] is much more complete than the earliest model; it includes axial migration and diffusion and time dependence. A recently-published extension of Nguyen's model[39] includes a CSTR at the outlet and partial recycle to the channel. Although their model does not include homogeneous chemical reactions within the channel, it could be easily extended to do so. Homogeneous reactions are particularly important in electro-organic reactions, such as the paired syntheses mentioned earlier.[1,2,11-17]

Sakellaropoulos and Francis[23-25] discuss channel flow cells in the context of controlling selectivity in cells with multiple competing reactions (parallel[24,25] or consecutive[23] reactions). Their model assumes plug flow in the channel, and it does not include ohmic drop or reactions at the other electrode.

Several limiting cases for channel-flow-cell behavior are analyzed by Pickett.[26] One limiting case, for "short electrodes," is the Lévêque solution, which applies when mass transfer alone dominates the current distribution. Effects of finite electrode width or short hydrodynamic entry lengths are also discussed. Another limiting case, for "long electrodes," is fully-developed mass transfer. In this case, however, Pickett assumes constant flux along the electrode surfaces and does not account for ohmic potential drop or electrode kinetics. These effects should be included in models of cells below the limiting current.

Bürgi *et al.*[27] present a model for mass-transfer controlled metal deposition in radial thin-layer cells, which are in the pilot-plant stage of development by BASF for electro-organic syntheses,[1,2,16,40-43] such as the production of adiponitrile.[16,40] A radial cell is similar to a channel cell, but the average linear velocity decreases with radial distance from the central entry port.

Other variations on the channel flow cell have been proposed. The "Swiss roll" cell consists of rolled-up nickel sheet electrodes and separators placed in a cylindrical hous-

ing. This cell is used commercially to carry out one of the reactions in the synthesis of vitamin C.[44] Another variation is the "pump cell" proposed by Fleischmann, Jansson, *et al.*[14,15,45,46]

1.2. A Mathematical Model of Thin-Gap Channel Flow Cells

1.2.1. Assumptions

Presented here is a multiple-reaction model of channel flow cells that does not invoke the assumption of thin diffusion layers or of straight current lines. The model accounts for mass transfer, electrode kinetics and thermodynamics, and ohmic potential drop. This model, however, does not include the effect of migration; thus it is only valid for systems with excess supporting electrolyte. Because most industrial processes use excess supporting electrolyte to decrease the cell resistance, migration of the reacting species is often negligible. The other assumptions in the model are that there are no homogeneous reactions or gas-evolving reactions, there is no separator, the fluid is in fully-developed laminar flow (the Reynolds number must be less than 2100), axial diffusion is negligible (true for Péclet number > 10), the fluid properties are constant, and the electrode kinetics are described by the Butler-Volmer equation. Although other kinetic equations could have been used, the Butler-Volmer equation is convenient, and it applies to most electrochemical reactions.

1.2.2. Governing Equations

Figure 1-1 shows the channel geometry. L is the electrode length, h is the gap thickness, and $B=h/2$ is the channel halfwidth.

The governing equations and their underlying assumptions are discussed below. In systems with excess supporting electrolyte, the potential in the solution, as measured by a reference electrode of a given kind, satisfies Laplace's equation[28]

$$\frac{\partial^2\Phi}{\partial x^2} + \frac{\partial^2\Phi}{\partial y^2} = 0 \,, \tag{1-3}$$

with boundary conditions

$$\frac{\partial\Phi}{\partial y} = 0 \text{ at } y{=}{-}B \text{ and } y{=}B \qquad x < 0 \text{ and } x > L \tag{1-4a}$$

and

$$\frac{\partial\Phi}{\partial y} = -\frac{i_{an}(x)}{\kappa_\infty} \text{ at } y{=}{-}B \qquad 0 < x < L \,, \tag{1-4b}$$

$$\frac{\partial\Phi}{\partial y} = \frac{i_{cath}(x)}{\kappa_\infty} \text{ at } y{=}B \qquad 0 < x < L \,, \tag{1-4c}$$

where $y{=}{-}B$ is the location of the anode and κ_∞ is the feed-solution conductivity. The normal current density i_{an} is taken to be positive, and i_{cath} is negative (current flows from the solution into the cathode).

The solution to equation 1-3, evaluated near the cathode surface, is[47]

$$\Phi^o_{cath}(x) = \Phi^* - \frac{1}{2\pi\kappa_\infty}\left\{\int_0^L\left[i_{cath}(x')\ln\sinh^2\left(\frac{\pi(x-x')}{2h}\right) + i_{an}(x')\ln\cosh^2\left(\frac{\pi(x-x')}{2h}\right)\right]dx'\right\}, \tag{1-5}$$

where Φ^* is an integration constant. This equation can be rewritten

$$\Phi^o_{cath}(x) = \Phi^* - \frac{1}{2\pi\kappa_\infty}\int_0^L [i_{cath}(x') + i_{an}(x')]\ln\cosh^2\left(\frac{\pi(x-x')}{2h}\right)dx' - \frac{1}{2\pi\kappa_\infty}\int_0^L i_{cath}(x')\ln\tanh^2\left(\frac{\pi(x-x')}{2h}\right)dx' \,. \tag{1-6}$$

This form is convenient for evaluating the potentials numerically and for evaluating the potentials infinitely far upstream and downstream. The expression for the potential near the anode surface is the same as equations 1-5 and 1-6, but with the subscripts reversed.

The integration constant, Φ^*, is determined by the requirement of equal total currents on the two electrodes:

$$\int_0^L [i_{an}(x') + i_{cath}(x')]\,dx' = 0\ , \tag{1-7}$$

where the total current on an electrode is the sum of the partial current densities from each electrode reaction, for example,

$$i_{an}(x) = \sum_j i_{j,an}(x)\ . \tag{1-8}$$

The flux of a species i is given, in the absence of migration, by Fick's law. Because the governing equations are linear, the Fick's-law expression can be rewritten as a superposition integral[48-50] involving the step-function-boundary-condition solutions to the convective diffusion equation without axial diffusion (see chapter 2):

$$\begin{aligned} N_{i,cath}(x) = &-\frac{D_i}{B}\int_0^x \frac{dc_{i,cath}}{dx}\Big|_{x'} \frac{\partial\theta}{\partial\xi}(\varsigma-\varsigma',\xi=-1)\,dx' \\ &+\frac{D_i}{B}\int_0^x \frac{dc_{i,an}}{dx}\Big|_{x'} \frac{\partial\theta}{\partial\xi}(\varsigma-\varsigma',\xi=1)\,dx'\ , \end{aligned} \tag{1-9a}$$

and

$$\begin{aligned} N_{i,an}(x) = &\frac{D_i}{B}\int_0^x \frac{dc_{i,an}}{dx}\Big|_{x'} \frac{\partial\theta}{\partial\xi}(\varsigma-\varsigma',\xi=-1)\,dx' \\ &-\frac{D_i}{B}\int_0^x \frac{dc_{i,cath}}{dx}\Big|_{x'} \frac{\partial\theta}{\partial\xi}(\varsigma-\varsigma',\xi=1)\,dx'\ . \end{aligned} \tag{1-9b}$$

Both fluxes are in the $+y$ direction. The dimensionless variables are defined as

$$\theta = \frac{c_i - c_f}{c_{i,o} - c_f} \tag{1-10}$$

$$\varsigma = x\,\frac{D_i}{\frac{3}{2}B^2 \langle v \rangle} \tag{1-11}$$

$$\xi = y/B\ , \tag{1-12}$$

where c_f is the feed concentration of species i and $c_{i,o}$ is the surface concentration of

species i. Here, the cathode is located at $y = -B$, and the anode is located at $y = B$. The functions $\partial\theta/\partial\xi$ at $\xi = -1$ and $\partial\theta/\partial\xi$ at $\xi = 1$, representing solutions to the problem with a step-change on one electrode, are presented in reference 51[†] and in chapter 2.

The flux is also given by Faraday's law:

$$N_i = -\sum_j \frac{s_{ij}\, i_j}{n_j F} \,. \tag{1-13}$$

The sum is over the electrode reactions, which can be written in the form

$$\sum_i s_{ij}\, M_i^{\,z_i} \rightarrow n_j\, e^- \,,$$

where n_j denotes the number of electrons transferred, M_i is a symbol for the chemical formula of species i, and z_i is the charge number of species i.

The current density, i_j, depends exponentially on the surface overpotential according to the Butler-Volmer kinetic expression:

$$i_j = i_{o,j}\left[\exp\left(\frac{\alpha_{a,j}F}{RT}\,\eta_{s,j}\right) - \exp\left(\frac{-\alpha_{c,j}F}{RT}\,\eta_{s,j}\right)\right], \tag{1-15}$$

where $i_{o,j}$ is the exchange current density for reaction j and depends on the surface concentrations c_{io}. Usually, this dependence is expressed as

$$i_{o,j} = i_{oj,ref} \prod_i \left(\frac{c_{io}}{c_{i,ref}}\right)^{\gamma_{ij}} \,, \tag{1-16}$$

where $i_{oj,ref}$ is the exchange current density evaluated at reference concentrations, which should be selected to be nonzero. Newman[28] gives a set of rules for estimating γ_{ij}, based on the stoichiometry and transfer coefficients of the reaction.

The surface overpotential for reaction j, $\eta_{s,j}$, is the potential of the working electrode relative to a reference electrode of the same kind as the working electrode, placed in the solution adjacent to the surface of the working electrode, but just outside the

[†]Reference 51 presents the Nusselt number, which is proportional to $\partial\theta/\partial\xi$. Unfortunately the Nusselt number was

diffuse double layer. Figure 1-2 illustrates the placement of reference electrodes for assessing potential variations in a solution.[52] The letter s designates reference electrodes of the same kind as the working electrode, and the letter g refers to reference electrodes of a given kind. Using the notation of figure 1-2, the surface overpotential for a reaction j is written

$$\eta_{s,j} = V - V_{rls} = V - V_{rlg} - (V_{rls} - V_{rlg}) , \tag{1-17a}$$

or

$$\eta_{s,j} = V - \Phi^o - U_{j,o} , \tag{1-17b}$$

where $\Phi^o = V_{rlg}$ is the potential just outside the double layer, as measured by a reference electrode of a given kind, and

$$U_{j,o} = V_{rls} - V_{rlg}$$
$$= \left[U_j^\theta - \frac{RT}{n_j F} \sum_i s_{ij} \ln\left(\frac{c_{io}}{\rho_o}\right)\right] - \left[U_{re}^\theta - \frac{RT}{n_{re} F} \sum_i s_{i,re} \ln\left(\frac{c_{i,re}}{\rho_o}\right)\right] . \tag{1-18}$$

The subscript re denotes the reference-electrode reaction. The electrode potential V in equation 1-17 is specified with respect to an arbitrary zero of potential. For example, if V_{cath} is set equal to zero, then V_{an} is equal to the total cell potential, which is specified in this problem formulation. Note that the average current density on the electrodes is not set here but is calculated from

$$< i > = \frac{1}{L} \int_0^L i_{an}(x)\, dx . \tag{1-19}$$

A summary of the equations and unknowns for i species and j reactions is given in table 1-1. The left column shows the number of equations and whether the equation is

based on the channel gap (h) instead of the equivalent diameter ($2h$).

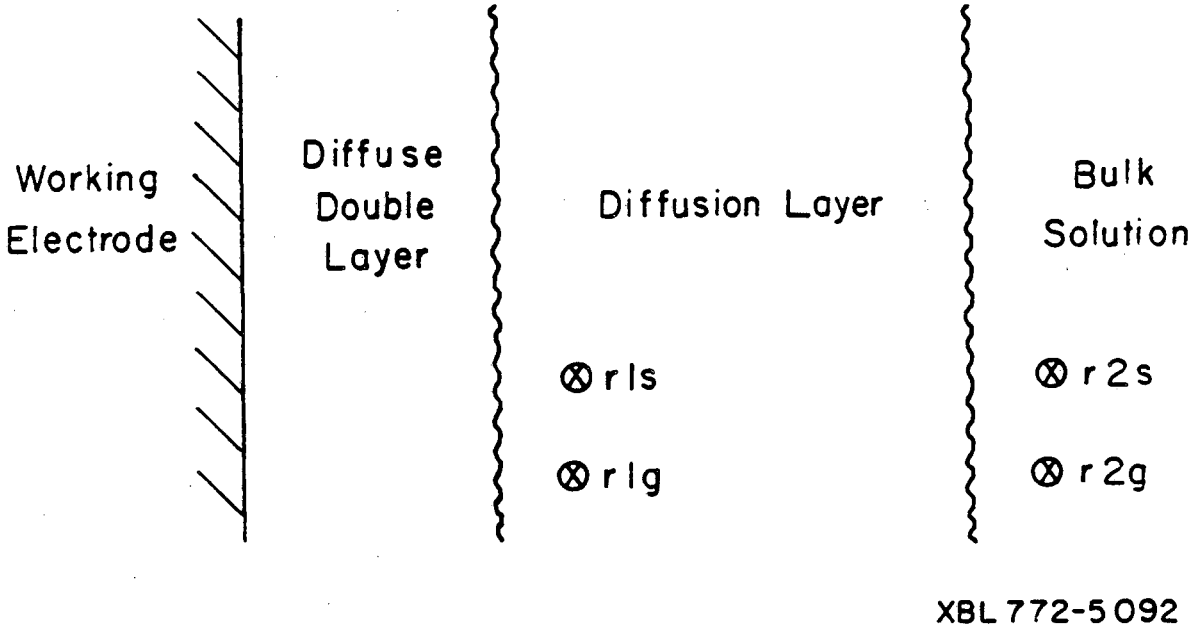


Figure 1-2. Reference electrodes, which may be imaginary, positioned in the bulk solution and within the diffusion layer.[50] The letter *s* designates reference electrodes of the same kind as the working electrode, and the letter *g* refers to reference electrodes of a given kind.

Table 1-1

Equations		Unknowns	
$2i(x)$	Fick's Law (1-9a,b)	$2i(x)$	Fluxes
$2i(x)$	Faraday's Law (1-13)	$2i(x)$	Surface Concentrations
$2j(x)$	Butler-Volmer (1-15,16)	$2j(x)$	Partial Currents
$2j(x)$	Surface Overpotential (1-17b)	$2j(x)$	Surface Overpotentials
$2j(x)$	$U_{j,o}$ (1-18)	$2j(x)$	$U_{j,o}$
$2(x)$	$\Phi^o(x)$ (1-5)	$2(x)$	$\Phi^o(x)$
1	Equal Currents (1-7)	1	Φ^*
$2(x)$	Total Currents (1-8)	$2(x)$	Total Currents
1	Cell Potential	1	V_{an}
1	Zero of Potential	1	V_{cath}

evaluated at each axial position, x.

1.3. Dimensionless Formulation

It is useful to write the equations in table 1-1 in dimensionless form, because fewer parameters are needed to define each problem. For a system with i species and j reactions, $(2 + 2ij + 4i + 4j)$ dimensionless parameters are needed. These are parameters characterizing the electrode reactions, the geometry, the transport properties, and the operating conditions. Table 1-2 lists the dimensionless parameters. The parameters J_j, N, and Gz are defined as follows:

$$J_j = \frac{-n_m}{s_{Rm}} \left(\frac{FL}{RT\kappa_\infty} \right) i_{oj,ref} , \tag{1-20}$$

$$N = \frac{n_m^2}{s_{Rm}^2} \frac{F^2 D_R c_{R\infty}}{RT\kappa_\infty} \left(\frac{6 < v > L^2}{hD_R} \right)^{1/3} , \tag{1-21}$$

$$Gz = \frac{\pi}{4} ReSc \frac{d_e}{L} = \frac{\pi d_e^2 < v >}{4 D_R L} . \tag{1-22}$$

As the Graetz number, Gz, approaches infinity (for example, high flowrate), the thin-diffusion-boundary-layer approximation becomes valid. As $Gz \rightarrow 0$, the boundary layer becomes thick.

The reader should note that the definition of the Graetz number in equation 1-22 differs from that used in the literature.[53,54] Based on Graetz's original work for mass transfer in a tube,[55] Drew[56] and others began to use the quantity $A < v > / DL$[†] to describe mass transfer in other geometries, where A is the flow cross-sectional area. By 1942 this quantity was being called a Graetz number.[54] For a tube, this presents no difficulty, because it reduces to

$$Gz_{tube} = \frac{\pi d_e^2 < v >}{4DL} . \tag{1-23}$$

For an annulus, however, this Graetz number becomes

$$\begin{aligned} Gz_{annulus} &= \frac{\pi (d_2^2 - d_1^2) < v >}{4DL} \\ &= \frac{\pi d_e^2 (1 + \kappa) < v >}{4DL (1 - \kappa)} , \end{aligned} \tag{1-24}$$

where κ is the ratio of the inner to the outer radius. As $\kappa \rightarrow 1$, the parallel plate geometry is approached, but, according to this definition, the Graetz number approaches infinity; therefore the square of the hydraulic diameter should appear in

[†] wC_p/kL in heat-transfer nomenclature.

place of the actual cross-sectional area. To be consistent with the definition of the Graetz number for a tube, we include the factor $\pi/4$ in equation 1-22.

1.4. Solution Techniques

In this section, two approaches and algorithms are presented, along with some details, such as mesh spacing and integration techniques. The first approach is a method of successive substitution applied to the potential distribution, and the second approach is a method of collocation with a Newton-Raphson iteration on the coefficients. The advantages and disadvantages of each method are listed in section 1.4.1.

Clearly, the assumptions incorporated into the governing equations of any mathematical model influence the choice of a numerical technique. For example, White *et al.*[18] assume straight current lines; therefore their partial differential equations are parabolic. This allows them to use a straightforward technique, in which they use implicit stepping in the x direction, and solve the boundary-value problem in the y variable by Newman's method.[28,57]

Although the choice of a numerical technique is influenced by the nature of the equations, the technique is not uniquely determined because there is often a choice of methods. In the present model, for example, Laplace's equation for the potential is not parabolic, and the implicit stepping technique cannot be used. White *et al.*[18] approach this problem by using an implicit alternating direction (IAD) technique. Here, we approach the problem with two methods, both of which use the known analytic solution to Laplace's equation. Our methods, however, cannot be used to solve White's nonlinear equations with migration and homogeneous reactions.

The first numerical method to be discussed here is similar to that of Parrish and Newman,[21] and the second (see appendix D) is similar to the orthogonal collocation

method used by Cabán and Chapman.[22] The motivation for both approaches is to take advantage of the linearity of the governing equations by using superposition integrals for the solution of Fick's second law of diffusion. The power of the superposition technique is that it reduces the original two-dimensional problem to a one-dimensional problem. (The y-dependence is eliminated.) Note that although the detailed concentration profiles within the channel are not obtained, one can still obtain the cup-mixing average concentration (or conversion) at the exit, or any other axial position, by performing a material balance.

Parrish and Newman's scheme[21] is to alternate between revising the surface concentrations and the potential distribution. The details of our procedure are slightly different from those of Parrish and Newman, however, because the electrodes cannot be treated separately, and because there are multiple reactions.

The procedure now contains four nested iterative loops to solve the equations in table 1-1: the innermost loop takes a given overpotential distribution and solves the flux equations for the anode and cathode surface concentrations and currents at each axial mesh point; the second loop chooses the integration constant from equation 1-5 to satisfy the condition that the total currents are equal on the anode and cathode; the next loop finds the average current density; and the outermost loop iterates on the potential distribution until the distribution used to calculate surface concentrations agrees with that calculated from the solution to Laplace's equation. The loop for average current density is not needed in principle, but it was added to promote stability in the iterations.

Figures 1-3 and 1-4 show the iterative scheme summarized above. Figure 1-3 is a detailed picture of the order for solving the governing equations, and figure 1-4 shows how the potential distributions are damped between iterations.

Before examining figure 1-3 in detail, one must first understand the physical

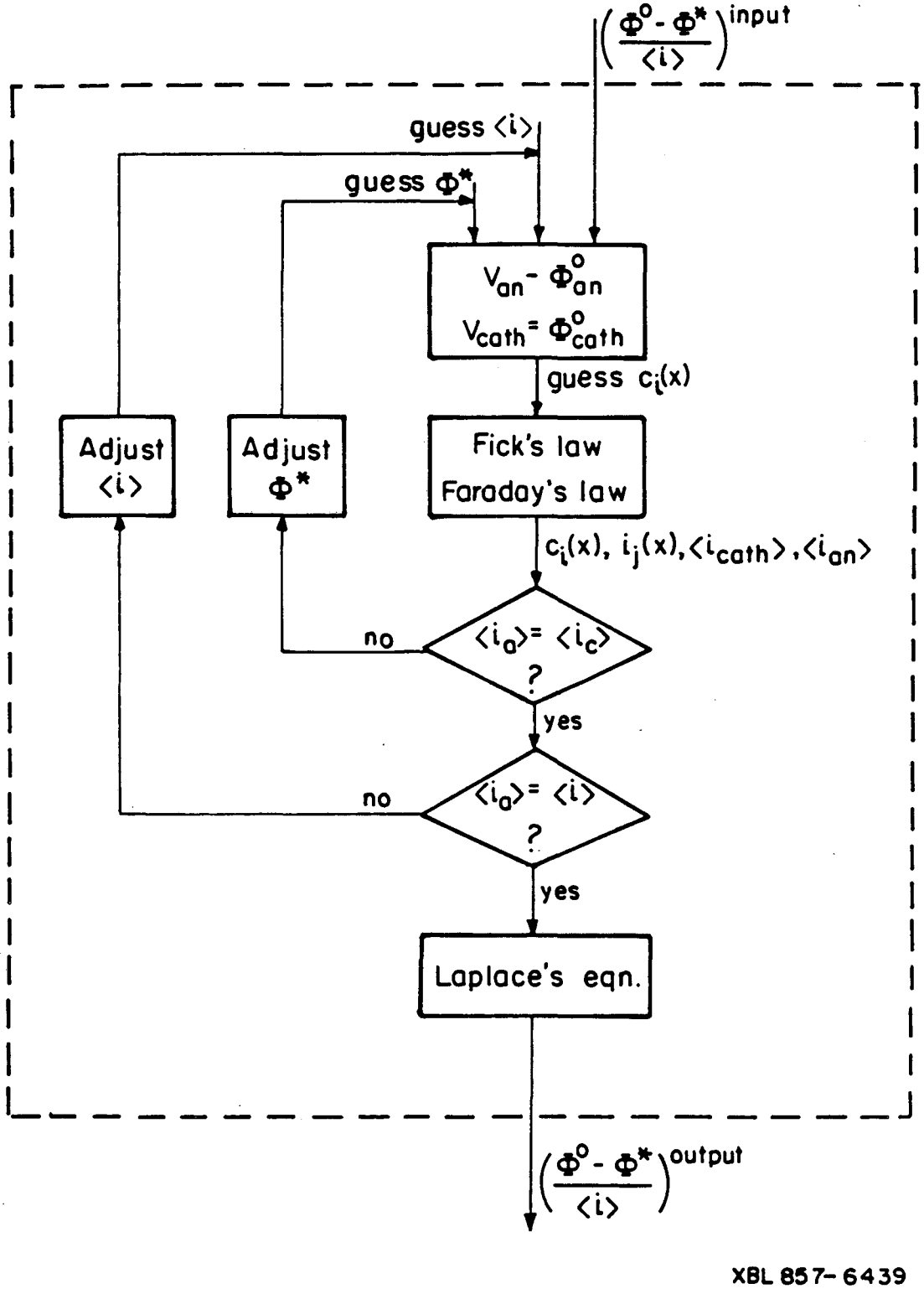


Figure 1-3. Scheme for calculating a new potential distribution.

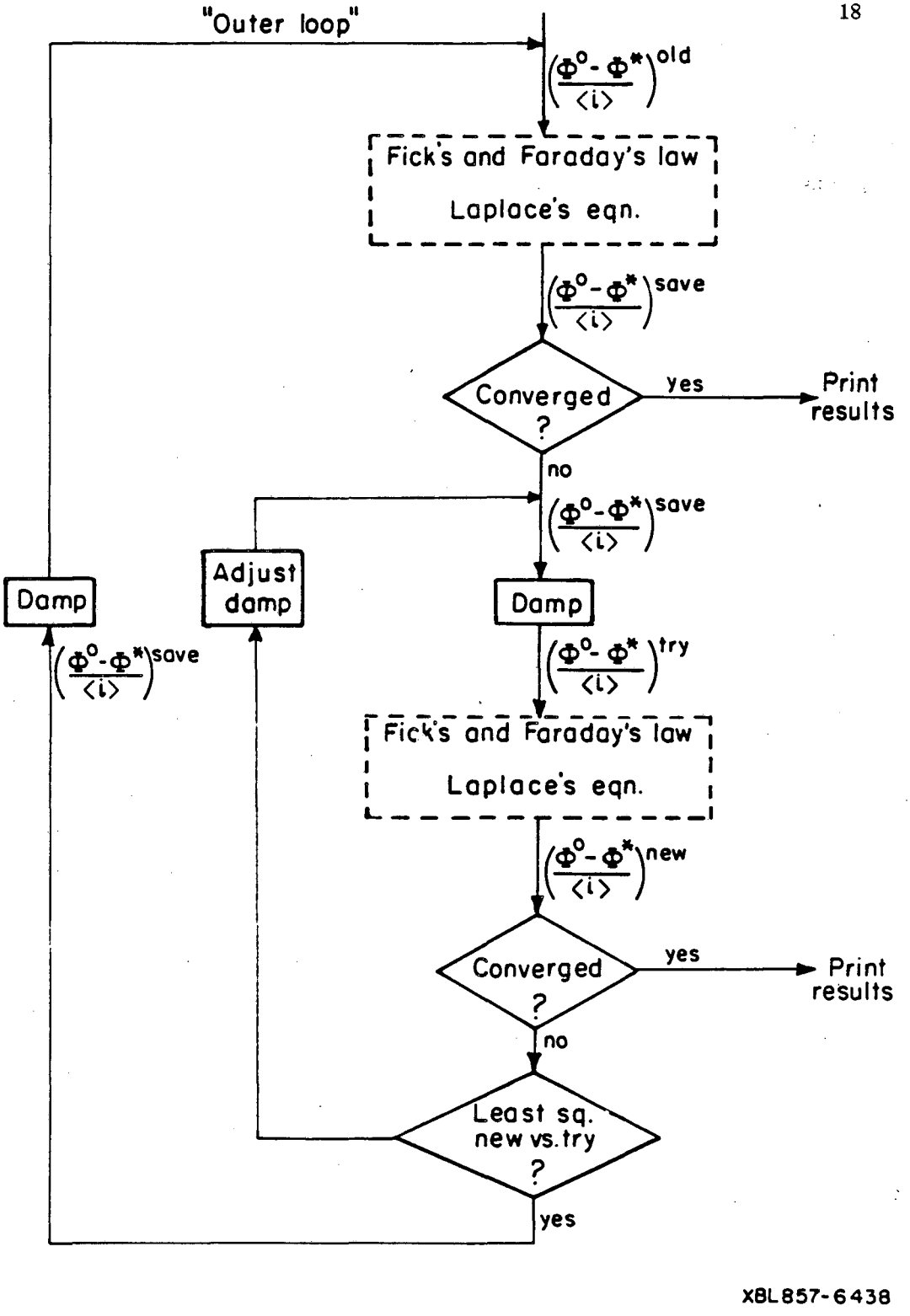
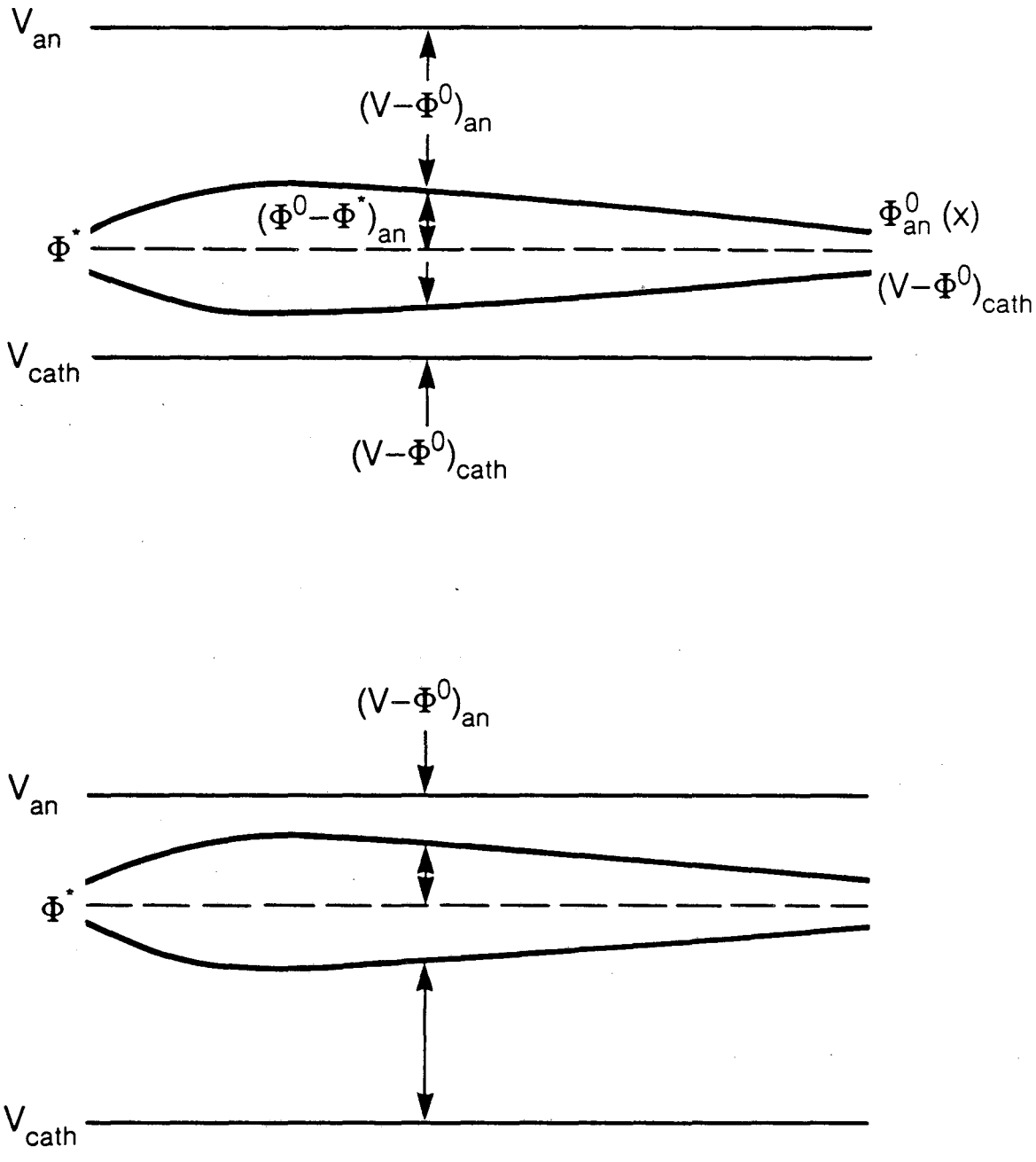


Figure 1-4. Damping scheme.

significance of the integration constant, Φ^*, and the average current density, $<i>$. Note that in equation 1-5, if one subtracts Φ^* from both sides and divides through by $<i>$, the left side is $(\Phi^o - \Phi^*)/<i>$, which represents the shape of the potential distribution; the right side shows how this shape depends on the shape of the current distributions, $i(x)/<i>$. The integration constant, Φ^*, in equation 1-5 sets the relative amounts of overpotential on the anode and cathode. For example, if Φ^* is too close to V_{an}, the total overpotential $(V - \Phi^o)$ on the anode will be too small, and the integral of $i_{an}(x)$ will be less in magnitude than the integral of $i_{cath}(x)$ (see figure 1-5). Thus, Φ^* is adjusted to satisfy the condition that the currents be equal on the two electrodes. The average current, $<i>$, sets the magnitude of the overpotentials. For example, if $<i>$ is too high, the overpotentials calculated with the Φ^o from equation 1-5 will be small, and, hence, the currents calculated from these overpotentials will be too small to agree with the input $<i>$ (see figure 1-6). In summary, $(\Phi^o - \Phi^*)/<i>$ describes the shape of the potential distribution, Φ^* determines the placement of Φ^o_{an} and Φ^o_{cath} relative to the electrode potentials, and $<i>$ sets the magnitude of $(V - \Phi^o)$.

We can now discuss the entire scheme of figure 1-3 in detail. Proceeding from top to bottom on figure 1-3, first we make a guess for the potential distributions and for Φ^* and $<i>$. Recall that the electrode potentials, V_{an} and V_{cath}, are specified. The potential distributions $\Phi^o_{an}(x)$ and $\Phi^o_{cath}(x)$ are then substituted into the definition of surface overpotential (1-17b with 1-18). Equation 1-17b is, in turn, substituted into the Butler-Volmer kinetic expression (1-15 with 1-16) to give each of the partial current densities, i_j. The partial current densities, $i_j(x)$, are then used in the Faraday's-law flux expression (1-13). The resulting Faraday's-law expression, therefore, contains the guessed potential distribution and unknown surface concentrations (arising from the concentration dependence of the exchange current density). For convenience, the concentrations may be brought outside the exponentials.[58] On the cathode, for example,



XBL 861-8627

Figure 1-5. Effect of Φ^* on the potential distribution.

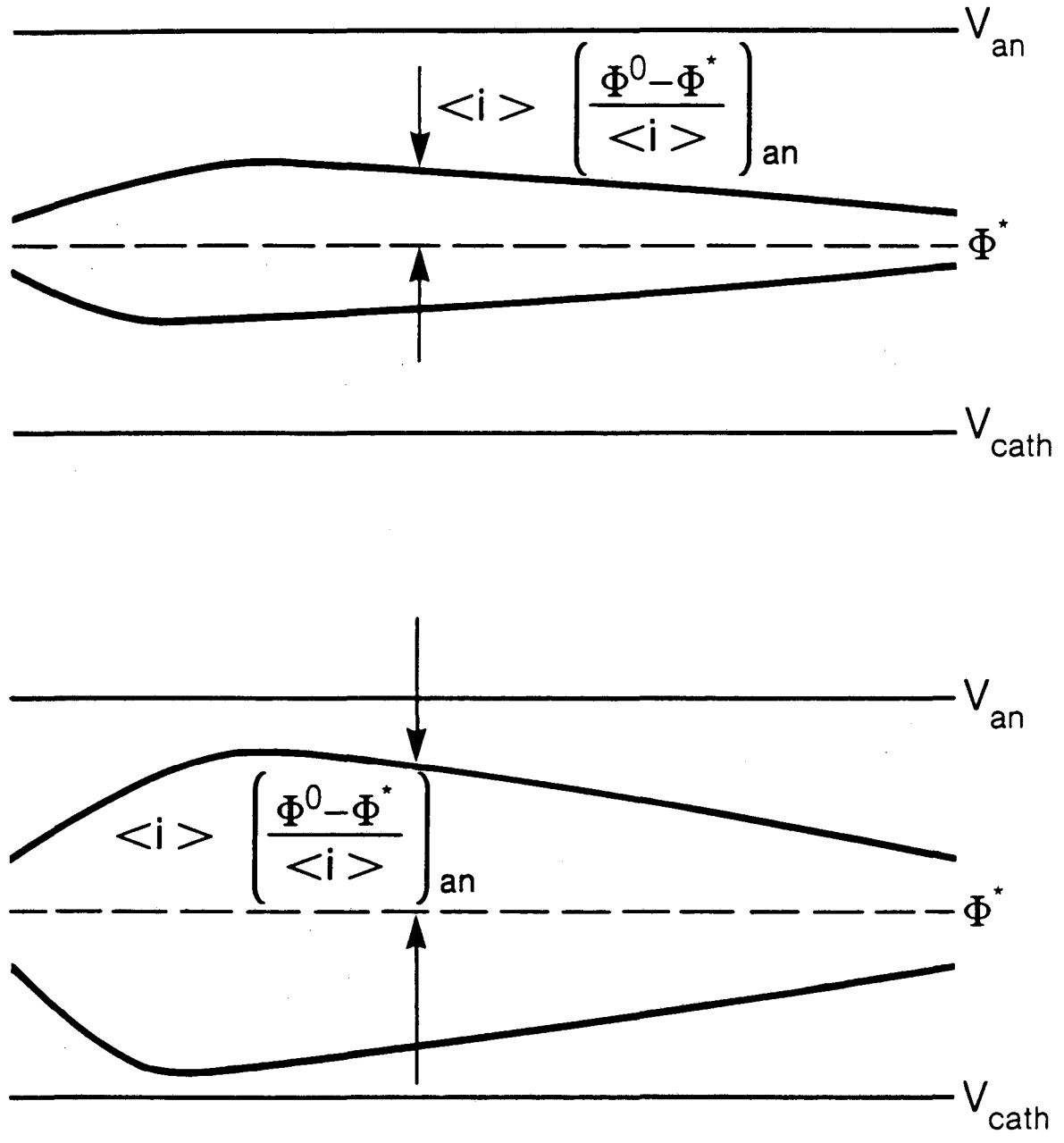


Figure 1-6. Effect of $<i>$ on the potential distribution.

$$N_{i,cath}(x) = -\sum_j \frac{s_{ij}}{n_j F}\, i_{oj,ref} \prod_k \left(\frac{c_{ko,cath}}{c_{k,ref}} \right)^{p_{kj}} \exp\left[\frac{\alpha_{aj} F}{RT} \left(V_{cath} - \Phi^o{}_{cath} - U_{j,ref} \right) \right]$$

$$+ \sum_j \frac{s_{ij}}{n_j F}\, i_{oj,ref} \prod_k \left(\frac{c_{ko,cath}}{c_{k,ref}} \right)^{q_{kj}} \exp\left[\frac{-\alpha_{cj} F}{RT} \left(V_{cath} - \Phi^o{}_{cath} - U_{j,ref} \right) \right], \quad \text{(1-25)}$$

where $U_{j,ref}$ is equation 1-18 with c_{io} replaced by $c_{i,ref}$,

$$p_{kj} = \gamma_{kj} + \alpha_{aj} \frac{s_{kj}}{n_j}, \quad \text{(1-26a)}$$

and

$$q_{kj} = \gamma_{kj} - \alpha_{cj} \frac{s_{kj}}{n_j}. \quad \text{(1-26b)}$$

The Fick's-law flux expression, developed in chapter 2, is discretized by the method of Acrivos and Chambré,[59] and it contains unknown local surface concentrations and the known upstream concentrations. By equating the Fick's-law and Faraday's-law expressions for the flux of each species on each electrode, one obtains, at each mesh point, a set of equations to be solved for the unknown surface concentrations, $c_i(x)$. These equations are solved by a multi-dimensional Newton-Raphson method. The spacing of the mesh points is fine near the leading edge but coarse downstream, where the profiles are generally smooth (see section 1.4.2). To speed the convergence of the Newton-Raphson method, the initial guesses are made for the surface concentrations by extrapolating linearly from the previous two axial positions. Let it be emphasized that the unknowns solved for simultaneously are the concentrations only at the present mesh point. Also, the electrodes do not interact at the present mesh point; rather the interaction propagates downstream from the already-solved mesh points.

After solving for the surface concentrations, Φ^* and $<i>$ are adjusted. The $<i>$ loop was added to promote stability in the outer potential-distribution loop. Φ^* is adjusted by a one-dimensional search to satisfy the condition that the currents are equal

on the two electrodes. $<i>$ is likewise adjusted to ensure that the magnitude of the current calculated from the kinetic expression agrees with the current that was substituted into the solution to Laplace's equation (equation 1-5).

The new current distributions from these inner loops are then substituted into Laplace's equation to calculate a new potential distribution. With the new potential distribution, one can check for convergence by comparing with the original guess.

Two possible ways to iterate on the potential distribution are the multi-dimensional Newton-Raphson method and successive substitution with damping. The relative merits of these two methods are discussed in appendix D and references 60 through 62. For this complex problem, successive substitution was chosen. A damping factor averages the projected potential distribution with that from the previous iteration:

$$\Phi^o = (damp)\Phi^o_{old} + (1-damp)\Phi^o_{new}. \qquad (1\text{-}27)$$

The damping factor must be chosen carefully, because if the damping factor is too small, the potential distribution oscillates from iteration to iteration, but if it is too large, convergence is too slow or is never achieved at all. Since the program does not converge well when the same damping factor is used for all iterations, the damping factor must be adjusted as the iteration proceeds. The criterion for this adjustment is that the optimum damping factor minimizes the mean square difference between the new potential distribution obtained with the damping factor and the distribution that would result on the subsequent iteration if that damping were then removed. The computer program finds the optimum damping factor by choosing initially three damping factors and fitting a parabola to the resulting curve (sum-of-squares deviation *vs.* damping factor). When the computer program has found the optimum damping from the fitting function, it proceeds to the next iteration. The iteration continues until the sum-of-squares deviation is as small as desired.

The damping scheme summarized above is shown in figure 1-4. First, the projected

potential distribution is calculated by the procedure shown in figure 1-3. This procedure is denoted by dotted boxes in both figures 1-3 and 1-4. The calculated distribution is stored in $(\Phi^o - \Phi^*)/<i>^{save}$.

Next, after checking for convergence, the optimum damping factor is sought. The lower part of figure 1-4 shows that a damping factor is applied to the saved distribution, and that damped distribution, denoted $(\Phi^o - \Phi^*)/<i>^{try}$, is used as input to calculate $(\Phi^o - \Phi^*)/<i>^{new}$. When the sum-of-squares difference between $(\Phi^o - \Phi^*)/<i>^{try}$ and $(\Phi^o - \Phi^*)/<i>^{new}$ is a minimum, the optimum has been found. When the optimum damping factor has been found, it is applied to the saved potential distribution, and the old potential distribution is updated.

Although a successful computer run can take more than 40 outer iterations and several hundred seconds on a CDC-7600 computer, the run time can be reduced to less than 100 seconds by using previous runs to provide the initial guesses for the potential distributions. This technique is particularly effective when performing a series of runs to study the effects of a given parameter.

Here, we shall briefly mention the second method for solving the equations of section 1.2; more details are given in appendix D (section D.5). The technique, which is similar to the collocation method of Cabán and Chapman,[22] is to express the partial current densities as superpositions of trial functions and to solve for the coefficients by a Newton-Raphson method.

1.4.1. Advantages and Disadvantages of the Two Solution Techniques

One advantage of the first scheme, particularly as originally conceived, is that the potentials are calculated by successive substitution instead of by the more time-consuming Newton-Raphson iteration. In addition, it is not necessary to invent good approximating functions and choose appropriate collocation points.

The disadvantages of the Parrish and Newman scheme are that it is time-consuming and convergence is hardly guaranteed. Although the damping-factor adjustment scheme is of some help, the program often finds that the optimum damping is complete damping, so that no movement is made toward the answer.

Another problem is that it is difficult to evaluate accurately the integrals for Laplace's equation because one must interpolate to find the currents at the quadrature roots (Parrish and Newman[21] found that Simpson's rule was less accurate than Gaussian quadrature with Lagrange interpolation). This problem is particularly acute for large aspect ratios (L/h) because the current decays rapidly in the edge region, making the interpolation difficult. In spite of the small size of the edge region, it plays an important role in determining the potential distribution, both locally and globally.

The collocation method does not share the problem of inaccurate interpolation because known current functions are used. The main advantage of the collocation method, however, is that the time-consuming integral equations (Laplace's equation and the superposition integrals) are evaluated only once at the start of the calculations instead of repeatedly inside the iteration loops. This savings is possible because of the linearity of the equations.

The major disadvantage of the collocation method is that the accuracy of the results depends on the choice of trial functions and the placement of collocation points; it is not sufficient to use a simple collection of orthogonal polynomials. Another disadvantage is that limiting-current problems must be treated separately and in a more complex manner (see appendix D).

1.4.2. Mesh Spacing

The superposition method described above is most easily carried out for evenly-spaced mesh points. Evenly-spaced points are unsuitable, however, because the concen-

tration profiles can change rapidly near the leading edge, but are smooth away from the leading edge. Therefore the mesh points should be finely spaced near the leading edge, but coarse downstream to avoid unnecessary calculations. For some problems, this difficulty is overcome by using, for example, logarithmically-spaced mesh points. Since this problem has integral equations, it is easiest to use the method of Acrivos and Chambré [59] and keep the mesh points evenly spaced. Closely-spaced mesh points are obtained near the leading edge by dividing the interval into "decades" and using evenly-spaced points inside the decades.[63] For example, if the axial-position variable ranges between 0 and 1, one can solve the problem between 0 and 0.1 with, say, 51 mesh points. One then discards all the mesh points except those corresponding to x= 0, 0.02, 0.04,..., 0.1, and proceeds to solve the problem from x=0.1 to x=1 with the remaining 45 mesh points. This idea can also be applied to the subdivisions to achieve even finer spacing near the leading edge. Note that the interval need not be divided into tenths; we found that it is best to divide the interval in half, because at x=0.5, the profiles are usually smooth, and there will not be a large discontinuity resulting from a drastic change in mesh size.

1.4.3 Integration Methods

Several integrals must be evaluated numerically. These are the integrals for the potential distribution (equation 1-6), the average current density (1-19), and the discretized superposition integrals (from 1-9a and 1-9b). A different method is chosen for each integral, according to its behavior.

The integrals for the potential distribution are evaluated by eliminating the singularity at x=x' by a method similar to that of Kantorovich and Krylov[64] and stretching the coordinates to smooth the integrand at the electrode edges.

The method of Kantorovich and Krylov is to write

$$\int_0^1 i_{cath}(x') \ln \tanh^2 \left[\epsilon(x-x')\right] dx' = i_{cath}(x) \int_0^1 \ln \tanh^2 \left[\epsilon(x-x')\right] dx'$$

$$+ \int_0^1 [i_{cath}(x') - i_{cath}(x)] \ln \tanh^2 \left[\epsilon(x-x')\right] dx', \quad (1\text{-}28)$$

with

$$\epsilon = \frac{\pi L}{2h}, \quad (1\text{-}29)$$

so that the integrand containing the unknown i_{cath} vanishes at $x=x'$ (because the singularity is logarithmic). Since the ln $\tanh^2$ integral is still difficult to evaluate, we add and subtract instead a term that can be integrated analytically, but behaves like ln $\tanh^2$ near the singularity:

$$\int_0^1 i_{cath}(x') \ln \tanh^2 \left[\epsilon(x-x')\right] dx' = i_{cath}(x) \int_0^1 \ln \left[\epsilon(x-x')\right]^2 dx'$$

$$+ \int_0^1 \left[i_{cath}(x') \ln \tanh^2 \left[\epsilon(x-x')\right] - i_{cath}(x) \ln \left[\epsilon(x-x')\right]^2 \right] dx'. \quad (1\text{-}30)$$

The first term in this equation is integrated analytically and the second term by Gaussian quadrature with stretching of the coordinates (to be discussed shortly).

The ln $\cosh^2$ term from equation 1-6 is also evaluated by Gaussian quadrature, but a correction term is added to ensure that the currents are exactly equal on the two electrodes:

$$\int_0^1 [i_{cath}(x') + i_{an}(x')] \ln \cosh^2 \left[\epsilon(x-x')\right] dx'$$

$$= \int_0^1 [i_{cath}(x') + i_{an}(x') - \langle i_{cath} + i_{an} \rangle] \ln \cosh^2 \left[\epsilon(x-x')\right] dx'$$

$$= \int_0^1 [i_{cath}(x') + i_{an}(x')] \ln \cosh^2 \left[\epsilon(x-x')\right] dx'$$

$$- \langle i_{cath} + i_{an} \rangle \int_0^1 \ln \cosh^2 \left[\epsilon(x-x')\right] dx', \quad (1\text{-}31)$$

where

$$< i_{cath} + i_{an} > = \int_0^1 [i_{cath}(x') + i_{an}(x')]\ dx'. \tag{1-32}$$

Note that $< i_{cath} + i_{an} >$ is zero in principle, but in practice it is equal to a convergence tolerance, such as 10^{-6}.

The stretching of the coordinates mentioned earlier is to let

$$Y = (x')^n \tag{1-33}$$

so that the integral becomes

$$\int_{x_{min}}^{x_{max}} i(x')\, f(x - x')dx' = \int_{(x_{min})^n}^{(x_{max})^n} i(x') \frac{Y^{1/n - 1}}{n} f(x - x')\ dY. \tag{1-34}$$

The choice $n = 1/2$ eliminates the $x^{-1/2}$ singularity in $i(x)$ for the primary distribution. Even for more uniform current distributions, the integrand $i(x')\sqrt{x'}f(x - x')$ is well-behaved. A similar stretching is done to eliminate possible singularities near x=1. The integral is broken up into regions, and the appropriate stretching is used in each one.

The integrals for the average current density are evaluated, using the finely-spaced mesh points, by Simpson's rule. The integral from the first mesh point to the second mesh point is treated separately to give more accurate results near the limiting current. In principle, the limiting current density is infinity at x=0 and decays as $x^{-1/3}$ (as long as the boundary layer is still thin). To avoid sensitivity to the large, somewhat arbitrary value of i_{cath} at the first mesh point (jx=1), the integral is evaluated as $1.5h\ i_{cath}(jx{=}2)$, where h is the mesh spacing. This form is accurate at the limiting current but will overestimate the integral, by a maximum of 50%, for more uniform current distributions. Recall, however, that this formula is applied only between the first and second mesh points, which are finely spaced.

The final integrals to be mentioned are the discretized superposition integrals. These are the most straightforward, since most of them can be evaluated analytically. Those that require numerical integration are integrated with the trapezoidal rule

because the discretized integrals are between adjacent mesh points.

1.5. Summary

A superposition-integral technique is used to solve for the concentration, current, and potential distribution in a thin-gap channel flow cell. The superposition technique takes advantage of the linearity of the governing equation by reducing the problem from two-dimensional to one-dimensional. The model includes multiple electrochemical reactions and interacting diffusion boundary layers but does not include migration, axial diffusion, homogeneous reactions, or turbulent flow.

The next chapter develops the solutions that are needed for the superposition integral, and chapter 3 presents theoretical results from the model.

Nomenclature for Chapter 1

A	flow cross-sectional area, cm^2
B	channel halfwidth, cm
$c_{i,an}$	anode surface concentration of species i, mol/cm^3
$c_{i,cath}$	cathode surface concentration of species i, mol/cm^3
$c_{i,ref}$	reference concentration of species i (see equation 1-12), mol/cm^3
c_{io}	surface concentration of species i, mol/cm^3
C_p	heat capacity, cm^2/s^2-K
d_1	inner radius of annulus, cm
d_2	outer radius of annulus, cm
d_e	equivalent diameter ($2h$ for a channel), cm
D	diffusion coefficient, cm^2/s
D_i	diffusion coefficient of species i, cm^2/s
$damp$	damping factor (see equation 1-23)
F	Faraday's constant, 96,487 C/eq

Gz	Graetz number (see equation 1-18 and discussion following)
h	interelectrode gap thickness, cm
$i(x)$	local current density, A/cm^2
$<i>$	average current density, A/cm^2
i_j	current density due to reaction j, A/cm^2
$i_{o,j}$	exchange current density for reaction j, evaluated at the local surface concentrations, A/cm^2
$i_{oj,ref}$	exchange current density for reaction j, evaluated at the reference concentrations, A/cm^2
J_j	dimensionless exchange current density defined in equation 1-16
jx	mesh point
k	thermal conductivity, g-cm/s^2-K
L	electrode length, cm
N	dimensionless parameter defined in equation 1-17 (represents dimensionless limiting current when boundary layers are thin)
N_i	flux of species i, mol/cm^2-s
n_j	number of electrons transferred in reaction j
p_{kj}	anodic reaction order for species k in reaction j

q_{jk}	cathodic reaction order for species k in reaction j
R	universal gas constant, 8.3143 J/mol-K
Re	Reynolds number ($2<v>h/\nu$ for a channel)
s_{ij}	stoichiometric coefficient for species i in reaction j
Sc	Schmidt number (ν/D)
T	absolute temperature, K
U_j^θ	standard electrode potential for reaction j, V
$U_{j,o}$	theoretical open-circuit potential for reaction j relative to a reference electrode of a given kind, evaluated at the surface concentrations, V
$U_{j,ref}$	theoretical open-circuit potential evaluated with reference concentrations, V
$<v>$	average fluid velocity, cm/s
V	electrode potential, V
w	mass flow rate, g/s
x	axial coordinate, cm
y	normal coordinate, cm or variable defined in equation 1-26b
z	dummy variable of integration in equation 1-25

Greek

$\alpha_{a,j}$	anodic transfer coefficient for reaction j
$\alpha_{c,j}$	cathodic transfer coefficient for reaction j
γ_{ij}	exponent for concentration dependence of exchange current for reaction j on species i
ϵ	$\pi L/(2h)$
ς	dimensionless axial coordinate defined in equation 1-8
$\eta_{s,j}$	surface overpotential for reaction j, V
θ	dimensionless concentration defined in equation 1-7
κ	ratio of inner to outer radius of annulus
κ_∞	conductivity of feed solution, $ohm^{-1}cm^{-1}$
Φ	potential in the solution as measured by a reference electrode of a given kind, V
Φ^o	potential in solution just outside diffuse double layer, V
Φ^*	integration constant in equation 1-3, V
ρ_o	pure solvent density, kg/cm^3
ξ	dimensionless normal coordinate defined in equation 1-9

Subscripts

an	anode
$cath$	cathode
f	feed
g	reference electrode of a given kind
i	species in solution
j	electrode reaction
k	dummy index for species
m	main reaction
o	electrode surface
R	principal reactant
$r1$	reference electrode positioned just outside the double layer
re	reference electrode reaction
ref	reference concentrations at which exchange current density $i_{oj,ref}$ was measured
s	reference electrode of the same kind as the working electrode
∞	feed solution

Superscripts

new	variables calculated from damped ("try") variables, see figure 1-4
old	variables from the previous iteration
save	variables calculated from "old" variables
try	weighted average of "old" and "save" variables
o	electrode surface
*	integration constant
θ	unit molality standard state

2. The Asymmetric Graetz Problem in Channel Flow

2.1. Introduction

The problem of mass transfer to fluids in laminar flow in ducts appears in many engineering applications. This problem has been solved for some special cases,[32,34,54,65-72] but here we shall consider the case of a flat duct, or channel, where the surface-concentration boundary conditions are arbitrary, and may differ on the two channel walls. This problem is of interest when the channel gap is thinner than the diffusion-boundary-layer thickness. In such cases, the fluxes at the two channel walls are not independent.

Since the detailed concentration profile within the flowing fluid is generally not needed, we shall emphasize the distribution of flux along the channel walls. Given this flux distribution, one can calculate the average concentration at the exit by performing an overall material balance.

To obtain the distribution of flux along the two channel walls, Duhamel's superposition principle[48-50] can be used to treat the nonlinear concentration boundary conditions.

2.2. Problem Statement

For laminar flow in a channel, with negligible axial diffusion, the dimensionless convective diffusion equation is

$$(1-\xi^2)\frac{\partial\theta}{\partial\varsigma} = \frac{\partial^2\theta}{\partial\xi^2}\,, \tag{2-1}$$

where

$$\theta = \frac{c_i - c_b}{c_o - c_b} \tag{2-2}$$

$$\xi = \frac{y}{B} \tag{2-3}$$

$$\varsigma = x \frac{D_i}{\frac{3}{2}B^2 \langle v_x \rangle} . \tag{2-4}$$

If the boundary conditions are arbitrary, Duhamel's superposition theorem can be used to write the flux in terms of the solution to the problem with a step-function concentration boundary condition on one wall. For example, if wall "-1" is located at ξ=—1, and wall "1" is located at ξ=1, then the flux of species i at wall "-1" is

$$N_{i,-1}(x) = -\frac{D_i}{B}\int_0^x \frac{dc_{i,-1}}{dx}\Big|_{x^*} \frac{\partial\theta}{\partial\xi}(\varsigma-\varsigma^*,\ \xi=-1)\ dx^* \tag{2-5}$$

$$+\frac{D_i}{B}\int_0^x \frac{dc_{i,1}}{dx}\Big|_{x^*} \frac{\partial\theta}{\partial\xi}(\varsigma-\varsigma^*,\ \xi=1)\ dx^*,$$

where $c_{i,-1}$ is the surface concentration of species i at ξ=—1 and $c_{i,1}$ is the concentration of species i at ξ=1. The flux (in the $+\xi$ direction) of species i at wall "1" is obtained by reversing the wall subscripts and the signs. In equation 2-5, $\theta(\varsigma,\xi)$ is the solution to equation 2-1 with boundary conditions

$$\theta = 1 \ at \ \xi = -1 \tag{2-6}$$

$$\theta = 0 \ at \ \xi = 1 \tag{2-7}$$

$$\theta = 0 \ at \ \varsigma = 0 . \tag{2-8}$$

2.3. The Lévêque Approach

In general, equation 2-1 must be solved numerically. If, however, the diffusion boundary layers are thin, an analytic solution for the flux can be obtained by assuming that the velocity profile is linear throughout the boundary layer. This approximation, known as the Lévêque approximation,[28,30,31,65] is not valid throughout a thin-gap channel, but it is useful for treating the entrance region, where the diffusion boundary layers are thin. Norris and Streid have solved this Lévêque problem for channel flow.[68]

One can extend the range of applicability of the Lévêque solution by writing a Lévêque series for the Nusselt number based on the equivalent diameter. This has been

done for the Graetz problem in a tube,[30,65] and a similar procedure can be used for a channel to give

$$Nu = -4 \frac{\partial \theta}{\partial \xi} |_{\xi=-1} = 2.7131949\varsigma^{-1/3} - 0.4 - 0.12146690\varsigma^{1/3} \tag{2-9}$$

(see appendix A).

2.4. The Graetz Approach

To treat the downstream region, the Graetz approach (separation of variables) should be used. To calculate the dimensionless concentration,

$$\theta = \frac{1}{2} - \frac{\xi}{2} + \sum_{k=1}^{\infty} A_k \, e^{-\lambda_k^2 \varsigma} \, Y_k(\xi) \, , \tag{2-10}$$

the coefficients A_k, the eigenvalues λ_k, and the eigenfunctions Y_k are needed. The coefficients are obtained by using the orthogonality of the eigenfunctions with respect to the weighting function $(1-\xi^2)$. The eigenvalues and eigenfunctions must be obtained numerically.

2.5. Solution of the Eigenvalue Problem

One can rewrite the eigenvalue problem in a form convenient for numerical solution by realizing that the eigenvalues λ are constant. Thus,

$$Y'' + \lambda^2(1-\xi^2) Y = 0 \tag{2-11}$$

$$\frac{d\lambda^2}{d\xi} = 0 \, . \tag{2-12}$$

To solve equations 2-11 and 2-12, three boundary conditions are needed. The first two boundary conditions result from equations 2-6 and 2-7. The third boundary condition is a normalization condition

$$Y' = 1 \text{ at } \xi = 1 \, . \tag{2-13}$$

The system of two ordinary differential equations with the three boundary conditions can be solved numerically using a finite-difference technique.[30,57] The computed

eigenfunctions are shown in figure 2-1.

2.6. Combining the Graetz and Lévêque Solutions

It should be noted that a truncated Graetz series (obtained from equation 2-10) is accurate for large ς, while the Lévêque series (equation 2-9) is valid for small ς. The value of ς that divides the two regions is that value at which the ratio of the two asymptotic solutions is closest to unity. For example, if three terms are used in each series, the maximum error in the Nusselt number is 0.48% at ς=0.11.

2.7. Asymptotic Forms for Large Eigenvalues

If greater accuracy than 0.48% is desired, then it is most efficient to add terms to the Graetz series. Therefore, it is useful to have simple asymptotic forms for calculating the higher eigenvalues and corresponding coefficients.

For the Graetz problem in a tube, Newman[30] extended the asymptotic forms of Sellars *et al.*[67] to achieve accuracy over a greater range of eigenvalues. Using a similar procedure, we devised an asymptotic form for the asymmetric Graetz problem:

$$\lambda = \lambda_0 + \frac{0.03254}{\lambda_0^{4/3}} - \frac{0.11}{\lambda_0^{14/3}}, \tag{2-14}$$

where

$$\lambda_0 = \frac{6k-1}{3} \qquad k=1, 2, \cdots \tag{2-15}$$

is the asymptotic form obtained by modifying the method of Sellars *et al.* (see reference 71). The function in equation 2-14 can be used for λ_6, λ_7,..., with a maximum error of 10^{-5}%.

The method of Sellars *et al.* predicts that the coefficients behave as $A_k = (-1)^{k+1}K/\lambda_k^{1/3}$ as λ_k becomes large, where

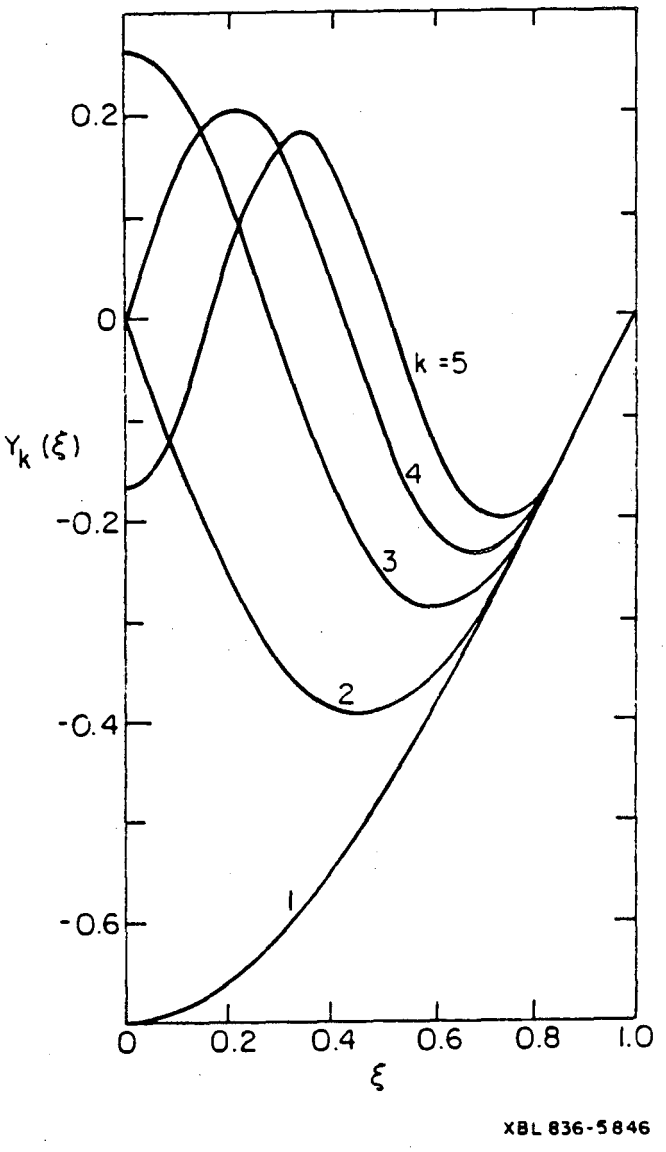


Figure 2-1. Eigenfunctions for the asymmetric Graetz problem

$$K = \frac{2^{4/3}\,\Gamma(\frac{2}{3})}{3^{1/6}\,\Gamma(\frac{4}{3})\,\pi} = 1.012787288\ . \tag{2-16}$$

This asymptotic form was modified to

$$A_k = (-1)^{k+1}\left(\frac{K}{\lambda_k^{1/3}}\right)(1 + 0.03\lambda_k^{-4/3} - 0.03\lambda_k^{-8/3})\ . \tag{2-17}$$

Equation 2-17 gives a maximum error of $4\times10^{-4}\%$ for $k\geq6$.

Table 2-1 shows the comparison between the eigenvalues and coefficients as calculated by solving the eigenvalue problem and those calculated from the asymptotic form in equations 2-14 and 2-17. The accurate results of Brown[66] are also shown.

Table 2-1

Eigenvalues and Coefficients for the Asymmetric Graetz Problem

	Eigenvalues, λ_k			Coefficients, A_k		
k	Brown	Calculated	Asymptotic	Brown	Calculated	Asymptotic
1	1.6815953222	1.6815953	1.6729924	0.8580866740	0.85808668	0.85807016
2		3.6722904	3.6721659		-0.65921993	-0.65931889
3	5.6698573459	5.6698573	5.6698540	0.5694628499	0.56946285	0.56949143
4		7.6688088	7.6688110		-0.51452221	-0.51453243
5	9.6682424625	9.6682424	9.6682441	0.4760654555	0.47606547	0.47606966
6		11.667894	11.667895		-0.44701873	-0.44702059
7	13.6676614426	13.667661	13.667662	0.4239737298	0.42397373	0.42397459
8		15.667496	15.667496		-0.40504973	-0.40505013
9	17.6673735653	17.667374	17.667374	0.3891087061	0.38910871	0.38910889
10		19.667279	19.667280		-0.37541429	-0.37541436

By using these asymptotic forms in a Graetz series with many terms, one can achieve a high degree of accuracy in the Nusselt number.

2.8. Empirical Approach

Recall that the Lévêque solution is applicable only on the wall with the step change in concentration. To fit the region of small ς on the opposite wall, it is more convenient to use an empirical function than it is to use many terms in the Graetz series.

The empirical function used here was derived by considering a simpler problem. If the fluid were in plug flow, rather than laminar flow, the mass-transfer problem would

be analogous to the problem of transient heat conduction in a finite slab, where short time, t, is analogous to small ς.

Based on the short-time solution for heat conduction in a finite slab, it is assumed that Nu is proportional to $e^{-b/\varsigma}$. To match the behavior at low $1/\varsigma$, a correction term of the form $ce^{-d/\varsigma}$ can be added. The constants c and d were obtained by performing a least squares fit between $\ln Nu$ as calculated from a 100-term Graetz series and $\ln Nu = a - b/\varsigma - ce^{-d/\varsigma}$, for various specified values of d. The least squares fit was designed to weight the region of small ς, where the empirical function is to be used. The resulting fitting function for small ς is

$$Nu = -4 \frac{\partial \theta}{\partial \xi} \Big|_{\xi=1} = 2 \exp(0.9594 - 0.6069 \frac{1}{\varsigma} - 0.4512 e^{-0.276/\varsigma}) . \qquad (2\text{-}18)$$

At $\varsigma = 0.18$, the error between the 3-term Graetz series and the empirical function is 0.013%.

2.9. Summary of Results

In summary, a convenient representation of the Nusselt number for the asymmetric Graetz problem has been obtained. For the wall with the step change in concentration, equation 2-9 is used for $\varsigma < 0.11$ and

$$Nu = 2 + 4 \sum_{k=1}^{3} |A_k| \; e^{-\lambda_k^2 \varsigma} \qquad \text{for } \varsigma \geq 0.11, \qquad (2\text{-}19)$$

where the eigenvalues and coefficients are listed in table 2-1. For the wall without the step change, equation 2-18 is used for $\varsigma < 0.18$ and

$$Nu = 2 - 4 \sum_{k=1}^{3} A_k e^{-\lambda_k^2 \varsigma} \qquad \text{for } \varsigma \geq 0.18. \qquad (2\text{-}20)$$

It has been proposed that this solution can be used in a superposition integral to determine the wall flux in problems where the channel wall concentrations are arbitrary and may differ on the two walls.

Nomenclature for Chapter 2

A	coefficients for the asymmetric Graetz problem
B	channel halfwidth, cm
c_b	bulk (feed) concentration of species i, mol/cm^3
c_i	concentration of species i, mol/cm^3
$c_{i,-1}$	concentration of species i at the wall located at $\xi = -1$, mol/cm^3
c_o	surface concentration of species i, mol/cm^3
D_i	diffusion coefficient of species i, cm^2/s
K	constant defined in equation 2-16
$N_{i,-1}$	flux of species i at the wall located at $\xi = -1$ (flux is positive in the $+\xi$ direction), mol/cm^2-s
Nu	Nusselt number (see equation 2-9)
$<v_x>$	average fluid velocity in the x-direction, cm/s
x	axial coordinate, cm
y	normal coordinate, cm

Y	eigenfunctions for the asymmetric Graetz problem

Greek

Γ	gamma function
ς	dimensionless axial coordinate defined in equation 2-4
θ	dimensionless concentration defined in equation 2-2
λ	eigenvalues for asymmetric Graetz problem
λ_0	asymptotic form for large eigenvalues
ξ	dimensionless normal coordinate defined in equation 2-3

Subscripts

b	bulk
i	species in solution
k	index for eigenvalues and eigenfunctions
o	surface
x	axial direction
-1	wall located at $\xi = -1$
0	asymptotic form for large eigenvalues
1	wall located at $\xi = 1$

Superscripts

$*$	dummy variable of integration
$'$	differentiation with respect to ξ

3. Thin-Gap-Model Results

3. Introduction

Mathematical models are useful for predicting the performance of complex systems under a variety of conditions, without having to do repeated experiments. While simplified models and dimensional analysis are useful for quick calculations, they cannot predict accurately the interaction of several phenomena.

The complex model discussed in chapters 1 and 2 predicts the total current and the distribution of current, concentration, and potential adjacent to the electrode surfaces in a channel flow cell. In addition, the conversion of a given reactant may be obtained, by a material balance, from the current distributions. One application for the model is to assess some of the economic tradeoffs associated with thin gaps. For example, thin gaps reduce the cell resistance, but increase the "chemical shorting" in the cell, where a product from one electrode diffuses across the cell and reacts back to the starting material. Another use for the model is to provide information that is difficult to obtain experimentally, such as the distributions of current, concentration, and potential. Current distribution can be measured experimentally by a sectioned-electrode technique or by measuring plated deposits, but it is difficult to measure potentials and concentrations in the solution adjacent to the electrodes.

We introduce this chapter with a discussion of the three limiting cases of current distribution and the results of other models and conclude with a discussion of the results from the thin gap model both for a plating and dissolution reaction (section 3.2) and for a redox reaction (section 3.3). Chapter 5 presents a comparison between theoretical and experimental results for the $Fe(CN)_6^{3-}/Fe(CN)_6^{4-}$ redox couple.

3.1. Limiting Cases of Current Distribution

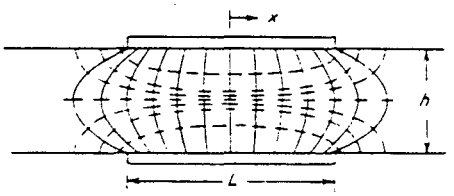
Before examining previous laminar-flow models, it is useful to understand the three limiting cases of current distribution in any electrochemical system.[28,47] These current distributions are called primary, secondary, and mass-transfer-limited current distributions. The primary distribution results when there are no kinetic or mass-transfer limitations. Only the ohmic potential drop in the solution governs the primary current distribution. If the current distribution is governed by ohmic potential drop and reaction kinetics, but not mass transfer, the distribution is called the secondary current distribution. If mass transfer alone dominates the current distribution, the electrochemical cell is at the "limiting current." That is, the current is limited by how fast the reactants can diffuse to the electrode. In general, all three effects — ohmic potential drop, electrode kinetics, and mass transfer — must be considered simultaneously.

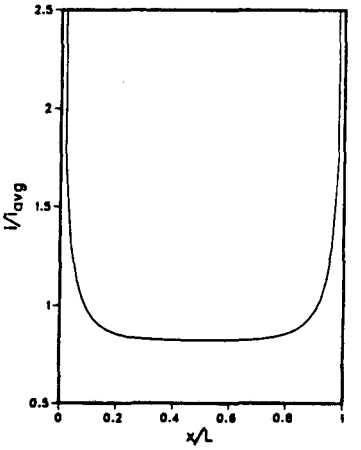
The shapes of two of the limiting cases of current distribution in a channel flow cell are shown in figure 3-1. The primary current density is infinite at the ends of the electrodes, the secondary current density is similar to the primary, but finite everywhere, and the mass-transfer limited current density is high only at the leading edge of the electrode. The primary distribution is calculated by solving Laplace's equation for the potential with a constant potential adjacent to each electrode,[28,47] and it is infinite at the edges because the current can flow out into the solution and collect back at the edges. (See figure 3-1.) The secondary distribution is similar to the primary, but because of kinetic limitations, the current cannot be infinite at the electrode edges. The shape of the secondary distribution generally depends on the geometry of the system, the kinetic parameters, and the average current density. Wagner[73] has solved the secondary current distribution problem in a channel for large and small aspect ratios. Section 117 of Newman's book[28] discusses the secondary current distribution for a disk elec-

PRIMARY CURRENT DISTRIBUTION

Fast kinetics, no mass-transfer limitations. (Ohmically dominated)

Constant potential near electrodes.

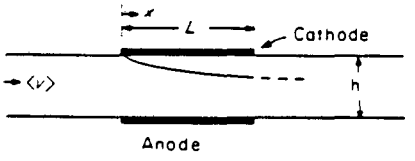


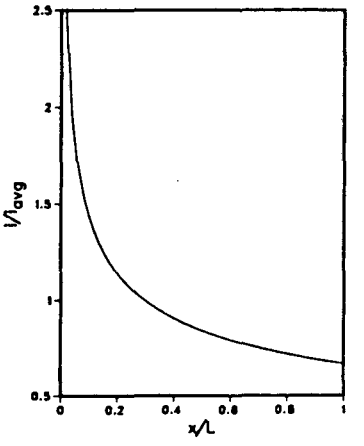


LIMITING CURRENT DISTRIBUTION

Mass-transfer limited.

Surface concentration is zero.





XBL 866-2510

Figure 3-1. Limiting cases of current distribution.

trode, and the same qualitative conclusions apply to channel electrodes. If mass transfer alone dominates the current distribution, the current density is high only at the leading edge of the electrode, where there is undepleted solution.

3.1.1. Results for Thin Diffusion Boundary Layers

Parrish and Newman[20,21] present current and concentration profiles for a single metal dissolution and deposition reaction. Their model includes ohmic potential drop, reaction kinetics, and mass transfer; however, in their earlier paper,[20] they do not consider any interaction between the two electrodes. In the second paper,[21] they account for interaction through the potential distribution, but not through the mass transfer. (They assume thin boundary layers.) Their results are particularly interesting below the limiting current, because the cathodic current distribution can have a maximum. Intuitively, one would expect that mass transfer is not important on the anode because the anodic dissolution reaction is not affected by the concentration of ions in the solution; therefore the current distribution should resemble the secondary current distribution, which is high at both edges. On the cathode, however, mass transfer begins to affect the current distribution in the downstream region, where the solution is depleted of metal ion; in the upstream region, the cathodic current distribution should behave like a secondary current distribution. If the downstream region is small enough, the cathodic current density can have a maximum arising from a compromise between the secondary current distribution and the mass-transfer-limited current distribution. The thin gap model reproduces these results for large Graetz numbers ($\geq$100), as it should.

3.1.2. Results for Straight Current Lines

White *et al.* present two multiple-reaction models, one that assumes straight current lines[18] and another that includes axial migration and diffusion and time depen-

dence.[19] Both papers discuss a copper system in which Cu(I) reacts at both electrodes and Cu(II) is a product on the anode and a reactant on the cathode. In the first paper, they examine the copper system with excess supporting electrolyte and discuss the current efficiency, selectivity for copper deposition, and conversion per pass. They note that the important independent variables are $Pe\ h/L$ (which is proportional to the Graetz number defined in chapter 1) and the cell potential.

One can infer from the paper that the average current efficiency for copper deposition can be improved by designing an electrode whose potential varies with axial position. In the example they discuss, the reactions are

$$\begin{aligned} &\mathrm{Cu(I)} \rightarrow \mathrm{Cu(II)} && \text{anode, reaction 1} \\ &\mathrm{Cu(II)} \rightarrow \mathrm{Cu(I)} && \text{cathode, reaction 2} \\ &\mathrm{Cu(I)} \rightarrow \mathrm{Cu(0)} && \text{cathode, reaction 3,} \end{aligned} \qquad (3\text{-}1)$$

and the concentration of Cu(I) is 0.5 M and the concentration of Cu(II) is 0.1 M. In this case, reactions 1 and 3 are claimed to be near limiting current, and reaction 2 is well below limiting current. The idea of designing the electrode with an axially-varying potential is to increase the driving force near the trailing edge so that reaction 2 can proceed faster there and provide Cu(I) to aid reaction 3.

White *et al.* also discuss the effects of $Pe\ h/L$ and cell potential on the selectivity of copper, which they define as the moles of Cu produced / moles of Cu(II) produced, and the conversion per pass of Cu(I). They note the existence of a case for which there is a mild maximum in the Cu selectivity *vs.* $Pe\ h/L$ and in the conversion per pass *vs.* $Pe\ h/L$.

The second paper discusses the effects of axial diffusion and migration and time dependence. The important results are that axial diffusion and migration may be neglected for $h/L < 0.5$ and that the effect of axial diffusion and migration is to smooth

the current distribution.

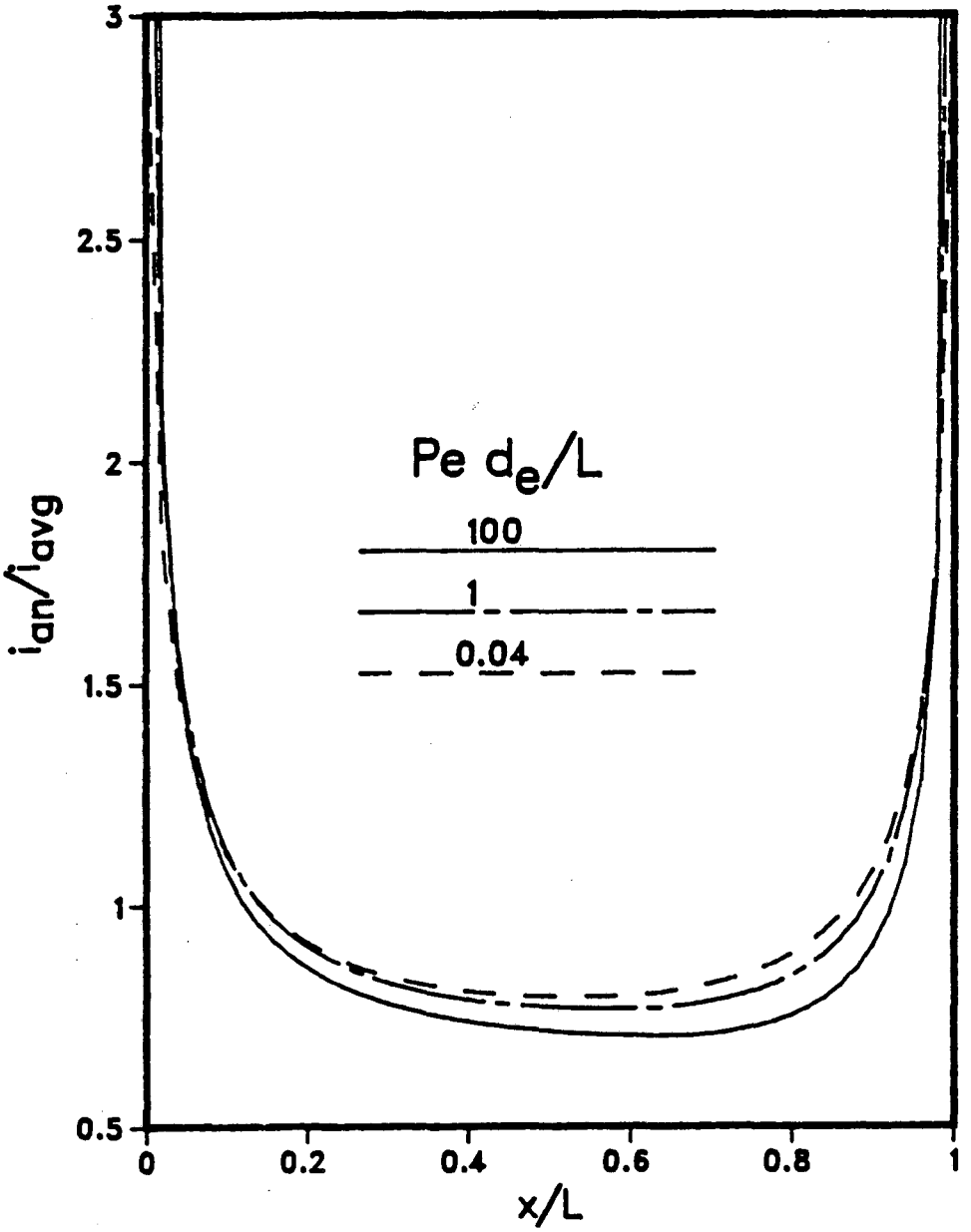
3.2. Thin-Gap-Model Results for a Plating Reaction

In this section, we examine the effects of interacting boundary layers and of the axial component of current in a system with excess supporting electrolyte (no migration). Since the effects of interacting boundary layers are most pronounced when mass transfer is controlling, only limiting-current results will be discussed. It is interesting that, for thick diffusion boundary layers, there is still a limiting current, even though products from the anode can diffuse to the cathode and react.

For simplicity, we shall consider a single electrode reaction and a single species, for example, a metal dissolution reaction on the anode and the metal plating reaction on the cathode. We shall also simplify the interpretation of the results by considering a case with weak interaction of the electrodes through Laplace's equation (h/L=1). If the electrodes interact through Laplace's equation, the anode current distribution begins to resemble the cathode current distribution, as shown by Parrish and Newman.[21] Thus, the source of interaction discussed here will be the diffusion of reacting species across the cell gap. After discussing the effect of interacting boundary layers, we shall discuss the effects of the axial component of current. Note that our choice of a large h/L enhances the axial current, as shown by Nguyen *et al.*[19] Figure 3-2 shows a dimensionless plot of the anode current distribution for three diffusion-boundary-layer thicknesses. The parameter $Pe\ d_e/L$ (which is proportional to the Graetz number†) is a measure of the dimensionless gap thickness relative to the boundary-layer thickness. For large $Pe\ d_e/L$ ($\geq$100), for example high flow rates, the boundary layer is thin compared to the interelectrode gap. The current distributions shown in figure 3-2 resemble secondary

† $Pe\ \dfrac{d_e}{L} = \dfrac{4}{\pi} Gz.$

Anode Current Distribution



XBL 856-2902

Figure 3-2. Effect of boundary-layer thickness on anode current distribution.

current distributions because there are no mass-transfer limitations on the anodic dissolution reaction and because the cathode does not have much influence through the potential distribution. (h/L=1). Therefore the anodic current distribution is not strongly affected by the boundary-layer thickness.

The cathode, on the other hand, is mass transfer limited, and therefore i_{cath} is strongly dependent on $Pe\ d_e/L$. Figure 3-3 shows an example of the cathode current deviation from the thin-boundary-layer prediction. (If $Pe\ d_e/L$ were 100 or higher, there would be no deviation, and the thin-boundary-layer approximation could be used.) In figure 3-3, the solid curve is the current density resulting from the hypothetical Graetz- Lévêque problem, which has a step change in cathode surface concentration (from the feed concentration to zero) and a constant anode surface concentration (c_{an}=c_{feed}). This curve decays as $x^{-1/3}$ near the leading edge, where the boundary layer is thin and levels off to a constant in the downstream region, where the concentration profile across the cell gap is linear (fully-developed mass transfer). The cathode current distribution predicted by the model is shown by the dotted line and decays as $x^{-1/3}$ near the leading edge, where the boundary layers are thin, but increases again when the anode boundary layer touches the cathode. That is, the solution near the cathode is no longer depleted because metal ions produced at the anode diffuse across the gap to the cathode. Away from the edges of the cell, the cathode current density is low, roughly following the behavior of the anode. Near the trailing edge, the anode current density increases from ohmic effects, and the cathode current increases because some of the ions produced on the anode have diffused across the cell gap.

We have seen how the interaction of the diffusion boundary layers affects the anodic and cathodic current distributions at the limiting current. Again, it is interesting that there is still a limiting current when the boundary layers interact. To understand why there is a limiting current and to understand the anode surface concentration profiles,

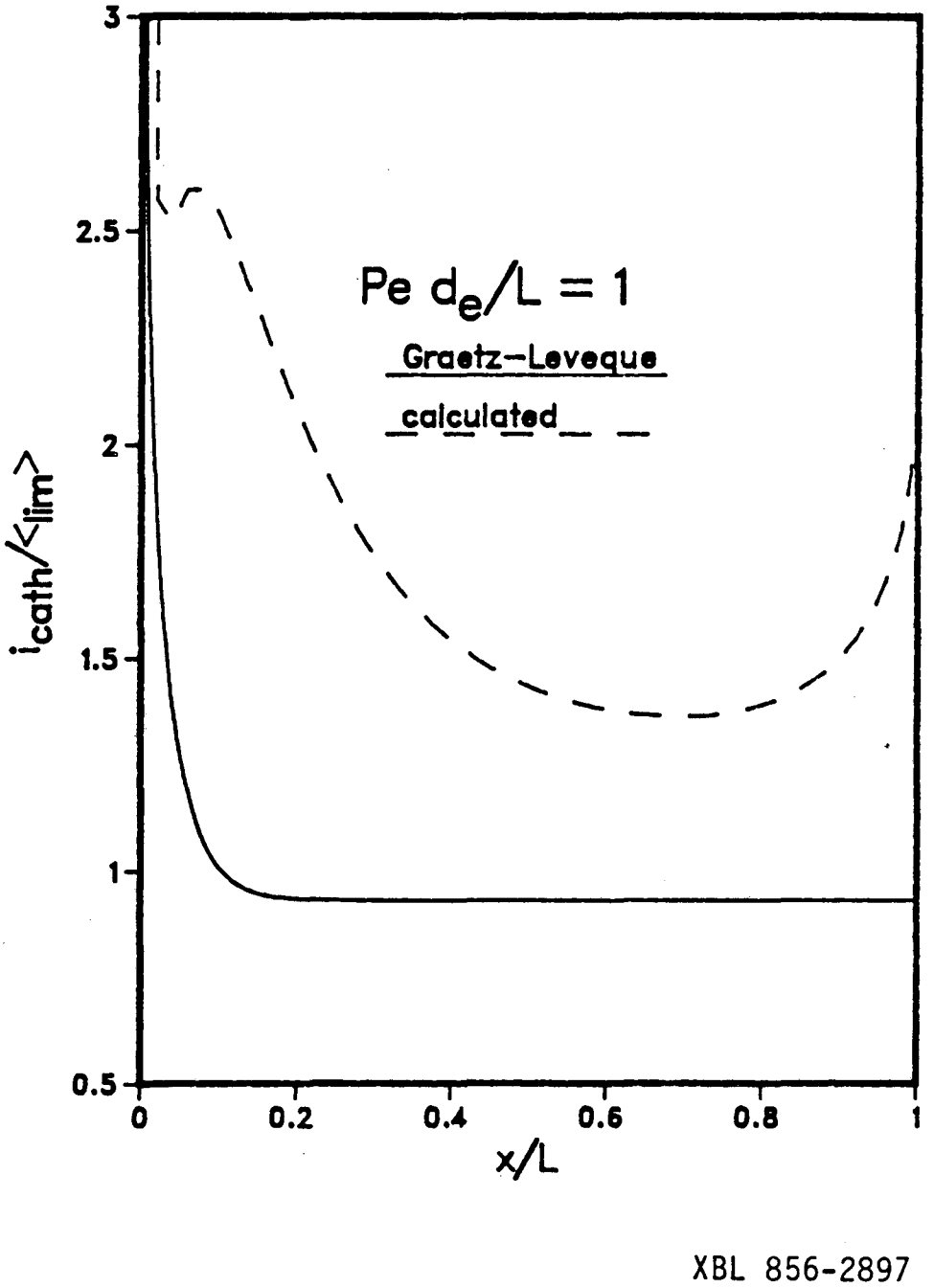


Figure 3-3. Deviation from thin-layer approximation (normalized by average current from step-function problem).

one must examine the axial component of current.

Using the anodic and cathodic current distributions, one can calculate, by a material balance, the average current density flowing in the axial (fluid-flow) direction across a vertical cross-section at position x:

$$I_{axial}(x) = \frac{1}{L}\int_0^L [i_{an}(x') + i_{cath}(x')]\,dx'. \tag{3-2}$$

Thus, negative axial currents indicate that there has been more cathodic reaction up to the axial position x. (Cathode currents are negative.) Since the cathodic reaction depletes the solution of metal ions, the cup-mixing average concentration at the position x is less than the feed concentration for negative axial currents. Figure 3-4 shows the axial current for $Pe\ d_e/L = 1$. Note that the axial current is zero at x=0 and x=L because there is no net flow of current beyond the electrode edges. The positive spike in I_{axial} near the leading edge results because the anode current density rises more rapidly toward the leading edge than the mass-transfer-limited cathode current density (approximately $x^{-1/2}$ *vs.* $x^{-1/3}$). Throughout most of the cell, the extra current produced at the downstream edge of the anode flows upstream before reaching the cathode. Note that this does not imply that metal ions diffuse upstream, because current flows by the migration of supporting-electrolyte ions.

The current and potential lines corresponding to figure 3-4 are shown in figure 3-5. The potential lines were calculated using the complete solution to Laplace's equation:

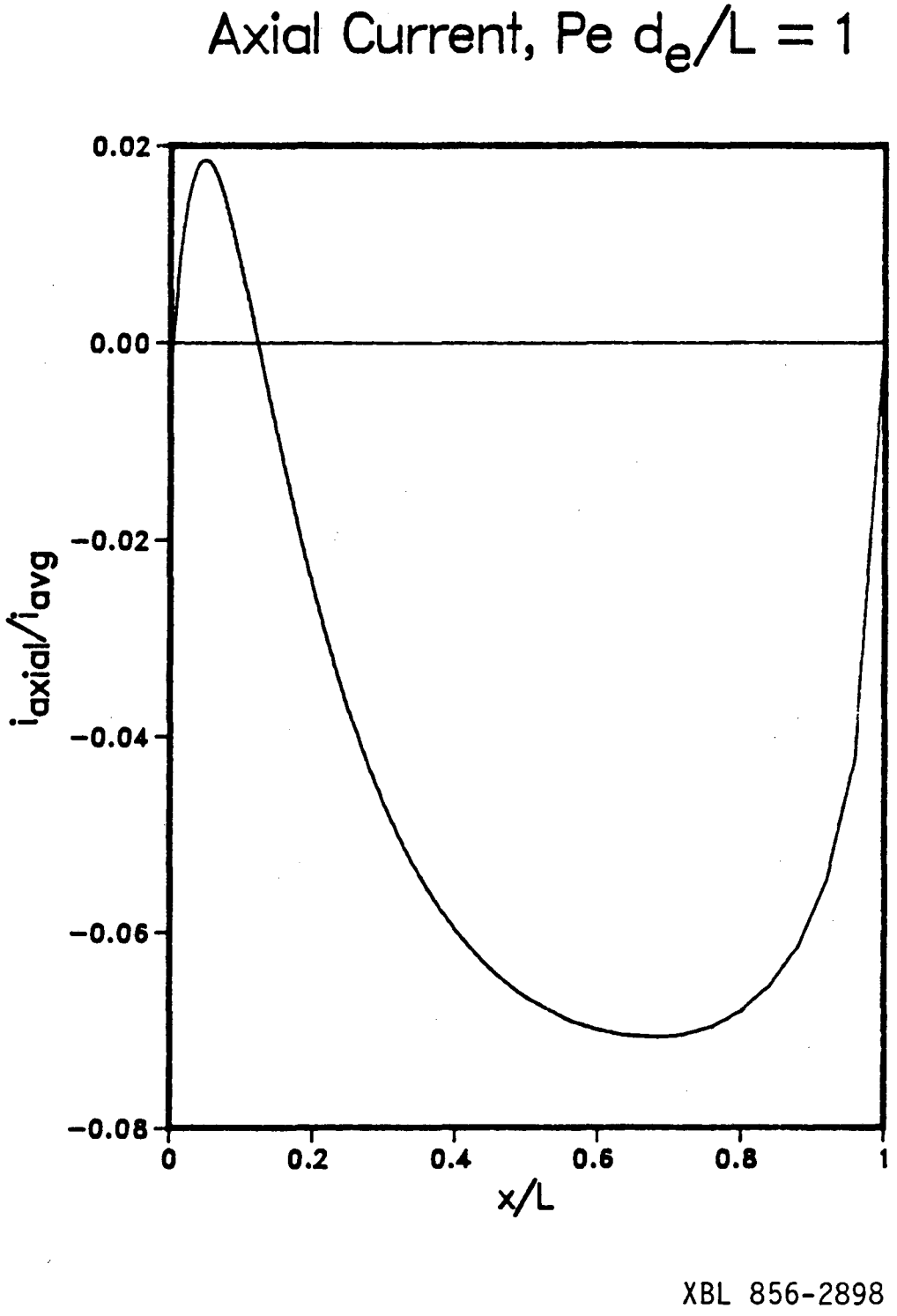


Figure 3-4. Axial current distribution for a thick boundary layer.

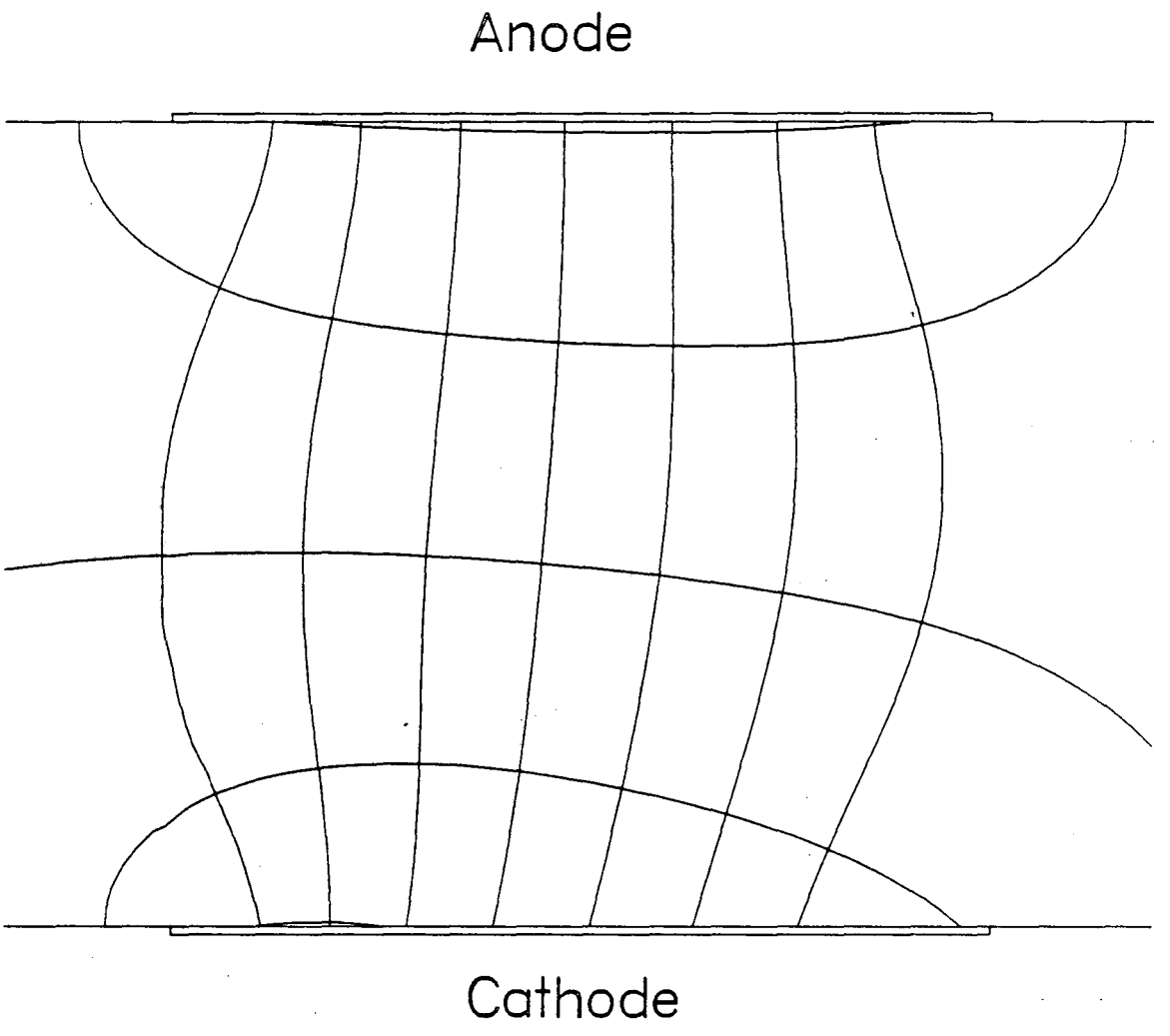


Figure 3-5. Current and potential lines for a thick boundary layer.

$$\Phi(x,y) = \Phi^* - \frac{1}{2\pi\kappa_\infty}\left\{\int_0^L i_{cath}(x')\ln\left[\sinh^2\left(\frac{\pi(x-x')}{2h}\right) + \sin^2\left(\frac{\pi(y-h)}{2h}\right)\right]\right.$$

$$\left. + i_{an}(x')\ln\left[\sinh^2\left(\frac{\pi(x-x')}{2h}\right) + \sin^2\left(\frac{\pi y}{2h}\right)\right]dx'\right\}. \qquad (3\text{-}3)$$

The current lines were plotted by starting at one electrode and stepping along a two-dimensional grid, numerically calculating the gradient of Φ at each x or y mesh point. In principle, it should be possible to obtain the same current lines starting on either electrode. However, since the results were slightly different, an average was taken.

The figures show that the axial current is negative throughout most of the cell. Therefore, one would expect the electrolyte to be depleted of metal ion throughout most of the cell. One can examine the depletion of the electrolyte by following the anode surface concentration as a function of axial position. Since the surface concentration of metal ion is zero at the cathode (limiting current), the cup-mixing concentration is roughly equal to the feed concentration when the anode surface concentration is twice the feed concentration. Figure 3-6 shows the anode surface concentration as a function of axial position. Note that the spike in c_{an}/c_{feed} above 2 corresponds to the positive spike in axial current. Farther downstream, however, c_{an}/c_{feed} is less than 2, and the axial current is negative; therefore the solution is depleted.

The effect of boundary-layer thickness is also shown in figure 3-6. As the diffusion boundary layer becomes thicker (*e.g.* lower flow rate), the electrolyte becomes more depleted. In fact, the anode surface concentration can fall below the feed concentration, as shown in figure 3-6. Although the results for $Pe\ h/L = 0.04$ should be viewed with some caution, because the Péclet number is low enough that axial diffusion will smooth the current and concentration profiles, the low surface concentrations are still expected.

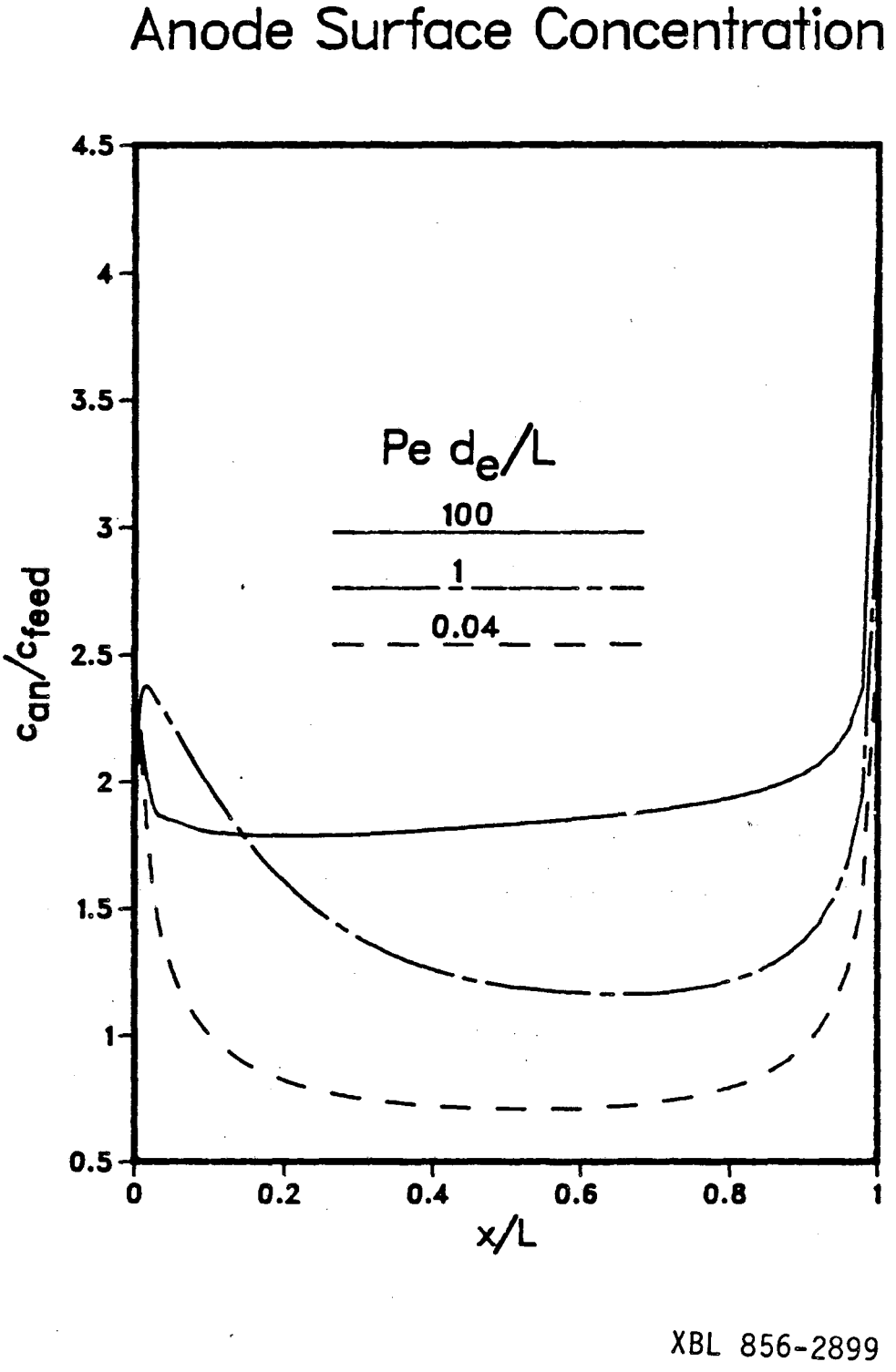


Figure 3-6. Effect of boundary-layer thickness on anode surface concentration.

We can now discuss the presence of a limiting current for interacting boundary layers. At first, one might think that, by increasing the cell potential, the anode would produce more ions that would diffuse over to the cathode and react, and that, therefore, there would not be a limiting current. However, this is not the case because if c_{an} were much greater than $2c_{feed}$ everywhere, the axial current would be positive everywhere; therefore, the production of metal ions on the anode is limited by the requirement of no net current flow beyond $x=L$. Another way of thinking about this is that the ions produced on the anode must travel some distance downstream before reaching the cathode. The ratio of the electrode length L to this distance is a rough indication of the amplification effect of ions reaching the cathode from the anode instead of just from the feed stream. This amplification factor can be large, but it is still finite, and there is still a limiting value for the current.

The effects of interacting diffusion boundary layers and of the axial component of current have been discussed for a single plating reaction. When $Pe\ d_e/L < 100$, the boundary layers interact and may not be considered thin. That is, if the gap is thin, the flow rate is low, the electrodes are long, or the diffusion coefficient is high, the electrodes must be treated simultaneously in the mathematical analysis.

An interesting result of the model is that there is a limiting current for thin gaps. Even though products from the anode can react on the cathode, the production of species on the anode is limited by the requirement of equal currents on the two electrodes.

The effect of axial currents is to deplete the electrolyte of metal ions throughout most of the cell. Thus, the two-dimensional structure of the potential distribution is important, and it should not be assumed that the current lines travel straight across

from one electrode to the other.[†]

3.3. Results for a Redox Reaction

We now examine results for a redox reaction in which there are two reacting species, such as ferri- and ferrocyanide ions. The cathodic reactant is denoted species "1" so that, on the cathode,

$$M_1 + e^- \rightarrow M_2. \tag{3-4}$$

The two additional parameters that arise in changing from a plating to a redox system are the ratios $(c_1/c_2)_{feed}$ and D_1/D_2. These parameters determine which of the two species is the limiting reactant.

For simplicity, we consider first cases in which $D_1 = D_2$; then if c_2 is greater than c_1, the limiting reactant is species 1. If $c_2 >> c_1$, then c_1 behaves like the single species in a plating/dissolution reaction; if $c_2 = c_1$, then both species behave the same, but on opposite electrodes. These two limiting cases serve as a check of the computer program for redox systems.

Figure 3-7 shows the cathode current distribution for the two limiting cases. For $c_2 >> c_1$, the limiting reactant, c_1, reacts at the cathode, so the cathode current density decays as $x^{-1/3}$ (for high Graetz numbers). This current density also prevails on the cathode if $c_1 = c_2$ because species 1 is still the limiting reactant. The anode current distribution, shown in figure 3-8, depends on the ratio $(c_2/c_1)_{feed}$ because the reactant, species 2, may or may not be limited. If it is limited, as in the case $c_{2,feed} = c_{1,feed}$, the anode current decays as $x^{-1/3}$; if it is not limited at all ($(c_2/c_1)_{feed} = \infty$), then the anode current resembles that of a single species. The intermediate values of $(c_2/c_1)_{feed}$ correspond to various fractions of limiting current on the anode. The maximum in the anode current distribution (for $(c_2/c_1)_{feed} = 1.2$ and 1.1 in figure 3-8) was also observed

[†] Nguyen *et al.*[19] suggest that straight current lines may be assumed for $h/L < 0.5$.

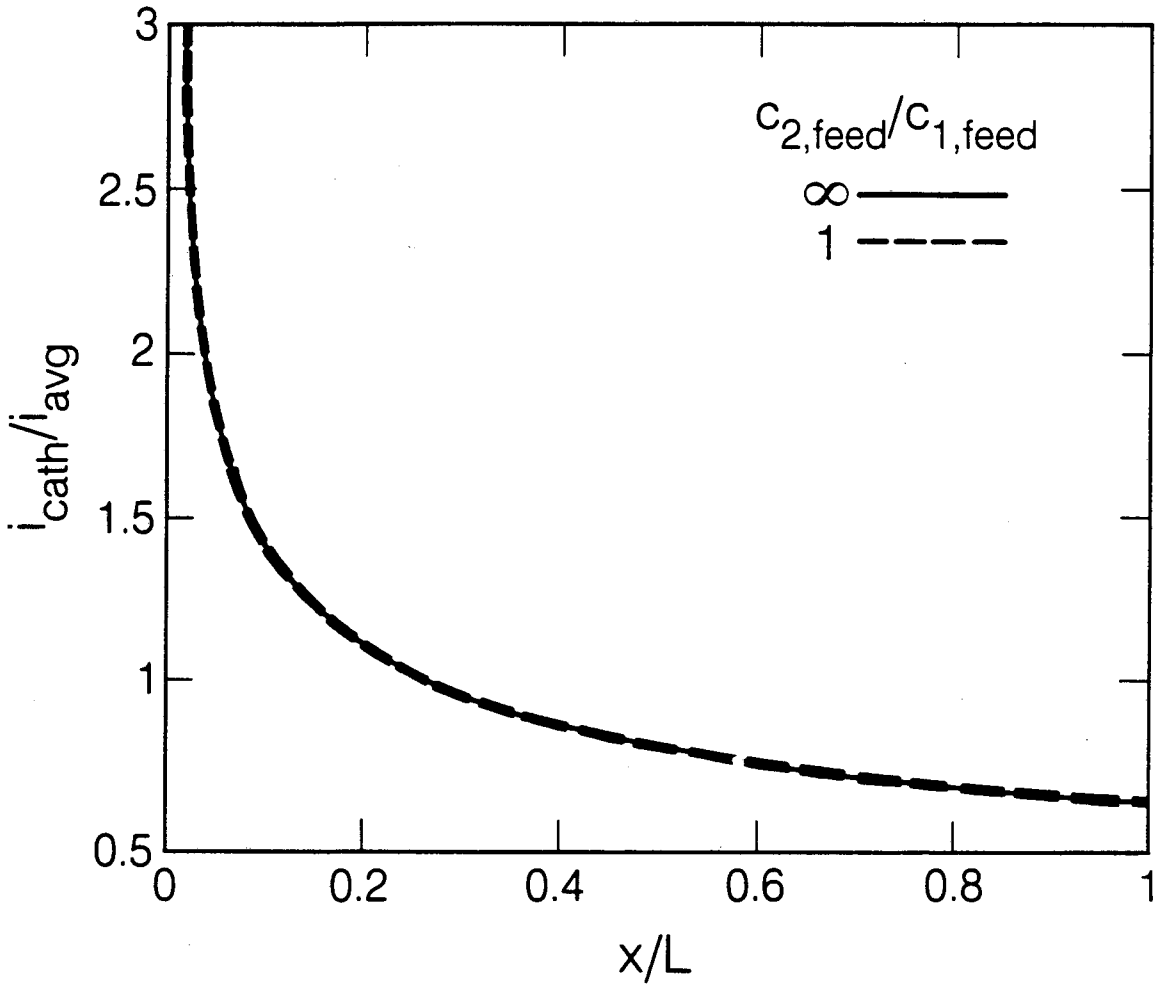
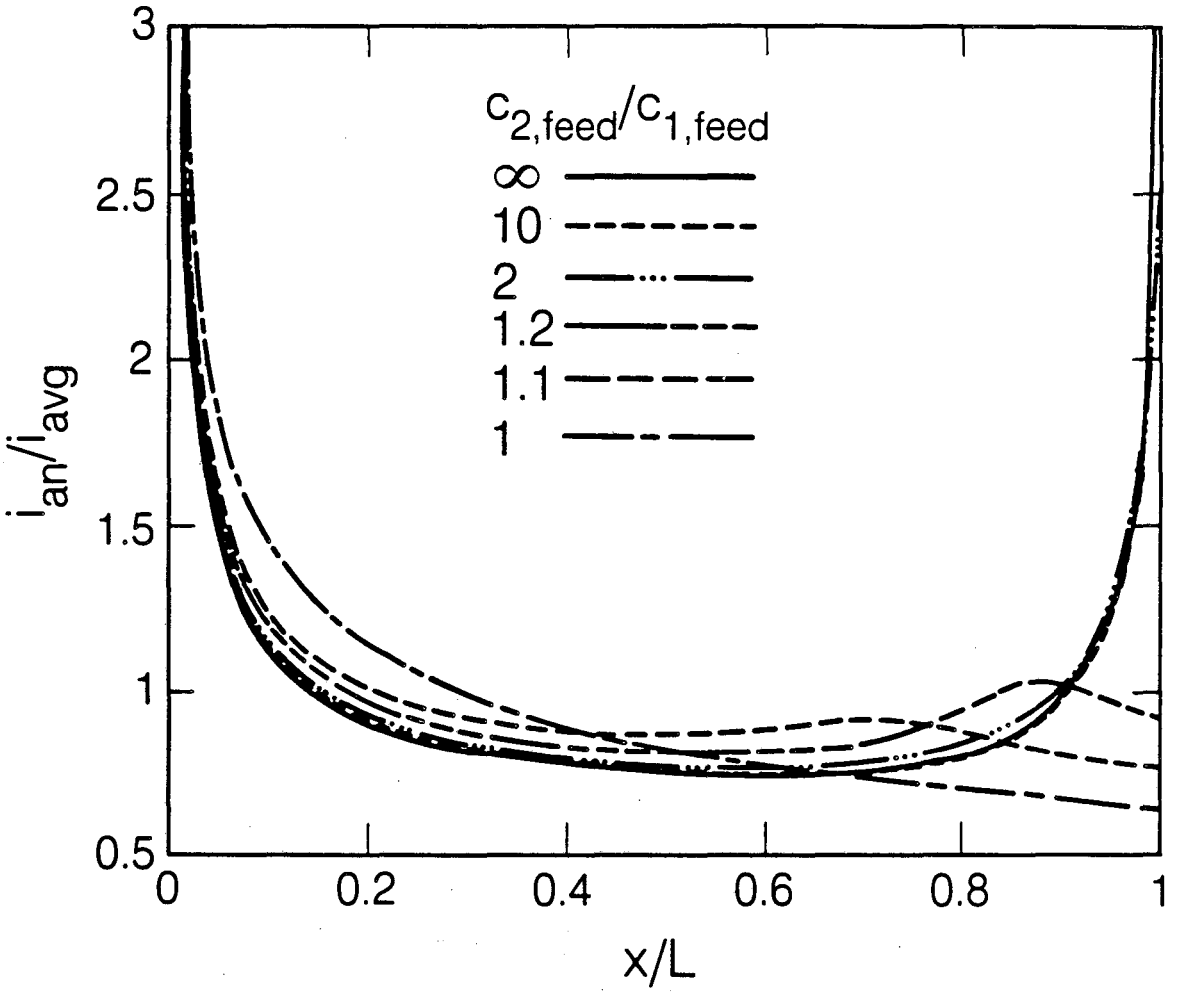


Figure 3-7. Cathode current distribution for a redox system with equal diffusion coefficients ($h/L = 1$).



XBL 865-8780

Figure 3-8. Anode current distribution for a redox system with equal diffusion coefficients ($h/L = 1$).

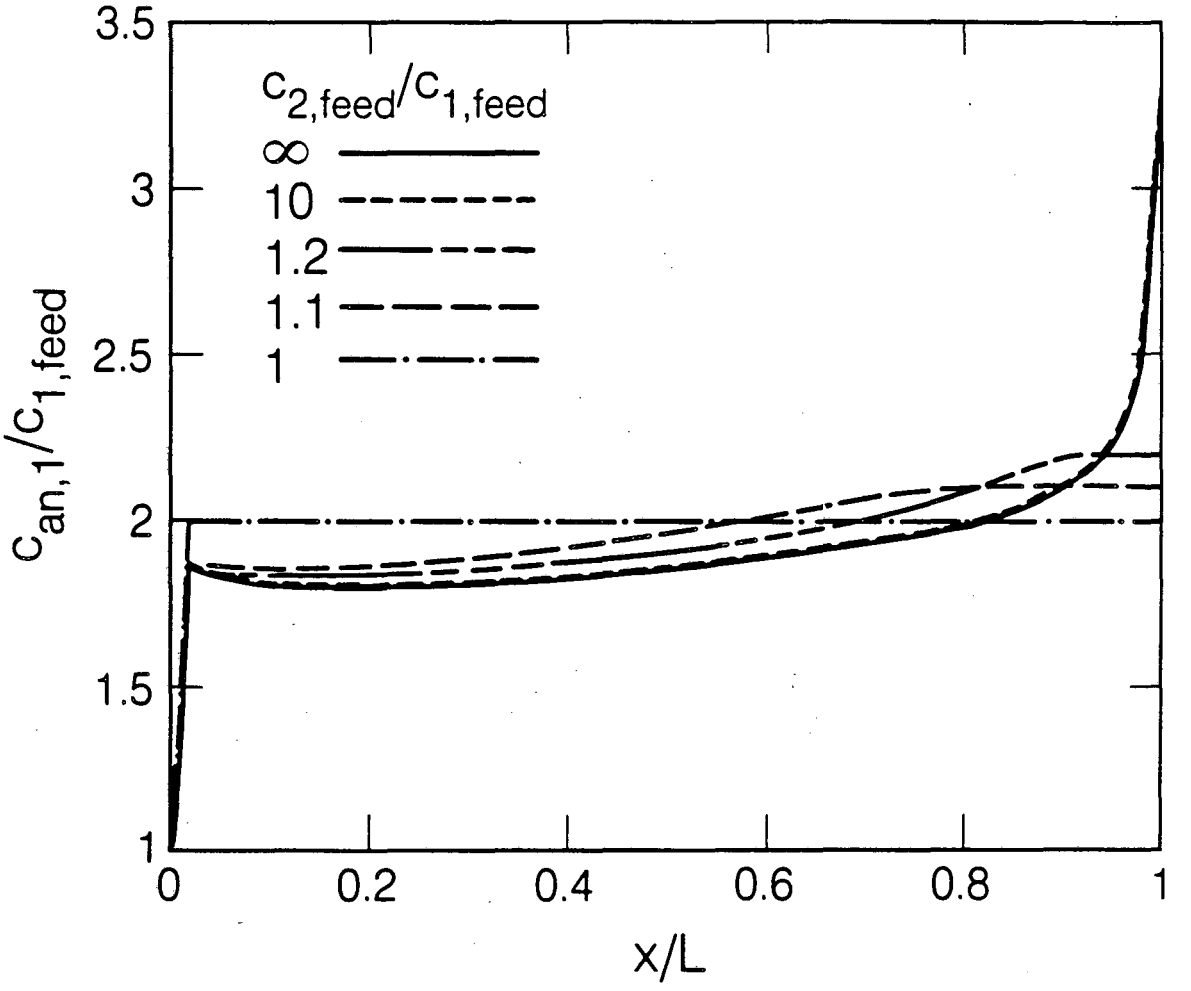
by Parrish for a single reactant;[21] it occurs because the current begins to rise near the trailing edge from ohmic effects, but decreases again because the reaction becomes mass-transfer limited (species 2 is depleted).

The surface concentration profiles corresponding to figures 3-7 and 3-8 are shown in figures 3-9 and 3-10. Figure 3-9 shows the concentration of species 1 on the anode. For $c_{2,feed} >> c_{1,feed}$, the concentration $c_{1,an}$ approaches c_{an} for a single-species dissolution reaction; for $c_{2,feed}=c_{1,feed}$, the anode concentration steps up to $2c_{feed}$, reflecting the limiting-current behavior on the cathode, where c_1 steps down to zero. The intermediate curves level off at the trailing edge because of mass-transfer limitations. Figure 3-10 shows $(c_{2,an} - c_{2,feed})/c_{1,feed}$ to illustrate that the two species depart from their feed values by the same amounts, but in the opposite direction, on a given electrode.

We consider next an example with unequal diffusion coefficients (see references 28, 74, and 75). At high Graetz numbers, results from the Lévêque approximation apply. One such result is that if the concentration of species 1 drops to zero, then the concentration of species 2 on that electrode steps up to

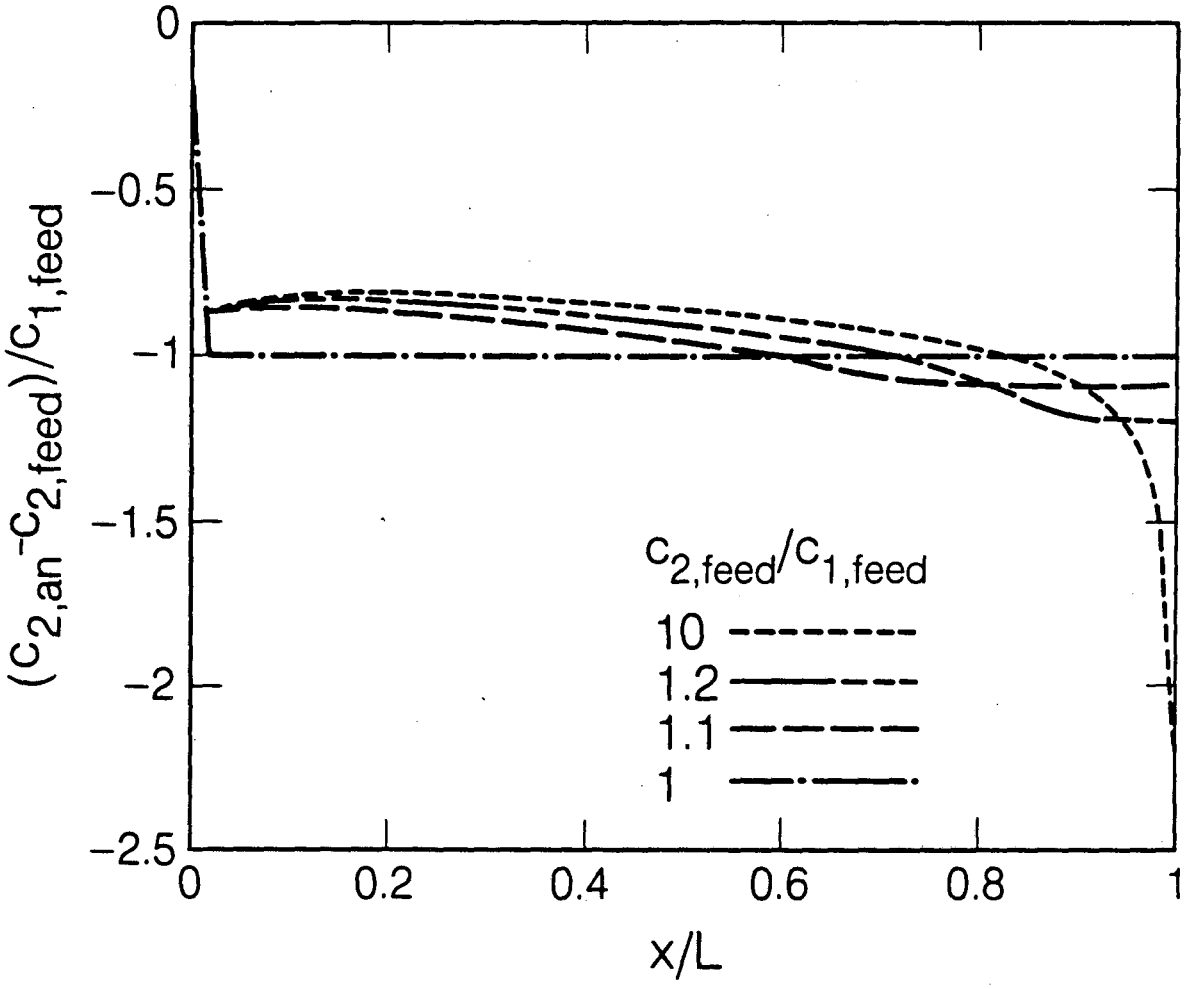
$$c_2 = c_{2,feed} + (D_1/D_2)^{2/3}\, c_{1,feed}. \tag{3-5}$$

(The same result with the subscripts reversed applies to the other electrode.) This high-Graetz-number result is obtained with the model, as shown in figures 3-11 and 3-12. The figures demonstrate the effects of the Graetz number and aspect ratio (h/L) for $(c_2/c_1)_{feed} = 1/4$ and $D_2/D_1 = 8$. At high Graetz numbers, the concentrations step to the values predicted by equation 3-5; at low Graetz numbers, however, the mass transfer becomes fully-developed, and the leading-edge surface concentrations cannot be maintained because of the requirement of equal current on the two electrodes. For fully-developed mass transfer, the surface concentrations are



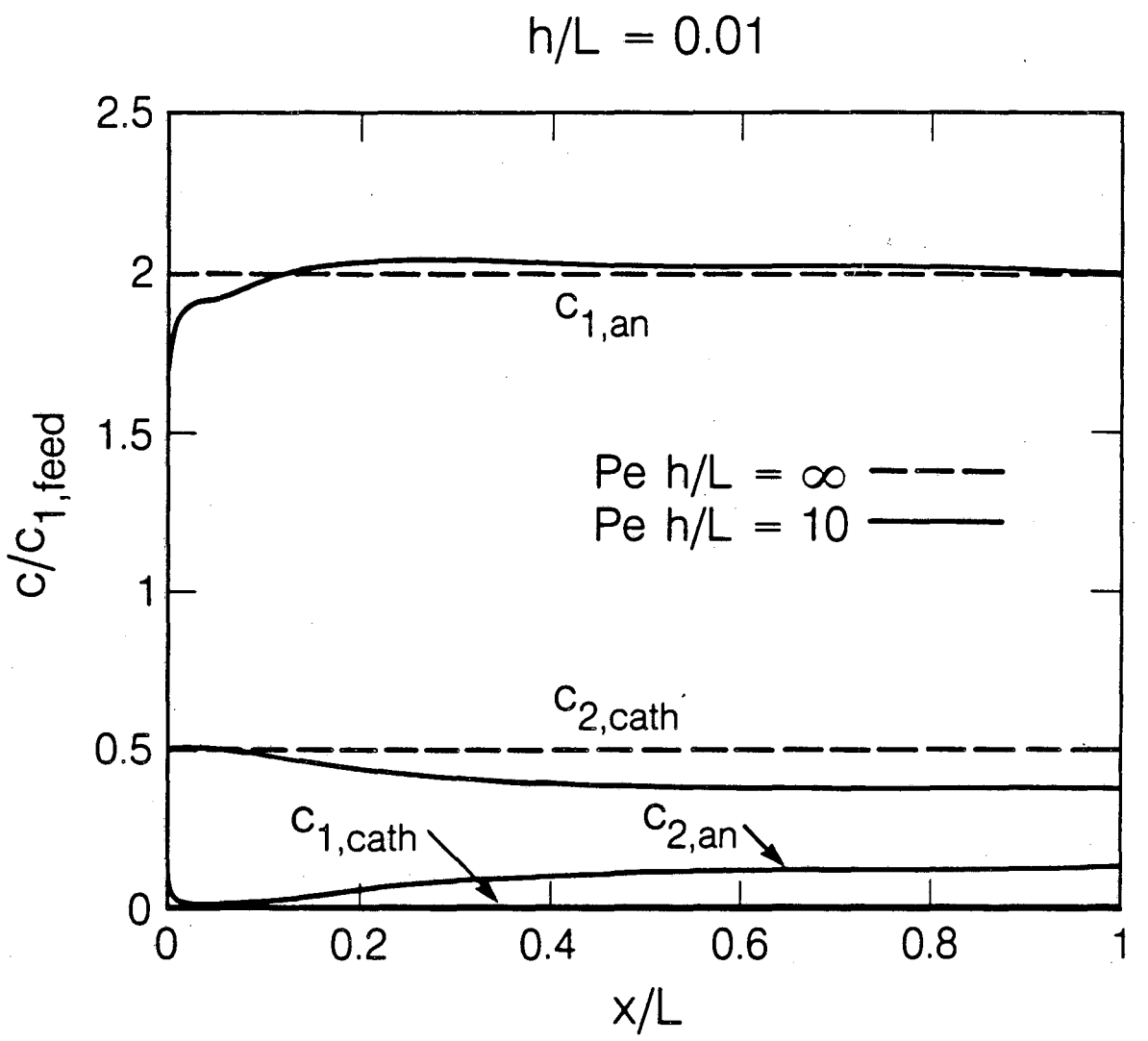
XBL 865-8779

Figure 3-9. Anode surface product concentration for a redox system with equal diffusion coefficients ($h/L = 1$).



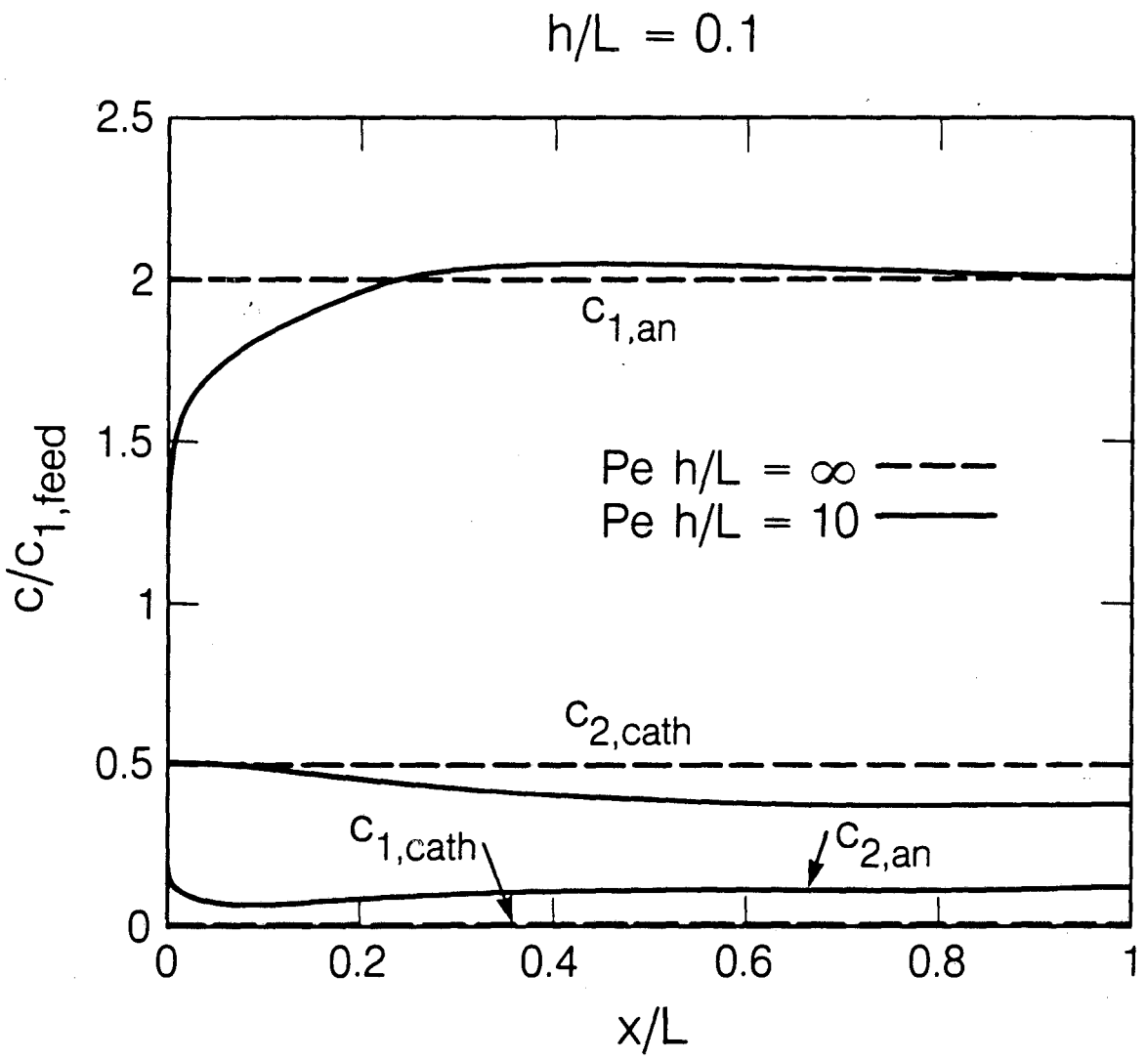
XBL 865-8775

Figure 3-10. Anode surface reactant concentration for a redox system with equal diffusion coefficients ($h/L = 1$).



XBL 865-8778

Figure 3-11. Surface concentration profiles for a redox system with unequal diffusion coefficients. ($(c_2/c_1)_{feed} = 0.25$, $D_2/D_1 = 8$, $h/L = 0.01$).



XBL 865-8776

Figure 3-12. Surface concentration profiles for a redox system with unequal diffusion coefficients. ($(c_2/c_1)_{feed} = 0.25$, $D_2/D_1 = 8$, $h/L = 0.1$).

$$c_{2,an} = \frac{1}{8} c_{1,feed}$$

$$c_{2,cath} = \frac{3}{8} c_{1,feed}$$

$$c_{1,an} = 2c_{1,feed}$$

$$c_{1,cath} = 0,$$

and the concentration profiles are linear across the cell gap. Note that these concentrations satisfy the condition of equal currents on the two electrodes. It is the ratio of diffusion coefficients that determines, for a given feed composition, which is the limiting species (species 1, in this example).

An interesting feature of figures 3-11 and 3-12 is that $c_{2,an}$ clearly shows the Lévêque (thin-boundary-layer) region and the fully-developed-mass-transfer region; in the Lévêque region, $c_{2,an} \approx 0$ and in the developed region, $c_{2,an} \approx 1/8\ c_{1,feed}$. Also shown in the figures is the effect of h/L. One would expect that for small h/L (<10) the electrodes would interact through the potential distribution[21] and that for very small h/L, both electrodes would behave like the limiting electrode. Parrish and Newman demonstrated this effect for a plating reaction.[21] Figures 3-11 and 3-12 also show this effect; for the small aspect ratio, $h/L = 0.01$, the anode is closer to the limiting current than for $h/L = 0.1$ ($c_{2,an}$ is closer to zero near the leading edge.)

In this section we have discussed the effects of $(c_1/c_2)_{feed}$, (D_1/D_2), Gz, and h/L for a redox system. $(c_1/c_2)_{feed}$ and (D_1/D_2) determine which species is limiting, the Graetz number (or $Pe\ d_e/L$) determines the size of the thin-boundary-layer region, and h/L determines the degree of interaction through the potential distribution.

Nomenclature for Chapter 3

c_{an}	anode surface concentration, mol/cm^3
c_1	concentration of species 1, mol/cm^3
d_e	equivalent diameter ($2h$ for a channel), cm
D_1	diffusion coefficient of species 1, cm^2/s
Gz	Graetz number ($\frac{\pi}{4} Pe \frac{d_e}{L}$ for a channel)
h	interelectrode gap thickness, cm
I_{axial}	current density flowing in the axial direction, A/cm^2
i_{an}	anode current density, A/cm^2
i_{cath}	cathode current density, A/cm^2
L	electrode length, cm
Pe	Péclet number ($2 < v > h/D$ for a channel)
x	axial coordinate, cm
y	normal coordinate, cm

Greek

κ_∞	conductivity of feed solution, ohm^{-1} cm^{-1}
Φ	potential in the solution as measured by a reference electrode of a given kind, V
Φ^*	integration constant in equation 3-3

Subscripts

an	anode
cath	cathode
1	species 1
2	species 2
∞	feed

4. Experimental Apparatus and Procedure

4.1. Introduction

To provide experimental confirmation of the model results, steady-state polarization curves (current *vs.* voltage) were measured in a channel flow cell containing the potassium ferri-/ferrocyanide redox couple in aqueous sodium hydroxide. The nickel electrodes, forming a portion of the walls of the flow channel, were separated by a polyethylene spacer. By varying the spacer thickness and the feed flow rate, delivered from a syringe pump, various Graetz-number regimes were studied. Since the polarization curves included a wide range of applied potentials, kinetically-controlled and mass-transfer-controlled currents were measured, but because the electrodes were not sectioned, no experimental information about the distribution of current was obtained.

Separate experiments with a rotating-disk electrode were carried out to measure the diffusion coefficient of the ferricyanide ion. Although many investigators had measured the diffusion coefficient of ferricyanide in basic solutions, their measurements were for equimolar solutions of ferri- and ferrocyanide. Since equimolar solutions are undesirable for limiting current measurements in a channel (see section 4.2), diffusion coefficients were required for solutions with excess ferrocyanide. Both Acosta[76] and Fischl[77] used such solutions, but they calculated their diffusion coefficients from the correlation developed by Gordon *et al.*[78] Although this correlation should be accurate, we felt that we might achieve better agreement between theory and experiment with a measured diffusion coefficient.

In section 4.2, we discuss the rationale for the selection of the electrochemical reaction, electrode materials, and reference electrode. Section 4.3 describes the electrolyte preparation. The flow channel itself is described in detail in section 4.4.1, and section

4.4.2 details the operating procedure. The rotating disk experimental procedure is outlined in section 4.5.

4.2. Selection of the Electrochemical System

The potassium ferri-/ferrocyanide redox couple has been used extensively for mass transfer studies.[77-91] Acosta[76] used the ferri-/ferrocyanide system to examine the economic tradeoff between increased mass transfer and increased pumping power in thin-gap channel flow-cells. He measured mass-transfer coefficients and friction factors in both the laminar and turbulent regimes and studied the effects of surface roughness. Fischl[77] investigated the effects of turbulence promoters on mass transfer and pressure drop in a channel flow cell.

The decision to use the $Fe(CN)_6^{3-}/Fe(CN)_6^{4-}$ redox couple for the present study was based on previous studies, which have shown that redox systems in general are suitable for forced-convection mass transfer studies because, compared to plating systems, they have less free convection (smaller density changes), they reach steady state faster, and the electrode surface stays smooth. The $Fe(CN)_6^{3-}/Fe(CN)_6^{4-}$ reaction in particular is fast and reversible, and it gives a long limiting-current plateau. Although we were interested in the entire polarization curve, the long plateau is advantageous for studying interacting boundary layers at the limiting current.

There are some disadvantages of the ferri-/ferrocyanide redox couple. One such disadvantage is that the solutions contain cyanide ions, which can react with hydrogen ions to form poisonous[92] HCN gas. This reaction is suppressed in basic solutions. Another disadvantage is that the solutions are unstable due to the light-induced decomposition of ferrocyanide (see equations 4-1 and 4-2) and the reaction of ferrocyanide with any dissolved oxygen to form ferricyanide and OH^-. A third disadvantage is that dissolved oxygen reacts on the cathode at approximately the potential of the ferricyanide

reduction, thus obscuring measurements of the current from the main cathodic reaction. The problems of the oxygen side reaction and the solution decomposition can be avoided by sparging the solution with nitrogen to remove the dissolved oxygen and by not storing the solution for more than a few days. Finally, the problem of electrode poisoning by cyanide ion has been reported, but it can be eliminated by careful washing and pretreatment of the electrodes,[76,89,90] which are usually nickel or platinum. We chose nickel because nickel electrodes are less susceptible to the poisoning than platinum and they are less expensive.

Having chosen the redox couple and electrode material, the next question was whether to use excess supporting electrolyte. In this study, supporting electrolyte is desirable to eliminate migration, which is neglected in the model. Excess supporting electrolyte is generally used in industrial cells to lower the cell resistance, so migration was neglected in the model and suppressed in the experiments.

Basic supporting electrolyte (NaOH) was chosen to hinder the HCN side reaction, which would occur in acid solution or by the decomposition of ferrocyanide in the presence of light:

$$Fe(CN)_6^{4-} + H_2O \overset{light}{\rightleftarrows} [Fe(CN)_5 \cdot H_2O]^{3-} + CN^- \qquad (4\text{-}1)$$

$$CN^- + H_2O \rightleftarrows HCN + OH^- . \qquad (4\text{-}2)$$

By adding excess OH^- ions, the equilibrium of the second reaction is driven to the left. Other advantages of basic supporting electrolytes compared to neutral supporting electrolytes are the higher solution conductivities and the longer cathodic limiting-current plateaus.

The feed composition was chosen so that the cathode (working electrode) is at the limiting current and the anode is below limiting current. Since the anode and cathode surface areas are equal in the channel and the diffusion coefficients are roughly equal (D_{Ferri} is about 20% less than D_{Ferro}),[76] the only way to ensure that the anode is below

limiting current is to use a solution with excess ferrocyanide. It is important that the counterelectrode be below the limiting current if the mass-transfer limitations are to be attributed to the working electrode. In limiting-current studies with a redox system, either electrode may be the limiting electrode. With the $Fe(CN)_6^{3-}/Fe(CN)_6^{4-}$ redox system, there are advantages to choosing the cathode as the working (limiting) electrode: in basic solutions, the cathode has the longer limiting-current plateau and oxygen evolution on the anode is minimized.

Clearly, the long limiting-current plateau is desirable for accurate measurements of the limiting current. The reason the cathodic limiting-current plateau is longer is that the cathode has the larger difference between the thermodynamic potentials of the main and side reactions. The main reaction is

$$\text{cathode: } Fe(CN)_6^{3-} + e^- \rightarrow Fe(CN)_6^{4-} \quad (\text{Ferri} \rightarrow \text{Ferro}) \tag{4-3}$$

$$\text{anode: } Fe(CN)_6^{4-} \rightarrow Fe(CN)_6^{3-} + e^- \quad (\text{Ferro} \rightarrow \text{Ferri}). \tag{4-4}$$

The side reactions are

$$\text{cathode: } 2H_2O + 2e^- \rightarrow H_2 + 2OH^- \tag{4-5}$$

$$\text{anode: } 4OH^- \rightarrow O_2 + 2H_2O + 4e^-. \tag{4-6}$$

Figure 4-1 [89] is a Pourbaix diagram showing the effect of pH on the length of the anodic and cathodic plateaus. The dotted horizontal line at 0.36 V represents the standard electrode potential for the ferri-/ferrocyanide redox couple, the line labeled "oxygen potential" is the standard potential of the anode side reaction as a function of pH, and the hydrogen-potential line is for the cathode side reaction. The length of the cathodic plateau is roughly $V^o_{Ferri/Ferro} - V_{H_2}{}^o$. The diagram shows that, in basic solutions, the cathodic plateau is longer.

The other advantage of using the cathode as the working electrode is that the

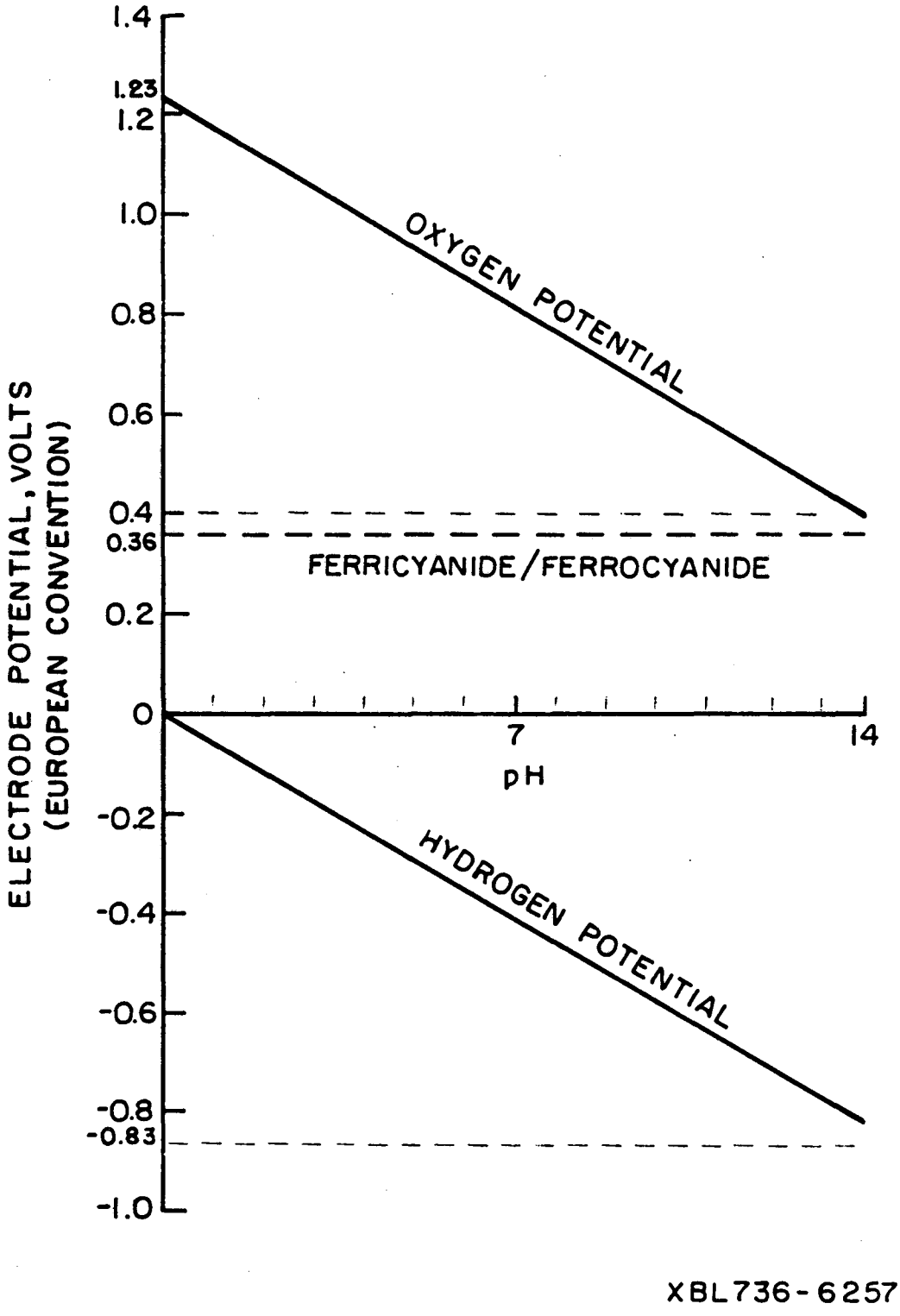


Figure 4-1. Electrode *vs.* pH diagram.

oxygen-evolution side reaction can be controlled by forcing the anode to be below limiting current. Oxygen evolution is undesirable because it may produce an oxide film on the anode and, if the boundary layers are thick, the oxygen produced on the anode can react on the cathode. Also, oxygen reacts homogeneously with ferrocyanide to produce ferricyanide. To guarantee limiting current on the cathode only, the electrolyte was chosen to have a 5:1 excess of ferrocyanide.[76,77] Excess sodium-hydroxide supporting electrolyte was used to suppress migration and the HCN side reaction.

The final task was to choose a reference electrode, although it is often omitted in studies requiring only the limiting current. Sections 34 and 35 of Newman's book[28] give some general comments on the selection of a reference electrode. For example, a good reference electrode is reproducible, and the reaction is reversible (it has a high exchange current density). Since the ferri-/ferrocyanide reaction itself meets these criteria, it makes a good reference-electrode reaction. The advantage of a ferri-/ferrocyanide reference electrode is that the liquid-junction potential is minimized because the electrode is placed in the feed solution; the disadvantage is that the open-circuit potential provides no information about the composition of the feed solution.

A platinum-screen reference electrode was chosen because platinum is inert and less susceptible to poisoning than nickel and because a screen electrode has a high surface area, which is desirable to minimize the current density flowing through the reference electrode (the smaller the current density, the smaller the overpotential at the reference electrode, which represents a deviation from its equilibrium potential).[28]

4.3. Electrolyte Properties and Preparation

The electrolyte compositions are tabulated below. The compositions were chosen to allow comparisons with other investigations. Solutions 1 and 2 were used in the channel experiments, and solutions 1 through 4 were used in the disk experiments.

Solution	NaOH	KOH	$K_3Fe(CN)_6$	$K_4Fe(CN)_6$
1. Fischl[77]	0.3 M		0.01 M	0.05 M
2. Eisenberg[79]	2.0 M		0.05 M	0.05 M
3. Selman[87]	0.4 M		0.0143 M	0.0143 M
4. Mohr[86]		0.85 M	0.005 M	0.005 M

All solutions were prepared from deoxygenated, deionized water. The water was deoxygenated in 500 ml batches by sparging with nitrogen for half an hour before the solution preparation and just before the experiment. We found that the solutions must be prepared with sparged water. When the solutions were sparged once before the experiment but not during the solution preparation, the results were scattered and irreproducible.† On the other hand, the carefully-prepared solutions gave reproducible results that fit the theoretical predictions (see chapter 5). A SYBRON/Barnstead ROpure purification system deionized the water to a resistivity of 16-18 MΩ-cm. The physical properties of the solutions are tabulated in appendix H.

4.4. Channel-Flow-Cell Apparatus and Procedure

4.4.1. Experimental Apparatus

Figures 4-2 and 4-3 show the experimental set-up, and figures 4-4 and 4-5 show the channel flow cell. The major equipment — the channel flow cell, syringe pump, and potentiostat — are described below.

The channel flow cell, shown in figures 4-3 through 4-5, consists of parallel plates of lucite, each containing a section with a 6 mm thick nickel 200 electrode (99.0% pure) embedded in Shell Epon 828 oven-cure epoxy resin. The epoxy resin was chosen based on previous work[76] to provide a smooth junction between the electrode and the insulator

† These results were discarded.

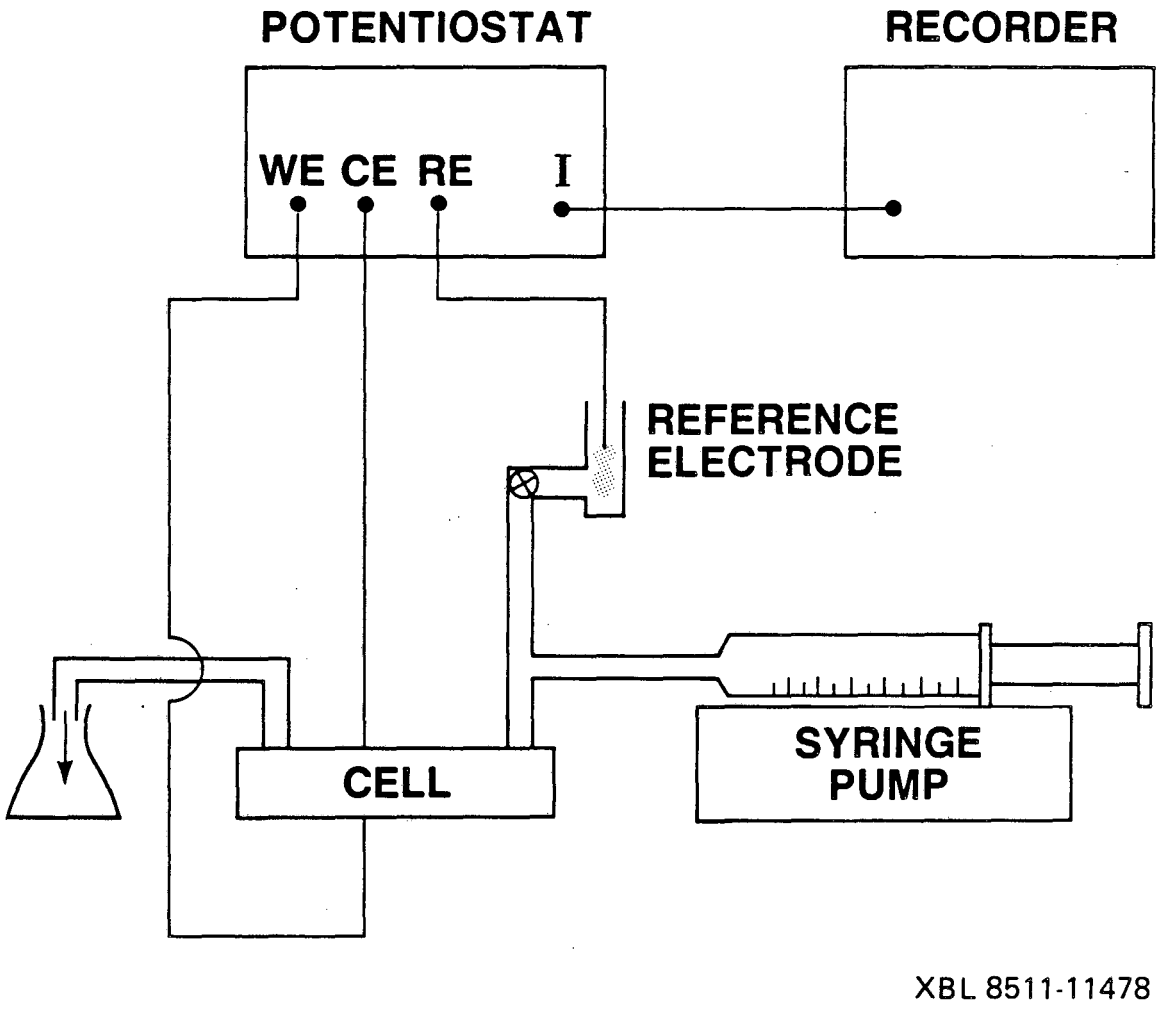
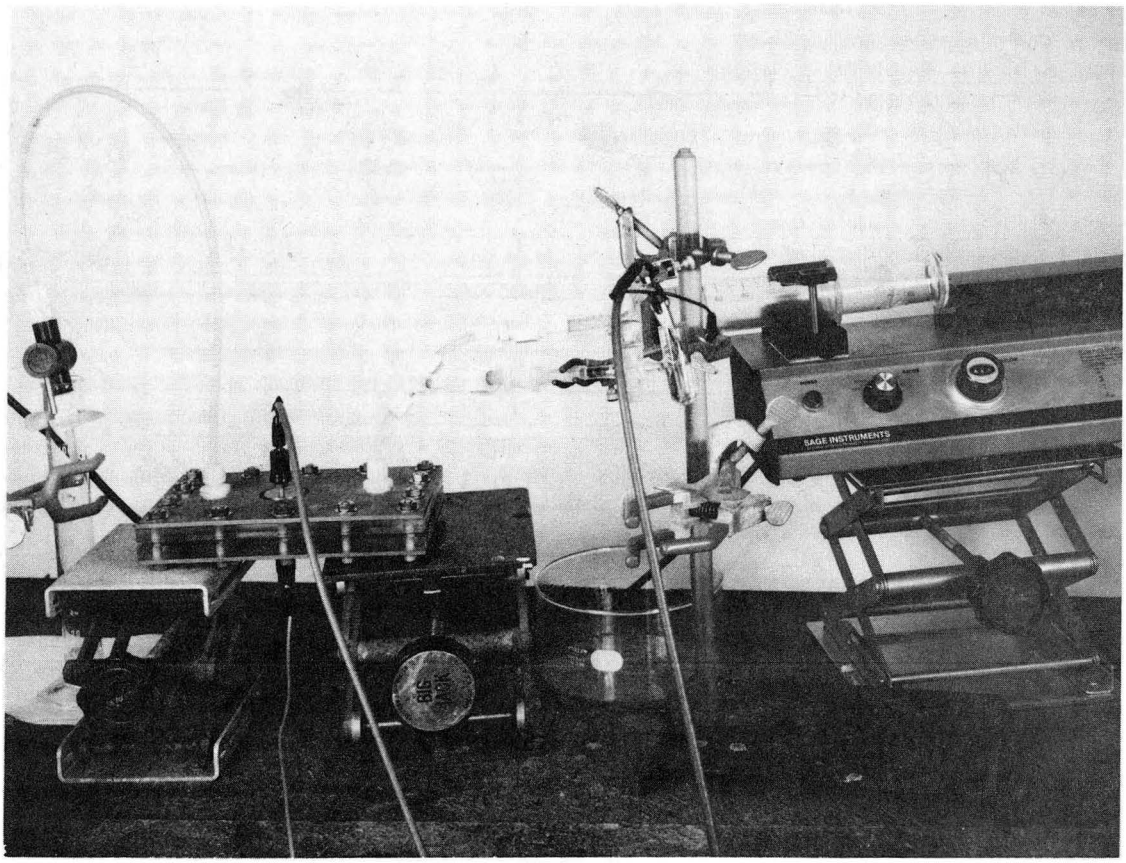
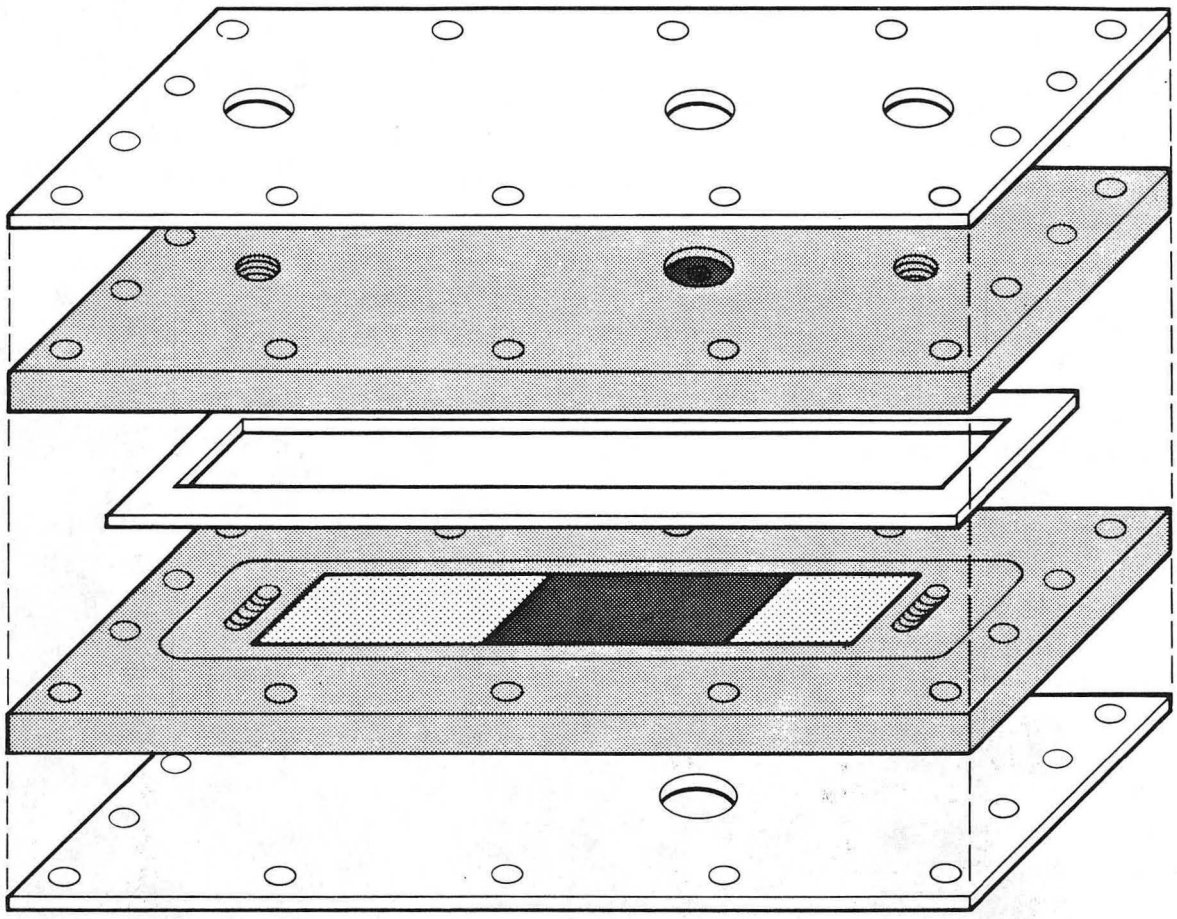


Figure 4-2. Schematic of experimental set-up.



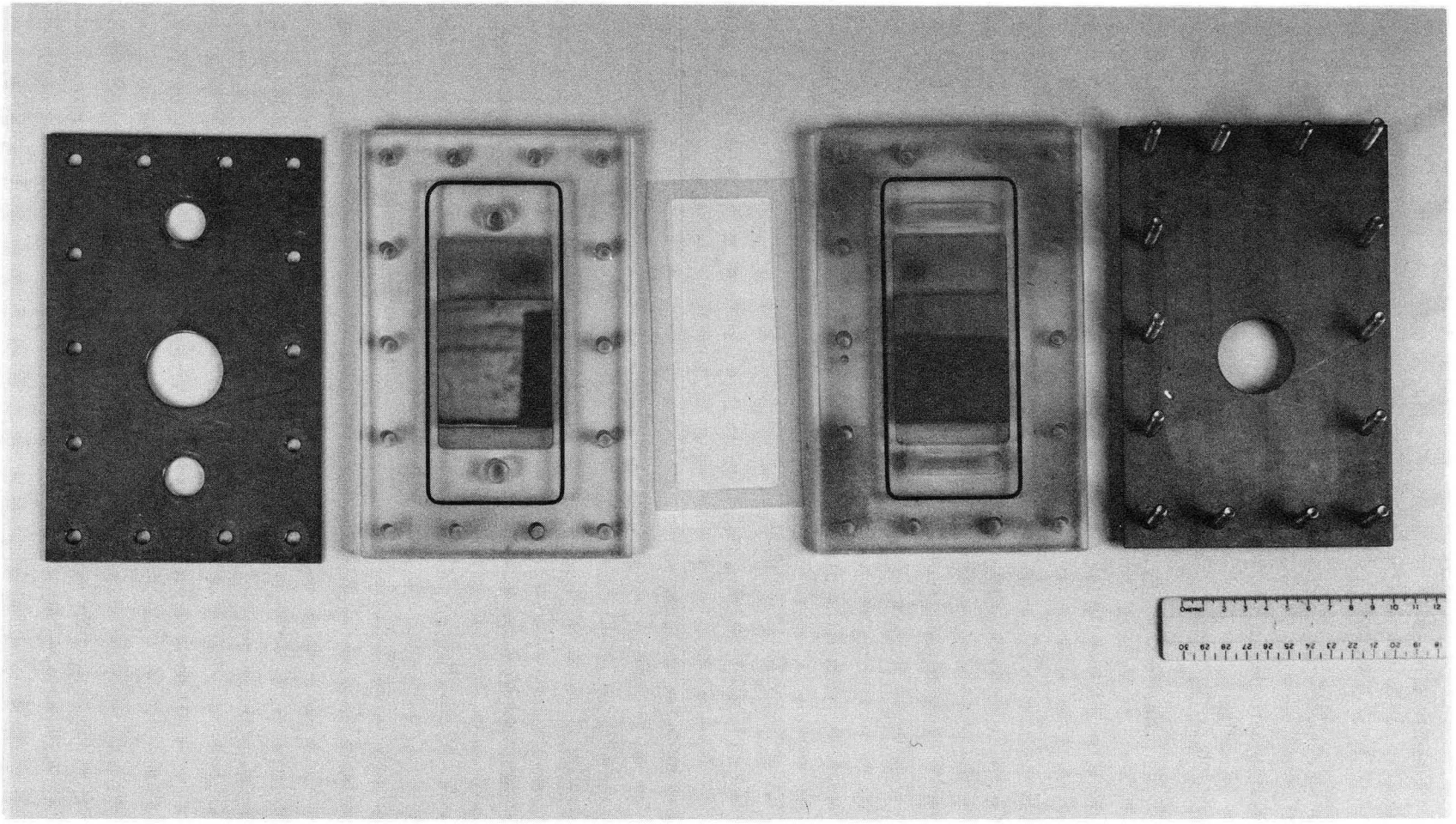
CBB 863-1512

Figure 4-3. Channel-flow-cell assembly (shown with the 0.323 cm spacer) and syringe pump.



XBL 861-8628

Figure 4-4. Assembly drawing of the channel flow cell showing nickel cover plates, electrode plates, and polyethylene spacer.



CBB 863-1508

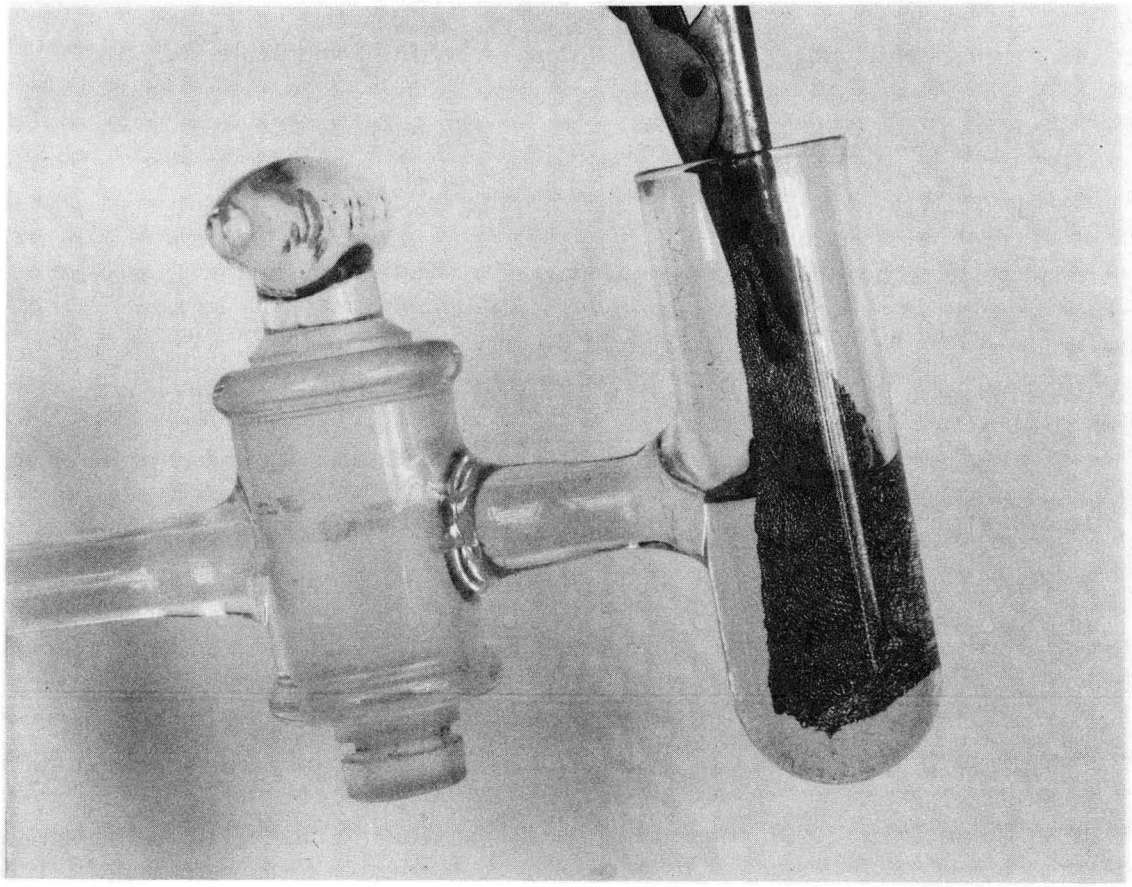
Figure 4-5. Nickel cover plates, electrodes, and thin (0.043 cm) spacer.

on polishing. To ensure laminar flow, an entry length of 3 cm on the epoxy surfaces and an exit length of 1 cm are provided before and after the metal electrodes. With the thin spacer, these lengths are 30 d_e and 10 d_e, respectively. Electrolyte enters and exits the cell through circular holes in the top plate. The holes are located over recesses in the lower plate, which are provided to act as distribution and collection plenums. The edges of the recesses are rounded (see figures 4-4 and 4-5). Electrical connections are made by banana plugs, screwed into blind-tapped holes on the back of each nickel electrode. There was no concern about using dissimilar metals because the cell was isothermal, and hundreds of millivolts were applied to the cell (except near open circuit, where the error in the potential measurement was about 1 millivolt).

A polyethylene spacer of either 0.043 cm or 0.323 cm thickness separates the plates. The spacer is sealed against the plates by Viton O-rings seated in grooves in the plates. Fourteen screws around the perimeter of the cell assembly act on 0.318 cm thick nickel cover plates, which are used to distribute the forces on the lucite plates, thus providing uniform pressure on the O-rings.

Fluid is pumped into the cell with a Sage model 355 syringe pump, which can deliver a maximum of 100 ml at any flow rate between 0.39 μl/hr and 126 ml/min. Flow rates are measured by the bucket-and-stopwatch method. The syringe is a Bekton-Dickenson 100 ml glass syringe with a corrugated uranyl glass fitting added to the tip. Three-eighth inch O.D. BEV-A-LINE V tubing is heat-shrunk over the corrugated tip and fastened to a teflon Swagelok fitting screwed into the entrance hole in the top plate of the cell assembly.

The platinum-screen reference electrode is placed upstream of the channel in a small reservoir containing the feed solution (see figures 4-3 and 4-6). A ground-glass stopcock prevents solution from flowing into the reference-electrode compartment but allows electrical connection to the working electrode by a thin film of electrolyte



CBB 863-1510

Figure 4-6. Reference-electrode side arm.

surrounding the stopcock (see figure 4-6).

Potential was applied to the cell with an AIS (Division of Floyd Bell) "Aardvark" model V-2LR-D potentiostat. A Pine Instruments RDE 3 potentiostat, tried in preliminary experiments under the same conditions, proved to be unstable.

4.4.2. Experimental Procedure

Before every four or five runs, the electrodes were polished down to one-micron diamond paste. Since the plates were too large for effective polishing with an 8- or 10-inch polishing wheel, the electrodes were hand-polished with a flat disk covered with a velvet polishing cloth. The plates were then washed with Liquinox soap and rinsed with deionized water and methanol, and the cell was assembled as shown in figure 4-4. The electrodes were not cleaned further by hydrogen evolution. A better experimental design would have provided a dummy counterelectrode to use for the hydrogen-evolution cleaning procedure, so that both channel electrodes would remain cathodically activated.

After the electrodes were polished, the cell was assembled, and the syringe was filled. Because the syringe was too large to draw the solution into the tip, it was filled from the top. Care was required to prevent the ground-glass plunger from jamming in the syringe barrel, so the plunger was greased before each run with Dow-Corning high vacuum (silicone) grease. The syringe was then connected to the cell, which was filled in the vertical position to eliminate air bubbles. The cell assembly was then leveled with anode over cathode to suppress free convection, and the syringe pump was started. After the flow became steady, the stopwatch for measuring flowrates was started.

The choice of flow rate depended on the desired Graetz number. For measuring diffusion coefficients, for example, high Graetz numbers (high flow rates) were used, so that the Lévêque approximation applies:

$$Nu_{avg} = 1.8488\ (Re\ Sc\ \frac{d_e}{L})^{1/3}. \qquad (4\text{-}7)$$

Polarization curves were obtained by measuring cathodic currents at various applied potentials and recording the current *vs.* time to check visually for steady state before moving to a new potential. To verify that the electrodes were not poisoned during the experiment, a data point on the limiting-current plateau was taken at the beginning and end of each run. If the limiting currents measured at the beginning and end of the run were significantly different, the data were discarded.

4.5. Rotating Disk Experiments

To measure the diffusion coefficient of $Fe(CN)_6^{3-}$ with the rotating disk system, the limiting current is measured at various rotation speeds, and, since the Levich equation[93] applies,

$$I_{\text{lim}} = 0.620nFAD^{2/3}\Omega^{1/2}\nu^{1/6}c_b\ , \qquad (4\text{-}8)$$

the diffusion coefficient is obtained from the slope of a plot of I_{lim} *vs.* $\Omega^{1/2}$.

4.5.1. Experimental Procedure

The disk experiments were carried out with a 5-mm platinum disk (Pine Instruments) and a Pine model PIR rotator. Experiments with a smaller nickel disk (1 mm) were unsuccessful, probably because of spherical diffusion effects. The potential of the working electrode was controlled by a Princeton Applied Research model 173 potentiostat with the PAR 175 Universal Programmer generating the potential sweeps. The platinum-screen reference and counterelectrodes were placed in the side-arms of the vessel (see figure 4-7).

The solutions were prepared, as in section 4.4, and sparged for one hour in the 300-ml vessel. Table 4-1 shows the four solution compositions used, corresponding to those of Fischl,[77] Eisenberg,[79] Selman,[87] and Mohr.[86]

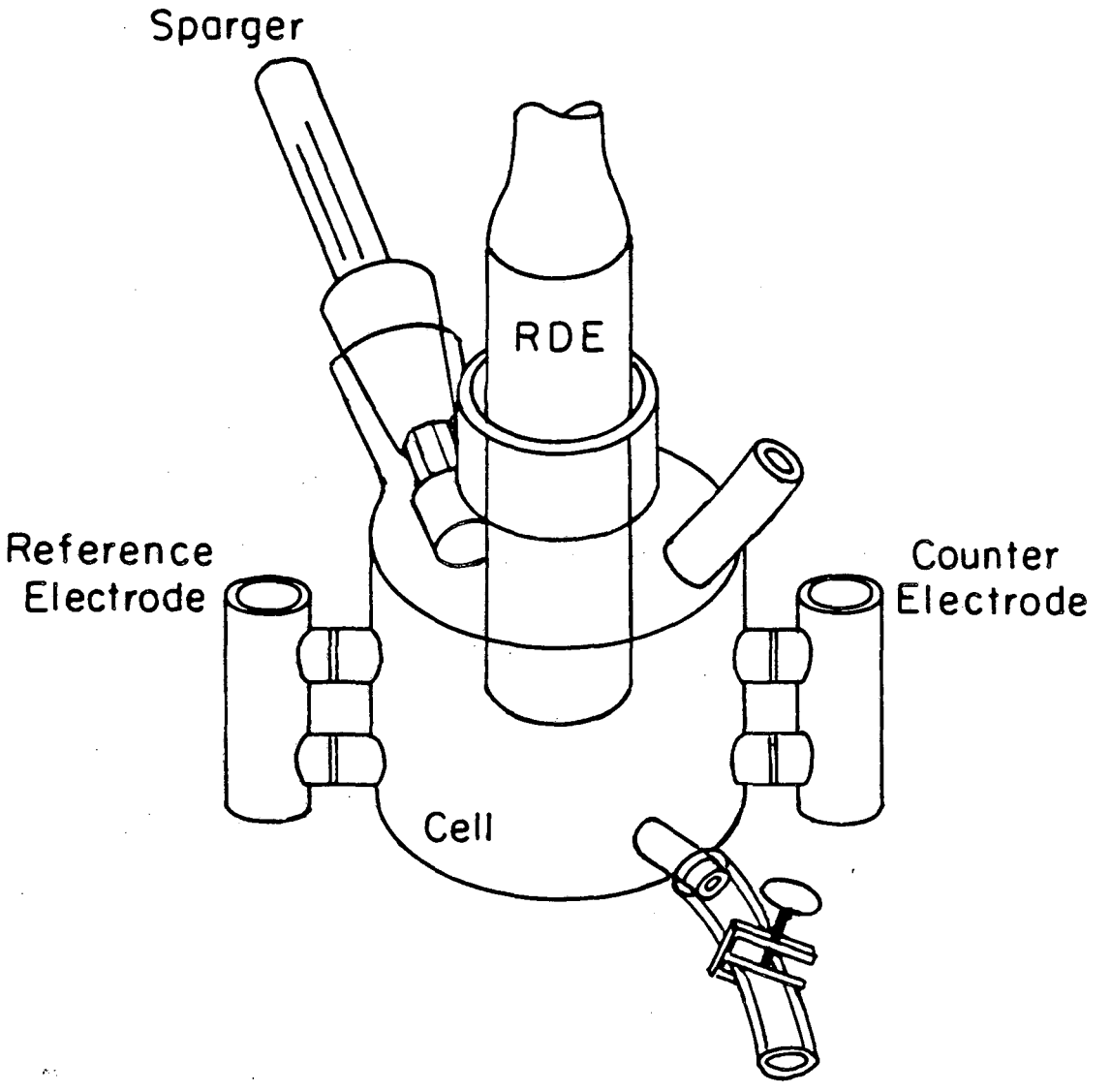


Figure 4-7. Sketch of rotating disk electrode apparatus.

Table 4-1. Solution Compositions

Solution	NaOH	KOH	$K_3Fe(CN)_6$	$K_4Fe(CN)_6$
#1 Fischl	0.3 M		0.01 M	0.05 M
#2 Eisenberg	2.0 M		0.05 M	0.05 M
#3 Selman	0.4 M		0.0143 M	0.0143 M
#4 Mohr		0.85 M	0.005 M	0.005 M

The electrodes were prepared by polishing down to one-micron diamond paste, washing with Liquinox soap, and rinsing with deionized water and acetone. Before each potential sweep, the working electrode was cleaned further by hydrogen evolution for two minutes. Recall that this extra cleaning procedure was not possible with the channel flow cell as designed. Potential sweeps of 5 mV/sec gave flat limiting-current plateaus for the rotation speeds of 400, 900, 1600, and 2500 rpm. Reproducibility was checked by repeating each experiment with a new 300-ml batch of solution and reversing the order of the rotation speeds (*i.e.* 2500, 1600, 900, and 400 rpm).

The results of both the disk and the channel experiments were reproducible and agreed well with the theory. One suggestion for improvement of the procedure is to obtain direct measurements of the physical properties of the solution, such as concentration and viscosity. Temperature control would also be helpful, although it may be somewhat difficult to achieve in the channel. A discussion of the experimental results can be found in section 5.3.

Nomenclature for Chapter 4

A	surface area of electrode, cm^2
c_b	bulk concentration, mol/cm^3
D	diffusion coefficient, cm^2/s
d_e	equivalent diameter ($2h$ for a channel), cm
F	Faraday's constant, 96,487 C/eq
Gz	Graetz number ($\frac{\pi}{4} Pe \frac{d_e}{L}$ for the channel)
h	interelectrode gap thickness, cm
I_{lim}	limiting current, A
L	electrode length, cm
n	number of electrons transferred, eq/mol
Nu_{avg}	average Nusselt number
Re	Reynolds number ($2<v>h/\nu$ for a channel)
Sc	Schmidt number (ν/D)
$V^o_{Ferri/Ferro}$	standard electrode potential for the ferri- / ferrocyanide redox reaction, V

$V^o_{H_2}$	standard electrode potential for hydrogen evolution, V

Greek

ν	kinematic viscosity, cm^2/s
Ω	rotation speed, radians/s

5. Comparison of Experimental and Theoretical Results

5.1. Introduction

This chapter presents a comparison between experimental and theoretical results at various Graetz numbers.* At high Graetz numbers, the boundary layers are thin, and simple equations allow diffusion coefficients to be obtained from limiting current measurements either on a rotating-disk electrode or in the channel. For low Graetz numbers, we investigate interacting boundary layers in the channel.

Preliminary comparisons between the theoretical and experimental results had shown some discrepancies, even for thin-boundary-layer conditions, therefore we decided to measure the diffusion coefficient of ferricyanide for the solutions we were using, instead of calculating it from Gordon's correlation.[78] Diffusion coefficients are often measured on a rotating-disk electrode by the limiting current technique. The basis for this technique is that, since the boundary layer is thin, the limiting current may be calculated from the simple Levich equation (see chapter 4, equation 8). The same idea may be applied to the channel geometry because, if the boundary layers are thin, the limiting current may be calculated from the Lévêque equation. The important difference between the two geometries is that the mass-transfer limited current distribution is highly nonuniform in the channel, but uniform on the disk. Another difference between the two experiments is that the disk electrode is platinum and the channel electrodes are nickel. Thus, the two experiments provide independent measurements of the diffusion coefficient.

Sections 5.2 and 5.3 present comparisons between diffusion coefficients measured on the disk, in the channel, and in the literature. Section 5.4 presents measured and simulated polarization curves for the channel at low Graetz numbers (interacting boundary

* Recall that the Graetz number, defined in chapter 1, is a measure of the boundary-layer interaction.

layers).

5.2. Results for Thin Boundary Layers

This section presents results for thin boundary layers (high Graetz numbers) both on the rotating-disk electrode and in the channel. The purpose of these experiments was to measure the composition-dependent diffusion coefficient of the ferricyanide ion for the solution that was used in most of the channel experiments. (No previous measurements for this solution were found.) The disk measurements were carried out with four solution compositions — the first was the composition used in the thin gap channel experiments, the same composition used by Fischl,[77] and the other three were solutions for which diffusion coefficients have been measured by other investigators. In this way, we could compare our disk and channel measurements to each other for the first solution composition and compare our measurements to the literature for the last three solutions.

The results for each solution composition are presented together to show the comparison between the various measurements. Tables 5-1 through 5-3 show the four solution compositions. Note that only two solutions were analyzed and that the ferricyanide calculated by weighing agrees with the analysis. This conclusion was also verified in separate experiments that showed that the concentration of ferricyanide calculated by weighing and by iodometric titration with thiosulfate differed by about 1.5%. For this reason, the remaining solutions were not analyzed.

Limiting-current measurements for each of the four electrolytes are summarized in tables 5-4 through 5-6 and in appendix H. We shall now discuss the results for each solution.

Figures 5-1 and 5-2 show the Levich plots (see equation 4-8) from the rotating-disk experiments with the solution composition used by Fischl (solution 1). The diffusion

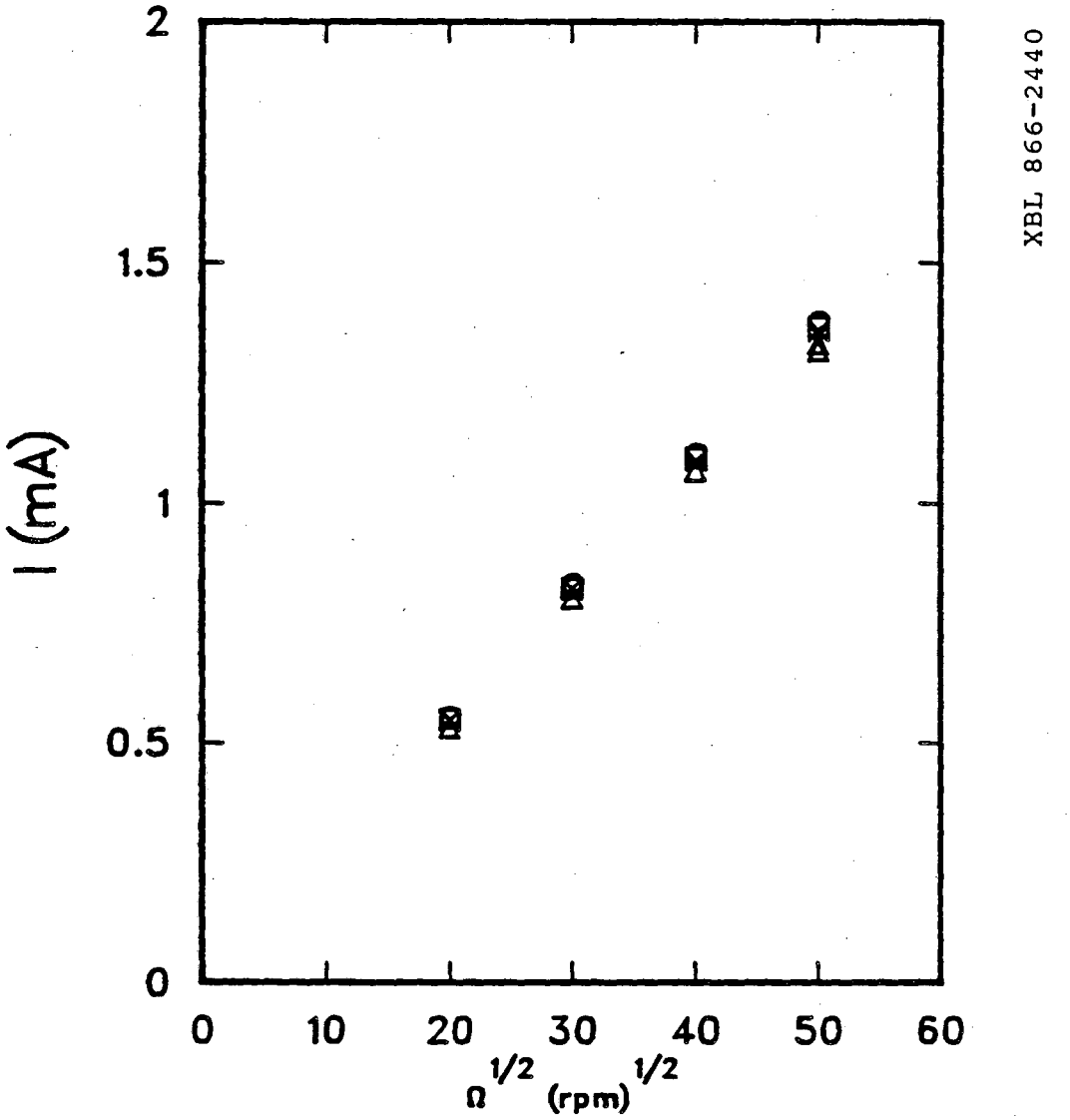


Figure 5-1. Levich plot for solution 1a.

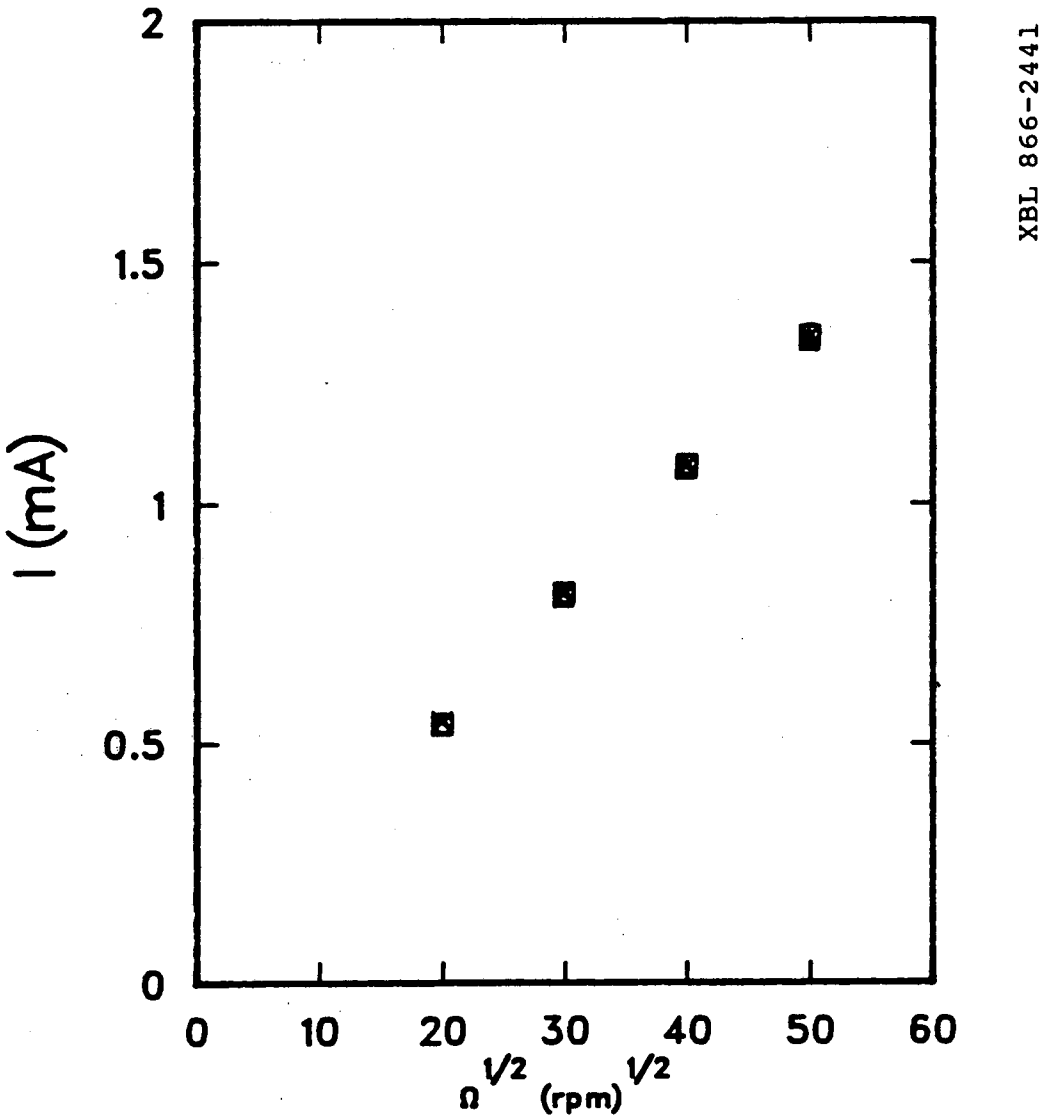


Figure 5-2. Levich plot for solution 1b.

Table 5-1. Nominal solution compositions (mol/liter).

Solution	NaOH	KOH	$K_3Fe(CN)_6$	$K_4Fe(CN)_6$
1. Fischl[77]	0.3 M		0.01 M	0.05 M
2. Eisenberg[79]	2.0 M		0.05 M	0.05 M
3. Selman[87]	0.4 M		0.0143 M	0.0143 M
4. Mohr[86]		0.85 M	0.005 M	0.005 M

Table 5-2. Solution concentrations for disk experiments (mol/liter).

Solution	NaOH	KOH	$K_3Fe(CN)_6$	$K_4Fe(CN)_6$
1a†	**0.2457**		**0.01000**	**0.05030**
	0.2582		0.01003	0.04983
1b	0.2992		0.01001	0.05003
2	2.1091		0.04993	0.05008
3	0.4015		0.01430	0.01479
4		0.7230	0.00472	0.05020

Table 5-3. Solution concentrations for channel experiments (mol/liter).

Solution	NaOH	$K_3Fe(CN)_6$	$K_4Fe(CN)_6$
1c	0.3190	0.01001	0.05009
1d	0.3415	0.01001	0.05109
1e†	**0.2832**	**0.01012**	**0.05758**
	0.2996	0.01002	0.050093
1f	0.301145	0.01004	0.05042
2	2.1091	0.04993	0.05008

† Boldface refers to solutions that were analyzed for ferricyanide by iodometric titration with zinc and for ferrocyanide by potentiometric titration with ceric sulfate. All other concentrations were calculated by weighing.

Table 5-4. Diffusion coefficients of $Fe(CN)_6^{3-}$ measured on the rotating-disk electrode: comparison to literature.

Solution	D (cm^2/s)	T ($^\circ$ C)	$\left(\frac{\mu D}{T}\right)^*_{est.}$ (dyne/K)	$\left(\frac{\mu D}{T}\right)_{lit.}$ (dyne/K)
1	6.19×10^{-6} ±3%	19.7	2.38×10^{-10}	$(2.35\times10^{-10})^*$
2	4.30×10^{-6} ±2%	–	2.30×10^{-10}	2.50×10^{-10}
3	5.98×10^{-6} ±2%	21.3	2.26×10^{-10}	2.60×10^{-10}
4	6.34×10^{-6} ±2%	20.0	2.38×10^{-10}	1.83×10^{-10}

Table 5-5. Comparison between diffusion coefficients measured on the disk and in the channel.

Solution	D_{disk} (cm^2/s)	$D_{channel}$ (cm^2/s)
1	6.19×10^{-6}	5.93×10^{-6}
2	4.30×10^{-6}	4.32×10^{-6}
3	5.98×10^{-6}	–
4	6.34×10^{-6}	–

coefficient obtained from the slope of these lines is $D_{Ferri} = 5.98\times10^{-6}$ cm^2/s ± 3%.

This "disk" diffusion coefficient may be compared to the diffusion coefficient measured in the channel. Figure 5-3 is a log-log plot of dimensionless limiting current ($Nu/(ReScd_e/L)$) *vs.* dimensionless flow rate (Red_e/L). The solid line is a fit of the data to the rearranged Lévêque equation,

$$\frac{Nu}{ReScd_e/L} = 1.8488(ReSc\frac{d_e}{L})^{-2/3}. \qquad (5\text{-}1)$$

The motivation for this equation is to plot a dimensionless current that does not contain the unknown diffusion coefficient. The channel diffusion coefficient, 5.93×10^{-6} cm^2/s, is obtained from the intercept of this line.

The results from solution 2 (see table 5-2) are shown in figures 5-4 and 5-5. Tables 5-4 and 5-5 summarize the data from the disk and channel; table 5-6 shows the

* Viscosities were estimated from the correlation of Boeffard *et al.*[85]

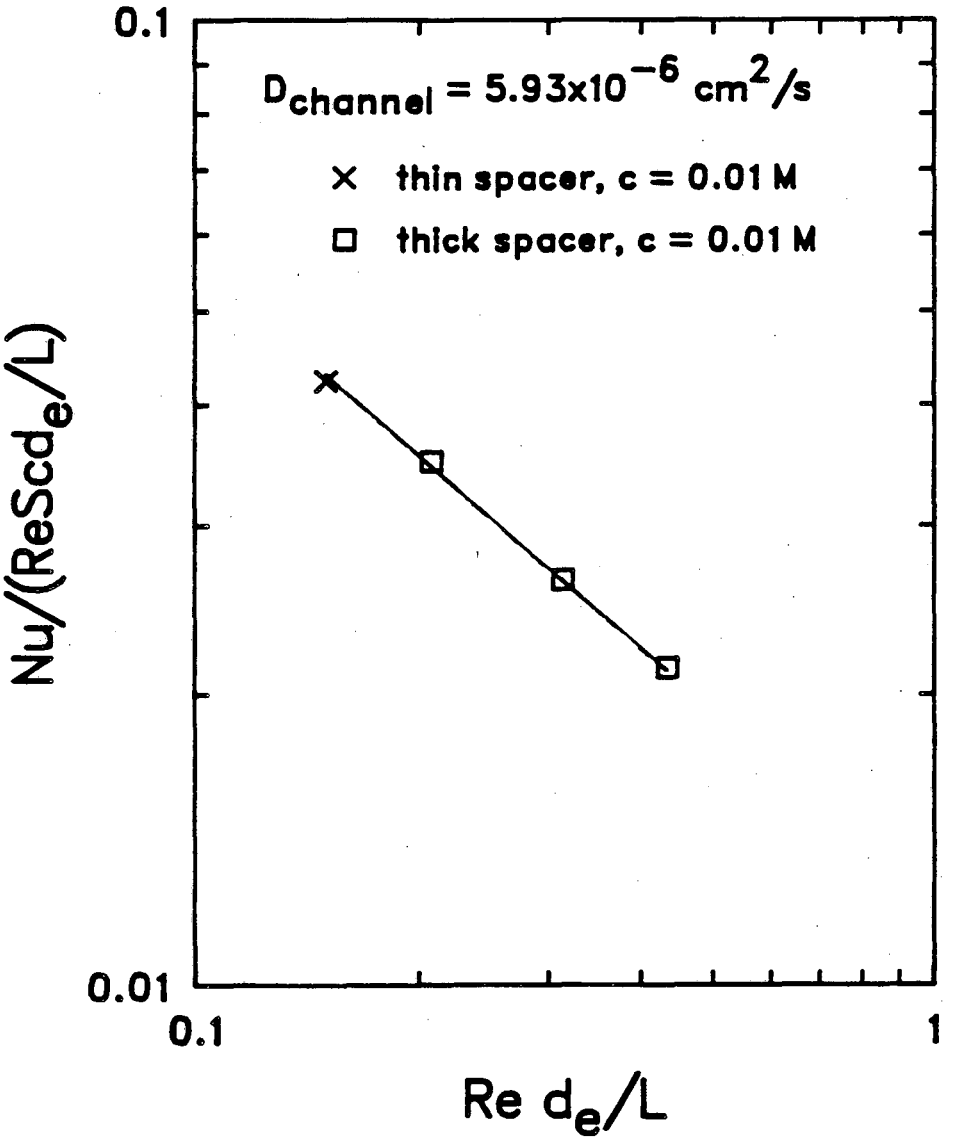


Figure 5-3. Fit of channel data to Lévêque equation for solutions 1c, 1d, and 1e.

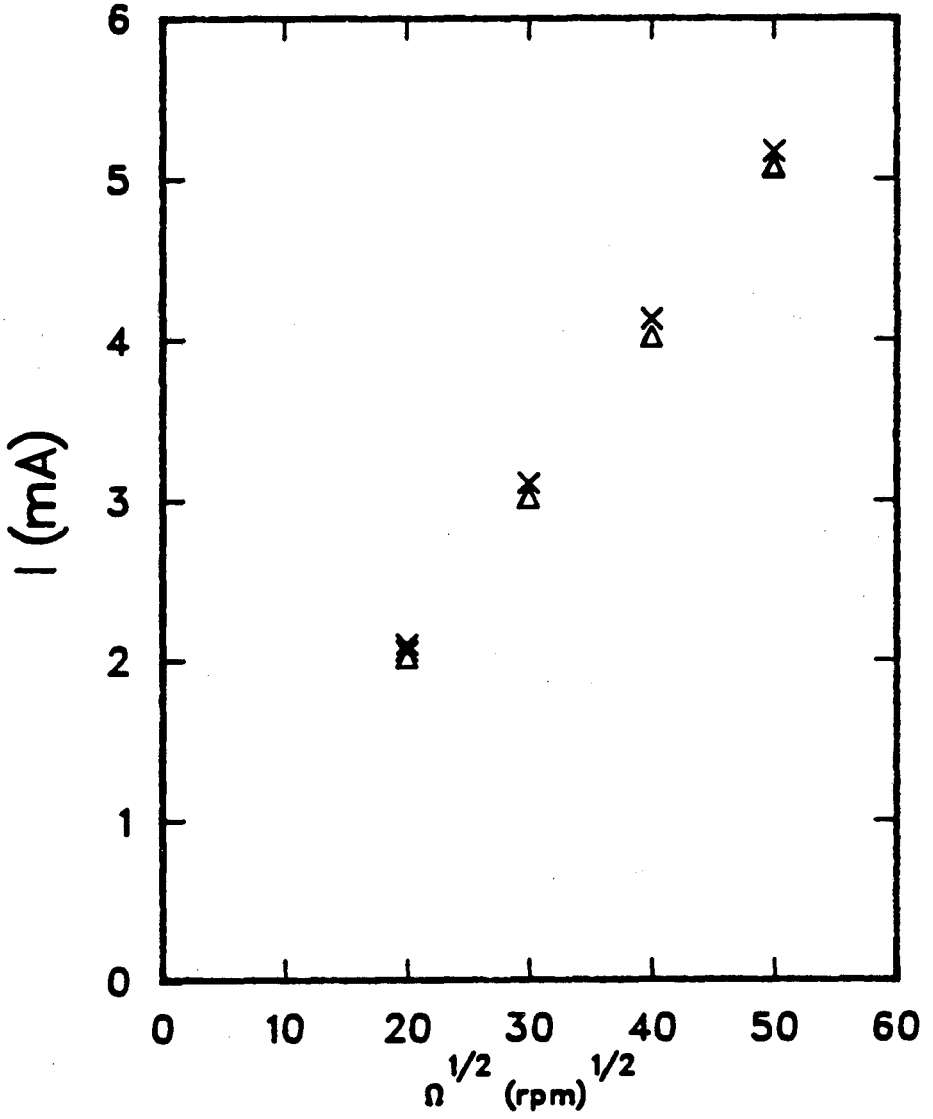


Figure 5-4. Levich plot for solution 2.

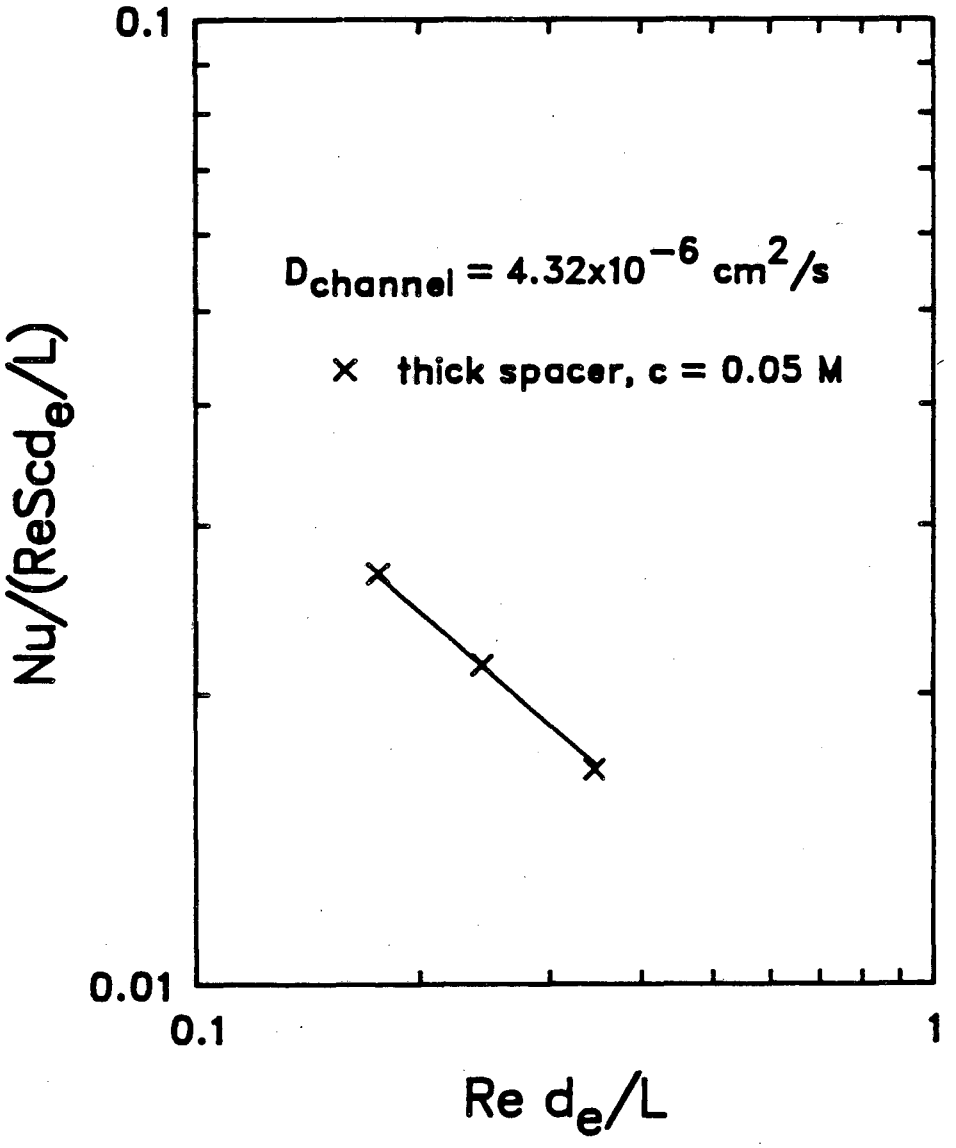


Figure 5-5. Fit of channel data to Lévêque equation for solution 2.

comparison to literature. There is some discrepancy between our data and Eisenberg's, but it is not surprising because the experimental techniques are different. Eisenberg used the capillary method, which is to immerse a capillary filled with the solution into a large reservoir of another solution that has no ferricyanide. The composition of the solution is then determined (by titration) after approximately one time constant later, and the diffusion coefficient is obtained from the solution of the one-dimensional transient diffusion equation. Although the titration measurements are generally accurate to 2 or 3%, the capillary method could yield diffusion coefficients that are too high if either spherical diffusion or convection is important. Also, the titration measurements may be less accurate for small volumes of solution. According to Gordon's correlation, which is based on accurate rotating disk measurements in equimolar solutions, the Stokes-Einstein coefficient should be lower than that predicted by Eisenberg; our measurements are only about 3% lower than Gordon's prediction (see table 5-6). Gordon's data suggest that $(D\mu/T)\times 10^{10}$ is never less than 2.34.

The results for solution 3 are shown in figure 5-6. These measurements are about 13% lower than those of Selman.[87] Figure 5-7 and table 5-4 show the results for Mohr's[86] KOH solution (solution 4). Here, our diffusion coefficient is higher than the direct measurement,[86] but close to the value predicted by the correlation of Gordon *et al.*[78]

The channel data for the different flow rates, concentrations, and gap thicknesses are summarized in figure 5-8. Note that all the data fit the Lévêque equation

$$Nu = 1.8488(ReSc\frac{d_e}{L})^{1/3}$$

because the Graetz number is high enough for the Lévêque approximation to apply. Table 5-6 summarizes all of the data from our experiments and the literature.

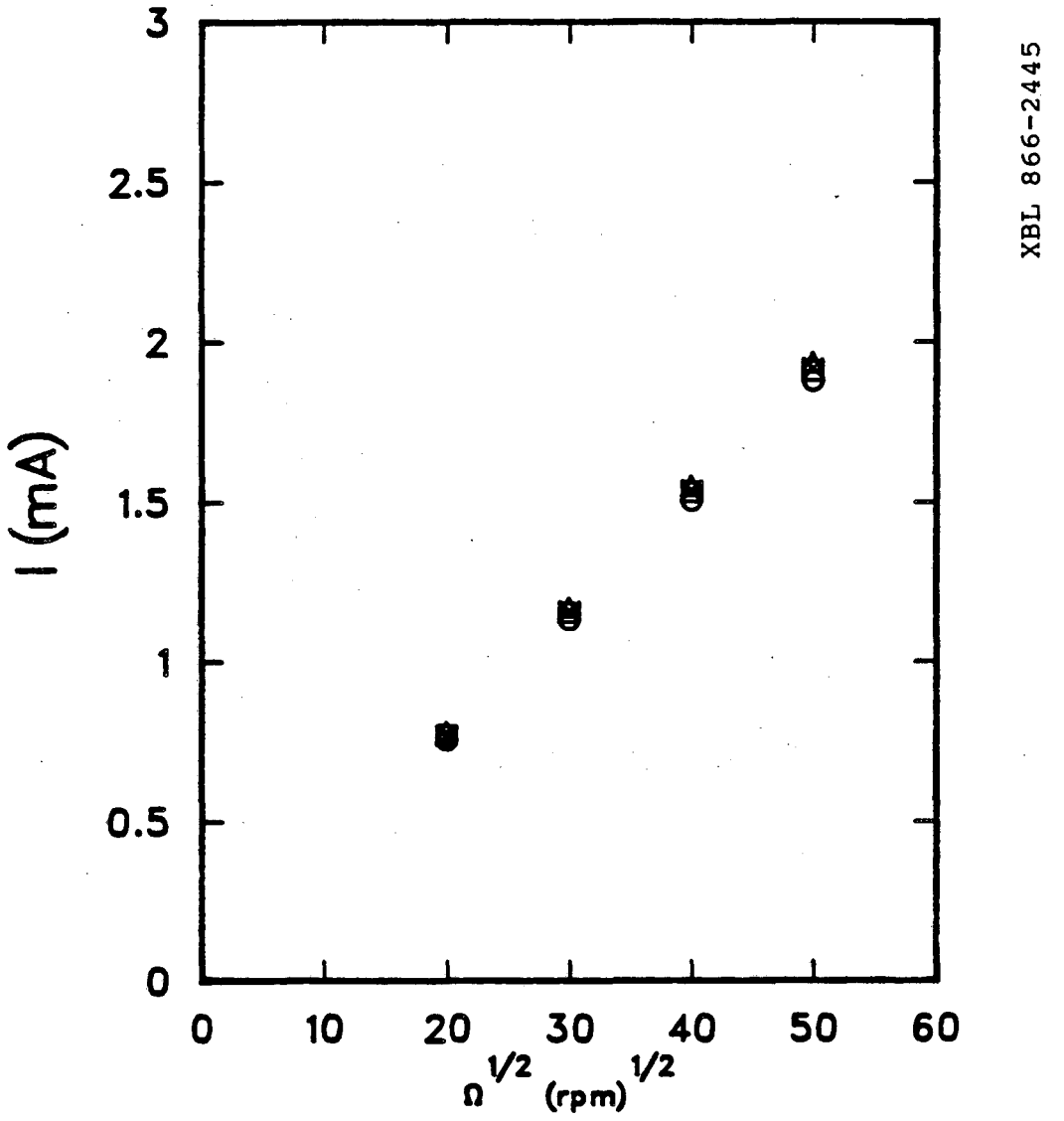


Figure 5-6. Levich plot for solution 3.

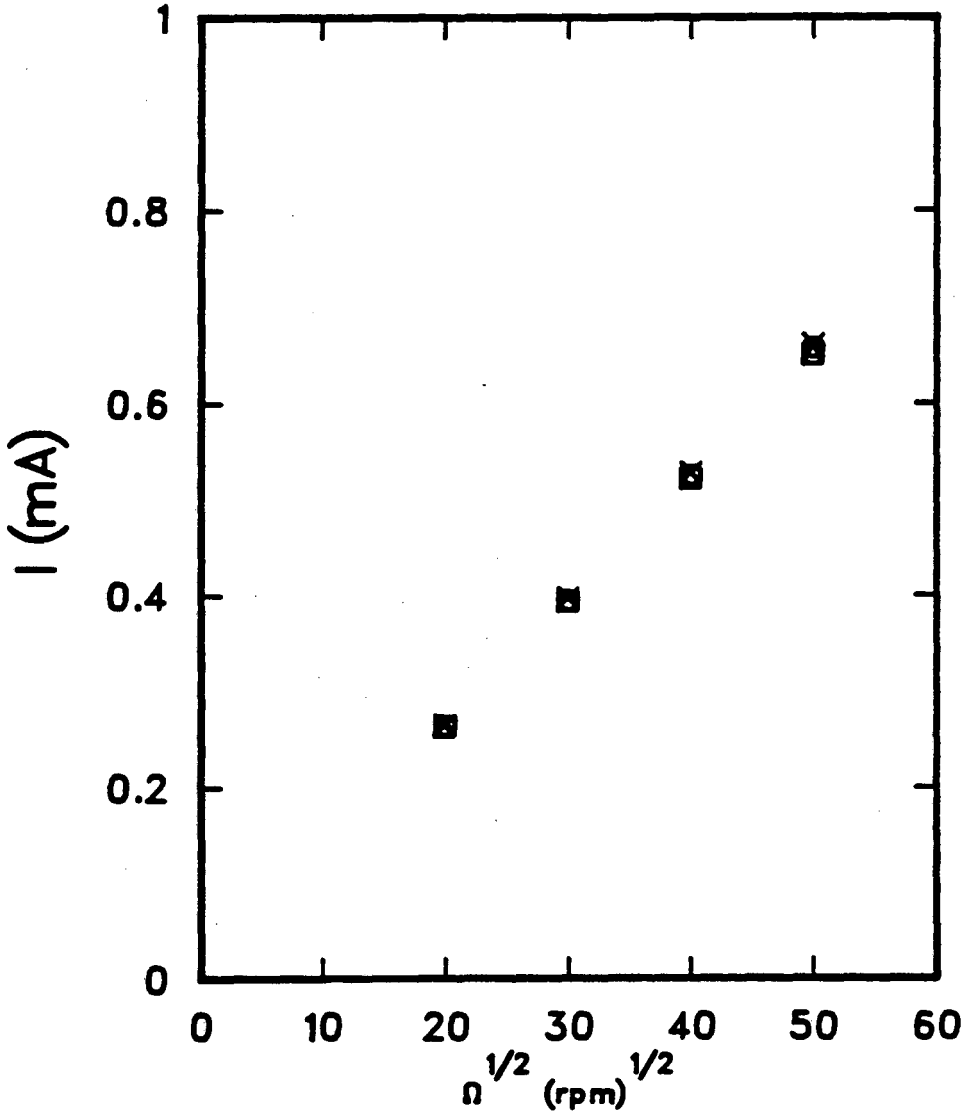


Figure 5-7. Levich plot for solution 4.

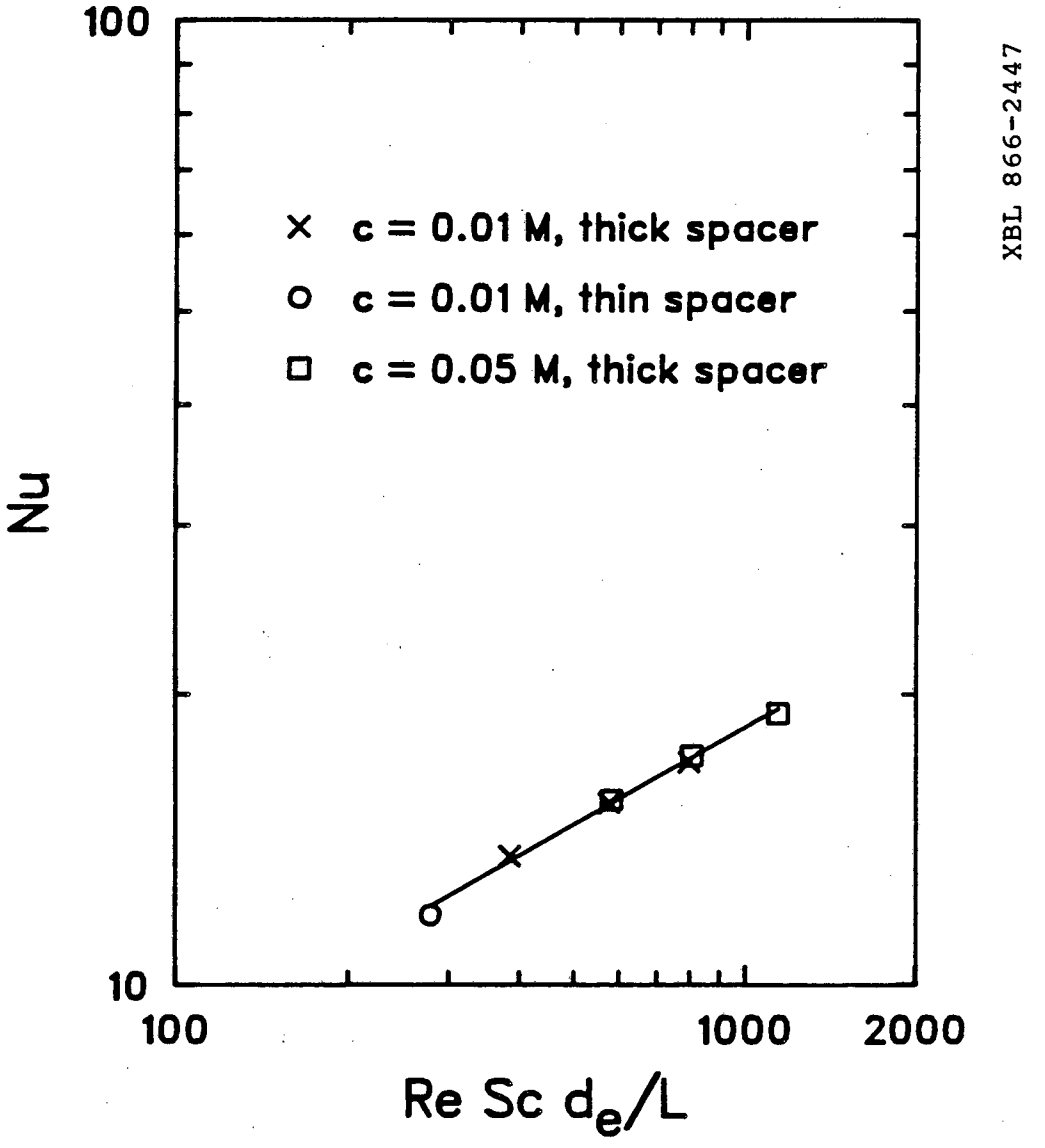


Figure 5-8. Lévêque plot of channel data using the diffusion coefficients obtained from figures 5-3 and 5-5.

5.3. Discussion

The accuracy of the experimental results clearly depends on the accuracy of the measurements of concentration, viscosity, and current. The discarded results of preliminary experiments with the rotating-disk electrode had shown that the solutions must be prepared with deoxygenated water and that it is not sufficient to remove the oxygen just prior to the experiment. The necessity of removing all oxygen stems from the reaction of oxygen with ferrocyanide to produce ferricyanide, which changes the composition of the solution (and renders it unknown). Although we were careful to keep out oxygen and light, some decomposition of the solutions may have occurred. A better procedure would be to work with oxygen-free solutions, use a reference electrode that is sensitive to ferri- and ferrocyanide, and analyze the solutions immediately before and after the experiments. The analysis of the effluent after the experiment would indicate whether oxygen had evolved on the anode, assuming no hydrogen is evolved on the cathode. When no oxygen is evolved, the concentration of the effluent is the same as that of the feed because the currents are equal on the two electrodes. On the other hand, if oxygen is evolved on the anode and is swept out before reacting homogeneously, more ferrocyanide is produced than ferricyanide; therefore the analysis of the effluent would reveal the total current for the oxygen reaction. The evolution of oxygen would only affect the results of the channel experiments with interacting boundary layers because the oxygen could diffuse to the cathode and react back to hydroxide. In thin boundary layer experiments, the limiting current would be the same, regardless of whether oxygen is evolved on the counterelectrode.

The viscosity of the solution is another property that should be determined accurately by direct measurement. Since the viscosity if sensitive to temperature, the experiments should be carried out in a constant-temperature bath. (In our experiments the temperature drifted by as much as 0.2 °C during the course of a run.)

Finally, the current is sensitive to flow patterns and to the electrode surface. As we found in preliminary experiments, free convection can change the current noticeably; therefore we leveled the cell with the anode over the cathode. Naturally, it is also important to remove any bubbles from the system. The electrode surface plays an important role because it can become poisoned by the cyanide ion. As mentioned previously, our experimental cell design should have provided a dummy counterelectrode to use for the cathodic-hydrogen-evolution cleaning procedure.

In addition to the improvements suggested above, we should consider the possibility of measuring the current by, for example, a sectioned-electrode technique. Another interesting way to learn about the current distribution without using a sectioned electrode is to use reference electrodes both upstream and downstream, as done by Matlosz.[94] The difference in potential between these two electrodes would indicate the amount of axial current, which can be significant even with a large aspect ratio (L/h).

Although our experiments could be improved by the techniques discussed above — use of an upstream and downstream reference electrode, analysis of the solutions before and after the run, measurement of the viscosity, constant-temperature control, and cathodic electrode pretreatment — the essential procedures — removal of oxygen, suppression of free convection, and electrode polishing — were carried out. This procedure led to reproducible results and, in general, the independent disk and channel measurements agree with each other and fall in the range of measurements by other investigators (see table 5-6).

Our measurements may be lower than some owing to the slight uncertainty in the viscosity and concentration of our solutions. Another possible cause of the discrepancy is that the nickel channel electrodes may have been partially poisoned by the cyanide ion. This explanation is unlikely, however, because the platinum disk electrode was

*Calculated from correlation of Gordon *et al.*[78]

Table 5-6. Values of $(\mu D/T)\times 10^{10}$ (dyne/K)

Ferri Ferro	0.01 M 0.05 M	0.05 M 0.05 M	0.01 M 0.01 M	0.005 M 0.005 M	Equimolar ferri/ferro			
Supp. Elec.	0.3 M NaOH	2 M NaOH	0.4 M NaOH	0.85 M KOH	KOH	NaOH	KCl	KNO_3
disk	2.06	1.92	2.03	2.25				
channel	2.09	2.10						
*Gordon[78]	2.35	2.38	2.35	2.36				
Eisenberg[79]		2.50						
Selman[87]			2.60					
Mohr[86]				1.83				
Arvía[95]		2.52	2.52	2.78	2.78	2.52	2.00	
Sih[88]								1.49
Appel[90]						1.68†		
Yip[89]						2.67‡		

cleaned carefully, and the diffusion coefficient measured with it was the same (for solution #1) or lower (solution #2) than that measured in the channel. Also, the channel diffusion coefficients were no higher with freshly polished electrodes.

---

* Calculated from correlation.

† From Smyrl's unpublished data, which Appel referenced as a personal communication. The unpublished D_{Ferri} is lower than the published[96] D_{Ferro}, contrary to what is expected.[78]

‡ Taken from Lin[97] by Yip.[89]

In summary, the measured diffusion coefficients are D=6.06×10^{-6} cm^2/s for an 0.01 M $K_3Fe(CN)_6$/ 0.05 M $K_4Fe(CN)_6$/ 0.3 M NaOH solution and D=4.31×10^{-6} cm^2/s for an 0.05 M $K_3Fe(CN)_6$/ 0.05 M $K_4Fe(CN)_6$/ 2.0 M NaOH solution.

5.4. Measured and Simulated Polarization Curves

Figures 5-9 through 5-18 show the comparison between the experimental and predicted results. The parameters used in the model are the measurements of the cell dimensions, the diffusion coefficient measured in the high-Graetz-number experiments, the thermodynamic parameters, and the kinetic parameters from Vetter.[98] As the figures show, the agreement is quite good without any adjustment of parameters. In the low-Graetz-number range, both the theoretical and experimental limiting currents are significantly higher than the value predicted from the Lévêque equation. The slight disagreements between the experimental and theoretical limiting currents are probably due to an uncertainty in the solution concentration, or, possibly, inaccurate flow measurements. The low-current region also shows slight discrepancies; the data show a steeper initial rise in the current *vs.* potential and an earlier appearance of mass-transfer limitations. Polarization curves obtained by other investigators, such as Eisenberg[79] and Mohr[86], show discrepancies of the same sign and magnitude. The parameters that typically affect this region are thermodynamic and geometrical parameters, and the transfer coefficients, which are all well-established and should not be adjusted. The exchange current density is less certain, but adjusting it will not improve the fit, as we shall see shortly.

The initial slope in the current-potential curve is often governed by kinetics and not mass transfer. In such cases, increasing the exchange current density will increase the initial slope because the surface overpotential is decreased. On the other hand, if the exchange current density is high, the surface overpotential is a negligible contribution to

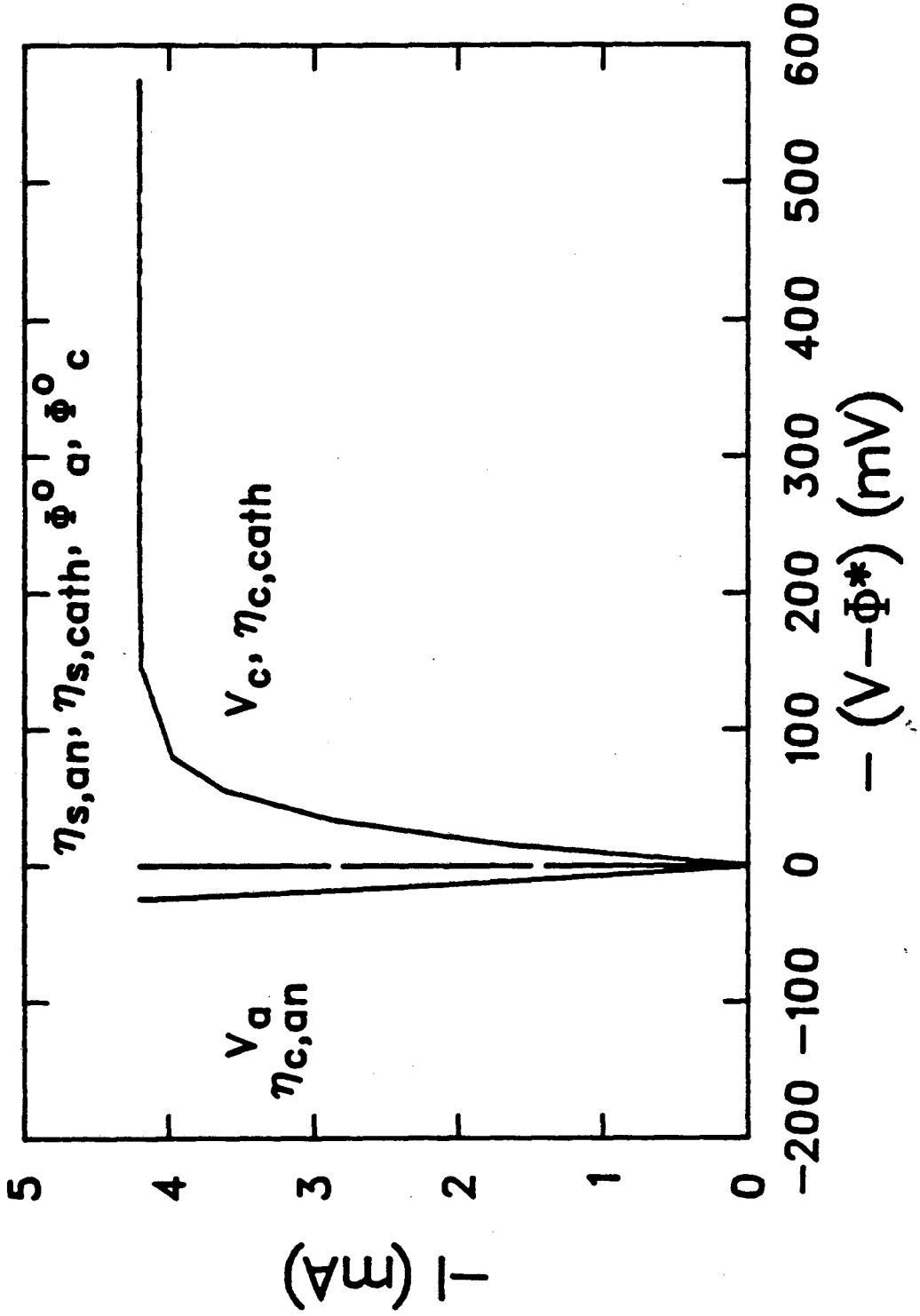


Figure 5-9. Components of the total cell potential for runs 1c-1 through 1c-4 with the thick (0.323 cm) spacer, $Ped_e/L = 586.15$.

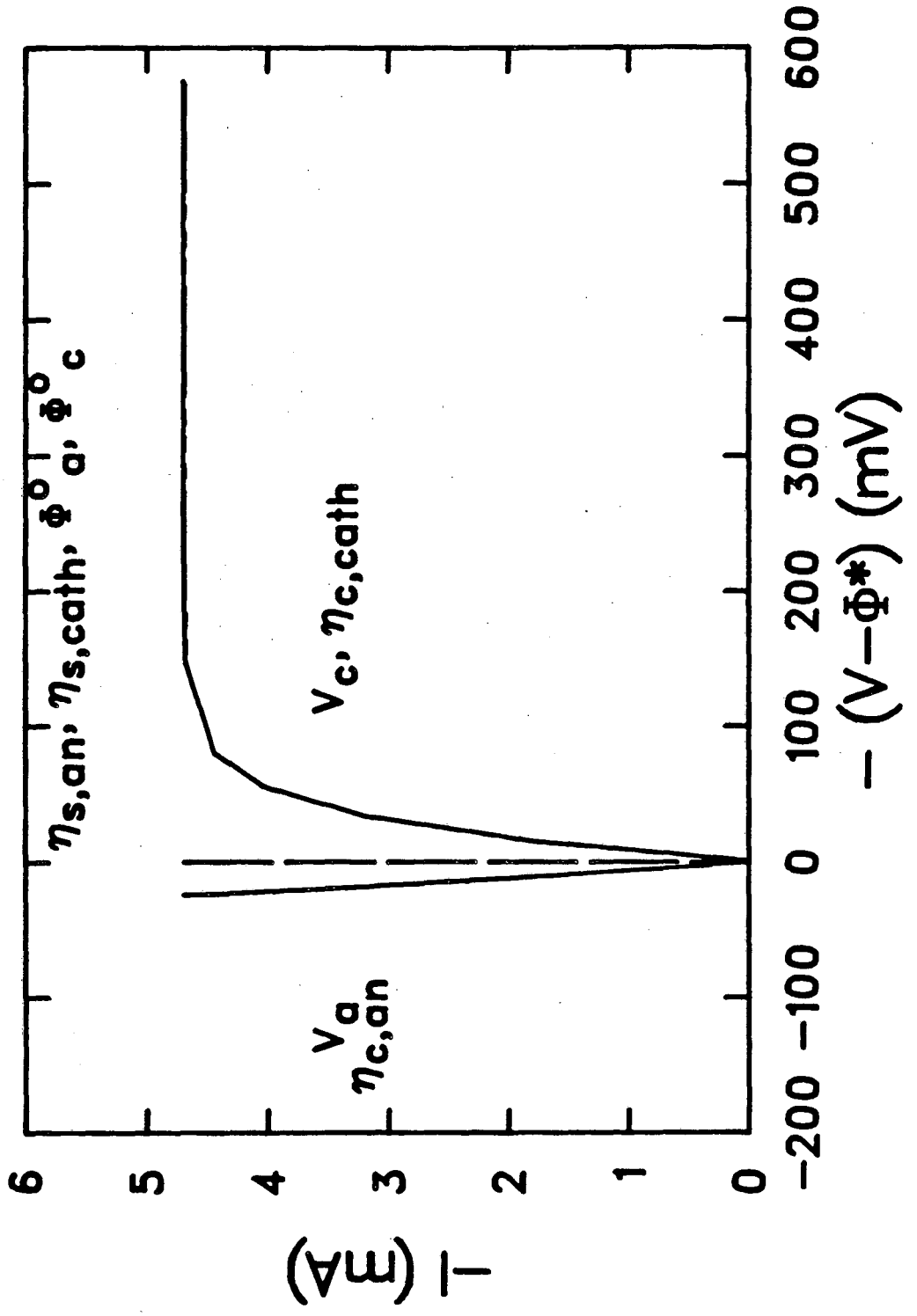


Figure 5-10. Components of the total cell potential for runs 1c-5 through 1c-8 with the thick (0.323 cm) spacer, $Ped_e/L = 809.5$.

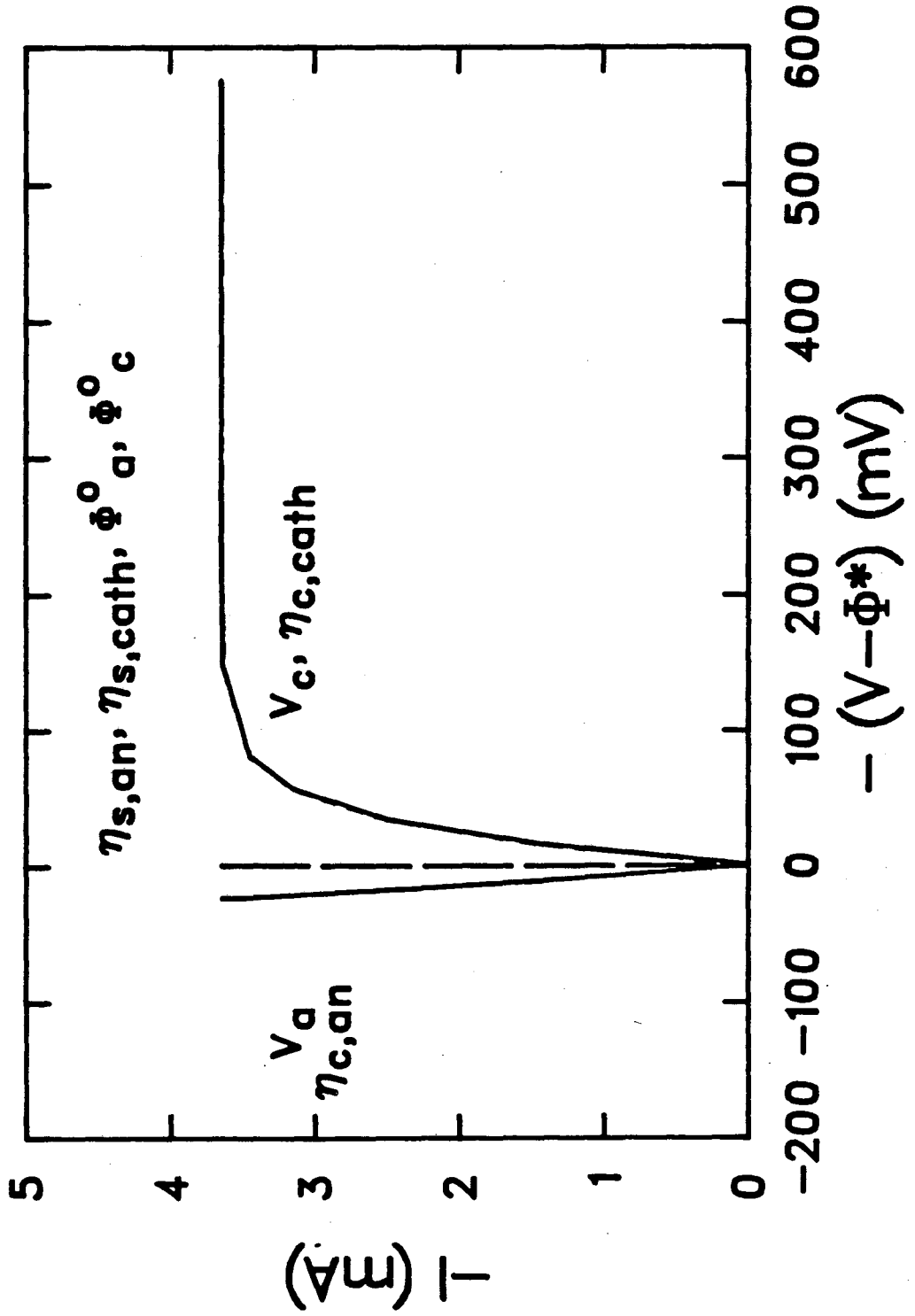
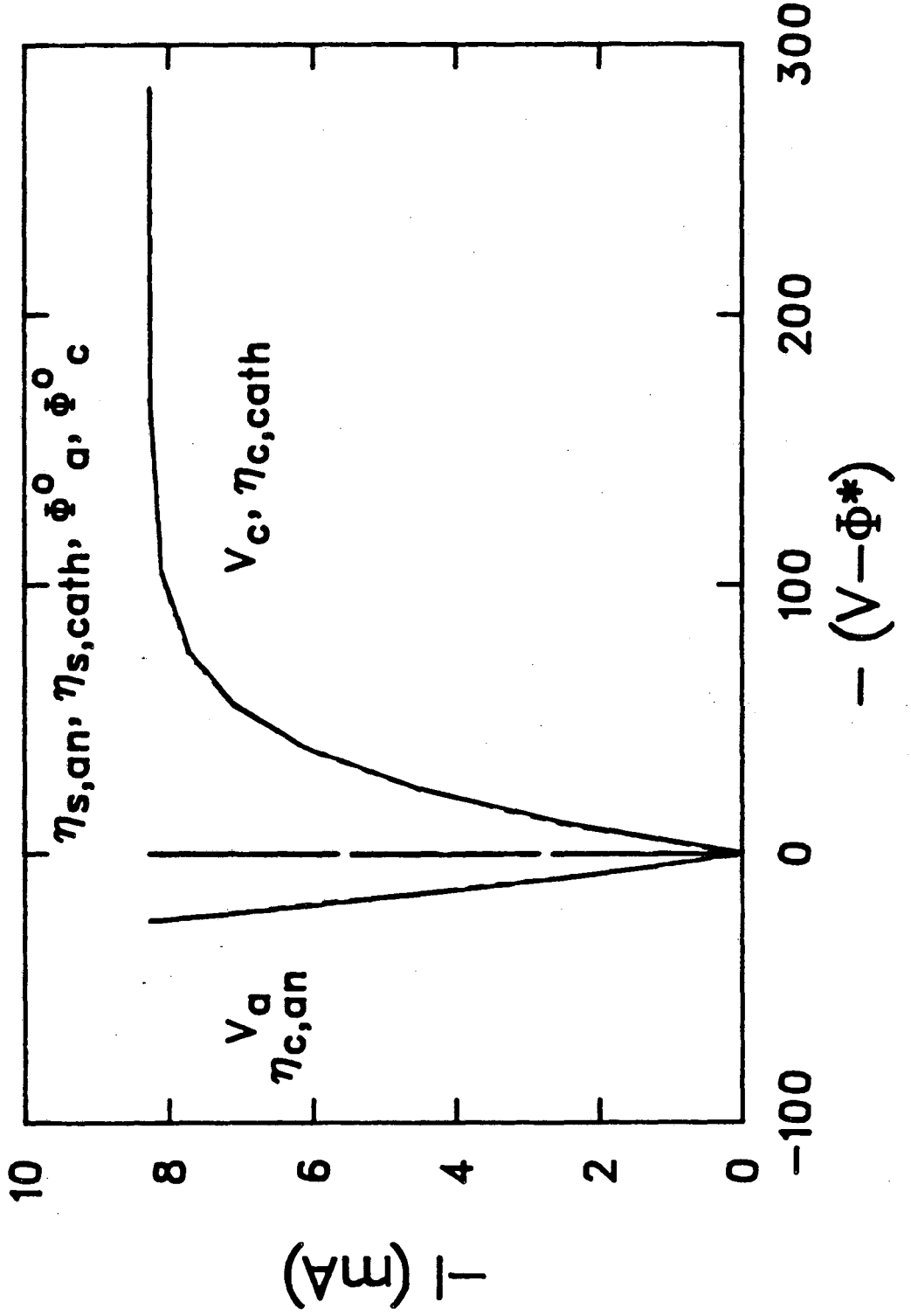


Figure 5-11. Components of the total cell potential for runs 1d-1 through 1d-6 with the thick (0.323 cm) spacer, $Ped_e/L = 388.14$.



XBL 866-2457

Figure 5-12. Components of the total cell potential for run 1f-1 with the thin (0.043 cm) spacer, $Ped_e/L = 3.4784$.

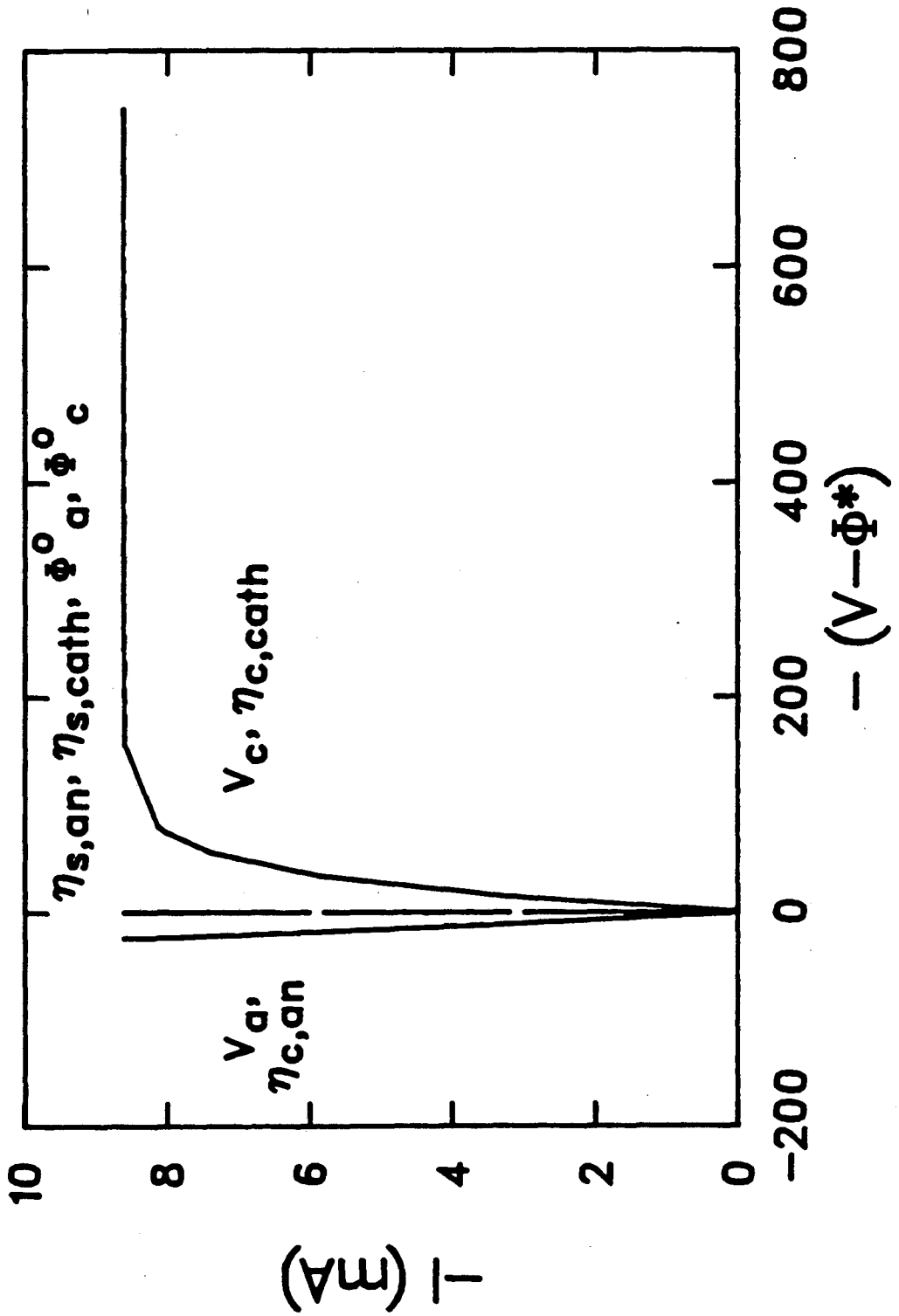


Figure 5-13. Components of the total cell potential for run 1f-2 with the thin (0.043 cm) spacer, $Ped_e/L = 809.5$.

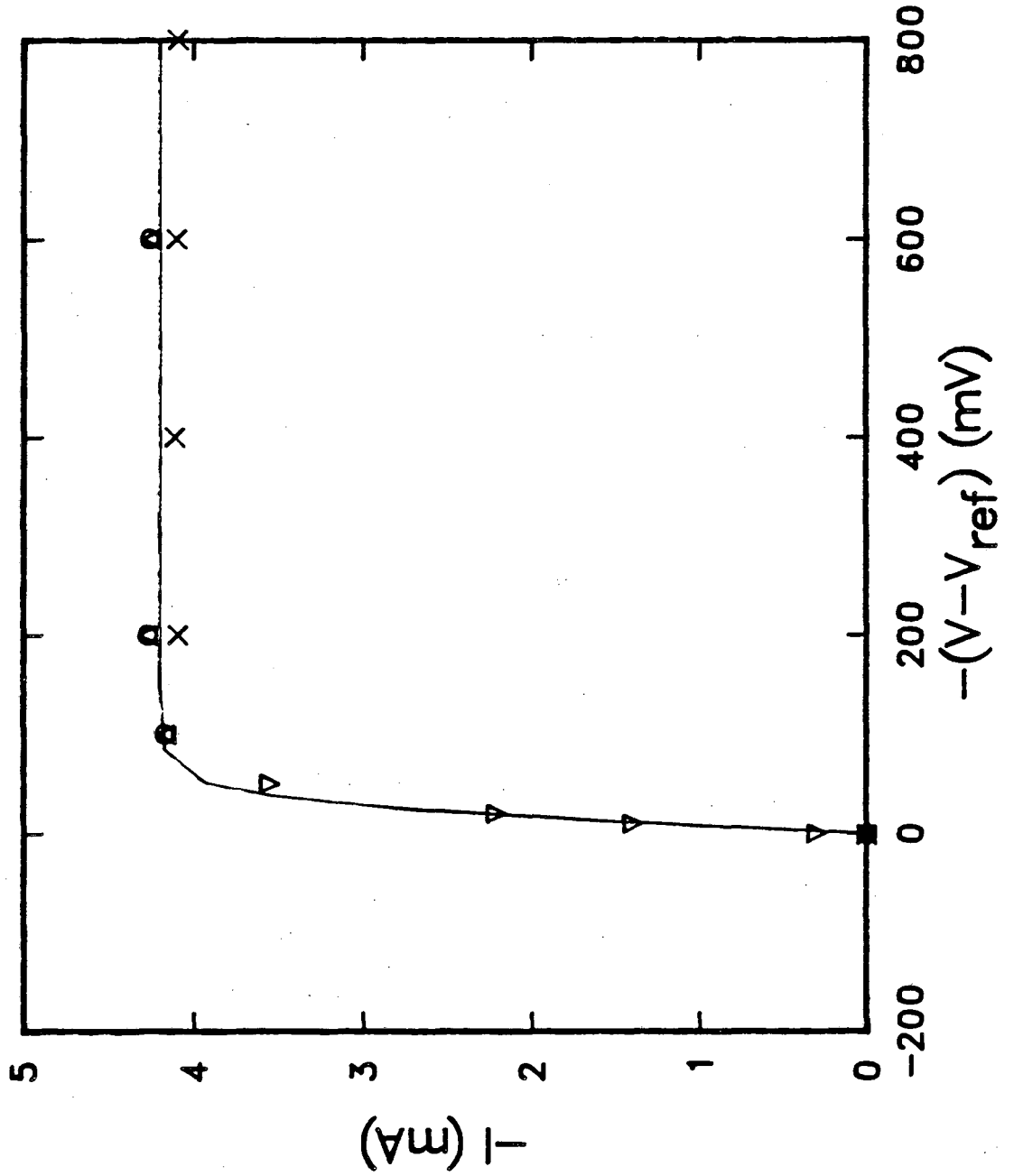


Figure 5-14. Comparison between theory and experiment for run 1c, $Ped_e/L = 586.15$.

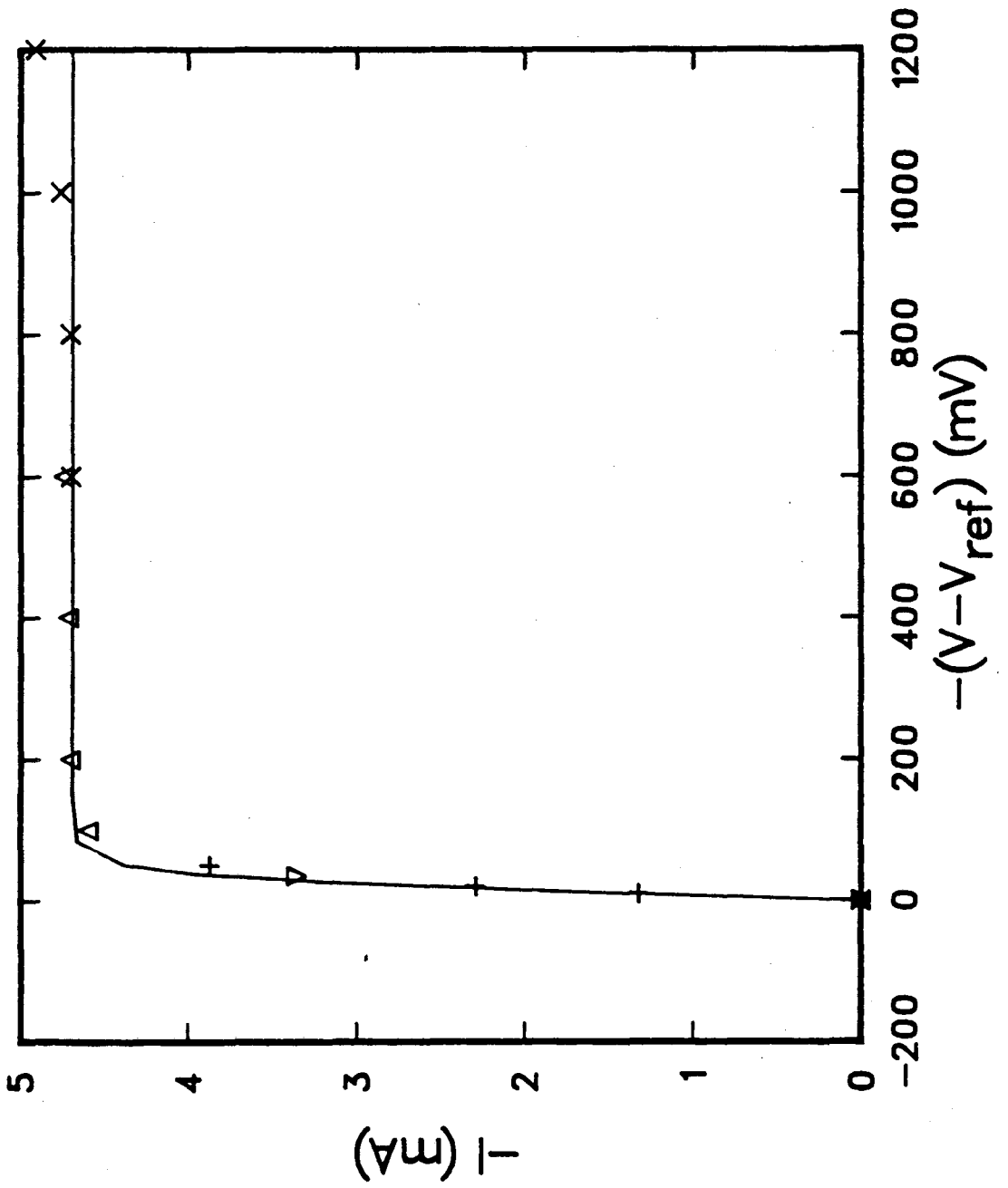


Figure 5-15. Comparison between theory and experiment for run 1c, $Ped_e/L = 809.50$.

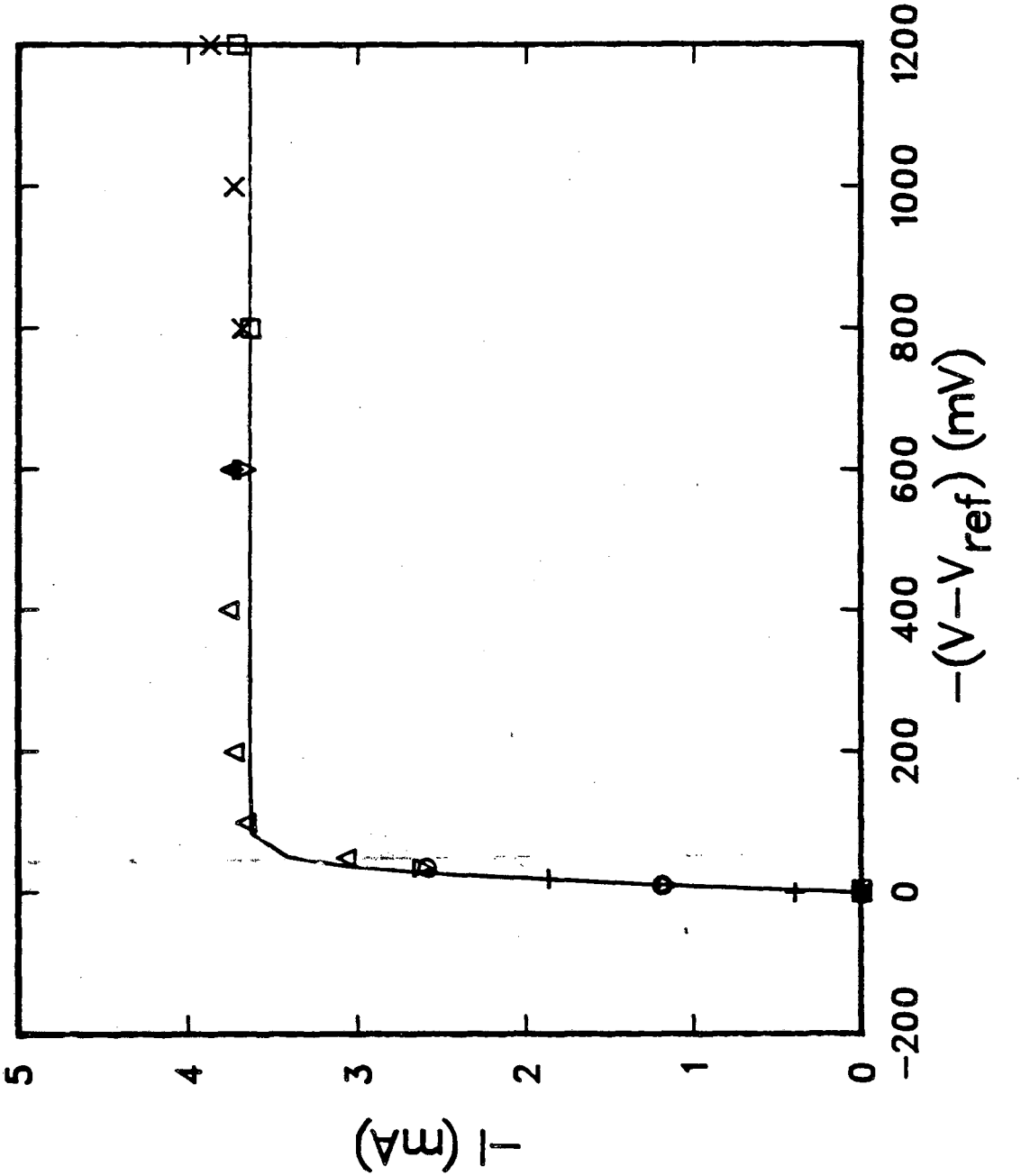


Figure 5-16. Comparison between theory and experiment for run 1d.

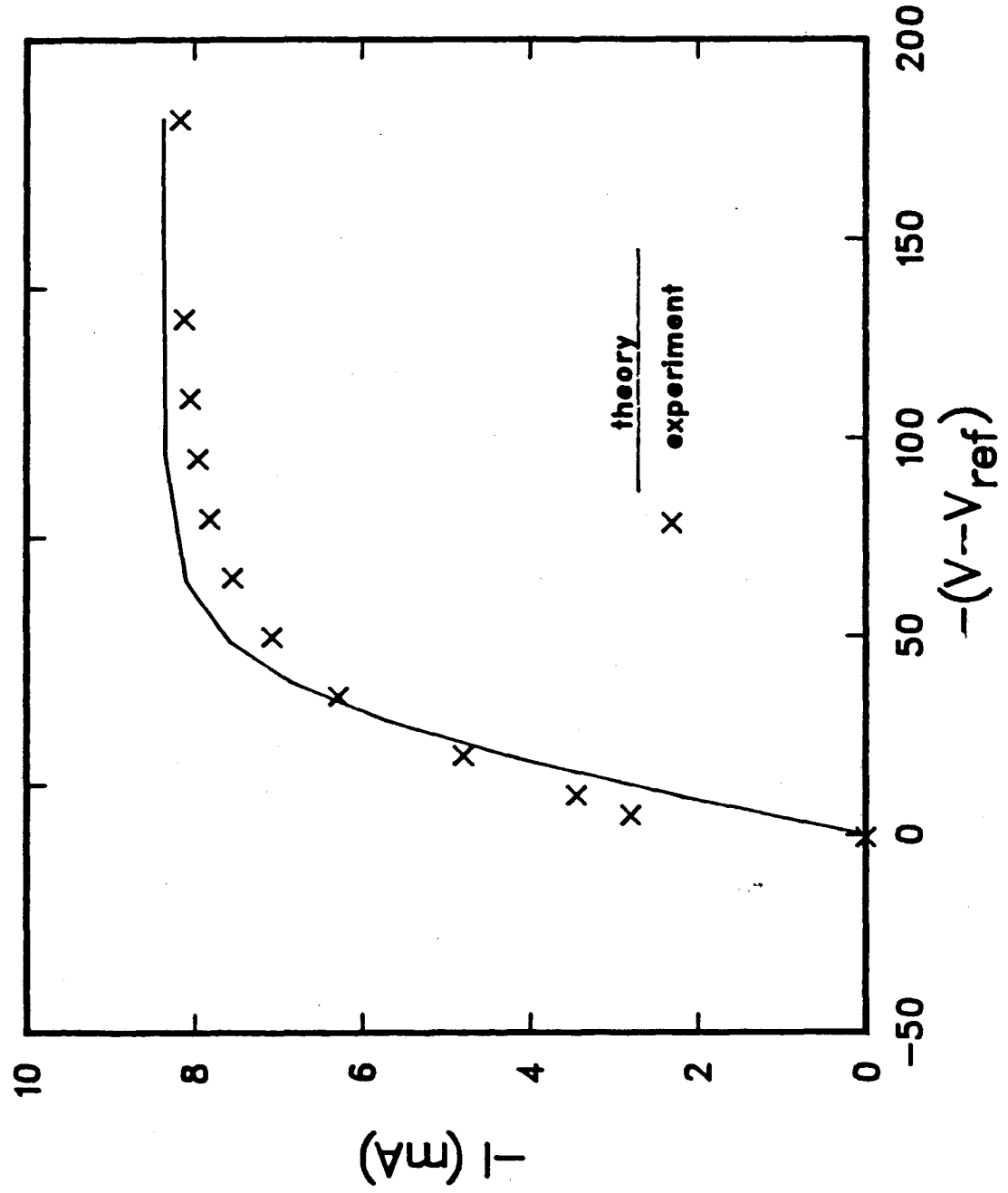


Figure 5-17. Comparison between theory and experiment for run 1f-1.

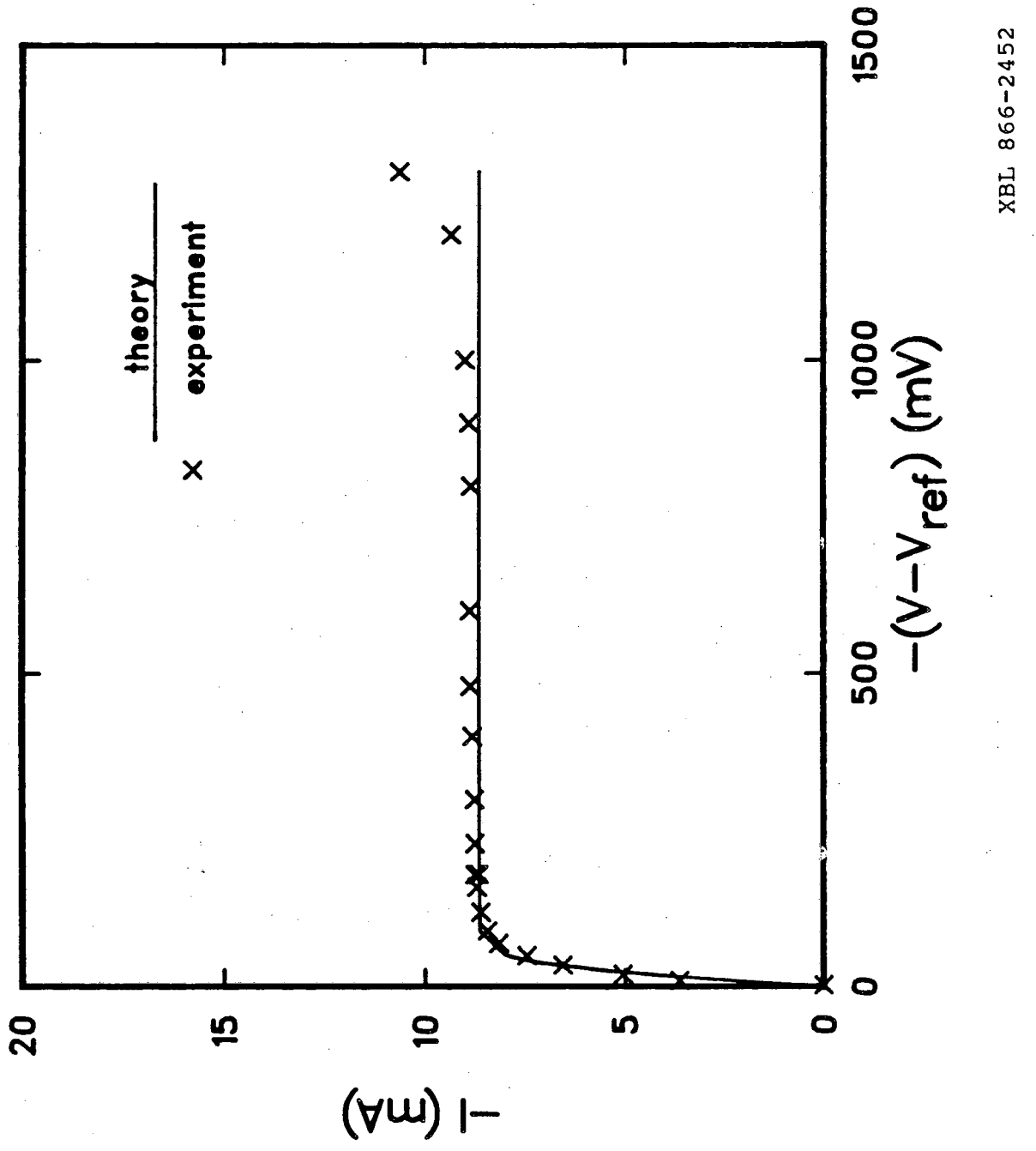


Figure 5-18. Comparison between theory and experiment for run 1f-2.

the cell potential, even below the limiting current. Figures 5-9 through 5-13 demonstrate that this is the case for the $K_3Fe(CN)_6$/ $K_4Fe(CN)_6$ system under investigation. The figures show theoretical predictions of each of the components of the total cell potential — $\Phi_a^o - \Phi^*$, $\Phi_c^o - \Phi^*$, surface overpotential, and concentration overpotential — evaluated at the center of the electrodes. The potentials in these plots are relative to Φ^* (see chapter 1), which is expected to be a rough estimate of V_{ref}, the potential measured by the upstream reference electrode.† The cell potential is the horizontal distance between the line on the far left, V_{an}, and the line on the far right, V_{cath}. The distance between the lines labeled ($\Phi_a^o - \Phi^*$) and ($\Phi_c^o - \Phi^*$) is a rough estimate of the ohmic drop. In each experiment, the ohmic drop is a negligible contribution to the cell potential. The important feature of the plots is that the ohmic drop and surface overpotential are negligible; therefore the cell potential is entirely due to concentration overpotential. (The lines for concentration overpotential are superimposed on the lines for total cell potential.) Eisenberg[79] also showed that, with carefully cleaned electrodes, the surface overpotential is negligible. Thus, adjusting the exchange current density cannot increase the initial slope of the theoretical current-potential curve, leading to the question of whether the discrepancy is an experimental artifact. Since some other investigations show the same discrepancy, this question is difficult to answer. One possible source of experimental error, in our experiments and others, is that the current measurements may not be at steady state. In our experiments, the potential was stepped to each new value, causing a spike in current, and the measurement was taken after a steady current was attained. Eisenberg used a similar method.

It might seem that another possible source of the discrepancy is that V_{ref} is not equal to Φ^*; that is, the experimental and theoretical curves are not plotted with respect

† Φ^* is actually the average between the upstream and downstream potentials, as we shall demonstrate shortly.

to the same potential. The difference between these potentials is small, however. Figure 5-19 shows a plot of the potential distribution for a thin-spacer experiment with non-interacting boundary layers. The upper curve shows the anode potential distribution, and the lower curve shows the cathode potential distribution. The short horizontal lines on the left and right are the upstream and downstream potentials, respectively. Converting into dimensional quantities shows that V_{ref} and Φ^* differ by only 0.45 mV.

Figure 5-19 has several interesting features. The center dashed line, for example, is a rough estimate of the potential along the centerline of the channel; it is calculated from the first term of equation 1-6:

$$\Phi(x) = \Phi^* - \frac{1}{2\pi\kappa_\infty} \int_0^L [i_{cath}(x') + i_{an}(x')] \ln \cosh^2\left(\frac{\pi(x-x')}{2h}\right) dx'. \qquad (5\text{-}2)$$

This quantity evaluated in the limit as $x \rightarrow -\infty$ is the first moment of $(i_a + i_c)$, and it represents the upstream potential. Note that the upstream and downstream potentials are related by

$$\Phi^o(-\infty) - \Phi^* = -(\Phi^o(+\infty) - \Phi^*); \qquad (5\text{-}3)$$

therefore Φ^* is the average of the upstream and downstream potentials. Equation 5-2 shows that if $i_a = i_c$ everywhere, the upstream potential, V_{ref}, equals Φ^*. The difference between V_{ref} and Φ^* therefore gives an indication of how well i_a mirrors i_c.

It is interesting that, for the thin-spacer experiment, one would expect[19] to be able to ignore axial currents because h/L is very small (0.0075353). Figures 5-19 and 5-20 show that this is not the case. The center-line potential distribution (see figure 5-19) shows a large region of significant axial variation. The potential variation in this edge region is caused by what can be regarded as a dipole of $(i_a + i_c)$ at the leading edge (see figure 5-20); at $x = 0$, $(i_a + i_c) << 0$, and immediately downstream of $x = 0$, $(i_a + i_c) > 0$. Note that this potential variation corresponds to having most of the anodic overpotential at the leading edge. Putting more anodic current at the front

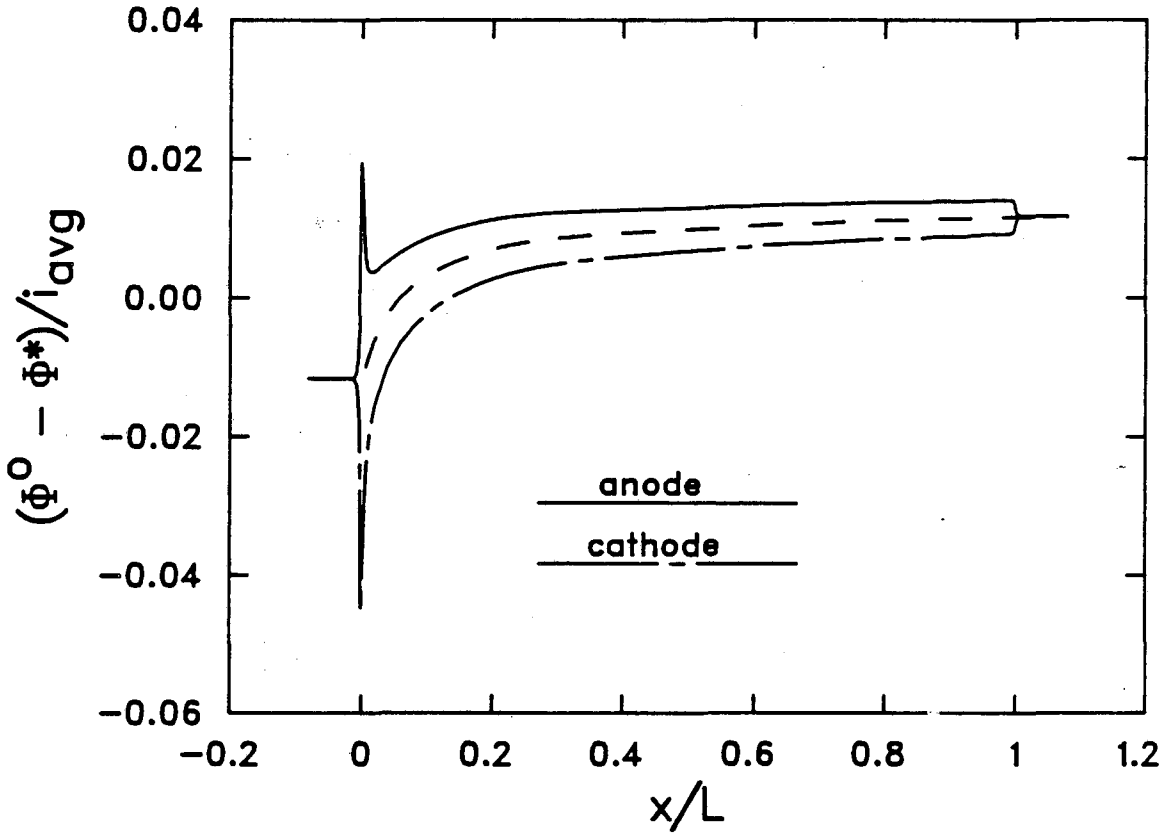


Figure 5-19. Computed potential distribution for run 1e.

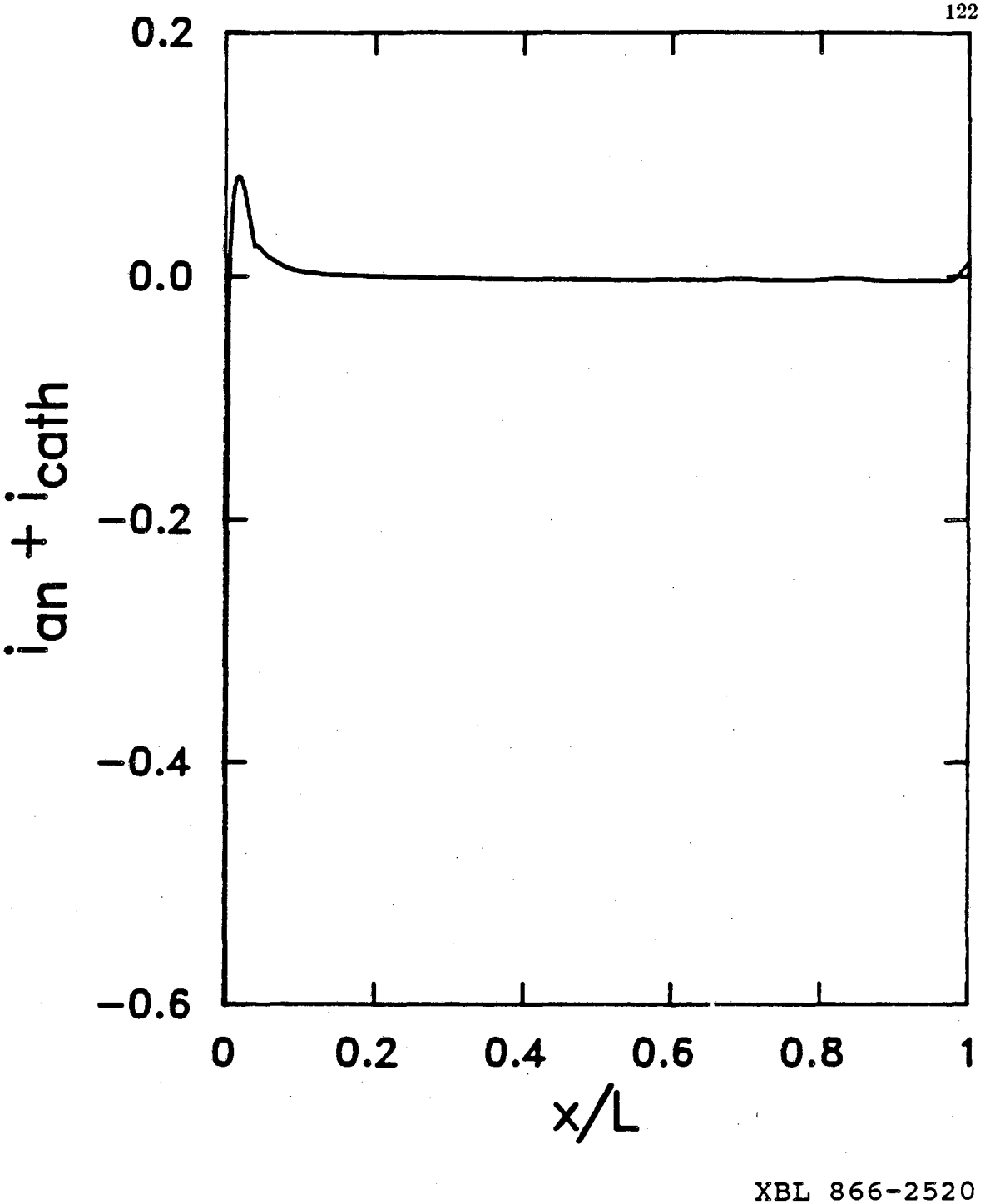


Figure 5-20. $(i_a + i_c)$ for run 1e.

tends to cancel the dipole; therefore the current and potential distributions represent a balance between competing effects. Also note that the region of axial potential variation is quite large. The potential distribution adjacent to the electrodes has superimposed on the center-line variation the effect of the local current density (see the second term of equation 1-6), which results in the spikes at the leading edge. Although the cathode is insensitive to the local potential variations because it is at limiting current, the anode current distribution is strongly dependent on the potential distribution, whether or not the surface overpotential is negligible. If the concentration overpotential dominates, the potential distribution on the anode will affect the current and concentration distribution. If the surface overpotential dominates, the effect of the $\approx$1-mV potential variations may be quite significant because the current depends exponentially on the potential. For the reasons discussed above, it is important to consider axial current and potential variations, even for small h/L. It would have been interesting to conduct the experiments with upstream and downstream reference electrodes to measure the amount of axial variation in the center-line potential.

In summary, the model has given detailed information about the potential distribution that would be difficult to obtain experimentally. The close agreement between the theoretical and experimental polarization curves, without any adjustment of parameters, suggests that the model is applicable over a wide range of Graetz numbers.

Nomenclature for Chapter 5

D	diffusion coefficient, cm^2/s
$D_{channel}$	diffusion coefficient measured with the channel, cm^2/s
D_{disk}	diffusion coefficient measured with the disk, cm^2/s
d_e	equivalent diameter ($2h$ for a channel), cm
D_{Ferri}	diffusion coefficient of the ferricyanide ion, cm^2/s
D_{Ferro}	diffusion coefficient of the ferrocyanide ion, cm^2/s
L	electrode length, cm
Nu	Nusselt number
Re	Reynolds number ($2<v>h/\nu$ for a channel)
Sc	Schmidt number (ν/D)
T	temperature, K (unless noted as °C)
V_{an}	potential of the anode, V

V_{cath}	potential of the cathode, V
V_{ref}	potential of the reference electrode, V

Greek

μ	viscosity, g/cm-s
Φ_a^o	potential in the solution adjacent to the anode (just outside the diffuse double layer), V
Φ_c^o	potential in the solution adjacent to the cathode (just outside the diffuse double layer), V
Φ^*	integration constant in equation 1-5, V

Appendix A. The Lévêque Series

A.1. Introduction

To describe mass transfer to the wall of a channel with a step change in concentration, the Lévêque solution[31] can be used as the first term in a series, which is valid in the mass-transfer entrance region. This approach[30,65] has been applied to the problem of mass transfer to the wall of a tube (the Graetz problem), and the procedure is illustrated for an annulus in problem 1 in chapter 17 of reference 28. Here, the channel problem is treated.

A.2. Mathematical Formulation

In terms of dimensionless variables, the convective diffusion equation for a channel flow cell is

$$(1-\xi^2)\frac{\partial\theta}{\partial\varsigma} = \frac{\partial^2\theta}{\partial\xi^2}\ , \tag{A-1}$$

where, θ is the dimensionless concentration, defined as $\frac{c_i-c_o}{c_b-c_o}$, ς is the dimensionless axial variable, and ξ is the dimensionless transverse variable, with the center of the channel at ξ=0. If the step change is at ξ=+1, the boundary conditions are

$$\theta = 1 \ \text{ at } \varsigma = 0 \tag{A-2}$$

$$\theta = 0 \ \text{ at } \xi = 1 \tag{A-3}$$

$$\theta \longrightarrow 1 \ \text{ as } \xi \longrightarrow -\infty \text{ (outside the diffusion layer).} \tag{A-4}$$

Equation A-1 can be rewritten in terms of the independent variables ς and η, where η is the similarity-transform variable:

$$\eta = (1-\xi)\left(\frac{2}{9\varsigma}\right)^{1/3}\ . \tag{A-5}$$

In terms of ς and η, equation A-1 is

$$\frac{\partial^2\theta}{\partial\eta^2} + 3\eta^2\frac{\partial\theta}{\partial\eta} - 9\varsigma\eta\frac{\partial\theta}{\partial\varsigma} = \frac{3}{2}\eta^3\left(\frac{9\varsigma}{2}\right)^{1/3}\frac{\partial\theta}{\partial\eta}\,, \tag{A-6}$$

with boundary conditions θ=0 at η=0 and $\theta\rightarrow 1$ as $\eta\rightarrow\infty$.

An additional variable substitution, $Z=\left(\frac{9\varsigma}{2}\right)^{1/3}$, transforms equation A-6 to

$$\frac{\partial^2\theta}{\partial\eta^2} + 3\eta^2\frac{\partial\theta}{\partial\eta} - 3\eta Z\frac{\partial\theta}{\partial Z} = \frac{3}{2}\eta^3 Z\frac{\partial\theta}{\partial\eta} - \frac{3}{2}Z^2\eta^2\frac{\partial\theta}{\partial Z}\,. \tag{A-7}$$

If θ, expressed as a power series in Z, is substituted into equation A-7, one can equate the coefficients of the powers of Z to obtain a set of equations. If θ is expressed as

$$\theta = \sum\theta_i Z^i\,, \tag{A-8}$$

then the set of equations is:

$$\theta_0'' + 3\eta^2\theta_0' = 0 \tag{A-9}$$

$$\theta_1'' + 3\eta^2\theta_1' - 3\eta\theta_1 = \frac{3}{2}\eta^3\theta_0' \tag{A-10}$$

$$\theta_2'' + 3\eta^2\theta_2' - 6\eta\theta_2 = -\frac{3}{2}\eta^2\theta_1 + \frac{3}{2}\eta^3\theta_1'\,, \tag{A-11}$$

where the left side of each equation is of the form

$$L_m(\theta_m) = \theta_m'' + 3\eta^2\theta_m' - 3m\eta\theta_m\,. \tag{A-12}$$

Note that equation A-9 is the Lévêque equation; therefore,

$$\theta_0 = \frac{1}{\Gamma(\frac{4}{3})}\int_0^\eta e^{-x^3}dx\,, \tag{A-13}$$

where $\Gamma(\frac{4}{3}) = 0.89298$.[99] Thus, A-10 becomes

$$\theta_1'' + 3\eta^2\theta_1' - 3\eta\theta_1 = \frac{3}{2}\eta^3\frac{1}{\Gamma(\frac{4}{3})}\,e^{-\eta^3}\,. \tag{A-14}$$

This equation can be solved by reduction of order, because a homogeneous solution, $\theta_1 = \eta$, is known. After integration by parts and rearrangement, the solution can be obtained:

$$\theta_1 = \frac{-\eta^2 e^{-\eta^3}}{10\,\Gamma(\frac{4}{3})} - \frac{1}{10\,\Gamma(\frac{4}{3})}\,\eta\int_{\eta}^{\infty} e^{-x^3}dx\ . \tag{A-15}$$

Differentiating this equation and substituting into equation A-11 gives

$$\theta_2'' + 3\eta^2\theta_2' - 6\eta\theta_2 = \frac{9}{20}\eta^7\,\frac{e^{-\eta^3}}{\Gamma(\frac{4}{3})}\ . \tag{A-16}$$

The particular solutions to this equation are of the form $\theta_2 = \eta^p e^{-\eta^3}$, since

$$L_2(\eta^p e^{-\eta^3}) = p(p-1)\eta^{p-2}e^{-\eta^3} - 3(p+4)\eta^{p+1}e^{-\eta^3}\ . \tag{A-17}$$

Because η is raised to the 7th power in equation A-16, the particular solution is

$$\theta_{2,p} = a\eta^6 e^{-\eta^3} + b\eta^3 e^{-\eta^3} + ce^{-\eta^3}\ , \tag{A-18}$$

where

$$a = \frac{-3}{200\,\Gamma(\frac{4}{3})}\ ,\ b = \frac{-3}{140\,\Gamma(\frac{4}{3})}\ ,\ c = \frac{-3}{280\,\Gamma(\frac{4}{3})}\ . \tag{A-19}$$

A homogeneous solution is [30,65]

$$\theta_{2,h} = K\int_0^1 \frac{x^{1/3}}{(1-x)^{2/3}}\,exp(\frac{-\eta^3}{1-x})\,dx\ , \tag{A-20}$$

where the constant K is obtained from the boundary condition $\theta_2(0) = 0$. Combining the particular and homogeneous solutions yields the complete solution:

$$\theta_2 = \frac{-3}{200\,\Gamma(\frac{4}{3})}\,\eta^6 e^{-\eta^3} - \frac{3}{140\,\Gamma(\frac{4}{3})}\,\eta^3 e^{-\eta^3} - \frac{3}{280\,\Gamma(\frac{4}{3})}\,e^{-\eta^3}$$

$$+ \frac{\Gamma(\frac{5}{3})}{280[\Gamma(\frac{4}{3})]^3}\int_0^1 \frac{x^{1/3}}{(1-x)^{2/3}}\exp(\frac{-\eta^3}{1-x})\,dx\ . \tag{A-21}$$

Using the 3-term expansion for θ, one can produce a 3-term Lévêque series for the Nusselt number, where Nu, based on the hydraulic diameter, is related to the derivative of θ:

$$Nu = -4\frac{\partial\theta}{\partial\xi}\Big|_{\xi=1}$$
$$= -4\frac{\partial\theta}{\partial\eta}\Big|_{\eta=0}\frac{\partial\eta}{\partial\xi}\Big|_{\xi=1}$$
$$= 2\left(\frac{2}{9\varsigma}\right)^{1/3}\frac{\partial\theta}{\partial\eta}\Big|_{\eta=0}.$$

The partial derivative, $\frac{\partial\theta}{\partial\eta}$, can be evaluated term by term, but it is somewhat difficult to evaluate $\theta_2'(0)$. The trick is to make the variable substitution,

$$t = \frac{\eta^3}{1-x}\,,\quad dt = \frac{\eta^3}{(1-x)^2}dx\,,$$
$$dt = \frac{\eta^3}{(1-x)^2}dx\,, \tag{A-23}$$

so that $\theta_2'(0)$ becomes

$$\theta_2'(0) = -3K\int_0^\infty t^{-1/3}e^{-t}dt = \frac{-9}{2}\,K\,\Gamma\left(\frac{5}{3}\right), \tag{A-24}$$

where $\Gamma(\frac{5}{3}) = 0.90275$.[99] The resulting 3-term Lévêque series is

$$Nu = \frac{4}{\Gamma(\frac{4}{3})}\left(\frac{2}{9}\right)^{1/3}\varsigma^{-1/3} - \frac{4}{10} - \frac{18(\frac{9}{2})^{1/3}}{280}\,\frac{[\Gamma(\frac{5}{3})]^2}{[\Gamma(\frac{4}{3})]^3}\,\varsigma^{1/3}$$

$$= 2.7131949\varsigma^{-1/3} - 0.4 - 0.12146690\varsigma^{1/3}\,,$$

or

$$= 1.8488\left(\frac{ReScd_e}{L}\right)^{1/3} - 0.4 - 0.2005\left(\frac{L}{\mathrm{Re}Scd_e}\right)^{1/3}. \tag{A-25}$$

The first two terms of this equation agree with the two terms given in problem 17-1 of reference 28.

Appendix B. Numerical Solution of the Asymmetric Graetz Problem

The finite-difference method developed by Newman, for solving coupled, ordinary differential equations, has been described elsewhere.[60,57] Here, the method is applied to the eigenvalue problem discussed in sections 2.4 and 2.5.

Equations 2-11 and 2-12 may be written in the form

$$\frac{d^2y_1}{dx^2} + y_2(1-x^2)y_1 = 0 \tag{B-1}$$

$$\frac{dy_2}{dx} = 0 \,. \tag{B-2}$$

Equation B-1 may be linearized by setting

$$y_1 = y_1^o + \Delta y_1 \tag{B-3}$$

$$y_2 = y_2^o + \Delta y_2 \,. \tag{B-4}$$

Here, y_1 is the exact solution, and y_1^o is an approximate solution or guess. Substituting B-3 and B-4 into B-1 and neglecting terms of order $(\Delta y)^2$ gives

$$\frac{d^2y_1^o}{dx^2} + \frac{d^2\Delta y_1}{dx^2} + y_2^o y_1^o(1-x^2) + y_2^o(1-x^2)\Delta y_1 + y_1^o(1-x^2)\Delta y_2 = 0 \,, \tag{B-5}$$

or

$$-\left[\frac{d^2\Delta y_1}{dx^2} + y_2^o(1-x^2)\Delta y_1 + y_1^o(1-x^2)\Delta y_2\right] = \frac{d^2y_1^o}{dx^2} + y_2^o y_1^o(1-x^2) \,. \tag{B-6}$$

Similarly, equation B-2 becomes

$$\frac{d\Delta y_2}{dx} = -\frac{dy_2^o}{dx} \,. \tag{B-7}$$

Both equations B-6 and B-7 are of the form

$$\sum_k a_{ik}(x)\frac{d^2y_k}{dx^2} + b_{ik}(x)\frac{dy_k}{dx} + c_{ik}(x)y_k = g_i(x) \,, \tag{B-8}$$

where

$$a_{11} = -1 \qquad\qquad a_{12} = 0$$

$$a_{21} = 0 \qquad a_{22} = 0$$

$$b_{11} = 0 \qquad b_{12} = 0$$

$$b_{21} = 0 \qquad b_{22} = 1$$

$$c_{11} = -y_2^o(1-x^2) \qquad c_{12} = -y_1^o(1-x^2)$$

$$c_{21} = 0 \qquad c_{22} = 0$$

$$g_1 = \frac{d^2y_1^o}{dx^2} + y_2^o y_1^o(1-x^2)$$

$$g_2 = -\frac{dy_2^o}{dx} \,. \tag{B-9}$$

The differential equations represented by B-8 can be written in finite-difference form:

$$\sum_k A_{ik}(j)Y_k(j-1) + B_{ik}(j)Y_k(j) + D_{ik}(j)Y_k(j+1) = G_i(j) \,. \tag{B-10}$$

Note that this is a banded-matrix equation for the vector of Y_i's. If central differences are used, then

$$A_{ik}(j) = a_{ik} - \frac{h}{2}\, b_{ik}$$

$$B_{ik}(j) = -2a_{ik} + h^2 c_{ik}$$

$$D_{ik}(j) = a_{ik} + \frac{h}{2}\, b_{ik}$$

$$G_i(j) = h^2 g_i \,. \tag{B-11}$$

These equations can be used to put equation B-6 in central-difference form. For equation B-7, however, the forward-difference form is accurate to order h^2 because y_2 is a constant. This eliminates the need for image points.

Because the eigenfunctions are all even or odd, it is necessary only to solve the problem in half of the domain, that is, from x=0 to x=L. For the even eigenfunctions, the boundary condition is y_1'=0 at x=0, and for the odd eigenfunctions, y_1=0 at x=0,

where x=0 corresponds to the center of the channel. For all eigenfunctions, y_1=0 at x=1, and the normalization condition, y_1'=1 at x=1, is applied.

To obtain the coefficients A_{ij},B_{ij},D_{ij}, and G_i at the endpoints j=1 and $j=j_m$, the boundary conditions are needed. For simplicity, we treat the even and odd eigenfunctions separately.

For the even eigenfunctions at j=1, y_1' is zero. To avoid the need for an image point, the differential equation B-6 is written at x=0:

$$-\frac{d^2\Delta y_1}{dx^2}(1) - y_2^o(1)\Delta y_1(1) - y_1^o(1)\Delta y_2(1) = \frac{d^2 y_1^o}{dx^2}(1) + y_2^o(1)y_1^o(1) \; . \tag{B-12}$$

The second derivative is written in central-difference form to ensure accuracy to order h^2, but the boundary condition y_1'=0 can be used to eliminate the image point at j=0 because $y_1(2)$=$y_1(0)$. Therefore the second derivative is

$$\frac{d^2 y_1}{dx^2}(1) = \frac{2}{h^2}[y_1(2) - y_1(1)] \; . \tag{B-13}$$

Equation B-7 remains the same, because there is no boundary condition on the eigenvalue, y_2.

For the odd eigenfunctions at j=1, the boundary condition is y_1=0. Therefore,

$$-\Delta y_1(1) = y_1^o(1) \; . \tag{B-14}$$

Again, equation B-7 remains the same.

For $j=j_m$, the boundary conditions are the same for both the even and odd eigenfunctions: y_1'=1 and y_1=0. From the boundary condition y_1=0,

$$-\Delta y_1(j_m) = y_1^o(j_m) \; . \tag{B-15}$$

For the boundary condition y_1'=1, the backward-difference form is accurate to order h^2, because the second derivative is zero at x=1:

$$\left[\frac{d^2y_1}{dx^2} + y_2(1-x^2)y_1\right]_{x=1} = \frac{d^2y_1}{dx^2}\Big|_{x=1} = 0 \,. \tag{B-16}$$

Therefore,

$$\frac{1}{h}(\Delta y_1(j_m) - \Delta y_1(j_m-1)) = -\frac{1}{h}(y_1^o(j_m) - y_1^o(j_m) - y_1^o(j_m-1)) + 1 \,. \tag{B-17}$$

Now that all the equations and boundary conditions have been written in the form of equation B-10, the coefficients A_{ij}, B_{ij}, D_{ij}, and G_i can be calculated. Subroutines BAND and MATINV (see appendix C) can then be used to solve for the Δy_i's, which are used to obtain new guesses y_1^o and y_2^o. This iteration over the nonlinearities is continued until the desired degree of convergence is reached.

Appendix C. Program EIGEN

```
c     PROGRAM GRAETZ(INPUT,OUTPUT)
      DIMENSION A(2,2),B(2,2),C(2,801),D(2,5),G(2),X(2,2),Y(2,2),XI(801)
     1,XI2(801),COLD(1,801),CSA(20,801),conc(6),concb(6)
     1,am(20),al(20)
      COMMON A,B,C,D,G,X,Y,N,NJ
c
c       This program solves the eigenvalue problem for the asymmetric Graetz
c       problem in a channel. (See V. Edwards and J. Newman, Int. J. Heat
c       Mass Trans., Vol. 28 (1985), 503-505.
c
  101 FORMAT (2I4)
  105 FORMAT (/)
  108 format (5h zeta,7x,12h cxi(zeta,1),3x,
     1  13h cxi(zeta,-1),3x,5h test,9x,7hleveque)
  109 format(21h no convergence after,i5,11h iterations)
  111 format(5(1x,e11.4,2x))
  112 format(3h l=,i5,3hnj=,i5)
  113 format(15h concentrations,/,5h zeta)
  114 format(1x,e11.4,/,6(1x,e11.4,2x),/,6(1x,e11.4,2x))
c
c     this program modifies sin and lambda initial guesses
c
      lmax = 5
c
c       L is the eigenfunction index
c       NJ is the number of mesh points
c       The program runs multiple eigenfunctions.  Set NJ=0 to stop.
c
    1 READ 101, L,NJ
      print 112,l,nj
      IF(NJ.EQ.0) STOP
c
c       N is the number of equations
c       H is the step size
c
      N=2
      H=1.0/(NJ-1)
      PRINT 105
      MOD=1
      IF(L.EQ.2*(L/2)) MOD=0
      pi = 3.141592654
c
c       Modify the slope of sqrt(lambda) vs. L to match the 7th eigenvalue
c       (to provide a better initial guess for the higher eigenvalues)
c
      cl=pi/2.0 + (15.65543836-pi/2.0)/7.0 * (float(l)-1.0)
      DO 2 J=1,NJ
```

```
      XI(J)=H*(J-1)
c
c       xi2 is the weighting function for calculating the coefficients
c
      XI2(J)=1.0-XI(J)**2
      C(1,J)=SIN(c1*FLOAT(NJ-J)*H)/c1
c
c       C(1,J) is the eigenfunction
c       al(1) = C(2,J) is the eigenvalue (lambda)
c
      al(1) = c1*c1
      if((2*(1/2)).ne.1) go to 22
      if(2*(1/4).eq.(1/2)) c(1,j) = -c(1,j)
      go to 23
   22 if(2*(1-1)/4.eq.(1-1)/2) c(1,j) = -c(1,j)
   23 continue
    2 C(2,J)= AL(1)
      JCOUNT= 0
    3 JCOUNT= JCOUNT + 1
      J=0
c
c       No image points because the boundary conditions are incorporated into
c       the equations at j=1 and j=nj
c
      DO 4 I=1,N
      DO 4 K=1,N
      Y(I,K)= 0.0
    4 X(I,K)= 0.0
      ALO= AL(1)
      COLD(1,NJ)=C(1,NJ)
    5 J=J+1
c
c       initialization
c
      DO 6 I=1,N
      G(I)= 0.0
      DO 6 K=1,N
      A(I,K)=0.0
      B(I,K)=0.0
    6 D(I,K)=0.0
      IF(J.EQ.NJ) GO TO 8
c
c       equation for eigenvalue
c
      G(2)=C(2,J+1)-C(2,J)
      COLD(1,J)=C(1,J)
      B(2,2)=1.0
      D(2,2)=-1.0
```

```
      IF(J.GT.1)GO TO 7
      IF(MOD.EQ.0)GOTO 11
c
c       The following is for j=1, odd L
c       BC is y'=0 at xi=0
c       Second equation is written in forward-difference form
c
      G(1)=2.0*(C(1,J+1)-C(1,J))/H/H+C(2,J)*C(1,J)
      D(1,1)=-2./H/H
      B(1,1)=-D(1,1)-C(2,J)
      B(1,2)=-C(1,J)
      GO TO 12
c
c       The following is for j=1, even L
c       BC is y=0 at xi=0
c
   11 G(1)=C(1,J)
      B(1,1)=-1.0
   12 CALL BAND(J)
      GO TO 5
c
c       The following is for j=2 to (nj-1)
c
    7 G(1)=(C(1,J+1)+C(1,J-1)-2.0*C(1,J))/H/H+XI2(J)*C(2,J)*C(1,J)
      A(1,1)=-1.0/H/H
      D(1,1)=A(1,1)
      B(1,2)=-XI2(J)*C(1,J)
      B(1,1)=-2.0*A(1,1)-XI2(J)*C(2,J)
      CALL BAND(J)
      GO TO 5
c
c       The following is for j=nj
c
    8 G(1)=C(1,J)
      B(1,1)=-1.0
      G(2)=2.0*(C(1,J-1)-C(1,J))/H/H+2./H
      B(2,1)=2.0/H/H
      A(2,1)=-B(2,1)
      CALL BAND(J)
c
c       Update the guesses
c
      AL(1)=C(2,1)+AL(1)
      DO 13 J=1,NJ
      C(1,J)=C(1,J)+COLD(1,J)
   13 C(2,J)=AL(1)
      IF(JCOUNT.GT.10)GO TO 18
c
```

```
c       If not converged, next jcount
c
      IF(ABS(AL(1)-ALO).GT.ABS(AL(1))*1.0E-10) GO TO 3
      do 19 j=1,nj
      if(abs(c(1,j)-cold(1,j)).gt.abs(c(1,j))*1.0e-5) go to 3
   19 continue
c
c       save Y(1)
c
      CSA(L,1)=C(1,1)
      SUM1=C(1,1)*H/3.
      SUM2=SUM1*C(1,1)
      DO 10 JJ=3,NJ,2
      J=JJ-1
c
c       Use Simpson's rule integration for coefficients
c
c       even j:
c
      CSA(L,J)=C(1,J)
      ADD=XI2(J)*C(1,J)*4.0*H/3.0
      SUM2=SUM2+ADD*C(1,J)
      IF(MOD.EQ.0) ADD=ADD*XI(J)
      SUM1=SUM1+ADD
      J=J+1
c
c       odd j:
c
      ADD=XI2(J)*C(1,J)*2.0*H/3.0
      SUM2=SUM2+ADD*C(1,J)
      IF(MOD.EQ.0) ADD=ADD*XI(J)
      CSA(L,J)=C(1,J)
   10 SUM1=SUM1+ADD
c
c       The AM are the coefficients
c
      AM(1)=-SUM1/(2.0*SUM2)
      IF(MOD.EQ.0) AM(1)=-AM(1)
      if(l.ne.lmax) go to 1
c      print 108
      zeta = 0.0
      dz = 0.001
      do 26 iz=1,31
      if(iz.ge.11) dz=0.01
      if(iz.ge.20) dz=0.1
      if(iz.ge.29) dz=1.0
c     print concs at a few xi's:
      print 113
```

```
      do 29 i=1,6
      concb(i) = 0.0
   29 conc(i) = 0.0
      istep = (nj-1)/5
      do 32 j=1,nj,istep
      iy = 1 + (j-1)/20
      do 30 i=1,lmax
      pml = 1
      if(i.eq.2*(i/2)) pml= -1
      concb(iy) = concb(iy)+am(i)*exp(-al(i)*zeta)*csa(i,j)
     1  *pml
   30 conc(iy) = conc(iy) + am(i)*exp(-al(i)*zeta)*csa(i,j)
      concb(iy) = concb(iy) + 0.5*(1.0 + xi(j))
   32 conc(iy) = conc(iy) + 0.5*(1.0 - xi(j))
      print 114,zeta,(conc(i),i=1,6),(concb(i),i=1,6)
c     sum1,sum2 are fluxes:
c      sum1 = 0.0
c      sum2 = 0.0
c      do 25 i=1,lmax
c      ali = al(i)
c      sum1 = sum1 + am(i)*exp(-ali*zeta)
c   25 sum2 = sum2 - abs(am(i))*exp(-ali*zeta)
c      sum1 = 1.0 - 2.0*sum1
c      sum2 = 1.0 - 2.0*sum2
   26 zeta = zeta + dz
      go to 21
   18 print 109,jcount
   21 stop
      END
      SUBROUTINE BAND(J)
      COMMON    A(2,2),B(2,2),C(2,801),D(2, 5),G(2),X(2,2),Y(2,2),N,NJ
      DIMENSION E(2,3,801)
c
c       This program solves coupled ODEs by casting them into finite
c       difference form and solving the resulting banded matrix
c
  101 FORMAT (15H DETERM=0 AT J=,I4)
      IF (J-2)  1,6,8
    1 NP1= N + 1
      DO 2 I=1,N
      D(I,2*N+1)= G(I)
      DO 2 L=1,N
      LPN = L + N
    2 D(I,LPN)= X(I,L)
      CALL MATINV(N,2*N+1,DETERM)
      IF (DETERM)  4,3,4
    3 PRINT 101, J
    4 DO 5 K=1,N
```

```
      E(K,NP1,1)= D(K,2*N+1)
      DO 5 L=1,N
      E(K,L,1)= - D(K,L)
      LPN= L + N
    5 X(K,L)= - D(K,LPN)
      RETURN
    6 DO 7 I=1,N
      DO 7 K=1,N
      DO 7 L=1,N
    7 D(I,K)= D(I,K) + A(I,L)*X(L,K)
    8 IF (J-NJ)  11,9,9
    9 DO 10 I=1,N
      DO 10 L=1,N
      G(I)= G(I) - Y(I,L)*E(L,NP1,J-2)
      DO 10 M=1,N
   10 A(I,L)= A(I,L) + Y(I,M)*E(M,L,J-2)
   11 DO 12 I=1,N
      D(I,NP1)= - G(I)
      DO 12 L=1,N
      D(I,NP1)= D(I,NP1) + A(I,L)*E(L,NP1,J-1)
      DO 12 K=1,N
   12 B(I,K)= B(I,K) + A(I,L)*E(L,K,J-1)
      CALL MATINV(N,NP1,DETERM)
      IF (DETERM)  14,13,14
   13 PRINT 101, J
   14 DO 15 K=1,N
      DO 15 M=1,NP1
   15 E(K,M,J)= - D(K,M)
      IF (J-NJ)  20,16,16
   16 DO 17 K=1,N
   17 C(K,J)= E(K,NP1,J)
      DO 18 JJ=2,NJ
      M= NJ - JJ + 1
      DO 18 K=1,N
      C(K,M)= E(K,NP1,M)
      DO 18 L=1,N
   18 C(K,M)= C(K,M) + E(K,L,M)*C(L,M+1)
      DO 19 L=1,N
      DO 19 K=1,N
   19 C(K,1)= C(K,1) + X(K,L)*C(L,3)
   20 RETURN
      END
      SUBROUTINE MATINV(N,M,DETERM)
      DIMENSION A(2,2),B(2,2),C(2,801),D(2,5),ID(2)
      COMMON A,B,C,D
c
c       This program inverts matrix B by pivoting.  The D matrix is n x (2n+1)
c       for use by subroutine BAND
```

```
c
      DETERM=1.0
      DO 1 I=1,N
    1 ID(I)=0
      DO 18 NN=1,N
      BMAX=1.1
      DO 6 I=1,N
      IF(ID(I).NE.0) GOTO 6
      BNEXT=0.0
      BTRY=0.0
      DO 5 J=1,N
      IF(ID(J).NE.0) GOTO 5
      IF(ABS(B(I,J)).LE.BNEXT) GOTO 5
      BNEXT=ABS(B(I,J))
      IF(BNEXT.LE.BTRY) GOTO 5
      BNEXT=BTRY
      BTRY=ABS(B(I,J))
      JC=J
    5 CONTINUE
      IF(BNEXT.GE.BMAX*BTRY) GOTO 6
      BMAX=BNEXT/BTRY
      IROW=I
      JCOL=JC
    6 CONTINUE
      IF(ID(JC).EQ.0) GOTO 8
      DETERM=0.0
      RETURN
    8 ID(JCOL)=1
      IF(JCOL.EQ.IROW) GOTO 12
      DO 10 J=1,N
      SAVE=B(IROW,J)
      B(IROW,J)=B(JCOL,J)
   10 B(JCOL,J)=SAVE
      DO 11 K=1,M
      SAVE=D(IROW,K)
      D(IROW,K)=D(JCOL,K)
   11 D(JCOL,K)=SAVE
   12 F=1.0/B(JCOL,JCOL)
      DO 13 J=1,N
   13 B(JCOL,J)=B(JCOL,J)*F
      DO 14 K=1,M
   14 D(JCOL,K)=D(JCOL,K)*F
      DO 18 I=1,N
      IF(I.EQ.JCOL) GO TO 18
      F=B(I,JCOL)
      DO 16 J=1,N
   16 B(I,J)=B(I,J)-F*B(JCOL,J)
      DO 17 K=1,M
```

```
  17   D(I,K)=D(I,K)-F*D(JCOL,K)
  18   CONTINUE
       RETURN
       END
```

Appendix D. Iterative Methods

D.1. Introduction

To solve for the concentration, current, and potential distributions in an electrochemical system, a set of coupled differential equations must be solved. Although the differential equations can be linear, as in the channel problem with negligible migration, the boundary conditions are often nonlinear because the kinetics depend exponentially on the potential driving force, which varies across the electrode surfaces. Thus, a set of nonlinear equations must be solved, and iteration is required.

The iteration can be viewed as a loop because variables are passed through a set of equations until new guesses for the same variables emerge. If the process is repeated with these new variables, the iteration is called successive substitution or Picard's method. Usually, however, these new variables are operated on by an iterating function, which, in the most general case, uses information from previous iterations to produce new guesses. Thus, one can think of "cutting" the loop to update the variables. In general, it is best to cut the loop where there are the fewest variables, to simplify and speed the updating procedure.

The iterating function for a vector equation $\mathbf{f}(\mathbf{x}) = 0$ may be written

$$\mathbf{x}_{i+1} = \phi(\mathbf{x},\mathbf{f}), \tag{D-1}$$

where $\mathbf{x}_{i+1}$ is the $(i+1)$st approximant to $\mathbf{x}$. ϕ may depend on the previous approximants to $\mathbf{x}$ and the values of $\mathbf{f}$ and its derivatives. Here, we shall discuss some special cases — the successive substitution method, the Newton-Raphson method, and the secant method (see, for example, references 61, 62, 100, and 101).

D.2. Comparison of Iterative Methods

The successive substitution method applied to $\mathbf{x} = \mathbf{g}(\mathbf{x})$ is

$$\mathbf{x}_{i+1} = \mathbf{g}(\mathbf{x}_i). \tag{D-2}$$

A variation on this method is successive substitution with damping:

$$\mathbf{x}_{i+1} = \mathbf{x}_i + k(\mathbf{g}(\mathbf{x}_i) - \mathbf{x}_i). \tag{D-3}$$

The successive substitution method is easy to program, but its convergence is slow (first order), and sometimes it will not converge at all.[61] The speed of convergence can be increased by using a second-order method such as the Newton-Raphson method

$$\mathbf{x}_{i+1} = \mathbf{x}_i - (\mathbf{df}(\mathbf{x}_i))^{-1}\mathbf{f}(\mathbf{x}_i), \tag{D-4}$$

where $\mathbf{df}(\mathbf{x}_i)$ is the Jacobian of $\mathbf{f}$. Second-order convergence means that the error at a given iteration is proportional to the square of the error at the previous iteration,

$$e_{i+1} = C_2\, e_i^2 \tag{D-5}$$

and that the number of significant figures, n_i, doubles with each iteration. With a first-order method, on the other hand, the number of significant figures increases linearly:[62]

$$n_i = -i \log C_1 + n_o, \tag{D-6}$$

where C_p is the asymptotic error constant defined, for pth order convergence, as

$$e_{i+1} = C_p\, {e_i}^p. \tag{D-7}$$

If C_p is close to unity, the convergence is slow.

Another way to examine the convergence order is to plot the errors so that the curve becomes linear as quadratic convergence is approached. Typically, a plot of this type will show that, in the early iterations, the truncated Taylor-series approximation of the function is inaccurate; then as the solution is approached, the convergence becomes quadratic within the limit of the machine accuracy.

Figure D-1 is a plot for a Newton-Raphson iteration of $\ln \left| \ln \left| \frac{e_{i+1}}{e_i} \right| \right|$ *vs.* itera-

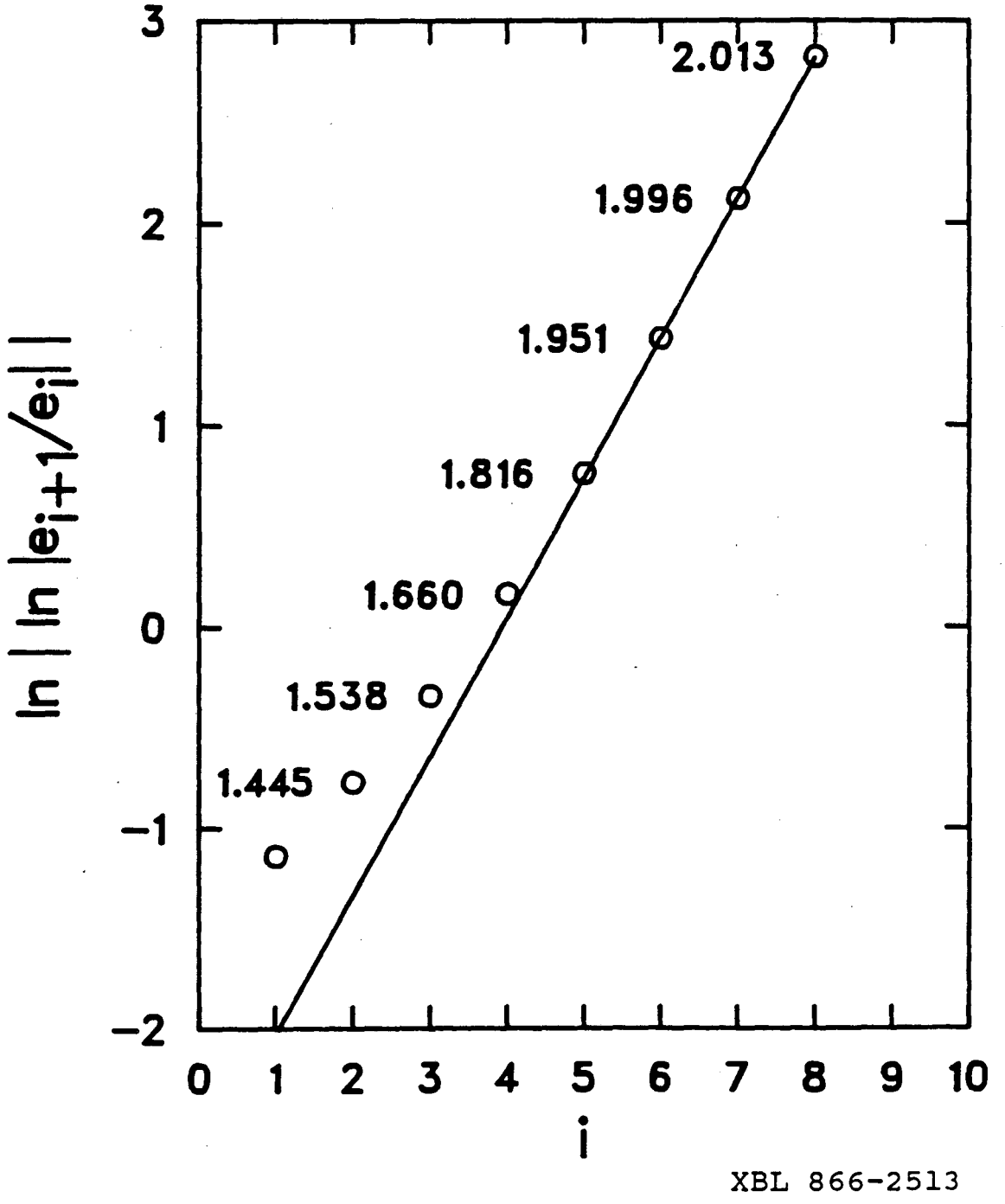


Figure D-1. Plot of absolute errors for a Newton-Raphson iteration.

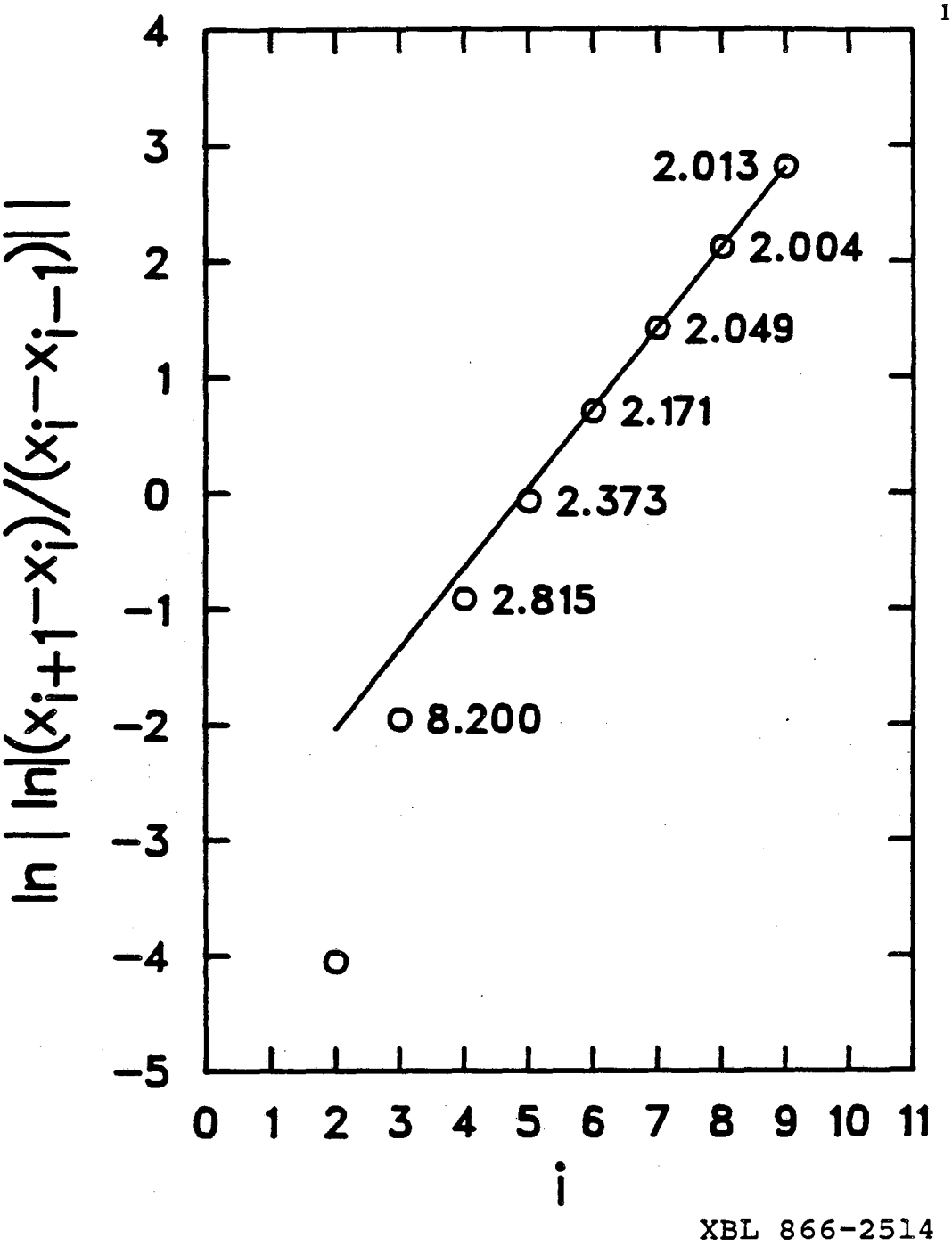


Figure D-2. Plot of relative errors for a Newton-Raphson iteration.

tion number, i, and figure D-2 is a plot of $\ln \left| \ln \left| \frac{x_i - x_{i+1}}{x_{i-1} - x_i} \right| \right|$, where $x_i - x_{i+1}$ is a "local" estimate of the error e_{i+1}. The motivation for plotting the log of the log of the error comes from the definition of quadratic convergence (equation D-5). By recursion, equation D-5 is rewritten $e_i = C_2{}^{(2^i-1)}\, e_0{}^{2^i}$, and further manipulation yields

$$\ln \left| \ln \left| \frac{e_{i+1}}{e_i} \right| \right| = i \ln 2 + \ln \left| \ln C_2 + \ln e_0 \right| . \tag{D-8}$$

For the more general convergence behavior, $e_{i+1} = C_p e_i{}^p$,

$$\ln \left| \ln \left| \frac{e_{i+1}}{e_i} \right| \right| = i \ln p + \ln \left| \ln C_p + \ln e_0{}^{p-1} \right| . \tag{D-9}$$

Thus the slope of the plot of $\ln \left| \ln \left| \frac{e_{i+1}}{e_i} \right| \right|$ indicates the order of convergence p. (The slope is $\ln p$.) In figures D-1 and D-2, the numbers next to each point are the exponentials of the slope; they approach 2.0 as the solution is approached quadratically and deviate in the last iteration owing to roundoff error. In figures D-3 and D-4, the iteration was by successive substitution, so the convergence is first order. The calculations for figures D-1 through D-4 were carried out in single precision on a CDC-7600 (16 significant figures).

Clearly, the Newton-Raphson method has the advantage of fast convergence. The disadvantage is that the method requires the evaluation and inversion of the Jacobian (the matrix of partial derivatives of f). An alternative to calculating the derivatives analytically is to calculate them numerically with a suitable step size ("discretized Newton iteration") or to use the same inverted Jacobian for several iterations.[100] Another approach is to estimate the derivatives based on information from the previous two iterations. For example, the secant method is

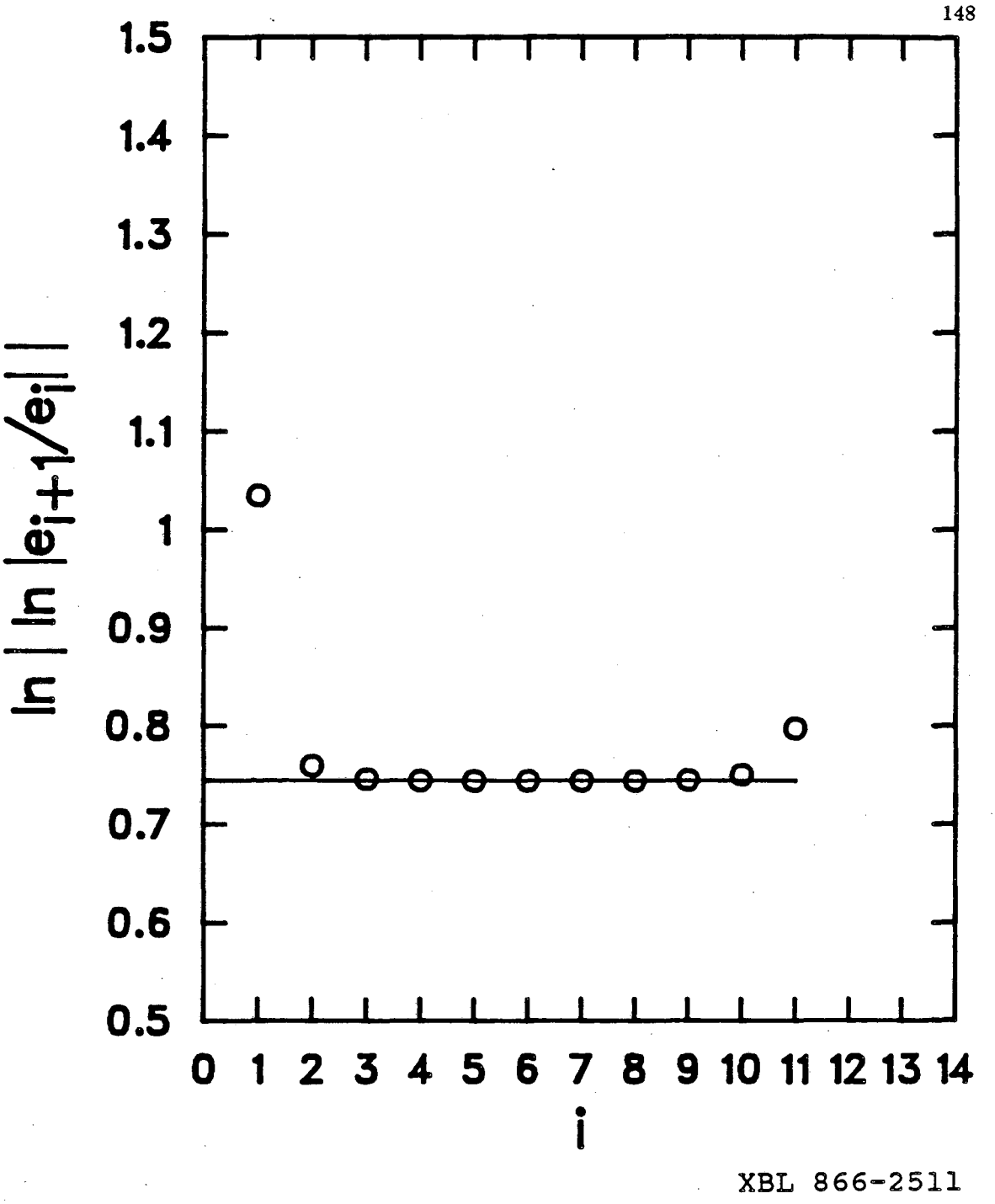


Figure D-3. Plot of absolute errors for successive substitution.

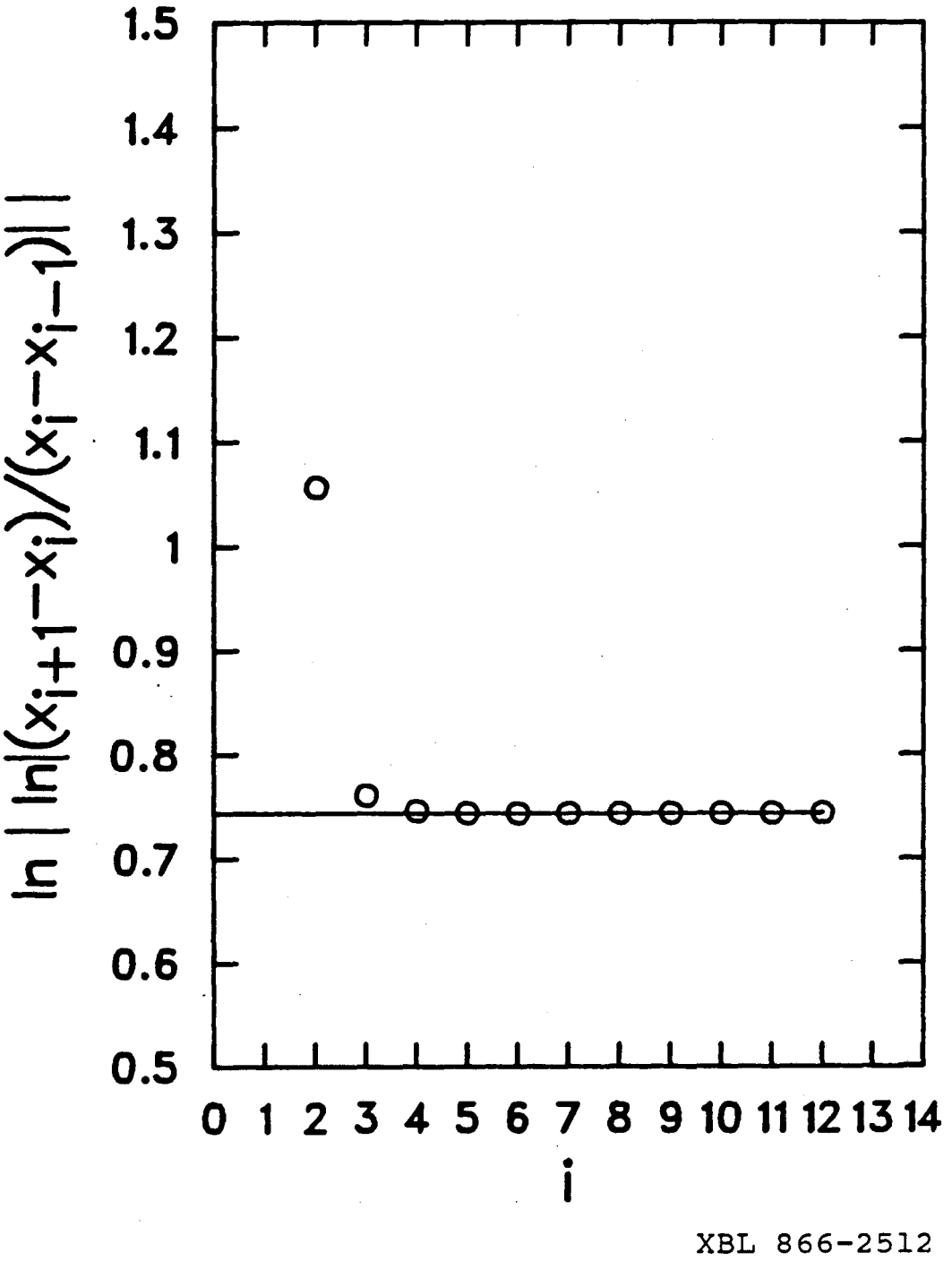


Figure D-4. Plot of relative errors for successive substitution.

$$x_{i+1} = x_i - \frac{f(x_i)\,(x_i - x_{i-1})}{f(x_i) - f(x_{i-1})}, \tag{D-10}$$

which is easily generalized to multidimensional problems.[100, 101] The advantage of the secant method is that f' is not needed, but, on the other hand, the disadvantages are that convergence is slower than quadratic (order 1.62) and more storage is required.

Traub[62] discusses the concept of "computational efficiency" to account for the cost of evaluating f and its derivatives and the number of iterations required for convergence to a given number of significant figures. For example, for a one-dimensional problem, if the cost of calculating the first derivative of f is more than 0.44 times the cost of calculating f, then the secant method is "cheaper" than the Newton-Raphson method.[101] In the one-dimensional case, the secant method is cheaper than the successive substitution method because both methods require just one evaluation of f at each iteration and the secant method converges faster. In the n-dimensional case, the comparison should consider the cost of calculating the entries of the Jacobian matrix and the cost of inverting it. Thus, for a large system of equations, the successive substitution method could be cheaper than a higher-order method.

D.3. Application of Some Iterative Methods to the Rotating-Disk Problem

The rotating-disk problem was used as an example problem to assess the feasibility of applying the Newton-Raphson method to the overpotential iterations in the thin-gap flow cell problem. The rotating-disk problem is described elsewhere,[102] but is presented here in summary.

Given a potential distribution, one can calculate the concentration and current distribution at the surface of a rotating-disk electrode by equating the Fick's-law and Faraday's-law expressions for the flux. The resulting current distribution is then substituted into Laplace's equation to calculate a new potential distribution. The solution to Laplace's equation evaluated at the disk surface represents the potential in the solution

just outside the double layer with the prevailing current distribution, but without concentration variations. For a rotating disk, this potential is

$$\Phi_o(r) = \sum_{n=0}^{\infty} B_n P_{2n}(\eta) \,, \tag{D-11}$$

where P_{2n} is a Legendre polynomial of order $2n$ and η is a rotational elliptic coordinate evaluated at the disk surface (see reference 102). The coefficients B_n are calculated from the current distribution:

$$B_n = [P_{2n}(0)]^2 \frac{(4n+1)\pi r_o}{2\kappa_\infty} \int_0^1 \eta P_{2n}(\eta) i \, d\eta \,. \tag{D-12}$$

If the total overpotential at the center of the disk,

$$E(0) = V - \Phi_o(0) \,, \tag{D-13}$$

is fixed, then the total overpotential, $E = V - \Phi_o$, at any other radial position is

$$E = V - \Phi_o(0) - \sum_{n=0}^{\infty} B_n [P_{2n}(\eta) - P_{2n}(1)] \tag{D-14}$$

Thus, a loop can be constructed in which a potential distribution produces a current distribution through the flux equations, and the current distribution, in turn, produces a set of B coefficients and a new potential distribution (equations D-9 through D-12).

D.3.1. Cutting the Loop

To apply a method of successive approximation to the rotating-disk problem, one must decide where to "cut" the iterative loop, that is, one must decide which variables are to be updated by the iterating function. For the rotating-disk problem, it is best to cut the loop at the B coefficients. The channel problem, however, does not have B coefficients, so the possibility of cutting the loop at the overpotentials, $E(r)$, was also considered.

D.3.2. Calculating the Jacobian

Because the successive substitution method is straightforward, we shall concentrate our discussion on the Newton-Raphson method, which is more complex because the Jacobian matrix must be calculated (either numerically or analytically). Although numerical derivatives are easier to program than the analytic derivatives, difficulties may arise because one must choose an appropriate step size to use in the difference calculations. This step size must be large enough to avoid roundoff error, but small enough that the derivative of the function is represented accurately by the numerical approximation. In spite of this disadvantage, the numerical derivatives are useful, and they provide an easy check for programming errors in calculating the analytic derivatives.

To calculate the analytic derivatives in a complex problem, one can envision two subroutines — one calculates the value of a function $f_i(x)$, and the other calculates $\partial f_i/\partial x_j$. The second subroutine can be regarded as the "derivative" of the first subroutine. To write the derivative subroutine, one would work with new variables that represent the partial derivatives of the original variables with respect to x_j.

Sample programs to illustrate the concept of "differentiating" a computer program are given in figures D-5 and D-6, and the results are plotted in figures D-1 through D-4. Program EXAMPLE (figure D-5) uses successive substitution, and program DEXAMPL (figure D-6) uses the Newton-Raphson method with analytic derivatives. In this simple one-dimensional example, there are three lines of code to calculate F, and, in program DEXAMPL, three lines of code to calculate DF, the 1×1 matrix of derivatives. Note that these three lines could be written in a separate subroutine, as was done for the rotating-disk problem.

Appendix E lists the programs for the rotating-disk problem. The programs are revisions of a program written by Newman to solve for the current distribution on a rotating disk below the limiting current using the method of successive substitution with

```
c       Sample program to solve x = (exp(x**2) - 0.5)/3.
c       by Newton's method
c
        implicit double precision (a-h,o-z)
c
        dimension x(51)
c
   40   format(' no convergence in',i4,' iterations')
   50   format(' iteration       x')
   60   format(1x,i3,5x,1pe17.10)
c
        itmax = 50
c
        iter = 1
        x(iter) = 0.d0
        print 60, iter, x(iter)
        do 20 iter = 1,itmax
c
        xold = x(iter)
        a = x(iter)**2
        b = dexp(a)
        f = (b - 0.5)/3.
c
c-----------------------------
        da = 2.*x(iter)
        db = b*da
        df = db/3.
c-----------------------------
c
c-----------------------------------------------------
        x(iter+1) = xold + 1./(1. - df) * (f - xold)
c-----------------------------------------------------
c
        print 60, iter, x(iter+1)
   20   if (dabs(x(iter+1)-xold) .le. 1.d-10*dabs(x(iter+1)))
     1    go to 25
c
        print 40, itmax
   25   stop
        end
```

Figure D-5. Sample program for successive substitution.

```
c       Sample program to solve x = (exp(x**2) - 0.5)/3.
c       by successive substitution
c
        implicit double precision (a-h,o-z)
c
        dimension x(51)
c
   40   format(' no convergence in',i4,' iterations')
   50   format(' iteration       x')
   60   format(1x,i3,5x,1pe17.10)
c
c--------------------------------
        damp = 1.d0
c--------------------------------
c
        itmax = 50
c
        iter = 1
        x(iter) = 0.d0
        print 60, iter, x(iter)
        do 20 iter = 1,itmax
c
        xold = x(iter)
        a = x(iter)**2
        b = dexp(a)
        f = (b - 0.5)/3.
c
c---------------------------------------
        x(iter+1) = xold + damp*(f-xold)
c---------------------------------------
c
        print 60, iter, x(iter+1)
   20   if (dabs(x(iter+1)-xold) .le. 1.d-10*dabs(x(iter+1)))
     1    go to 25
c
        print 40, itmax
   25   stop
        end
```

Figure D-6. Sample program for Newton-Raphson method.

damping. Program CURDB uses the Newton-Raphson method to iterate on the B coefficients, and program CURDE iterates on the overpotentials, $E(r)$. Programs CURDBN and CURDEN use numerical derivatives; they were written to check for programming errors in computing the analytic derivatives.

Although the converged results of programs CURD, CURDB, and CURDE are the same, the computer time and the number of iterations are different. In general, the Newton-Raphson method takes fewer iterations but more computer time than successive substitution (see table D-1). Note that the computer time for the Newton-Raphson method depends on the size of the Jacobian, not only because it is expensive to calculate the derivatives, but also because the "cost" of inverting an $n \times n$ matrix is proportional to n^3.[103] It is because of the large Jacobian that the successive substitution method is more attractive for the rotating-disk problem.

D.4. Iterative Methods for the Channel Problem

D.4.1. Successive Substitution

Based on our comparison of successive substitution to the Newton-Raphson method for the rotating-disk problem, we chose to use successive substitution with damping for the channel problem. Unfortunately, several problems would not converge with this method. Some improvement was made with the scheme discussed in chapter 1 for

Table D-1. Comparison of Iterative Methods for the Rotating Disk Problem

	Successive Substitution	$(\text{Newton})_B$	$(\text{Newton})_E$
Iterations	41	6	6
CP Seconds	0.693	1.739	2.943

adjusting the damping factor as the iterations proceed, but the computer time was significant (several hundred seconds on a CDC-7600 or about 20 minutes on a VAX-8600, given a good initial guess).

D.4.2. Newton-Raphson with a Global Approximation Method

The reason the successive substitution method had seemed more attractive for the channel problem than a Newton-Raphson method was that the Newton-Raphson method would be too time-consuming because of the large Jacobian. For example, with 51 mesh points on each electrode, the Jacobian would be 102×102.

Fortunately, the problem of large Jacobians can be circumvented by using a global approximation method, such as orthogonal collocation,[22, 104, 105] which is a method of weighted residuals. The idea of collocation is to express the solution as a sum of trial functions multiplied by coefficients that are determined by minimizing the error between the approximate solution and the exact solution. The advantage of a global approximation, compared to a local approximation, is that the Jacobian is small; only a small set of coefficients is required, instead of a large set of variables evaluated at mesh points. A good approximation to the solution between the collocation points is available from the global trial functions and the coefficients. Naturally, the accuracy of the solution between collocation points depends on the choice of approximating functions (see section D.5.2).

D.5. Collocation Applied to the Channel Problem

D.5.1. Introduction

Cabán and Chapman[22] used orthogonal collocation with polynomial trial functions to solve the problem, originally solved by Parrish and Newman,[21] of current distribution in a channel flow cell with thin boundary layers. With just seven collocation points on each electrode, they could approximate with reasonable accuracy the current distribution along the electrodes. One advantage of their method is that few mesh points are needed, so the Jacobian is small enough to use the Newton-Raphson method, which has second-order convergence. Another important advantage they mention is that the numerical integrations — Laplace's equation and the superposition integrals — are performed once, instead of repeatedly, as they were in the successive-substitution method used here (see chapter 1) and in Parrish and Newman's work. The successive substitution method requires repeated integrations for the potentials and fluxes because the integrands contain unknowns. Cabán and Chapman's method, on the other hand, uses known approximating polynomials for the current, and the iteration is on the unknown coefficients, which can be taken outside the integrals. Because the trial functions are polynomials, however, their method cannot predict accurately the singular behavior of the current distribution near the electrode edges in cases such as the primary or limiting current distribution. Other investigators[106-108] have addressed this problem by using functions that correspond to known solutions of the governing equations. For example, Rhee[106] solved the problem of asymmetric suction and heating in a flow channel with one porous wall by superposing functions that have a step-change in temperature or flux on one wall. Miksis[107,108] solved the problem of primary current distribution in a rotating ring/disk system by adding to a collection of Legendre polynomials some special functions to match the singularities at the electrode edges.

To apply Miksis' method to the channel problem, one must decide which variables to represent as a sum of trial functions, *e.g.* current or potential. At first, we tried to preserve the method of chapter 1; therefore the only choice was to approximate the potentials. After realizing that this method would still be time-consuming (because the integrations are carried out inside the nested iteration loops), we decided to restructure the iteration procedure to take advantage of the linearity of Laplace's equation and Fick's law, as done by Cabán and Chapman. Thus, the integrals are evaluated once with the trial functions, and the subsequent iteration is over the nonlinearities in the Butler-Volmer equation.

There are many possible ways to structure the iteration scheme for the channel problem. One question is which variables to approximate — current, concentration, potential, or a mixture. A related question is how to write the superposition integrals. For example, Cabán and Chapman use trial functions for the current and write the concentration as a superposition integral over the currents.

Several factors determine which variables are best to approximate. One consideration is the ease of solving for the other variables. For example, if potentials were approximated, it would be difficult to extract the currents from Laplace's equation. Another factor is the accuracy of the approximation. Because concentration and flux are related by an integral equation, instead of by a direct proportionality, a set of current polynomials does not produce a set of concentration polynomials, and *vice versa.* This means that a finite set of polynomials for one variable may not adequately approximate the other variables; for example a set of current polynomials cannot accurately represent the limiting current, which has a step change in concentration. A third criterion, which makes current functions attractive, is that some special cases are known, such as the primary and limiting current distributions (constant-potential and constant-concentration boundary conditions, respectively). These limiting cases may be

used to supplement the collection of orthogonal polynomials, as was done by Miksis.[107, 108]

D.5.2. Choice of Trial Functions

A good choice of approximating functions can increase the accuracy, particularly for limiting cases such as the primary or limiting current distribution, where a series of orthogonal polynomials is inadequate to represent the singularity at the electrode edge. Accordingly, we adopt a method similar to that of Miksis,[107, 108] wherein special functions are added to the collection of orthogonal polynomials.

The trial functions chosen were a set of Legendre polynomials, an approximation to the limiting-current distribution, a function that is similar to the limiting current, but finite at the leading edge, and two functions that each approximate the behavior of a secondary current distribution on one edge only. For the limiting-current function, we superposed the Graetz functions from chapter 2 to produce a function that corresponds to a step change in concentration of the limiting species on both electrodes. Stepping to zero on both electrodes has the advantage that the current has the proper singular behavior at the leading edge, but is small away from the region of interest. The second special function is simply

$$i(x) = (x + \Delta_1)^{-1/3}, \tag{D-15}$$

a modification of the Lévêque solution, and the last two functions are

$$i(x) = (x + \frac{\Delta_2}{L/h + 1})^{-1/2} \tag{D-16}$$

and

$$i(x) = (1 - x + \frac{\Delta_3}{L/h + 1})^{-1/2}. \tag{D-17}$$

The parameters Δ are used to make the current finite at the electrode edges, and they may be chosen based on an estimate of the expected current at the edge. The aspect

ratio appears in the denominator because the size of the region whose current deviates from the primary distribution depends on the geometry of the system.

D.5.3. Basic Iteration Scheme

We can now discuss the basic iteration scheme, which is built around approximating the partial current densities, i_j, as superpositions of trial functions. First the linear calculations are carried out; then the Newton-Raphson method iterates to find the coefficients. We shall later mention some revisions to this basic method that can improve the convergence behavior, particularly for limiting-current problems.

Recall that the advantage of the new scheme over successive substitution is that the linear integral equations are evaluated only once. These equations are the two flux expressions, Fick's and Faraday's laws, and Laplace's equation for the potential. Although it might seem convenient to follow Cabán and Chapman's method and write the Fick's-law flux as a superposition integral with the fluxes (currents) inside the integrals, we chose not to do so because their equation incorporates the thin-boundary-layer assumption. Since we had already calculated the thick-boundary-layer equation with concentrations inside the integrals, we chose to use the integral (from chapter 1):

$$N_{i,cath}(x) = -\frac{D_i}{B}\int_0^x \frac{dc_{i,cath}}{dx}\Big|_{x'} \frac{\partial\theta}{\partial\xi}(\varsigma-\varsigma',\xi=-1)\,dx'$$

$$+\frac{D_i}{B}\int_0^x \frac{dc_{i,an}}{dx}\Big|_{x'} \frac{\partial\theta}{\partial\xi}(\varsigma-\varsigma',\xi=1)\,dx' \,. \qquad \text{(D-18)}$$

The other linear equations are Faraday's law

$$N_i = -\sum_j \frac{s_{ij}}{n_j}\, i_j, \qquad \text{(D-19)}$$

and the solution to Laplace's equation

$$\Phi^{o}{}_{cath}(x) = \Phi^{*} - \frac{1}{2\pi\kappa_{\infty}} \left\{ \int_{0}^{L} [i_{cath}(x') \ln \sinh^{2}(\frac{\pi(x-x')}{2h}) \right.$$

$$\left. + i_{an}(x') \ln \cosh^{2}(\frac{\pi(x-x')}{2h})] \, dx' \right\}. \tag{D-20}$$

The method begins by expressing the partial current densities for each reaction as a sum of coefficients multiplied by trial functions:

$$i_{j,an}(x) = \sum_{k=1}^{np} A_{j,k,a} \; i_{j,k,a}(x) \tag{D-21a}$$

and

$$i_{j,cath}(x) = \sum_{k=1}^{np} A_{j,k,c} \; i_{j,k,c}(x). \tag{D-21b}$$

This representation will be used to set up the preliminary linear calculations.

Given the entire collection of approximating functions and a guessed set of coefficients, the potentials and concentrations can be calculated from Laplace's equation and the flux expressions, respectively.

Since

$$i_{an} = \sum_{j} i_{j,an}, \tag{D-22}$$

we have

$$i_{an} = \sum_{j=1}^{nrxn} \sum_{k=1}^{np} i_{j,k,a} \, A_{j,k,a}, \tag{D-23}$$

which we shall rewrite as

$$i_{an} = \sum_{jka=1}^{(np)(nrxn)} i_{jka} \, A_{jka}, \tag{D-24a}$$

and

$$i_{cath} = \sum_{jkc=1}^{(np)(nrxn)} i_{jkc} \, A_{jkc}, \tag{D-24b}$$

where *jka* and *jkc* are compound indices that collapse the two sums into one. For example, *jka*, the anode index, is defined as

$$jka = (j-1)np + k. \qquad \text{(D-25)}$$

Equations D-24a and D-24b are the starting point for setting up the linear calculations. Note that each of the trial functions in equations D-24 corresponds to a prescribed partial current density on one electrode and zero current on the other. The coefficients and trial functions are triply subscripted because the partial current density for each reaction on each electrode (subscripts j and a) is a different superposition of the trial functions (index k). In principle, different trial functions could be used for each partial current and for each electrode. Here, we choose to use the same collection of functions for each electrochemical reaction, j, but, in some cases, different functions for the two electrodes (see section D.5.5). The sums in equations D-23 run from k=1 to np, where np is both the number of trial functions and the number of collocation points. The goal of the method is to solve for the set of coefficients that makes equations D-23 accurate at the collocation points, x_p. That is, the approximation to the current must agree with the current computed from the kinetic expression at the collocation points.

It is important to understand that the kinetic expression is nonlinear, but that the expressions for the potential and concentration are linear; therefore one can calculate, once, at the start of the program, the potentials and concentrations resulting from the individual current functions. These concentrations and potentials can then be viewed as trial functions because they are superposed with the same coefficients as those used to generate currents.

This can be seen by substituting into equation D-20 the simplified representations of i_a and i_c (equations D-24):

$$\Phi^{o}_{cath}(x) - \Phi^{*} = \sum_{jkc} \left(\int_{0}^{L} i_{jkc}(x') \ln \sinh^2\left(\frac{\pi(x-x')}{2h}\right) dx' \right) A_{jkc}$$

$$+ \sum_{jka} \left(\int_{0}^{L} i_{jka}(x') \ln \cosh^2\left(\frac{\pi(x-x')}{2h}\right) dx' \right) A_{jka} \qquad \text{(D-26)}$$

and

$$\Phi^{o}_{an}(x) - \Phi^{*} = \sum_{jka} \left(\int_{0}^{L} i_{jka}(x') \ln \sinh^2\left(\frac{\pi(x-x')}{2h}\right) dx' \right) A_{jka}$$

$$+ \sum_{jkc} \left(\int_{0}^{L} i_{jkc}(x') \ln \cosh^2\left(\frac{\pi(x-x')}{2h}\right) dx' \right) A_{jkc}. \qquad \text{(D-27)}$$

To calculate the concentrations, we use the two flux expressions — Faraday's law,

$$N_{i,cath} = - \sum_{jkc} \frac{s_{ij}}{n_j} i_{jkc} A_{jkc} \qquad \text{(D-28a)}$$

$$N_{i,jkc} = - \frac{s_{ij}}{n_j} i_{jkc}, \qquad \text{(D-28b)}$$

and Fick's law,

$$N_{i,jkc}(x) = - \frac{D_i}{B} \int_{0}^{z} \frac{dc_{i,cath,jkc}}{dx}\Big|_{x'} \frac{\partial\theta}{\partial\xi}(\zeta-\zeta', \xi=-1)\, dx'$$

$$+ \frac{D_i}{B} \int_{0}^{z} \frac{dc_{i,an,jkc}}{dx}\Big|_{x'} \frac{\partial\theta}{\partial\xi}(\zeta-\zeta', \xi=1)\, dx' . \qquad \text{(D-29)}$$

Equation D-34 is solved by the method of Acrivos and Chambré [59] to give $c_{i,an,jkc}$ and $c_{i,cath,jkc}$ evaluated at each of the collocation points along the electrodes. $c_{i,an,jkc}$ can be regarded as the concentration of species i on the anode resulting from current i_{jkc} on the cathode and zero current on the anode. Note that the linearity of the equations implies

$$c_{i,an}(x_p) - c_{i,feed} = \sum_{jka} c_{i,an,jka}(x_p) A_{jka} + \sum_{jkc} c_{i,an,jkc}(x_p) A_{jkc}. \qquad \text{(D-30)}$$

This equation may be written in matrix form as

$$\mathbf{c} = \underline{\mathbf{P}}\mathbf{A}, \qquad \text{(D-31)}$$

where $\mathbf{c}$ is the vector of concentrations of each species on each electrode at each

collocation point, x_p, $\mathbf{A}$ is a vector containing the A_{jka} and A_{jkc} coefficients, and $\underline{\mathbf{P}}$ is a $(2)(nspec)(np) \times (2)(nrxn)(np)$ matrix. This matrix form will be useful shortly for discussing the method for treating limiting-current problems.

It is useful to know the shapes of the concentration and potential functions because they must be superposed to produce reasonable concentration and potential profiles to be substituted into the Butler-Volmer kinetic expression. Clearly, if the potential is poorly represented, the current calculated from it will also be inaccurate, particularly for Tafel kinetics, where the current depends exponentially on the potential.

Figures D-7 through D-11 show the potential and concentration functions corresponding to functions for uniform current, a ramp current (shifted Legendre polynomial $P_1(x)$), and limiting current. Note that the uniform current produces a nonuniform potential (figure D-7). When this nonuniform potential distribution is substituted into the Butler-Volmer equation, a nonuniform current distribution results. This example demonstrates the need for choosing a set of current functions that can strike a balance between competing factors, such as ohmic potential drop and reaction kinetics.

With the potentials and concentrations computed from the potential and concentration functions, the Butler-Volmer partial currents are calculated, and the coefficients are found by a Newton-Raphson method that zeros the errors between the Butler-Volmer currents and the approximate currents at the collocation points, x_p, where

$$error_{an} = i_{j,B-V,an}(x_p) - \sum_{jka} i_{jka}(x_p)\, A_{jka} \tag{D-32a}$$

and

$$error_{cath} = i_{j,B-V,cath}(x_p) - \sum_{jkc} i_{jkc}(x_p)\, A_{jkc}. \tag{D-32b}$$

Also, the integration constant Φ^* from equation D-20 must be determined to satisfy the condition of equal currents on the two electrodes. The vector of coefficients $\mathbf{A}$ is determined by solving for

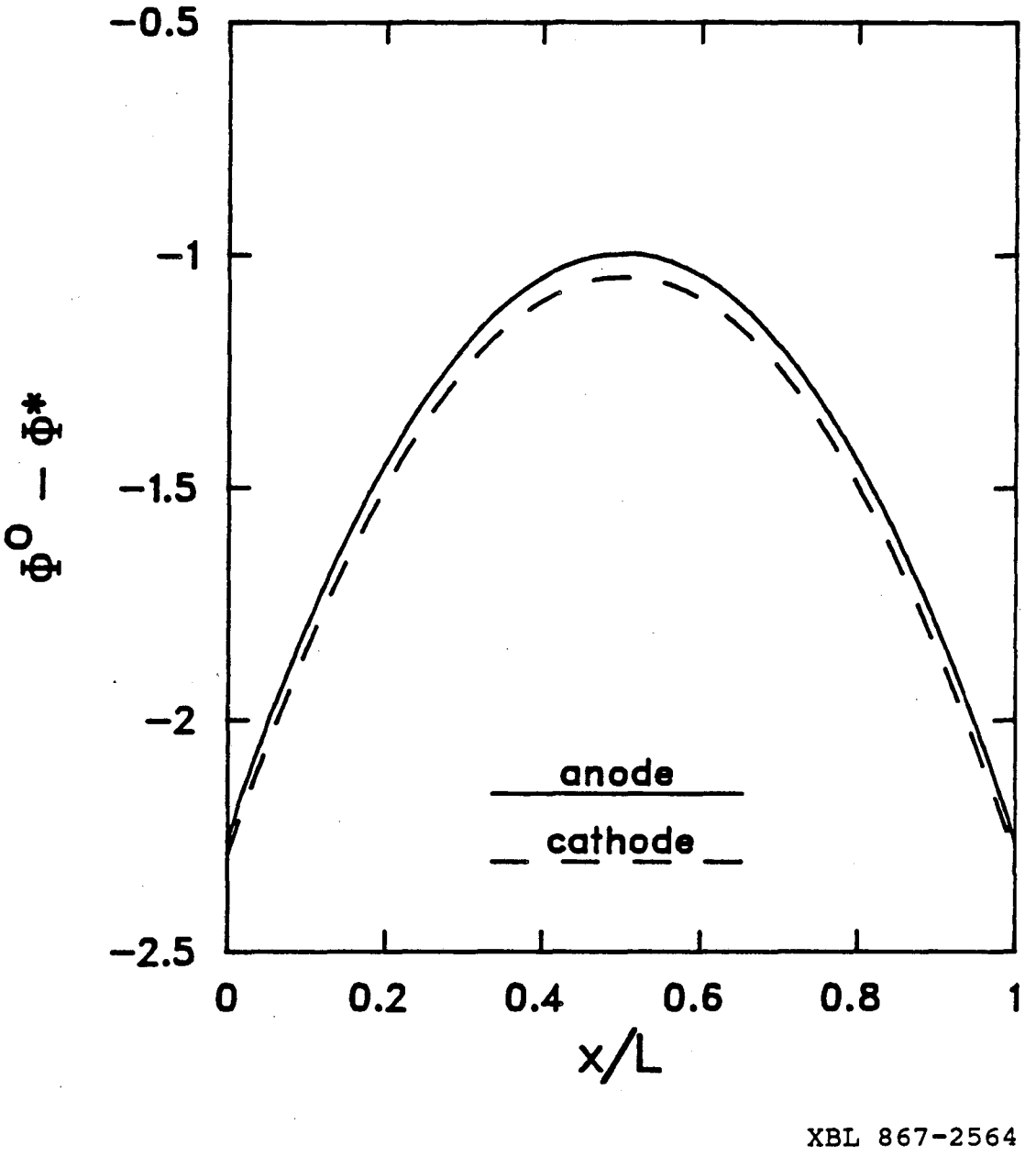


Figure D-7. Potential functions resulting from a uniform current function on the anode ($h/L = 0.1, Pe\ h/L = 100$).

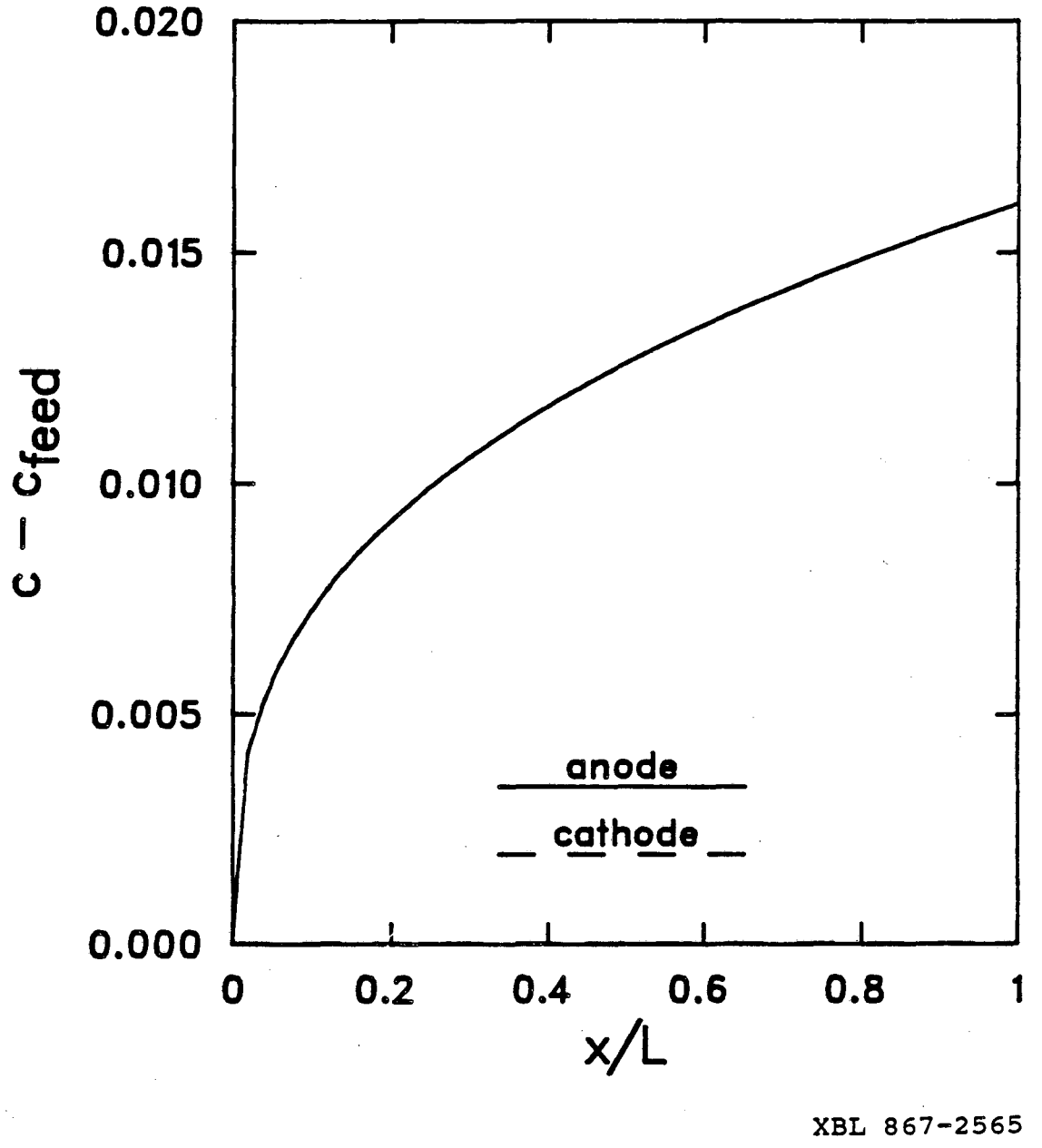


Figure D-8. Concentration functions resulting from a uniform current function on the anode. ($h/L = 0.1, Pe\ h/L = 100$).

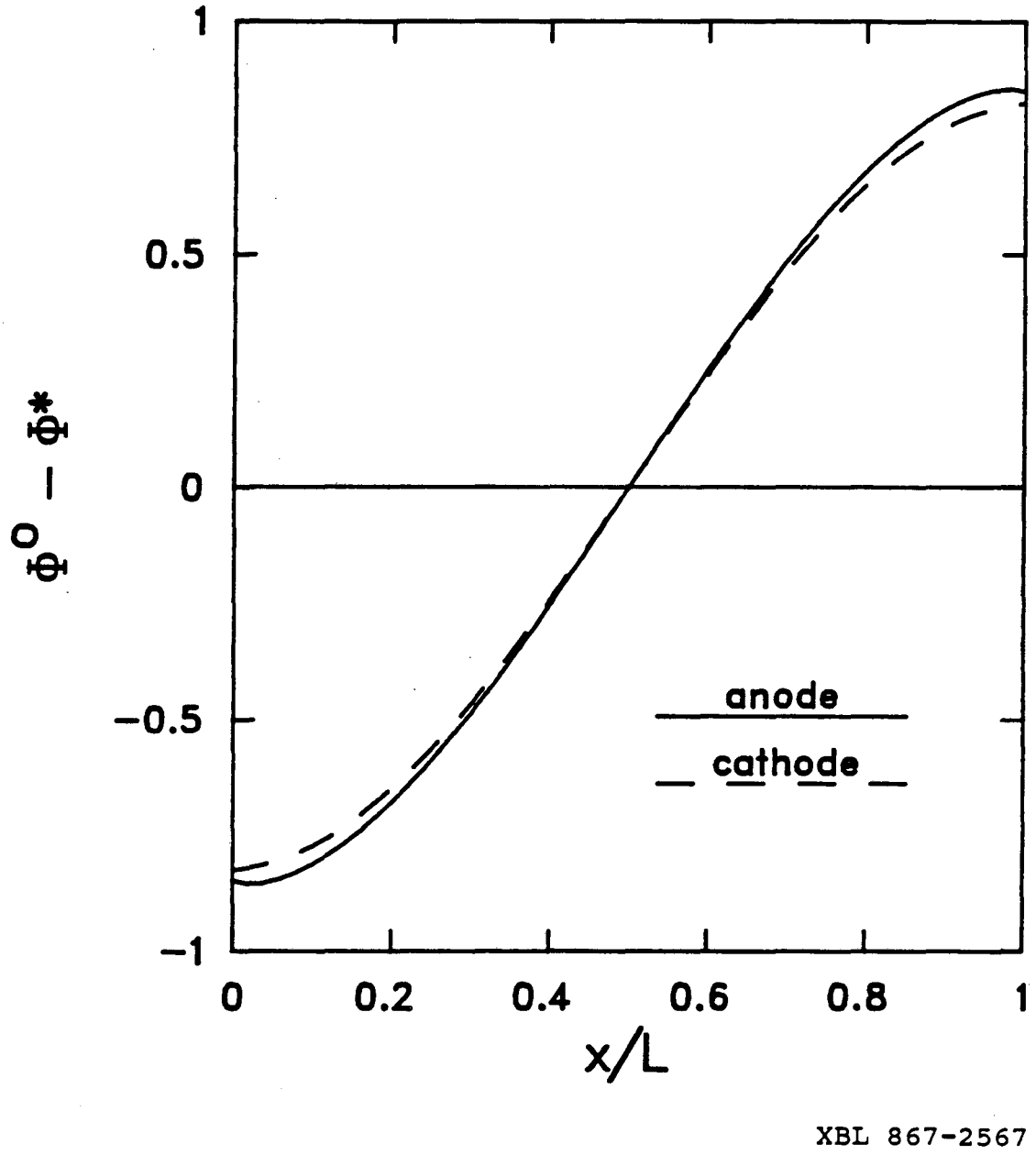


Figure D-9. Potential functions resulting from a ramp current function on the anode. ($h/L = 0.1, Pe\ h/L = 100$).

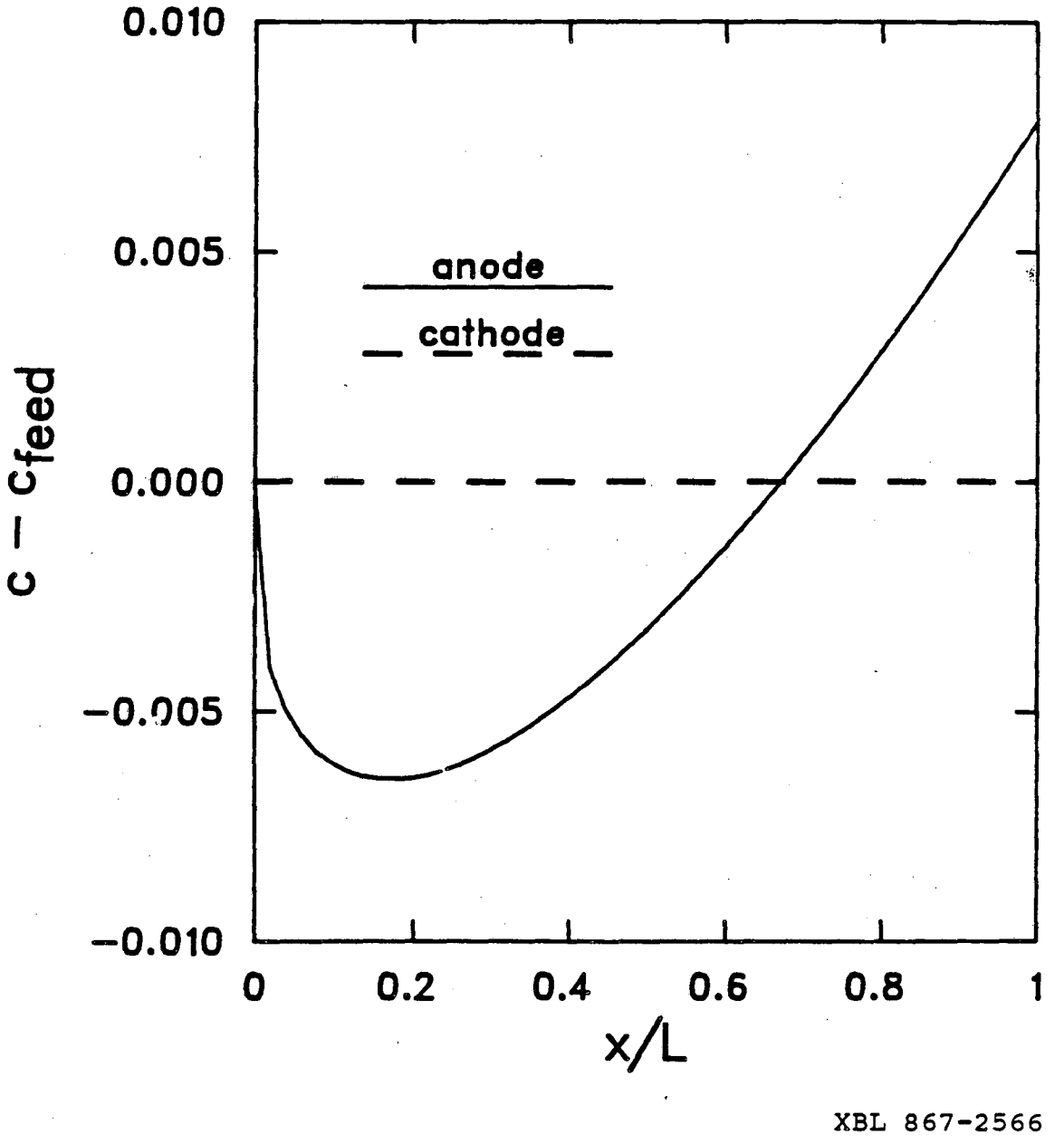


Figure D-10. Concentration functions resulting from a ramp current function on the anode. ($h/L = 0.1, Pe\ h/L = 100$).

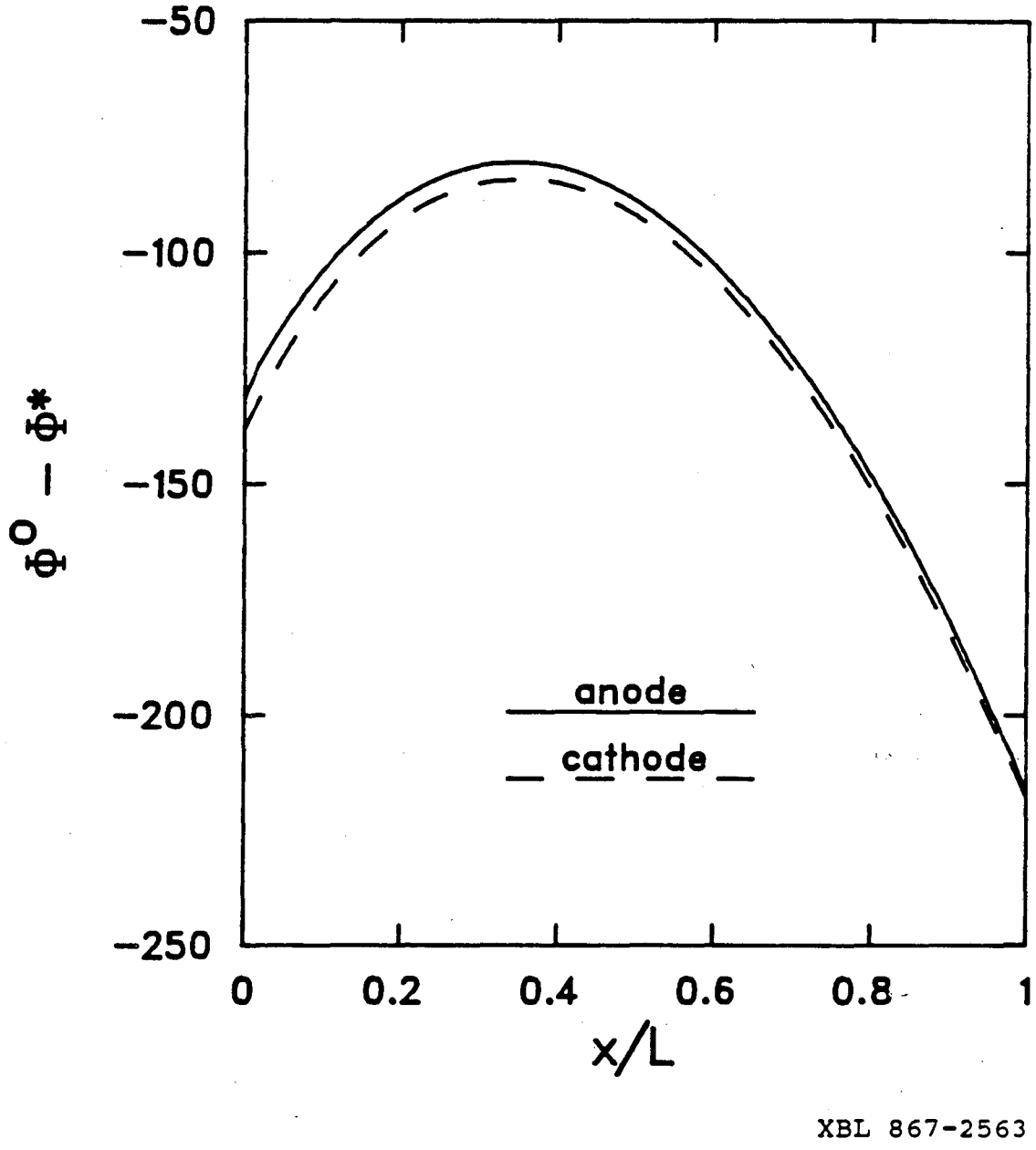


Figure D-11. Potential functions resulting from a limiting current function on the anode. ($h/L = 0.1, Pe\ h/L = 100$).

$$\Delta \mathbf{A} = \mathbf{A}^{new} - \mathbf{A}^{old} = -[\mathbf{Derror}]^{-1}\,(\mathbf{error})\Big|_{\mathbf{A}^{old}}, \tag{D-33}$$

where [**Derror**] is the Jacobian containing $\frac{\partial(i_{j,\mathrm{B-V}}(x_p) - i_{j,\mathrm{approx}}(x_p))}{\partial A_{jk}}$. The elements of the Jacobian are calculated by the chain rule

$$\frac{\partial i_{B-V,an}}{\partial A_{jkc}} = \sum_i \left(\frac{\partial c_i}{\partial c_{i,an}} \frac{\partial c_{i,an}}{\partial A_{jkc}} \right) + \frac{\partial i_{B-V,an}}{\partial (\Phi^o_{an} - \Phi^*)} \frac{\partial (\Phi^o_{an} - \Phi^*)}{\partial A_{jkc}}, \tag{D-34}$$

and the partial derivatives of concentration and potential with respect to the coefficients are simply the concentration and potential functions; for example,

$$\frac{\partial c_{i,an}}{\partial A_{jkc}} = c_{i,an,jkc}, \tag{D-35}$$

from equation D-30. Note that the matrix equation D-33 can be augmented with an extra row and column containing the equation for equal currents and the unknown Φ^*, respectively.

D.5.4. Limiting-Current Problems

The method described above is most successful below the limiting current. A disadvantage of the method is that it is possible, during the iterations, to get negative concentrations, even at the collocation points, x_p. Negative concentrations are not only unrealistic, but also they cannot be used in the Butler-Volmer equation to calculate new currents. To avoid this problem we treat limiting-current problems separately and set the surface concentrations to correspond to limiting current.

We shall first outline how to treat limiting-current problems with a single species. On the limiting electrode, for example, the cathode, we replace the equations D-32b with an equation for the error in concentration

$$c_{cath}(x_p) = 0 = c_{feed} + \sum_{jka} c_{an,jka}(x_p) A_{jka} + \sum_{jkc} c_{an,jkc}(x_p) A_{jkc}, \tag{D-36}$$

(*i.e.* some of the rows of the matrix equation (D-32) are replaced).

It is also useful to use different trial functions on the two electrodes. In particular,

the limiting-current function is used on the cathode only and the modified limiting-current function (equation D-15) is used on the anode only. This is done by replacing the equations for the errors at the last collocation points with equations that set the appropriate coefficients to zero.

Another manipulation for the limiting-current calculations is to provide an initial guess for Φ^* and the coefficients. The guessed coefficients are provided by equation D-31, with $c_{cath} = 0$ and $c_{an} = 2$, and equations that set the appropriate limiting-current-function coefficients to zero.

For problems with more than one species, it is cautioned that not all of the concentrations may be specified. The coefficients will still be obtained from equation D-36, but since the full $\underline{\mathbf{P}}$ matrix may be singular, not all of the concentrations may be constrained.

The number of concentrations that can be constrained depends on the rank of the (s_{ij}/n_j) matrix because Faraday's law must be inverted to extract the current coefficients from the fluxes. For example, the matrix for the $Fe(CN)_6^{3-}/Fe(CN)_6^{4-}$ system with the hydrogen-evolution side reaction is

$$\frac{s_{ij}}{n_j} = \begin{bmatrix} -1 & 0 \\ 1 & 0 \end{bmatrix}, \tag{D-37}$$

where the rows are for ferricyanide and ferrocyanide, respectively, and the columns are for the main and side reactions. Clearly, it is impossible to calculate the coefficients for the side reaction currents because the rank of (s_{ij}/n_j) is one. Note that, although $nspec = nrxn$, and $\underline{\mathbf{P}}$ is a square matrix, $\underline{\mathbf{P}}$ cannot be inverted to give the coefficients $\mathbf{A}$ from the concentrations $\mathbf{c}$ because the side-reaction current coefficients are multiplied by zeros. Therefore, only the A_{mk} coefficients can be calculated from the equation

$$\mathbf{c}_R = \underline{\mathbf{M}}\mathbf{A}_m, \tag{D-38}$$

where m is the index for the main reaction, R is the index for the limiting species on the

given electrode, *elec*, and $\underline{\mathbf{M}}$ is the nonsingular $(2)(rank)(np)\times(2)(rank)(np)$ matrix containing $c_{R,elec,mk}(x_p)$ (see equation D-32). In the ferricyanide example, R might be ferricyanide on the cathode and ferrocyanide on the anode; therefore by constraining $c_{ferri,cath}$ and $c_{ferro,anode}$, one can calculate $A_{1k,an}$ and $A_{1k,cath}$ from equation D-38 by inverting $\underline{\mathbf{M}}$. The method then proceeds with the new $\mathbf{A}_1$ and the old $\mathbf{A}_2$ vectors.

For problems that do not converge readily, the continuation method (section D.5.5) is sometimes helpful. The best results are usually obtained by starting with a small cell voltage and applying continuation to reach the desired cell voltage.

D.5.5. First-Order Continuation

It is useful in any iterative problem to use a method that provides a good initial guess. One such method is first-order continuation.[109] The idea behind the method is that if the problem can be made to converge by changing one parameter, then the iterations can start from the "easy" problem and proceed to the difficult problem by gradually changing the parameter. The initial guess for each new problem is provided by a first-order extrapolation from the converged solution of the previous problem. For example, to solve $\mathbf{f}(\mathbf{x}) = 0$, one would write

$$\mathbf{f}(\mathbf{x},p) = \mathbf{f}(\mathbf{x}_o,p_o) + \frac{\partial \mathbf{f}}{\partial \mathbf{x}}\Big|_{\mathbf{x}_o,p_o}\Delta\mathbf{x} + \frac{\partial \mathbf{f}}{\partial p}\Big|_{\mathbf{x}_o,p_o}\Delta p, \tag{D-39}$$

where p_o is the value of the parameter p that gives convergence. Since

$$\mathbf{f}(\mathbf{x}_o,p_o) = 0, \tag{D-40}$$

$$\Delta\mathbf{x} = -\left[\frac{\partial \mathbf{f}}{\partial \mathbf{x}}\Big|_{\mathbf{x}_o,p_o}\right]^{-1}\frac{\partial \mathbf{f}}{\partial p}\Big|_{\mathbf{x}_o,p_o}\Delta p; \tag{D-41}$$

therefore, for a given Δp, one can guess a new $\mathbf{x}$:

$$\mathbf{x}_{new} = \mathbf{x}_o + \Delta\mathbf{x}. \tag{D-42}$$

Note that $\frac{\partial \mathbf{f}}{\partial \mathbf{x}}$ is the Jacobian for solving $\mathbf{f}(\mathbf{x}) = 0$ by the Newton-Raphson method;

therefore some time can be saved by using "LU decomposition" rather than matrix inversion.[103] As expected, Δp must be chosen carefully; it should be large enough to make progress toward the desired p, but small enough that the linear extrapolation provides a good enough initial guess to achieve convergence in each new problem. An automatic parameter step-size adjustment procedure is to count the number of Newton iterations required for each problem and adjust Δp according to the following scheme:[110]

Newton Iterations	Δp^{new}
< 3	$3\ \Delta p^{old}$
$= 3$	Δp^{old}
> 3	$0.56\ \Delta p^{old}$

This procedure is an attractive candidate for the channel problem because convergence can be achieved under certain conditions, such as small cell voltages or small exchange current densities for side reactions. A disadvantage of the method is that convergence can be too slow.

D.6. Results at the Limiting Current

Figures D-12 through D-15 show results from the limiting-current algorithm (see program NEWCHAN, appendix G) for large and small aspect ratios ($h/L = 1$ and 0.1). Nine collocation points were used on each electrode, and about 20 iterations were required to achieve convergence without a "restart." The computer time for these runs was about 4 minutes on a VAX 8600, roughly 20 times faster than for the old program (program CHANNEL).

Figures D-12 and D-13 show the anode current density as calculated from the trial functions and as calculated from the nonlinear Butler-Volmer expression. The circles on

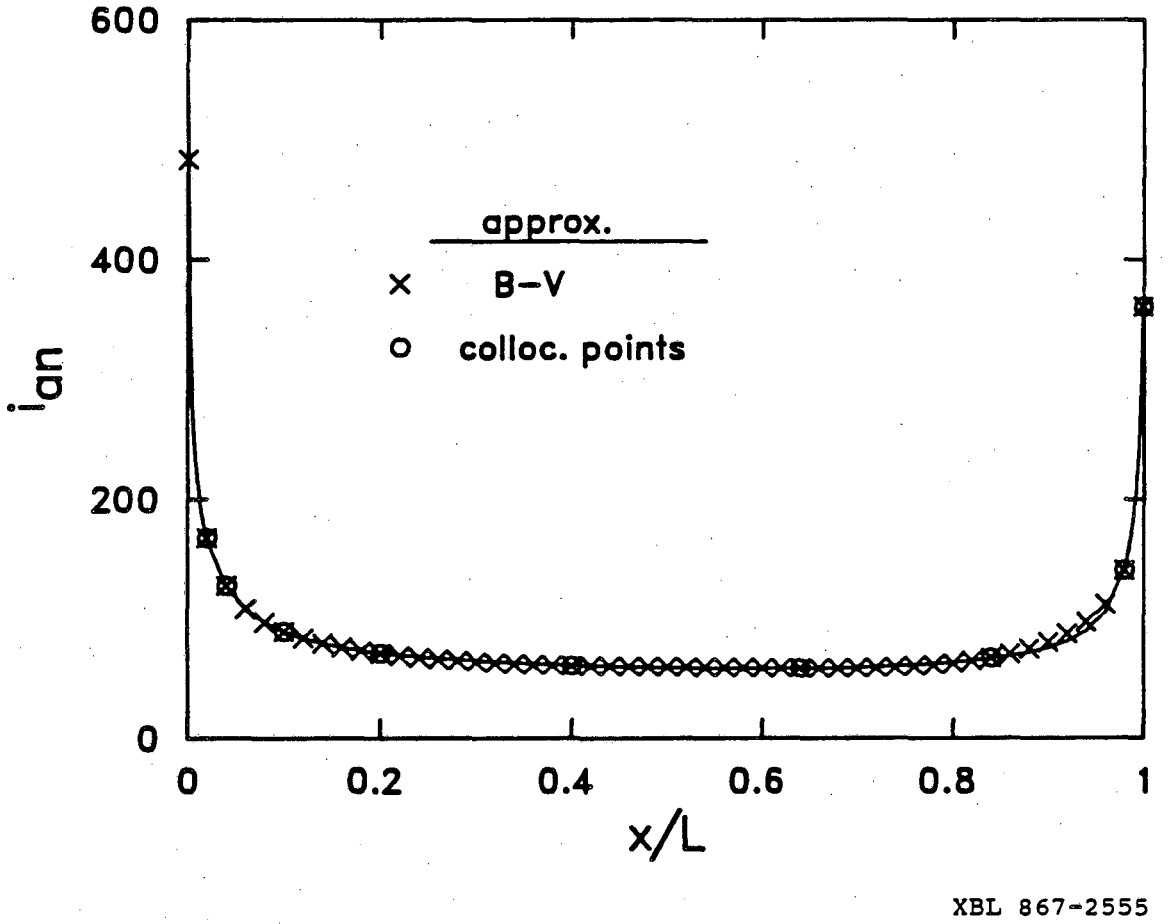


Figure D-12. Anode current distributions calculated from trial functions and Butler-Volmer expression. ($h/L = 1$, $Peh/L = 100$).

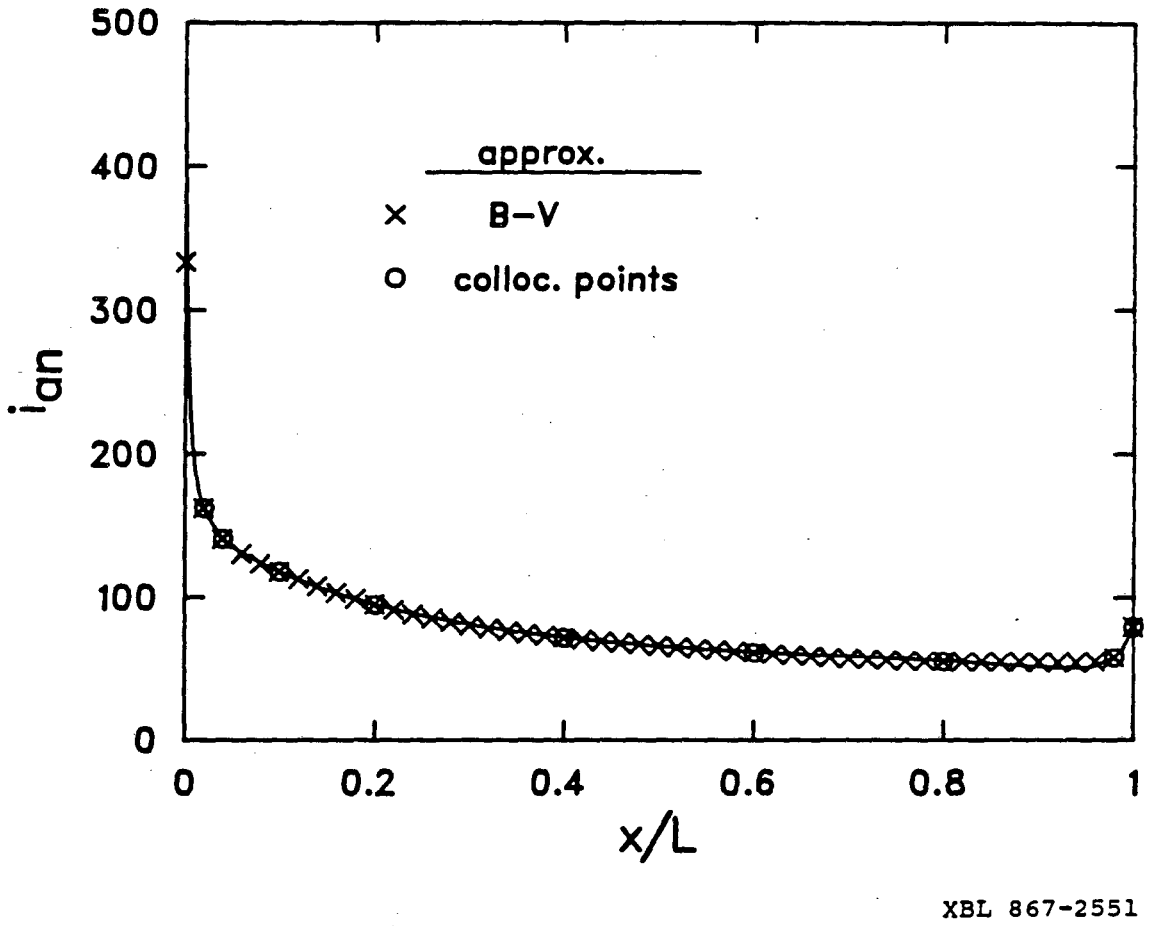


Figure D-13. Anode current distributions calculated from trial functions and Butler-Volmer expression. ($h/L = 0.1$, $Peh/L = 100$).

the figures are the collocation points at which agreement was required. The close agreement between the currents at points other than these nine points shows that an appropriate set of trial functions was used. With a poor set of functions, the two curves would have to oscillate to agree at the nine points. Note that the choice of the parameters Δ for the special trial functions is important. To obtain figures D-12 and D-13, we guessed a set of Δ's and estimated i_{max}/i_{min} from the converged (but oscillatory) solution. The Δ's were then refined so that each trial function had the correct i_{max}/i_{min}. One additional refinement was sufficient to give the fit shown in figure D-12.

Figures D-14 and D-15 show the potential distributions. Recall that the potential becomes uniform across the channel infinitely far upstream and downstream, provided the total currents are equal on the two electrode. If the total currents were unequal on the electrodes, the potentials would continue to change, corresponding to current flowing to $+\infty$ and $-\infty$. The distance over which the potentials decay to the constant value depends on the amount of fringing of the current lines (see figure 3-1). The fringing generally occurs over a distance of order h, as shown in the figures (see also figure 5-19).

Figure D-14 shows the potential distribution for large h/L ($h/L = 1$). Note that the potential adjacent to the anode is nearly flat, corresponding to the nearly-primary current distribution. Although the cathode is at limiting current, the anode potential distribution is unaffected because h/L is large enough to suppress interaction between the two electrodes. The cathode potential distribution is highly nonuniform.

Figure D-15 shows the potential distribution for $h/L = 0.1$. For this smaller h/L, there is an interaction through Laplace's equation, and, therefore, the current distributions on the anode and cathode are nearly identical, except at the leading edge. As in section 5.4, the dipole of $(i_a + i_c)$ leads to a significant difference between the upstream and downstream potentials. Note that although the axial current is smaller for $h/L = 0.1$ than for $h/L = 1$, the difference between the upstream and downstream

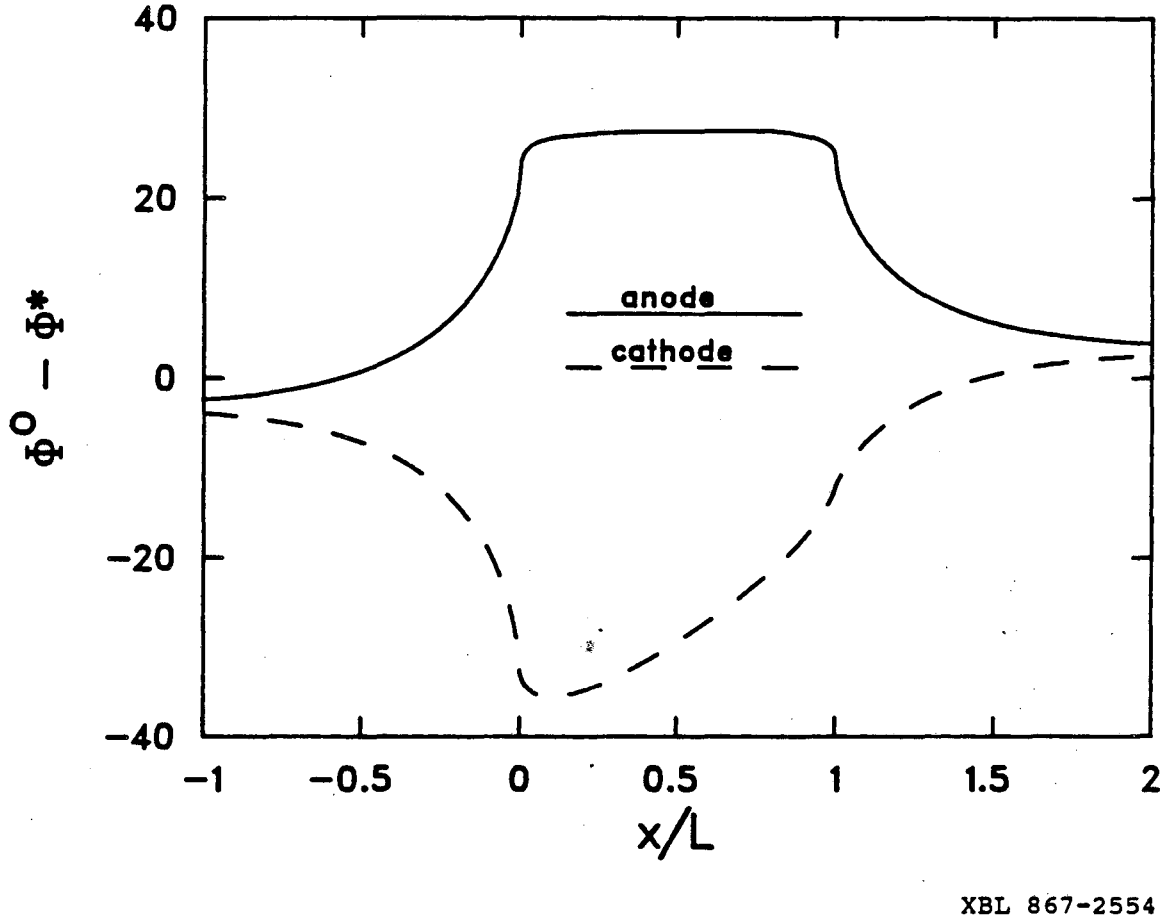


Figure D-14. Potential distributions for $h/L = 1$, $Peh/L = 100$).

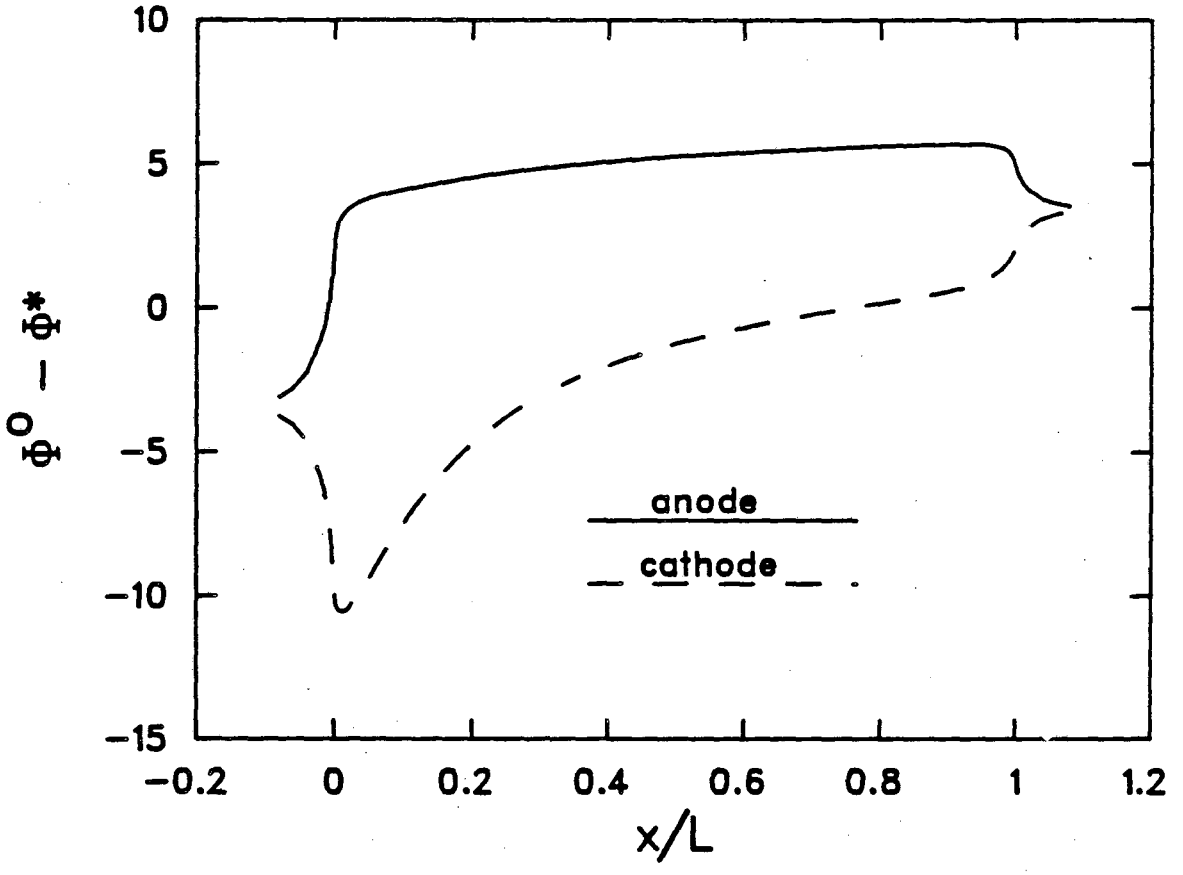


Figure D-15. Potential distributions for $h/L = 0.1$, $Peh/L = 100$).

potentials is larger. This example illustrates the importance of accounting for axial current and potential variations, even for small h/L.

D.7. Summary and Suggestions for Future Work

We have investigated iterative techniques with the objective of devising an efficient and accurate method for solving the thin-gap-flow-cell problem. The collocation method as revised by Miksis[107,108] has proven to be faster than the successive substitution method and more accurate than the simple collocation method used by Cabán and Chapman.[22] The disadvantages are that the current functions must be selected carefully.

Program NEWCHAN (appendix G), in its present form, is not guaranteed to be robust. Suggestions for improving the program are to automate the selection of the parameters contained in the special functions and the automate the choice between the basic algorithm and the limiting-current algorithm.

To automate the selection of the Δ parameters from equations D-15 through D-17, one should have an *a priori* estimate of the total current from either the limiting, primary, or uniform current distribution. Next, one would estimate the ratio of the current at the edge to the minimum current for linear or Tafel kinetics:[111]

$$\text{Linear:} \qquad i_{max} = \sqrt{\frac{J}{2}\,\frac{\tanh\epsilon}{\epsilon}}\,\bar{\phi}_o\, i_{min} \tag{D-43a}$$

$$\text{Tafel:} \qquad i_{max} = \frac{\delta\, e^{\bar{\phi}_o}\, i_{min}}{K(\tanh^2\epsilon)}, \tag{D-43b}$$

where δ in equation D-43b is the dimensionless average current. The parameters Δ would then be chosen to match the ratio i_{max}/i_{min}.

The choice between the basic algorithm and the limiting-current algorithm might be governed by the surface concentration; if it falls below some threshold, the program would switch to the limiting-current method. The goal would be to produce a program

that could generate by first-order continuation an entire polarization curve with current, concentration, and potential distributions at each applied potential.

Appendix E. Programs for Investigating Iterative Methods for the Rotating-Disk Problem

E.1. Program CURDB

Description:

Program CURDB is Newman's program CURD rewritten for iteration on the B coefficients by a Newton-Raphson method instead of successive substitution with damping. The program solves for the current, concentration, and potential distribution on a rotating-disk electrode below the limiting current (see John Newman, "Current Distribution on a Rotating Disk below the Limiting Current," *Journal of the Electrochemical Society*, **113** (1966), 1235-1241 and Ralph Edward White, *Simultaneous Reactions on a Rotating Disk Electrode*, dissertation, University of California, Berkeley (March, 1977).

Input Variables

LMAX = number of radial mesh points

NMAX = number of B coefficients

IH = half the number of Gaussian quadrature points

IHH : not used

IM = total number of Gaussian quadrature points

X(I) = Gaussian quadrature roots (in the interval 0 to 1)

W(I) = Gaussian quadrature weights

C(1) = c/c_∞ at center of disk (input C(1) = 0 to stop running cases)

AN = dimensionless limiting current = N

TPLUS = transference number

ALPHA = anodic transfer coefficient

BETA = cathodic transfer coefficient

GAMMA = exponent for concentration dependence of exchange current density

EXCH = dimensionless exchange current density = J

DAMP = damping factor (not used)

Output Variables

ERR = 1 if too many iterations, 0 otherwise

ANS(N) = converged B(n)

ITFIN = number of iterations required for convergence

ERR(N,IT) = error in B(n) at iteration IT = e_{IT}

ORD(N,IT) = $\ln \left| \ln \left| \frac{e_{IT+1}}{e_{IT}} \right| \right|$

SLOPE = slope of ORD *vs.* IT

V = disk potential

AVG = $\delta/N = i_{avg}/i_{lim}$

R(J) = radial position

C(J) = surface concentration c/c_∞

CUR(J) = dimensionless current density, $i/<i>$

E(J) = total overpotential = $V - \Phi_o$

B(I) = B coefficients

TAF = 0 if Tafel approximation is used, 1 if not

RAT2 = (V - E(1))/B(1), proportional to effective resistance

RAT1 = CUR(1)/(AVG*N*$\theta_o'(0)$), ratio of current at center to average current

JCOUNT = iterations required

```
      PROGRAM CURDB(INPUT,OUTPUT)
C     CURRENT DISTRIBUTION ON A ROTATING DISK WITH AN INTEGRAL EQUATION
C     FOR THE DIFFUSION LAYER
c
c     This is program curd rewritten for iteration on B coefficients
c     by a Newton-Raphson method
c
      DIMENSION PM(21),B(21),C(201),CUR(201),E(201),PP(21,201),R(201),AA
     1(200),BB(200),X(40),W(40),CUG(40),PG(21,40),de(201),bd(21,21) ,
     1d(21,43),bin(21),xarr(51,51),err(51,51),ord(51,51),ans(51)
c
      COMMON/a/ E,CUR,C,LMAX,TPLUS,AN,EXCH,ALPHA,BETA,GAMMA,TAF,AA,BB,C1
     1,C2,EX
      common/b/nmax,pp,im,x,dz,r,pg,w,pm
c
  101 FORMAT (6H ERROR,I4)
  102 FORMAT (3H N=,F10.4,10H ,  TPLUS=,F8.4,6H ,  V=,F10.5,8H ,  AVG=,
     1F10.6/41H0      R          C          CUR        ETA/(4F11.5))
  103 FORMAT (3I4)
  104 FORMAT (9E8.4)
  105 FORMAT (7H1ALPHA=,F8.4,9H ,  BETA=,F8.4,10H ,  GAMMA=,F8.4,9H ,  E
     1XCH=,F8.4)
  107 FORMAT (2H0B,F9.5,5F11.5/(6F11.5))
  108 FORMAT (6E12.9)
  109 format (* calculated derivatives*)
  132 format(1x,3e22.15)
c
      print 109
      READ 103, LMAX,NMAX
      mmax = 2*nmax+1
      EX = 2.0/3.0
c
      DO 29 L=1,LMAX
      A = L
      AA(L) = 2.0*A**EX - (A+1.0)**EX - (A-1.0)**EX
   29 BB(L) = A**EX - (A-1.0)**EX
c
      DZ = 1.0/(LMAX-1)
c
      DO 1 L=1,LMAX
        Z = (L-1)*DZ
        R(L) = Z**(1.0/3.0)
        ETA = SQRT(1.0-R(L)*R(L))
      DO 1 N=1,NMAX
    1 PP(N,L) = P(2*N-2,ETA)
c
      DO 2 N=1,NMAX
    2 PM(N) = - 0.6366198/P(2*N-2,0.0)**2
c
      EX = 1.0/3.0
      C2 = 1.11984652
```

```
      READ 103, IH,IHH,IM
      IHP1 = IH + 1
      READ 108, (X(I),I=IHP1,IM)
      READ 108, (W(I),I=IHP1,IM)
c
      DO 34 I=1,IM
        IF (I-IH)  31,31,32
   31   IR = IM - I + 1
        X(I) = 0.5 - 0.5*X(IR)
        W(I) = W(IR)
        GO TO 33
   32   X(I) = 0.5 + 0.5*X(I)
   33   XX = SQRT(1.0-X(I)**2)
      DO 34 N=1,NMAX
   34 PG(N,I) = P(2*N-2,XX)
c
    3 READ 104, C(1),AN,TPLUS,ALPHA,BETA,GAMMA,EXCH,DAMP
      C1 = C2*(1.0-C(1))
      IF (C(1))  4,4,5
    4 STOP
    5 JCOUNT = 0
      TAF = 1.0
c
c     Test for Tafel approximation
c
      IF (EXCH-400)  7,7,6
c
    6 TAF = 0.0
      EXCH = 1.0
    7 ETAC = ALOG(C(1)) + TPLUS*(1.0-C(1))
      CUR(1) = - (1.0-C(1))*AN*EXCH*1.11984652/C(1)**GAMMA
      ETAS = -ALOG(TAF-CUR(1))/BETA
c
      IF (CUR(1))  8,10,8
c
    8 DO 9 J=1,100
        F = TAF*EXP(ALPHA*ETAS) - EXP(-BETA*ETAS)
        FP = TAF*ALPHA*EXP(ALPHA*ETAS) + BETA*EXP(-BETA*ETAS)
        IF (ABS(CUR(1)-F) - 0.0000001*ABS(CUR(1))) 10,10,9
    9 ETAS = ETAS + (CUR(1)-F)/FP
c
   10 CUR(1) = 1.11984652*AN*(1.0-C(1))
      DO 11 L=1,LMAX
        C(L) = C(1)
   11 E(L) = ETAC + ETAS
      B(1) = 0.0
      PRINT 105, ALPHA,BETA,GAMMA,EXCH
      BOLD = B(1)
c
      JCOUNT = JCOUNT + 1
      CALL THETA
c
```

```
      DO 16 I=1,IM
        LI = X(I)**3/DZ + 1
   16 CUG(I)= CUR(LI)+(CUR(LI+1)-CUR(LI))*(X(I)**2-R(LI)**2)/(R(LI+1)**2
     1-R(LI)**2)
c
      DO 15 N=1,NMAX
        B(N) = 0.0
        DO 14 I=1,IM
   14   B(N) = B(N) + CUG(I)*X(I)*PG(N,I)*W(I)
   15 B(N) = 0.5*B(N)*(4*N-3)/PM(N)
c
   12   bold=b(1)
c
        do 310 n=1,nmax
          xarr(n,jcount) = b(n)
c         print 160,jcount,b(n),bold,xarr(n,jcount)
  310   continue
c
        jcount = jcount+1
c
        do 90 i=1,nmax
   90   bin(i) = b(i)
c
c       calculate Jacobian
c
        do 100 i=1,nmax
  100   call deriv(i,bd,b,v)
c
c       do 90 i=1,nmax
c  90   bin(i)=b(i)
c       call theta
c     DO 216 I=1,IM
c     LI= X(I)**3/DZ + 1
c 216 CUG(I)= CUR(LI)+(CUR(LI+1)-CUR(LI))*(X(I)**2-R(LI)**2)/(R(LI+1)**2
c    1-R(LI)**2)
c     DO 215 N=1,NMAX
c     B(N)= 0.0
c     DO 214 I=1,IM
c 214 B(N)= B(N) + CUG(I)*X(I)*PG(N,I)*W(I)
c 215 B(N)= 0.5*B(N)*(4*N-3)/PM(N)
c
c
c       print 145
c 145   format(* hand-calc new bs, b2,b1*)
        b2 = (b(2)-bd(2,2)*bin(2)) / (1. - bd(2,2))
        b1 = b(1) + bd(1,2)*(b2-bin(2))
c       print 132,b2,b1
c
        do 110 i=1,nmax
        do 110 j=1,mmax
  110   d(i,j) = 0.
c
```

```
        do 111 i=1,nmax
        do 111 j=1,nmax
          xi = 0.
          if(i.eq.j) xi = 1.0
  111   bd(i,j) = xi - bd(i,j)
c
        do 112 i=1,nmax
  112   d(i,1) = b(i) - bin(i)
c
        call matinv(nmax,mmax,determ,bd,d)
        if(determ.eq.0.) print 116
  116   format(27h zero determinant in matinv)
c
c       update
c
        do 117 i=1,nmax
  117   b(i) = bin(i) + d(i,1)
c
        do 320 n=1,nmax
        xarr(n,jcount) = b(n)
c       if(jcount.gt.1) print 321,n,xarr(n,jcount)-xarr(n,jcount-1)
c 321   format(* n=*,i2,* deltax=*,1pe11.4)
  320   continue
c
      JERR = 1
      IF (JCOUNT-50)  19,19,20
   19 crit = (ABS(B(1)-BOLD) - 0.000001*ABS(B(1)))
        do 330 n=1,nmax
c       print 160,jcount,b(n),bold,xarr(n,jcount)
  160   format(* jcount=*,i2,* b(n)=*,1pe11.4,* bold=*,1pe11.4,* xarr=*,
     1    1pe11.4)
  330   continue
      IF (ABS(B(1)-BOLD) - 1.e-10*ABS(B(1)))  21,21,12
c     IF (ABS(B(1)-BOLD) - 0.000001*ABS(B(1)))  21,21,12
   20 PRINT 101, JERR
   21 AVG = - B(1)/0.7853982/AN/1.11984652
c
c       Test errors for quadratic convergence
c
        do 340 n=1,nmax
          ans(n) = b(n)
  340   continue
c
        itfin = jcount
        do 350 n=1,nmax
          print 70,n,ans(n),itfin
   70     format(* n=*,i2,* answer=*,1pe17.10,* itfin=*,i2)
  350   continue
c
        print 59
   59   format(* it      err(n,it)*)
c
```

```
      do 30 n=1,nmax
        print 61,n
 61     format(* n=*,i2)
      do 30 it=1,itfin
        err(n,it) = xarr(n,it) - ans(n)
        print 60,it,err(n,it)
 60     format(1x,i3,5x,1pe15.8)
 30   continue
c
      print 91
 91   format(*  it          ord(it)*)
c
      itfm1 = itfin - 1
      do 35 n=1,nmax
        print 61,n
      do 35 it=1,itfm1
        ord(n,it) = 0.
        if(err(n,it).eq.0. .or. err(n,it+1).eq.0.) go to 36
        ord(n,it) = alog(abs(alog(abs(err(n,it+1)/err(n,it)))))
 36     print 60,it,ord(n,it)
 35   continue
c
      itfm2 = itfin - 2
      do 37 n=1,nmax
        print 61,n
      do 37 it=1,itfm2
        slope = ord(n,it+1) - ord(n,it)
        if(ord(n,it+1).eq.0.) go to 37
        print 80,it,slope,exp(slope)
 80     format(1x,i3,5x,1pe11.4,5x,e11.4)
 37   continue
c
     DO 22 L=1,LMAX
 22  CUR(L) = CUR(L)/AN/1.11984652
     RAT1 = CUR(1)/AVG
     RAT2 = (V-E(1))/B(1)
     PRINT 102, AN,TPLUS,V,AVG,(R(J),C(J),CUR(J),E(J),J=1,LMAX)
     PRINT 107, (B(I),I=1,NMAX),TAF,RAT2,RAT1
     PRINT 103, JCOUNT
     GO TO 3
     END
c
c
     SUBROUTINE THETA
c
C    SUBPROGRAM FOR CALCULATING CONCENTRATION by equating Fick's-law
c    and Faraday's-law fluxes
c    (X1-X2)/PN + (C1+X3))*EXCH = 0
c
     DIMENSION E(201),CUR(201),A(200),B(200),TH(201)
c
     COMMON/a/ E,CUR,TH,NZT1,T,PN,EXCH,AL,BE,GAM,TAF,A,B,C1,C2,EX
```

```
c
  101 FORMAT (17HONOT CONVERGED AT,I4)
c
      NZT = NZT1-1
      DEVM = 0.0001
      DZ = 1.0/NZT
      S = TH(1)
      DO 60  NZ = 2,NZT1
      Z = (NZ - 1)*DZ
      SUM = 0.0
      IF (NZ .LE. 2) GO TO 42
C     CALC. SUM(TH(J)*A(K))
      DO 40 J=3,NZ
        K = NZ - J + 1
   40 SUM = SUM + TH(J-1)*A(K)
   42 ETA = E(NZ)
      NJ = NZ - 1
c
      DO 56 N=1,20
        X1 = TAF*S**(GAM - AL)*EXP(AL*ETA)*EXP(AL*T*(S - 1.0))
        DX1 = X1*((GAM - AL)/S + AL*T)
        X2 = S**(GAM + BE)*EXP(-BE*ETA)*EXP(BE*T*(1.0 - S))
        DX2 = X2*((GAM + BE)/S - BE*T)
        C3 = 1.50*C2*Z**EX/DZ**(1./3.)
        X3 = C3*(TH( 1)*B(NJ) + SUM - S)
        DTH= S - ((X1-X2)/PN + (C1+X3)*EXCH)/((DX1-DX2)/PN - C3*EXCH)
        CUR(NZ) = PN*(C1+X3)
        IF (ABS(S-DTH) - DEVM*ABS(DTH))  60,60,56
   56 S = DTH
c
      PRINT 101, NZ
   60 TH(NZ) = DTH
      RETURN
      END
c
c
      FUNCTION P(N,X)
C     CALCULATION OF LEGENDRE POLYNOMIALS
      P1= 1.0
      P2= X
      IF (N-1)  1,2,3
    1 P= P1
      RETURN
    2 P= P2
      RETURN
    3 NM1= N - 1
      DO 4 NU=1,NM1
      P=(X*FLOAT (2*NU+1)*P2-FLOAT (NU)*P1)/FLOAT (NU+1)
      P1= P2
    4 P2= P
      RETURN
      END
```

```
c
c
      subroutine deriv(i,bd,b,v)
c
c     This subroutine calculates derivatives with respect to the B
c     coefficients
c
      DIMENSION PM(21),B(21),C(201),CUR(201),E(201),PP(21,201),R(201),AA
     1(200),BB(200),X(40),W(40),CUG(40),PG(21,40),de(201),bd(21,21) ,
     1dcur(201),thd(201),btemp(21),dcug(40)
c
      COMMON/a/ E,CUR,C,LMAX,TPLUS,AN,EXCH,ALPHA,BETA,GAMMA,TAF,AA,BB,C1
     1,C2,EX
      common/b/ nmax,pp,im,x,dz,r,pg,w,pm
c
      v = e(1)
      DO 1 j=1,NMAX
        v = v + b(j)*pp(j,1)
    1 Btemp(j) = 0.0
c
      btemp(i) = 1.0
      de(1) = 0.
      dcur(1) = 0.
c
      DO 3 L=2,LMAX
        phi = v
        dPHI = 0.0
        DO 2 N=1,NMAX
          phi = phi - b(n)*pp(n,1)
    2   dPHI = dPHI - Btemp(N)*(PP(N,L) - pp(n,1))
        e(l) = phi
    3 dE(L) = dphi
c
   71 format(1x,3e22.15)
      CALL THETAp(de,i,dcur,thd)
c
      DO 4 Ii=1,IM
        LI = X(Ii)**3/DZ + 1
        CUG(Ii) = CUR(LI) + (CUR(LI+1)-CUR(LI))*(X(Ii)**2-R(LI)**2)/
     1  (R(LI+1)**2 - R(LI)**2)
    4   dCUG(Ii) = dCUR(LI)+(dCUR(LI+1)-dCUR(LI))*(X(Ii)**2-R(LI)**2)/
     1(R(LI+1)**2 - R(LI)**2)
c
      V = E(1)
      DO 6 N=1,NMAX
        if(i.eq.nmax) B(N) = 0.0
        Bd(N,i) = 0.0
        DO  5 Ii=1,IM
          if(i.eq.nmax) B(N) = B(N) + CUG(Ii)*X(Ii)*PG(N,Ii)*W(Ii)
    5   Bd(N,i) = Bd(N,i) + dCUG(Ii)*X(Ii)*PG(N,Ii)*W(Ii)
        Bd(N,i) = 0.5*Bd(N,i)*(4*N-3)/PM(N)
        if(i.eq.nmax) B(N) = 0.5*B(N)*(4*N-3)/PM(N)
```

```
    6 V = V + Bd(N,i)*PP(N,1)
c
        print 76,i
   76   format(* bd, column*,i2)
        print 71,(bd(n,i),n=1,nmax)
        return
        end
c
c
      SUBROUTINE THETAp(de,je,dcur,thd)
c
C     SUBPROGRAM FOR CALCULATING CONCENTRATION derivatives with
c     respect to the B coefficients
c
      DIMENSION E(201),CUR(201),A(200),B(200),TH(201),thd(201),de(201) ,
     1dcur(201)
c
      COMMON/a/ E,CUR,TH,NZT1,T,PN,EXCH,AL,BE,GAM,TAF,A,B,C1,C2,EX
c
  101 FORMAT (17HONOT CONVERGED AT,I4)
c
      NZT = NZT1-1
      DEVM = 0.0001
      DZ = 1.0/NZT
      S = TH(1)
        ds = 0.
        thd(1) = 0.
      DO 60  NZ = 2,NZT1
      Z = (NZ - 1)*DZ
      SUM = 0.0
      dsum = 0.0
      IF (NZ .LE. 2) GO TO 42
c
C     CALC. SUM(TH(J)*A(K))
c
      DO 40 J=3,NZ
      K = NZ - J + 1
        dsum = dsum + thd(j-1)*a(k)
   40 SUM = SUM + TH(J-1)*A(K)
c
   42 ETA = E(NZ)
      deta = de(nz)
      NJ = NZ - 1
c
      DO 56 N=1,20
      X1 = TAF*S**(GAM - AL)*EXP(AL*ETA)*EXP(AL*T*(S - 1.0))
      DX1 = X1*((GAM - AL)/S + AL*T)
      x1d = al*x1*deta + dx1*ds
      dx1d = x1d*((gam-al)/s + al*t) - x1*(gam-al)/s**2*ds
      X2 = S**(GAM + BE)*EXP(-BE*ETA)*EXP(BE*T*(1.0 - S))
      DX2 = X2*((GAM + BE)/S - BE*T)
      x2d = -be*x2*deta + dx2*ds
```

```
      dx2d = x2d*((gam+be)/s - be*t) - x2*(gam+be)/s**2*ds
      C3 = 1.50*C2*Z**EX/DZ**(1./3.)
      X3 = C3*(TH( 1)*B(NJ) + SUM - S)
      dx3 = c3*(dsum-ds)
      DTH= S - ((X1-X2)/PN + (C1+X3)*EXCH)/((DX1-DX2)/PN - C3*EXCH)
      dthd= ds - (((x1d-x2d)/pn + dx3*exch)*((dx1-dx2)/pn - c3*exch)
     1  - ((x1-x2)/pn + (c1+x3)*exch)*(dx1d-dx2d)/pn)/((dx1-dx2)/pn
     1  - c3*exch)**2
      dCUR(NZ) = PN*dX3
      CUR(NZ) = PN*(C1+X3)
      IF (ABS(S-DTH) - DEVM*ABS(DTH))  61,61,57
   57 S = DTH
   56 ds = dthd
c
      PRINT 101, NZ
   61 thd(nz) = dthd
   60 TH(NZ) = DTH
      RETURN
      END
c
c
      SUBROUTINE MATINV(N,M,DETERM,b,d)
c
c       This subroutine solves a matrix equation of the form BX=D.
c       The answer X is put into the first column of D.  Note that
c       D should be dimensioned nxm, where m=2n+1 and that all
c       columns of D (except the first) should be initiallized with
c       zeros
c
c       Input variables
c               n = dimension of square b matrix
c               m = number of columns in d matrix
c               b = matrix in equation bx=d
c               d : first column contains vector d in bx=d
c
c       Output variables
c               d : first column contains vector x in bx=d
c               determ = 0. if matrix has zero determinant
c
      DIMENSION ID(21),b(21,21),d(21,43)
c
      DETERM=1.0
      DO 1 I=1,N
   1  ID(I)=0
      DO 18 NN=1,N
      BMAX=1.1
      DO 6 I=1,N
      IF(ID(I).NE.0) GOTO 6
      BNEXT=0.0
      BTRY=0.0
      DO 5 J=1,N
      IF(ID(J).NE.0) GOTO 5
```

```
      IF(ABS(B(I,J)).LE.BNEXT) GOTO 5
      BNEXT=ABS(B(I,J))
      IF(BNEXT.LE.BTRY) GOTO 5
      BNEXT=BTRY
      BTRY=ABS(B(I,J))
      JC=J
    5 CONTINUE
      IF(BNEXT.GE.BMAX*BTRY) GOTO 6
      BMAX=BNEXT/BTRY
      IROW=I
      JCOL=JC
    6 CONTINUE
      IF(ID(JC).EQ.0) GOTO 8
      DETERM=0.0
      RETURN
    8 ID(JCOL)=1
      IF(JCOL.EQ.IROW) GOTO 12
      DO 10 J=1,N
      SAVE=B(IROW,J)
      B(IROW,J)=B(JCOL,J)
   10 B(JCOL,J)=SAVE
      DO 11 K=1,M
      SAVE=D(IROW,K)
      D(IROW,K)=D(JCOL,K)
   11 D(JCOL,K)=SAVE
   12 F=1.0/B(JCOL,JCOL)
      DO 13 J=1,N
   13 B(JCOL,J)=B(JCOL,J)*F
      DO 14 K=1,M
   14 D(JCOL,K)=D(JCOL,K)*F
      DO 18 I=1,N
      IF(I.EQ.JCOL) GO TO 18
      F=B(I,JCOL)
      DO 16 J=1,N
   16 B(I,J)=B(I,J)-F*B(JCOL,J)
      DO 17 K=1,M
   17 D(I,K)=D(I,K)-F*D(JCOL,K)
   18 CONTINUE
      RETURN
      END
~789
  41   3
  20  10  40
3877241751-21160840707-11926975807-12681521850-13419940908-14137792044-1
4830758017-15494671251-16125538897-16719566846-17273182552-17783056514-1
8246122308-18659595032-19020988070-19328128083-19579168192-19772599500-1
9907262387-19982377097-1
7750594798-27703981816-27611036190-27472316906-27288658240-27061164739-2
1791204582-26480401346-26130624249-25743976910-25322784698-24869580764-2
4387090819-23878216797-23346019528-22793700698-22224584919-21642105838-2
1049828453-24521277099-3
 70000-1 78800+1 36300-1 15000-0 50000-1 42000-1 89600-1 30000-1
```

```
10000-0 78800+1 36300-1 15000-0 50000-1 42000-1 89600-1 10000-0
50000-2 78800+1 36300-1 15000-0 50000-1 42000-1 89600-1 10000-0
20000-1 78800+1 36300-1 15000-0 50000-1 42000-1 89600-1 80000-1
30000-1 78800+1 36300-1 15000-0 50000-1 42000-1 89600-1 60000-1
40000-1 78800+1 36300-1 15000-0 50000-1 42000-1 89600-1 40000-1
50000-1 78800+1 36300-1 15000-0 50000-1 42000-1 89600-1 30000-1
60000-1 78800+1 36300-1 15000-0 50000-1 42000-1 89600-1 30000-1
70000-1 78800+1 36300-1 15000-0 50000-1 42000-1 89600-1 30000-1
80000-1 78800+1 36300-1 15000-0 50000-1 42000-1 89600-1 30000-1
90000-1 78800+1 36300-1 15000-0 50000-1 42000-1 89600-1 30000-1
95000-1 78800+1 36300-1 15000-0 50000-1 42000-1 89600-1 80000-1
80000-1 20000+1 50000-1 50000-1 50000-1 50000-1 50000+2 70000-1
80000-1 50000+1 50000-1 50000-1 50000-1 50000-1 50000+2 50000-1
20000-1 50000-0 50000-1 50000-1 50000-1 50000-1 50000+2 80000-1
50000-1 10000+2 50000-1 50000-1 50000-1 50000-1 50000+2 30000-1
50000-1 10000+1 50000-1 50000-1 50000-1 50000-1 50000+2 70000-1
50000-1 20000+1 50000-1 50000-1 50000-1 50000-1 50000+2 60000-1
```

E.2. Program CURDBN

Description:

Program CURDBN is Newman's program CURD rewritten for iteration on the B coefficients by a Newton-Raphson method with numerical derivatives instead of successive substitution with damping. The program solves for the current, concentration, and potential distribution on a rotating-disk electrode below the limiting current (see John Newman, "Current Distribution on a Rotating Disk below the Limiting Current," *Journal of the Electrochemical Society*, **113** (1966), 1235-1241 and Ralph Edward White, *Simultaneous Reactions on a Rotating Disk Electrode*, dissertation, University of California, Berkeley (March, 1977).

Input Variables

LMAX = number of radial mesh points

NMAX = number of B coefficients

IH = half the number of Gaussian quadrature points

IHH : not used

IM = total number of Gaussian quadrature points

X(I) = Gaussian quadrature roots (in the interval 0 to 1)

W(I) = Gaussian quadrature weights

C(1) = c/c_∞ at center of disk (input C(1) = 0 to stop running cases)

AN = dimensionless limiting current = N

TPLUS = transference number

ALPHA = anodic transfer coefficient

BETA = cathodic transfer coefficient

GAMMA = exponent for concentration dependence of exchange current density

EXCH = dimensionless exchange current density = J

DAMP = damping factor (not used)

Output Variables

ERR = 1 if too many iterations, 0 otherwise

ANS(N) = converged B(n)

ITFIN = number of iterations required for convergence

ERR(N,IT) = error in B(n) at iteration IT = e_{IT}

$$\text{ORD(N,IT)} = \ln \left| \ln \left| \frac{e_{IT+1}}{e_{IT}} \right| \right|$$

SLOPE = slope of ORD *vs.* IT

V = disk potential

AVG = $\delta/N = i_{avg}/i_{lim}$

R(J) = radial position

C(J) = surface concentration c/c_∞

CUR(J) = dimensionless current density, $i/<i>$

E(J) = total overpotential = $V - \Phi_o$

B(I) = B coefficients

TAF = 0 if Tafel approximation is used, 1 if not

RAT2 = (V - E(1))/B(1), proportional to effective resistance

RAT1 = CUR(1)/(AVG*N*$\theta_o'(0)$), ratio of current at center to average current

JCOUNT = iterations required

```
      PROGRAM CURDBN(INPUT,OUTPUT)
C     CURRENT DISTRIBUTION ON A ROTATING DISK WITH AN INTEGRAL EQUATION
C     FOR THE DIFFUSION LAYER
c
c     This is program curd rewritten for iteration on B coefficients
c     by a Newton-Raphson method with numerical derivatives
c
      DIMENSION PM(21),B(21),C(201),CUR(201),E(201),PP(21,201),R(201),AA
     1(200),BB(200),X(40),W(40),CUG(40),PG(21,40),db(21),bo(21),de(201),
     leo(201),thd(201),tho(201),d(21,43),bin(21),bd(21,21),
     lxarr(51,51),err(51,51),ord(51,51),ans(51)
c
      COMMON/a/ E,CUR,C,LMAX,TPLUS,AN,EXCH,ALPHA,BETA,GAMMA,TAF,AA,BB,
     1C1,C2,EX
      common/b/nmax,pp,im,x,dz,r,pg,w,pm,v
c
  101 FORMAT (6H ERROR,I4)
  102 FORMAT (3H N=,F10.4,10H ,  TPLUS=,F8.4,6H ,  V=,F10.5,8H ,  AVG=,
     1F10.6/41H0      R          C          CUR        ETA/(4F11.5))
  103 FORMAT (3I4)
  104 FORMAT (9E8.4)
  105 FORMAT (7H1ALPHA=,F8.4,9H ,  BETA=,F8.4,10H ,  GAMMA=,F8.4,9H ,  E
     1XCH=,F8.4)
  107 FORMAT (2H0B,F9.5,5F11.5/(6F11.5))
  108 FORMAT (6E12.9)
  109 format(* numerical derivatives*)
c
      print 109
      READ 103, LMAX,NMAX
      mmax = 2*nmax + 1
      EX = 2.0/3.0
c
      DO 29 L=1,LMAX
        A = L
        AA(L) = 2.0*A**EX - (A+1.0)**EX - (A-1.0)**EX
   29 BB(L) = A**EX - (A-1.0)**EX
c
      DZ = 1.0/(LMAX-1)
c
      DO 1 L=1,LMAX
        Z = (L-1)*DZ
        R(L) = Z**(1.0/3.0)
        ETA = SQRT(1.0-R(L)*R(L))
      DO 1 N=1,NMAX
    1 PP(N,L) = P(2*N-2,ETA)
c
      DO 2 N=1,NMAX
    2 PM(N) = - 0.6366198/P(2*N-2,0.0)**2
      EX = 1.0/3.0
      C2 = 1.11984652
      READ 103, IH,IHH,IM
```

```
      IHP1 = IH + 1
      READ 108, (X(I),I=IHP1,IM)
      READ 108, (W(I),I=IHP1,IM)
c
      DO 34 I=1,IM
        IF (I-IH)  31,31,32
   31   IR = IM - I + 1
        X(I) = 0.5 - 0.5*X(IR)
        W(I) = W(IR)
        GO TO 33
   32   X(I) = 0.5 + 0.5*X(I)
   33   XX = SQRT(1.0-X(I)**2)
      DO 34 N=1,NMAX
   34 PG(N,I) = P(2*N-2,XX)
c
    3 READ 104, C(1),AN,TPLUS,ALPHA,BETA,GAMMA,EXCH,DAMP
      C1 = C2*(1.0-C(1))
      IF (C(1))  4,4,5
    4 STOP
    5 JCOUNT = 0
      TAF = 1.0
c
c     Test for Tafel approximation
c
      IF (EXCH-400)  7,7,6
c
    6 TAF = 0.0
      EXCH = 1.0
    7 ETAC = ALOG(C(1)) + TPLUS*(1.0-C(1))
      CUR(1) = - (1.0-C(1))*AN*EXCH*1.11984652/C(1)**GAMMA
      ETAS = -ALOG(TAF-CUR(1))/BETA
c
      IF (CUR(1))  8,10,8
c
    8 DO 9 J=1,100
        F = TAF*EXP(ALPHA*ETAS) - EXP(-BETA*ETAS)
        FP = TAF*ALPHA*EXP(ALPHA*ETAS) + BETA*EXP(-BETA*ETAS)
        IF (ABS(CUR(1)-F) - 0.0000001*ABS(CUR(1))) 10,10,9
    9 ETAS = ETAS + (CUR(1)-F)/FP
c
   10 CUR(1) = 1.11984652*AN*(1.0-C(1))
      DO 11 L=1,LMAX
      C(L) = C(1)
   11 E(L) = ETAC + ETAS
      B(1) = 0.0
      PRINT 105, ALPHA,BETA,GAMMA,EXCH
      BOLD = B(1)
      JCOUNT = JCOUNT + 1
      j=1
      CALL THETA(x1,dx1,x2,dx2,x3,x1o,dx1o,x2o,dx2o,x3o,denom,j)
c
      DO 16 I=1,IM
```

```
      LI = X(I)**3/DZ + 1
   16 CUG(I)= CUR(LI)+(CUR(LI+1)-CUR(LI))*(X(I)**2-R(LI)**2)/(R(LI+1)**2
     1-R(LI)**2)
c
      V = E(1)
      DO 15 N=1,NMAX
        B(N) = 0.0
        DO 14 I=1,IM
   14   B(N) = B(N) + CUG(I)*X(I)*PG(N,I)*W(I)
        B(N) = 0.5*B(N)*(4*N-3)/PM(N)
   15 V= V + B(N)*PP(N,1)
c
   12   bold = b(1)
c
        do 310 n=1,nmax
          xarr(n,jcount) = b(n)
c         print 160,jcount,b(n),bold,xarr(n,jcount)
  310   continue
c
        jcount = jcount + 1
c
c       Calculate numerical derivatives
c
        do 110 i=1,nmax
  110   bo(i) = b(i)
        jdum = 0
        call bees(bo,b,jdum,x1,dx1,x2,dx2,x3,x1o,dx1o,x2o,dx2o,x3o,
     1    denom)
        x1o = x1
        dx1o = dx1
        x2o = x2
        dx2o = dx2
        x3o = x3
c
        do 111 l=1,lmax
          tho(l) = c(l)
  111   eo(l) = e(l)
c
        fact = 1.e-10
        do 120 j=1,nmax
          denom = fact*bo(j)
          bo(j) = (1. + fact)*bo(j)
          call bees(bo,db,j,x1,dx1,x2,dx2,x3,x1o,dx1o,x2o,dx2o,x3o,
     1      denom)
          x1d = x1
          dx1d = dx1
          x2d = x2
          dx2d = dx2
          x3d = x3
c
          do 133 l=1,lmax
            thd(l) = c(l)
```

```
c
  133     de(l) = e(l)
          bo(j) = bo(j)/(1.+fact)
c         print 131,j
  131     format(* dbs column*,i2)
          den = fact*bo(j)
c
          do 130 n=1,nmax
            db(n) = (db(n) - b(n))/den
            bd(n,j) = db(n)
  130     continue
c
c         print 132,(db(n),n=1,nmax)
  132     format(1x,5e12.4)
          do 134 l=1,lmax
            thd(l) = (thd(l) - tho(l))/den
  134     de(l) = (de(l) - eo(l))/den
          denom = den
          x1d = (x1d-x1o)/den
          dx1d = (dx1d-dx1o)/den
          x2d = (x2d-x2o)/den
          dx2d = (dx2d-dx2o)/den
          x3d = (x3d-x3o)/den
  120   continue
c
        do 90 i=1,nmax
   90   bin(i) = b(i)
        j = 0
      CALL THETA(x1,dx1,x2,dx2,x3,x1o,dx1o,x2o,dx2o,x3o,denom,j)
c
      DO 216 I=1,IM
      LI = X(I)**3/DZ + 1
  216 CUG(I)= CUR(LI)+(CUR(LI+1)-CUR(LI))*(X(I)**2-R(LI)**2)/(R(LI+1)**2
     1-R(LI)**2)
c
      V = E(1)
c
      DO 215 N=1,NMAX
        B(N) = 0.0
        DO 214 I=1,IM
  214   B(N) = B(N) + CUG(I)*X(I)*PG(N,I)*W(I)
  215 B(N) = 0.5*B(N)*(4*N-3)/PM(N)
c
        do 210 i=1,nmax
        do 210 j=1,mmax
  210   d(i,j) = 0.
        do 211 i=1,nmax
        do 211 j=1,nmax
          xi = 0.
          if(i.eq.j) xi = 1.
  211   bd(i,j) = xi - bd(i,j)
        do 112 i=1,nmax
```

```
  112   d(i,1) = b(i) - bin(i)
c
        call matinv(nmax,mmax,determ,bd,d)
        if(determ.eq.0.) print 116
  116   format(27h zero determinant in matinv)
c
        do 117 i=1,nmax
  117   b(i) = b(i) + d(i,1)
c
        do 320 n=1,nmax
  320   xarr(n,jcount) =b (n)
c
      JERR= 1
c
      IF (JCOUNT-50)  19,19,20
c
   19 crit=(ABS(B(1)-BOLD) - 0.000001*ABS(B(1)))
c
        do 330 n=1,nmax
c       print 160,jcount,b(n),bold,xarr(n,jcount)
  160   format(* jcount=*,i2,* b(n)=*,1pe11.4,* bold=*,1pe11.4,* xarr=*,
     1    1pe11.4)
  330   continue
c
      IF (ABS(B(1)-BOLD) - 1.e-12*ABS(B(1)))  21,21,12
c     IF (ABS(B(1)-BOLD) - 0.000001*ABS(B(1)))  21,21,12
   20 PRINT 101, JERR
   21 AVG = - B(1)/0.7853982/AN/1.11984652
c
c       Test errors for quadratic convergence
c
        do 340 n=1,nmax
          ans(n) = b(n)
  340   continue
c
        itfin = jcount
        do 350 n=1,nmax
          print 70,n,ans(n),itfin
   70     format(* n=*,i2,* answer=*,1pe17.10,* itfin=*,i2)
  350   continue
c
        print 59
   59   format(* it      err(n,it)*)
        do 30 n=1,nmax
          print 61,n
  61      format(* n=*,i2)
        do 30 it=1,itfin
          err(n,it) = xarr(n,it) - ans(n)
          print 60,it,err(n,it)
   60     format(1x,i3,5x,1pe15.8)
   30   continue
c
```

```
      print 91
   91 format(*  it          ord(it)*)
c
      itfm1 = itfin - 1
      do 35 n=1,nmax
        print 61,n
      do 35 it=1,itfm1
        ord(n,it) = 0.
        if(err(n,it).eq.0. .or. err(n,it+1).eq.0.) go to 36
        ord(n,it) = alog(abs(alog(abs(err(n,it+1)/err(n,it)))))
   36   print 60,it,ord(n,it)
   35 continue
c
      itfm2 = itfin - 2
c
      do 37 n=1,nmax
        print 61,n
      do 37 it=1,itfm2
        slope = ord(n,it+1) - ord(n,it)
        if(ord(n,it+1).eq.0.) go to 37
        print 80,it,slope,exp(slope)
   80   format(1x,i3,5x,1pe11.4,5x,e11.4)
   37 continue
c
     DO 22 L=1,LMAX
   22 CUR(L) = CUR(L)/AN/1.11984652
     RAT1 = CUR(1)/AVG
     RAT2 = (V-E(1))/B(1)
     PRINT 102, AN,TPLUS,V,AVG,(R(J),C(J),CUR(J),E(J),J=1,LMAX)
     PRINT 107, (B(I),I=1,NMAX),TAF,RAT2,RAT1
     PRINT 103, JCOUNT
     GO TO 3
     END
c
c
     SUBROUTINE THETA(x1,dx1,x2,dx2,x3,x1o,dx1o,x2o,dx2o,x3o,denom,
    1  jdum)
c
C    SUBPROGRAM FOR CALCULATING CONCENTRATION by equating Fick's-law
c    and Faraday's-law fluxes
c    (X1-X2)/PN + (C1+X3))*EXCH = 0
c
     DIMENSION E(201),CUR(201),A(200),B(200),TH(201)
c
     COMMON/a/ E,CUR,TH,NZT1,T,PN,EXCH,AL,BE,GAM,TAF,A,B,C1,C2,EX
c
  101 FORMAT (17H0NOT CONVERGED AT,I4)
c
     NZT= NZT1-1
     DEVM = 1.e-10
c    DEVM = 0.0001
     DZ = 1.0/NZT
```

```
      S = TH(1)
c
      DO 60 NZ = 2,NZT1
      Z = (NZ - 1)*DZ
      SUM = 0.0
      IF (NZ .LE. 2) GO TO 42
c
C     CALC. SUM(TH(J)*A(K))
c
      DO 40 J=3,NZ
        K = NZ - J + 1
   40 SUM = SUM + TH(J-1)*A(K)
   42 ETA = E(NZ)
      NJ = NZ - 1
c
      DO 56 N=1,20
        X1 = TAF*S**(GAM - AL)*EXP(AL*ETA)*EXP(AL*T*(S - 1.0))
        DX1 = X1*((GAM - AL)/S + AL*T)
        X2 = S**(GAM + BE)*EXP(-BE*ETA)*EXP(BE*T*(1.0 - S))
        DX2 = X2*((GAM + BE)/S - BE*T)
        C3 = 1.50*C2*Z**EX/DZ**(1./3.)
        X3 = C3*(TH( 1)*B(NJ) + SUM - S)
        DTH = S - ((X1-X2)/PN + (C1+X3)*EXCH)/((DX1-DX2)/PN - C3*EXCH)
        CUR(NZ) = PN*(C1+X3)
        if(jdum.ne.2 .or. nz.ne.2) go to 44
        x1s = x1
        dx1s = dx1
        x2s = x2
        dx2s = dx2
        x3s = x3
        x1d = (x1-x1o)/denom
        dx1d = (dx1-dx1o)/denom
        x2d = (x2-x2o)/denom
        dx2d = (dx2-dx2o)/denom
        x3d = (x3-x3o)/denom
   44   continue
        IF (ABS(S-DTH) - DEVM*ABS(DTH))  60,60,56
   56 S = DTH
c
      PRINT 101, NZ
   60 TH(NZ) = DTH
c
      RETURN
      END
c
c
      FUNCTION P(N,X)
C     CALCULATION OF LEGENDRE POLYNOMIALS
      P1= 1.0
      P2= X
      IF (N-1)  1,2,3
    1 P= P1
```

```
      RETURN
    2 P= P2
      RETURN
    3 NM1= N - 1
      DO 4 NU=1,NM1
      P=(X*FLOAT (2*NU+1)*P2-FLOAT (NU)*P1)/FLOAT (NU+1)
      P1= P2
    4 P2= P
      RETURN
      END
c
c
      subroutine bees(bo,b,j,x1s,dx1s,x2s,dx2s,x3s,x1o,dx1o,x2o,dx2o,
     1  x3o,denom)
c
      DIMENSION PM(21),B(21),C(201),CUR(201),E(201),PP(21,201),R(201),AA
     1(200),BB(200),X(40),W(40),CUG(40),PG(21,40),bo(21)
c
      COMMON/a/ E,CUR,C,LMAX,TPLUS,AN,EXCH,ALPHA,BETA,GAMMA,TAF,AA,BB,
     1C1,C2,EX
      common/b/nmax,pp,im,x,dz,r,pg,w,pm,v
c
      v=e(1)
c
      do 19 n=1,nmax
   19 v = v + bo(n)*pp(n,1)
c
      do 18 l=2,lmax
        phi = v
        do 17 n=1,nmax
   17   phi = phi - bo(n)*pp(n,l)
   18 e(l) = phi
c
      CALL THETA(x1s,dx1s;x2s,dx2s,x3s,x1o,dx1o,x2o,dx2o,x3o,denom,j)
c
      DO 16 I=1,IM
      LI = X(I)**3/DZ + 1
   16 CUG(I)= CUR(LI)+(CUR(LI+1)-CUR(LI))*(X(I)**2-R(LI)**2)/(R(LI+1)**2
     1-R(LI)**2)
c
      DO 15 N=1,NMAX
        B(N) = 0.0
        DO 14 I=1,IM
   14   B(N) = B(N) + CUG(I)*X(I)*PG(N,I)*W(I)
   15 B(N) = 0.5*B(N)*(4*N-3)/PM(N)
c
      return
      end
c
c
      SUBROUTINE MATINV(N,M,DETERM,b,d)
c
```

```
c       This subroutine solves a matrix equation of the form BX=D.
c       The answer X is put into the first column of D.  Note that
c       D should be dimensioned nxm, where m=2n+1 and that all
c       columns of D (except the first) should be initiallized with
c       zeros
c
c       Input variables
c               n = dimension of square b matrix
c               m = number of columns in d matrix
c               b = matrix in equation bx=d
c               d : first column contains vector d in bx=d
c
c       Output variables
c               d : first column contains vector x in bx=d
c               determ = 0. if matrix has zero determinant
c
      DIMENSION ID(21),b(21,21),d(21,43)
c
      DETERM=1.0
      DO 1 I=1,N
    1 ID(I)=0
      DO 18 NN=1,N
      BMAX=1.1
      DO 6 I=1,N
      IF(ID(I).NE.0) GOTO 6
      BNEXT=0.0
      BTRY=0.0
      DO 5 J=1,N
      IF(ID(J).NE.0) GOTO 5
      IF(ABS(B(I,J)).LE.BNEXT) GOTO 5
      BNEXT=ABS(B(I,J))
      IF(BNEXT.LE.BTRY) GOTO 5
      BNEXT=BTRY
      BTRY=ABS(B(I,J))
      JC=J
    5 CONTINUE
      IF(BNEXT.GE.BMAX*BTRY) GOTO 6
      BMAX=BNEXT/BTRY
      IROW=I
      JCOL=JC
    6 CONTINUE
      IF(ID(JC).EQ.0) GOTO 8
      DETERM=0.0
      RETURN
    8 ID(JCOL)=1
      IF(JCOL.EQ.IROW) GOTO 12
      DO 10 J=1,N
      SAVE=B(IROW,J)
      B(IROW,J)=B(JCOL,J)
   10 B(JCOL,J)=SAVE
      DO 11 K=1,M
      SAVE=D(IROW,K)
```

```
      D(IROW,K)=D(JCOL,K)
   11 D(JCOL,K)=SAVE
   12 F=1.0/B(JCOL,JCOL)
      DO 13 J=1,N
   13 B(JCOL,J)=B(JCOL,J)*F
      DO 14 K=1,M
   14 D(JCOL,K)=D(JCOL,K)*F
      DO 18 I=1,N
      IF(I.EQ.JCOL) GO TO 18
      F=B(I,JCOL)
      DO 16 J=1,N
   16 B(I,J)=B(I,J)-F*B(JCOL,J)
      DO 17 K=1,M
   17 D(I,K)=D(I,K)-F*D(JCOL,K)
   18 CONTINUE
      RETURN
      END
c
c
~789
   41  21
   20  10  40
3877241751-21160840707-11926975807-12681521850-13419940908-14137792044-1
4830758017-15494671251-16125538897-16719566846-17273182552-17783056514-1
8246122308-18659595032-19020988070-19328128083-19579168192-19772599500-1
9907262387-19982377097-1
7750594798-27703981816-27611036190-27472316906-27288658240-27061164739-2
1791204582-26480401346-26130624249-25743976910-25322784698-24869580764-2
4387090819-23878216797-23346019528-22793700698-22224584919-21642105838-2
1049828453-24521277099-3
 50000-2 78800+1 36300-1 15000-0 50000-1 42000-1 89600-1 10000-0

 20000-1 78800+1 36300-1 15000-0 50000-1 42000-1 89600-1 80000-1
 30000-1 78800+1 36300-1 15000-0 50000-1 42000-1 89600-1 60000-1
 40000-1 78800+1 36300-1 15000-0 50000-1 42000-1 89600-1 40000-1
 50000-1 78800+1 36300-1 15000-0 50000-1 42000-1 89600-1 30000-1
 60000-1 78800+1 36300-1 15000-0 50000-1 42000-1 89600-1 30000-1
 70000-1 78800+1 36300-1 15000-0 50000-1 42000-1 89600-1 30000-1
 80000-1 78800+1 36300-1 15000-0 50000-1 42000-1 89600-1 30000-1
 90000-1 78800+1 36300-1 15000-0 50000-1 42000-1 89600-1 30000-1
 95000-1 78800+1 36300-1 15000-0 50000-1 42000-1 89600-1 80000-1
 80000-1 20000+1 50000-1 50000-1 50000-1 50000-1 50000+2 70000-1
 80000-1 50000+1 50000-1 50000-1 50000-1 50000-1 50000+2 50000-1
 20000-1 50000-0 50000-1 50000-1 50000-1 50000-1 50000+2 80000-1
 50000-1 10000+2 50000-1 50000-1 50000-1 50000-1 50000+2 30000-1
 50000-1 10000+1 50000-1 50000-1 50000-1 50000-1 50000+2 70000-1
 50000-1 20000+1 50000-1 50000-1 50000-1 50000-1 50000+2 60000-1
```

E.3. Program CURDE

Description:

Program CURDE is Newman's program CURD rewritten for iteration on the overpotentials by a Newton-Raphson method instead of successive substitution with damping. The program solves for the current, concentration, and potential distribution on a rotating-disk electrode below the limiting current (see John Newman, "Current Distribution on a Rotating Disk below the Limiting Current," *Journal of the Electrochemical Society*, **113** (1966), 1235-1241 and Ralph Edward White, *Simultaneous Reactions on a Rotating Disk Electrode*, dissertation, University of California, Berkeley (March, 1977).

Input Variables

LMAX = number of radial mesh points

NMAX = number of B coefficients

IH = half the number of Gaussian quadrature points

IHH : not used

IM = total number of Gaussian quadrature points

X(I) = Gaussian quadrature roots (in the interval 0 to 1)

W(I) = Gaussian quadrature weights

C(1) = c/c_∞ at center of disk (input C(1) = 0 to stop running cases)

AN = dimensionless limiting current = N

TPLUS = transference number

ALPHA = anodic transfer coefficient

BETA = cathodic transfer coefficient

GAMMA = exponent for concentration dependence of exchange current density

EXCH = dimensionless exchange current density = J

DAMP = damping factor (not used)

Output Variables

JCOUNT = iteration number

BOLD = B(1) at previous iteration

CRIT = local error = | B(1) - BOLD |

ERR = 1 if too many iterations, 0 otherwise

ANS(N) = converged B(n)

ITFIN = number of iterations required for convergence

ERR(N,IT) = error in B(n) at iteration IT = e_{IT}

$$\text{ORD(N,IT)} = \ln \left| \ln \left| \frac{e_{IT+1}}{e_{IT}} \right| \right|$$

SLOPE = slope of ORD *vs.* IT

V = disk potential

AVG = $\delta/N = i_{avg}/i_{lim}$

R(J) = radial position

C(J) = surface concentration c/c_∞

CUR(J) = dimensionless current density, $i/<i>$

E(J) = total overpotential = $V - \Phi_o$

B(I) = B coefficients

TAF = 0 if Tafel approximation is used, 1 if not

RAT2 = (V - E(1))/B(1), proportional to effective resistance

RAT1 = CUR(1)/(AVG*N*$\theta_o'(0)$), ratio of current at center to average current

JCOUNT = iterations required

```
      PROGRAM CURDE(INPUT,OUTPUT)
C     CURRENT DISTRIBUTION ON A ROTATING DISK WITH AN INTEGRAL EQUATION
C     FOR THE DIFFUSION LAYER
c
c     This is program curd rewritten for iteration on overpotentials
c     by a Newton-Raphson method
c
      DIMENSION PM(21),B(21),C(201),CUR(201),E(201),PP(21,201),R(201),AA
     1(200),BB(200),X(40),W(40),CUG(40),PG(21,40) ,ed(41,41),d(41,83),
     1ein(41)
c
      COMMON/a/ E,CUR,C,LMAX,TPLUS,AN,EXCH,ALPHA,BETA,GAMMA,TAF,AA,BB,C1
     1,C2,EX
      common/b/nmax,pp,im,x,dz,r,pg,w,pm
c
  101 FORMAT (6H ERROR,I4)
  102 FORMAT (3H N=,F10.4,10H ,  TPLUS=,F8.4,6H ,  V=,F10.5,8H ,  AVG=,
     1F10.6/41H0      R          C          CUR        ETA/(4F11.5))
  103 FORMAT (3I4)
  104 FORMAT (9E8.4)
  105 FORMAT (7H1ALPHA=,F8.4,9H ,  BETA=,F8.4,10H ,  GAMMA=,F8.4,9H ,  E
     1XCH=,F8.4)
  107 FORMAT (2H0B,F9.5,5F11.5/(6F11.5))
  108 FORMAT (6E12.9)
c
      READ 103, LMAX,NMAX
      mmax = 2*lmax+1
      EX = 2.0/3.0
c
      DO 29 L=1,LMAX
        A= L
        AA(L) = 2.0*A**EX - (A+1.0)**EX - (A-1.0)**EX
   29 BB(L) = A**EX - (A-1.0)**EX
c
      DZ= 1.0/(LMAX-1)
c
      DO 1 L=1,LMAX
        Z = (L-1)*DZ
        R(L) = Z**(1.0/3.0)
        ETA = SQRT(1.0-R(L)*R(L))
      DO 1 N=1,NMAX
    1 PP(N,L)= P(2*N-2,ETA)
c
      DO 2 N=1,NMAX
    2 PM(N)= - 0.6366198/P(2*N-2,0.0)**2
c
      EX = 1.0/3.0
```

```
      C2 = 1.11984652
      READ 103, IH,IHH,IM
      IHP1 = IH + 1
      READ 108, (X(I),I=IHP1,IM)
      READ 108, (W(I),I=IHP1,IM)
c
      DO 34 I=1,IM
        IF (I-IH)  31,31,32
   31   IR = IM - I + 1
        X(I) = 0.5 - 0.5*X(IR)
        W(I) = W(IR)
        GO TO 33
   32   X(I) = 0.5 + 0.5*X(I)
   33   XX = SQRT(1.0-X(I)**2)
      DO 34 N=1,NMAX
   34 PG(N,I) = P(2*N-2,XX)
c
    3 READ 104, C(1),AN,TPLUS,ALPHA,BETA,GAMMA,EXCH,DAMP
      C1 = C2*(1.0-C(1))
      IF (C(1))  4,4,5
    4 STOP
    5 JCOUNT = 0
      TAF = 1.0
c
c     Test for Tafel approximation
c
      IF (EXCH-400)  7,7,6
c
    6 TAF = 0.0
      EXCH = 1.0
    7 ETAC = ALOG(C(1)) + TPLUS*(1.0-C(1))
      CUR(1) = - (1.0-C(1))*AN*EXCH*1.11984652/C(1)**GAMMA
      ETAS = -ALOG(TAF-CUR(1))/BETA
c
      IF (CUR(1))  8,10,8
c
    8 DO 9 J=1,100
        F = TAF*EXP(ALPHA*ETAS) - EXP(-BETA*ETAS)
        FP = TAF*ALPHA*EXP(ALPHA*ETAS) + BETA*EXP(-BETA*ETAS)
        IF (ABS(CUR(1)-F) - 0.0000001*ABS(CUR(1))) 10,10,9
    9 ETAS = ETAS + (CUR(1)-F)/FP
   10 CUR(1) = 1.11984652*AN*(1.0-C(1))
c
      DO 11 L=1,LMAX
        C(L) = C(1)
   11 E(L) = ETAC + ETAS
c
      B(1) = 0.0
```

```
      PRINT 105, ALPHA,BETA,GAMMA,EXCH
   12 BOLD = B(1)
      JCOUNT = JCOUNT + 1
c
      do 90 l=1,lmax
   90 ein(l) = e(l)
c
      call theta
c
      DO 16 iI=1,IM
        LI = X(iI)**3/DZ + 1
   16 CUG(iI)= CUR(LI)+(CUR(LI+1)-CUR(LI))*(X(iI)**2-R(LI)**2)/
     1(R(LI+1)**2-R(LI)**2)
c
      V = E(1)
      DO 15 N=1,NMAX
        B(N)= 0.0
        DO 14 iI=1,IM
   14   B(N) = B(N) + CUG(iI)*X(iI)*PG(N,iI)*W(iI)
        B(N) = 0.5*B(N)*(4*N-3)/PM(N)
   15 V = V + B(N)*PP(N,1)
c
c       calculate Jacobian
c
        do 100 l=1,lmax
  100   call deriv(l,ed,b,v)
c
        do 110 i=1,lmax
        do 110 j=1,mmax
  110   d(i,j) = 0.
c
        do 111 i=1,lmax
        do 111 j=1,lmax
          xi = 0.
          if(i.eq.j) xi = 1.0
  111   ed(i,j) = xi - ed(i,j)
c
        do 112 i=1,lmax
  112   d(i,1) = e(i) - ein(i)
c
        call matinv(lmax,mmax,determ,ed,d)
        if(determ.eq.0.) print 116
  116   format(27h zero determinant in matinv)
c
c       update
c
        do 117 i=1,lmax
  117   e(i) = ein(i) + d(i,1)
```

```
c
      JERR = 1
      IF (JCOUNT-50)  19,19,20
   19 crit=(ABS(B(1)-BOLD))
      print 160,jcount,b(1),bold,crit
  160 format(* jcount=*,i2,* b(1)=*,1pe11.4,* bold=*,1pe11.4,* crit=*,
     1  1pe11.4)
      IF (ABS(B(1)-BOLD) - 1.e-09*ABS(B(1)))  21,21,12
   20 PRINT 101, JERR
   21 AVG= - B(1)/0.7853982/AN/1.11984652
      DO 22 L=1,LMAX
   22 CUR(L)=   CUR(L)/AN/1.11984652
      RAT1= CUR(1)/AVG
      RAT2= (V-E(1))/B(1)
      PRINT 102, AN,TPLUS,V,AVG,(R(J),C(J),CUR(J),E(J),J=1,LMAX)
      PRINT 107, (B(I),I=1,NMAX),TAF,RAT2,RAT1
      PRINT 103, JCOUNT
      GO TO 3
      END
c
      SUBROUTINE THETAp(de,je,dcur,thd)
c
C     SUBPROGRAM FOR CALCULATING CONCENTRATION derivatives
c     This subroutine is the derivative of subroutine theta
c     with respect to E(je)
c
      DIMENSION E(201),CUR(201),A(200),B(200),TH(201),thd(201),de(201) ,
     1dcur(201)
c
      COMMON/a/ E,CUR,TH,NZT1,T,PN,EXCH,AL,BE,GAM,TAF,A,B,C1,C2,EX
c
  101 FORMAT (17H0NOT CONVERGED AT,I4)
c
      NZT = NZT1-1
      DEVM = 0.0001
      DZ = 1.0/NZT
      S = TH(1)
      ds=0.
      thd(1)=0.
c
      DO 60 NZ = 2,NZT1
      ds=0.
      Z = (NZ - 1)*DZ
      SUM = 0.0
      dsum = 0.0
      IF (NZ .LE. 2) GO TO 42
c
C     CALC. SUM(TH(J)*A(K))
```

```
c
      DO 40 J=3,NZ
        K = NZ - J + 1
        dsum = dsum + thd(j-1)*a(k)
   40 SUM = SUM + TH(J-1)*A(K)
   42 ETA = E(NZ)
c
c     derivative of e(nz) with respect to e(je)
c
      deta = de(nz)
c
      S = th(nz)
c
c     S is not updated because the thetas were calculated in subroutine
c     theta
c
      NJ = NZ - 1
      X1 = TAF*S**(GAM - AL)*EXP(AL*ETA)*EXP(AL*T*(S - 1.0))
      DX1 = X1*((GAM - AL)/S + AL*T)
      x1d = al*x1*deta
      X2 = S**(GAM + BE)*EXP(-BE*ETA)*EXP(BE*T*(1.0 - S))
      DX2 = X2*((GAM + BE)/S - BE*T)
      x2d = -be*x2*deta
      C3 = 1.50*C2*Z**EX/DZ**(1./3.)
      X3 = C3*(TH( 1)*B(NJ) + SUM - S)
      dx3 = c3*dsum
      dthd = - ((x1d-x2d)/pn + dx3*exch)/((dx1-dx2)/pn - c3*exch)
      dCUR(NZ) = PN*c3*(dsum-dthd)
      x1dn = x1d + dx1*dthd
      x2dn = x2d + dx2*dthd
      dx3n = dx3 - c3*dthd
      thd(nz) = dthd
   60 continue
c
      RETURN
      END
c
c
      FUNCTION P(N,X)
C     CALCULATION OF LEGENDRE POLYNOMIALS
      P1= 1.0
      P2= X
      IF (N-1)  1,2,3
    1 P= P1
      RETURN
    2 P= P2
      RETURN
    3 NM1= N - 1
```

```
      DO 4 NU=1,NM1
      P=(X*FLOAT (2*NU+1)*P2-FLOAT (NU)*P1)/FLOAT (NU+1)
      P1= P2
    4 P2= P
      RETURN
      END
c
c
      SUBROUTINE MATINV(N,M,DETERM,b,d)
c
c       This subroutine solves a matrix equation of the form BX=D.
c       The answer X is put into the first column of D.  Note that
c       D should be dimensioned nxm, where m=2n+1 and that all
c       columns of D (except the first) should be initiallized with
c       zeros
c
c       Input variables
c               n = dimension of square b matrix
c               m = number of columns in d matrix
c               b = matrix in equation bx=d
c               d : first column contains vector d in bx=d
c
c       Output variables
c               d : first column contains vector x in bx=d
c               determ = 0. if matrix has zero determinant
c
      DIMENSION ID(41),b(41,41),d(41,83)
c
      DETERM=1.0
      DO 1 I=1,N
   1  ID(I)=0
      DO 18 NN=1,N
      BMAX=1.1
      DO 6 I=1,N
      IF(ID(I).NE.0) GOTO 6
      BNEXT=0.0
      BTRY=0.0
      DO 5 J=1,N
      IF(ID(J).NE.0) GOTO 5
      IF(ABS(B(I,J)).LE.BNEXT) GOTO 5
      BNEXT=ABS(B(I,J))
      IF(BNEXT.LE.BTRY) GOTO 5
      BNEXT=BTRY
      BTRY=ABS(B(I,J))
      JC=J
   5  CONTINUE
      IF(BNEXT.GE.BMAX*BTRY) GOTO 6
      BMAX=BNEXT/BTRY
```

```
      IROW=I
      JCOL=JC
    6 CONTINUE
      IF(ID(JC).EQ.0) GOTO 8
      DETERM=0.0
      RETURN
    8 ID(JCOL)=1
      IF(JCOL.EQ.IROW) GOTO 12
      DO 10 J=1,N
      SAVE=B(IROW,J)
      B(IROW,J)=B(JCOL,J)
   10 B(JCOL,J)=SAVE
      DO 11 K=1,M
      SAVE=D(IROW,K)
      D(IROW,K)=D(JCOL,K)
   11 D(JCOL,K)=SAVE
   12 F=1.0/B(JCOL,JCOL)
      DO 13 J=1,N
   13 B(JCOL,J)=B(JCOL,J)*F
      DO 14 K=1,M
   14 D(JCOL,K)=D(JCOL,K)*F
      DO 18 I=1,N
      IF(I.EQ.JCOL) GO TO 18
      F=B(I,JCOL)
      DO 16 J=1,N
   16 B(I,J)=B(I,J)-F*B(JCOL,J)
      DO 17 K=1,M
   17 D(I,K)=D(I,K)-F*D(JCOL,K)
   18 CONTINUE
      RETURN
      END
c
c
      subroutine deriv(i,ed,b,v)
c
c     This subroutine calculates the derivatives of the calculated
c     overpotentials with respect to the input overpotentials, and
c     it updates the overpotentials the last time it is called.
c
      DIMENSION PM(21),B(21),C(201),CUR(201),E(201),PP(21,201),R(201),AA
     1(200),BB(200),X(40),W(40),CUG(40),PG(21,40) ,ed(41,41),d(41,83),
     lein(41),bd(41),dcur(201),de(201),thd(201),dcug(40)
c
      COMMON/a/ E,CUR,C,LMAX,TPLUS,AN,EXCH,ALPHA,BETA,GAMMA,TAF,AA,BB,C1
     1,C2,EX
      common/b/nmax,pp,im,x,dz,r,pg,w,pm
c
c     de is derivative of overpotentials with respect to e(i)
```

```
c
      do 1 L=1,lmax
    1 de(L) = 0.
      de(i) = 1.
      dcur(1) = 0.
      CALL THETAp(de,i,dcur,thd)
c
      DO 16 iI=1,IM
      LI= X(iI)**3/DZ + 1
   16 dCUG(iI) = dCUR(LI)+(dCUR(LI+1)-dCUR(LI))*(X(iI)**2-R(LI)**2)/
     1(R(LI+1)**2-R(LI)**2)
c
      dv = 0.
c
      DO 15 N=1,NMAX
        Bd(N) = 0.0
        DO 14 iI=1,IM
   14   Bd(N) = Bd(N) + dCUG(iI)*X(iI)*PG(N,iI)*W(iI)
        Bd(N) = 0.5*Bd(N)*(4*N-3)/PM(N)
   15 dV = dV + Bd(N)*PP(N,1)
c
      ed(1,i) = 0.
c
      DO 18 L=2,LMAX
        PHI = V
        dPHI = dV
        DO 17 N=1,NMAX
          dPHI = dPHI - Bd(N)*PP(N,L)
   17   PHI = PHI - B(N)*PP(N,L)
        Ed(L,i) = dphi
   18 if(i.eq.lmax) E(L) = phi
c
      return
      end
c
c
      SUBROUTINE THETA
c
C     SUBPROGRAM FOR CALCULATING CONCENTRATION by equating Fick's-law
c     and Faraday's-law fluxes
c     (X1-X2)/PN + (C1+X3)*EXCH = 0
c
      DIMENSION E(201),CUR(201),A(200),B(200),TH(201)
c
      COMMON/a/ E,CUR,TH,NZT1,T,PN,EXCH,AL,BE,GAM,TAF,A,B,C1,C2,EX
c
  101 FORMAT (17HONOT CONVERGED AT,I4)
c
```

```
      NZT = NZT1-1
      DEVM = 0.0001
      DZ = 1.0/NZT
      S = TH(1)
      DO 60  NZ = 2,NZT1
      Z = (NZ - 1)*DZ
      SUM = 0.0
      IF (NZ .LE. 2) GO TO 42
c
C     CALC. SUM(TH(J)*A(K))
c
      DO 40 J=3,NZ
      K = NZ - J + 1
   40 SUM = SUM + TH(J-1)*A(K)
   42 ETA = E(NZ)
      NJ = NZ - 1
c
      DO 56 N=1,20
      X1 = TAF*S**(GAM - AL)*EXP(AL*ETA)*EXP(AL*T*(S - 1.0))
      DX1 = X1*((GAM - AL)/S + AL*T)
      X2 = S**(GAM + BE)*EXP(-BE*ETA)*EXP(BE*T*(1.0 - S))
      DX2 = X2*((GAM + BE)/S - PE*T)
      C3 = 1.50*C2*Z**EX/DZ**(1./3.)
      X3 = C3*(TH( 1)*B(NJ) + SUM - S)
      DTH = S - ((X1-X2)/PN + (C1+X3)*EXCH)/((DX1-DX2)/PN - C3*EXCH)
      CUR(NZ) = PN*(C1+X3)
      IF (ABS(S-DTH) - DEVM*ABS(DTH))  60,60,56
   56 S = DTH
c
      PRINT 101, NZ
   60 TH(NZ) = DTH
      RETURN
      END
-789
  41  21
  20  10  40
3877241751-21160840707-11926975807-12681521850-13419940908-14137792044-1
4830758017-15494671251-16125538897-16719566846-17273182552-17783056514-1
8246122308-18659595032-19020988070-19328128083-19579168192-19772599500-1
9907262387-19982377097-1
7750594798-27703981816-27611036190-27472316906-27288658240-27061164739-2
1791204582-26480401346-26130624249-25743976910-25322784698-24869580764-2
4387090819-23878216797-23346019528-22793700698-22224584919-21642105838-2
1049828453-24521277099-3
 70000-1 78800+1 36300-1 15000-0 50000-1 42000-1 89600-1 30000-1

 50000-2 78800+1 36300-1 15000-0 50000-1 42000-1 89600-1 10000-0
 20000-1 78800+1 36300-1 15000-0 50000-1 42000-1 89600-1 80000-1
```

```
30000-1 78800+1 36300-1 15000-0 50000-1 42000-1 89600-1 60000-1
40000-1 78800+1 36300-1 15000-0 50000-1 42000-1 89600-1 40000-1
50000-1 78800+1 36300-1 15000-0 50000-1 42000-1 89600-1 30000-1
60000-1 78800+1 36300-1 15000-0 50000-1 42000-1 89600-1 30000-1
70000-1 78800+1 36300-1 15000-0 50000-1 42000-1 89600-1 30000-1
80000-1 78800+1 36300-1 15000-0 50000-1 42000-1 89600-1 30000-1
90000-1 78800+1 36300-1 15000-0 50000-1 42000-1 89600-1 30000-1
95000-1 78800+1 36300-1 15000-0 50000-1 42000-1 89600-1 80000-1
80000-1 20000+1 50000-1 50000-1 50000-1 50000-1 50000+2 70000-1
80000-1 50000+1 50000-1 50000-1 50000-1 50000-1 50000+2 50000-1
20000-1 50000-0 50000-1 50000-1 50000-1 50000-1 50000+2 80000-1
50000-1 10000+2 50000-1 50000-1 50000-1 50000-1 50000+2 30000-1
50000-1 10000+1 50000-1 50000-1 50000-1 50000-1 50000+2 70000-1
50000-1 20000+1 50000-1 50000-1 50000-1 50000-1 50000+2 60000-1
```

E.4. Program CURDEN

Description:

Program CURDEN is Newman's program CURD rewritten for iteration on the overpotentials by a Newton-Raphson method with numerical derivatives instead of successive substitution with damping. The program solves for the current, concentration, and potential distribution on a rotating-disk electrode below the limiting current (see John Newman, "Current Distribution on a Rotating Disk below the Limiting Current," *Journal of the Electrochemical Society*, **113** (1966), 1235-1241 and Ralph Edward White, *Simultaneous Reactions on a Rotating Disk Electrode*, dissertation, University of California, Berkeley (March, 1977).

Input Variables

LMAX = number of radial mesh points

NMAX = number of B coefficients

IH = half the number of Gaussian quadrature points

IHH : not used

IM = total number of Gaussian quadrature points

X(I) = Gaussian quadrature roots (in the interval 0 to 1)

W(I) = Gaussian quadrature weights

C(1) = c/c_∞ at center of disk (input C(1) = 0 to stop running cases)

AN = dimensionless limiting current = N

TPLUS = transference number

ALPHA = anodic transfer coefficient

BETA = cathodic transfer coefficient

GAMMA = exponent for concentration dependence of exchange current density

EXCH = dimensionless exchange current density = J

DAMP = damping factor (not used)

Output Variables

ERR = 1 if too many iterations, 0 otherwise

ANS(N) = converged B(n)

ITFIN = number of iterations required for convergence

ERR(N,IT) = error in B(n) at iteration IT = e_{IT}

$$\text{ORD(N,IT)} = \ln \left| \ln \left| \frac{e_{IT+1}}{e_{IT}} \right| \right|$$

SLOPE = slope of ORD *vs.* IT

V = disk potential

AVG = $\delta/N = i_{avg}/i_{lim}$

R(J) = radial position

C(J) = surface concentration c/c_∞

CUR(J) = dimensionless current density, $i/<i>$

E(J) = total overpotential = $V - \Phi_o$

B(I) = B coefficients

TAF = 0 if Tafel approximation is used, 1 if not

RAT2 = (V - E(1))/B(1), proportional to effective resistance

RAT1 = CUR(1)/(AVG*N*$\theta_o'(0)$), ratio of current at center to average current

JCOUNT = iterations required

```
c      PROGRAM CURDEN(INPUT,OUTPUT)
c
C      CURRENT DISTRIBUTION ON A ROTATING DISK WITH AN INTEGRAL EQUATION
C      FOR THE DIFFUSION LAYER
c
c      This is program curd rewritten for iteration on overpotentials
c      by a Newton-Raphson method with numerical derivatives
c
       implicit double precision (a-h,o-z)
c
       DIMENSION PM(21),B(21),C(201),CUR(201),E(201),PP(21,201),R(201),AA
      1(200),BB(200),X(40),W(40),CUG(40),PG(21,40),de(201),eo(201),
      1thd(201),tho(201),d(21,83),ein(41),ed(41,41),dde(41),bo(21),bd(21)
      1,curo(41),cursd(41),co(41),cd(41)
c
       COMMON/a/ CUR,C,LMAX,TPLUS,AN,EXCH,ALPHA,BETA,GAMMA,TAF,AA,BB,
      1C1,C2,EX
       common/b/nmax,pp,im,x,dz,r,pg,w,pm,v
c
  101 FORMAT (6H ERROR,I4)
  102 FORMAT (3H N=,F10.4,10H ,  TPLUS=,F8.4,6H ,  V=,F10.5,8H ,  AVG=,
     1F10.6/41H0      R          C          CUR        ETA/(4F11.5))
  103 FORMAT (3I4)
  104 FORMAT (9E8.4)
  105 FORMAT (7H1ALPHA=,F8.4,9H ,  BETA=,F8.4,10H ,  GAMMA=,F8.4,9H ,  E
     1XCH=,F8.4)
  107 FORMAT (2H0B,F9.5,5F11.5/(6F11.5))
  108 FORMAT (6E12.9)
  109 format(' numerical derivatives')
c
       print 109
       READ 103, LMAX,NMAX
       mmax = 2*lmax+1
       EX = 2.0d0/3.0d0
c
       DO 29 L=1,LMAX
         A = dfloat(L)
         AA(L) = 2.0*A**EX - (A+1.0)**EX - (A-1.0)**EX
   29  BB(L) = A**EX - (A-1.0)**EX
c
       DZ = 1.0d0/(LMAX-1)
c
       DO 1 L=1,LMAX
         Z = (L-1)*DZ
         R(L) = Z**(1.0/3.0)
         ETA = dSQRT(1.0-R(L)*R(L))
       DO 1 N=1,NMAX
```

```
    1 PP(N,L) = P(2*N-2,ETA)
c
      DO 2 N=1,NMAX
    2 PM(N) = - 0.6366198/P(2*N-2,0.0)**2
      EX = 1.0d0/3.0d0
      C2 = 1.11984652d0
      READ 103, IH,IHH,IM
      IHP1 = IH + 1
      READ 108, (X(I),I=IHP1,IM)
      READ 108, (W(I),I=IHP1,IM)
c
      DO 34 I=1,IM
        IF (I-IH)  31,31,32
   31   IR = IM - I + 1
        X(I) = 0.5d0 - 0.5d0*X(IR)
        W(I) = W(IR)
        GO TO 33
   32   X(I) = 0.5d0 + 0.5d0*X(I)
   33   XX = dSQRT(1.0-X(I)**2)
      DO 34 N=1,NMAX
   34 PG(N,I) = P(2*N-2,XX)
c
    3 READ 104, C(1),AN,TPLUS,ALPHA,BETA,GAMMA,EXCH,DAMP
      C1 = C2*(1.0-C(1))
      IF (C(1))  4,4,5
    4 STOP
    5 JCOUNT = 0
      TAF = 1.0d0
c
c     Test for Tafel approximation
c
      IF (EXCH-400)  7,7,6
c
    6 TAF = 0.0d0
      EXCH = 1.0d0
    7 ETAC = dLOG(C(1)) + TPLUS*(1.0-C(1))
      CUR(1) = - (1.0-C(1))*AN*EXCH*1.11984652/C(1)**GAMMA
      ETAS = -dLOG(TAF-CUR(1))/BETA
c
      IF (CUR(1))  8,10,8
c
    8 DO 9 J=1,100
        F = TAF*dEXP(ALPHA*ETAS)-dEXP(-BETA*ETAS)
        FP = TAF*ALPHA*dEXP(ALPHA*ETAS)+BETA*dEXP(-BETA*ETAS)
        IF (dABS(CUR(1)-F) - 0.0000001*dABS(CUR(1))) 10,10,9
    9 ETAS = ETAS + (CUR(1)-F)/FP
c
   10 CUR(1) = 1.11984652*AN*(1.0-C(1))
```

```
      DO 11 L=1,LMAX
        C(L) = C(1)
        E(L) = ETAC + ETAS
   11 de(1) = e(1)
      B(1) = 0.0d0
      PRINT 105, ALPHA,BETA,GAMMA,EXCH
   12 BOLD = B(1)
c
      JCOUNT = JCOUNT + 1
c
      do 110 l=1,lmax
  110 eo(l) = e(l)
      jdum = 0
      call ees(eo,e,b,jdum,denom,x1,dx1,x2,dx2,x3,s,sums)
      print 345
  345 format(' b(n)')
      print 346,(b(n),n=1,nmax)
  346 format(1x,1pe12.4,5e11.4)
      do 111 n=1,nmax
  111 bo(n) = b(n)
      do 113 l=1,lmax
        co(l) = c(l)
  113 curo(l) = cur(l)
      sumo = sums
      so = s
      vo = v
      x1o = x1
      dx1o = dx1
      x2o = x2
      dx2o = dx2
      x3o = x3
c
      do 120 j=1,lmax
        denom = 0.000001d0*eo(j)
        eo(j) = 1.000001d0*eo(j)
        call ees(eo,de,b,j,denom,x1,dx1,x2,dx2,x3,s,sums)
        if(j.ne.2) go to 335
        do 112 n=1,nmax
  112   bd(n) = b(n)
c
        do 114 l=1,lmax
          cd(l) = c(l)
  114   cursd(l) = cur(l)
c
        sumd = sums
        ds = s
        vd = v
        x1d = x1
```

```
        dx1d = dx1
        x2d = x2
        dx2d = dx2
        x3d = x3
  335   continue
        eo(j) = eo(j)/1.000001d0
c
        do 130 n=1,lmax
          dde(n) = (de(n) - e(n))/(0.000001*eo(j))
          ed(n,j) = dde(n)
          if(j.eq.2) cursd(n) = (cursd(n)-curo(n))/(0.000001*eo(j))
          if(j.eq.2) cd(n) = (cd(n)-co(n))/(0.000001*eo(j))
  130   continue
c
        if(j.ne.2) go to 120
        denom = 0.000001*eo(j)
        sumd = (sumd-sumo)/denom
        ds = (ds-so)/denom
        vd = (vd-vo)/denom
        x1d = (x1d-x1o)/denom
        dx1d = (dx1d-dx1o)/denom
        x2d = (x2d-x2o)/denom
        dx2d = (dx2d-dx2o)/denom
        x3d = (x3d-x3o)/denom
  120 continue
c
c     stop
      JERR = 1
c
      IF (JCOUNT-100)  19,19,20
c
   19 crit = (dABS(B(1)-BOLD) - 0.000001*dABS(B(1)))
      print 160,jcount,b(1),bold,crit
  160 format(' jcount=',i2,' b(1)=',1pe11.4,' bold=',1pe11.4,' crit=',
     1  1pe11.4)
      IF (dABS(B(1)-BOLD) - 0.000001*dABS(B(1)))  21,21,12
   20 PRINT 101, JERR
   21 AVG = - B(1)/0.7853982/AN/1.11984652
      DO 22 L=1,LMAX
   22 CUR(L) = CUR(L)/AN/1.11984652
      RAT1 = CUR(1)/AVG
      RAT2 = (V-E(1))/B(1)
      PRINT 102, AN,TPLUS,V,AVG,(R(J),C(J),CUR(J),E(J),J=1,LMAX)
      PRINT 107, (B(I),I=1,NMAX),TAF,RAT2,RAT1
      PRINT 103, JCOUNT
      GO TO 3
      END
c
```

```
c
      SUBROUTINE THETA(e,x1s,dx1s,x2s,dx2s,x3s,
     1  jdum,ss,sums)
c
C     SUBPROGRAM FOR CALCULATING CONCENTRATION by equating Fick's-law
c     and Farady's-law fluxes
c     (X1-X2)/PN + (C1+X3)*EXCH = 0
c
      implicit double precision (a-h,o-z)
c
      DIMENSION E(201),CUR(201),A(200),B(200),TH(201)
c
      COMMON/a/ CUR,TH,NZT1,T,PN,EXCH,AL,BE,GAM,TAF,A,B,C1,C2,EX
c
  101 FORMAT (17HONOT CONVERGED AT,I4)
c
      NZT = NZT1-1
      DEVM = 0.000001d0
      DZ = 1.0d0/NZT
      S = TH(1)
c
      DO 60  NZ = 2,NZT1
      Z = (NZ - 1)*DZ
      SUM = 0.0d0
      IF (NZ .LE. 2) GO TO 42
c
C     CALC. SUM(TH(J)*A(K))
c
      DO 40 J=3,NZ
      K = NZ - J + 1
   40 SUM = SUM + TH(J-1)*A(K)
   42 ETA = E(NZ)
      NJ = NZ - 1
c
      DO 56 N=1,20
        X1 = TAF*S**(GAM - AL)*dEXP(AL*ETA)*dEXP(AL*T*(S - 1.0))
        DX1 = X1*((GAM - AL)/S + AL*T)
        X2 = S**(GAM + BE)*dEXP(-BE*ETA)*dEXP(BE*T*(1.0 - S))
        DX2 = X2*((GAM + BE)/S - BE*T)
        C3 = 1.50d0*C2*Z**EX/DZ**(1./3.)
        DTH = S - ((X1-X2)/PN + (C1+X3)*EXCH)/((DX1-DX2)/PN - C3*EXCH)
        X3 = C3*(TH( 1)*B(NJ) + SUM - dth)
        CUR(NZ) = PN*(C1+X3)
        if(.not.((jdum.eq.2 .or. jdum.eq.0) .and. nz.eq.4)) go to 44
        sums = sum
        ss = s
        x1s = x1
        dx1s = dx1
```

```
        x2s = x2
        dx2s = dx2
        x3s = x3
   44   continue
        IF (dABS(S-DTH) - DEVM*dABS(DTH))  60,60,56
   56 S = DTH
c
      PRINT 101, NZ
   60 TH(NZ) = DTH
c
      RETURN
      END
c
c
      FUNCTION P(N,X)
      implicit double precision (a-h,o-z)
C     CALCULATION OF LEGENDRE POLYNOMIALS
      P1= 1.0d0
      P2= X
      IF (N-1)  1,2,3
    1 P= P1
      RETURN
    2 P= P2
      RETURN
    3 NM1= N - 1
      DO 4 NU=1,NM1
      P=(X*dFLOAT (2*NU+1)*P2-dFLOAT (NU)*P1)/dFLOAT (NU+1)
      P1= P2
    4 P2= P
      RETURN
      END
c
c
      subroutine ees(eo,e,b,j,denom,x1,dx1,x2,dx2,x3,ss,sums)
c
      implicit double precision (a-h,o-z)
c
      DIMENSION PM(21),B(21),C(201),CUR(201),E(201),PP(21,201),R(201),AA
     1(200),BB(200),X(40),W(40),CUG(40),PG(21,40),eo(201)
c
      COMMON/a/ CUR,C,LMAX,TPLUS,AN,EXCH,ALPHA,BETA,GAMMA,TAF,AA,BB,
     1C1,C2,EX
      common/b/nmax,pp,im,x,dz,r,pg,w,pm,v
c
      CALL THETA(eo,x1,dx1,x2,dx2,x3,j,ss,sums)
c
      DO 16 I=1,IM
        LI= X(I)**3/DZ + 1
```

```
   16 CUG(I)= CUR(LI)+(CUR(LI+1)-CUR(LI))*(X(I)**2-R(LI)**2)/(R(LI+1)**2
     1-R(LI)**2)
c
      v = e(1)
c
      DO 15 N=1,NMAX
        B(N) = 0.0d0
        DO 14 I=1,IM
   14   B(N) = B(N) + CUG(I)*X(I)*PG(N,I)*W(I)
        B(N) = 0.5*B(N)*(4*N-3)/PM(N)
   15 v = v + b(n)*pp(n,1)
c
      do 18 l=2,lmax
        phi = v
        do 17 n=1,nmax
   17   phi = phi - b(n)*pp(n,l)
   18 e(l) = phi
c
        return
        end
```

Appendix F. Program CHANNEL

F.1. Program Description

Program CHANNEL solves for the current, concentration, and potential distribution in a channel flow cell that may have multiple reactions and interacting boundary layers. Superposition integrals are used to express the flux in terms of the surface concentrations, and the Butler-Volmer equation and Faraday's law give a second expression for the flux. With a guessed potential distribution, the surface concentrations are obtained from the flux equations by a multidimensional Newton-Raphson method. The iteration on the potentials is by successive substitution with damping. The damping factor is adjusted as the iterations proceed to attempt to speed the convergence without producing instabilities. This program does not converge well, but the restart feature is often helpful.

Input Variables:

NG = half the number of Gaussian quadrature points to be used in the integrals for the potential distribution (maximum allowable = 48)

ROOT(JXP) = roots for Gaussian quadrature (input the roots from 0 to 1)

GW(JXP) = Gaussian quadrature weights

XLAMDA(I) = eigenvalues from the asymmetric Graetz problem

AI(I) = coefficients from the asymmetric Graetz problem

NJ = number of evenly-spaced mesh points

NDEC = number of "decades" or regions with finely-spaced mesh points. Decades start from x=0 and are nested inside each other

ITSTRT = 0 if the run is not a restart, nonzero otherwise

IFINE : not used

NC : not used

IWHITE = 0 to read dimensionless parameters, nonzero to read dimensional parameters for comparison with White's results[18]

H = gap thickness (cm)

XL = electrode length (cm)

VCELL = applied cell voltage (V)

RHO0 = solvent density (kg/cm^3)

DR = diffusion coefficient of main reactant (cm^2/s)

CR = feed concentration of main reactant (mol/cm^3)

XNM = n_m = number of electrons in main reaction

SRM = s_{Rm} = stoichiometric coefficient of main reactant in main reaction

IUNIFC(I) = nonzero if uniform current is guessed for electrode I

ILIMC(I) = nonzero if limiting current is guessed for electrode I

IPRIMC(I) = nonzero if primary current is guessed for electrode I

IZEROP = nonzero for no ohmic drop

HOL = aspect ratio, h/L

ICUT = specifies size of each "decade" of finely-spaced points. A decade starts from $x = 0$ and ends at $x = 1/\text{ICUT}$

XN = N = $\frac{n_m^2 F^2 D_R c_{R,ref}}{s_{RM}^2 RT\kappa_{feed}} * (\frac{6vL^2}{hD_R})^{1/3}$ = dimensionless limiting current density†

PEHOL = $Pe\ h/L = \frac{2<v>h^2}{DL}$ = dimensionless parameter characterizing boundary layer thickness (directly proportional to Graetz number)

RHO0 = $\rho_o/c_{R,ref}$ = solvent density / reference concentration of main reactant

VCELL = $V_{cell}\frac{-n_m F}{s_{Rm}RT}$ = dimensionless cell voltage

---

SON = s_{Rm}/n_m

ITMAXO = maximum number of iterations in outer loop

ISEC = nonzero if running secondary current distribution

NSPEC = number of species

NRXN = number of reactions

NRE = index for reference-electrode reaction

XIEND = arbitrary large value of the current at the endpoints for the primary current distribution

ONE(IELEC,J) = 1.d0 if reaction J occurs on electrode IELEC (zero otherwise)

AAJ = $\alpha_{a,j}$ = anodic transfer coefficient for reaction J

ACJ = $\alpha_{c,j}$ = cathodic transfer coefficient for reaction J

XJ(J) = exchange current density for reaction J (A/cm^2)

UTHJ = standard electrode potential for reaction J (V)

NSE = number of species in solution that do not participate in electrode reactions (used to calculate conductivity)

DSE(I) = diffusion coefficient of nonparticipating species I

CSE(I) = feed concentration of nonparticipating species I

ZSE(I) = charge number of nonparticipating species I

DI = diffusion coefficient of species I (a participating species)

CREFI = reference concentration of species I (mol/cm^3)

CREI = concentration of species I in reference-electrode compartment (mol/cm^3)

CI = feed concentration of species I (mol/cm^3)

ZI = charge number of species I

GAM(I,J) = exponent for concentration dependence of reaction-J exchange current den-

---

† This interpretation holds only for thin boundary layers.

sity on species I

RATSON(I,J) $= \frac{s_{ij}}{n_j}$ = ratio of stoichiometric coefficient of species I in reaction J to number of electrons in reaction J

ASON(1,J) $= \alpha_{a,j} s_{Rm}/n_m$ = dimensionless anodic transfer coefficient for reaction J

ASON(2,J) $= \alpha_{c,j} s_{Rm}/n_m$ = dimensionless cathodic transfer coefficient for reaction J

XJ(J) $= J_j = -n_m F L i_o/(s_{Rm} R T \kappa_\infty)$ = dimensionless exchange current density for reaction J

UJTH(J) $= U_j^\theta(-n_m F)/(s_{Rm} R T)$ = dimensionless standard electrode potential for reaction J

UJTH(NRE) = dimensionless standard electrode potential of the reference-electrode reaction

DIODR(I) $= D_i/D_R$ = dimensionless diffusion coefficient of species I

CREF(I) = dimensionless reference concentration of species I

CRE(I) = dimensionless concentration of species I in the reference-electrode compartment

CIOCR(I) $= c_{i,feed}/c_{R,ref}$ = dimensionless feed concentration of species I

GAM(I,J) = exponent for concentration dependence of reaction-J exchange current density on species I

RATSON(I,J) $= \frac{s_{ij} n_m}{s_{Rm} n_j}$ = dimensionless ratio of stoichiometric coefficient of species I in reaction J to number of electrons in reaction J

Output Variables:

COEFF $= -N(16Z)^{-1/3}$, where $Z = \frac{3h^2 < v >}{8 D_R L}$, dimensionless parameter that multiplies superposition integrals

XIAV $= \dfrac{-n_m F L <i>}{s_{Rm} R T \kappa_\infty}$ = dimensionless average current density

PHISTR = dimensionless integration constant in solution to Laplace's equation for the potential

UREF(J) = dimensionless theoretical open-circuit potential for reaction J, evaluated at the reference concentrations

PORQ(IP,I,J) = p_{ij} = anodic reaction order of species I in reaction J

PORQ(IQ,I,J) = p_{ij} = cathodic reaction order of species I in reaction J

CSURF(IAN,I,JX) = dimensionless anode surface concentration of species I at mesh point JX

XIJ(IAN,J,JX) = dimensionless anodic partial current density from reaction J at mesh point JX

XI(IAN,JX) = $i_{an}(-n_m/s_{Rm})FL/(RT\kappa_\infty)$ = dimensionless anodic current density

FINE(IAN,IDEC,JX) = anode current density at mesh point JX in decade IDEC (currents at the finely-spaced mesh points)

CAN(IDEC,I,K) = dimensionless anode surface concentration at mesh point K in decade IDEC

CCATH(IDEC,I,K) = dimensionless cathode surface concentration

POMPS(IAN,JX) = $(\Phi^o_{an} - \Phi^*)(-n_m F)/(s_{Rm} RT)$ = potential calculated from Laplace's equation

F.2. Major Subroutines

F.2.1. Subroutine INTCXI

This subroutine calculates XINT0 (I, N-K+1, IDEC), which is

$$\frac{1}{\Delta x}\int_{(k-1)\Delta x}^{k\Delta x} C_{\xi}(n\Delta x - x')\,dx',$$

where C is the flux calculated from the asymmetric Graetz problem.

F.2.2. Subroutine EQFIVE

Calculates the dimensionless form of the $(\Phi^o - \Phi^*)$ given by equation 5 in Parrish and Newman[21] (equation 1-6 in this work). The singularity in the equation was eliminated by a modified method of Kantorovich and Krylov.[64] (See section 1.4.3.)

F.2.3. Subroutine PHI

Calculates the new potential distribution that is damped in the main program.

F.2.4. Subroutine IAVLP

Contains the iteration loop for $<i>$. The average current used to calculate the potential distribution from Laplace's equation must agree with the average current resulting from the kinetics and mass transfer. The iteration is by a regula-falsi method with bisection or extrapolation, as needed.

F.2.5. Subroutine PHILOOP

Contains the iteration loop for Φ^* (similar to subroutine IAVLP).

F.2.6. Subroutine NEWT

Calculates concentrations by solving the superposition-integral equation with the method of Acrivos and Chambre. Negative concentrations are reset to 10^{-6} times the previous (positive) value. Subroutine FCNS calculates the functions needed for the matrix equation.

F.2.7. Function DFDC

Calculates Jacobian for Newton-Raphson on surface concentrations.

F.2.8. Subroutine NEWTLC

Special version of subroutine NEWT for limiting current. Calculates cathode current from the mass transfer instead of the kinetics. (Calls FCNSLC.)

```
c
      implicit double precision (a-h,o-z)
      logical goback,idump
c
      dimension csurf(2,10,101),iunifc(2),ilimc(2),iprimc(2),
     1  xij(2,5,101),gam(10,5),svdist(2,101,5),old(2,101,5),
     1  fine(2,5,101),
     1  xi(2,101),ujth(5),ciocr(10),x(3),y(3),fit(3),ydum(3),
     1  p0mpsi(2,101,5),root(96),p0mpso(2,101,5),eta(2,101,5),
     1  tox(2,101),can(5,10,101),ccath(5,10,101),xitotj(2,5),
     1  toxj(2,101,5),etas(2,101)
c
      common/evprob/xlamda(3),ai(3),z,diodr(10)
      common/co/coeff,xn
      common/rxnpars/cref(10),cre(10),porq(2,10,5),ip,iq,uref(5),
     1  nspec,nrxn,one(2,5),rho0
      common/matcom/b(20,20),d(20,41)
      common/kinpars/xj(5),ratson(10,5),ason(2,5),ifor,iback
      common/cdata/cnctol,ftol,xint0(10,101,5),
     1  xint1(10,101,5),n2,nd,isec
      common/elecs/v(2),ian,icath
      common/geom/pi,hol,nj,ndec,icut
      common/gauss/xp(96),gw(96),ngt,xielec(201),
     1  xiothr(201),njmore,xx(201)
      common/nrdata/phicrt,totcrt,crit,vcell
c
      data pi/3.14159265358979d+00/, ian/1/, icath/2/, ip/1/, iq/2/,
     1  ifor/1/, iback/2/
c
c     trace subprogram time
c
      call readin(iunifc,ilimc,iprimc,iunifp,izerop,iprimp,
     1  xn,ujth,ciocr,gam,son,ng,root,itmaxo,d2,d3,nre,
     1  irstrt,ifine,nc,xiend)
c
      n2=2*nspec
      nd=2*n2+1
      coeff=-xn*(16.*z)**(-1./3.)
      njm1=nj-1
c
c     Rearrange the Gaussian roots so they are in the interval 0 to 1
c     instead of -1 to 1.
c
      ngt = 2*ng
c
      do 111 jxp=1,ngt
  111 xp(jxp)=(root(jxp)+1.)/2.
```

```
c
        do 170 i=1,nspec
          do 171 jx=1,nj
          do 171 ielec=ian,icath
            csurf(ielec,i,jx)=ciocr(i)
            if(ciocr(i).eq.0.) csurf(ielec,i,jx)=1.d-15
c
c           temp
c           if(jx.gt.1) csurf(ielec,i,jx)=1.
  171     continue
        do 170 j=1,nrxn
          porq(ip,i,j)=gam(i,j)+ason(ian,j)*ratson(i,j)*son
          porq(iq,i,j)=gam(i,j)-ason(icath,j)*ratson(i,j)*son
          print 177,i,j,porq(ip,i,j),porq(iq,i,j)
  177     format(1x,' p(',i2,',',i2,')=',1pe11.4,' q=',1pe11.4)
  170   continue
c
        do 172 j=1,nrxn
          sum = 0.d+00
          do 173 i=1,nspec
            if(cref(i) .lt. 1.d-15) go to 91
            sum = sum + ratson(i,j)*dlog(cref(i)/rho0)
   91       if(cre(i) .lt. 1.d-15) go to 173
            sum = sum - ratson(i,nre)*dlog(cre(i)/rho0)
  173     continue
          uref(j)=ujth(j)-ujth(nre)+sum
  172   continue
c
c       calculate integrals of ln sinh**2 and ln cosh**2 that appear
c       in the equation for the potential distribution after application
c       of the method of Kantorovich and Krylov (L. V. Kantorovich and
c       V. J. Krylov, "Approximate Methods of Higher Analysis,"
c       Interscience Publishers, Inc., New York (1959), p.  101.)
c
c
        call intcxi(nspec,xint0,xint1)
c
c       temp.
c
c       v(icath)=-6.167d+00
c       v(ian)=v(icath)+vcell
        v(icath)=0.d+00
        v(ian)=vcell
c
        phistr=vcell*ason(ian,1)/(ason(ian,1)+ason(icath,1))+v(icath)
c
c
        xiav=0.74058*xn
```

```
c
        if(irstrt.ne.0) print 280
  280   format(' this run is a restart')
c
        if(irstrt.ne.0) go to 206
                call iguess(iunifc,ilimc,iprimc,iunifp,izerop,iprimp,
     1            xi,p0mpsi,xiend)
                go to 207
  206   call rstrt(ifine,nc,xiav,phistr,xi,p0mpsi)
        print 208,xiav,phistr
  208   format(' after call rstrt xiav=',1pe11.4,' phistr=',1pe11.4)
c
  207   continue
c
c       temp
c       phistr=120.d+00
c
        n=3
        m=2*n+1
c
ccccccccccccccccccccccccccccccccccccccccccccccccc
c                                                 c
c       main iterative loop                       c
c                                                 c
ccccccccccccccccccccccccccccccccccccccccccccccccc
c
        x(3)=d3
        x(2)=d2
        x(1)=1.0d+00
        yold=0.d+00
        irep=3
c       print 198,damp
  198   format(' damp=',1pe11.4)
c
        do 190 it=1,itmaxo
c
        call dump(phistr,xiav,xi,p0mpsi)
c
        damp=x(3)
        do 209 idec = 1,ndec
        do 209 ielec=ian,icath
        do 209 jx=1,nj
  209   old(ielec,jx,idec) = p0mpsi(ielec,jx,idec)
c
        inrit=0
  300   inrit = inrit + 1
c
        print 191,it,inrit,phistr,xiav
```

```
 191    format(/,' outer iteration ',i3,' sub iteration',i3,/,
     1     ' phistr=',1pe11.4,' xiav=',1pe11.4)
c
       phiin = phistr
       xiavin = xiav
c
       i3=0
c
       do 238 idec = 1,ndec
       do 238 ielec=ian,icath
       do 238 jx=1,nj
  238  p0mpso(ielec,jx,idec)=p0mpsi(ielec,jx,idec)
c
       call phi(phiin,xiavin,xitotc,xitota,
     1   p0mpsi,eta,csurf,xij,xi,phistr,xiav,it,tox,fine,can,ccath,
     1   xitotj,toxj,son,etas)
c      print 233,phistr
c 233  format(' atfer call phi, phistr=',1pe11.4)
c
c      print 103
c 103  format(' converged concentrations',
c    1   /' anode        cathode         anode       cathode')
c      do 105 jx=1,nj
c        print 204,csurf(ian,1,jx),csurf(icath,1,jx)
c 204    format(1x,1pe11.4,1x,e11.4)
c 105  continue
c
c      print 114
c 114  format(' converged currents',/' anode        cathode')
c      do 205 jx=1,nj
c        print 214,xi(ian,jx),xi(icath,jx)
c 214    format(1x,1pe11.4,1x,e11.4)
c 205  continue
c
c
c      print 232,phicrt
c 232  format(' relative delta phi criterion=',1pe11.4)
c
       if(inrit.ne.1) go to 236
       do 235 idec=1,ndec
       do 235 ielec=ian,icath
       do 235 jx=1,nj
  235  svdist(ielec,jx,idec) = p0mpsi(ielec,jx,idec)
  236  sumsq = 0.d+00
c
       do 210 ielec=ian,icath
          if(ielec.eq.ian) print 231,ielec
  231     format(/,' potential distr. on electrode',i2,/,
```

```
     1       ' jx      old           new ')
          do 210 idec=1,ndec
          do 210 jx=1,nj
            if(idec.eq.ndec) print 230,jx,p0mpso(ielec,jx,ndec),
     1        p0mpsi(ielec,jx,ndec)
   230      format(1x,i2,3x,1pe11.4,3x,e11.4)
            sumsq = sumsq + (p0mpso(ielec,jx,ndec)
     1        - p0mpsi(ielec,jx,ndec))**2
            p0mpsi(ielec,jx,idec)=p0mpso(ielec,jx,idec) + (1.-damp)*
     1        (p0mpsi(ielec,jx,idec)-p0mpso(ielec,jx,idec))
            if(jx.eq.nj) print *,' elec',ielec,' sumsq=',sumsq
            if(jx.eq.1 .or. jx.eq.nj) go to 210
c           if(it.ne.itmaxo) go to 210
            if(dabs((p0mpso(ielec,jx,ndec)-p0mpsi(ielec,jx,ndec))/p0mpsi(
     1        ielec,jx,ndec))
     1        .ge.phicrt) go to 210
            if(sumsq.lt.0.001d+00) go to 258
               print 259
   259         format(' would have converged')
               go to 210
c
   258      if(ielec.eq.icath .and. jx.eq.njm1) go to 193
   210    continue
c
c         call phiint(p0mpsi)
c
          print 211,damp,it,inrit,sumsq
   211    format(' damp=',1pe11.4,' outer it=',i2,' sub it=',i2,/,
     1      ' sum of squares=',1pe11.4)
c
          if(inrit.eq.1) go to 11
c
            if(inrit.eq.2) go to 22
              if(inrit.eq.3) go to 33
                   y(irep)=sumsq
                   go to 37
c
    33           y(2)=sumsq
                 yold=y(2)
                 irep=2
    37           continue
                 n = 3
                 m = 2*n + 1
                 do 35 i=1,n
    35           fit(i)=1.d+00
                 call abc(n,m,x,y,fit,idump)
                 if(.not.idump) go to 40
                   call dump(phistr,xiav,xi,p0mpsi)
```

```
                print 42
   42           format(' bombed in abc')
                stop
   40         damp = -fit(2)/(2.*fit(1))
              print 198,damp
              if(inrit.gt.3 .and. dabs((y(irep)-yold)/y(irep)) .le.
     1          1.0d-03) goback=.false.
              mid=1
              if( (x(3) - x(2))*(x(1) - x(2)) .le. 0.d+00) mid=2
              if( (x(1) - x(3))*(x(2) - x(3)) .le. 0.d+00) mid=3
              if(mid.eq.1 .and. y(1).lt.y(2) .and. y(1).lt.y(3))
     1          goback=.false.
              if(mid.eq.2 .and. y(2).lt.y(1) .and. y(2).lt.y(3))
     1          goback=.false.
              if(mid.eq.3 .and. y(3).lt.y(2) .and. y(3).lt.y(1))
     1          goback=.false.
              if(x(mid).ge.1.0d+00) goback=.true.
              if(.not.goback) go to 36
c             print 38,mid
c  38         format(' mid=',i2)
              i3=0
              if(fit(1).le.0.d+00) go to 39
                irep=1
                if (dabs(y(2)) .gt. dabs(y(1)) ) irep=2
                if (dabs(y(3)) .gt. dabs(y(irep)) ) irep=3
              x(irep)=damp
              i3=1
              go to 36

c
c
   39    n=2
         m=2*n+1
c
         if(y(1).gt.y(2)) go to 200
           icomp=2
           y1=y(1)
           x1=x(1)
           go to 212
  200    icomp=1
         y1=y(2)
         x1=x(2)
  212    continue
c
         if(y(3).gt.y(icomp)) go to 220
           y2=y(3)
           x2=x(3)
           go to 240
```

```
  220   y2=y(icomp)
        x2=x(icomp)
  240   continue
c
        if(x1.le.x2) go to 101
          xt=x1
          yt=y1
          x1=x2
          y1=y2
          x2=xt
          y2=yt
  101   if(y1.gt.y2) damp=2.*x2-x1
        if(y2.ge.y1) damp=2.*x1-x2
        y(1)=y1
        y(2)=y2
        x(1)=x1
        x(2)=x2
        print 198,damp
        irep=3
        x(3)=damp
        goback = .true.
c
c
   36   x3=damp
        yold=y(irep)
c
              do 430 idec=1,ndec
              do 430 ielec=ian,icath
              do 430 jx=1,nj
                p0mpsi(ielec,jx,idec) = old(ielec,jx,idec) + (1.-damp)*
     1            (svdist(ielec,jx,idec)-old(ielec,jx,idec))
  430         continue
              damp=0.d+00
              go to 189
c
   22     y(3)=sumsq
          damp=x(2)
          print 198,damp
          do 330 idec=1,ndec
          do 330 ielec=ian,icath
          do 330 jx=1,nj
  330     p0mpsi(ielec,jx,idec) = old(ielec,jx,idec) + (1.-damp)*
     1      (svdist(ielec,jx,idec)-old(ielec,jx,idec))
          damp=0.d+00
          goback = .true.
          go to 189
c
   11   y(1)=sumsq
```

```
        damp=0.d+00
        goback = .true.
c
  189   continue
c       call phiint(p0mpsi)
        if(goback .and. inrit.lt.7) go to 300
c
        ibest=1
        if(x(1).ge.1.0d+00) ibest=2
        if(y(2).lt.y(1) .and. x(2).lt.1.0d+00) ibest=2
        if(y(3).lt.y(ibest) .and. x(3).lt.1.0d+00 .and. i3.eq.0) ibest=3
        save=x(ibest)
c
        x(1)=1.0d+00
        x(2)=save
        x(3)=x3
        print 194,x(1),x(2),x(3)
  194   format(' x(1,2,3)=',3e12.4)
  190   continue
c
        print 320,itmaxo
  320   format(' no conv in outer loop after',i2,' iterations')
  193   continue
c
c
        call prtout(coeff,nj,ndec,icut,cref,porq,uref,nspec,nrxn,
     1    ciocr,xij,xi,csurf,p0mpsi,xiav,phistr,tox,fine,can,ccath,
     1    xitotj,toxj,etas)
c
        stop
        end
c
c
        subroutine phi(phiin,xiavin,xitotc,xitota,
     1    p0mps,eta,csurf,xij,xi,phistr,xiav,it,tox,fine,can,ccath,
     1    xitotj,toxj,son,etas)
c
        implicit double precision (a-h,o-z)
        dimension csurf(2,10,101),xintgd(101),
     1    xij(2,5,101),xi(2,101),p0mps(2,101,5),eta(2,101,5),
     1    xiint(2,96,5),tox(2,101),fine(2,5,101),can(5,10,101),
     1    ccath(5,10,101),xitotj(2,5),toxj(2,101,5),etas(2,101)
c
        common/evprob/xlamda(3),ai(3),z,diodr(10)
        common/co/coeff,xn
        common/rxnpars/cref(10),cre(10),porq(2,10,5),ip,iq,uref(5),
     1    nspec,nrxn,one(2,5),rho0
        common/matcom/b(20,20),d(20,41)
```

```
      common/kinpars/xj(5),ratson(10,5),ason(2,5),ifor,iback
      common/cdata/cnctol,ftol,xint0(10,101,5),
     1   xint1(10,101,5),n2,nd,isec
      common/elecs/v(2),ian,icath
      common/geom/pi,hol,nj,ndec,icut
      common/gauss/xp(96),gw(96),ngt,xielec(201),
     1   xiothr(201),njmore,xx(201)
      common/nrdata/phicrt,totcrt,crit,vcell
c
      njm1=nj-1
c
ccccccccccccccccccccccccc
c                         c
c       i<avg> loop       c
c                         c
ccccccccccccccccccccccccc
c
      call iavlp (phiin,xiavin,xitotc,xitota,
     1   p0mps,eta,csurf,xij,xi,phistr,xiav,it,tox,fine,can,ccath,
     1   xitotj,toxj,son,etas)
c
c       print 251,phistr
c 251   format(' after call iavlp, phistr=',1pe11.4)
c
      print *,'currents '
      njmore = 0
      xmax=-1.
      do 249 idec=1,ndec
        h=dfloat(icut)**(idec-ndec)/dfloat(nj-1)
      do 249 jx = 1,nj
        x = dfloat(jx-1)*h
     if(x.gt.xmax) then
       xmax=x
        njmore = njmore + 1
        xx(njmore) = x
        xielec(njmore) = fine(1,idec,jx)/xiav
        xiothr(njmore) = fine(2,idec,jx)/xiav
     endif
  249 print 268,x,
     1   fine(1,idec,jx),fine(2,idec,jx),fine(1,idec,jx) +
     1   fine(2,idec,jx)
  268 format(1pe11.4,1x,2(1pe20.13,1x),1x,1pe11.4)
      do 248 jx=1,nj
        x = dfloat(jx-1)*h
  248 print 247,x,tox(1,jx)+tox(2,jx)
  247 format(2(1pe11.4,1x))
      print *,'normalize i by',xiav
      do 250 ielec=ian,icath
```

```
      do 250 jx=1,nj
250   xi(ielec,jx)=xi(ielec,jx)/xiav
c
c     calculate phi 0 - phi * (p0mps) from solution to Laplace's eqn.
c
c
      do 200 jx=1,nj
      jreal = jx
      xol = dfloat(jx-1)/dfloat(nj-1)
      call eqfive(xol,phian,phicath,p2,p3,p5)
c     print 181,xol,p2,p3,phicath,p5,phian
      if(jreal.ge.1 .and. jreal.le.nj) p0mps(ian,jreal,ndec) = phian
      if(jreal.ge.1 .and. jreal.le.nj) p0mps(icath,jreal,ndec)
     1   = phicath
181   format(0pf8.4,x,1p5e12.4)
200   continue
      call phiint(p0mps)
      if(ndec.eq.1) go to 202
      do 201 idec=1,ndec-1
      do 201 jx=1,nj
        xol = dfloat(jx-1)/dfloat(nj-1)/icut**(ndec-idec)
        if(xol.lt.0.02) then
          call eqfive(xol,phian,phicath,p2,p3,p5)
          p0mps(ian,jreal,idec) = phian
          p0mps(icath,jreal,idec) = phicath
        end if
201   continue
c
202   print *,'p0mps'
      print 269,((dfloat(jx-1)/dfloat(nj-1),p0mps(1,jx,ndec),
     1   p0mps(2,jx,ndec)),jx=1,nj)
269   format(3(1pe11.4,1x))
      istop = 0
      if(istop.ne.0) stop
      return
      end
c
c
      subroutine abc(n2,nd,x,y,coeff,idump)
c
      implicit double precision (a-h,o-z)
      logical idump
      common/matcom/b(20,20),d(20,41)
      dimension x(3),y(3),coeff(3)
c
      idump = .false.
      print 122,(x(i),i=1,n2)
122   format(' x(i)=',3e12.4)
```

```
      print 222,(y(i),i=1,n2)
  222 format(' y(i)=',3e12.4)
      print 223,(coeff(i),i=1,n2)
  223 format(' coeff(i)=',3e12.4)
c     print 224,n2,nd
c 224 format(' n2=',i2,' nd=',i2)
c
      n2p1=n2+1
c
      do 100 i=1,n2
      do 110 j=1,n2
        b(i,j)=0.d+00
  110 d(i,j) = 0.d+00
      do 100 j=n2p1,nd
  100 d(i,j)=0.d+00
c
      do 120 j=1,n2
      do 120 i=1,n2
  120 b(i,j)=x(i)**(n2-j)
c
      do 127 i=1,n2
      d(i,1)=0.d+00
      do 128 j=1,n2
  128 d(i,1)=d(i,1)-coeff(j)*x(i)**(n2-j)
  127 d(i,1)=d(i,1)+y(i)
c
c     print 137
c 137 format(' bij')
c     do 124 i=1,n2
c     print 123,(b(i,j),j=1,n2)
c 123 format(3e12.4)
c 124 continue
c
c     print 136
c 136 format(' dij')
c     do 126 i=1,n2
c     print 134,(d(i,j),j=1,nd)
c 134 format(7e12.4)
c 126 continue
c
      call matinv(n2,nd,determ)
      if(determ.eq.0.d+00) print 121
  121 format(' zero determinant')
      if(determ.ne.0.d+00) go to 131
        idump=.true.
        return
c
  131 continue
```

```
      do 117 i=1,n2
  117 coeff(i) = coeff(i) + d(i,1)
c
c     xx = -coeff(2)/(2.*coeff(1))
c     yy = 0.d+00
c     do 119 j=1,n2
c 119 yy = yy + coeff(j)*xx**(n2-j)
c
c     print 125,y(1),y(2),y(3),yy,xx,coeff(1),coeff(2),coeff(3)
c 125 format(' y1,y2,y3=',3e12.4,' y=',e12.4,/,' x=',1pe11.4,/,
c    1   ' aa,bb,cc=',3e12.4)
c
c     print 130
c 130 format(' coefficients')
c     print 129,(coeff(i),i=1,n2)
c 129 format(5e12.4)
c
      return
      end
c
c
      subroutine iguess(iunifc,ilimc,iprimc,iunifp,izerop,iprimp,
     1   xi,p0mps,xiend)
c
      implicit double precision (a-h,o-z)
      dimension iunifc(2),ilimc(2),iprimc(2),sign(2),xi(2,101),
     1   p0mps(2,101,5),xiint(2,96,5)
c
      common/elecs/v(2),ian,icath
      common/geom/pi,hol,nj,ndec,icut
      common/gauss/xp(96),gw(96),ngt,xielec(201),
     1   xiothr(201),njmore,xx(201)
c
      ndm1=ndec-1
      icertx=0
      sign(ian)=1.d+00
      sign(icath)=-1.d+00
c
      do 100 ielec=ian,icath
        if(iunifc(ielec).ne.0) go to 10
   11   if(ilimc(ielec).ne.0) call ilim(ielec,xi,xiint)
  100 if(iprimc(ielec).ne.0) call primry(iprimc,ielec,xi,x,icertx,
     1   xix,xiint,xiend)
c
      if(iprimp.ne.0) go to 20
   12 if(iunifp.ne.0) go to 30
   13 if(izerop.ne.0) go to 40
   14 if(iprimp.ne.0 .or. iunifp.ne.0 .or. izerop.ne.0) return
```

```
c
        do 200 jx=1,nj
        jreal = jx
        xol = dfloat(jx-1)/dfloat(nj-1)
        call eqfive(xol,phian,phicath,p2,p3,p5)
c       print 181,xol,p2,p3,phicath,p5,phian
        if(jreal.ge.1 .and. jreal.le.nj) p0mps(ian,jreal,ndec) = phian
        if(jreal.ge.1 .and. jreal.le.nj) p0mps(icath,jreal,ndec)
     1    = phicath
  181   format(0pf8.4,x,1p5e12.4)
  200 continue
c
        call phiint(p0mps)
c
        return
c
c       uniform current
c
   10   do 120 jx=1,nj
  120   xi(ielec,jx)=sign(ielec)
        if(iunifp.ne.0) go to 11
        do 121 idec=1,ndec
        do 121 jg=1,ngt
  121   xiint(ielec,jg,idec)=sign(ielec)
        go to 11
c
c       primary distribution
c
   20   do 130 ielec=ian,icath
        do 130 jx=1,nj
          p0mps(ielec,jx,ndec)=sign(ielec)*hol/pi
          if(hol.gt.1.0d+00) p0mps(ielec,jx,ndec)=sign(ielec)/pi*
     1      (1.+dlog(hol))
          if(ndec.eq.1) go to 130
          do 131 idec=1,ndm1
  131     p0mps(ielec,jx,idec)=p0mps(ielec,jx,ndec)
  130   continue
        go to 12
c
c       potential distribution for uniform current distribution
c
   30   do 140 ielec=ian,icath
        do 140 jx=1,nj
          p0mps(ielec,jx,ndec)=sign(ielec)*hol/pi
          if(hol.gt.1.0d+00) p0mps(ielec,jx,ndec)=sign(ielec)/pi*
     1      (1.+dlog(hol))
          if(ndec.eq.1) go to 140
          do 141 idec=1,ndm1
```

```
  141     p0mps(ielec,jx,idec)=p0mps(ielec,jx,ndec)
  140   continue
        call phiint(p0mps)
        go to 13
c
c       no ohmic drop
c
   40   do 150 ielec=ian,icath
        do 150 jx=1,nj
        do 150 idec=1,ndec
  150   p0mps(ielec,jx,idec)=0.d+00
        go to 14
c
        end
c
c
        subroutine ilim(ielec,xi,xiint)
c
        implicit double precision (a-h,o-z)
        dimension xi(2,101),xiint(2,96,5),sign(2)
c
        common/geom/pi,hol,nj,ndec,icut
        common/gauss/xp(96),gw(96),ngt,xielec(201),
     1    xiothr(201),njmore,xx(201)
c
        njm1=nj-1
        denom=dfloat(njm1)
        sign(1)=1.d+00
        sign(2)=-1.d+00
c
        xi(ielec,1)=sign(ielec)*1.0e+20
c
        do 100 jx=2,nj
          x=dfloat(jx-1)/denom
          xi(ielec,jx)=sign(ielec)*2./3. * x**(-1./3.)
  100   continue
c
        hold=0.d+00
c
        do 141 idec=1,ndec
          h=dfloat(icut)**(idec-ndec)/denom
          do 149 jg=1,ngt
            x=xp(jg)*(njm1*h - hold) + hold
            xiint(ielec,jg,idec)=sign(ielec)*2./3. * x**(-1./3.)
  149     continue
          hold=h*njm1
  141   continue
c
```

```
      return
      end
c
c
      function iother(i)
c
c     this function is used for generating the index of the 'other'
c     electrode
c
      iother=3-i
      return
      end
c
c
      subroutine totcj(jbegin,h,fj,savej,toxj,j)
c
      implicit double precision (a-h,o-z)
      dimension save(2),fj(2,5,101),xintgd(101),tox(2,101),savej(2,5),
     1   toxj(2,101,5)
c
      common/geom/pi,hol,nj,ndec,icut
      common/elecs/v(2),ian,icath
c
c     this subroutine integrates the function fj (call is inside
c     idec loop and j loop)
c
      do 95 ielec=ian,icath
        save(ielec) = savej(ielec,j)
c       print 94,ielec,j,savej(ielec,j)
c 94    format(' elec',i2,' j=',i2,' savej input=',1pe11.4)
      do 95 jx=1,nj
 95   tox(ielec,jx) = toxj(ielec,jx,j)
c
      kend=(nj-1)/icut + 1
      jb=jbegin
      npts=nj-1
      if(jbegin.eq.2) go to 367
      jb=jbegin-1
      npts=nj-kend+1
367   continue
c
      do 463 ielec=ian,icath
        do 363 jx=jb,nj
          jj=jx-jb+1
c         print 93,j,ielec,jj,fj(ielec,j,jx)
c 93      format(' rxn',i2,' elec',i2,' jj=',i2,' xintgd=',1pe11.4)
363     xintgd(jj)=fj(ielec,j,jx)
c       print *,' tox=',tox(ielec,jb)
```

```
          call simps(xintgd,npts,h,save(ielec),ielec,jb,tox)
c         print *,' save=',save(ielec)
  463   continue
c
        do 92 ielec=ian,icath
          savej(ielec,j)=save(ielec)
          do 92 jx=1,nj
   92   toxj(ielec,jx,j)=tox(ielec,jx)
        if(jbegin.ne.2) return
c
        do 100 ielec=ian,icath
          tox(ielec,1) = 0.d+00
c         print 465
c 465     format(' in totcurr initializing tox(1)')
          g1=fj(ielec,j,1)
          g2=fj(ielec,j,2)
          if(g1.eq.0.d+00 .or. g2.eq.0.d+00) go to 373
          if(g1.ne.g2 .and. g1*g2/dabs(g1*g2).gt.0.d+00) go to 372
  373     save(ielec)=save(ielec) + h/2.*(g1+g2)
          tox(ielec,2) = h/2.*(g1+g2)
          do 98 jx=3,nj
   98     tox(ielec,jx)=tox(ielec,jx) + h/2.*(g1+g2)
          go to 99
  372     continue
          save(ielec)=save(ielec) + 1.5*h*g2
          tox(ielec,2) = 1.5*h*g2
          do 97 jx=3,nj
   97     tox(ielec,jx)=tox(ielec,jx) + 1.5*h*g2
   99     continue
c         print 466
c 466     format(' in totcurr initializing tox(2)')
  100   continue
c
        do 96 ielec=ian,icath
          savej(ielec,j) = save(ielec)
        do 96 jx=1,nj
   96   toxj(ielec,jx,j) = tox(ielec,jx)
c
        return
        end
c
c
        subroutine readin(iunifc,ilimc,iprimc,iunifp,izerop,iprimp,
     1    xn,ujth,ciocr,gam,son,ng,root,itmaxo,d2,d3,
     1    nre,irstrt,ifine,nc,xiend)
c
c       this subroutine reads and prints the input data for
c       program channel
```

```
c
c       input data are in file
c           channel.data
c
c       The parameter iwhite tells whether input data are dimensional
c           or dimensionless (iwhite=0 means dimensionless)
c
c   I.  The input and output dimensionless parameters are:
c
c       1)  Dimensionless parameters describing the problem:
c
c           xn = N = n<m>**2/s<Rm>**2 F**2 D<R> c<R,b>/(RT K<b>)*
c                (6v L**2/(h D<R>))**(1/3) = dimensionless
c                limiting current density
c           rho0 = rho<0>/c<R,b> = solvent density / bulk concentration
c                of main reactant
c           ujth(j) = U<j>[theta] (-n<m> F/(s<Rm> RT)) = dimensionless
c                standard electrode potential for reaction j
c           ciocr(i) = c<i,b>/c<R,b> = dimensionless bulk concentration
c                of species i
c           gam(i,j) = exponent in composition dependence of exchange
c                current density
c
c           z,diodr
c
c           hol
c
c           cref,cre,nspec,nrxn,one
c
c           xj,ratson,ason
c
c           vcell
c
c       2)  Variables related to the solution method
c
c           ng = 1/2 the number of Gaussian quadrature points to be
c                used in the integrals for the potential distribution
c                (maximum allowable=48)
c           root(jxp) = roots for Gaussian quadrature (range from -1
c                to 1)
c           isec = 0 if tertiary current distribution, nonzero if
c                secondary
c           itmaxo = maximum number of iterations allowed in outer loop
c           nre = which reaction is reference electrode reaction
c
c           xlamda,ai
c
c           nj,ndec
```

```
c
c           cnctol,ftol
c
c           gw
c
c           phicrt,totcrt,crit
c
c  II.  The dimensional parameters (if iwhite.ne. 0) are
c
c           h = gap thickness (cm)
c           xl = electrode length (cm)
c           vcell = applied cell voltage (V)
c           rho = solvent density (kg/cc)
c           vel = average linear velocity through channel (cm/s)
c           dr = diffusion coefficient of main reactant (cm2/s)
c           cr = feed concentration of main reactant (mol/cc)
c           xnm = n<m> = no. of electrons in main rxn
c           srm = stoichiometric coeff. of main reactant in main rxn
c           aaj = anodic transfer coefficient for reaction j
c           acj = cathodic transfer coefficient for reaction j
c           xj(j) = exchange current density for reaction j (A/cm2)
c           uthj = standard electrode potential for reaction j (V)
c           nse = number of species in solution that do not participate
c                in electrolyte reactions (for ex. supporting
c                electrolyte species)
c           dse(i) = diffusion coefficient of nonparticipating species i
c           cse(i) = feed concentration of nonparticipating species i
c           zse(i) = charge number of nonparticipating species i
c           di = diffusion coefficient of species i (a participating
c                species)
c           crefi = reference concentration of species i (mol/cc)
c           crei = concentration of species i in reference electrode
c                compartment (mol/cc)
c           ci = feed concentration of species i (mol/cc)
c           zi = charge number of species i
c           gam(i,j) = exponent for concentration dependence of rxn j
c                exchange current density on species i
c           ratson(i,j) = ratio of s<i,j>/n<j> (stoichiometric coeff.
c                of species i in rxn j to number of electrons in rxn j
c
c
      implicit double precision (a-h,o-z)
      dimension root(96),ujth(5),ciocr(10),gam(10,5),iunifc(2),
     1   ilimc(2),iprimc(2),dse(10),cse(10),zse(10)
c
      common/evprob/xlamda(3),ai(3),z,diodr(10)
      common/rxnpars/cref(10),cre(10),porq(2,10,5),ip,iq,uref(5),
     1   nspec,nrxn,one(2,5),rho0
```

```
      common/kinpars/xj(5),ratson(10,5),ason(2,5),ifor,
     1   iback
      common/cdata/cnctol,ftol,xint0(10,101,5),
     1   xint1(10,101,5),n2,nd,isec
      common/geom/pi,hol,nj,ndec,icut
      common/gauss/xp(96),gw(96),ngt,xielec(201),
     1   xiothr(201),njmore,xx(201)
      common/nrdata/phicrt,totcrt,crit,vcell
c
      read  *,ng
      if(ng.gt.48 .or. ng.lt.1) print 600,ng
  600 format(' ng=',i2,' is incorrect input')
c
c     Read the Gaussian quadrature roots and weights for the interval
c     0 to 1.  Then put in the roots and weights for the interval
c     -1 to 0.
c
      do 101 k=1,ng
        karg=ng+1-k
        kpng=k+ng
        read  *,(root(karg),gw(karg))
        root(kpng)=root(karg)
        gw(kpng)=gw(karg)
  101 root(karg)=-root(karg)
c
      read *,(xlamda(i),i=1,3)
      read *,(ai(i),i=1,3)
      read 670,ichar
  670 format(a72)
      read *,nj,ndec,irstrt,ifine,nc
      print 617,irstrt
  617 format(' restart=',i3)
      if(nj.gt.101 .or. nj.lt.1)
     1   print 615,nj
  615 format(' nj=',i3,' is incorrect input')
      if(ndec.gt.5 .or. ndec.lt.1) print 616,ndec
  616 format(' ndec=',i3,' is incorrect input')
c
      read 670,ichar
      read *,iwhite
      if(iwhite.ne.0) print 619
  619 format(' Input Ralph White dimensional parameters')
c
      if(iwhite.ne.0) go to 300
      go to 299
  300 read 670,ichar
      read *,h,xl,vcell,rho,vel
      read 670,ichar
```

```
      read *,dr,cr,xnm,srm
      print 621,h,xl,vcell,rho,vel,dr,cr
  621 format(' h=',1pe11.4,' l=',1pe11.4,' vcell=',1pe11.4,' rho=',
     1   1pe11.4,' vel=',1pe11.4,/,' dr=',1pe11.4,' cr=',1pe11.4)
c
      hol=h/xl
      z=3.*h**2*vel/(8.*dr*xl)
      rho0=rho/(1000.*cr)
      vcell=-vcell*xnm/(srm*0.0257)
      go to 301
c
  299 read 670,ichar
      read *,(iunifc(i),i=1,2),(ilimc(i),i=1,2),(iprimc(i),i=1,2)
c
      do 516 i=1,2
        if(iunifc(i).ne.0) print 514,i
  514   format(' uniform current on electrode',i2)
  516 continue
c
      do 513 i=1,2
        if(ilimc(i).ne.0) print 512,i
  512   format(' limiting current on electrode',i2)
  513 continue
c
      do 511 i=1,2
        if(iprimc(i).ne.0) print 509,i
  509   format(' primary current on electrode',i2)
  511 continue
c
      read 670,ichar
      read *,izerop
      iunifp=0
      iprimp=0
      if(iunifc(1).ne.0 .and. iunifc(2).ne.0) iunifp=1
      if(iprimc(1).ne.0 .and. iprimc(2).ne.0) iprimp=1
      if(izerop.ne.0) print 508
  508 format(' no ohmic drop')
c
      read 670,ichar
      read *,hol,icut,xn,pehol,rho0,vcell,son
      print 620,nj,ndec,hol,icut,xn,pehol,rho0,vcell
  620 format(' nj=',i3,' ndec=',i3,' hol=',1pe11.4,' icut=',i3,
     1   ' xn=',1pe11.4,/,' pehol=',1pe11.4,
     1   ' rho0=',1pe11.4,' v(cell)=',1pe11.4)
c
      z=3./16. * pehol
c
  301 read 670,ichar
```

```
      read *,isec,itmaxo,nspec,nrxn,nre,xiend
      read 670,ichar
      read *,cnctol,ftol,crit,totcrt,phicrt,d2,d3
c
      if(isec.ne.0) print 629
  629 format(' secondary current distribution')
      print 628,itmaxo,nspec,nrxn,nre,cnctol,ftol,crit,totcrt,
     1   phicrt,d2,d3
  628 format(' max iterations=',i3,/,
     1   i3,' species,',i3,' reactions, reference rxn is rxn',i3,/,
     1   ' conc tolerance=',1pe11.4,' conc eqn tolerance=',1pe11.4,/,
     1   ' conv crit for i avg=',1pe11.4,
     1   ' conv crit for phi star loop=',
     1   1pe11.4,/,' conv crit for potl distribution=',1pe11.4,
     1   /,' d2=',1pe11.4,' d3=',1pe11.4)
c
      if(nspec.gt.10 .or. nspec.lt.1) print 625,nspec
  625 format(' nspec=',i3,' is incorrect input')
      if(nrxn.gt.5 .or. nrxn.lt.1) print 626,nrxn
  626 format(' nrxn=',i3,' is incorrect input')
      if(nre.gt.5 .or. nre.lt.1) print 627,nre
  627 format(' nre=',i3,' is incorrect input')
c
      do 162 ielec=1,2
        read 670,ichar
        read *,(one(ielec,j),j=1,5)
        print 631,(one(ielec,j),j=1,nrxn)
  631   format(' one=',5f2.0)
  162 continue
c
      if(iwhite.eq.0) go to 302
c
      do 171 j=1,nrxn
        read 670,ichar
        read *,aaj,acj,xj(j),uthj
        print 632,aaj,acj,xj(j),uthj
  632   format(' aaj,acj=',1pe11.4,2x,1pe11.4,' xj=',1pe11.4,' uth=',
     1    1pe11.4)
        ason(1,j)=-aaj*srm/xnm
        ason(2,j)=-acj*srm/xnm
        ujth(j)=-uthj*xnm/(srm*0.0257)
  171 continue
c
      if(nre.eq.nrxn) go to 502
      read 670,ichar
      read *,uthj
      ujth(nre)=-uthj*xnm/(srm*0.0257)
c
```

```
  502   read 670,ichar
        read *,nse
        do 181 i=1,nse
          read 670,ichar
          read *,dse(i),cse(i),zse(i)
  181   continue
c
        xk=0.d+00
        do 172 i=1,nspec
          read 670,ichar
          read *,di,crefi,crei,ci,zi
          print 636,di,crefi,crei,ci,zi
  636     format(' di=',1pe11.4,' cref=',1pe11.4,' cre=',1pe11.4,' ci=',
     1      1pe11.4,' zi=',1pe11.4)
          diodr(i)=di/dr
          cref(i)=crefi/cr
          cre(i)=crei/cr
          ciocr(i)=ci/cr
  172   xk=xk+zi**2*di*ci
        do 182 i=1,nse
  182   xk=xk+zse(i)**2*dse(i)*cse(i)
        xk=xk*96487./(0.0257)
        print 637,xk
  637   format(' xk=',1pe11.4)
c
        xn=(xnm/srm)**2*96487.*dr*cr/(0.0257*xk)*(6.*vel*xl**2/(h*dr))
     1    **(1./3.)
c
        do 173 j=1,nrxn
  173   xj(j)=xj(j)*(-xnm/srm)*xl/(0.0257*xk)
c
        do 174 i=1,nspec
          read 670,ichar
  174   read *,(gam(i,j),j=1,nrxn)
c
        do 176 i=1,nspec
          read 670,ichar
          read *,(ratson(i,j),j=1,nrxn)
          do 175 j=1,nrxn
            print 641,ratson(i,j),i,j
            ratson(i,j)=ratson(i,j)/(srm/xnm)
            if(nre.gt.nrxn) go to 175
            print 642,gam(i,j),i,j
  642       format(' pij=',1pe11.4,' i=',i2,' j=',i2)
            gam(i,j)=gam(i,j)+ason(l,j)*ratson(i,j)
  175     continue
  176   continue
c
```

```
      print 620,nj,ndec,hol,icut,xn,z,rho0,vcell
      do 177 j=1,nrxn
  177 print 630,ason(1,j),ason(2,j),xj(j),ujth(j)
      do 178 i=1,nspec
  178 print 635,diodr(i),cref(i),cre(i),ciocr(i)
      do 179 i=1,nspec
      do 179 j=1,nrxn
          print 641,ratson(i,j),i,j
          if(nre.gt.nrxn) go to 179
          print 640,gam(i,j),i,j
  179 continue
c
      go to 304
  302 continue
c
      do 163 j=1,nrxn
        read 670,ichar
        read *,ason(1,j),ason(2,j),xj(j),ujth(j)
        print 630,ason(1,j),ason(2,j),xj(j),ujth(j)
  630   format(' ason(1)=',1pe11.4,' ason(2)=',1pe11.4,
     1    ' xj=',1pe11.4,
     1    ' ujth=',1pe11.4)
  163 continue
c
      if(nre.eq.nrxn) go to 503
      read 670,ichar
      read *,ujth(nre)
c
  503 continue
c
      do 164 i=1,nspec
        read 670,ichar
        read *,diodr(i),cref(i),cre(i),ciocr(i)
        print 635,diodr(i),cref(i),cre(i),ciocr(i)
  635   format(' diodr=',1pe11.4,' cref=',1pe11.4,
     1    ' cre=',1pe11.4,' ciocr=',1pe11.4)
  164 continue
c
      do 166 i=1,nspec
        read 670,ichar
        read *,(gam(i,j),j=1,nrxn)
        if(cref(i).lt.1.0d-15 .and. gam(i,j).ne.0.) print *,' incorrect input
     1    for gam(i,j)'
        do 199 j=1,nrxn
          print 640,gam(i,j),i,j
  640     format(' gam=',1pe11.4,' i=',i2,' j=',i2)
  199   continue
  166 continue
```

```
c
      do 168 i=1,nspec
        read 670,ichar
        read *,(ratson(i,j),j=1,nrxn)
        do 201 j=1,nrxn
          print 641,ratson(i,j),i,j
  641     format(' ratson=',1pe11.4,' i=',i2,' j=',i2)
  201   continue
  168 continue
c
  304 return
      end
c
c
      subroutine prtout(coeff,nj,ndec,icut,cref,porq,uref,nspec,nrxn,
     1  ciocr,xij,xi,csurf,p0mps,xiav,phistr,tox,fine,can,ccath,
     1  xitotj,toxj,etas)
c
c     This subroutine prints the final results of program channel
c
      implicit double precision (a-h,o-z)
      dimension csurf(2,10,101),ciocr(10),xij(2,5,101),
     1  uref(5),porq(2,10,5),tox(2,101),fine(2,5,101),
     1  xi(2,101),cref(10),p0mps(2,101,5),can(5,10,101),xitotj(2,5),
     1  toxj(2,101,5),ccath(5,10,101),etas(2,101)
c
      print 139,coeff,xiav,phistr
  139 format(/,' coeff=',1pe11.4,' xiav=',1pe11.4,' phistr=',1pe11.4)
      print 138,(xitotj(1,j),j=1,nrxn)
  138 format(' xitotj, anode',2x,1pe11.4,4(1x,e11.4))
      print 137,(xitotj(2,j),j=1,nrxn)
  137 format(' xitotj, cathode',1pe11.4,4(1x,e11.4))
c
      do 145 j=1,nrxn
        print 146,j,uref(j)
  146   format(' j=',i2,' ujref=',1pe11.4)
  145 continue
c
      do 144 i=1,nspec
        print 143,i,(j,porq(1,i,j),j=1,nrxn)
  143   format(' i=',i2,5(' j=',i2,' pij=',1pe11.4))
        print 142,i,(j,porq(2,i,j),j=1,nrxn)
  142   format(' i=',i2,5(' j=',i2,' qij=',1pe11.4))
  144 continue
c
      do 149 i=1,nspec
        print 147,i
  147   format(/,' species',i3,/,'    x      ccath     can')
```

```
      do 149 jx=1,nj
        x=dfloat(jx-1)/dfloat(nj-1)
        print 148,x,csurf(2,i,jx),csurf(1,i,jx)
 148    format(1x,1pe9.2,1x,1pe11.4,2x,1pe11.4)
 149  continue
c
      do 150 j=1,nrxn
        print 151,j
 151    format(/,' reaction',i2,/,
     1    '     x          xicj          xiaj      ')
      do 150 jx=1,nj
        x=dfloat(jx-1)/dfloat(nj-1)
        print 121,x,xij(2,j,jx),xij(1,j,jx)
 121    format(1x,1pe9.2,1x,1pe11.4,2x,1pe11.4)
 150  continue
c
c     print 170
c 170  format(/,' total currents',/
c    1   '     x          ian          icath')
c     do 187 jx=1,nj
c       x=dfloat(jx-1)/dfloat(nj-1)
c       print 188,x,xi(1,jx),xi(2,jx)
c 188   format(1x,1pe9.2,1x,1pe11.4,2x,1pe11.4)
c 187  continue
c
      print 191
 191  format(/,' axial currents',/'      x           iaxial')
      do 192 idec=1,ndec
        h=dfloat(icut)**(idec-ndec)/dfloat(nj-1)
      do 192 jx=1,nj
        x=h*dfloat(jx-1)
        print 193,x,fine(1,idec,jx)+fine(2,idec,jx),fine(1,idec,jx),
     1    fine(2,idec,jx)
 193    format(1x,1pe9.2,1x,1pe11.4,1x,2(e11.4,1x))
 192  continue
c
c     m=(nj+1)/2
c     print 180,m,xi(1,1)/xi(1,m)
c 180  format(' midpt=',i2,' i(1)/i(m)=',1pe11.4)
c
      do 16 idec=1,ndec
        h=dfloat(icut)**(idec-ndec)/dfloat(nj-1)
        print 479,idec
 479    format(' decade',i2,/,'    x          c anode(x)')
      do 16 k=1,nj
        x=h*dfloat(k-1)
        print 480,x,(can(idec,i,k),i=1,nspec)
 480    format(1x,1pe9.2,1x,1pe11.4,4(1x,e11.4))
```

```
  16    continue
        do 17 idec=1,ndec
          h=dfloat(icut)**(idec-ndec)/dfloat(nj-1)
          print 481,idec
  481     format(' decade',i2,/,'    x          c cathode(x)')
        do 17 k=1,nj
          x=h*dfloat(k-1)
          print 480,x,(ccath(idec,i,k),i=1,nspec)
  17    continue
c
        write (7,184),xiav,phistr
  184   format(1x,1pe11.4,1x,e11.4)
        print 168
  168   format(/,
     1    '    x         ian/iavg    icath/iavg    p0mpsa        p0mpsc
     1    etasa    etasc')
        do 152 jx=1,nj
          x=dfloat(jx-1)/dfloat(nj-1)
          print 123,x,xi(1,jx),xi(2,jx),p0mps(1,jx,ndec),
     1      p0mps(2,jx,ndec),etas(1,jx),etas(2,jx)
          write (7,122),x,xi(1,jx),xi(2,jx),p0mps(1,jx,ndec),
     1      p0mps(2,jx,ndec)
  122     format(1x,1pe9.2,1x,1pe11.4,2x,1pe11.4,2x,1pe11.4,2x,1pe11.4)
  123     format(1x,1pe9.2,6(2x,1pe11.4))
  152   continue
c
        return
        end
c
c
        subroutine rstrt(ifine,nc,xiav,phistr,xiold,p0mps)
c
c
        implicit double precision (a-h,o-z)
        dimension xi(2,101),p0mps(2,101,5),coarse(2,101),xiold(2,101),
     1    p0mpso(2,101)
c
        common/geom/pi,hol,nj,ndec,icut
        common/elecs/v(2),ian,icath
c
        nf=nj
        nj=nf
        if(ifine.eq.0) go to 102
        nj=nc
  102   continue
        print 103
  103   format(' restart file')
        read (4,*),xiav,phistr
```

```
  100   format(e11.4,2x,e11.4)
        print 100,xiav,phistr
c
        do 105 jx=1,nj
          read (4,*),x,xi(1,jx),xi(2,jx),p0mps(1,jx,ndec),p0mps(2,jx,ndec)
          print 106,x,xi(1,jx),xi(2,jx),p0mps(1,jx,ndec),
     1      p0mps(2,jx,ndec)
  106     format(1x,1pe9.2,1x,1pe11.4,2x,1pe11.4,2x,1pe11.4,2x,1pe11.4)
          do 104 ielec=ian,icath
          p0mpso(ielec,jx)=p0mps(ielec,jx,ndec)
  104     xiold(ielec,jx)=xi(ielec,jx)
  105   continue
c
        if(ifine.eq.0) go to 107
        print 108
  108   format(' going to finer points')
c
        do 120 ielec=ian,icath
        do 120 jx=1,nc
  120   coarse(ielec,jx)=xi(ielec,jx)
c
        call finer(coarse,nc,xi,nf)
c
        do 130 ielec=ian,icath
        do 130 jx=1,nc
  130   coarse(ielec,jx)=p0mpso(ielec,jx)
c
        call finer(coarse,nc,p0mpso,nf)
c
        do 140 ielec=ian,icath
        do 140 jx=1,nf
          p0mps(ielec,jx,ndec)=p0mpso(ielec,jx)
  140   xiold(ielec,jx)=xi(ielec,jx)
c
  107   call phiint(p0mps)
        nj=nf
c
        return
        end
c
c
        subroutine finer(coarse,nc,fine,nf)
c
        implicit double precision (a-h,o-z)
        dimension coarse(2,101),fine(2,101),xintgd(101)
c
        print 49,nf,nc
   49   format(' in finer, nf=',i2,' nc=',i2)
```

```
c
        do 50 ielec=1,2
        do 50 jx=1,nc
c         print 51,ielec,jx,coarse(ielec,jx)
c  51     format(' elec',i2,' jx=',i2,' coarse=',1pe11.4)
   50   continue
c
        x1 = 0.d+00
        xn = 1.d+00
        h = 1./dfloat(nc-1)
        m = 3
c
        do 100 ielec=1,2
          do 105 jx=1,nc
  105     xintgd(jx)=coarse(ielec,jx)
        do 100 jx=1,nf
          x = dfloat(jx-1)/dfloat(nf-1)
          call interpl(x1,xn,h,nc,xintgd,x,m,yout)
          fine(ielec,jx)=yout
c         print 101,ielec,jx,x,fine(ielec,jx)
c 101     format(' elec',i2,' jx=',i2,' x=',1pe11.4,' fine=',1pe11.4)
  100   continue
c
        return
        end
c
c
        subroutine phiint(p0mps)
c
        implicit double precision (a-h,o-z)
        dimension xintgd(101),p0mps(2,101,5)
c
        common/geom/pi,hol,nj,ndec,icut
        common/elecs/v(2),ian,icath
c
        if(ndec.eq.1) go to 290
        ndm1=ndec-1
        njm1=nj-1
        hc=1./dfloat(njm1)
c
c       use Lagrange interpolation to
c       find p0mps at the Gaussian quadrature points
c
        do 312 ielec=ian,icath
          do 311 jx=1,nj
            xintgd(jx)=p0mps(ielec,jx,ndec)
  311     continue
          do 313 idec=1,ndm1
```

```
            decend=dfloat(icut)**(idec-ndec)
            h=decend/dfloat(njm1)
          do 313 jx=1,nj
            x=dfloat(jx-1)*h
            call interpl(0.d+00,1.d+00,hc,nj,xintgd,x,3,
     1        p0mps(ielec,jx,idec))
  313     continue
  312   continue
c
  290   continue
c
        return
        end
c
c
        subroutine dump(phistr,xiav,xi,p0mps)
c
        implicit double precision (a-h,o-z)
        dimension xi(2,101),p0mps(2,101,5)
        common/geom/pi,hol,nj,ndec,icut
c
        print 139,xiav,phistr
  139   format(' xiav=',1pe11.4,' phistr=',1pe11.4)
c
        print 168
  168   format(/,
     1    '    x        ian/iavg    icath/iavg    p0mpsa        p0mpsc')
        do 152 jx=1,nj
          x=dfloat(jx-1)/dfloat(nj-1)
          print 122,x,xi(1,jx),xi(2,jx),p0mps(1,jx,ndec),
     1      p0mps(2,jx,ndec)
  122     format(1x,1pe9.2,1x,1pe11.4,2x,1pe11.4,2x,1pe11.4,2x,1pe11.4)
  152   continue
c
        return
        end
c
c
        subroutine iavlp (phiin,xiavin,xitotc,xitota,
     1    p0mps,eta,csurf,xij,xi,phistr,xiav,it,tox,fine,can,ccath,
     1    xitotj,toxj,son,etas)
c
        implicit double precision (a-h,o-z)
        logical secset,yloset,yhiset
c
        dimension csurf(2,10,101),tox(2,101),fine(2,5,101),
     1    xij(2,5,101),xi(2,101),p0mps(2,101,5),eta(2,101,5),
     1    can(5,10,101),ccath(5,10,101),xitotj(2,5),toxj(2,101,5),
```

```
     1    etas(2,101)
c
      common/evprob/xlamda(3),ai(3),z,diodr(10)
      common/co/coeff,xn
      common/rxnpars/cref(10),cre(10),porq(2,10,5),ip,iq,uref(5),
     1    nspec,nrxn,one(2,5),rho0
      common/matcom/b(20,20),d(20,41)
      common/kinpars/xj(5),ratson(10,5),ason(2,5),ifor,iback
      common/cdata/cnctol,ftol,xint0(10,101,5),
     1    xint1(10,101,5),n2,nd,isec
      common/elecs/v(2),ian,icath
      common/geom/pi,hol,nj,ndec,icut
      common/gauss/xp(96),gw(96),ngt,xielec(201),
     1    xiothr(201),njmore,xx(201)
      common/nrdata/phicrt,totcrt,crit,vcell
c
  105 format(/,' iavg iteration',i3,' x=',1pe23.16,' y=',1pe23.16)
c 106 format(' xofyhi=',1pe23.16,' xofylo=',1pe23.16,' yhi=',1pe11.4,
c    1    ' ylo=',1pe11.4)
c 114 format(' x=',1pe11.4,
c    1    ' y=',1pe11.4,/,' xbest=',1pe11.4,' ybest=',1pe11.4)
  115 format(' iavlp',/,' x=',1pe11.4,
     1    ' y=',1pe11.4,/,' xsec=',1pe11.4,' ysec=',1pe11.4,' xbest=',
     1    1pe11.4,' ybest=',1pe11.4)
  215 format(' beware y worse than ylo and yhi')
  335 format(' iavlp no convergence in',i3,' iterations',/,
     1    ' x=',1pe11.4,' y=',1pe11.4)
  405 format(/,' converged in',i3,' iterations',/,' x=',1pe11.4,' y=',
     1    1pe11.4)
c
      ycrit = 1.0d-06
      xcrit = 1.0d-06
      itmax = 20
c     itmax = 1
      slope = -1.d+00
      xguess = xiavin
      deltax = xiavin/100.
      phistr = phiin
c
c     --- the following can be initialized to anything since the
c     --- flags will tell whether they have really been intialized
c
      ybest=1.0d+100
      ysec=1.0d+100
      yhi=1.0d+100
      xofyhi=-1.0d+100
      ylo=-1.0d+100
      xofylo=1.0d+100
```

```
        xbest=0.d+00
c
c       --- make sure it knows nothing has been initialized ---
c
        secset = .false.
        yhiset = .false.
        yloset = .false.
c
        xnew = xguess
        iter = 0
c
c
c
 100    iter = iter + 1
        if(iter.gt.itmax) stop
        phiold = phistr
c
c
c       ---- evaluate y and z at xnew ----
c
        call evali(ynew,xnew,phiold,
     1    xitotc,xitota,p0mps,eta,csurf,xij,xi,phistr,it,tox,fine,can,
     1    ccath,xitotj,toxj,son,etas)
c       print 108,phistr
c 108   format(' after call evali, phistr=',1pe11.4)
        print 105,iter,xnew,ynew
c       if(yhiset.and.yloset) print 106,xofyhi,xofylo,yhi,ylo
c       print 114,xnew,ynew,xbest,ybest
c
        if (dabs(ynew).lt.ycrit .or. (dabs(xnew-xbest).le.xcrit))
     1    go to 400
c
c       ---- update best, second-best ----
c
        if (iter .eq. 1) go to 130
          if (dabs(ynew) .lt. dabs(ybest)) go to 120
            if (secset .and. (dabs(ynew).gt.dabs(ysec)) .and.
     1        (ynew*ysec).gt.0.d+00) go to 110
                ysec = ynew
                xsec = xnew
                secset = .true.
                go to 140
  110       continue
            print 315
  315       format(' in iavlp ynew worse than ysec')
            istop=1
            if(istop.ne.0) stop
            if( ynew*ybest .gt. 0.d+00 .and. ynew*ysec .gt. 0.d+00)
```

```
     1        go to 317
              call dump(phistr,xnew,xi,p0mps)
              stop
  317       xnew = 2.*xbest - xnew
            print 115,xnew,ynew,xsec,ysec,xbest,ybest
            if(iter.lt.itmax) go to 100
            print 316,itmax
  316       format(' in iavlp it=itmax=',i2)
            call dump(phistr,xnew,xi,p0mps)
            stop
c
  120     ysec = ybest
          xsec = xbest
          secset = .true.
  130   ybest = ynew
        xbest = xnew
c
  140   continue
c
c       ---- update bounds ----
c
        if ( (ynew .gt. 0.d+00) .or.
     1    ((dabs(ynew) .gt. dabs(ylo)) .and. yloset) ) go to 200
              xmove = slope*deltax
              if(iter.eq.1) print 131,xmove,deltax
  131         format(' xmove=',1pe11.4,' deltax=',1pe11.4)
              ylo = ynew
              yloset = .true.
              xofylo = xnew
  200   if ( (ynew .lt. 0.d+00) .or.
     1    ((dabs(ynew) .gt. dabs(yhi)) .and. yhiset) ) go to 210
              xmove = -slope*deltax
              if(iter.eq.1) print 131,xmove,deltax
              yhi = ynew
              yhiset = .true.
              xofyhi = xnew
c
  210   if ((dabs(ynew) .gt. dabs(ylo)) .and. (dabs(ynew).gt.dabs(yhi))
     1    .and. yhiset .and. yloset ) print 215
c
c
        if (iter .eq. 1) go to 320
c
c         --- try ynew ---
c
          xnew = extrpl(ybest,ysec,xbest,xsec,slope,deltax)
          if(dabs(xnew-xbest).gt.10.d+00) xnew=xbest+10.*(xnew-xbest)/
     1      dabs(xnew-xbest)
```

```
            print 800,xnew,xbest,dabs((xnew-xbest)/xbest)
  800       format(' iavlp at try y,xnew=',1pe11.4,' xbest=',1pe11.4,
     1        /,' rel diff=',1pe11.4)
            if (.not. (yhiset.and.yloset)) go to 330
            if ((xnew-xofyhi) * (xnew-xofylo) .gt. 0.d+00) go to 310
            go to 330
c
c           --- try bisect ---
c
  310       continue
c           print 311
c 311       format(' bisect')
            print 801,xnew,xbest,dabs((xnew-xbest)/xbest)
  801       format(' iavlp at bisect,xnew=',1pe11.4,' xbest=',1pe11.4,
     1        /,' rel diff=',1pe11.4)
            if (.not.(yloset.and.yhiset)) go to 320
            xnew = 0.5 * (xofyhi + xofylo)
            if(dabs(xnew-xbest).gt.10.d+00) xnew=xbest+10.*(xnew-xbest)
     1        /dabs(xnew-xbest)
            go to 330
c
c         --- move x ---
c
  320     xnew = xnew + xmove
          print 321,xmove
  321     format(' move x, xmove=',1pe11.4)
c
  330     if (iter .lt. itmax) go to 100
          print 335,itmax,xnew,ynew
          call dump(phistr,xnew,xi,p0mps)
          print 318
  318     format(' too many iterations in iavlp')
          stop
c
  400     print 405,iter,xnew,ynew
c 400     continue
          xiav = xnew
c
c         temp
          iprt=0
          call newt(xiav,phistr,p0mps,eta,csurf,xij,xi,xitotc,xitota,
     1      tox,fine,can,ccath,iprt,xitotj,toxj,son,etas)
          return
          end
c
c
          function extrpl(yb,ys,xb,xs,slope,deltax)
c
```

```
      implicit double precision (a-h,o-z)
      if ((yb .eq. ys) .or. (xb .eq. xs)) go to 100
            extrpl = xb - yb*(xb-xs)/(yb-ys)
c           print 10,extrpl,(yb-ys),(xb-xs)
c  10       format(' extrpl=',1pe23.16,' dy=',1pe11.4,' dx=',
c    1        1pe11.4)
            return
  100   if (yb .gt. 0.d+00) extrpl = xb - slope*deltax
      if (yb .le. 0.d+00) extrpl = xb + slope*deltax
c     print 20,extrpl
c 20  format(' decided to step, extrpl=',1pe23.16)
c
      return
      end
c
c
c
      subroutine evali(yfcn,xiavin,phiin,
     1  xitotc,xitota,p0mps,eta,csurf,xij,xi,phistr,it,tox,fine,can,
     1  ccath,xitotj,toxj,son,etas)
c
      implicit double precision (a-h,o-z)
      dimension csurf(2,10,101),tox(2,101),fine(2,5,101),
     1  xij(2,5,101),xi(2,101),p0mps(2,101,5),eta(2,101,5),
     1  can(5,10,101),ccath(5,10,101),xitotj(2,5),toxj(2,101,5),
     1  etas(2,101)
c
      common/evprob/xlamda(3),ai(3),z,diodr(10)
      common/co/coeff,xn
      common/rxnpars/cref(10),cre(10),porq(2,10,5),ip,iq,uref(5),
     1  nspec,nrxn,one(2,5),rho0
      common/matcom/b(20,20),d(20,41)
      common/kinpars/xj(5),ratson(10,5),ason(2,5),ifor,iback
      common/cdata/cnctol,ftol,xint0(10,101,5),
     1  xint1(10,101,5),n2,nd,isec
      common/elecs/v(2),ian,icath
      common/geom/pi,hol,nj,ndec,icut
      common/gauss/xp(96),gw(96),ngt,xielec(201),
     1  xiothr(201),njmore,xx(201)
      common/nrdata/phicrt,totcrt,crit,vcell
c
      call philoop(phiin,xiavin,xitotc,xitota,p0mps,eta,csurf,
     1  xij,xi,phistr,it,tox,fine,can,ccath,xitotj,toxj,son,etas)
c     print 108,phistr
c 108 format(' after call philoop, phistr=',1pe11.4)
      yfcn = xitota - xiavin
c
      return
```

```
      end
c
c
c
      subroutine philoop (phiin,xiavin,xitotc,xitota,
     1   p0mps,eta,csurf,xij,xi,phistr,it,tox,fine,can,ccath,
     1   xitotj,toxj,son,etas)
c
c
c     ---- This program searches for the zero of a monotonic function
c     ---- It also allows for an alternate function to be used to
c     ---- speed the convergence
c
      implicit double precision (a-h,o-z)
      logical secset,setz(3),yloset,yhiset,blowup
c     logical usez
c
      integer best,secbst
c
      dimension csurf(2,10,101),tox(2,101),fine(2,5,101),
     1   xij(2,5,101),xi(2,101),p0mps(2,101,5),eta(2,101,5),
     1   can(5,10,101),ccath(5,10,101),xitotj(2,5),toxj(2,101,5),
     1   etas(2,101)
c
      common/evprob/xlamda(3),ai(3),z,diodr(10)
      common/co/coeff,xn
      common/rxnpars/cref(10),cre(10),porq(2,10,5),ip,iq,uref(5),
     1   nspec,nrxn,one(2,5),rho0
      common/matcom/b(20,20),d(20,41)
      common/kinpars/xj(5),ratson(10,5),ason(2,5),ifor,iback
      common/cdata/cnctol,ftol,xint0(10,101,5),
     1   xint1(10,101,5),n2,nd,isec
      common/elecs/v(2),ian,icath
      common/geom/pi,hol,nj,ndec,icut
      common/gauss/xp(96),gw(96),ngt,xielec(201),
     1   xiothr(201),njmore,xx(201)
      common/nrdata/phicrt,totcrt,crit,vcell
      common/diagn/iterp
c
      data best/1/, secbst/2/, new/3/
c
  105 format(/,' phistr iteration',i3,' x=',1pe23.16,' y=',1pe23.16)
c 106 format(' xofyhi=',1pe23.16,' xofylo=',1pe23.16,' yhi=',1pe11.4,
c    1   ' ylo=',1pe11.4)
c 114 format(' x=',1pe11.4,
c    1   ' y=',1pe11.4,/,' xbest=',1pe11.4,' ybest=',1pe11.4)
  115 format(' philoop',/,' x=',1pe11.4,
     1   ' y=',1pe11.4,/,' xsec=',1pe11.4,' ysec=',1pe11.4,' xbest=',
```

```
     1    1pe11.4,' ybest=',1pe11.4)
  215    format(' beware y worse than ylo and yhi')
  335    format(' philoop no convergence in',i3,' iterations',/,
     1    ' x=',1pe11.4,' y=',1pe11.4)
  405    format(/,' converged in',i3,' iterations',/,' x=',1pe11.4,' y=',
     1    1pe11.4)
c
        ycrit = 1.0d-06
        xcrit = 1.0d-08
        itmax = 40
c       itmax = 1
        slope = -1.d+00
        xguess = phiin
        deltax = 2.0d+00
c       usez = .true.
        ipass=0
c       slopeo=0.d+00
c
        ybest=1.0d+100
        ysec=1.0d+100
        yhi=1.0d+100
        xofyhi=-1.0d+100
        ylo=-1.0d+100
        xofylo=1.0d+100
        xbest=0.d+00
c
c       --- make sure it knows nothing has been initialized ---
c
        secset = .false.
        yhiset = .false.
        yloset = .false.
c
        do 20 i=best,new
  20    setz(i) = .false.
c
c
        xnew = xguess
        iterp = 0
c
c
c
 100    iterp = iterp + 1
        if(iterp.gt.itmax) stop
c
c
c       ---- evaluate y and z at xnew ----
c
        call evalp(ynew,znew,setz(new),xnew,xiavin,
```

```
     1    xitotc,xitota,p0mps,eta,csurf,xij,xi,tox,fine,can,ccath,
     1    xitotj,toxj,son,etas)
          print 105,iterp,xnew,ynew
          print *,'xitota',xitota,'xitotc',xitotc
c         print *,'  jx    ian       icath'
c         do 402 jjx=1,nj
c           print 401,jjx,xi(1,jjx),xi(2,jjx)
c 401       format(1x,i2,1x,1pe11.4,1x,e11.4)
c 402     continue
c         print 106,xofyhi,xofylo,yhi,ylo
c         print 114,xnew,ynew,xbest,ybest
c
          if (dabs(ynew).lt.ycrit .or. (dabs(xnew-xbest).le.xcrit))
     1    go to 400
c
c         ---- update best, second-best ----
c
          if (iterp .eq. 1) go to 130
            if (dabs(ynew) .lt. dabs(ybest)) go to 120
              if (secset .and. (dabs(ynew).gt.dabs(ysec)) .and.
     1          (ynew*ysec).gt.0.d+00) go to 110
                  ysec = ynew
                  xsec = xnew
                  secset = .true.
                  setz(secbst) = setz(new)
                  go to 140
  110         continue
c 110         print 115,xnew,ynew,xsec,ysec,xbest,ybest
              print 315
  315         format(' in philoop ynew worse than ysec')
              if( ynew*ybest .gt. 0.d+00 .and. ynew*ysec .gt. 0.d+00 )
     1            go to 317
                call dump(xnew,xiavin,xi,p0mps)
                stop
  317         xnew = 2.*xbest - xnew
              print 115,xnew,ynew,xsec,ysec,xbest,ybest
              if(iterp.lt.itmax) go to 100
              print 316,itmax
  316         format(' in philop it=itmax=',i2)
              call dump(xnew,xiavin,xi,p0mps)
              stop
c
  120       ysec = ybest
            xsec = xbest
            secset = .true.
            setz(secbst) = setz(new)
  130     ybest = ynew
          xbest = xnew
```

```
        setz(best) = setz(new)
c
  140   continue
c
c       ---- update bounds ----
c
c       print 141,ynew,ylo,yhi
c 141   format(' ynew=',1pe11.4,' ylo,yhi=',1pe11.4,1x,e11.4)
c
        if ( (ynew .gt. 0.d+00) .or.
     1    ((dabs(ynew) .gt. dabs(ylo)) .and. yloset) ) go to 200
              xmove = slope*deltax
              ylo = ynew
              yloset = .true.
              xofylo = xnew
  200   if ( (ynew .lt. 0.d+00) .or.
     1    ((dabs(ynew) .gt. dabs(yhi)) .and. yhiset) ) go to 210
              xmove = -slope*deltax
              yhi = ynew
              yhiset = .true.
              xofyhi = xnew
c
  210   if ((dabs(ynew) .gt. dabs(ylo)) .and. (dabs(ynew).gt.dabs(yhi))
     1    .and. yhiset .and. yloset ) print 215
c
c
        if (iterp .eq. 1) go to 320
c
c         --- try znew ---
c
c       temp
c
        go to 310
c
c
c
c         --- try ynew ---
c
  300     xnew = extrpl(ybest,ysec,xbest,xsec,slope,deltax)
c         print 306,xnew,xofyhi,xofylo,xbest,xsec
c 306     format(' at try y, extrpl=',1pe23.16,/,' xofyhi=',
c    1      1pe23.16,' xofylo=',1pe23.16,/,' xbest=',1pe11.4,' xsec=',
c    1      1pe11.4)
          if(blowup(nrxn,p0mps,xiavin,xnew)) print 341
  341     format(' blows up at try y')
          if(blowup(nrxn,p0mps,xiavin,xnew)) go to 310
          if(dabs(xnew-xbest).gt.10.d+00) xnew=xbest+10.*(xnew-xbest)
     1          /dabs(xnew-xbest)
```

```
          if (.not. (yhiset.and.yloset)) go to 330
          if ((xnew-xofyhi) * (xnew-xofylo) .gt. 0.d+00) go to 310
                go to 330
c
c         --- try bisect ---
c
  310     continue
          if(ybest*ysec .le. 0.d+00) ipass=1
          if(ipass.ne.0) go to 300
          if(.not.(yhiset.and.yloset)) go to 320
          xnew = 0.5 * (xofyhi + xofylo)
          if(dabs(xnew-xbest).gt.10.d+00) xnew=xbest+10.*(xnew-xbest)
     1      /dabs(xnew-xbest)
c
          print 800,xnew,xbest,dabs((xnew-xbest)/xbest)
  800     format(' philoop at bisect,xnew=',1pe11.4,' xbest=',1pe11.4,
     1      /,' rel diff=',1pe11.4)
          if(blowup(nrxn,p0mps,xiavin,xnew)) print 312,xnew
  312     format(' blows up at bisect, xnew=',1pe11.4)
          if(.not. blowup(nrxn,p0mps,xiavin,xnew)) go to 330
c
             call dump(phistr,xiavin,xi,p0mps)
             print 319
  319        format(' blowup in philoop')
             stop
c
c       --- move x ---
c
  320   xnew = xnew + xmove
c       print 321,xnew
c 321   format(' move x,xnew=',1pe11.4)
        if(.not.blowup(nrxn,p0mps,xiavin,xnew)) go to 330
        xnew = xbest + xmove
c       print 321,xnew
        if(blowup(nrxn,p0mps,xiavin,xnew)) print 322
  322   format(' blows up at move x')
        if(.not. blowup(nrxn,p0mps,xiavin,xnew)) go to 330
             call dump(phistr,xiavin,xi,p0mps)
             print 319
             stop
c
  330   if (iterp .lt. itmax) go to 100
        print 335,itmax,xnew,ynew
        call dump(xnew,xiavin,xi,p0mps)
        stop
c
  400   print 405,iterp,xnew,ynew
c 400   continue
```

```
      phistr = xnew
      return
      end
c
c
      subroutine evalp(yfcn,zfcn,zinrng,phiin,xiavin,
     1  xitotc,xitota,p0mps,eta,csurf,xij,xi,tox,fine,can,ccath,
     1  xitotj,toxj,son,etas)
c
      implicit double precision (a-h,o-z)
      logical zinrng
c
      dimension csurf(2,10,101),tox(2,101),fine(2,5,101),
     1  xij(2,5,101),xi(2,101),p0mps(2,101,5),eta(2,101,5),
     1  can(5,10,101),ccath(5,10,101),xitotj(2,5),toxj(2,101,5),
     1  etas(2,101)
c
      common/evprob/xlamda(3),ai(3),z,diodr(10)
      common/co/coeff,xn
      common/rxnpars/cref(10),cre(10),porq(2,10,5),ip,iq,uref(5),
     1  nspec,nrxn,one(2,5),rho0
      common/matcom/b(20,20),d(20,41)
      common/kinpars/xj(5),ratson(10,5),ason(2,5),ifor,iback
      common/cdata/cnctol,ftol,xint0(10,101,5),
     1  xint1(10,101,5),n2,nd,isec
      common/elecs/v(2),ian,icath
      common/geom/pi,hol,nj,ndec,icut
      common/gauss/xp(96),gw(96),ngt,xielec(201),
     1  xiothr(201),njmore,xx(201)
      common/nrdata/phicrt,totcrt,crit,vcell
      common/diagn/iterp
c
      print *,' at call newt, iterp=',iterp
      iprt=0
      call newt(xiavin,phiin,p0mps,eta,csurf,xij,xi,xitotc,xitota,
     1  tox,fine,can,ccath,iprt,xitotj,toxj,son,etas)
c
c     print 102,xitotc,xitota
c 102 format(' xitotc=',1pe11.4,' xitota=',1pe11.4)
c
c
c     print 103
c 103 format(' converged concentrations ',
c    1  /' anode        cathode         anode       cathode')
c     do 105 jx=1,nj
c       print 104,csurf(ian,1,jx),csurf(icath,1,jx),csurf(ian,2,jx),
c    1    csurf(icath,2,jx)
c 104   format(1x,1pe11.4,1x,e11.4,3x,e11.4,1x,e11.4)
```

```
c 105   continue
c
        yfcn = xitota + xitotc
        zfcn = yfcn
        zinrng = .true.
c
        return
        end
c
c
        logical function blowup(nrxn,p0mps,xiavin,phistr)
c
        implicit double precision (a-h,o-z)
        common/geom/pi,hol,nj,ndec,icut
        common/kinpars/xj(5),ratson(10,5),ason(2,5),ifor,iback
        common/elecs/v(2),ian,icath
c
        dimension p0mps(2,101,5)
c
c       print 100,phistr
c 100   format(' in blowup, phistr=',1pe11.4)
        expmax=2.303d+00*290.d+00
        blowup=.false.
c
        do 270 idec=1,ndec
        do 270 jx=1,nj
        do 270 ielec=ian,icath
          eta=v(ielec)-(p0mps(ielec,jx,idec)*xiavin+phistr)
c
          do 271 j=1,nrxn
c           if(idec.eq.1 .and. jx.eq.1) print 101,ason(ifor,j)*eta,
c    1         ason(iback,j)*eta
c 101       format(' in blowup, alp f eta=',1pe11.4,/,' alp b eta=',
c    1         1pe11.4)
            if(dabs(ason(ifor,j)*eta) .lt. expmax .or.
     1         dabs(ason(iback,j)*eta) .lt. expmax)
     1         go to 271
c
            blowup=.true.
            return
c
  271     continue
c
  270   continue
c
        return
        end
c
```

```
c
      subroutine primry(iprimc,ielec,xi,x,icertx,xix,xiint,xiend)
c
c     This subroutine calculates the primary distribution in a
c     channel flow cell (except at the end points)
c
c     Input variables
c
c            pi,hol,nj
c
c            ian,icath
c
c     Output variables
c
c            xi(ielec,jx) = normalized current density on electrode
c                    ielec at location jx
c
      implicit double precision (a-h,o-z)
      dimension xi(2,101),sign(2),iprimc(2),xiint(2,96,5)
c
      common/elecs/v(2),ian,icath
      common/geom/pi,hol,nj,ndec,icut
      common/gauss/xp(96),gw(96),ngt,xielec(201),
     1   xiothr(201),njmore,xx(201)
c
      sign(ian)=1.d+00
      sign(icath)=-1.d+00
      xi(ielec,1)=xiend*sign(ielec)
      xi(ielec,nj)=xi(ielec,1)*sign(ielec)
c
      a0=1.3862944d+00
      a1=0.1119723d+00
      a2=0.0725296d+00
      b0=0.5d+00
      b1=0.1213478d+00
      b2=0.0288729d+00
c
      eps=pi/(2.*hol)
      xm1=1.-(dtanh(eps))**2
      njm1=nj-1
      term=dlog( (dexp(2.*eps) + 2. + dexp(-2.*eps)) / 4.)
c
      xk = a0 + a1*xm1 + a2*xm1*xm1 + (b0 + b1*xm1 + b2*xm1*xm1)*
     1    term
c
      xnum=eps*dcosh(eps)/xk
c
      if(icertx.ne.0) go to 101
```

```
c
        do 100 jx=2,njm1
          x=dfloat(jx-1)/dfloat(njm1)
  100     xi(ielec,jx)=sign(ielec)*xnum/dsqrt((dsinh(eps))**2 -
     1      (dsinh((2.*x-1.)*eps))**2)
c
        go to 102
c
  101   xix=xnum/dsqrt((dsinh(eps))**2 - (dsinh((2.*x-1.)*eps))**2)
  102   if(iprimc(1).ne.0 .and. iprimc(2).ne.0) return
        denom=dfloat(njm1)
        hold=0.d+00
c
        do 141 idec=1,ndec
          h=dfloat(icut)**(idec-ndec)/denom
          do 149 jg=1,ngt
            x=xp(jg)*(njm1*h - hold) + hold
            xiint(ielec,jg,idec)=sign(ielec)*xnum/
     1        dsqrt((dsinh(eps))**2 - (dsinh((2.*x-1.)*eps))**2)
  149     continue
          hold=h*njm1
  141   continue
c
        return
        end
c
c
        subroutine intcxi(nspec,xint0,xint1)
c
c       This subroutine calculates xint0(i,n-k+1,idec), 1/dx times the
c       integral from (k-1)dx to kdx of dC/dxi, where C is the
c       solution to the asymmetric Graetz problem in channel flow and
c       xi is the dimensionless normal variable, y\h.
c
c       Input variables
c               nspec = number of species
c
c               xlamda,ai,z,diodr
c
c               nj,ndec
c
c       Output variables
c               xint0(i,n-k+1,idec)     (described above)
c               xint1(i,n-k+1,idec)
c
        implicit double precision (a-h,o-z)
        dimension xint0(10,101,5),xint1(10,101,5),array(5)
c
```

```
      common/evprob/xlamda(3),ai(3),z,diodr(10)
      common/geom/pi,hol,nj,ndec,icut
c
      a=1.35659745d+00
      b=0.2d+00
      c=0.060733452d+00
c
      aa=0.9594d+00
      bb=0.6069d+00
      cc=0.4512d+00
      dd=0.276d+00
c
      njm1=nj-1
      n=njm1
      h=dfloat(icut)**(1-ndec)/dfloat(njm1)
c
      do 1000 idec=1,ndec
      do 999 ixi=1,2
      do 999 i=1,nspec
      do 999 k=1,n
        nmk=n-k
        nmkp1=n-k+1
        den=z/(h*diodr(i))
        zeta=nmk/den
        zetall=(nmkp1)/den
        if(ixi .eq. 2) go to 500
c
c       xi=0
c
        if(zeta.lt.0.11d+00 .and. zetall.gt.0.11d+00) go to 600
        if(zeta.ge.0.11d+00) go to 100
        xint0(i,nmkp1,idec)=1.5*den**(1./3.)*a*
     1    (nmk**(2./3.)-(nmk+1)**(2./3.))
     1    + b -0.75*den**(-1./3.)*c*(nmk**(4./3.)-(nmk+1)**(4./3.))
        go to 999
c
  100   sum=0.0d+00
        do 110 j=1,3
  110   sum=sum+dabs(ai(j))/xlamda(j) * (dexp(-xlamda(j)*zeta) - dexp
     1    (-xlamda(j)*zetall))
        xint0(i,nmkp1,idec)=-1. - 2.*den*sum
        go to 999
c
  600   xint0(i,nmkp1,idec)=1.5*den**(1./3.)*a*(nmk**(2./3.) -
     1    (0.11*den)
     1    **(2./3.)) + b*(0.11*den-nmk) - 0.75*den**(-1./3.)*c*
     1    (nmk**(4./3.) - (0.11*den)**(4./3.))
        sum=0.0d+00
```

```
          do 610 j=1,3
  610     sum=sum+dabs(ai(j))/xlamda(j) * (dexp(-xlamda(j)*0.11) -
     1      dexp(-xlamda(j)*zetall))
          sum=0.11*den - nmkpl - 2.*den*sum
          xint0(i,nmkpl,idec)=xint0(i,nmkpl,idec)+sum
          go to 999
c
c         xi=1
c
  500     if(zeta.le.0.18d+00 .and. zetall.ge.0.18d+00) go to 620
          if(zeta.gt.0.18d+00) go to 510
          f0=0.d+00
          if(n.eq.k) go to 505
          f0=-dexp(aa - bb/zeta -cc*dexp(-dd/zeta))
  505     xint1(i,nmkpl,idec)=(-dexp(aa - bb/zetall -
     1      cc*dexp(-dd/zetall)) + f0)/2.
          go to 999
c
  510     continue
          sum=0.d+00
          do 520 j=1,3
  520     sum=sum+ai(j)/xlamda(j)*(dexp(-xlamda(j)*zeta) -
     1      dexp(-xlamda(j)*zetall))
          xint1(i,nmkpl,idec)=-1.+2.*den*sum
          go to 999
c
  620     z11=0.18d+00
          f0=0.d+00
          if(n.eq.k) go to 640
          f0=-dexp(aa - bb/zeta - cc*dexp(-dd/zeta))
  640     xint1(i,nmkpl,idec)=(-dexp(aa - bb/z11 -
     1     cc*dexp(-dd/z11)) + f0)/2.
     1      *(-nmk+z11*den)
          sum=0.0d+00
          do 660 j=1,3
  660     sum=sum+ai(j)/xlamda(j) * (dexp(-xlamda(j)*0.18) - dexp(
     1      -xlamda(j)*zetall))
          sum=2.*den*sum + z11*den-nmkpl
          xint1(i,nmkpl,idec)=xint1(i,nmkpl,idec) + sum
  999     continue
          h=h*dfloat(icut)
 1000   continue
c
        return
        end
c
c
        subroutine oldint(ielec,nj,yarray,x,y)
```

```
c
      implicit double precision (a-h,o-z)
      common/elecs/v(2),ian,icath
c
      dimension yarray(nj),ydum(101)
c
      njm1=nj-1
      h=1./dfloat(njm1)
      x1=0.d+00
      xn=1.d+00
c
c     unrefined:
c
      al=0.3d+00
      ar=0.2d+00
      if(ielec.eq.icath) ar=0.5d+00
c
      eps1 = (al/yarray(1))**2
      eps2 = (ar/yarray(nj))**2
c
      do 100 jx=1,nj
        xx=dfloat(jx-1)/dfloat(njm1)
        ydum(jx) = yarray(jx) * dsqrt( (xx+eps1)*(1.-xx+eps2) )
  100 continue
c
      call interp1(x1,xn,h,nj,ydum,x,3,yout)
      y = yout / dsqrt( (x+eps1)*(1.-x+eps2) )
c
      return
      end
c
c
      subroutine interp1(x1,xn,h,n,y,x,m,yout)
c
c     subroutine to interpolate in a table of uniformly spaced values.
c      parameters are -
c       x1,xn     beginning and ending x values
c       h         delta x - the uniform spacing
c       n         number of entries in the table
c       y         array of function values
c       x         value at which y is to be interpolated
c       m         degree of interpolating polynomial.  The subroutine
c                   will handle up to 10th degree, but usually the
c                   degree will be less than 10 to avoid round-off
c                   errors.
c       yout      the interpolated y value returned to the caller.
c
c       An array d is used in the subroutine to hold the delta y's.
```

```
c
c       This subroutine was taken from Curtis F. Gerald, Applied
c       Numerical Analysis, Second Edition, Addison Wesley, 1978.
c
        implicit double precision (a-h,o-z)
        dimension y(n),d(10)
c
c       First find proper subscript for y0 so that x is centered in
c         the domain as well as possible.  This subscript value is j.
c
        fm = m + 1
        j = (x - x1)/h - fm/2. + 2.
        if (x .le. x1 + fm/2.*h) j = 1
        if (x .ge. xn - fm/2.*h) j = n - m
        fj = j
        x0 = x1 + (fj - 1.)*h
        y0 = y(j)
c
c       compute the differences that are needed
c
        do 10 i=1,m
          d(i) = y(j+1) - y(j)
          j = j + 1
   10   continue
c
        if (m .eq. 1) go to 25
c
        do 20 j=2,m
          do 15 i=j,m
            k = m - i + j
            d(k) = d(k) - d(k-1)
   15     continue
   20   continue
c
c       compute s value
c
   25   s = (x - x0)/h
c
c       compute interpolated y value
c
        yout = y0
        fnum = s
        den = 1.d+00
c
        do 30 i=1,m
          fi = i
          yout = yout + fnum/den*d(i)
          fnum = fnum*(s - fi)
```

```
          den = den*(fi + 1.)
   30   continue
c
        return
        end
c
c
        subroutine extrap(n,x,f,xin,fout)
c
c       This subroutine uses a Lagrange formula to extrapolate a
c       function
c
c       Input variables
c
c               n = number of points to be used in extrapolation
c               x = array of values at which function is given
c               f = array of function values
c               xin = value at which function is to be extrapolated
c
c       Output variables
c
c               fout = value of function at x=xin
c
        implicit double precision (a-h,o-z)
        dimension x(n),f(n)
c
        fout=0.d+00
c
        do 100 i=1,n
          if(xin.eq.x(i)) go to 999
          prodn=1.d+00
          prodd=1.d+00
          do 110 j=1,n
            if(i.eq.j) go to 110
            prodn=prodn*(xin-x(j))
            prodd=prodd*(x(i)-x(j))
  110     continue
  100   fout=fout + f(i)*prodn/prodd
c
        go to 120
c
  999   print 1000,xin
 1000   format(' wrong input for xin, xin=',1pe11.4)
c
  120   return
        end
c
c
```

```
      SUBROUTINE MATINV(N,M,DETERM)
c
c       This subroutine solves a matrix equation of the form BX=D.
c       The answer X is put into the first column of D.  Note that
c       D should be dimensioned nxm, where m=2n+1 and that all
c       columns of D (except the first) should be initiallized with
c       zeros
c
c       Input variables
c               n = dimension of square b matrix
c               m = 2n+1
c               b = matrix in equation bx=d
c               d : first column contains vector d in bx=d
c
c       Output variables
c               d : first column contains vector x in bx=d
c               determ = 0. if matrix has zero determinant
c
        implicit double precision (a-h,o-z)
      DIMENSION ID(20)
c
        common/matcom/b(20,20),d(20,41)
c
      DETERM=1.0
      DO 1 I=1,N
    1 ID(I)=0
      DO 18 NN=1,N
      BMAX=1.1
      DO 6 I=1,N
      IF(ID(I).NE.0) GOTO 6
      BNEXT=0.0
      BTRY=0.0
      DO 5 J=1,N
      IF(ID(J).NE.0) GOTO 5
      IF(ABS(B(I,J)).LE.BNEXT) GOTO 5
      BNEXT=ABS(B(I,J))
      IF(BNEXT.LE.BTRY) GOTO 5
      BNEXT=BTRY
      BTRY=ABS(B(I,J))
      JC=J
    5 CONTINUE
      IF(BNEXT.GE.BMAX*BTRY) GOTO 6
      BMAX=BNEXT/BTRY
      IROW=I
      JCOL=JC
    6 CONTINUE
      IF(ID(JC).EQ.0) GOTO 8
      DETERM=0.0
```

```
      RETURN
    8 ID(JCOL)=1
      IF(JCOL.EQ.IROW) GOTO 12
      DO 10 J=1,N
      SAVE=B(IROW,J)
      B(IROW,J)=B(JCOL,J)
   10 B(JCOL,J)=SAVE
      DO 11 K=1,M
      SAVE=D(IROW,K)
      D(IROW,K)=D(JCOL,K)
   11 D(JCOL,K)=SAVE
   12 F=1.0/B(JCOL,JCOL)
      DO 13 J=1,N
   13 B(JCOL,J)=B(JCOL,J)*F
      DO 14 K=1,M
   14 D(JCOL,K)=D(JCOL,K)*F
      DO 18 I=1,N
      IF(I.EQ.JCOL) GO TO 18
      F=B(I,JCOL)
      DO 16 J=1,N
   16 B(I,J)=B(I,J)-F*B(JCOL,J)
      DO 17 K=1,M
   17 D(I,K)=D(I,K)-F*D(JCOL,K)
   18 CONTINUE
      RETURN
      END
c
c
      subroutine newt(xiav,phistr,p0mps,eta,csurf,xij,xi,xitotc,
     1   xitota,tox,fine,can,ccath,iprt,xitotj,toxj,son,etas)
c
c     run with routines
c         curnt
c         iother
c         fcns
c         matinv
c         bigtrm
c         simps
c
c     This subroutine solves for c<i,cath>(x) and c<i,an>(x) using
c     a multidimensional Newton-Raphson method.  The equations to be
c     solved result from equating the Fick's-Law and Faraday's-Law
c     expressions for the flux.  The Fick's-Law expression
c     incorporates the method of Acrivos and Chambre to discretize
c     the integral equation (Ind. Eng. Chem., 49, 1025(1957))
c
c     Input variables
c             isec = 0 if tertiary current distribution, nonzero
```

```
c                       if secondary
c               xiav = i<avg> (-n<m>/s<Rm>) FL/(RT K<b>)
c               phistr = (phi *) (-n<m> F/(s<Rm> RT))
c               p0mps(ielec,jx,idec) = (phi 0 - phi *) K<b> / (i<avg> L)
c               csurf (input at jx=1 only)
c
c               diodr
c
c               coeff
c
c               nj,ndec
c
c               cref,porq,ip,iq,uref,nspec,nrxn,one
c
c               xj,ratson,ason,ifor,iback
c
c               cnctol,ftol,xint0,xint1,n2,nd
c
c               v,ian,icath
c
c       Intermediate variables in common
c
c               b,d
c
c       Output variables
c
c               eta(ielec,jx,idec) = (V<ielec> - phi 0) *
c                       (-n<m> F/(s<Rm> RT))
c               csurf(ielec,i,jx) = dimensionless surface concentration
c                       of species i on electrode ielec at mesh point jx
c               can(idec,i,jx) = dimensionless anode surface conc
c                       of species i at mesh point jx in decade idec
c               ccath(idec,i,jx) = dimensionless cathode surface conc
c                       of species i at mesh point jx in decade idec
c               xij(ielec,j,jx) = -n<m>/s<R,m> (FL/(RT K<b>)) i<j,elec>
c                       = partial current density of reaction j on
c                       electrode ielec at location jx
c               xi(ielec,jx) = current density on electrode ielec at
c                       mesh point jx
c               xitotc = integral of current along cathode
c               xitota = integral of current along anode
c
c
      implicit double precision (a-h,o-z)
      logical limcur(2)
      dimension csurf(2,10,101),eta(2,101,5),fs(20),tf(2,5,101),
     1   tb(2,5,101),big(4),p0mps(2,101,5),xij(2,5,101),xi(2,101),
     1   xintgd(101),xitot(2),save(2),sum0(2,10),sum1(2,10),tox(2,101),
```

```
     1    fine(2,5,101),can(5,10,101),xitotj(2,5),savej(2,5),savem(2),
     1    toxj(2,101,5),ccath(5,10,101),etas(2,101),
     1    farpot(2),savdum(2)
c
        common/evprob/xlamda(3),ai(3),z,diodr(10)
        common/co/coeff,xn
        common/geom/pi,hol,nj,ndec,icut
        common/rxnpars/cref(10),cre(10),porq(2,10,5),ip,iq,uref(5),
     1    nspec,nrxn,one(2,5),rho0
        common/matcom/b(20,20),d(20,41)
        common/kinpars/xj(5),ratson(10,5),ason(2,5),ifor,iback
        common/cdata/cnctol,ftol,xint0(10,101,5),
     1    xint1(10,101,5),n2,nd,isec
        common/elecs/v(2),ian,icath
        common/diagn/iterp
c
c
        istop=0
c
        limcur(ian)=.true.
        limcur(icath)=.true.
        rtonf=8.3143*298.*son/96487.
        if(iprt.ne.0) print 158
 158    format(' big printout goes here')
        icount = 0
        n2p1=n2+1
        expmax=2.303d+00*290.d+00
c
c       damping factor if concentrations go negative
c
        cdamp=1.0d-06
c
c       calculate V - phi 0
c
        do 270 idec=1,ndec
        do 270 jx=1,nj
        do 270 ielec=ian,icath
          eta(ielec,jx,idec)=v(ielec)-(p0mps(ielec,jx,idec)*xiav+phistr)
c
          if(dabs(eta(ielec,jx,idec)*rtonf).lt.2.5d0)
     1      limcur(ielec)=.false.
c
          do 271 j=1,nrxn
            if(dabs(ason(ifor,j)*eta(ielec,jx,idec)) .lt. expmax .or.
     1         dabs(ason(iback,j)*eta(ielec,jx,idec)) .lt. expmax)
     1         go to 271
c
            call dump(phistr,xiav,xi,p0mps)
```

```
            print 276
  276       format(' stopped in newt because exp blows up')
            stop
c
  271     continue
c
  270   continue
c
        if(limcur(ian)) print *,' limiting current assumed on anode'
        if(limcur(icath)) print *,' limiting current assumed on cath'
c       if(iprt.ne.0) print 370
c 370   format(' eta anode')
c       if(iprt.ne.0) print 371,((eta(ian,jx,ndec)),jx=1,nj)
        if(iprt.ne.0) print 470
  470   format(' rtonf eta anode')
        if(iprt.ne.0) print 371,((eta(ian,jx,ndec)*rtonf),jx=1,nj)
  371   format(5e12.4)
c       if(iprt.ne.0) print 372
c 372   format(' eta cathode')
c       if(iprt.ne.0) print 371,((eta(icath,jx,ndec)),jx=1,nj)
        if(iprt.ne.0) print 472
  472   format(' rtonf eta cathode')
        if(iprt.ne.0) print 371,((eta(icath,jx,ndec)*rtonf),jx=1,nj)
        if(istop.ne.0) stop
c
        if(limcur(ian) .or. limcur(icath)) call newtlc(xiav,phistr,
     1    p0mps,
     1    eta,csurf,xij,xi,xitotc,
     1    xitota,tox,fine,can,ccath,iprt,xitotj,toxj,son,limcur,etas)
        if(limcur(ian) .or. limcur(icath)) return
c
c
c       get currents at first mesh point
c
c
        jx=1
        idec = ndec
        do 376 ielec=ian,icath
  376   call curnt(ielec,csurf,eta,tf,tb,xij,xi,jx,idec,etas)
        if(iprt.ne.0) print 375,jx,xi(ian,1),xi(icath,1)
  375   format(' jx=',i2,' ian=',1pe11.4,' icath=',1pe11.4)
c
        kend=(nj-1)/icut + 1
        njm1=nj-1
c
        do 260 ielec=ian,icath
          tox(ielec,1)=0.d+00
          xitot(ielec)=0.d+00
```

```
          farpot(ielec) = 0.d0
        do 260 j=1,nrxn
          toxj(ielec,1,j)=0.d+00
  260   xitotj(ielec,j)=0.d+00
c
        jbegin=2
        do 115 idec=1,ndec
c
c       temp
c
c         print 116,idec
c 116     format(' input for decade',i2,/,
c    1      '   can(1)         ccath     can(2)        ccath')
c         do 118 jx=1,nj
c           print 117,csurf(ian,1,jx),csurf(icath,1,jx),csurf(ian,2,jx),
c    1        csurf(icath,2,jx)
c 117       format(1x,1pe11.4,1x,e11.4,1x,e11.4,1x,e11.4)
c 118     continue
c
          h=dfloat(icut)**(idec-ndec)/dfloat(njm1)
c
          do 170 jx=jbegin,nj
c
          if(jx.eq.2) go to 179
          jxm2 = jx - 2
          jxm1 = jx - 1
          if(jx.eq.3) go to 184
c
          do 180 i=1,nspec
            csurf(ian,i,jx) = 2.*csurf(ian,i,jxm1) - csurf(ian,i,jxm2)
            csurf(icath,i,jx) = dexp( 2.*dlog(csurf(icath,i,jxm1)) -
     1        dlog(csurf(icath,i,jxm2)) )
            if(csurf(ian,i,jx).le.0.d+00) csurf(ian,i,jx) = dexp( 2.*
     1        dlog(csurf(ian,i,jxm1)) - dlog(csurf(ian,i,jxm2)) )
              istop=0
              if(csurf(ian,i,jx).le.0.d+00 .or.
     1          csurf(icath,i,jx).le.0.d+00) istop=1
              if(istop.eq.1) print 181,jx,idec,csurf(ian,i,jx),
     1          csurf(icath,i,jx)
  181         format(' at jx=',i2,' decade',i2,' can,cath=',1pe11.4,1x,
     1          e11.4)
              if(istop.ne.1) go to 180
                call dump(phistr,xiav,xi,p0mps)
                print 183
  183           format(' stopped in newt because predicted conc neg.')
                stop
  180     continue
          go to 179
```

```
  184     do 185 i=1,nspec
          do 185 ielec=ian,icath
  185     csurf(ielec,i,jx) = csurf(ielec,i,jxml)
c
  179     continue
c
            do 172 i=1,nspec
c
              do 173 ielec=ian,icath
                sum0(ielec,i)=0.d+00
  173           suml(ielec,i)=0.d+00
c
              if(jx.eq.2) go to 172
              n=jx-1
              nml=n-1
c
              do 171 k=1,nml
                kpl=k+1
                nmkpl=n-k+1
              do 171 ielec=ian,icath
                sum0(ielec,i) = sum0(ielec,i) + (csurf(ielec,i,kpl)
     1            -csurf(ielec,i,k))*xint0(i,nmkpl,idec)
  171           suml(ielec,i) = suml(ielec,i) + (csurf(ielec,i,kpl)
     1            -csurf(ielec,i,k))*xintl(i,nmkpl,idec)
c
  172         continue
c
  177         continue
c           print 178,jx,sum0(ian,1),sum0(icath,1),suml(ian,1),
c    1        suml(icath,1)
c 178       format(' jx=',i2,' sum0 an,cath=',1pe11.4,1x,e11.4,
c    1        ' suml an,cath=',1pe11.4,1x,e11.4)
c
            iexit=0
            itmax = 40
c
            do 130 iter=1,itmax
c
c
c             initiallize current to zero for summing
c
              if(isec.eq.0.d+00)  go to 377
              do 378 ielec=ian,icath
  378         call curnt(ielec,csurf,eta,tf,tb,xij,xi,jx,idec,etas)
  377         if(iprt.ne.0 .and. isec.ne.0) print 375,jx,xi(ian,jx),
     1            xi(icath,jx)
              if(isec.ne.0) go to 170
              do 50 jcol=1,n2
```

```
            do 50 irow=1,n2
              b(irow,jcol)=0.d+00
   50       d(irow,jcol)=0.d+00
            do 51 jcol=n2p1,nd
            do 51 irow=1,n2
   51       d(irow,jcol)=0.d+00
c
            ibefor=0
            if(iexit.ne.0) ibefor=1
            call fcns(sum0,sum1,idec,csurf,jx,eta,fs,b,big,xij,xi,etas)
c
c           if(idec.eq.ndec) print 375,jx,xi(ian,jx),xi(icath,jx)
c
c
            do 100 irow=1,n2
  100       d(irow,1)=-fs(irow)
  112       format(4(1x,e11.4))
c
          call matinv(n2,nd,determ)
c
c
          if(determ.ne.0.d+00) go to 119
          print 110,idec,jx
  110     format(27h zero determinant in matinv,' idec=',i2,' jx=',i2)
          do 121 i=1,n2
            print 113,(b(i,j),j=1,n2)
  113       format(' b matrix',1x,1pe11.4,3(1x,e11.4))
  121     continue
          stop
c
  119     iexit=0
          do 120 i=1,nspec
          do 120 ielec=ian,icath
            ishift=0
            if(ielec.eq.icath) ishift=nspec
            irow=i+ishift
            if((d(irow,1)+csurf(ielec,i,jx)).le.0.d+00) print *,
     1        ' d+csurf.le.0, go to 150'
            if((d(irow,1)+csurf(ielec,i,jx)).le.0.d+00) go to 150
            csurf(ielec,i,jx)=d(irow,1)+csurf(ielec,i,jx)
c           if((dabs(d(irow,1)/csurf(ielec,i,jx)).ge.cnctol .and.
c    1        csurf(ielec,i,jx).gt.cnctol)) print *,
c    1        ' d/c.ge.cnctol, go to 219,d,csurf',d(irow,1),
c    1        csurf(ielec,i,jx)
c           if((dabs(d(irow,1)/csurf(ielec,i,jx)).ge.cnctol .and.
c    1        csurf(ielec,i,jx).le.cnctol)) print *,' warning in newt'
            if((dabs(d(irow,1)/csurf(ielec,i,jx)).ge.cnctol .and.
     1        csurf(ielec,i,jx).gt.cnctol)) go to 219
```

```
            go to 120
  150       csurf(ielec,i,jx)=csurf(ielec,i,jx)*cdamp
c           if((dabs(d(irow,1)/csurf(ielec,i,jx)).ge.cnctol)) go to 219
  219       iexit=1
  120       continue
c
c           make sure it converges by N-R method and not by damping
c
c           if(iexit.ne.0) print *,' iexit.ne.0',iexit
c           if(ibefor.ne.0) print *,' ibefor.ne.0',ibefor
            if(iexit.ne.0 .or. ibefor.ne.0) go to 130
c
c       make sure that original equation is satisfied
c
          call bigtrm(big,biggst)
c
c       dangerous
c
          if(ftol*biggst.le.1.d-10) go to 163
c
          do 160 irow=1,n2
c         if(dabs(fs(irow)).gt.ftol*biggst) print *,
c    1      ' fs.gt.ftol*biggst at irow',irow,fs(irow),ftol*biggst
          if(dabs(fs(irow)).gt.ftol*biggst) go to 130
  160     continue
c
  163     continue
          icount = icount + iter
c
          if(iter.ge.20) print 162,iter,idec,jx
  162     format(' newt took',i3,' iterations for dec',i2,' jx=',i2)
          if(iter.ge.20 .and. jx.gt.3) stop
          go to 170
  130   continue
c
        if(itmax.ne.1) print 140,itmax,idec,jx,csurf(ian,1,jx),
     1    csurf(icath,1,jx)
  140   format(' no convergence in newt after',i4,
     1    ' iterations at decade',i2,'jx=',i3,' can,cath=',1pe11.4,
     1    1x,e11.4)
        if(itmax.eq.1) go to 170
          call dump(phistr,xiav,xi,p0mps)
          stop
c
  170   continue
c
c       call totcurr(jbegin,h,xi,savdum,tox)
        call gaussq(1,xi,jbegin,idec,save(ian),save(icath),tox)
```

```
        call gaussq(2,xi,jbegin,idec,savem(ian),savem(icath),tox)
        print *,'decade',idec,'savem',savem(1)
c
        do 473 ielec=ian,icath
          xitot(ielec)=xitot(ielec)+save(ielec)
          farpot(ielec)=farpot(ielec)-savem(ielec)/(2.*hol)
          do 474 jx=jbegin,nj
            tox(ielec,jx) = tox(ielec,jx) + tox(ielec,jbegin-1)
  474     continue
  473   continue
c
        do 15 k=1,nj
        do 15 ielec=ian,icath
          fine(ielec,idec,k)=xi(ielec,k)
   15   continue
c
        do 16 i=1,nspec
        do 16 k=1,nj
          ccath(idec,i,k) = csurf(icath,i,k)
  16    can(idec,i,k)= csurf(ian,i,k)
c
        if(idec.eq.ndec) go to 115
c
        do 12 k=1,kend
          jj = k*icut - icut + 1
        do 12 ielec=ian,icath
          xi(ielec,k) = xi(ielec,jj)
          tox(ielec,k) = tox(ielec,jj)
          etas(ielec,k) = etas(ielec,jj)
          tox(ielec,k) = tox(ielec,jj)
          do 14 j=1,nrxn
            toxj(ielec,k,j)=toxj(ielec,jj,j)
   14     xij(ielec,j,k)=xij(ielec,j,jj)
        do 12 i=1,nspec
   12   csurf(ielec,i,k)=csurf(ielec,i,jj)
c
c
        jbegin=kend+1
  115   continue
c
        xitotc=xitot(icath)
        xitota=xitot(ian)
        print *,'far potential',farpot(1)
c
c       print 116,icount
c 116   format(' total newt iterations',i5)
c
        return
```

```
      end
c
c
      subroutine bigtrm(big,biggst)
c
      implicit double precision (a-h,o-z)
      dimension big(4)
c
      biggst=0.d+00
      do 100 i=1,4
        absb=dabs(big(i))
        if(absb.gt.biggst) biggst=absb
  100 continue
c
      return
      end
c
c
      subroutine fcns(sum0,sum1,idec,csurf,jx,eta,fs,dfdcs,big,xij,xi,
     1   etas)
c
c     run with
c           curnt
c           f
c           dfdc
c           iother
c           irow
c           for
c           back
c           delfcn
c
c     This subroutine calculates the functions that are needed to
c     solve for the surface concentrations, and it calculates the
c     partial current densities for the given guessed surface
c     concentrations
c
c     Input variables
c
c         idec = index for which decade
c         csurf(ielec,i,jx) = dimensionless surface concentration
c             of species i on electrode ielec at mesh point jx
c         jx = current mesh point
c         eta(ielec,jx,idec) = (V<ielec> - phi 0) (-n<m> F/(s<Rm> RT))
c
c         diodr
c
c         coeff
c
```

```
c           cref,porq,ip,iq,uref,nspec,nrxn,one
c
c           xj,ratson,ason,ifor,iback
c
c           xint0,xint1,n2
c
c           ian,icath
c
c       Output variables
c
c           fs(irow) = dimensionless equation for Fick's Law flux minus
c               Faraday's Law flux
c           dfdcs(irow,jcol) = derivative of f with respect to
c               dimensionless surface concentrations
c           big(1-4) = terms in equation f
c           xij(ielec,j,jx) = -n<m>/s<R,m> (FL/(RT K<b>)) i<j,ielec>
c
        implicit double precision (a-h,o-z)
        logical same
c
        common/evprob/xlamda(3),ai(3),z,diodr(10)
        common/co/coeff,xn
        common/rxnpars/cref(10),cre(10),porq(2,10,5),ip,iq,uref(5),
     1    nspec,nrxn,one(2,5),rho0
        common/kinpars/xj(5),ratson(10,5),ason(2,5),ifor,iback
        common/cdata/cnctol,ftol,xint0(10,101,5),
     1    xint1(10,101,5),n2,nd,isec
        common/elecs/v(2),ian,icath
        common/diagn/iterp
c
        dimension csurf(2,10,101),eta(2,101,5),fs(20),etas(2,101),
     1    dfdcs(20,20),prod(2),big(4),xij(2,5,101),sum0(2,10),
     1    sum1(2,10),tf(2,5,101),tb(2,5,101),bignew(4),xi(2,101)
c
        n=jx-1
        npl=jx
c
        do 90 ib=1,4
   90   big(ib) = 0.d+00
c
        do 91 ielec=ian,icath
   91   call curnt(ielec,csurf,eta,tf,tb,xij,xi,jx,idec,etas)
c
        do 100 ielec=ian,icath
        do 100 i=1,nspec
          call fsub(n,npl,ielec,i,jx,idec,csurf,sum0,sum1,
     1      tf,tb,f,bignew)
          do 105 ib=1,4
```

```
  105     if (dabs(bignew(ib)) .gt. dabs(big(ib))) big(ib) = bignew(ib)
  100   fs(irow(i,ielec)) = f
c
        do 110 ielecr=ian,icath
        do 110 ielecc=ian,icath
        do 110 ir=1,nspec
        do 110 ic=1,nspec
        dfdcs(irow(ir,ielecr),irow(ic,ielecc)) = dfdc(ir,ic,ielecr,
     1    ielecc,jx,idec,csurf,tf,tb)
c       print 111,irow(ir,ielecr),irow(ic,ielecc),dfdcs(irow(ir,ielecr),
c    1     irow(ic,ielecc))
c 111   format(1x,' dfdcs(',i2,',',i2,')=',1pe11.4)
  110   continue
c
        return
        end
c
c
        function dfdc(ir,ic,ielecr,ielecc,jx,idec,csurf,tf,tb)
c
c       run with
c               iother
c               delfcn
c
        implicit double precision (a-h,o-z)
        common/evprob/xlamda(3),ai(3),z,diodr(10)
        common/co/coeff,xn
        common/cdata/cnctol,ftol,xint0(10,101,5),
     1    xint1(10,101,5),n2,nd,isec
        common/rxnpars/cref(10),cre(10),porq(2,10,5),ip,iq,uref(5),
     1    nspec,nrxn,one(2,5),rho0
        common/kinpars/xj(5),ratson(10,5),ason(2,5),ifor,iback
c
        dimension csurf(2,10,101),tf(2,5,101),tb(2,5,101)
c
        iothr = iother(ielecr)
c
        sumj = 0.d+00
        if(ielecr.ne.ielecc) go to 99
          do 110 j=1,nrxn
            sumj = sumj + ratson(ir,j)*(tf(ielecr,j,jx)*porq(ip,ic,j)/
     1        csurf(ielecr,ic,jx) - tb(ielecr,j,jx)*porq(iq,ic,j)/
     1        csurf(ielecr,ic,jx))
c           if(j.eq.1) print *,' f=',tf(ielecr,j,jx)*porq(ip,ic,j)/
c    1        csurf(ielecr,ic,jx),' b=',tb(ielecr,j,jx)*porq(iq,ic,j)/
c    1        csurf(ielecr,ic,jx)
  110     continue
   99     dfdc = delfcn(ir,ic)*coeff*diodr(ir)*(-xint0(ir,1,idec)*
```

```
     1      delfcn(ielecr,ielecc) + xintl(ir,1,idec)*
     1      delfcn(iothr,ielecc))
c         print 112,dfdc,delfcn(ir,ic),delfcn(ielecr,ielecc),
c    1      delfcn(iothr,ielecc)
c 112     format(1x,' dfdc=',1pe11.4,' del(ir,ic)',1pe11.4,/,
c    1      ' del(ielecr,ielecc)',1pe11.4,' del(iothr,ielecc)',
c    1      1pe11.4)
          dfdc = dfdc + sumj
  100   continue
c
c       print *,' ir',ir,' ic',ic,' iothr',iothr,' ielecc',ielecc
c       print 111,jx,ielecr,ielecc,ir,ic,sumj,dfdc
c 111   format(' jx=',i2,' ielecr=',i2,' ielecc=',i2,' ir',i2,' ic',i2,
c    1    ' sumj=',1pe11.4,' dfdc=',1pe11.4)
c
        return
        end
c
c
        function delfcn(i,j)
c
        implicit double precision (a-h,o-z)
          delfcn=0.d+00
          if (i.eq.j) delfcn=1.
c
        return
        end
c
c
        subroutine newtlc(xiav,phistr,p0mps,eta,csurf,xij,xi,xitotc,
     1    xitota,tox,fine,can,ccath,iprt,xitotj,toxj,son,limcur,etas)
c
c       subroutine newt for limiting current
c

        implicit double precision (a-h,o-z)
        logical limcur(2)
        dimension csurf(2,10,101),eta(2,101,5),fs(20),tf(2,5,101),
     1    tb(2,5,101),big(4),p0mps(2,101,5),xij(2,5,101),xi(2,101),
     1    xintgd(101),xitot(2),save(2),sum0(2,10),sum1(2,10),tox(2,101),
     1    fine(2,5,101),can(5,10,101),xitotj(2,5),savej(2,5),
     1    toxj(2,101,5),ccath(5,10,101),etas(2,101)
c
        common/evprob/xlamda(3),ai(3),z,diodr(10)
        common/co/coeff,xn
        common/geom/pi,hol,nj,ndec,icut
        common/rxnpars/cref(10),cre(10),porq(2,10,5),ip,iq,uref(5),
     1    nspec,nrxn,one(2,5),rho0
```

```
      common/matcom/b(20,20),d(20,41)
      common/kinpars/xj(5),ratson(10,5),ason(2,5),ifor,iback
      common/cdata/cnctol,ftol,xint0(10,101,5),
     1   xint1(10,101,5),n2,nd,isec
      common/elecs/v(2),ian,icath
      common/diagn/iterp
c
c
      if(iprt.ne.0) print 158
  158 format(' big printout goes here')
      icount = 0
      n2p1=n2+1
      expmax=2.303d+00*290.d+00
c
c     damping factor if concentrations go negative
c
      cdamp=1.0d-06
c
c     get currents at first mesh point
c
      jx=1
      idec = ndec
      do 374 ielec=ian,icath
  374 call curnt(ielec,csurf,eta,tf,tb,xij,xi,jx,idec,etas)
      if(iprt.ne.0) print 375,jx,xi(ian,1),xi(icath,1)
  375 format(' jx=',i2,' ian=',1pe11.4,' icath=',1pe11.4)
c
      kend=(nj-1)/icut + 1
      njm1=nj-1
c
      do 260 ielec=ian,icath
        tox(ielec,1)=0.d+00
        xitot(ielec)=0.d+00
      do 260 j=1,nrxn
        toxj(ielec,1,j)=0.d+00
  260 xitotj(ielec,j)=0.d+00
c
      jbegin=2
      do 115 idec=1,ndec
c
        h=dfloat(icut)**(idec-ndec)/dfloat(njm1)
c
        do 170 jx=jbegin,nj
c
        if(limcur(ian).and.son.gt.0.) go to 116
          go to 117
  116   ielim=ian
        iolim=icath
```

```
  117     if(limcur(icath).and.son.lt.0.) go to 118
            go to 123
  118     ielim=icath
          iolim=ian
  123     continue
          do 124 i=1,nspec
            if(ratson(i,1).le.0.) go to 124
            limsp=i
            csurf(ielim,i,2)=0.
  124     continue
c
c
c         if((limcur(ian).and.son.lt.0.) .or. (limcur(icath).and.
c    1      son.gt.0.)) print *,' newtlc not appropriate'
c         if((limcur(ian).and.son.lt.0.) .or. (limcur(icath).and.
c    1      son.gt.0.)) stop
c
          if(jx.eq.2) go to 179
          jxm2 = jx - 2
          jxm1 = jx - 1
          if(jx.eq.3) go to 186
c
          do 180 i=1,nspec
            do 184 ielec=ian,icath
  184       csurf(ielec,i,jx) = 2.*csurf(ielec,i,jxm1) -
     1        csurf(ielec,i,jxm2)
            if(i.eq.limsp) csurf(ielim,i,jx) = 0.
            if(csurf(iolim,i,jx).le.0.d+00) csurf(iolim,i,jx) =

Job ZAC2 (queue CSA4_NORMAL, entry 36) completed
     1        dexp( 2.*
     1        dlog(csurf(iolim,i,jxm1)) - dlog(csurf(iolim,i,jxm2)) )
              istop=0
              if(csurf(iolim,i,jx).le.0.d+00) istop=1
              if(istop.eq.1) print 181,jx,idec,csurf(iolim,i,jx),
     1          csurf(ielim,i,jx)
  181         format(' at jx=',i2,' decade',i2,
     1          ' cothr,celec=',1pe11.4,1x,e11.4)
              if(istop.ne.1) go to 180
                call dump(phistr,xiav,xi,p0mps)
                print 183
  183           format(' stopped in newtlc because predicted conc neg.')
                stop
  180     continue
          go to 179
  186     do 185 i=1,nspec
          do 185 ielec=ian,icath
  185     csurf(ielec,i,jx) = csurf(ielec,i,jxm1)
```

```
c
  179     continue
CHANNEL.FOR                                                    298

c
            do 172 i=1,nspec
c
              do 173 ielec=ian,icath
                sum0(ielec,i)=0.d+00
  173           sum1(ielec,i)=0.d+00
c
              if(jx.eq.2) go to 172
              n=jx-1
              nm1=n-1
c
              do 171 k=1,nm1
                kp1=k+1
                nmkp1=n-k+1
              do 171 ielec=ian,icath
                sum0(ielec,i) = sum0(ielec,i) + (csurf(ielec,i,kp1)
     1            -csurf(ielec,i,k))*xint0(i,nmkp1,idec)
  171           sum1(ielec,i) = sum1(ielec,i) + (csurf(ielec,i,kp1)
     1            -csurf(ielec,i,k))*xint1(i,nmkp1,idec)
c
  172       continue
c
c
          iexit=0
          itmax = 40
c         itmax = 10
c
          do 130 iter=1,itmax
c
c
c           initiallize current to zero for summing
c
            if(isec.eq.0.d+00) go to 216
              do 217 ielec=ian,icath
  217           call curnt(ielec,csurf,eta,tf,tb,xij,xi,jx,idec,etas)
  216         if(iprt.ne.0 .and. isec.ne.0) print 375,jx,xi(ian,jx),
     1            xi(icath,jx)
            if(isec.ne.0) go to 170
            do 50 jc=1,n2
            do 50 ir=1,n2
              b(ir,jc)=0.d+00
   50       d(ir,jc)=0.d+00
            do 51 jc=n2p1,nd
            do 51 ir=1,n2
   51       d(ir,jc)=0.d+00
c
```

```
            ibefor=0
            if(iexit.ne.0) ibefor=1

            if(iterp.eq.4 .and. jx.eq.2) print *,'call fcnslc(lims,io)'
            call fcnslc(limsp,iolim,sum0,sum1,idec,csurf,jx,eta,fs,b,
     1        big,xij,xi,etas)
c
            n2m1 = n2 - 1
            ndm1 = 2*n2m1 + 1
c
            do 100 irowi=1,n2m1
c            if(jx.eq.2) print *,' fs(irowi)',irowi,fs(irowi)
  100       d(irowi,1)=-fs(irowi)
c
            if(nspec.eq.1) d(irow(1,iolim),1)=-fs(irow(1,iolim))/
     1        b(irow(1,iolim),irow(1,iolim))
            if(nspec.eq.1) go to 119
            if(iterp.eq.4) print *,' before call matinv, fs'
            if(iterp.eq.4) print 112,(fs(irowi),irowi=1,n2)
  112       format(4(1x,e11.4))
c
            if(iterp.eq.4) print *,' before call matinv'
            do 122 i=1,n2
              if(iterp.eq.4) print 112,(b(i,j),j=1,n2)
  122       continue
              if(iterp.eq.4) print *,' can',(csurf(ian,i,jx),i=1,nspec)
              if(iterp.eq.4) print *,' ccath',(csurf(icath,i,jx),
     1          i=1,nspec)
              call matinv(n2m1,ndm1,determ)
c             if(jx.eq.2) print *,' d',(d(i,1),i=1,3)
              if(determ.ne.0.d+00) go to 119
              print 110,idec,jx
  110         format(27h zero determinant in matinv,' idec=',i2,
     1        ' jx=',i2)
              do 121 i=1,n2
                print 113,(b(i,j),j=1,n2)
  113           format(' b matrix',1x,1pe11.4,3(1x,e11.4))
  121         continue
              stop
c
  119     iexit=0
          do 120 i=1,nspec
          do 120 ielec=ian,icath
            if(i.eq.limsp .and. ielec.eq.ielim) go to 120
            ishift=0
            if(ielec.eq.icath) ishift=nspec
            irsm=i+ishift
            if(irsm.ge.irow(limsp,ielim)) irsm = irsm - 1
c           if(ielec.eq.icath .and. i.eq.2) print *,' irsm',irsm
```

```
            if((d(irsm,1)+csurf(ielec,i,jx)).le.0.d+00) go to 150
            csurf(ielec,i,jx)=d(irsm,1)+csurf(ielec,i,jx)
CHANNEL.FOR                                                          300

c           if((dabs(d(irsm,1)/csurf(ielec,i,jx)).ge.cnctol)) go to 219
c           if(dabs(d(irsm,1)/csurf(ielec,i,jx)).ge.cnctol .and.
c    1        csurf(ielec,i,jx).gt.cnctol) print *,
c    1        ' d/c.ge.cnctol, go to 219,d,csurf',d(irsm,1),
c    1        csurf(ielec,i,jx)
c           if(dabs(d(irsm,1)/csurf(ielec,i,jx)).ge.cnctol .and.
c    1        csurf(ielec,i,jx).le.cnctol) print *,' warning in newtlc'
            if((dabs(d(irsm,1)/csurf(ielec,i,jx)).ge.cnctol .and.
     1        csurf(ielec,i,jx).gt.cnctol)) go to 219
            go to 120
  150       csurf(ielec,i,jx)=csurf(ielec,i,jx)*cdamp
c           if((dabs(d(ir,1)/csurf(ielec,i,jx)).ge.cnctol)) go to 219
  219       iexit=1
  120       continue
c
            if(iterp.eq.4) print *,' newtlc iteration',iter,' decade',
     1        idec,' jx',jx
            if(iterp.eq.4) print *,' can',(csurf(ian,i,jx),i=1,nspec)
            if(iterp.eq.4) print *,' ccath',(csurf(icath,i,jx),
     1        i=1,nspec)
c
c           make sure it converges by N-R method and not by damping
c
c           print *,' iexit,ibefor',iexit,ibefor
            if(iexit.ne.0 .or. ibefor.ne.0) go to 130
c
c       make sure that original equation is satisfied
c
          call bigtrm(big,biggst)
c
c       dangerous
c
          if(ftol*biggst.le.1.d-10) go to 163
c
          irsm = irow(1,iolim)
          if(irsm.ge.irow(limsp,ielim)) irsm = irsm - 1
          if(dabs(fs(irsm)).gt.ftol*biggst) print *,' fs',fs(irsm),
     1      'ftol*biggst',ftol*biggst
          if(dabs(fs(irsm)).gt.ftol*biggst) go to 130
c
  163     continue
c
          icount = icount + iter
c
          if(iter.ge.20) print 162,iter,idec,jx
  162     format(' newtlc took',i3,' iterations for dec',i2,' jx=',i2)
```

```
          if(iter.ge.20 .and. jx.gt.3) stop
          go to 170
CHANNEL.FOR                                                    301

  130   continue
c
        if(itmax.ne.1) print 140,itmax,idec,jx,csurf(ian,1,jx),
     1    csurf(icath,1,jx)
  140   format(' no convergence in newtlc after',i4,
     1    ' iterations at decade',i2,'jx=',i3,' can,cath=',1pe11.4,
     1    1x,e11.4)
        if(itmax.eq.1) go to 170
          call dump(phistr,xiav,xi,p0mps)
          stop
c
  170   continue
c
        print *,' x      ian    icath'
        print 571,((dfloat(jx-1)*h,xi(1,jx),xi(2,jx)),
     1    jx=jbegin,nj)
  571   format(3(1pe11.4,1x))
        call totcurr(jbegin,h,xi,save,tox)
c
        do 298 j=1,nrxn
          call totcj(jbegin,h,xij,savej,toxj,j)
  298   continue
c
        do 473 ielec=ian,icath
          xitot(ielec)=xitot(ielec)+save(ielec)
          do 575 j=1,nrxn
            xitotj(ielec,j) = xitotj(ielec,j) + savej(ielec,j)
  575     continue
          do 474 jx=jbegin,nj
            tox(ielec,jx) = tox(ielec,jx) + tox(ielec,jbegin-1)
            do 574 j=1,nrxn
  574       toxj(ielec,jx,j) = toxj(ielec,jx,j)+toxj(ielec,jbegin-1,j)
  474     continue
  473   continue
c
        do 15 k=1,nj
        do 15 ielec=ian,icath
          fine(ielec,idec,k)=tox(ielec,k)
   15   continue
c
        do 16 i=1,nspec
        do 16 k=1,nj
          ccath(idec,i,k) = csurf(icath,i,k)
  16    can(idec,i,k) = csurf(ian,i,k)
c
        if(idec.eq.ndec) go to 115
```

```
c
        do 12 k=1,kend
CHANNEL.FOR                                                          302

          jj = k*icut - icut + 1
        do 12 ielec=ian,icath
          xi(ielec,k) = xi(ielec,jj)
          etas(ielec,k) = etas(ielec,jj)
          tox(ielec,k) = tox(ielec,jj)
          do 14 j=1,nrxn
            toxj(ielec,k,j)=toxj(ielec,jj,j)
   14     xij(ielec,j,k)=xij(ielec,j,jj)
        do 12 i=1,nspec
   12   csurf(ielec,i,k)=csurf(ielec,i,jj)
c
c
        jbegin=kend+1
  115   continue
c
        xitotc=xitot(icath)
        xitota=xitot(ian)
c
        return
        end
c
c
        subroutine fcnslc(limsp,iolim,sum0,sum1,idec,csurf,jx,eta,fs,
     1    dfdcs,big,xij,xi,etas)
c
c       subroutine fcns for limiting current
c
        implicit double precision (a-h,o-z)
        logical same
c
        common/evprob/xlamda(3),ai(3),z,diodr(10)
        common/co/coeff,xn
        common/rxnpars/cref(10),cre(10),porq(2,10,5),ip,iq,uref(5),
     1    nspec,nrxn,one(2,5),rho0
        common/kinpars/xj(5),ratson(10,5),ason(2,5),ifor,iback
        common/cdata/cnctol,ftol,xint0(10,101,5),
     1    xint1(10,101,5),n2,nd,isec
        common/elecs/v(2),ian,icath
        common/diagn/iterp
c
        dimension csurf(2,10,101),eta(2,101,5),fs(20),etas(2,101),
     1    dfdcs(20,20),prod(2),big(4),xij(2,5,101),sum0(2,10),
     1    sum1(2,10),tf(2,5,101),tb(2,5,101),bignew(4),xi(2,101)
c
        ielim=iother(iolim)
        n=jx-1
```

```
        npl=jx
c
CHANNEL.FOR                                                    303

        do 90 ib=1,4
   90   big(ib) = 0.d+00
c
        if(iterp.eq.4 .and. jx.eq.2) print *,' call curnt(iolim)'
        call curnt(iolim,csurf,eta,tf,tb,xij,xi,jx,idec,etas)
c
          if(iterp.eq.4.and.jx.eq.2) print *,' call fsub(ielim,limsp)',
     1      ' tf',tf(1,1,jx),tf(1,2,jx),' tb',tb(1,1,jx),tb(1,2,jx)
          call fsub(n,npl,ielim,limsp,jx,idec,csurf,sum0,suml,tf,tb,f,
     1      bignew)
c           if(jx.eq.2) print *,' ielim, limsp'
c           if(jx.eq.2) print 378,jx,(bignew(ii),ii=1,4),bignew(1)
c    1         +bignew(2),bignew(3)+bignew(4)
c 378       format(' jx',i2,' big(i)=',4e12.4,/,' 1+2,3+4',2e12.4)
c       print *,'limsp',limsp
        do 115 j=1,nrxn
c       print *,' j',j,' ratson',ratson(limsp,1)
          xij(ielim,j,jx)=-(bignew(1)+bignew(2))/ratson(limsp,1)
  115   xi(ielim,jx)=xij(ielim,j,jx)
        if(iterp.eq.4 .and. jx.eq.2) print *,' tf,tb change:'
        do 95 j=1,nrxn
          tf(ielim,j,jx)=xij(ielim,j,jx)
          tb(ielim,j,jx)=0.
   95   continue
c
        do 100 i=1,nspec
        do 100 ielec=ian,icath
          if(i.eq.limsp .and. ielec.eq.ielim) go to 100
          call fsub(n,npl,ielec,i,jx,idec,csurf,sum0,suml,
     1      tf,tb,f,bignew)
          if(iterp.eq.4.and.jx.eq.2) print *,' call fsub, ielec',
     1      ielec,' species',i
c           if(jx.eq.2) print 378,jx,(bignew(ii),ii=1,4),bignew(1)
c    1         +bignew(2),bignew(3)+bignew(4)
          do 105 ib=1,4
  105     if (dabs(bignew(ib)) .gt. dabs(big(ib))) big(ib) = bignew(ib)
          ismall = irow(i,ielec)
          if(ismall.ge.irow(limsp,ielim)) ismall = ismall - 1
          fs(ismall) = f
  100   continue
c
c
        do 110 ielecr=ian,icath
        do 110 ielecc=ian,icath
        do 110 ir=1,nspec
        do 110 ic=1,nspec
```

```
          irsm = irow(ir,ielecr)
          icsm = irow(ic,ielecc)
          if(irsm.eq.irow(limsp,ielim) .or. icsm.eq.irow(limsp,ielim))
     1      go to 110
          if(irsm.ge.irow(limsp,ielim)) irsm = irsm - 1
          if(icsm.ge.irow(limsp,ielim)) icsm = icsm - 1
          dfdcs(irsm,icsm) = dfdclc(ielim,ir,ic,ielecr,
     1      ielecc,jx,idec,csurf,tf,tb)
  110   continue
c
        return
        end
c
c
        subroutine curnt(ielec,csurf,eta,tf,tb,xij,xi,jx,idec,etas)
c
c       run with
c              for
c              back
c
        implicit double precision (a-h,o-z)
        common/rxnpars/cref(10),cre(10),porq(2,10,5),ip,iq,uref(5),
     1    nspec,nrxn,one(2,5),rho0
        common/kinpars/xj(5),ratson(10,5),ason(2,5),ifor,iback
          common/elec3/v(2),ian,icath
          common/diag./iterp
c
        dimension tf(2,5,101),tb(2,5,101),xij(2,5,101),csurf(2,10,101),
     1    eta(2,101,5),xi(2,101),etas(2,101)
c
        xi(ielec,jx) = 0.d+00
c
        do 100 j=1,nrxn
c
          call for(csurf,eta,ielec,j,jx,idec,tf(ielec,j,jx),etas)
          call back(csurf,eta,ielec,j,jx,idec,tb(ielec,j,jx),etas)
          xij(ielec,j,jx) = tf(ielec,j,jx) - tb(ielec,j,jx)
          xi(ielec,jx) = xi(ielec,jx) + xij(ielec,j,jx)
c
  100   continue
c
        return
        end
c
c
        subroutine for(csurf,eta,ielec,j,jx,idec,tf,etas)
c
        implicit double precision (a-h,o-z)
```

```
      common/rxnpars/cref(10),cre(10),porq(2,10,5),ip,iq,uref(5),
     1   nspec,nrxn,one(2,5),rho0
CHANNEL.FOR                                                    305

      common/kinpars/xj(5),ratson(10,5),ason(2,5),ifor,iback
      common/diagn/iterp
c
      dimension csurf(2,10,101),eta(2,101,5),etas(2,101)
c
      prod = 1.d+00
c
      do 120 k=1,nspec
        if(csurf(ielec,k,jx).eq.0. .and. porq(ip,k,j).eq.0.) go to 120
        if(csurf(ielec,k,jx).ne.0.) go to 118
        prod = 0.
        go to 120
  118   if(cref(k).lt.1.0d-15) go to 119
        prod = prod*(csurf(ielec,k,jx)/cref(k))**porq(ip,k,j)
        go to 120
  119   prod = prod*(csurf(ielec,k,jx)/rho0)**porq(ip,k,j)
  120 continue
c
      if(j.eq.1) etas(ielec,jx) =
     1   eta(ielec,jx,idec)-uref(j)
      eterm=dexp(ason(ifor,j)*(eta(ielec,jx,idec)-uref(j)))
      tf = one(ielec,j)*prod*xj(j)*eterm
c
      return
      end
c
c
      subroutine back(csurf,eta,ielec,j,jx,idec,tb,etas)
c
      implicit double precision (a-h,o-z)
      common/rxnpars/cref(10),cre(10),porq(2,10,5),ip,iq,uref(5),
     1   nspec,nrxn,one(2,5),rho0
      common/kinpars/xj(5),ratson(10,5),ason(2,5),ifor,iback
      common/diagn/iterp
c
      dimension csurf(2,10,101),eta(2,101,5),etas(2,101)
c
      prod = 1.d+00
c
      do 120 k=1,nspec
        if(csurf(ielec,k,jx).eq.0. .and. porq(iq,k,j).eq.0.) go to 120
        if(csurf(ielec,k,jx).ne.0.) go to 118
        prod = 0.
        go to 120
  118   if(cref(k).lt.1.0d-15) go to 119
        prod = prod*(csurf(ielec,k,jx)/cref(k))**porq(iq,k,j)
```

```
          go to 120
  119     prod = prod*(csurf(ielec,k,jx)/rho0)**porq(iq,k,j)
CHANNEL.FOR                                                    306

  120   continue
c
        if(j.eq.1) etas(ielec,jx) =
     1    eta(ielec,jx,idec)-uref(j)
        eterm=dexp(-ason(iback,j)*(eta(ielec,jx,idec)-uref(j)))
        tb = one(ielec,j)*prod*xj(j)*eterm
c
        return
        end
c
c
        subroutine fsub(n,npl,ielec,i,jx,idec,csurf,sum0,suml,tf,tb,
     1    f,big)
c
c       run with
c               iother
c
        implicit double precision (a-h,o-z)
        common/evprob/xlamda(3),ai(3),z,diodr(10)
        common/co/coeff,xn
        common/cdata/cnctol,ftol,xint0(10,101,5),
     1    xintl(10,101,5),n2,nd,isec
        common/rxnpars/cref(10),cre(10),porq(2,10,5),ip,iq,uref(5),
     1    nspec,nrxn,one(2,5),rho0
        common/kinpars/xj(5),ratson(10,5),ason(2,5),ifor,iback
        common/diagn/iterp
c
        dimension csurf(2,10,101),tf(2,5,101),tb(2,5,101),sum0(2,10),
     1    suml(2,10),big(4)
c
        iothr=iother(ielec)
c
        sumlf = -(csurf(ielec,i,npl)-csurf(ielec,i,n))*xint0(i,1,idec)
     1    -sum0(ielec,i)
c
        sum2f = (csurf(iothr,i,npl)-csurf(iothr,i,n))*xintl(i,1,idec)
     1    +suml(iothr,i)
c
        big(1) = coeff*diodr(i)*sumlf
        big(2) = coeff*diodr(i)*sum2f
c
        big(3)=0.d+00
        big(4)=0.d+00
c
        do 100 j=1,nrxn
          big(3) = ratson(i,j)*tf(ielec,j,jx) + big(3)
```

```
          big(4) = -ratson(i,j)*tb(ielec,j,jx) + big(4)
  100   continue
CHANNEL.FOR                                                    307

c
        f = big(1) + big(2) + big(3) + big(4)
c
        return
        end
c
c
        function dfdclc(ielim,ir,ic,ielecr,ielecc,jx,idec,csurf,tf,tb)
c
c       function dfdc for limiting current
c
        implicit double precision (a-h,o-z)
        common/evprob/xlamda(3),ai(3),z,diodr(10)
        common/co/coeff,xn
        common/cdata/cnctol,ftol,xint0(10,101,5),
     1    xint1(10,101,5),n2,nd,isec
        common/rxnpars/cref(10),cre(10),porq(2,10,5),ip,iq,uref(5),
     1    nspec,nrxn,one(2,5),rho0
        common/kinpars/xj(5),ratson(10,5),ason(2,5),ifor,iback
c
        dimension csurf(2,10,101),tf(2,5,101),tb(2,5,101)
c
        iothr = iother(ielecr)
c
        if(csurf(ielecc,ic,jx).eq.0.) print *,' ielecc,ic,jx',ielecr,
     1    ic,jx
        sumj = 0.d+00
        if(ielecr.ne.ielecc .or. ielecr.eq.ielim) go to 99
          do 110 j=1,nrxn
            sumj = sumj + ratson(ir,j)*(tf(ielecr,j,jx)*porq(ip,ic,j)/
     1        csurf(ielecr,ic,jx) - tb(ielecr,j,jx)*porq(iq,ic,j)/
     1        csurf(ielecr,ic,jx))
  110     continue
   99     dfdclc = delfcn(ir,ic)*coeff*diodr(ir)*(-xint0(ir,1,idec)*
     1      delfcn(ielecr,ielecc) + xint1(ir,1,idec)*
     1      delfcn(iothr,ielecc))
          dfdclc = dfdclc + sumj
  100   continue
c
        return
        end
c
c
        function irow(i,ielec)
        implicit double precision (a-h,o-z)
          common/rxnpars/cref(10),cre(10),porq(2,10,5),ip,iq,uref(5),
```

```
     1       nspec,nrxn,one(2,5),rho0
             common/elecs/v(2),ian,icath
CHANNEL.FOR                                                          308

             if(ielec.eq.ian) irow = i
             if(ielec.eq.icath) irow = i + nspec
           return
           end
c
c
      SUBROUTINE GAUSSQ(INTEGR,XI,JBEGIN,IDEC,SAVEA,SAVEC,TOX)
c
c     Gaussian quadrature integration
c     integr = 1 is for integrating current
c     integr = 2 is for integrating to find phi(-infinity)
c
      implicit double precision (a-h,o-z)
c
      dimension xi(2,101),x1(101),x2(101),y1(101),y2(101),tox(2,101)
c
      common/geom/pi,hol,nj,ndec,icut
      common/gauss/xp(96),gw(96),ngt,xielec(201),
     1  xiothr(201),njmore,xx(201)
      common/elecs/v(2),ian,icath
c
           h=dfloat(icut)**(idec-ndec)/dfloat(nj-1)
           js = jbegin
           if(jbegin.eq.2) js = 1
           savea = 0.d0
           savec = 0.d0
c
c          set up limits of integration
c
           nf = nj
           if(integr.eq.1) nf = js
           do 577 n = nj,nf,-1
           do 577 ielec=ian,icath
c          print *,'electrode',ielec
           xmin1 = dfloat(js-1)*h
           xmax1 = dfloat(n-1)*h
           xmin2 = xmax1
           jxm = n
           do 572 jx = js,n
c          print *,'diff',dabs(xi(ielec,jx))-dabs(xi(ielec,jx+1))
           if((idec.lt.ndec .and.
     1       dabs(xi(ielec,jx)) - dabs(xi(ielec,jx+1)) .gt. 0.)
     1       .or. (idec.eq.ndec .and .jx.lt.n .and.
     1       dabs(xi(ielec,jx)) - dabs(xi(ielec,jx+1)).gt.0.) ) then
             go to 572
           else
```

```
            xmax1 = dfloat(jx-1)*h
            xmin2 = xmax1

            jxm = jx
            go to 573
          end if
  572     continue
  573   xmax2 = dfloat(n-1)*h
c       print 100,xmin1,xmax1,xmin2,xmax2,jxm
c 100   format('xmin1',1pe11.4,'xmax1',1pe11.4,'xmin2',1pe11.4,
c    1    'xmax2',1pe11.4,'jxm',i2)
c
      nx1 = jxm - js + 1
      nx2 = n - jxm + 1
      do 574 jx = js,jxm
        x1(jx+1-js) = dfloat(jx-1)*h
  574 y1(jx+1-js) = xi(ielec,jx)
      do 575 jx = jxm,n
        x2(jx+1-jxm) = dfloat(jx-1)*h
  575 y2(jx+1-jxm) = xi(ielec,jx)
      save1 = 0.d0
      save2 = 0.d0
c       print *,'nx1',nx1,'nx2',nx2
c     print *,'xprm1    f1       f1x      xprm2      f2       f2x'
      do 576 jxp = 1,ngt
        xprm1 = (xmax1-xmin1)*xp(jxp) + xmin1
        xprm2 = (xmax2-xmin2)*xp(jxp) + xmin2
        call interp(1,x1,y1,xprm1,f1,nx1,3)
        call interp(0,x2,y2,xprm2,f2,nx2,3)
c       if(integr.eq.2) print 578,xprm1,f1,f1*dsqrt
c    1    (xprm1)*(xprm1-0.5),xprm2,
c    1    f2,f2*dsqrt(xprm2)*(xprm2-0.5)
c 578   format(6(1pe11.4,1x))
        if(integr.eq.1) then
          save1 = save1 + f1*dsqrt(xprm1)*gw(jxp)
          save2 = save2 + f2*dsqrt(xprm2)*gw(jxp)
        else
          save1 = save1 + f1*dsqrt(xprm1)*(xprm1-0.5)*gw(jxp)
          save2 = save2 + f2*dsqrt(xprm2)*(xprm2-0.5)*gw(jxp)
        end if
  576 continue
      save1 = save1*(dsqrt(xmax1) - dsqrt(xmin1))
      save2 = save2*(dsqrt(xmax2) - dsqrt(xmin2))
      if(ielec.eq.ian.and.n.eq.nj) savea = save1 + save2
      if(ielec.eq.icath.and.n.eq.nj) savec = save1 + save2
      if(integr.eq.1) tox(ielec,n) = save1 + save2
c     if(integr.eq.2) print *,'elec',ielec,'savea',savea,'savec',savec
      if(integr.eq.2) savea = savea + savec
c     if(integr.eq.2) print *,'save1',save1,'save2',save2
```

```
  577 continue
c
CHANNEL.FOR                                                    310

      return
      end
c
c
        SUBROUTINE INTERP(ileft,x,y,xj,f,nx,m)
c
c       Lagrangian interpolation subroutine adapted from Parrish
c       y(x)*sqrt(x) is assumed to be linear in sqrt(x)
c       v = y sqrt(x), u = sqrt(x)
c
c       y is the array of known values at the evenly-spaced x points
c       y(zj) = f is sought.  nx is the number of evenly-spaced x points
c       (x ranges between 0 and 1)
c
        implicit double precision (a-h,o-z)
c
        dimension y(101),x(101),u(101),v(101)
c
        if(ileft.eq.1) then
          zj = dsqrt(xj)
        else
          zj = dsqrt(1.-xj)
        end if
c
c       print *,'  u     v     x     y'
        do 100 ix=1,nx
          if(ileft.eq.1) then
            u(ix) = dsqrt(x(ix))
            v(ix) = y(ix) * u(ix)
c           print *,ileft,u(ix),v(ix),x(ix),y(ix)
          else
            u(1+nx-ix) = dsqrt(1.-x(ix))
            v(1+nx-ix) = y(ix) * u(1+nx-ix)
c           print *,ileft,u(ix),v(ix),x(ix),y(ix)
          end if
  100   continue
c       print 110,((u(ix),v(ix),x(ix),y(ix)),ix=1,nx)
c 110   format(4(1pe11.4,1x))
c
        kp = 1
        n = m + 1
        ne = 1
        if( (n/2)*2 .eq. n) ne = 0
c
        do 20 k = kp,nx
        if(u(k)-zj) 20,60,24
```

```
c
   20   continue
CHANNEL.FOR                                                    311

   22   jjs = nx - n + 1
        jjf = nx
        go to 30
c
   24   kp = k
        jjs = kp - n/2 -ne
        jjf = kp + n/2 -1
        if (jjf .ge. nx) go to 22
        if (jjs .gt. 0) go to 30
        jjs = 1
        jjf = n
   30   f = 0.d0
        if (jjs.lt.1) jjs = 1
        if(jjf.gt.nx) jjf = nx
c       print *,'jjs',jjs,'jjf',jjf
        do 50 k = jjs,jjf
          c = 1.d0
          do 40 kk = jjs,jjf
            if (kk .eq. k) go to 40
            c = c* (zj - u(kk)) / (u(k) - u(kk))
   40     continue
   50   f = f + c*v(k)
c       print *,'raw f',f,'zj',zj
        go to 80
c
   60   f = y(k)
        return
   80   f = f/zj
c
        return
        end
c
c
        subroutine totcurr(jbegin,h,f,save,tox)
c
        implicit double precision (a-h,o-z)
        dimension save(2),f(2,101),xintgd(101),tox(2,101)
c
        common/geom/pi,hol,nj,ndec,icut
        common/elecs/v(2),ian,icath
c
c       this subroutine integrates the function f (call is inside
c       idec loop
c
c         do 95 ielec=ian,icath
c           print 94,ielec,save(ielec)
```

```
c  94       format(' elec',i2,' save input=',1pe11.4)
c  95     continue
CHANNEL.FOR                                                    312

        kend=(nj-1)/icut + 1
        jb=jbegin
        npts=nj-1
        if(jbegin.eq.2) go to 367
        jb=jbegin-1
        npts=nj-kend+1
  367   continue
c
        do 463 ielec=ian,icath
          do 363 jx=jb,nj
            jj=jx-jb+1
c           print 93,ielec,jj,f(ielec,jx)
c  93       format(' elec',i2,' jj=',i2,' xintgd=',1pe11.4)
  363     xintgd(jj)=f(ielec,jx)
c         print *,' tox=',tox(ielec,jb)
          call simps(xintgd,npts,h,save(ielec),ielec,jb,tox)
c         print *,' save=',save(ielec)
  463   continue
c
        if(jbegin.ne.2) return
c
        do 100 ielec=ian,icath
          tox(ielec,1) = 0.d+00
c         print 465
c 465     format(' in totcurr initializing tox(1)')
          g1=f(ielec,1)
          g2=f(ielec,2)
          if(g1.eq.0.d+00 .or. g2.eq.0.d+00) go to 373
          if(g1.ne.g2 .and. g1*g2/dabs(g1*g2).gt.0.d+00) go to 372
  373     save(ielec)=save(ielec) + h/2.*(g1+g2)
          tox(ielec,2) = h/2.*(g1+g2)
          do 98 jx=3,nj
   98     tox(ielec,jx)=tox(ielec,jx) + h/2.*(g1+g2)
          go to 99
  372     continue
          save(ielec)=save(ielec) + 1.5*h*g2
          tox(ielec,2) = 1.5*h*g2
          do 97 jx=3,nj
   97     tox(ielec,jx)=tox(ielec,jx) + 1.5*h*g2
   99     continue
c         print 466
c 466     format(' in totcurr initializing tox(2)')
  100   continue
c
        return
        end
```

```
c
c
      subroutine simps(f,n,h,result,ielec,jb,tox)
c
c     This routine performs Simpson's rule integration of a
c     function defined by an array of equispaced values.
c     This routine is adapted from Curtis F. Gerald, Applied
c     Numerical Analysis, 2nd edition.
c
c     THIS ROUTINE HAS BEEN MODIFIED TO HANDLE SMALL N
c
c     Parameters are -
c       f     array of values of the function
c       n     number of points
c       h     the uniform spacing between x values
c       result   estimate of the integral that is returned to caller
c
      implicit double precision (a-h,o-z)
      dimension f(101),tox(2,101)
c
c     Check to see if number of panels is even.  Number of panels
c     is n-1
c
c
      npanel=n-1
      nhalf=npanel/2
      nbegin=1
      result=0.d+00
c
      save=tox(ielec,jb)
      tox(ielec,jb) = 0.d+00
c     print 302,jb
c 302 format(' in simps initializing tox(',i2,')')
c
      if((npanel-2*nhalf).eq.0) go to 5
c
c     Number of panels is odd.  Use 3/8 rule on first three panels,
c     1/3 rule on the rest of them.
c
c     Test to see if npanel is 1
c
      if(npanel.eq.1) go to 100
      result=3.*h/8.*(f(1) + 3.*f(2) + 3.*f(3) + f(4))
c
      tox(ielec,jb+3) = result
      tox(ielec,jb+2) = (2./3.) * result
      tox(ielec,jb+1) = (1./3.) * result
c     print 300,jb+1,jb+2,jb+3
```

```
c 300   format(' in simps initializing tox(',3i2,')')
        if(n.ne.4) go to 6
CHANNEL.FOR                                                    314

          tox(ielec,jb)=save
          return
c
    6   nbegin=4
c
c       apply 1/3 rule - add in first, second, last values.
c
    5   result=result + h/3.*(f(nbegin) + 4.*f(nbegin+1) + f(n))
        nbegin=nbegin+2
        if(nbegin.ne.n) go to 110
          tox(ielec,n+jb-1) = result
          tox(ielec,n+jb-2) = 0.5*(tox(ielec,n+jb-1)+tox(ielec,n+jb-3))
c         print 301,n+jb-1,n+jb-2
c 301     format(' in simps initializing tox(',2i2,')')
          tox(ielec,jb)=save
          return
c
c       The pattern after nbegin+2 is repetitive.  Get nend, the
c       place to stop
c
  110   nend=n-2
c
        tox(ielec,nbegin+jb-1) = result + h/3.*(f(nbegin)-f(n))
        tox(ielec,nbegin+jb-2) = 0.5*(tox(ielec,nbegin+jb-1) +
     1     tox(ielec,nbegin+jb-3))
c       print 301,nbegin+jb-2,nbegin+jb-1
c
        do 10 i=nbegin, nend, 2
          result=result + h/3.*(2.*f(i) + 4.*f(i+1))
          tox(ielec,i+jb+1) = result + h/3.*(f(i+2)-f(n))
          tox(ielec,i+jb) = 0.5*(tox(ielec,i+jb-1) + tox(ielec,i+jb+1))
c         print 301,i+jb,i+jb+1
   10   continue
c
        tox(ielec,jb)=save
        return
c
  100   result=(f(1)+f(2))/2.*h
        tox(ielec,jb+1) = result
c       print 302,jb+1
        tox(ielec,jb)=save
        return
        end
c
c
        SUBROUTINE NINTERP(x,y,xj,f,nx)
```

```
c
c       y(x)*sqrt(x) is assumed to be linear in sqrt(x)
CHANNEL.FOR                                                    315

c       v = y sqrt(x), u = sqrt(x)
c
c       y is the array of known values at the x points
c       y(xj) = f is sought.  nx is the number of x points
c       (x ranges between 0 and 1)
c
        implicit double precision (a-h,o-z)
c
        dimension y(201),x(201)
c
      ileft = 1
      if(xj.ge.0.5) ileft = 0
        do 20 k = 1,nx
      kp = k
        if(x(k)-xj) 20,60,30
   20   continue
c
   30 if(kp.ne.nx .or. y(kp)/y(kp-1).le.1.) go to 40
      b = 1./( (y(kp)/y(kp-1))**2 -1.)
      a = y(kp)*dsqrt(b)
      f=a/dsqrt(b+ (x(kp)-xj)/(x(kp)-x(kp-1)))
      return
   40 if(kp.ne.2 .or. y(1)/y(2).le.1.) go to 50
         b = 1./( (y(1)/y(2))**2 -1.)
         a = y(1)*dsqrt(b)
         f = a/dsqrt(b+xj/x(2))
         return
   50 if(ileft.eq.1)then
         ul=dsqrt(x(kp-1))
         u0=dsqrt(x(kp))
         zj = dsqrt(xj)
      else
         ul=dsqrt(1.-x(kp-1))
         u0=dsqrt(1.-x(kp))
         zj = dsqrt(1.-xj)
      endif
      f=(y(kp-1)*ul*(u0-zj)+y(kp)*u0*(zj-ul))/(u0-ul)/zj
        return
c
   60   f = y(k)
        return
        end
c
c
      SUBROUTINE EQFIVE(XOL,PHIAN,PHICATH,P2,P3,P5)
c
```

```
c
c     This subroutine calculates the dimensionless form of the
CHANNEL.FOR                                                          316

c     ((phi 0) - (phi *))/i<avg> given by equation 5 in Parrish
c     and Newman, JES, 117, 43-48, 1970.  The singularity in
c     the equation was eliminated by the method of Kantorovich
c     and Krylov
c
      implicit double precision (a-h,o-z)
c
      common/geom/pi,hol,nj,ndec,icut
      common/elecs/v(2),ian,icath
        common/gauss/xp(96),gw(96),ngt,xielec(201),
     1    xiothr(201),njmore,xx(201)
c
      ielec=ian
      iothr = icath
      c = pi/(2.*hol)
      an = 0.5d0
c
      if(xol.gt.0.d0 .and. xol.lt.1.d0) then
          xmin = xol/2.
          xmax = xmin + 0.5
          call ninterp(xx,xielec,xol,xie,njmore)
          call ninterp(xx,xiothr,xol,xio,njmore)
      else
          xmin = 0.5
          xmax = xmin
          xie = 0.
          xio = 0.
      end if
  130   sumra = 0.d0
        sumrb = 0.d0
        sumrc = 0.d0
cf      sumrf = 0.d0
        sumrs = 0.d0
        sumrt = 0.d0
c
        do 180 iregn=1,4
        go to (101,102,103,104),iregn
  101       xll = 0.d0
            xul = xmin**an
            go to 150
  102       if(xol.ge.1.d0 .or. xol.le.0.d0) go to 180
            if(xol.ge.0.75) then
              xul = (1.-xmin)**an
              xll = (1.-xol)**an
            else
              xll = xmin**an
```

```
              xul = xol**an
            end if
CHANNEL.FOR                                                    317

            go to 150
  103       if(xol.le.0.d0 .or. xol.ge. 1.d0) go to 180
            if(xol.le.0.25) then
              xll = xol**an
              xul = xmax**an
            else
              xul = (1.-xol)**an
              xll = (1.-xmax)**an
            end if
            go to 150
  104       xul = (1.-xmax)**an
            xll = 0.d0
  150     suma = 0.d0
          sumb = 0.d0
          sumc = 0.d0
cf      sumf = 0.d0
          sums = 0.d0
c         sumt = 0.d0
          do 170 jxp=1,ngt
            xprm = xp(jxp)*(xul-xll) + xll
            z = xprm**(1./an)
            if ((iregn.eq.3.and.xol.gt.0.25) .or. iregn.eq.4 .or.
     1        (iregn.eq.2.and.xol.gt.0.75)) z = 1.    z
            y = c*(xol-z)
            sinh2 = (dexp(y) - dexp(-y))**2/4.0
            if(dabs(y).lt.0.01) sinh2 = (y*(1. + y**2/6.*(1. +
     1          y**2/20.*(1. + y**2/42.))))**2
            cosh2 = 1. + sinh2
            if(y.ne.0) alnth2=dlog(sinh2/cosh2)
c
              ileft = 1
              if(z.gt.0.5) ileft = 0
              call ninterp(xx,xielec,z,aa,njmore)
              call ninterp(xx,xiothr,z,bb,njmore)
c
            bb = bb + aa
            Y2 = xprm**(1./an - 1.) / an
            clnch2 = dlog(cosh2)*Y2
c
            if(y.ne.0.) suma = suma + gw(jxp)*(aa*alnth2 - xie*dlog(y**2))*Y2
          if(y.ne.0.)sumb=sumb+gw(jxp)*((bb-aa)*alnth2-xio*dlog(y**2))*Y2
            sumc = sumc + gw(jxp)*bb*clnch2
cf        sumf = sumf + gw(jxp)*bb*(z-0.5)*Y2
            sums = sums + gw(jxp)*bb*Y2
            sumt = sumt + gw(jxp)*clnch2
  170     continue
```

```
c
          sumra = suma/2.*(xul-xll) + sumra
          sumrb = sumb/2.*(xul-xll) + sumrb
          sumrc = sumc/2.*(xul-xll) + sumrc
cf     sumrf = sumf/2.*(xul-xll) + sumrf
          sumrs = sums/2.*(xul-xll) + sumrs
          sumrt = sumt/2.*(xul-xll) + sumrt
  180   continue
c
          g2 = 0.d0
        if(xol.lt.1.d0 .and. xol.gt.0.d0) g2 = dlog(c**2) +
     1    2.*(1.-xol)*(dlog(1.-xol) - 1.) + 2.*xol*(dlog(xol) - 1.)
c
cf     p1 = sumrf/(2.*hol)
        p2 = -sumra/(2.*pi)
        p3 = -(g2*xie+sumra)/(2.*pi)
        p5 = -(sumrc-sumrs*sumrt)/(2.*pi)
        phicath = -(g2*xio + sumrb)/(2.0*pi)+p5
        phian = -(g2*xie + sumra)/(2.0*pi)+p5
c
      return
      end
```

Appendix G. Program NEWCHAN

G.1. Program Description

Program NEWCHAN solves for the current, concentration, and potential distribution in a channel flow cell by a method of collocation[22] applied to the partial current densities. Current functions (four special functions plus a set of Legendre polynomials) are input to the linear equations — Faraday's law, the superposition integrals for the flux, and Laplace's equation — to produce concentrations and potentials for substitution into the Butler-Volmer equation. The program solves for the set of coefficients that minimizes the error between the series representation of the current and the Butler-Volmer current at the specified collocation points.

A variation of the method is used to improve convergence at the limiting current; instead of minimizing the current errors on the limiting electrode, we require the surface concentrations to be zero.

Another feature of the program is the continuation method, which may be used to improve the convergence. The continuation variable may be either the cell voltage or the exchange current density of one of the reactions. The cell voltage is the recommended continuation variable.

The program may be run with or without a restart by setting IRSTRT = 1 or 0, respectively. In either case, a restart file is generated from every run. If the parameter IALL is set equal to 0, the current and/or concentration errors are calculated only at the collocation points, but one can set IALL = 1 to provide the detailed distribution of errors at the NJ evenly-spaced mesh points. This detailed information is needed to assess the accuracy of the results.

Upstream and downstream potentials may be calculated by adding extra mesh points. The points are 8 times more closely spaced in the intervals immediately before

and after the electrode edges.

Input Variables:

ICHAR = character string for identifying variables in the data file

NG = half the number of Gaussian quadrature points to be used in the integrals for the potential distribution (maximum allowable = 48)

ROOT(JXP) = roots for Gaussian quadrature (input the roots from 0 to 1)

GW(JXP) = Gaussian quadrature weights

XLAMDA(I) = eigenvalues from the asymmetric Graetz problem

AI(I) = coefficients from the asymmetric Graetz problem

NJ = number of evenly-spaced mesh points

NDEC = number of "decades" or regions with finely-spaced mesh points. Decades start from x=0 and are nested inside each other

ITSTRT = 0 if the run is not a restart, nonzero otherwise

HOL = aspect ratio, h/L

ICUT = specifies size of each "decade" of finely-spaced points. A decade starts from $x = 0$ and ends at $x = 1/\text{ICUT}$

XN = N = $(n_m^2/s_{RM}^2)\, F^2 D_R c_{R,ref}/(RT\kappa_{feed})\ *(6vL^2/(hD_R))^{1/3}$ = dimensionless limiting current density†

PEHOL = $Pe\ h/L = 2<v>h^2/(DL)$ = dimensionless parameter characterizing boundary layer thickness (directly proportional to Graetz number)

RHO0 = $\rho_o/c_{R,ref}$ = solvent density / reference concentration of main reactant

VCELL = $V_{cell}(-n_m F)/(s_{Rm}RT)$ = dimensionless cell voltage

VSTART = starting dimensionless cell voltage for continuation method (set VSTART = VCELL if the continuation method is not to be used)

† This interpretation holds only for thin boundary layers.

XJJ = starting dimensionless exchange current density for continuation method

JRXN : continuation may be applied to the exchange current of reaction JRXN

IVOLTS = nonzero if continuation is on cell voltage, zero otherwise

DELTAP = initial step size of continuation variable

ERRTST = relative convergence criterion for error in currents

MAXIT = maximum number of iterations for continuation method

IR = index for principal reactant

M = index for main reaction

ISEC = nonzero if running secondary current distribution

NSPEC = number of species

NRXN = number of reactions

NRE = index for reference-electrode reaction

UJTH(NRE) = dimensionless standard electrode potential of the reference-electrode reaction

ALPHA(1,J) = anodic transfer coefficient for reaction J

ALPHA(2,J) = cathodic transfer coefficient for reaction J

NSUBJ(J) = number of electrons transferred in reaction J

XJ(J) = $J_j = -n_m F L i_o/(s_{Rm} R T \kappa_\infty)$ = dimensionless exchange current density for reaction J

UJTH(J) = $U_j^\theta(-n_m F)/(s_{Rm} R T)$ = dimensionless standard electrode potential for reaction J

DIODR(I) = D_i/D_R = dimensionless diffusion coefficient of species I

CREF(I) = dimensionless reference concentration of species I

CRE(I) = dimensionless concentration of species I in the reference-electrode compartment

CIOCR(I) = $c_{i,feed}/c_{R,ref}$ = dimensionless feed concentration of species I

SIJ(I,J) = stoichiometric coefficient of species I in reaction J

NPTS = 1 + number of equations on an electrode

ICOLL(IELEC,IC) = mesh-point index for collocation points on electrode IELEC

IALL = 1 for computing the errors at all of the mesh points, 0 for calculating errors at the collocation points only

DELTA1 = parameter for limiting-current trial function (see equation D-15)

DELTA2 = parameter for trial function given in equation D-16

DELTA3 = parameter for trial function given in equation D-17

NPEXTR = total number of extra mesh points at which potentials are calculated

Output Variables:

XIAV = dimensionless average current density

PHISTR = integration constant in solution to Laplace's equation for the potential

UREF(J) = dimensionless theoretical open-circuit potential for reaction J, evaluated at the reference concentrations

PORQ(IP,I,J) = p_{ij} = anodic reaction order of species I in reaction J

PORQ(IQ,I,J) = p_{ij} = cathodic reaction order of species I in reaction J

CSURF(IAN,I,JX) = dimensionless anode surface concentration of species I at mesh point JX

XIJ(IAN,J,JX) = dimensionless anodic partial current density from reaction J at mesh point JX

XI(IAN,JX) = $i_{an}(-n_{m/}s_{Rm})FL/(RT\kappa_{\infty})$ = dimensionless anodic current density

POMPS(IAN,JX) = $(\Phi^{o}_{an}-\Phi^{*})(-n_{m}F)/(s_{Rm}RT)$ = potential calculated from Laplace's equation

NITS = last iteration count

G.2. Major Subroutines

G.2.1. Subroutine GAUSSQ

This subroutine calculates by Gaussian-quadrature integration the total currents on both electrodes and, if desired, the potentials at $+\infty$ and $-\infty$. The integrals are divided into regions to improve the accuracy.

G.2.2. Subroutine INTCXI

This subroutine calculates XINT0 (I, N-K+1, IDEC), which is

$$\frac{1}{\Delta x}\int_{(k-1)\Delta x}^{k\Delta x} C_{\xi}(n\Delta x - x')\,dx',$$

where C is the flux calculated from the asymmetric Graetz problem.

G.2.3. Subroutine ERR

Iterates to find the coefficients that minimize at the collocation points the errors between the Butler-Volmer currents and the approximate currents. The equation $<i>_{an} = <i>_{cath}$ is also included. If a "limiting electrode" (LIMELEC) is specified, the concentrations are set equal to zero on that electrode, the coefficient for the limiting-current trial function on the anode is set equal to zero, and the coefficient for the modified limiting-current function on the cathode is set equal to zero.

If the continuation method is being used, the subroutine is called twice for evaluating the derivatives with respect to the continuation variable. When the subroutine is called for the second time, there is no iteration on the coefficients.

Subroutines Required:

SUMIK

CFROMI

IKP (function subprogram)

EQFIVE

MATINV

CURRENT

SUMI

Function IKP collapses three subscripts into one (K, J, and IELEC)

G.2.4. Subroutine EQFIVE

Calculates the dimensionless form of the $(\Phi^o - \Phi^*)$ given by equation 5 in Parrish and Newman[21] (equation 1-6 in this work). The singularity in the equation was eliminated by the method of Kantorovich and Krylov[64] as discussed in section 1.4.3. The subroutine is written to accept trial current functions to produce an array of potentials.

G.2.5. Subroutine CURRENT

Calculates the current from the Butler-Volmer equation (unless the surface concentration is zero) and evaluates the derivatives of the partial current densities with respect to the current coefficients. The derivatives of the concentrations and potentials with respect to the coefficients must be provided (they are calculated in subroutines CFROMI and EQFIVE, respectively).

G.2.6. Subroutine CFROMI

Calculates concentration functions from the current functions, using Faraday's law (subroutine NFROMI) and the superposition integrals (subroutine CFROMN).

G.2.7. Subroutine CFROMN

Calculates concentrations from fluxes by solving the superposition-integral equation with the method of Acrivos and Chambré.[59] The Stieltjes contributions are calculated separately (subroutines FLAK and FLCK). To improve the accuracy for the current-

step problem, a correction factor is added to provide the correct concentration at the second finely-spaced mesh point.

```
c     PROGRAM NEWCHAN
c
c     This program solves for the current, concentation, and
c     potential distribution in a channel flow cell by a
c     method of collocation.  Current functions (three special
c     functions plus a set of Legendre polynomials) are input to
c     the linear equations -- Faraday's law, the superposition
c     integrals, and Laplace's equation -- to produce the inputs
c     for the Butler-Volmer equation, which gives the currents.  The
c     program solves for the set of coefficients that minimizes
c     the error between the series representation of the current and
c     the Butler-Volmer current at the specified collocation points.
c     For problems that do not converge, the continuation method has
c     been added.  The continuation may be on either the cell voltage
c     or the exchange current of a given reaction.
c
      implicit double precision (a-h,o-z)
c
      dimension csurf(2,10,101),xj(5),
     1  xij(2,5,101),save(2),cik(40,40),xijnew(2,5,101),
     1  xi(2,101),ujth(5),xinew(2,101),limsp(2),
     1  root(96),error(2,5,101),delc(2,10,101),bdum(40,40),
     1  xitotjk(41,2,5),array(2,101),csave(2,10,101),
     1  nsubj(5),ratio(2),tryr(2),tox(2,101),xitot(2),
     1  p0mps(2,101),xijka(41,5,404),xitotj(2,5),cord(41),
     1  p0mpsk(41,2,101),v(2),errsav(2,5,101),
     1  xijkc(41,5,404),cank(41,10,101),ccathk(41,10,101),
     1  co(41,2,5),cold(41,2,5),delco(41,2,5),b(40,40),
     1  d(40,3),dxijka(41,5,101),dxijkc(41,5,101),
     1  fluxak(41,10,404),pfine(41,2,229),ptemp(2,229)
c
      common/evprob/xlamda(3),ai(3),z,diodr(10),ciocr(10)
      common/co/coeff,xn
      common/geom/pi,hol,nj,ndec,icut
      common/rxnpars/cref(10),cre(10),porq(2,10,5),ip,iq,
     1  uref(5),nspec,nrxn,rho0
      common/kinpars/sij(10,5),ratson(10,5),ason(2,5),ifor,iback
      common/cdata/xint0(10,101,5),xint1(10,101,5),n2,nd,isec
      common/elecs/ian,icath
        common/gauss/xp(96),gw(96),ngt,xx(201),xielec(201),xiothr(20
     11),njmore
      common/colloc/npts,icoll(2,41),npextr,np(7)
      common/equalc/ikboth
      common/params/delta1,delta2,delta3
c
      data pi/3.14159265358979d+00/, ian/1/, icath/2/, ip/1/,
     1  iq/2/, ifor/1/, iback/2/
```

```
c
c
      call readin(xn,xj,ujth,nsubj,ng,root,nre,m,irstrt,vcell,
     1  vstart,jrxn,xjj,ivolts,deltap,errtst,maxit,iread)
      nptstot = 2*(npts-1)*nrxn + 1
      ncik = 2*(npts-1)
c
      print *,' points on anode',(icoll(ian,ik),ik=2,npts)
      print *,' points on cathode',(icoll(icath,ik),ik=2,npts)
c
      n2 = 2*nspec
      nd = 2*n2+1
      coeff = -xn*(16.*z)**(-1./3.)
      njm1 = nj - 1
c
c     Rearrange the Gaussian roots so they are in the interval
c     0 to 1 instead of -1 to 1.
c
      ngt = 2*ng
c
      do 10 jxp=1,ngt
   10 xp(jxp) = (root(jxp)+1.)/2.
c
      do 40 i=1,nspec
      do 40 j=1,nrxn
        if(sij(i,j).gt.0.d0) then
          porq(ip,i,j) = sij(i,j)/nsubj(j)
          porq(iq,i,j) = 0.d0
        end if
        if(sij(i,j).lt.0.d0) then
          porq(ip,i,j) = 0.d0
          porq(iq,i,j) = -sij(i,j)/nsubj(j)
        end if
        if(sij(i,j).eq.0.d0) then
          porq(ip,i,j) = 0.d0
          porq(iq,i,j) = 0.d0
        end if
        print 30,i,j,porq(ip,i,j),porq(iq,i,j)
   30   format(1x,' p(',i2,',',i2,')=',1pe11.4,' q=',1pe11.4)
   40 continue
c
      do 70 j=1,nrxn
        sum = 0.d0
        do 60 i=1,nspec
          if(ratson(i,j).eq.0.d0) go to 50
          if(cref(i) .eq. 0.d0) then
            print *,' error in cref(i),i=',i,'j=',j
            stop
```

```
          end if
          sum = sum + ratson(i,j)*dlog(cref(i)/rho0)
   50     if(ratson(i,nre) .eq. 0.d0) go to 60
          if(cref(i) .eq. 0.d0) then
            print *,' error in cref(i),i=',i,'j=',j
            stop
          end if
          sum = sum - ratson(i,nre)*dlog(cre(i)/rho0)
   60    continue
         print *,'j=',j,'ujth(j)',ujth(j),'ujth(nre)',ujth(nre)
         uref(j) = ujth(j) - ujth(nre) + sum
         print *,' j=',j,'ujref=',uref(j)
   70 continue
c
      call intcxi(nspec,xint0,xint1)
c
      istop=0
      if(istop.ne.0) stop
      v(icath) = 0.d0
      v(ian) = vcell
c
      phistr = vcell*ason(ian,1) / (ason(ian,1)+ason(icath,1))
     1  + v(icath)
      print *,' first phistr=',phistr
      print *,' x   current function'
c
      do 100 idec=1,ndec
        h = dfloat(icut)**(idec-ndec)/dfloat(nj-1)
      do 100 jx=1,nj
        jxfine = ifine(idec,jx)
        x = (jx-1)*h
      do 100 j=1,nrxn
      do 100 ik=2,npts
        xijka(ik,j,jxfine) = curfcn(ik,ian,x)
        xijkc(ik,j,jxfine) = -xijka(ik,j,jxfine)
        if(ik.eq.npts) print 98,x,(xijka(ikd,j,jxfine),ikd=2,npts)
   98   format(6(1pe11.4,1x))
  100 continue
c
      istop = 0
      if(istop.ne.0) stop
      do 140 ik = 2,npts
      do 140 j = 1,nrxn
c
          call gaussq(ik,xitot(ian),xitot(icath),dwnstr)
c
      do 140 ielec=ian,icath
        xitotjk(ik,ielec,j) = xitot(ielec)
```

```
      print 249,dwnstr,xitot(ian),xitot(icath)
  249 format(' downstream potential',e12.4,' xitota',e12.4,' xitotc',
     1  e12.4)
  140 continue
c
      itmax = 20
c
      limelec = 2
      do 175 ic=2,npts
  175 cord(ic) = 0.d0
      do 180 ik=1,npts
      do 180 j=1,nrxn
      do 180 ielec=ian,icath
        cold(ik,ielec,j) = 0.d0
  180 delco(ik,ielec,j) = 0.d0
      p0 = vstart
      pj0 = xjj
      phistr = p0*ason(ian,1)/(ason(ian,1)+ason(icath,1))
c     phistr = 456.d0
      print *,'phistr=',phistr,'p0',p0,'ason(ian,1)',ason(ian,1)
      cold(1,ian,1) = phistr
      co(1,ian,1) = phistr
      delco(2,ian,1) = 0.d0
      co(2,ian,1) =0.d0
      delco(2,icath,1) = 0.d0
      co(2,icath,1) = 0.d0
c
c     choose coefficients to give total current zero
c
c     sum = 0.d0
c     do 185 ik=2,npts
c     do 185 ielec=ian,icath
c     if(ik.eq.2.and.ielec.eq.ian) go to 185
c       sum = sum + co(ik,ielec,1)*xitotjk(ik,ielec,1)
c 185 continue
c     print *,'sum',sum
c     co(2,ian,1) = -sum/xitotjk(2,ian,1)
c     delco(2,ian,1) = co(2,ian,1)
      istop = 0
      if (istop.ne.0) stop
c
      if(irstrt.ne.0) then
        read (4,190)
  190   format(a72)
        print *,'coeffs read in'
        read (4,350),(cold(1,1,1),(((cold(ik,ielec,j),ielec=1,2)
     1    ,ik=2,npts),j=1,nrxn))
        print 350,(cold(1,1,1),(((cold(ik,ielec,j),ielec=1,2)
```

```
     1    ,ik=2,npts),j=1,nrxn))
          read (4,190)
          print *,'delco read in'
          read (4,350),(delco(1,1,1),(((delco(ik,ielec,j),
     1    ielec=1,2),ik=2,npts),j=1,nrxn))
          print 350,(delco(1,1,1),(((delco(ik,ielec,j),
     1    ielec=1,2),ik=2,npts),j=1,nrxn))
          print *,'p0mpsk read in '
          read (4,190)
          do 200 j=1,nrxn
          do 200 ik=2,npts
          do 200 ieleck=ian,icath
          do 200 jx=1,nj+28+npextr
            xol = xprt(jx,jreal)
            read(4,360),(pfine( ikp(ik,ieleck,j),ielec, jx),
     1      ielec=ian,icath)
            if(jreal.ge.1 .and. jreal.le.nj) then
              p0mpsk(ikp(ik,ieleck,j),ian,jreal) =
     1        pfine(ikp(ik,ieleck,j),ian,jx)
              p0mpsk(ikp(ik,ieleck,j),icath,jreal) =
     1        pfine(ikp(ik,ieleck,j),icath,jx)
            end if
            print 360,(pfine( ikp(ik,ieleck,j),ielec, jx),
     1      ielec=ian,icath)
  200     continue
          read (4,190)
          print *,'conc functions read in '
          do 210 j=1,nrxn
          do 210 ik=2,npts
          do 210 ieleck=ian,icath
          do 210 i=1,nspec
          do 210 jx=1,nj
            read(4,360),cank( ikp(ik,ieleck,j), i, jx),
     1      ccathk( ikp(ik,ieleck,j), i, jx)
            print 360,cank( ikp(ik,ieleck,j),i, jx),
     1      ccathk( ikp(ik,ieleck,j), i, jx)
  210     continue
        end if
c
        do 340 iter=1,maxit
          p0pd = 0.d0
          pjpd = 0.d0
          matcalc = 1
          print *,'try cell voltage',p0,'deltap',deltap
          print *,'try exch current',pj0,'deltap',deltap,'errtst',
     1    errtst
c
          call err(cord,iter,ivolts,p0,p0pd,xj,pj0,pjpd,jrxn,error,
```

```
     1    errsav,matcalc,b,nits,xijka,xijkc,xij,xi,csurf,
     1    phistr,xitotjk,v,p0mps,p0mpsk,pfine,ptemp,delco,cold,co,cank,
     1    ccathk,ncik,nptstot,irstrt,errtst,ierr,iread)
c
        print *,nits,'iterations'
        rewind 7
        write(7,220)
  220   format('coefficients')
        write (7,350),(cold(1,1,1),
     1    (((cold(ik,ielec,j),
     1    ielec=1,2),ik=2,npts),j=1,nrxn))
        write (7,230)
  230   format('delco')
        write (7,350),(delco(1,1,1),
     1    (((delco(ik,ielec,j),
     1    ielec=1,2),ik=2,npts),j=1,nrxn))
        write(7,240)
  240   format('potential derivatives')
        do 250 j=1,nrxn
        do 250 ik=2,npts
        do 250 ieleck=ian,icath
        do 250 jx=1,nj+28+npextr
          write(7,360),(pfine( ikp(ik,ieleck,j),
     1      ielec, jx), ielec=ian,icath)
  250   continue
        write (7,264)
  264   format('concentration derivatives')
        do 270 j=1,nrxn
        do 270 ik=2,npts
        do 270 ieleck=ian,icath
        do 270 i=1,nspec
        do 270 jx=1,nj
          write(7,360),cank( ikp(ik,ieleck,j),
     1      i, jx), ccathk( ikp(ik,ieleck,j), i,
     1      jx)
  270   continue
        write (7,271),nits
  271   format(i3)
c
        if(ierr.ne.0) then
          print *,'no convergence in err'
          go to 370
        end if
        istop=0
        if(istop.ne.0) stop
        if(ivolts.eq.1) then
          if(p0.ge.vcell) then
            print *,'continuation method converged, p0=',p0,
```

```
     1        'vcell',vcell
            go to 370
          end if
        else
          if(pj0.ge.xj(jrxn)) then
            print *,'continuation method converged, pj0=',pj0,
     1        'xj',xj(jrxn)
            go to 370
          end if
        end if
c
        matcalc=0
        if(ivolts.eq.1) then
          p0pd = p0 + deltap*1.d-06
          print *,'cell voltage',p0+deltap*1.d-06
        else
          pjpd = pj0 + deltap*1.d-06
          print *,'exch current',pj0+deltap*1.d-06,'errtst',
     1      errtst
        end if
c
        call err(cord,iter,ivolts,p0,p0pd,xj,pj0,pjpd,jrxn,error,
     1    errsav,matcalc,bdum,nits,xijka,xijkc,xij,xi,csurf,
     1    phistr,xitotjk,v,p0mps,p0mpsk,pfine,ptemp,delco,cold,co,cank,
     1    ccathk,ncik,nptstot,irstrt,errtst,ierr,iread)
c
        do 280 j=1,nrxn
        do 280 ik=1,npts
        do 280 ielec=ian,icath
  280   cold(ik,ielec,j) = co(ik,ielec,j)
        do 300 ielec=ian,icath
        do 300 j=1,nrxn
        do 300 ic=2,npts
          d( ikp(ic,ielec,j),1 ) = -(error(ielec,j,icoll(ielec,ic))-
     1      errsav(ielec,j,icoll(ielec,ic)))/1.d-06
  300   continue
        d(1,1) = 0.d0
c       print *,'b matrix for continuation'
c       print 320,((b(i,j),j=2,nptstot),i=2,nptstot)
c       print *,'row 1'
c       print 310,(b(1,j),j=1,nptstot)
c       print *,'column 1'
c       print 310,(b(i,1),i=1,nptstot)
  310   format(/,9(1x,1pe10.3))
  320   format(8(1x,1pe10.3))
c       print *,'d vector for continuation'
c       print 320,(d(i,1),i=1,nptstot)
        call matinv(nptstot,nptstot,1,b,d,determ)
```

```
c       print *,'d vector after continuation'
c       print 320,(d(i,1),i=1,nptstot)
        if(determ.eq.0.d0) then
          print *,' zero det in matinv for new p'
          stop
        end if
        do 330 j=1,nrxn
        do 330 ik=1,npts
        do 330 ielec=ian,icath
  330   delco(ik,ielec,j) = d( ikp(ik,ielec,j), 1 )
        if(nits.lt.3) deltap = deltap*3.
        if(nits.gt.3) deltap = deltap*0.56
        if(ivolts.eq.1) then
          p0 = p0 + deltap
          if(p0.gt.vcell) p0 = vcell
        else
          pj0 = pj0 + deltap
          if(pj0.gt.xj(jrxn)) pj0 = xj(jrxn)
        end if
  340 continue
      print *,'continuation method not converged'
c
  350 format(1x,1pe15.8)
  360 format(2(1x,1pe15.8))
  370 call prtout(co,
     1  xij,xi,csurf,xiav,phistr,xitotjk,v,p0mps,ptemp)
c
      stop
      end
c
c
      FUNCTION CURFCN(IFCN,IELEC,X)
c
c
      implicit double precision (a-h,o-z)
      dimension xijka(41,5,404),xijkc(41,5,404),fluxck(41,10,404)
c
      common/evprob/xlamda(3),ai(3),z,diodr(10),ciocr(10)
      common/co/coeff,xn
      common/geom/pi,hol,nj,ndec,icut
      common/elecs/ian,icath
      common/gauss/xp(96),gw(96),ngt,xx(201),xielec(201),xiothr(201),
     1 njmore
      common/colloc/npts,icoll(2,41),npextr,np(7)
      common/params/delta1,delta2,delta3
c
c
c
```

```
      if(ifcn.eq.2) then
        zeta = x*diodr(1)/z
        curfcn = coeff*diodr(1)*(flak(zeta) - flck(zeta))
c       curfcn = -coeff*diodr(1)*flck(zeta)
      end if
c
      if(ifcn.eq.3) then
       curfcn = (x+delta1)**-1./3.
      end if
c
      if (ifcn.eq.4) then
          curfcn = 1.d0/dsqrt(x + delta2/(1./hol + 1.))
      end if
c
      if (ifcn.eq.5) then
          curfcn = 1.d0/dsqrt(1. - x + delta3/(1./hol + 1.))
      end if
c
      if(ifcn.ge.6) then
        do 90 ik=6,npts
        if(ifcn.eq.ik) then
          curfcn = p(ik-6,x)
        end if
   90   continue
      end if
c
c
      if(ielec.eq.icath) curfcn = -curfcn
      return
      end
c
c
      FUNCTION IOTHER(I)
c
c     this function is used for generating the index of the
c     'other' electrode
c
      iother = 3 - i
      return
      end
c
c
      SUBROUTINE PRTOUT(CO,XIJ,XI,CSURF,XIAV,PHISTR,
     1  XITOTJK,V,POMPS,PTEMP)
c
c     This subroutine prints the final results
c
      implicit double precision (a-h,o-z)
```

```
      dimension csurf(2,10,101),xij(2,5,101),
     1  ptemp(2,101),
     1  xi(2,101),xitotjk(41,2,5),
     1  v(2),p0mps(2,101),co(41,2,5)
c
      common/rxnpars/cref(10),cre(10),porq(2,10,5),ip,iq,
     1  uref(5),nspec,nrxn,rho0
      common/evprob/xlamda(3),ai(3),z,diodr(10),ciocr(10)
      common/co/coeff,xn
      common/geom/pi,hol,nj,ndec,icut
      common/elecs/ian,icath
      common/gauss/xp(96),gw(96),ngt,xx(201),xielec(201),xiothr(20
     11),njmore
      common/colloc/npts,icoll(2,41),npextr,np(7)
      common/params/delta1,delta2,delta3
c
      print 100,coeff,xiav,phistr
  100 format(/,' coeff=',1pe11.4,' iavg=',1pe11.4,
     1  ' phistr=',1pe11.4)
c
      do 140 j=1,nrxn
        print 130,j,uref(j)
  130   format(' j=',i2,' ujref=',1pe11.4)
  140 continue
c
      do 170 i=1,nspec
        print 150,i,(j,porq(1,i,j),j=1,nrxn)
  150   format(' i=',i2,5(' j=',i2,' pij=',1pe11.4))
        print 160,i,(j,porq(2,i,j),j=1,nrxn)
  160   format(' i=',i2,5(' j=',i2,' qij=',1pe11.4))
  170 continue
c
        do 200 i=1,nspec
        print 180,i
        write(8,180),i
  180   format(/,' species',i3,/,'    x      ccath     can')
      do 200 jx=1,nj
        x=dfloat(jx-1)/dfloat(nj-1)
        print 190,x,csurf(2,i,jx),csurf(1,i,jx)
        write (8,190),x,csurf(2,i,jx),csurf(1,i,jx)
  190   format(1x,1pe9.2,1x,1pe11.4,2x,1pe11.4)
  200 continue
c
      do 230 j=1,nrxn
        print 210,j
  210   format(/,' reaction',i2,/,
     1    '     x          xicj          xiaj      ')
      do 230 jx=1,nj
```

```
        x=dfloat(jx-1)/dfloat(nj-1)
        print 220,x,xij(2,j,jx),xij(1,j,jx)
  220   format(1x,1pe9.2,1x,1pe11.4,2x,1pe11.4)
  230 continue
c
      print 390
  390 format(/,
     1  '   x         ian        icath        p0mpsa        p0mpsc')
c
      print *,'potentials'
      do 420 jx=1,nj+28+npextr
        print 400,xprt(jx,jreal),ptemp(1,jx),ptemp(2,jx)
  400   format(1x,0pf8.4,1x,1pe11.4,2x,1pe11.4,2x)
  420 continue
c
      print *,'partial currents'
      do 430 j=1,nrxn
        print *,'reaction',j
      do 430 jx=1,nj+14
        if(jx.le.9) xol = dfloat(jx-1)/dfloat(nj-1)/8.
        if(jx.gt.9.and.jx.le.nj+6) xol = dfloat(jx-8)/dfloat(nj-1)
        if(jx.gt.nj+6) xol = dfloat(nj-2)/dfloat(nj-1)
     1    + dfloat(jx-(nj+6))/dfloat(nj-1)/8.
        ani = 0.d0
        cathi = 0.d0
        do 425 ik=2,npts
          ani = ani + co(ik,ian,j)*curfcn(ik,ian,xol)
          cathi = cathi + co(ik,icath,j)*curfcn(ik,icath,xol)
  425   continue

        print 400,xol,ani,cathi
  430 continue
c
      return
      end
c
c
      SUBROUTINE CURRENT(XIJOLD,XIJKA,XIJKC,IALL,YNEW,PHISTR,
     1  XITOTC,XITOTA,P0MPS,CSURF,CANK,CCATHK,XIJ,XI,
     1  XITOTJK,P0MPSK,DXIJKA,DXIJKC,V,XJ)
c
c     Calculates the current from the Butler-Volmer equation
c
      implicit double precision (a-h,o-z)
c
      dimension csurf(2,10,101),cank(41,10,101),
     1  ccathk(41,10,101),xij(2,5,101),xi(2,101),p0mps(2,101),
     1  eta(2,101),xitotjk(41,2,5),xitot(2),save(2),savej(2,5),
```

```
     1  p0mpsk(41,2,101),dprodf(41),dprodb(41),xj(5),
     1  dxik(41,2,101),dxijka(41,5,101),dxijkc(41,5,101),
     1  dfor(41),dback(41),def(41),deb(41),v(2),
     1  xijold(2,5,101),xijka(41,5,404),xijkc(41,5,404)
c
      common/geom/pi,hol,nj,ndec,icut
      common/kinpars/sij(10,5),ratson(10,5),ason(2,5),ifor,iback
      common/rxnpars/cref(10),cre(10),porq(2,10,5),ip,iq,
     1  uref(5),nspec,nrxn,rho0
      common/elecs/ian,icath
      common/colloc/npts,icoll(2,41),npextr,np(7)
c
      njm1 = nj - 1
      kend = njm1/icut + 1
      expmax = 2.303d+00*270.d0
c
      if(iall.eq.0) then
        jx1 = 2
        jx2 = npts
      else
        jx1 = 1
        jx2 = nj
      end if
c
      do 120 ic=jx1,jx2
      do 120 ielec=ian,icath
        if(iall.eq.0) then
          jx = icoll(ielec,ic)
        else
          jx = ic
        end if
        eta(ielec,jx) = v(ielec) - (p0mps(ielec,jx) + phistr)
        print 121,ielec,jx,eta(ielec,jx),phistr,p0mps(ielec,jx)
  121   format('elec',i2,'jx',i2,'eta',1pe11.4,'phistr',1pe11.4,
     1  'p0mps',1pe11.4)
c
c
        do 110 j=1,nrxn
          if(dabs(ason(ifor,j)*eta(ielec,jx)) .ge. expmax .or.
     1      dabs(ason(iback,j)*eta(ielec,jx)) .ge. expmax) then
c
            print 100
  100       format(' stopped in current because exp blows up')
            print *,'elec',ielec
            print *,'jx',jx,'eta',eta(ielec,jx),'v',v(ielec),
     1        'p0mps',p0mps(ielec,jx)
            stop
          end if
```

```
  110   continue
c
  120 continue
c
c
      jbegin = 2
      h = 1./dfloat(njml)
c
      do 280 ielec=ian,icath
        ilimc = 0
        if(iall.eq.0) then
          jx1 = 2
          jx2 = npts
        else
          jx1 = 1
          jx2 = nj
        end if
      do 280 ic=jx1,jx2
        if(iall.eq.0) then
          jx = icol1(ielec,ic)
        else
          jx = ic
        end if
      do 280 j=1,nrxn
        prodf = 1.d0
        prod2f = 1.d0
        prodb = 1.d0
        prod2b = 1.d0
        do 160 k=1,nspec
          p = porq(ip,k,j)
          q = porq(iq,k,j)
          if(csurf(ielec,k,jx).eq.0.d0 .and. p.eq.0.d0)
     1      go to 140
          if(csurf(ielec,k,jx).gt.1.d-10) go to 130
          if(prodf.ne.0.d0) prdfoc=prodf
          prodf = 0.d0
          ilimc = 1
          go to 140
  130     continue
          prodf = prodf*(csurf(ielec,k,jx)/cref(k))**p
          prod2f = prod2f*(csurf(ielec,k,jx)/cref(k))**
     1      (p+ason(ifor,j)*ratson(k,j))
  140     if(csurf(ielec,k,jx).eq.0.d0 .and. q.eq.0.d0)
     1      go to 160
          if(csurf(ielec,k,jx).gt.1.d-10) go to 150
          if(prodb.ne.0.d0) prdboc = prodb
          ilimc = 1
          prodb = 0.d0
```

```
          go to 160
  150     prodb = prodb*(csurf(ielec,k,jx)/cref(k))**q
          prod2b = prod2b*(csurf(ielec,k,jx)/cref(k))**
     1      (p+ason(ifor,j)*ratson(k,j))
  160   continue
c
        do 190 jk=1,nrxn
        do 190 ieleck=ian,icath
        do 190 ik=2,npts
          dprodf(ikp(ik,ieleck,jk)) = 0.d0
          dprodb(ikp(ik,ieleck,jk)) = 0.d0
        do 190 i=1,nspec
          p = porq(ip,i,j)
          q = porq(iq,i,j)
          ci = csurf(ielec,i,jx)
          if(ielec.eq.ian) dcidcok = cank(ikp(ik,ieleck,jk),i,jx)
          if(ielec.eq.icath) dcidcok = ccathk(ikp(ik,ieleck,jk),i,jx)
          if(p.eq.0.d0) go to 170
c
          if(ci.eq.0.d0) then
            print *,'ci=0, ielec,i,jx',ielec,i,jx
            if(p.gt.1.d0) go to 170
            if(p.lt.1.d0) then
              print *,'c=0 and p<1 on elec',ielec,'spec',i
              stop
            end if
            if(p.eq.1.d0) then
              dprodf(ikp(ik,ieleck,jk)) = prdfoc*dcidcok
     1          + dprodf(ikp(ik,ieleck,jk))
              go to 170
            end if
          end if
c
          dprodf(ikp(ik,ieleck,jk)) = prodf*p/ci * dcidcok
     1     + dprodf(ikp(ik,ieleck,jk))
c
  170     continue
          if(q.eq.0.d0) go to 180
c
          if(ci.eq.0.d0) then
          if(q.gt.1.d0) go to 180
            if(q.lt.1.d0) then
              print *,'c=0 and q<1 on elec',ielec,'spec',i
              stop
            end if
            if(q.eq.1.d0) then
              dprodb(ikp(ik,ieleck,jk)) = prdfob*dcidcok
     1          + dprodb(ikp(ik,ieleck,jk))
```

```
              go to 180
            end if
          end if
c
          dprodb(ikp(ik,ieleck,jk)) = prodb*q/ci * dcidcok
     1      + dprodb(ikp(ik,ieleck,jk))
c
  180     continue
  190   continue
c
        etasjf = ason(ifor,j) * (eta(ielec,jx) - uref(j))
c       print *,'j,jx',j,jx,'etasjf',etasjf
        etermf=dexp(ason(ifor,j) * (eta(ielec,jx) - uref(j)))
c
        do 200 ieleck=ian,icath
        do 200 ik=2,npts
        do 200 jk=1,nrxn
  200   def(ikp(ik,ieleck,jk)) = -ason(ifor,j)
     1    *p0mpsk(ikp(ik,ieleck,jk),ielec,jx)*etermf
c
        def(1) = -ason(ifor,j)*etermf
        dprodf(1) = 0.d0
        if(prod2f.eq.0.) then
          print *,'prod2f=0'
          go to 215
        end if
        if(dabs(etasjf/(xj(j)*prod2f)).le.0.01d0) then
c       print *,'linear approx for forw term, elec',ielec,'jx',jx
          etermf = (1. +
     1      ason(ifor,j)*(eta(ielec,jx)-uref(j)))
          do 210 ieleck=ian,icath
          do 210 jk=1,nrxn
          do 210 ik=2,npts
  210     def(ikp(ik,ieleck,jk)) = -ason(ifor,j)
     1      *p0mpsk(ikp(ik,ieleck,jk),ielec,jx)
          def(1) = -ason(ifor,j)
        end if
  215   for = prodf*xj(j)*etermf
c       print *,'elec',ielec,'jx',jx,'for',for
c
        do 220 jk=1,nrxn
        do 220 ieleck=ian,icath
        do 220 ik=1,npts
          dfor(ikp(ik,ieleck,jk)) = xj(j) *
     1      (prodf*def(ikp(ik,ieleck,jk))
     1      + dprodf(ikp(ik,ieleck,jk))*etermf)
  220   continue
c
```

```
        etasjb = -ason(iback,j) * (eta(ielec,jx) - uref(j))
c       print *,'j,jx',j,jx,'etasjb',etasjb
        etermb = dexp(-ason(iback,j) * (eta(ielec,jx) - uref(j)))
c
        do 230 ieleck=ian,icath
        do 230 jk=1,nrxn
        do 230 ik=2,npts
  230   deb(ikp(ik,ieleck,jk)) = ason(iback,j)
     1    *p0mpsk(ikp(ik,ieleck,jk),ielec,jx)*etermb
c
        deb(1) = ason(iback,j)*etermb
        dprodb(1) = 0.d0
        if(prod2b.eq.0.) then
c         print *,'prod2b=0'
          go to 245
        end if
        if(dabs(etasjb/(xj(j)*prod2b)).le.0.01d0) then
c         print *,'linear approx for back term, elec',ielec,'jx',jx
          etermb = (1. -
     1      ason(iback,j)*(eta(ielec,jx)-uref(j)))
          do 240 ieleck=ian,icath
          do 240 jk=1,nrxn
          do 240 ik=2,npts
  240     deb(ikp(ik,ieleck,jk)) = ason(iback,j)
     1      *p0mpsk(ikp(ik,ieleck,jk),ielec,jx)
          deb(1) = ason(iback,j)
        end if
  245   back = prodb*xj(j)*etermb
c       print *,'elec',ielec,'jx',jx,'back',back
c
        do 250 ieleck=ian,icath
        do 250 jk=1,nrxn
        do 250 ik=1,npts
          dback(ikp(ik,ieleck,jk)) = xj(j) *
     1      (prodb*deb(ikp(ik,ieleck,jk))
     1      + dprodb(ikp(ik,ieleck,jk))*etermb)
  250   continue
c
        xij(ielec,j,jx) = for - back
        if(ilimc .ne. 0 .and. ielec.eq.icath) xij(ielec,j,jx) =
     1    xijold(ielec,j,jx)
c       if(ilimc .ne. 0 .and. ielec.eq.ian) print *,'lim curr on an'
        if(ielec.eq.ian) then
          do 260 ieleck=ian,icath
          do 260 jk=1,nrxn
          do 260 ik=1,npts
            dxijka(ikp(ik,ieleck,jk),j,jx) = dfor(ikp(ik,
     1      ieleck,jk)) - dback(ikp(ik,ieleck,jk))
```

```
c           if(ilimc.ne.0) dxijka(ikp(ik,ieleck,jk),j,jx) = xijka(
c     1       ik,j,ifine(ndec,jx))
  260     continue
        end if
        if(ielec.eq.icath) then
          do 270 ieleck=ian,icath
          do 270 jk=1,nrxn
          do 270 ik=1,npts
            dxijkc(ikp(ik,ieleck,jk),j,jx) = dfor(ikp(ik,
     1       ieleck,jk)) - dback(ikp(ik,ieleck,jk))
c           if(ilimc.ne.0) dxijkc(ikp(ik,ieleck,jk),j,jx) = xijkc(
c     1       ik,j,ifine(ndec,jx))
  270     continue
        end if
  280 continue
c
      call sumi(xi,xij)
      do 290 ielec=ian,icath
      do 290 j=1,nrxn
      do 290 ik=1,npts
  290 call sumik(dxik,j,ikp(ik,ielec,j),ikp(ik,ielec,j),
     1  dxijka,dxijkc,41,5,101,1)
c
c
      return
      end
c
c
      SUBROUTINE CFROMI(XIJKA,XIJKC,IK,J,IELECK,IKA,CANK,
     1  CCATHK)
c
c     Calculates concentration functions from the current functions
c
      implicit double precision (a-h,o-z)
c
      dimension cank(41,10,101),ccathk(41,10,101),
     1  xijka(41,5,404),xijkc(41,5,404),fluxak(41,10,404),
     1  fluxck(41,10,404)
c
      common/evprob/xlamda(3),ai(3),z,diodr(10),ciocr(10)
      common/co/coeff,xn
      common/geom/pi,hol,nj,ndec,icut
      common/rxnpars/cref(10),cre(10),porq(2,10,5),ip,iq,
     1  uref(5),nspec,nrxn,rho0
      common/kinpars/sij(10,5),ratson(10,5),ason(2,5),ifor,iback
      common/cdata/xint0(10,101,5),xint1(10,101,5),n2,nd,isec
      common/elecs/ian,icath
        common/gauss/xp(96),gw(96),ngt,xx(201),xielec(201),xiothr(20
```

```
     11),njmore
      common/colloc/npts,icoll(2,41),npextr,np(7)
c
      call nfromi(xijka,xijkc,ik,j,ika,fluxak,fluxck)
      call cfromn(fluxak,fluxck,cank,ccathk,ika,ieleck)
c
      return
      end
c
c
      SUBROUTINE NFROMI(XIJKA,XIJKC,IK,J,IKA,FLUXAK,FLUXCK)
c
c     Calculates flux from current using Faraday's law
c
      implicit double precision (a-h,o-z)
c
      dimension xijka(41,5,404),xijkc(41,5,404),
     1  fluxak(41,10,404),fluxck(41,10,404)
c
      common/geom/pi,hol,nj,ndec,icut
      common/rxnpars/cref(10),cre(10),porq(2,10,5),ip,iq,
     1  uref(5),nspec,nrxn,rho0
      common/kinpars/sij(10,5),ratson(10,5),ason(2,5),ifor,iback
      common/elecs/ian,icath
c
      do 100 idec=1,ndec
      do 100 i=1,nspec
      do 100 jx=1,nj
        jxfine = ifine(idec,jx)
        fluxak(ika,i,jxfine) = -ratson(i,j)*xijka(ik,j,jxfine)
  100 fluxck(ika,i,jxfine) = - ratson(i,j)*xijkc(ik,j,jxfine)
c
c
      return
      end
c
c
      SUBROUTINE CFROMN(FLUXAK,FLUXCK,CANK,CCATHK,IK,IELECK)
c
c     Calculates concentrations from fluxes by solving the
c     superposition-integral equation
c
      implicit double precision (a-h,o-z)
      dimension fluxak(41,10,404),fluxck(41,10,404),
     1  cank(41,10,101),ccathk(41,10,101),sum0(2,10),
     1  suml(2,10),b(40,40),d(40,1)
c
      common/cdata/xint0(10,101,5),xintl(10,101,5),n2,nd,isec
```

```
      common/elecs/ian,icath
      common/rxnpars/cref(10),cre(10),porq(2,10,5),ip,iq,
     1  uref(5),nspec,nrxn,rho0
      common/kinpars/sij(10,5),ratson(10,5),ason(2,5),ifor,iback
      common/geom/pi,hol,nj,ndec,icut
      common/co/coeff,xn
      common/evprob/xlamda(3),ai(3),z,diodr(10),ciocr(10)
c
      cdamp=1.0d-06
      n2 = 2*nspec
      jbegin = 1
      kend = (nj-1)/icut + 1
c
c     print *,'in cfromn, ik=',ik
      do 100 jx=1,nj
      do 100 i=1,nspec
        cank(ik,i,jx) = 0.d0
  100 ccathk(ik,i,jx) = 0.d0
      if(isec.ne.0) return
c
      do 190 idec=1,ndec
        h = dfloat(icut)**(idec-ndec)/dfloat(nj-1)
        do 170 jx=jbegin,nj
          jxfine = ifine(idec,jx)
c
          do 130 i=1,nspec
c
            zeta = (jx-1)*h*diodr(i)/z
            do 110 ielec=ian,icath
              sum0(ielec,i) = 0.d0
  110       suml(ielec,i) = 0.d0
c
            if(jx.le.2) go to 130
            n = jx - 1
            nml = n - 1
c
            do 120 k=1,nml
              kpl = k + 1
              nmkpl = n - k + 1
              sum0(ian,i) = sum0(ian,i) + (cank(ik,i,kpl)
     1          -cank(ik,i,k))*xint0(i,nmkpl,idec)
              sum0(icath,i) = sum0(icath,i) + (ccathk(ik,i,kpl)
     1          -ccathk(ik,i,k))*xint0(i,nmkpl,idec)
              suml(ian,i) = suml(ian,i) + (cank(ik,i,kpl)
     1          -cank(ik,i,k))*xintl(i,nmkpl,idec)
  120       suml(icath,i) = suml(icath,i) + (ccathk(ik,i,kpl)
     1        -ccathk(ik,i,k))*xintl(i,nmkpl,idec)
c
```

```
  130     continue
c
          irow = 0
          do 140 ielec=ian,icath
            iothr=iother(ielec)
          do 140 i=1,nspec
            jcol = 0
            irow = irow + 1
          do 140 jelec=ian,icath
          do 140 jc=1,nspec
c
c           Correction factor to get accurate results for the
c           constant-current problem
c
            corr = 1.0d0
            if(jx.eq.2.and.idec.eq.1) corr = 4.*pi/(9.*dsqrt(3.d0))
            jcol = jcol + 1
            b(irow,jcol) = 0.d0
            if(irow.eq.jcol) then
              if(jx.ne.1) b(irow,jcol) = -xint0(i,1,idec)
     1          *coeff*diodr(i)*corr
              if(jx.eq.1) b(irow,jcol) = flck(zeta)*
     1          coeff*diodr(i)
            else
              if(i.eq.jc.and.jx.ne.1) b(irow,jcol) =
     1          xint1(i,1,idec)*coeff*diodr(i)
              if(i.eq.jc.and.jx.eq.1) b(irow,jcol) =
     1          -flak(zeta)*coeff*diodr(i)
            end if
            if(ieleck.eq.ian) then
              flxak = fluxak(ik,i,jxfine)
              flxck = 0.d0
            else
              flxak = 0.d0
              flxck = fluxck(ik,i,jxfine)
            end if
            if(ielec.eq.ian) then
              if(jx.ne.1) then
                d(irow,1) = (+sum0(ielec,i) - cank(ik,i,jx-1)*
     1          xint0(i,1,idec)*corr - sum1(iothr,i) +
     1          ccathk(ik,i,jx-1)*xint1(i,1,idec))*coeff
     1          *diodr(i) + flxak + (-cank(ik,i,1)*flck(zeta)
     1          + ccathk(ik,i,1)*flak(zeta))*coeff*diodr(i)
              else
                d(irow,1) = flxak
              end if
            else
              if(jx.ne.1) then
```

```
                d(irow,1) = (+sum0(ielec,i) - ccathk(ik,i,jx-1)*
     1          xint0(i,1,idec)*corr - sum1(iothr,i) +
     1          cank(ik,i,jx-1)*xint1(i,1,idec))*coeff
     1          *diodr(i) + flxck + (-ccathk(ik,i,1)*flck(zeta)
     1          + cank(ik,i,1)*flak(zeta))*coeff*diodr(i)
     1
              else
                d(irow,1) = flxck
              end if
            end if
  140     continue
c
  145     format(4(1x,1pe11.4))
          call matinv(n2,n2,1,b,d,determ)
c
          if(determ.eq.0.d0) print *,' zero det in matinv'
          irow = 0
          do 160 ielec=ian,icath
          do 160 i=1,nspec
            irow = irow + 1
            if(d(irow,1).lt.-ciocr(i).and.d(irow,1).gt.-(ciocr(i)
     1        +1.d-06)) d(irow,1) = -ciocr(i)
            if(ielec.eq.ian) then
              cank(ik,i,jx) = d(irow,1)
            else
              ccathk(ik,i,jx) = d(irow,1)
            end if
  160     continue
  170   continue
        if(idec.eq.ndec) go to 190
        do 180 i=1,nspec
        do 180 k=1,kend
          jj = k*icut - icut + 1
          cank(ik,i,k) = cank(ik,i,jj)
  180   ccathk(ik,i,k) = ccathk(ik,i,jj)
        jbegin = kend + 1
  190 continue
c
  200 continue
c
      return
      end
c
c
      SUBROUTINE MATINV(L,N,M,b,d,DETERM)
c
c       This subroutine solves a matrix equation of the form
c       BX = D.  The answer X is put into the first column of D.
```

```
c
c       Input variables
c               n = dimension of square b matrix
c               m = number of columns in d matrix
c               b = matrix in equation bx=d
c               d : first column contains vector d in bx=d
c
c       Output variables
c               d : first column contains vector x in bx=d
c               determ = 0. if matrix has zero determinant
c
      implicit double precision (a-h,o-z)
      DIMENSION id(40),b(40,40),d(40,3)
c
      DETERM=1.0
      DO 1 I=1,L
    1 ID(I)=0
      DO 18 NN=1,N
      BMAX=1.1
      DO 6 I=1,N
      IF(ID(I).NE.0) GOTO 6
      BNEXT=0.0
      BTRY=0.0
      DO 5 J=1,L
      IF(ID(J).NE.0) GOTO 5
      IF(ABS(B(I,J)).LE.BNEXT) GOTO 5
      BNEXT=ABS(B(I,J))
      IF(BNEXT.LE.BTRY) GOTO 5
      BNEXT=BTRY
      BTRY=ABS(B(I,J))
      JC=J
    5 CONTINUE
      IF(BNEXT.GE.BMAX*BTRY) GOTO 6
      BMAX=BNEXT/BTRY
      IROW=I
      JCOL=JC
    6 CONTINUE
      IF(ID(JC).EQ.0) GOTO 8
      DETERM=0.0
      RETURN
    8 ID(JCOL)=1
      IF(JCOL.EQ.IROW) GOTO 12
      DO 10 J=1,L
      SAVE=B(IROW,J)
      B(IROW,J)=B(JCOL,J)
   10 B(JCOL,J)=SAVE
      DO 11 K=1,M
      SAVE=D(IROW,K)
```

```
      D(IROW,K)=D(JCOL,K)
   11 D(JCOL,K)=SAVE
   12 F=1.0/B(JCOL,JCOL)
      DO 13 J=1,L
   13 B(JCOL,J)=B(JCOL,J)*F
      DO 14 K=1,M
   14 D(JCOL,K)=D(JCOL,K)*F
      DO 18 I=1,N
      IF(I.EQ.JCOL) GO TO 18
      F=B(I,JCOL)
      DO 16 J=1,L
   16 B(I,J)=B(I,J)-F*B(JCOL,J)
      DO 17 K=1,M
   17 D(I,K)=D(I,K)-F*D(JCOL,K)
   18 CONTINUE
      RETURN
      END
c
c
      SUBROUTINE SUMIK(XIK,J,IK,IKA,XIJKA,XIJKC,L,M,N,IDEC)
c
c     Sums the partial-current-density functions
c
      implicit double precision (a-h,o-z)
c
      dimension xik(41,2,101),xijka(l,m,n),xijkc(l,m,n)
c
      common/geom/pi,hol,nj,ndec,icut
      common/rxnpars/cref(10),cre(10),porq(2,10,5),ip,iq,
     1  uref(5),nspec,nrxn,rho0
      common/elecs/ian,icath
c
      do 100 jx=1,nj
        jxfine = ifine(idec,jx)
        do 90 ielec=ian,icath
   90   xik(ika,ielec,jx) = 0.
        xik(ika,ian,jx) = xik(ika,ian,jx) + xijka(ik,j,jxfine)
        xik(ika,icath,jx) = xik(ika,icath,jx) +
     1    xijkc(ik,j,jxfine)
  100 continue
c
      return
      end
c
c
      FUNCTION XK(XM1)
c
c     Complete elliptic integral of the first kind
```

```
c
      implicit double precision (a-h,o-z)
c
      common/geom/pi,hol,nj,ndec,icut
c
      a0 = 1.3862944d+00
      a1 = 0.1119723d+00
      a2 = 0.0725296d+00
      b0 = 0.5d0
      b1 = 0.1213478d+00
      b2 = 0.0288729d+00
c
      term = dlog(1./xm1)
      xk = a0 + a1*xm1 + a2*xm1*xm1 +
     1  (b0 + b1*xm1 + b2*xm1*xm1)*term
c
      return
      end
c
c
      SUBROUTINE SUMI(XI,XIJ)
c
c     Sums the partial current densities
c
      implicit double precision (a-h,o-z)
c
      dimension xi(2,101),xij(2,5,101)
c
      common/geom/pi,hol,nj,ndec,icut
      common/rxnpars/cref(10),cre(10),porq(2,10,5),ip,iq,
     1  uref(5),nspec,nrxn,rho0
      common/elecs/ian,icath
c
      do 100 ielec=ian,icath
      do 100 jx=1,nj
        xi(ielec,jx) = 0.d0
      do 100 j=1,nrxn
  100 xi(ielec,jx) = xi(ielec,jx) + xij(ielec,j,jx)
c
      return
      end
c
c
      FUNCTION IKP(IKIN,IELECIN,JIN)
c
c     Function for compressing three subscripts into one
c
      implicit double precision (a-h,o-z)
```

```
c
      common/colloc/npts,icoll(2,41),npextr,np(7)
      common/elecs/ian,icath
      common/rxnpars/cref(10),cre(10),porq(2,10,5),ip,iq,
     1  uref(5),nspec,nrxn,rho0
c
      ikp = 1
      if(ikin.eq.1) return
c
      do 100 j=1,nrxn
      do 100 ik=2,npts
      do 100 ielec=ian,icath
        ikp = ikp + 1
  100 if(j.eq.jin .and. ik.eq.ikin .and. ielec.eq.ielecin)
     1  return
c
      print *,'error in ikp,ikin=',ikin,'ielecin',ielecin,
     1  'jin',jin
      stop
c
      end
c
c
      SUBROUTINE INTCXI(NSPEC,XINT0,XINT1)
c
c     This subroutine calculates xint0(i,n-k+1,idec), 1/dx
c     times the integral from (k-1)dx to kdx of dC/dxi, where
c     C is the solution to the asymmetric Graetz problem in
c     channel flow and xi is the dimensionless normal variable,
c     y\h.
c
c
      implicit double precision (a-h,o-z)
      dimension xint0(10,101,5),xint1(10,101,5),array(5)
c
      common/evprob/xlamda(3),ai(3),z,diodr(10)
      common/geom/pi,hol,nj,ndec,icut
c
      a = 1.35659745d0
      b = 0.2d0
      c = 0.060733452d0
c
      aa = 0.9594d0
      bb = 0.6069d0
      cc = 0.4512d0
      dd = 0.276d0
c
      njm1 = nj-1
```

```
      n = njml
c
      do 1000 idec=1,ndec
      h = dfloat(icut)**(-ndec+idec)/dfloat(njml)
        do 999 ixi=1,2
        do 999 i=1,nspec
        do 999 k=1,n
          nmk = n-k
          nmkpl = n-k+1
          den = z/(h*diodr(i))
          zeta = nmk/den
          zetall = nmkpl/den
          if(ixi .eq. 2) go to 500
c
c         xi = 0
c
          if(zeta.lt.0.11d+00 .and. zetall.gt.0.11d+00)
     1      go to 600
          if(zeta.ge.0.11d+00) go to 100
          xint0(i,nmkpl,idec) = 1.5*den**(1./3.)*a*
     1      (nmk**(2./3.) - (nmk+1)**(2./3.))
     1      + b - 0.75*den**(-1./3.)*c*(nmk**(4./3.) -
     1      (nmk+1)**(4./3.))
          go to 999

c
  100     sum = 0.0d0
          do 110 j=1,3
  110     sum=sum+dabs(ai(j))/xlamda(j) * (dexp(-xlamda(j)*zeta)
     1      - dexp (-xlamda(j)*zetall))
          xint0(i,nmkpl,idec) = -1. - 2.*den*sum
          go to 999
c
  600     xint0(i,nmkpl,idec) = 1.5*den**(1./3.)*a*(nmk**(2./3.) -
     1      (0.11*den)**(2./3.)) + b*(0.11*den-nmk) -
     1      0.75*den**(-1./3.)*c*
     1      (nmk**(4./3.) - (0.11*den)**(4./3.))
          sum = 0.0d+00
          do 610 j=1,3
  610     sum = sum + dabs(ai(j))/xlamda(j) * (dexp(-xlamda(j)*0.11)
     1      - dexp(-xlamda(j)*zetall))
          sum = 0.11*den - nmkpl - 2.*den*sum
          xint0(i,nmkpl,idec) = xint0(i,nmkpl,idec) + sum
          go to 999
c
c         xi = 1
c
  500     if(zeta.le.0.18d+00 .and.zetall.ge.0.18d+00) go to 620
```

```
          if(zeta.gt.0.18d+00) go to 510
          f0 = 0.d+00
          if(n.eq.k) go to 505
          f0 = -dexp(aa - bb/zeta -cc*dexp(-dd/zeta))
  505     xintl(i,nmkpl,idec) = (-dexp(aa - bb/zetall -
     1      cc*dexp(-dd/zetall)) + f0)/2.
          go to 999
c
  510     continue
          sum = 0.d0
          do 520 j=1,3
  520     sum = sum + ai(j)/xlamda(j)*(dexp(-xlamda(j)*zeta) -
     1      dexp(-xlamda(j)*zetall))
          xintl(i,nmkpl,idec) = -1. + 2.*den*sum
          go to 999
c
  620     zll = 0.18d0
          f0 = 0.d0
          if(n.eq.k) go to 640
          f0 = -dexp(aa - bb/zeta - cc*dexp(-dd/zeta))
  640     xintl(i,nmkpl,idec) = (-dexp(aa - bb/zll -
     1     cc*dexp(-dd/zll)) + f0)/2. * (-nmk + zll*den)
          sum = 0.0d0
          do 660 j=1,3
  660     sum = sum + ai(j)/xlamda(j) * (dexp(-xlamda(j)*0.18)
     1      - dexp(-xlamda(j)*zetall))
          sum = 2.*den*sum + zll*den - nmkpl
          xintl(i,nmkpl,idec) = xintl(i,nmkpl,idec) + sum
  999   continue
        h = h*dfloat(icut)
 1000 continue
c
      istop = 0
      if(istop.ne.0) stop
      return
      end
c
c
      FUNCTION IFINE(IDECI,JXI)
c
c     Generates index for finely-spaced points
c
      implicit double precision (a-h,o-z)
c
      common/geom/pi,hol,nj,ndec,icut
c
      ifine = 0
      do 100 idec=1,ndec
```

```
      do 100 jx=1,nj
        ifine = ifine + 1
  100 if(idec.eq.ideci .and. jx.eq.jxi) return
c
      print *,'error in ifine,ideci=',ideci,'jxi',jxi
      stop
      end
c
c
      SUBROUTINE ERR(CORD,ITER,IVOLTS,VCELL,VPDELTA,XJ,PJ0,PJPD,
     1  JRXN,ERROR,ERRSAV,MATCALC,BSAVE,NITS,XIJKA,XIJKC,XIJ,
     1  XI,CSURF,PHISTR,XITOTJK,V,POMPS,POMPSK,PFINE,PTEMP,DELCO,
     1  COLD,CO,CANK,CCATHK,NCIK,NPTSTOT,IRSTRT,ERRTST,IERR,IREAD)
c
c     This program calculates the errors between the Butler-
c     Volmer currents and the approximate currents at the
c     collocation points
c
      implicit double precision (a-h,o-z)
c
      dimension csurf(2,10,101),errsav(2,5,101),
     1  xij(2,5,101),cik(40,40),xijnew(2,5,101),
     1  xi(2,101),xinew(2,101),limsp(2),xj(5),
     1  error(2,5,101),delc(2,10,101),cord(41),
     1  xitotjk(41,2,5),array(2,101),csave(2,10,101),
     1  bsave(40,40),xjdum(5),pfine(41,2,229),ptemp(2,229),
     1  p0mps(2,101),xijka(41,5,404),xitotj(2,5),
     1  p0mpsk(41,2,101),v(2),
     1  xijkc(41,5,404),cank(41,10,101),ccathk(41,10,101),
     1  co(41,2,5),cold(41,2,5),delco(41,2,5),b(40,40),
     1  d(40,3),dxijka(41,5,101),dxijkc(41,5,101)
c
      common/evprob/xlamda(3),ai(3),z,diodr(10),ciocr(10)
      common/co/coeff,xn
      common/geom/pi,hol,nj,ndec,icut
      common/rxnpars/cref(10),cre(10),porq(2,10,5),ip,iq,
     1  uref(5),nspec,nrxn,rho0
      common/kinpars/sij(10,5),ratson(10,5),ason(2,5),ifor,iback
      common/cdata/xint0(10,101,5),xint1(10,101,5),n2,nd,isec
      common/elecs/ian,icath
        common/gauss/xp(96),gw(96),ngt,xx(201),xielec(201),xiothr(20
     11),njmore
      common/colloc/npts,icoll(2,41),npextr,np(7)
      common/equalc/ikboth
c
      iitmax = 1
      if(matcalc.eq.1) iitmax = 20
      limelec = 2
```

```
      print *,'lim curr assumed on elec',limelec
c
      v(icath) = 0.d0
      v(ian) = vcell
c
      do 100 j=1,nrxn
  100 xjdum(j) = xj(j)
c
      xjdum(jrxn) = pj0
c
c     Calculate initial concentration
c
      if(irstrt.eq.0) then
        iend = 1
      else
        read(4,*),iend,iall
        print *,'iitstart',iend,'iall',iall
      end if
      iitf = iend + iitmax - 1
      do 440 iit=iend,iitf
c
c
        print *,'iteration iit',iit
        if(matcalc.eq.0) go to 230
        if(isec.ne.0) then
          do 110 ielec=ian,icath
          do 110 j=1,nrxn
          do 110 ik=1,npts
  110     co(ik,ielec,j) = cold(ik,ielec,j) + delco(ik,ielec,j)
          print *,'coefficients from delco at 110'
          print 120,co(1,1,1)
          print 120,(((co(ik,ielec,j),ielec=1,2),ik=2,npts),
     1      j=1,nrxn)
  120     format(1x,1pe15.8)
        end if
c
        if(iit.eq.1.and.iter.eq.1) then
c
          if(irstrt.eq.0) then
            j = 1
c           print *,'can,ccath cur fcn on anode'
c           print *,'points',(icoll(ian,ic),ic=2,npts)
            do 140 ik=2,npts
c             print *,'ik',ik
            do 140 ieleck=ian,icath
              call cfromi(xijka,xijkc,ik,j,ieleck,
     1          ikp(ik,ieleck,j),cank,ccathk)
c             if(ieleck.eq.icath) print 142,
```

```
c     1          ((dfloat(jx-1)/dfloat(nj-1),cank(ikp(ik,1,1),1,jx),
c     1        ccathk(ikp(ik,1,1),1,jx)),jx=1,nj)
  140       continue
  142       format(3(1pe11.4))
            istop = 0
            if(istop.ne.0) stop
c
c           print *,'potential fcns'
            ikboth = 0
            do 180 ik=2,npts
c
      nextra = 0
      do 203 jx=1,nj+28+npextr
        nextra = nextra + 1
        xol = xprt(jx,jreal)
        call eqfive(ik,xol,phian,phicath,p2,p3,p5)
        ptemp(ian,nextra) = phian
        ptemp(icath,nextra) = phicath
c       print 181,xol,p2,p3,phicath,p5,
c     1   phian
        if(jreal.ge.1 .and. jreal.le.nj) p0mps(ian,jreal) = phian
        if(jreal.ge.1 .and. jreal.le.nj) p0mps(icath,jreal)
     1    = phicath
  181   format(0pf8.4,x,1p5e12.4)
  203 continue
c
              do 146 j=1,nrxn
              do 146 ielec=ian,icath
                do 144 jx=1,nj
  144           p0mpsk(ikp(ik,ian,j),ielec,jx) = p0mps(ielec,jx)
                do 143 nextra = 1,nj+28+npextr
  143           pfine(ikp(ik,ian,j),ielec,nextra) = ptemp(ielec,nextra)
  146         continue
c
              do 147 j=1,nrxn
              do 147 ielec=ian,icath
                do 152 jx=1,nj
                p0mpsk(ikp(ik,icath,j),ielec,jx)= -p0mpsk(ikp(ik,ian,j),
     1            iother(ielec),jx)
c               if(ielec.eq.icath.and.ik.eq.2) print 148,
c     1           dfloat(jx-1)/dfloat(nj-1),
c     1           p0mpsk(ikp(ik,ian,j),
c     1           ian,jx),p0mpsk(ikp(ik,ian,j),icath,jx),p0mpsk(ikp
c     1           (ik,icath,j),ian,jx),p0mpsk(ikp(ik,icath,j),icath,jx)
  148           format(5(1pe11.4,1x))
  152           continue
                do 153 jx = 1,nj+28+npextr
  153           pfine(ikp(ik,icath,j),ielec,jx) = -pfine(ikp(ik,ian,j),
```

```
     1            iother(ielec),jx)
   147         continue
             do 180 ic=2,npts
               if(nrxn.eq.1) go to 180
               do 170 j=2,nrxn
               do 170 ieleck=ian,icath
               do 170 i=1,nspec
                 cank(ikp(ik,ieleck,j),i,icoll(ian,ic)) =
     1            cank(ikp(ik,ieleck,1),i,icoll(ian,ic))
   170         ccathk(ikp(ik,ieleck,j),i,icoll(icath,ic)) =
     1           ccathk(ikp(ik,ieleck,1),i,icoll(icath,ic))
   180       continue
           end if
c
         end if
c
         istop = 0
         if(istop.ne.0) stop
         icount = 0
c
c        print *,'calculate csave'
         do 182 i=1,nspec
         do 182 ic=2,npts
         do 182 ielec=ian,icath
             csave(ielec,i,icoll(ielec,ic)) = csurf(ielec,i,
     1         icoll(ielec,ic))
   182   continue
c
c
   183   continue
c
c        Update coefficients for the appropriate electrode(s)
c
         factor=1.0d0
c        if (iit.gt.4) factor=0.7d0
c        if (iit.gt.4) factor=0.5d0
         do 190 ielec=ian,icath
         do 190 j=1,nrxn
         do 190 ik=1,npts
   190   co(ik,ielec,j) = cold(ik,ielec,j)
     1     + delco(ik,ielec,j)*factor
         print *,'cold(1,1,1)',cold(1,1,1),'delco',delco(1,1,1)
c
         print *,'coefficients from delco'
         print 120,co(1,1,1)
         print 120,(((co(ik,ielec,j),ielec=1,2),ik=2,npts),
     1     j=1,nrxn)
c
```

```
c       if(iit.eq.1) go to 230
        do 210 i=1,nspec
        do 210 ic=2,npts
          csurf(ian,i,icoll(ian,ic)) = 0.d0
          csurf(icath,i,icoll(icath,ic)) = 0.d0
          do 200 j=1,nrxn
          do 200 ieleck=ian,icath
          do 200 ik=2,npts
            csurf(ian,i,icoll(ian,ic)) =
     1        co(ik,ieleck,j)*
     1        cank(ikp(ik,ieleck,j),i,icoll(ian,ic)) +
     1        csurf(ian,i,icoll(ian,ic))
            csurf(icath,i,icoll(icath,ic)) =
     1        co(ik,ieleck,j)
     1        *ccathk(ikp(ik,ieleck,j),i,icoll(icath,ic)) +
     1        csurf(icath,i,icoll(icath,ic))
  200     continue
        do 210 ielec=ian,icath
          jx = icoll(ielec,ic)
          csurf(ielec,i,jx) = ciocr(i) + csurf(ielec,i,jx)
  210   continue
  230   continue
        print *,'concentrations with new coeffs '
        print *,'species 1'
        print 280,((csurf(ielec,1,icoll(ielec,ic)),ielec=ian,icath),
     1    ic=2,npts)
c
c       Find the cathode coefficients that give zero concentration
c       on the cathode (if the cathode is the limiting electrode)
c
        print *,'limelec',limelec,'ian',ian,'icath',icath
        if(limelec.ne.ian .and. limelec.ne.icath) go to 75
        if(iit.eq.1) then
        j = 1
        do 90 irow = 2,npts
        do 90 ielecr = ian,icath
          jx = icoll(ielecr,irow)
          ib = ikp(irow,ielecr,j) - 1
          if(limelec.eq.ielecr) then
            csurf(ielecr,1,jx) = 0.d0
          else
            csurf(ielecr,1,jx) = 2.d0
          end if
          d(ib,1) = 0.d0
          if(irow.ne.npts) d(ib,1) = csurf(ielecr,1,jx) - ciocr(1)
        do 89 icol = 2,npts
        do 89 ielecc = ian,icath
          jb = ikp(icol,ielecc,j) - 1
```

```
          b(ib,jb) = 0.d0
          if(irow.ne.npts) then
            if(ielecr.eq.ian .and. ielecc.eq.ian) b(ib,jb) =
     1        cank(ikp(icol,ielecc,j),1,jx)
            if(ielecr.eq.icath .and. ielecc.eq.icath) b(ib,jb) =
     1        ccathk(ikp(icol,ielecc,j),1,jx)
          end if
          cik(ib,jb) = b(ib,jb)
   89   continue
        if(irow.eq.npts) then
          if(ielecr.eq.ian) then
            b(ib,1) = 1.d0
            cik(ib,1) = 1.d0
          else
            b(ib,4) = 1.d0
            cik(ib,4) = 1.d0
          end if
        end if
   90   continue
c
c       print *,'cik matrix'
c       print 92,((b(i,j),j=1,2*(npts-1)),i=1,(2*npts-1))
c       print *,'d vector'
c       print 92,(d(i,1),i=1,2*(npts-1))
  92    format(10(1x,1pe11.4))
        call matinv(2*(npts-1),2*(npts-1),1,b,d,determ)
c       print *,'after matinv d vector'
c       print 92,(d(i,1),i=1,2*(npts-1))
        if(determ.eq.0.d0) then
          print *,'zero determinant in cik'
          stop
        end if
c
        j = 1
        do 70 icol=2,npts
        do 70 ielecc=ian,icath
   70   co(icol,ielecc,j) = d(ikp(icol,ielecc,j)-1,1)
        print *,'coefficients from cik'
        print 120,co(1,1,1)
        print 120,(((co(ik,ielec,j),ielec=1,2),ik=2,npts),
     1    j=1,nrxn)
c
   75   continue
        do 76 ithru = 1,2
        do 60 i=1,nspec
        do 60 ic=2,npts
          csurf(ian,i,icoll(ian,ic)) = 0.d0
          csurf(icath,i,icoll(icath,ic)) = 0.d0
```

```
          do 50 j=1,nrxn
          do 50 ieleck=ian,icath
          do 50 ik=2,npts
            csurf(ian,i,icoll(ian,ic)) = co(ik,ieleck,j)*
     1        cank(ikp(ik,ieleck,j),i,icoll(ian,ic)) +
     1        csurf(ian,i,icoll(ian,ic))
            csurf(icath,i,icoll(icath,ic)) = co(ik,ieleck,j)
     1        *ccathk(ikp(ik,ieleck,j),i,icoll(icath,ic)) +
     1        csurf(icath,i,icoll(icath,ic))
   50     continue
          do 60 ielec=ian,icath
            jx = icoll(ielec,ic)
          csurf(ielec,i,jx) = ciocr(i) + csurf(ielec,i,jx)
   60   continue
        print *,'concentrations with cik coeffs '
        print *,'species 1'
        print 280,((csurf(ielec,1,icoll(ielec,ic)),ielec=ian,icath),
     1    ic=2,npts)
        if(ithru.gt.1) go to 76
        j = 1
        do 990 irow = 2,npts
        do 990 ielecr = ian,icath
          jx = icoll(ielecr,irow)
          ib = ikp(irow,ielecr,j) - 1
          d(ib,1) = 0.d0
          if(irow.ne.npts) then
            if(ielecr.eq.ian) d(ib,1) = 2. - csurf(ielecr,1,jx)
            if(ielecr.eq.icath) d(ib,1) = - csurf(ielecr,1,jx)
          end if
  990   continue
        call matinv(2*(npts-1),2*(npts-1),1,cik,d,determ)
        print *,'after matinv delta coeffs'
        print 92,(d(i,1),i=1,2*(npts-1))
c
        j = 1
        do 970 icol=2,npts
        do 970 ielecc=ian,icath
  970   co(icol,ielecc,j) = d(ikp(icol,ielecc,j)-1,1) +
     1    co(icol,ielecc,j)
c
        print *,'new coefficients from cik'
        print 120,co(1,1,1)
        print 120,(((co(ik,ielec,j),ielec=1,2),ik=2,npts),
     1    j=1,nrxn)
   76   continue
c
        end if
        istop = 0
```

```
        if(istop.ne.0) stop
c
  238   do 250 j=1,nrxn
        do 250 ic=2,npts
          do 240 ielec=ian,icath
          jx = icoll(ielec,ic)
  240     xij(ielec,j,jx) = 0.
        do 250 ik=2,npts
          jxa = icoll(ian,ic)
          jfinea = ifine(ndec,jxa)
          jxc = icoll(icath,ic)
          jfinec = ifine(ndec,jxc)
          xij(ian,j,jxa) = co(ik,ian,j)*xijka(ik,j,jfinea) +
     1      xij(ian,j,jxa)
          xij(icath,j,jxc) = co(ik,icath,j)*xijkc(ik,j,jfinec) +
     1      xij(icath,j,jxc)
  250   continue
c
        call sumi(xi,xij)
c
        do 270 ic=2,npts
        do 270 ielec=ian,icath
          jx = icoll(ielec,ic)
          p0mps(ielec,jx) = 0.
          do 260 j=1,nrxn
          do 260 ieleck=ian,icath
          do 260 ik=2,npts
            p0mps(ielec,jx) = co(ik,ieleck,j)*
     1        p0mpsk(ikp(ik,ieleck,j),ielec,jx) + p0mps(ielec,jx)
  260     continue
  270   continue
c
c       print *,'concentrations input to B-V'
c       print *,'species 1'
c       print 280,((csurf(ielec,1,icoll(ielec,ic)),ielec=ian,icath),
c    1    ic=2,npts)
  280   format(2(1x,1pe15.8))
        print *,'    '
        istop = 0
        if(istop.ne.0) stop
        xnew = co(1,1,1)
        id1 = 1
        if(matcalc.eq.0) id1 = 2
        do 310 id=1,id1
          if(id.gt.1) then
            if(ivolts.eq.1) then
              v(icath) = 0.
              v(ian) = vpdelta
```

```
            else
              xjdum(jrxn) = pjpd
            end if
          end if
c
          iitcut = 2
          max = 1
c         if(iit.eq.1 .and. matcalc.ne.0) max = 40
          if(matcalc.ne.0.and.iit.lt.iitcut) max = 50
          do 295 inner=1,max
            print *,'inner',inner,'phistr',xnew
            call current(xij,xijka,xijkc,iall,ynew,xnew,
     1        xitotc,xitota,p0mps,csurf,cank,ccathk,xijnew,xinew,
     1        xitotjk,p0mpsk,dxijka,dxijkc,v,xjdum)
              nav = (npts+2)/2
            if(limelec.ne.ian) then
              delphi = (xij(ian,1,icoll(ian,nav)) - xijnew(ian,1,
     1          icoll(ian,nav))) / dxijka(1,1,icoll(ian,nav))
            else
              delphi = (xij(icath,1,icoll(icath,nav)) - xijnew(icath,1,
     1          icoll(icath,nav))) / dxijkc(1,1,icoll(icath,nav))
            end if
            if(dabs(delphi).lt.1.d-02 .or. max.eq.1) go to 296
            print *,'delphi',delphi,'xij',xij(ian,1,icoll(ian,nav)),
     1        'xijnew',xijnew(ian,1,icoll(ian,nav))
            if(dabs(delphi).gt.4.d0) delphi = 4.d0*dabs(delphi)/delphi
            xnew = xnew + delphi
  295     continue
          print *,'no convergence on initial guess for phi *'
          stop
  296     co(1,1,1) = xnew
c
  297     phistr = xnew
          print *,'    error(elec)       iBV(elec)      iapp(cath)'
          if(iall.eq.0) then
            jx1 = 2
            jx2 = npts
          else
            jx1 = 1
            jx2 = nj
          end if
          do 300 j=1,nrxn
          do 300 ielec=ian,icath
          do 300 ic=jx1,jx2
            if(iall.eq.0) then
              jx = icoll(ielec,ic)
            else
              jx = ic
```

```
            end if
            if(id.eq.1) then
              error(ielec,j,jx) = xijnew(ielec,j,jx) -
     1          xij(ielec,j,jx)
              if(ielec.eq.ian) print 290,dfloat(jx-1)/dfloat(nj-1),
     1          error(ian,j,jx),xijnew(ian,j,jx),xij(ian,j,jx)
              if(ielec.eq.icath) print 290,dfloat(jx-1)/dfloat(nj-1),
     1          csurf(icath,1,jx),
     1          xijnew(icath,j,jx),xij(icath,j,jx)
            else
              errsav(ielec,j,jx) = xijnew(ielec,j,jx) -
     1        xij(ielec,j,jx)
            end if
  290       format(1x,0pf8.4,1x,1pe20.13,2(1x,1pe11.4))
  300     continue
  310   continue
c
        istop = 0
        if(istop.ne.0) stop
        if(matcalc.eq.0) return
        ierr = 0
c
        do 320 j=1,nrxn
        do 320 ielec=ian,icath
        do 320 ic=2,npts
          if(dabs(error(ielec,j,icoll(ielec,ic))*xijnew(ielec,j,
     1      icoll(ielec,ic))) .gt. errtst) then
            ierr=ierr+1
c           print *,'ielec',ielec,'jx',icoll(ielec,ic),'error',
c    1      error(ielec,j,icoll(ielec,ic)),'xijnew',
c    1      xijnew(ielec,j,icoll(ielec,ic)),'errtst',errtst
          end if
  320   continue
c
c       if(iit.eq.iitmax .or. ierr.eq.0 .or. iuncon.eq.0) then
          print *,'calc concentrations everywhere'
          do 340 jx=1,nj
          do 340 i=1,nspec
            csurf(ian,i,jx) = 0.d0
            csurf(icath,i,jx) = 0.d0
            do 330 j=1,nrxn
            do 330 ieleck=ian,icath
            do 330 ik=2,npts
              csurf(ian,i,jx) = co(ik,ieleck,j)*
     1          cank(ikp(ik,ieleck,j),i,jx) + csurf(ian,i,jx)
  330       csurf(icath,i,jx) = co(ik,ieleck,j)*
     1        ccathk(ikp(ik,ieleck,j),i,jx) + csurf(icath,i,jx)
          do 340 ielec=ian,icath
```

```
  340     csurf(ielec,i,jx) = ciocr(i) + csurf(ielec,i,jx)
          print *,'species 1'
          print 280,((csurf(ielec,1,icoll(ielec,ic)),ielec=ian,icath),
     1      ic=2,npts)
c
          do 350 jx=1,nj
            jxfine = ifine(ndec,jx)
          do 350 ielec=ian,icath
            p0mps(ielec,jx) = 0.d0
          do 350 j=1,nrxn
            xij(ielec,j,jx) = 0.d0
          do 350 ik=2,npts
            ia = ikp(ik,ian,j)
            ic = ikp(ik,icath,j)
            if(ielec.eq.ian) xij(ian,j,jx) = co(ik,ian,j)*
     1        xijka(ik,j,jxfine) + xij(ian,j,jx)
            if(ielec.eq.icath) xij(icath,j,jx) = co(ik,icath,j)*
     1        xijkc(ik,j,jxfine) + xij(icath,j,jx)
          do 350 ieleck=ian,icath
  350     p0mps(ielec,jx) = co(ik,ieleck,j)*
     1      p0mpsk(ikp(ik,ieleck,j),ielec,jx) + p0mps(ielec,jx)
c
          do 351 nextra = 1,nj+28+npextr
          do 351 ielec = ian,icath
            ptemp(ielec,nextra) = 0.d0
          do 351 j=1,nrxn
          do 351 ik=2,npts
            ia = ikp(ik,ian,j)
            ic = ikp(ik,icath,j)
          do 351 ieleck=ian,icath
  351     ptemp(ielec,nextra) = co(ik,ieleck,j)*pfine(ikp(ik,ieleck,j),
     1      ielec,nextra) + ptemp(ielec,nextra)
c
c
          call sumi(xi,xij)
c       end if
c
c       Set up matrix equation for calculating new current coefficients
c       (and concentration ordinates, if they are being used)
c
c
        do 370 ik=2,npts
        do 370 ielec=ian,icath
        do 370 j = 1,nrxn
  370   b(1,ikp(ik,ielec,j)) = xitotjk(ik,ielec,j)
c
        d(1,1) = 0.d0
        do 380 ielec = ian,icath
```

```
        do 380 j=1,nrxn
        do 380 ik=2,npts
  380   d(1,1) = -co(ik,ielec,j)*xitotjk(ik,ielec,j) + d(1,1)
c
        do 400 jc=1,nrxn
        do 400 ic=2,npts
        do 400 ielecc=ian,icath
          if(ielecc.eq.ian) then
            b(ikp(ic,ielecc,jc), 1) = dxijka( 1, jc, icoll(ian,ic) )
            if(limelec.eq.ian) b(ikp(ic,ielecc,jc), 1)
     1        = 0.d0
          else
            b(ikp(ic,ielecc,jc), 1) = dxijkc( 1, jc, icoll(icath,ic) )
            if(limelec.eq.icath) b(ikp(ic,ielecc,jc), 1)
     1        = 0.d0
          end if
          if(ic.eq.npts) b(ikp(ic,ielecc,jc),1) = 0.d0
          do 390 jk=1,nrxn
          do 390 ik=2,npts
          do 390 ieleck=ian,icath
            ib = ikp(ic,ielecc,jc)
            jb = ikp(ik,ieleck,jk)
            if(ielecc .eq. ian) then
              if(limelec .eq. ian) then
                if(ieleck.eq.ian) then
                  b(ib,jb) = cank(jb,1,icoll(ian,ic))
                else
                  b(ib,jb) = 0.d0
                end if
              else
                xijk = xijka(ik, jc, ifine(ndec,icoll(ian,ic)))
                b( ib, jb ) = dxijka( jb, jc, icoll(ian,ic))
                if(ielecc .eq. ieleck .and. jk.eq.jc) then
                  b(ib,jb) = b(ib,jb) - xijk
                end if

              end if
            else
              if (limelec.eq.icath) then
                if(ieleck.eq.icath) then
                  b(ib,jb) = ccathk(jb,1,icoll(icath,ic))
                else
                  b(ib,jb) = 0.d0
                end if
              else
                xijk = xijkc(ik, jc, ifine(ndec,icoll(icath,ic)))
                b( ib ,jb ) = dxijkc( jb, jc, icoll(icath,ic))
                if(ielecc .eq. ieleck .and. jk.eq.jc) then
```

```
                b(ib,jb) = b(ib,jb) - xijk
              end if
            end if
          end if
          if(ic.eq.npts) then
            b(ib,jb) = 0.d0
            if(ik.eq.2.and.ieleck.eq.ian.and.ielecc.eq.ian)
     1        b(ib,jb) = 1.d0
            if(ik.eq.3.and.ieleck.eq.icath.and.ielecc.eq.icath)
     1        b(ib,jb) = 1.d0
          end if
  390     continue
          if(ielecc.eq.limelec) then
            d(ib,1) = -csurf(ielecc,1,icoll(ielecc,ic))
            print *,'-csurf',d(ib,1)
          else
            d(ib,1) = -error(ielecc,jc,icoll(ielecc,ic))
          end if
          if(ic.eq.npts) d(ib,1) = 0.d0
  400   continue
c       do 405 ik=1,npts
c       do 405 ieleck = ian,icath
c         jb = ikp(ik,ieleck,1)
c         b(ikp(npts,ian,1),jb) = 0.d0
c         print *,'ikp,jb',ikp(npts,ian,1),jb,'b',b(ikp(npts,ian,1),jb)
c 405   continue
c       b(ikp(npts,ian,1),ikp(npts,ian,1)) = 1.d0
c       print *,'ikp,ikp',ikp(npts,ian,1),ikp(npts,ian,1),'b',b(ikp(
c    1    npts,ian,1),ikp(npts,ian,1))
c       d(ikp(npts,ian,1),1) = 0.d0
c
        do 410 i=1,nptstot
        do 410 j=1,nptstot
  410   bsave(i,j) = b(i,j)
        print *,'in err bsave matrix'
        print 420,((bsave(i,j),j=1,nptstot),i=1,nptstot)
  420   format(10(1x,1pe11.4))
        print *,'d vector'
        print 420,(d(i,1),i=1,nptstot)
        call matinv(nptstot,nptstot,1,b,d,determ)
        print *,'after matinv, d vector'
        print 420,(d(i,1),i=1,nptstot)
        if(determ.eq.0.) then
          print *,'zero det in matinv'
          print *,'b matrix'
          print 420,((b(i,j),j=1,nptstot),i=1,nptstot)
          stop
        end if
```

```
c
c       Don't let phi * change by more than 4
c
        iuncon = 0
        coldsv = cold(1,1,1)
        do 430 j=1,nrxn
        do 430 ik=1,npts
        do 430 ielec=ian,icath
          delco(ik,ielec,j) = d(ikp(ik,ielec,j), 1)
          if (dabs(delco(ik,ielec,j)).gt.1.d-12) iuncon = 1
c         delco(ik,ielec,j) = 0.d0
  430   cold(ik,ielec,j) = co(ik,ielec,j)
        if(dabs(delco(1,1,1)).gt.4.d0)
     1    delco(1,1,1) = 4.d0*delco(1,1,1)/dabs(delco(1,1,1))
c       delco(2,2,1) = 1.d-06
        print *,'delco(1,1,1)',delco(1,1,1),'cold',cold(1,1,1)
c
          nits = iit
        if(ierr.eq.0 .or. iuncon.eq.0) then
          ierr = 0
          return
        end if
c
  440 continue
c
      ierr = 0
      if(matcalc.eq.1) ierr = 1
c
      return
      end
c
c
      SUBROUTINE NLIM(FLUXCK,IK)
c
c     Calculates the flux due to a step change on one electrode
c     and zero flux on the other electrode
c
      implicit double precision (a-h,o-z)
      dimension fluxak(41,10,404),fluxck(41,10,404),
     1  cank(41,10,101),sum0(2,10),suml(2,10),cafine(41,10,404),
     1  b(40,40),d(40,1),asum0(2,10),asuml(2,10)
c
      common/cdata/xint0(10,101,5),xintl(10,101,5),n2,nd,isec
      common/elecs/ian,icath
      common/rxnpars/cref(10),cre(10),porq(2,10,5),ip,iq,
     1  uref(5),nspec,nrxn,rho0
      common/kinpars/sij(10,5),ratson(10,5),ason(2,5),ifor,iback
      common/geom/pi,hol,nj,ndec,icut
```

```
      common/co/coeff,xn
      common/evprob/xlamda(3),ai(3),z,diodr(10),ciocr(10)
c
c
      cdamp = 1.0d-06
      n2 = 2*nspec
      jbegin = 2
      kend = (nj-1)/icut + 1
c
c
      i = 1
      do 10 jx=1,nj
   10 cank(ik,i,jx) = ciocr(i)
c
      do 19 idec=1,ndec
        h = dfloat(icut)**(idec-ndec)/dfloat(nj-1)
        do 17 jx=jbegin,nj
          zeta = (jx-1)*h*diodr(i)/z
          jxfine = ifine(idec,jx)
c
          do 11 ielec=ian,icath
   11     sum0(ielec,i) = 0.d0
c
          if(jx.eq.2) go to 13
          n = jx - 1
          nm1 = n - 1
c
          do 12 k=1,nm1
            kp1 = k + 1
            nmkp1 = n - k + 1
c         term = xint0(i,nmkp1,idec) * (cank(ik,i,kp1)-cank(ik,i,k))
c         print *,'jxfine',jx,' k',k,' term',term,'xint0(i,nmkp1,idec)',
c    1      xint0(i,nmkp1,idec)
   12     sum0(ian,i) = sum0(ian,i) + (cank(ik,i,kp1)
     1      -cank(ik,i,k))*xint0(i,nmkp1,idec)
c
   13     continue
c
          bb = -xint0(i,1,idec)
          dd = sum0(ian,i) - cank(ik,i,jx-1)*
     1      xint0(i,1,idec) - flak(zeta)
c
          cank(ik,i,jx) = dd/bb
   17   continue
        do 20 jx=1,nj
   20   cafine(ik,i,ifine(idec,jx)) = cank(ik,i,jx)
        if(idec.eq.ndec) go to 19
        do 18 k=1,kend
```

```
          jj = k*icut - icut + 1
   18     cank(ik,i,k) = cank(ik,i,jj)
          jbegin = kend + 1
   19   continue
        istop = 0
        if(istop.ne.0) stop
c
        jbegin = 2
c
        do 190 idec=1,ndec
          h = dfloat(icut)**(idec-ndec)/dfloat(nj-1)
          do 170 jx=jbegin,nj
            zeta = (jx-1)*h*diodr(i)/z
            jxfine = ifine(idec,jx)
            do 110 ielec=ian,icath
              asum0(ielec,i) = 0.d0
  110       asum1(ielec,i) = 0.d0
c
            if(jx.eq.1) go to 130
            n = jx - 1
c
            do 120 k=1,n
              kp1 = k + 1
              nmkp1 = n - k + 1
              asum0(ian,i) = asum0(ian,i) + (cafine(ik,i,
     1          ifine(idec,kp1))
     1          -cafine(ik,i,ifine(idec,k)))*xint0(i,nmkp1,idec)
              asum1(ian,i) = asum1(ian,i) + (cafine(ik,i,
     1          ifine(idec,kp1))
     1          -cafine(ik,i,ifine(idec,k)))*xint1(i,nmkp1,idec)
  120       continue
c
  130       continue
c
            do 140 ielec=ian,icath
              iothr = iother(ielec)
              if(ielec.eq.ian) then
                fluxak(ik,i,jxfine) = (+asum0(ielec,i)
     1            - flak(zeta))*coeff*diodr(i)
              else
                fluxck(ik,i,jxfine) = (flck(zeta)
     1            - asum1(iothr,i))*coeff*diodr(i)
              end if
  140       continue
  170     continue
c
          fluxck(ik,i,ifine(idec,1)) = 1.d30*coeff*diodr(i)
          fluxak(ik,i,ifine(idec,1)) = 0.d0
```

```
        jbegin = kend + 1
  190 continue
c
c     fill in the other decades
c
      if(ndec.eq.1) return
      do 180 idec=2,ndec
      do 180 k=1,kend
        jj = k*icut - icut + 1
  180 fluxck(ik,i,ifine(idec,k)) = fluxck(ik,i,ifine(idec-1,jj))
c
c     do 200 idec=1,ndec
c       print *,'idec',idec
c     do 200 jx=1,nj
c      jxfine = ifine(idec,jx)
c      print *,'jx',jx,'fluxc',fluxck(ik,1,jxfine),'fluxa',
c    1 fluxak(ik,1,jxfine)
c 200 continue
      istop=0
      if(istop.ne.0) stop
      return
      end
c
c
      FUNCTION P(N,X)
c
      implicit double precision (a-h,o-z)
c
c     Calculation of Legendre polynomials
c
      p1 = 1.0
      p2 = 2.0*x - 1.0
      if (n-1)  1,2,3
    1 p = p1
      return
    2 p = p2
      return
    3 nm1 = n - 1
      do 4 nu=1,nm1
      p = ((x*dfloat(4*nu+2)-(2*nu+1))*p2 - dfloat(nu)*p1) /
     1  dfloat(nu+1)
      p1 = p2
    4 p2 = p
c
      return
      end
c
c
```

```
      FUNCTION FLAK(ZETA)
c
c     Solution to Graetz problem:
c     flux due to a step change in concentration on the other electrode
c
      implicit double precision (a-h,o-z)
      common/evprob/xlamda(3),ai(3),z,diodr(10),ciocr(10)
c
      aa = 0.9594d+00
      bb = 0.6069d+00
      cc = 0.4512d+00
      dd = 0.276d+00
      flak = 0.
      if(zeta.eq.0.) return
c
      if(zeta.gt.0.18d0) then
        sum = 0.d0
        do 52 j=1,3
   52   sum = sum + ai(j)*dexp(-xlamda(j)*zeta)
        flak = 1. - 2.*sum
      else
        flak = dexp(aa-bb/zeta - cc*dexp(-dd/zeta))
      end if
      return
      end
c
c
      FUNCTION FLCK(ZETA)
c
c     Solution to Graetz problem:
c     flux due to a step change in concentration on the same electrode
c
      implicit double precision (a-h,o-z)
      common/evprob/xlamda(3),ai(3),z,diodr(10),ciocr(10)
c
      a = 1.35659745d0
      b = 0.2d0
      c = 0.060733452d0
c
      if(zeta.eq.0.) then
        flck = 1.d30
        return
      end if
c
      if (zeta.lt.0.11) then
        flck = a*zeta**(-1./3.) - b - c*zeta**(1./3.)
      else
        sum = 0.d0
```

```
        do 100 j=1,3
  100   sum = sum + dabs(ai(j))*dexp(-xlamda(j)*zeta)
      end if
c
      return
      end
c
c
      SUBROUTINE EQFIVE(IK,XOL,PHIAN,PHICATH,P2,P3,P5)
c
c
c     This subroutine calculates the dimensionless form of the
c     ((phi 0) - (phi *))/i<avg> given by equation 5 in Newman
c     and Parrish, JES, 117, 43-48, 1970.  The singularity in
c     the equation was eliminated by the method of Kantorovich
c     and Krylov
c
      implicit double precision (a-h,o-z)
c
      common/geom/pi,hol,nj,ndec,icut
        common/gauss/xp(96),gw(96),ngt,xx(201),xielec(201),xiothr(20
     11),njmore
      common/elecs/ian,icath
      common/equalc/ikboth
      common/params/deltal,delta2,delta3
c
      ielec=ian
      iothr = icath
      c = pi/(2.*hol)
      an = 0.5d0
c
      if(xol.gt.0.d0 .and. xol.lt.1.d0) then
          xmin = xol/2.
          xmax = xmin + 0.5
          xie = curfcn(ik,ian,xol)
          if(ik.ne.ikboth) then
            xio = 0.d0
          else
            xio = curfcn(ik,icath,xol)
          end if
      else
          xmin = 0.5
          xmax = xmin
          xie = 0.
          xio = 0.
      end if
  130   sumra = 0.d0
        sumrb = 0.d0
```

```
        sumrc = 0.d0
cf     sumrf = 0.d0
        sumrs = 0.d0
        sumrt = 0.d0
c
        do 180 iregn=1,4
        go to (101,102,103,104),iregn
  101       xll = 0.d0
            xul = xmin**an
            go to 150
  102       if(xol.ge.1.d0 .or. xol.le.0.d0) go to 180
            if(xol.ge.0.75) then
              xul = (1.-xmin)**an
              xll = (1.-xol)**an
            else
              xll = xmin**an
              xul = xol**an
            end if
            go to 150
  103       if(xol.le.0.d0 .or. xol.ge. 1.d0) go to 180
            if(xol.le.0.25) then
              xll = xol**an
              xul = xmax**an
            else
              xul = (1.-xol)**an
              xll = (1.-xmax)**an
            end if
            go to 150
  104       xul = (1.-xmax)**an
            xll = 0.d0
  150     suma = 0.d0
          sumb = 0.d0
          sumc = 0.d0
cf     sumf = 0.d0
          sums = 0.d0
          sumt = 0.d0
c
          do 170 jxp=1,ngt
            xprm = xp(jxp)*(xul-xll) + xll
            z = xprm**(1./an)
            if ((iregn.eq.3.and.xol.gt.0.25) .or. iregn.eq.4 .or.
     1        (iregn.eq.2.and.xol.gt.0.75)) z = 1. - z
            y = c*(xol-z)
            sinh2 = (dexp(y) - dexp(-y))**2/4.0
            if(dabs(y).lt.0.01) sinh2 = (y*(1. + y**2/6.*(1. +
     1          y**2/20.*(1. + y**2/42.))))**2
            cosh2 = 1. + sinh2
            if(y.ne.0) alnth2=dlog(sinh2/cosh2)
```

```
c
            aa = curfcn(ik,ian,z)
            if(ik.ne.ikboth) then
              bb = aa
            else
              bb = aa + curfcn(ik,icath,z)
            end if
            Y2 = xprm**(1./an - 1.) / an
            clnch2 = dlog(cosh2)*Y2
c
            if(y.ne.0.) suma = suma + gw(jxp)*(aa*alnth2 - xie*dlog(y**2))*Y2
          if(y.ne.0.)sumb=sumb+gw(jxp)*((bb-aa)*alnth2-xio*dlog(y**2))*Y2
            sumc = sumc + gw(jxp)*bb*clnch2
cf        sumf = sumf + gw(jxp)*bb*(z-0.5)*Y2
            sums = sums + gw(jxp)*bb*Y2
            sumt = sumt + gw(jxp)*clnch2
  170     continue
c
          sumra = suma/2.*(xul-xll) + sumra
          sumrb = sumb/2.*(xul-xll) + sumrb
          sumrc = sumc/2.*(xul-xll) + sumrc
cf      sumrf = sumf/2.*(xul-xll) + sumrf
          sumrs = sums/2.*(xul-xll) + sumrs
          sumrt = sumt/2.*(xul-xll) + sumrt
  180   continue
c
          g2 = 0.d0
        if(xol.lt.1.d0 .and. xol.gt.0.d0) g2 = dlog(c**2) +
     1    2.*(1.-xol)*(dlog(1.-xol) - 1.) + 2.*xol*(dlog(xol) - 1.)
c
cf      p1 = sumrf/(2.*hol)
        if(ik.ne.ikboth) sumrs = 0.d0
        p2 = -sumra/(2.*pi)
        p3 = -(g2*xie+sumra)/(2.*pi)
        p5 = -(sumrc-sumrs*sumrt)/(2.*pi)
        phicath = -(g2*xio + sumrb)/(2.0*pi)+p5
        phian = -(g2*xie + sumra)/(2.0*pi)+p5
c
      return
      end
c
c
      SUBROUTINE READIN(XN,XJ,UJTH,NSUBJ,NG,ROOT,
     1  NRE,M,IRSTRT,VCELL,VSTART,JRXN,XJJ,IVOLTS,DELTAP,
     1  ERRTST,MAXIT,IREAD)
c
c     This subroutine reads and prints the input data
c
```

```
      implicit double precision (a-h,o-z)
      dimension root(96),ujth(5),alpha(2,5),
     1  nsubj(5),xj(5)
c
      common/evprob/xlamda(3),ai(3),z,diodr(10),ciocr(10)
      common/geom/pi,hol,nj,ndec,icut
      common/rxnpars/cref(10),cre(10),porq(2,10,5),ip,iq,
     1  uref(5),nspec,nrxn,rho0
      common/kinpars/sij(10,5),ratson(10,5),ason(2,5),ifor,iback
      common/cdata/xint0(10,101,5),xint1(10,101,5),n2,nd,isec
        common/gauss/xp(96),gw(96),ngt,xx(201),xielec(201),xiothr(20
     11),njmore
      common/colloc/npts,icoll(2,41),npextr,np(7)
      common/params/delta1,delta2,delta3
c
      read 120,ichar
      read  *,ng
      if(ng.gt.48 .or. ng.lt.1) print 100,ng
  100 format(' ng=',i2,' is incorrect input')
c
c     Read the Gaussian quadrature roots and weights for the
c     interval 0 to 1.  Then put in the roots and weights for
c     the interval -1 to 0.
c
      do 110 k=1,ng
        karg = ng + 1 - k
        kpng = k + ng
        read  *,(root(karg),gw(karg))
        root(kpng) = root(karg)
        gw(kpng) = gw(karg)
  110 root(karg) = -root(karg)
c
      read *,(xlamda(i),i=1,3)
      read *,(ai(i),i=1,3)
      read 120,ichar
  120 format(a72)
      read *,nj,ndec,irstrt
      print 130,irstrt
  130 format(' restart=',i3)
      if(nj.gt.101 .or. nj.lt.1) print 140,nj
  140 format(' nj=',i3,' is incorrect input')
      if(ndec.gt.5 .or. ndec.lt.1) print 150,ndec
  150 format(' ndec=',i3,' is incorrect input')
c
      read 120,ichar
      read *,hol,icut,xn,pehol,rho0
      read 120,ichar
      read *,vcell,vstart,xjj,jrxn,ivolts,deltap,errtst,maxit,
```

```
     1  ir,m
      if(ivolts.eq.1) then
        print *,'continuation on voltage'
      else
        print *,'continuation on J(',jrxn,')'
      end if
      print 155,errtst,maxit
  155 format('convergence criterion',1pe11.4,
     1  'maximum iterations',i2)
      print 220,nj,ndec,hol,icut,xn,pehol,rho0,vcell
  220 format('nj=',i3,' ndec=',i3,' hol=',1pe11.4,' icut=',i3,
     1  ' xn=',1pe11.4,/,' pehol=',1pe11.4,
     1  ' rho0=',1pe11.4,' v(cell)=',1pe11.4)
      print *,'principal reactant',ir,'main rxn',m
c
      z=3./16. * pehol
c
      read 120,ichar
      read *,isec,nspec,nrxn,nre
c
      if(isec.ne.0) print 230
  230 format(' secondary current distribution')
      print 240,nspec,nrxn,nre
  240 format(i3,' species,',i3,
     1  ' reactions, reference rxn is rxn',i3,/)
c
      if(nspec.gt.10 .or. nspec.lt.1) print 250,nspec
  250 format(' nspec=',i3,' is incorrect input')
      if(nrxn.gt.5 .or. nrxn.lt.1) print 260,nrxn
  260 format(' nrxn=',i3,' is incorrect input')
      if(nre.gt.5 .or. nre.lt.1) print 270,nre
  270 format(' nre=',i3,' is incorrect input')
c
      read 120,ichar
      read *,ujth(nre)
      do 310 j=1,nrxn
        read 120,ichar
        read *,alpha(1,j),alpha(2,j),nsubj(j),xj(j),ujth(j)
        if(j.eq.nre .and. ujth(j).ne.ujth(nre)) print *,
     1    'error in ujth(nre)'
        print 300,alpha(1,j),alpha(2,j),xj(j),ujth(j),nsubj(j)
  300   format(' alpha(a)=',1pe11.4,' alpha(c)=',1pe11.4,
     1    ' xj=',1pe11.4,' ujth=',1pe11.4,/,' nsubj=',i2)
  310 continue
c
c
      do 340 i=1,nspec
        read 120,ichar
```

```
        read *,diodr(i),cref(i),cre(i),ciocr(i)
        print 330,diodr(i),cref(i),cre(i),ciocr(i)
  330   format(' diodr=',1pe11.4,' cref=',1pe11.4,
     1    ' cre=',1pe11.4,' ciocr=',1pe11.4)
  340 continue
c
      do 370 i=1,nspec
        read 120,ichar
        read *,(sij(i,j),j=1,nrxn)
        do 360 j=1,nrxn
          print 350,sij(i,j),i,j
  350     format(' sij=',1pe11.4,' i=',i2,' j=',i2)
  360   continue
  370 continue
c
      do 400 j=1,nrxn
        ason(1,j) = -alpha(1,j)*sij(ir,m)/nsubj(m)
        ason(2,j) = -alpha(2,j)*sij(ir,m)/nsubj(m)
        print *,'ason(1,j)=',ason(1,j),' ason(2,j)=',ason(2,j)
        do 390 i=1,nspec
          ratson(i,j)=(sij(i,j)/nsubj(j))/(sij(ir,m)/nsubj(m))
          print 380,ratson(i,j),i,j
  380     format(' ratson=',1pe11.4,' i=',i2,' j=',i2)
  390   continue
  400 continue
c
      read 120,ichar
      read *,npts
      read 120,ichar
      read *,(icoll(1,ic),ic=2,npts)
      read 120,ichar
      read *,(icoll(2,ic),ic=2,npts)
      read 120,ichar
      read *,iread
      read 120,ichar
      read *,delta1,delta2,delta3
      read 120,ichar
      read *,npextr
      np(1) = npextr/2
      np(2) = 1 + np(1)
      np(3) = 8 + np(1)
      np(4) = 13 + np(1)
      np(5) = 16 + np(1)
      np(6) = 21 + np(1)
      np(7) = 29 + np(1)
      return
      end
c
```

```
c
      function xprt(jx,jreal)
c
      implicit double precision (a-h,o-z)
c
      common/colloc/npts,icoll(2,41),npextr,np(7)
      common/geom/pi,hol,nj,ndec,icut
c
        xprt = dfloat(jx-np(2))/dfloat(nj-1)
        jreal = 0
        if(jx.eq.np(3)) jreal = 1
        if(jx.ge.np(5).and.jx.le.nj+np(4)) then
          jreal = jx - (np(4)+1)
          xprt = dfloat(jreal-1)/dfloat(nj-1)
        end if
        if(jx.eq.nj+np(6)) jreal = nj
        if(jx.gt.np(1) .and. jx.lt.np(5)) xprt = dfloat(jx-np(3))
     1    /dfloat(nj-1)/8.
        if(jx.gt.nj+np(4) .and. jx.lt.nj+np(7)) xprt =
     1    dfloat(jx-(nj+np(6)))/
     1    dfloat(nj-1)/8. + 1.0
        if(jx.ge.nj+np(7)) xprt = dfloat(jx-np(7))/dfloat(nj-1)
c
        return
        end
c
c
      SUBROUTINE GAUSSQ(ik,aviAN,aviCATH,dwnstr)
c
      implicit double precision (a-h,o-z)
c
      common/evprob/xlamda(3),ai(3),z,diodr(10),ciocr(10)
      common/co/coeff,xn
      common/geom/pi,hol,nj,ndec,icut
      common/elecs/ian,icath
      common/gauss/xp(96),gw(96),ngt,xx(201),xielec(201),xiothr(20
     11),njmore
      common/colloc/npts,icoll(2,41),npextr,np(7)
      common/params/delta1,delta2,delta3
c
      ielec=ian
      iothr = icath
      c = pi/(2.*hol)
      an = 0.5d0
c
        sumrf = 0.d0
        sumrs = 0.d0
        sumrt = 0.d0
```

```
c
        do 170 iregn=1,2
            xll = 0.d0
            xul = 0.5**an
c
          do 170 jxp=1,ngt
            xprm = xp(jxp)*(xul-xll) + xll
            zp = xprm**(1./an)
            if (iregn.eq.2) zp = 1. - zp
c
              aa = curfcn(ik,ian,zp)
              bb = curfcn(ik,icath,zp)
c
            Y2 = xprm**(1./an - 1.) / an
c
            sumrf = sumrf + gw(jxp)*(aa+bb)*(zp-0.5)*Y2
            sumrs = sumrs + gw(jxp)*aa*Y2
            sumrt = sumrt + gw(jxp)*bb*Y2
  170     continue
c
          sumrf = sumrf/2.*(xul-xll)
          sumrs = sumrs/2.*(xul-xll)
          sumrt = sumrt/2.*(xul-xll)
c
      dwnstr = sumrf/(2.*hol)
      avian=sumrs
      avicath=sumrt
c
      return
      end
```

Appendix H. Tables of Experimental Parameters.

Table H-1. Solution concentrations for disk experiments (mol/liter).

Solution	NaOH	KOH	$K_3Fe(CN)_6$	$K_4Fe(CN)_6$
1a[†]	**0.2457**		**0.01000**	**0.05030**
	0.2582		0.01003	0.04983
1b	0.2992		0.01001	0.05003
2	2.1091		0.04993	0.05008
3	0.4015		0.01430	0.01479
4		0.7230	0.00472	0.05020

[†] Boldface refers to solutions that were analyzed for ferricyanide by iodometric titration with zinc and for ferrocyanide by potentiometric titration with ceric sulfate. All other concentrations were calculated by weighing.

Table H-2. Measured parameters for the 5.00-mm disk experiments.

Run	T (°C)	Ω (rpm)	I_{lim} (mA)	O. C. P. (mV)
1a-1		400	0.530	+1
		900	0.800	
		1600	1.066	
		2500	1.330	+2
	19.8	2500	1.316	+2
1a-2	19.4	2500	1.358	-1
		1600	1.088	
		900	0.820	
		400	0.546	
	19.8	400	0.546	-1
1a-3		400	0.549	-1
		900	0.820	
		1600	1.094	
		2500	1.366	
	20.1	2500	1.367	
1a-4	19.8	2500	1.377	-1.5
		1600	1.102	
		900	0.830	
	20.0	400	0.553	
1b-1		400	0.540	3.5
		900	0.807	
		1600	1.074	
	19.6	2500	1.339	
1b-2		2500	1.346	-2
		1600	1.078	
		900	0.811	
	19.35	400	0.545	
1b-3		400	0.540	1
		900	0.813	
		1600	1.080	
	19.6	2500	1.347	
1b-4		2500	1.350	-1.5
		1600	1.080	
		900	0.812	
	—	400	0.541	

Run	T (°C)	Ω (rpm)	I_{lim} (mA)	O. C. P. (mV)
2-1		400	2.02	0
		900	3.02	
		1600	4.02	
		2500	5.07	
		2500	5.07	
	—	2500	5.08	
2-2		2500	5.17	11
		1600	4.13	
		1600	4.13	
		900	3.11	
		400	2.10	
	—	400	2.06	
3-1		400	0.782	3
		900	1.169	
		1600	1.551	
	20.1	2500	1.936	
3-2		2500	1.916	-1
		1600	1.535	
		900	1.156	
	19.4	400	0.769	
3-3		400	0.770	-1
		900	1.152	
		1600	1.530	
	19.8	2500	1.908	
3-4		2500	1.877	-1
		1600	1.503	
		900	1.129	
	19.5	400	0.754	

Run	T (°C)	Ω (rpm)	I_{lim} (mA)	O. C. P. (mV)
4-1	20.1	400	0.263	1
		900	0.394	
		1600	0.522	
		2500	0.650	
4-2	19.9	2500	0.661	—
		1600	0.528	
		900	0.397	
		400	0.265	
4-3	20.1	400	0.264	-1
		900	0.393	
		1600	0.522	
		2500	0.651	
4-4	19.9	2500	0.657	-1
		1600	0.525	
		900	0.394	
		400	0.264	

Table H-3. Calculated parameters for the 5.00-mm disk experiments.*

Run	c_{Ferri} (mol/l)	T (K)	μ (cp)	ρ (g/cc)	ν (cm^2/s)	$D \times 10^6$ (cm^2/s)	Sc	$\mu D/T \times 10^{10}$ (dyne/K)
1a-1	**0.01000**	292.9	1.127512	1.021008	1.1043	6.001	1840	2.31
1a-2		292.7	1.127521		1.1043	6.197	1784	2.39
1a-3		293.2	1.124752		1.1016	6.250	1763	2.40
1a-4		293.0	1.127507		1.1043	6.322	1747	2.43
1b-1	0.01001	292.7	1.136076	1.023245	1.1103	6.061	1832	2.35
1b-2		292.5	1.136089		1.1103	6.109	1818	2.37
1b-3		292.7	1.136076		1.1103	6.116	1815	2.37
1b-4		293.1	1.136056		1.1102	6.136	1809	2.38
2-1	0.049934	293.1	1.568597	1.0949	1.4326	4.209	3404	2.25
2-2						4.399	3257	2.35
3-1	0.014302	293.2	1.104543	1.021	1.0818	6.036	1792	2.27
3-2		292.5	1.107264		1.0845	6.039	1796	2.29
3-3		292.9	1.107251		1.0845	6.002	1807	2.27
3-4		292.6	1.107261		1.0845	5.856	1852	2.21
4-1	0.0047219	293.2	1.101	1.030	1.069	6.268	1705	2.35
4-2		293.0	1.103		1.071	6.431	1665	2.42
4-3		293.2	1.101		1.069	6.282	1702	2.36
4-4		293.0	1.103		1.071	6.372	1681	2.40

* See notes on the following page.

Notes for table H-3.

Viscosities were estimated by first using Boeffard's equation to obtain the viscosity at 25 °C:

$$\mu(\text{cps}) = 0.96714 + 0.09622c_{\text{NaOH}} - 0.20528c_{\text{Ferri}} + 0.090255c_{\text{Ferro}} \quad \text{(H-1)}$$
$$+ 0.05404c_{NaOH}^2 + 0.53303c_{Ferri}^2 + 0.43505c_{Ferro}^2$$
$$+0.23546c_{NaOH}c_{Ferri} + 0.302585c_{NaOH}c_{Ferro} + 0.99923c_{Ferro}c_{Ferri}$$

and then using modified equations from the CRC Handbook[112] to account for the temperature dependence. The original equations (for pure water) are

$$\log_{10}[\mu_T(\text{cp})] = \frac{1301}{998.333 + 8.1855(\text{T}-20) + 0.00585(\text{T}-20)^2} - 3.330233 \quad \text{(H-2)}$$
$$(0 - 20\,^\circ\text{C})$$

$$\log_{10}\frac{\mu_T}{\mu_{20}} = \frac{1.3272(20-T) - 0.00153(T-20)^2}{T + 105} \quad \text{(H-3)}$$
$$(20 - 100\,^\circ\text{C}).$$

These equations are based on $\mu(20\,^\circ)$=1.002 cp. Equation H-2 was modified by substituting μ_{25} into equation H-3 to obtain μ_{20} for the ferricyanide solution and adjusting the final constant (3.330233) to give the proper μ_{20}.

The densities were estimated from Boeffard's correlation:

$$\rho(\text{g/cm}^3) = 0.99702 + 0.04423c_{\text{NaOH}} + 0.17118c_{\text{Ferri}} + 0.23119c_{\text{Ferro}} \quad \text{(H-4)}$$
$$- 0.00133c_{NaOH}^2 - 0.00787c_{NaOH} - 0.00978c_{NaOH}c_{Ferro},$$

where the concentrations are in mol/l.

Table H-4. Solution concentrations for channel experiments (mol/liter).

Solution	NaOH	$K_3Fe(CN)_6$	$K_4Fe(CN)_6$
1c	0.3190	0.01001	0.05009
1d	0.3415	0.01001	0.05109
1e†	**0.2832**	**0.01012**	**0.05758**
	0.2996	0.01002	0.050093
1f	0.301145	0.01004	0.05042
2	2.1091	0.04993	0.05008

† Boldface refers to solutions that were analyzed for ferricyanide by iodometric titration with zinc and for ferrocyanide by potentiometric titration with ceric sulfate. All other concentrations were calculated by weighing.

H-5. Measured parameters for the channel experiments

(L = 2.370 in, W = 2.007 in).

Run	h (in)	Q (ml/min)	O.C.P. (mV)	$V-V_{ref}$ (mV)	I (mA)
1c-1	0.127	4.97	+2.0	-800	-4.09
				-600	-4.10
				-400	-4.11
				-200	-4.09
1c-2		4.91	+1.7	-100	-4.16
				-200	-4.27
				-600	-4.25
1c-3		4.98	+1.0	-600	-4.26
				-200	-4.27
				-100	-4.17
1c-4		4.98	+1.1	-50	-3.55
				-20	-2.21
				-10	-1.40
				0	-0.3
1c-5	0.127	—	+1	-600	-4.74
				-200	-4.71
				-200	-4.70
				-100	-4.59
1c-6		6.90	+0.3	-50.1	-3.87
				-20	-2.29
				-9.9	-1.32
1c-7		6.80	0.0	-600	-4.70
				-800	-4.70
				-1000	-4.76
				-1200	-4.91
1c-8		—	-0.4	-35	-3.36

Run	h (in)	Q (ml/min)	O.C.P. (mV)	$V-V_{ref}$ (mV)	I (mA)
1d-1	0.127	3.3	+1.0	-600	-3.75
				-400	-3.76
				-200	-3.73
				-100	-3.66
				-50	-3.07
1d-2		3.35	+0.8	-20	-1.87
				0	-0.4
				-600	-3.73
1d-3		3.69	+0.0	-800	-3.68
				-1000	-3.73
				-1200	-3.87
1d-4		3.23	—	-600	-3.66
				-35	-2.62
				-9.9	-1.16
1d-5		3.21	+3.5	-10	-1.19
				-35	-2.59
1d-6		3.23	-3.0	-1200	-3.71
				-800	-3.63
1e-1	0.0178	16.71	+2.0	-400	-22.7
				-600	-22.9
				-800	-23.1
				-1000	-23.1
1e-2		17.02	+1.0	-200	-23.7
				-100	-22.9
				-200	-23.6
				-50	-19.1
				-35	-16.1
				-20	-11.4
				-10.1	-6.75
1e-3	0.0178	16.85	—	-400	-23.5
				-600	-23.7
				-800	-23.7
				-1000	-23.6
				-1200	-24.0
				-1400	-48.2
				-400	-23.1

Run	h (in)	Q (ml/min)	O.C.P. (mV)	$V-V_{ref}$ (mV)	I (mA)
1e-4		16.77	-1.0	-400	-22.7
				-600	-22.7
				-400	-22.7
				-800	-22.8
				-1100	-22.5
1f-1	0.0178	0.21	+2.7	-20	-4.8
				-10	-3.45
				-5	-2.8
				-35	-6.28
				-50	-7.08
				-65	-7.55
				-79.9	-7.82
				-95	-7.96
				-110	-8.05
				-130	-8.11
				-180	-8.17
1f-2		0.4	+2.7	-180	-8.65
				-160	-8.70
				-120	-8.6
				-90	-8.40
				-70	-8.11
				-50	-7.45
				-35	-6.55
				-20	-5.05
				-10	-3.63
				-180	-8.75
				-230	-8.75
				-300	-8.76
				-400	-8.82
				-480	-8.85
				-600	-8.88
				-800	-8.85
				-900	-8.9
				-1000	-9.00
				-1200	-9.3
				-1300	-10.65

Run	h (in)	Q (ml/min)	O.C.P. (mV)	$V-V_{ref}$ (mV)	I (mA)
2-1	0.127	4.94	+1	-600	-16.6
				-400	-16.5
				-200	-16.6
				-100	-15.7
				-50	-11.4
2-2		5.04	+1	-35	-10.2
				-20	-6.73
				-10	-4.06
2-3		4.94	-6	-600	-17.0
2-4		7.01	—	-600	-18.8
2-5		3.57	—	-600	-15.3
2-6		5.05	-2	-600	-17.0
				-800	-16.9
				-1000	-16.9
				-1200	-17.1
				-1400	-17.5
2-7		7.0	—	-600	-18.8

Table II-6. Calculated dimensional parameters for the channel experiments.*

Run	c_{Ferri} (mol/l)	c_{Ferro} (mol/l)	$\mu(20^{\circ}C)$ (cp)	ρ (g/cc)	ν (cp)	$(\Omega^{-1}cm^{-1})$	<v> (cm/s)	I_{lim} (mA)	$D \times 10^{6}$ (cm^2/s)	$\mu D/T \times 10^{10}$ (dyne/K)
1c	0.01001	0.05009	1.13935	1.024107	1.11253	0.1212	0.050270	4.2	5.93	2.31
							0.069426	4.7	5.93	2.31
1d	0.01001	0.05109	1.14340	1.025297	1.11519	0.1275	0.033345	3.7	5.94	2.32
1e	0.01012	0.05758	1.13532	1.024301	1.10838	0.1179	1.2177	23.2	5.91	2.29
1f	0.01004	0.05042	1.13647	1.023422	1.11046	0.1171	0.015186	8.2	5.93	2.30
							0.028925	8.8	5.93	2.30
2-1	0.04993	0.05008	1.568597	1.0949	1.43264	0.5855	0.050068	16.6	4.32	2.31
2-2							0.051081	—		
2-3							0.050068	17.0		
2-4							0.071047	18.8		
2-5							0.36182	15.3		

† Boldface refers to solutions that were analyzed for ferricyanide by iodometric titration with zinc and for ferrocyanide by potentiometric titration with ceric sulfate. All other concentrations were calculated by weighing.

Table B-7. Calculated dimensionless parameters for the channel experiments.

Run	c_{Ferro}/c_{Ferri}	J	N	δ	Sc	Re	$\frac{h}{L}$	$Pe\,\frac{d_e}{L}$	Nu_{Lev}	Nu_{exp}
1c	5.00244	400.86	0.32883	0.26539	1876	2.915	0.053763	586.15	15.472	15.417
			0.36619	0.29642	1876	4.026		809.50	17.230	17.252
1d	5.10168		0.27291	0.21911	1877	1.929		388.14	13.486	13.559
1e	5.68972		1.8993	1.5230	1875	9.935	0.0075353	279.87	12.093	11.846
1f	5.01899		0.44094	0.54131	1873	0.1237		3.4783	2.8012	4.1907
			0.54659	0.56297	1873	0.2355		6.6253	3.4724	4.5138
2-1	1.00300	450.0	0.27452	—	2646	2.8254	0.053763	801.36	17.172	16.769
2-2			0.27636			2.8826		817.58	17.288	—
2-3			0.27452			2.8254		801.36	17.172	17.173
2-4			0.30849			4.0093		1137.1	19.297	18.991
2-5			0.24635			2.0418		579.12	15.410	15.455

References

1. Manuel M. Baizer and Henning Lund, eds., *Organic Electrochemistry: An Introduction and a Guide*, Second Edition, Marcel Dekker, Inc., New York, 1983.

2. *Kirk-Othmer Encyclopedia of Chemical Technology, Volume 8*, Third edition, Herman Mark *et al.* eds., Wiley, New York (1979).

3. J. H. Prescott, "Monsanto's Unique Process Brings Electrochemistry to Organics," *Chemical Engineering*, (Nov. 8, 1965), 238-242.

4. R. M. Hurd, "Adiponitrile Made Electrochemically," *Hydrocarbon Processing*, **43**, No. 11 (1964), 154-156.

5. Charles Russell Campbell, Donald E. Danly, and Werner Heinrich Mueller (to Monsanto Co.), U.S. Pat. 3,830,712 (Aug. 20, 1974).

6. William A. Heckle and Donald L. Sadler (to Monsanto Co.), U.S. Pat. 3,897,318 (July 29, 1975).

7. John Harvey Lester, Jr. and James S. Stewart (to Monsanto Co.), U.S. Pat. 3,898,140 (Aug. 5, 1975).

8. D. E. Danly and R. W. McWhorter (to Monsanto Co.), Belg. Pat. 699,284 (May 31, 1967).

9. J. L. Fitzjohn, "Electro-organic Synthesis," *Chemical Engineering Progress*, **71** (1975), 85-91.

10. Johnsee Lee, J. R. Selman, "Effects of Separator and Terminal on the Current Distribution in Parallel-Plate Electrochemical Flow Reactors," *Journal of the Electrochemical Society*, **129** (1982), 1670-1678.

11. J. A. M. Le Duc (to Pullman, Inc.), U.S. Pat. 3,342,717 (Sept. 19, 1967); Chem. Abstr., **67** (1967), 7528v.

12. W. Kroenig and J. Grolig (to Farbenfabriken Bayer A.-G.), U.S. Pat. 3,451,905

(1969).

13. Richard C. Alkire and James D. Lisius, " Incorporation of Complex Reaction Sequences in Engineering Models of Electrolytic Cells. I. Paired Synthesis of Propylene Oxide in an Undivided Cell," *Journal of the Electrochemical Society,* **132** (1985), 1879-1888.

14. M. Fleischmann, R. E. W. Jansson, G. A. Ashworth, and P. J. Ayre, Brit. Prov. Pat. No. 18305/74.

15. R. E. W. Jansson and Martin Fleischmann, "Effect of Cell Design on Selectivity and Conversion in Electroorganic Processes," *AIChE Symposium Series No. 185,* **75**, (1979), 2-7.

16. F. Beck, " Electrosynthesis of Adiponitrile in Undivided Cells," *Journal of Applied Electrochemistry,* **2** (1972), 59-69.

17. K. Park, P. N. Pintauro, M. M. Baizer, and K. Nobe, "Flow Reactor Studies of the Paired Electro-Oxidation and Electroreduction of Glucose," *Journal of the Electrochemical Society,* **132** (1985), 1850-1855.

18. R. E. White, Mike Bain, and Mike Raible, "Parallel Plate Electrochemical Reactor Model," *Journal of the Electrochemical Society,* **130** (1983), 1037-1042.

19. T. V. Nguyen, C. W. Walton, R. E. White, J. Van Zee, "Parallel-Plate Electrochemical Reactor Model: A Method for Determining the Time-Dependent Behavior and the Effects of Axial Diffusion and Axial Migration," *Journal of the Electrochemical Society,* **133** (1986), 81-87.

20. W. R. Parrish and John Newman, "Current Distribution on a Plane Electrode below the Limiting Current," *Journal of the Electrochemical Society,* **116** (1969), 169-172.

21. W. R. Parrish and John Newman, "Current Distributions on Plane, Parallel Electrodes in Channel Flow," *Journal of the Electrochemical Society,* **117** (1970), 43-48.

22. Reinaldo Cabán and Thomas W. Chapman, "Rapid Computation of Current Distribution by Orthogonal Collocation," *Journal of the Electrochemical Society,* **123** (1976), 1036-1041.

23. George P. Sakellaropoulos, "Criteria for Selective Path Promotion in Electrochemical Reaction Sequences," *AIChE Journal,* **25** (1979), 781-793.

24. George P. Sakellaropoulos and Gary A. Francis, "Electrochemical Reactor Analysis: Selectivity of Multiple Competing Reactions," *Journal of the Electrochemical Society,* **126** (1979), 1928-1937.

25. George P. Sakellaropoulos and Gary A. Francis, "Selectivity and Reactor Analysis for Parallel Electrochemical Reactions," *Journal of Chemical Technology and Biotechnology,* **30** (1980), 102-109.

26. D. J. Pickett, *Electrochemical Reactor Design,* Elsevier, Amsterdam (1977).

27. D. Bürgi, P. Noorlander, and H. Siegenthaler, "Steady-State Mass Transfer in Radial Flow-Through Thin Layer Cells — I. Reversible Metal Systems," *Electrochimica Acta,* **26** (1981), 1289-1297.

28. John Newman, *Electrochemical Systems,* Prentice-Hall, Englewood Cliffs, New Jersey (1973).

29. R. E. Acosta, R. H. Muller, and C. W. Tobias, "Transport Processes in Narrow (Capillary) Channels," *AIChE Journal,* **31** (1985), 473-482.

30. John Newman, "The Fundamental Principles of Current Distribution and Mass Transport in Electrochemical Cells," *Electroanalytical Chemistry, Volume 6,* Allen J. Bard, ed., Marcel Dekker, New York (1973).

31. M. A. Lévêque, "Les Lois de la Transmission de Chaleur par Convection," *Annales des Mines, Memoires,* ser. 12, **13**, 201-299, 305-362, 381-415 (1928).

32. T. K. Ross and A. A. Wragg, "Electrochemical Mass Transfer Studies in Annuli," *Electrochimica Acta,* **10** (1965), 1093-1106.

33. W. M. Kays and M. E. Crawford, *Convective Heat and Mass Transfer* (Second Edition), McGraw Hill, New York (1980), pp. 88-132.

34. R. Alkire and P. K. Ng, "Two-Dimensional Current Distribution within a Packed-Bed Electrochemical Flow Reactor," *Journal of the Electrochemical Society,* **121** (1974), 95-103.

35. R. Alkire and P. K. Ng, "Studies on Flow-By Porous Electrodes Having Perpendicular Directions of Current and Electrolyte Flow," *Journal of the Electrochemical Society,* **124** (1977), 1220-1227.

36. Peter S. Fedkiw, "Ohmic Potential Drop in Flow-Through and Flow-By Porous Electrodes," *Journal of the Electrochemical Society,* **128** (1981), 831-838.

37. Timothy Risch and John Newman, "A Theoretical Comparison of Flow-Through and Flow-By Porous Electrodes at the Limiting Current," *Journal of the Electrochemical Society,* **131** (1984), 2551-2556.

38. M. J. Mader, C. W. Walton, and R. E. White, "Parallel Plate Electrochemical Reactor Model: Material Balance Closure and a Simplification," *Journal of the Electrochemical Society,* **133** (1986), 1124-1130.

39. T. V. Nguyen, C. W. Walton, and R. E. White, "A Mathematical Model for a Parallel Plate Electrochemical Reactor, CSTR, and Associated Recirculation System, *Journal of the Electrochemical Society,* **133** (1986), 1130-1138.

40. F. Beck and H. Guthke, "Entwicklung Neuer Zellen für Elektro-organische Synthesen," *Chemie Ing. Techn.*, **41** (1969), 943-950.

41. F. Beck and co-workers (to Badische Anilin- und Soda-Fabrik A.G.), U.S. Pat. 3,652,430 (March 28, 1972).

42. J. Nohe and F. Beck (to Badische Anilin- und Soda-Fabrik A.G.), U.S. Pat. 3,899,401 (Aug. 12, 1975).

43. Franz Wenisch, Heinz Nohe, Heinz Hannebaum, R. K. Horn, Manfred Stroezel,

and Dieter Degner, "Experiences with an Undivided Cell," *AIChE Symposium Series No. 185,* **75**, (1979), 14-18.

44. P. M. Robertson, P. Berg, H. Reimann, K. Schleich, and P. Seiler, "Application of the 'Swiss-Roll' Electrolysis Cell in Vitamin-C Production," *Journal of the Electrochemical Society,* **130** (1983), 591-596.

45. R. E. W. Jansson and R. J. Marshall, "Bipolar Electrochemical Pump Cell," *The Chemical Engineer* (1976), 769-772.

46. R. E. W. Jansson and G. A. Ashworth, "The Continuous Deposition of Metal Powders in a Pump Cell," *Journal of Applied Electrochemistry,* **7** (1977), 309-314.

47. John Newman, "Engineering Design of Electrochemical Systems," *Industrial and Engineering Chemistry,* **60**, No. 4 (April, 1968), 12-27.

48. Francis B. Hildebrand, *Advanced Calculus for Applications,* Second Edition, pp. 463-466, Prentice-Hall, Englewood Cliffs, New Jersey (1976).

49. C. Ray Wylie, *Advanced Engineering Mathematics,* Fourth Edition, pp. 309-319, McGraw Hill, New York (1960).

50. H. S. Carslaw and J. C. Jaeger, *Conduction of Heat in Solids,* (Second Edition), Oxford University Press, London (1959).

51. Victoria Edwards and John Newman, "The Asymmetric Graetz Problem in Channel Flow," *International Journal of Heat and Mass Transfer,* **28** (1985), 503-505.

52. Ralph Edward White, *Simultaneous Reactions on a Rotating Disk Electrode,* dissertation, University of California, Berkeley (March, 1977).

53. M. Jakob, *Heat Transfer,* Wiley, New York, Volume I (1949).

54. William H. McAdams, *Heat Transmission,* second edition, McGraw-Hill, New York (1942).

55. L. Graetz, "Über die Wärmeleitungsfähigkeit von Flüssigkeiten," *Annalen der Physik und Chemie,* **18**, 79-94 (1883), **25**, 337-357 (1885).

56. Thomas Bradford Drew, "Heat Transfer in Stream-Line Flow — II. Experiments with Glycerine," *Trans. AIChE*, **27** (1931), 171-189.

57. John Newman, "Numerical Solution of Coupled, Ordinary Differential Equations," *Industrial and Engineering Chemistry Fundamentals*, **7** (August 1968), 514-517.

58. Ralph White and John Newman, "Simultaneous Reactions on a Rotating-Disk Electrode," *Journal of Electroanalytical Chemistry*, **82** (1977) 173-186.

59. Andreas Acrivos and Paul L. Chambré, "Laminar Boundary Layer Flows with Surface Reactions," *Industrial and Engineering Chemistry*, **49** No. 6 (June, 1957), 1025-1029.

60. Ralph White, Charles M. Mohr, Peter Fedkiw, and John Newman, "The Fluid Motion Generated by a Rotating Disk: A Comparison of Solution Techniques," LBL-3910, Lawrence Berkeley Laboratory, November, 1975.

61. W. Allen Smith, *Elementary Numerical Analysis*, Harper and Row, New York, 1979.

62. J. F. Traub, *Iterative Methods for the Solution of Equations*, Prentice-Hall, Englewood Cliffs, New Jersey, 1964.

63. Limin Hsueh, *Diffusion and Migration in Electrochemical Systems*, dissertation, University of California, Berkeley (December, 1968).

64. L. V. Kantorovich and V. J. Krylov, "Approximate Methods of Higher Analysis," translated by C.D. Benster, Interscience Publishers, Inc., New York (1959), p. 101.

65. John Newman, "Extension of the Lévêque Solution," *Transactions of the ASME, Part C, Journal of Heat Transfer*, **91** (February, 1969), 177-178.

66. George Martin Brown, "Heat or Mass Transfer in a Fluid in Laminar Flow in a Circular or Flat Conduit," *AIChE Journal*, **6** (June, 1960), 179-183.

67. J. R. Sellars, Myron Tribus, and J. S. Klein, "Heat Transfer to Laminar Flow

in a Round Tube or Flat Conduit - The Graetz Problem Extended," *Transactions of the ASME*, **78** (February, 1956), 441-448.

68. R. H. Norris and D. D. Streid, "Laminar-Flow Heat-Transfer Coefficients for Ducts," *Transactions of the ASME* (August, 1940), 525-533.

69. C. S. Yih and Jack E. Cermak, "Laminar Heat Convection in Pipes and Ducts," Civil Engineering Department, Colorado Agricultural and Mechanical College, Fort Collins, Colorado (September, 1951). ONR Contract Number N90nr-82401, NR063-O71/1-19-49.

70. J. Schenk and H. L. Beckers, "Heat Transfer in Laminar Flow between Parallel Plates," *Applied Scientific Research*, **A4** (1954), 405-413.

71. A. P. Hatton and J. S. Turton, "Heat Transfer in the Thermal Entry Length with Laminar Flow between Parallel Walls at Unequal Temperatures," *International Journal of Heat and Mass Transfer*, **5** (1962), 673-679.

72. R. D. Cess and E. C. Shaffer, "Laminar Heat Transfer between Parallel Plates with an Unsymmetrically Prescribed Heat Flux at the Walls," *Applied Scientific Research*, **A9** (1959), 64-70.

73. Carl Wagner,"Theoretical Analysis of the Current Density Distribution in Electrolytic Cells," *Journal of the Electrochemical Society*, **98** (1951), 116-128.

74. Peter Pierini, Peter Appel, and John Newman, "Current Distribution on a Disk Electrode for Redox Reactions," *Journal of the Electrochemical Society*, **123** (1976), 366-369.

75. Peter Pierini and John Newman, "Current Distribution on a Rotating Ring-Disk Electrode below the Limiting Current," *Journal of the Electrochemical Society*, **124** (1977), 701-706.

76. Raul Edmundo Acosta, *Transport Processes in High Rate Electrolysis*, dissertation, University of California, Berkeley (May, 1974), LBL-2242.

77. Daniel Seth Fischl, *Effects of Small Flow Obstacles on the Limiting Current and Pressure Drop in a Square Duct,* M. S. Thesis, University of California, Berkeley (August, 1983).

78. Stanley. L. Gordon, John. S. Newman, and Charles. W. Tobias, "The Role of Ionic Migration in Electrolytic Mass Transport; Diffusivities of $[Fe(CN)_6]^{3-}$ and $[Fe(CN)_6]^{4-}$ in KOH and NaOH Solutions," *Berichte der Bunsengesellschaft für Physikalische Chemie,* **70** (1966), 414-420.

79. Morris Eisenberg, *Studies of Rates of Solid Dissolution and of Electrode Reactions at Rotating Cylindrical Bodies,* dissertation, University of California, Berkeley (October, 1953).

80. M. Eisenberg, C. W. Tobias, and C. R. Wilke, *J. Electrochem. Soc.*, **101** (1954), 306-319.

81. O. Essin, S. Derendiayev, and N. Ladygin, *J. Applied Chem. (USSR),* **13** (1940), 971.

82. J. V. Petrocelli and A. A. Paolucci, "Overvoltage at Oxidation-Reduction Electrodes," *J. Electrochem. Soc.,* **98** (1951), 291-295.

83. L. Philip Reiss and Thomas J. Hanratty, "An Experimental Study of the Unsteady Nature of the Viscous Sublayer," *AIChE Journal,* **9** (1963), 154-160.

84. Jaime S. Son and Thomas J. Hanratty, "Limiting Relation for the Eddy Diffusivity Close to a Wall," *AIChE Journal,* **13** (1967), 689-696.

85. Alain Jean-Louis Pierre Marie Boeffard, *Ionic Mass Transport by Free Convection in a Redox System,* M. S. Thesis, University of California, Berkeley (1966), UCRL-16624.

86. Charles Milton Mohr, Jr., *Mass Transfer in Rotating Electrode Systems,* dissertation, University of California, Berkeley (October, 1975), LBL-3913.

87. Jan Robert Selman, *Measurement and Interpretation of Limiting Currents,*

dissertation, University of California, Berkeley (June 1971), UCRL-20557.

88. Ping Sih, *Mass Transfer to the Rear of an Object at Low Reynolds Number Flow,* dissertation, University of California, Berkeley (May 1971), UCRL-20509.

89. Harry Hung-Kwan Yip, *Mass Transfer Coefficient in Packed Beds at Low Reynolds Numbers,* M. S. Thesis, University of California, Berkeley (June 1973), LBL-1831.

90. Peter Willem Appel, *Electrochemical Systems: Impedance of a Rotating Disk and Mass Transfer in Packed Beds,* dissertation, University of California, Berkeley (May, 1976).

91. Peter Eugene Pierini, *A Study of Ring and Ring-Disk Electrodes,* dissertation, University of California, Berkeley (May, 1981), LBL-12776.

92. *The Merck Index — An Encyclopedia of Chemicals, Drugs, and Biologicals, Tenth Edition,* Merck & Co., Rahway, New Jersey (1983).

93. B. Levich, "The Theory of Concentration Polarization," *Acta Physicochimica U.R.S.S.,* **17** (1942), 257-307.

94. Michael John Matlosz, *Experimental Methods and Software Tools for the Analysis of Electrochemical Systems,* dissertat ion, University of California, Berkeley (March, 1985).

95. A. J. Arvía, S. L. Marchiano, and J. J. Podestá, "The Diffusion of Ferrocyanide and Ferricyanide Ions in Aqueous Solutions of Potassium Hydroxide," *Electrochimica Acta,* **12** (1967), 259-266.

96. William H. Smyrl and John Newman, "Ring-Disk and Sectioned Disk Electrodes," *Journal of the Electrochemical Society,* **119** (1972), 212-219.

97. C. S. Lin, R. W. Moulton, G. L. Putnam, "Mass Transfer between Solid Wall and Fluid Streams — Mechanism and Eddy Distribution Relationships in Turbulent Flow," *Ind. Eng. Chem.,* **45** (1953), 636-645.

98. Klaus J. Vetter, *Electrochemical Kinetics — Theoretical and Experimental*

Aspects, Academic Press, New York (1967).

99. Milton Abramowitz and Irene A. Stegun, eds., *Handbook of Mathematical Functions,* Dover Publications, New York (1972).

100. J. M. Ortega and W. C. Rheinboldt, *Iterative Solution of Nonlinear Equations in Several Variables,* Academic Press, New York, 1970.

101. T. A. Jeeves, "Secant Modification of Newton's Method," *Comm ACM,* **1** (8), (1958), 9-10.

102. John Newman, "Current Distribution on a Rotating Disk below the Limiting Current," *Journal of the Electrochemical Society,* **113** (1966), 1235-1241.

103. Bruce A. Finlayson, *Nonlinear Analysis in Chemical Engineering,* McGraw Hill, New York, 1980.

104. John Villadsen and Michael L. Michelsen, *Solution of Differential Equation Models by Polynomial Approximation,* Prentice-Hall, Englewood Cliffs, New Jersey (1978).

105. Warren E. Stewart, "Solution of Transport Problems by Collocation Methods," *AIChE Continuing Education Series, Lectures in Transport Phenomena* (1969), 114-135.

106. S. J. Rhee and D. K. Edwards, "Laminar Entrance Flow in a Flat Plate Duct with Asymmetric Suction and Heating," *Numerical Heat Transfer,* **4** (1981), 85-100.

107. Joseph John Miksis, Jr., *Primary Resistances for Ring-Disk Electrodes,* M.S. thesis, University of California, Berkeley (November, 1975), LBL-4537.

108. Joseph J. Miksis, Jr. and John Newman, "Primary Resistances for Ring-Disk Electrodes," *Journal of the Electrochemical Society,* **123** (1976), 1030-1036.

109. Germund Dahlquist, Åke Björck, *Numerical Methods,* Prentice-Hall, Englewood Cliffs, New Jersey, 1974.

110. Professor David Graves, personal communication (April, 1986)

111. Professor John Newman, personal communication (June, 1986).

112. *CRC Handbook of Chemistry and Physics, 58th Edition,* Robert C. Weast, editor, CRC Press, West Palm Beach (1977-1978).

LAWRENCE BERKELEY LABORATORY
TECHNICAL INFORMATION DEPARTMENT
UNIVERSITY OF CALIFORNIA
BERKELEY, CALIFORNIA 94720