

CREVICER Simulator

- Installation and Getting Started

Hongwei Wang
Francisco J. Presuel-Moreno
and Robert G. Kelly



The Center for Electrochemical Science and Engineering
Department of Materials Science and Engineering
University of Virginia
July 1, 2002
This documentation was written for CREVICER version 1.0

Contents

Preface

Part I: Installation

1 Want to have a free crevice corrosion simulator?

- 1.1 To whom this guide is addressed and how it is organized
- 1.2 Yet another crevice corrosion simulator?
- 1.3 Copyright claim
- 1.4 System requirements
- 1.5 Which version should I use?
- 1.6 Crevice models

2 Building the crevice corrosion system: Compiling the program

- 2.1 Getting a development environment under Windows
- 2.2 Download the CREVICER source code
- 2.3 Compiling CREVICER under Windows
- 2.4 Running the CREVICERGUI or CREVICERSOLVER

3 Installing CREVICER

- 3.1 Installing the binary distribution on a Windows system
- 3.2 Installing documentation

Part II: Playing with CREVICER

4 Crevice corrosion parameters and menus

- 4.1 Starting the GUI
- 4.2 Implementation of functions of GUI
- 4.3 Case study

5 Future improvements to the CREVICER

- 5.1 Kevin's wish (1994-1999)
- 5.2 DeJong's wish (1997-1999)
- 5.3 Lee's wish (1999-2001)
- 5.4 Wang' wish for new developments (2001-2003)

6 Reference

Part III: Appendices

A Missed approach: If anything refuses to work

- A.1 CREVICER problem reports
- A.2 General problems
- A.3 Potential problems under Windows

B Finishing: Some further thoughts before leaving the crevice

- B.1 A not so short history of CREVICER
- B.2 Those, who did the work

B.3 What remains to be done

B.4 Acknowledgements

C GNU General Public License

C.1 GNU general public license

C.2 How to apply these terms to your new programs

Preface

CREVICER is a free crevice corrosion simulator developed initially by Professor R. G. Kelly at University of Virginia. It was released on July 1, 2002 with the aim of developing improved versions through collaboration over the Internet. This “Installation and Getting Started” is meant to be a guide for beginners in getting CREVICER up and running. It is not intended to provide complete documentation of all the features and add-ons of CREVICER but, instead, focuses on those aspects necessary to get into the CREVICER.

This guide is split into three parts. The first part describes how to install the program, the second part details on how to actually play with CREVICER while the third part records the background information. The chapters concentrate on the following aspects:

Part I: Installation

Chapter 1, Want to have a free corrosion testing? Take CREVICER, introduces the concept, describes the system requirements, and classifies the different versions available.

Chapter 2, Building the crevice: Compiling the program, explains how to build (compile and link) the simulator. Depending on your platform this may or may not be required. Generally, there will be executable programs (binaries) available for several platforms. Those on such systems who want to use immediately, without going through the potentially troublesome process of compiling, may skip this chapter.

Chapter 3, Installing CREVICER, you will find instructions for installing the binaries in case you did not build them yourself as specified in the previous chapter. You will need to other support files collected in the base package.

Part II: Playing with CREVICER

Chapter 4, Crevice corrosion parameters and menus, describes how to operate the program, i.e., how to actually play with CREVICER shown by a case study.

Chapter 5, Future improvements to the CREVICER, discusses the future improvements to the CREVICER.

Chapter 6, Reference, you can find major publication or links used for the writing of this manual.

Part III: Appendices

In Appendix A, Missed approach: If anything refuses to work, we will try to help you work through some common problems faced when using CREVICER.

In the Appendix B, Finishing: Some further thoughts before leaving the crevice, we would like to give credit to those who deserve it, sketch an overview on the development of CREVICER, and point out what remains to be done.

In the Appendix C, GNU general public license, something can be learned from the usage of this open source CREVICER software and how to implement its philosophy into your new programs.

Accordingly, we suggest reading the chapters as follows:

Installation	
Users of binary distributions (under Windows):	1, 3
Operation	
Program start (all users):	4
Troubleshooting	
General issues:	A
Optionally (for developer)	2, B, C, 5

While this introductory guide is meant to be self-contained, we strongly suggest having a look into further documentation at

<http://www.virginia.edu/%7Ecese/research/crevicer/designdocumentation.html>

Finally, we know, most people hate reading manuals. If you are using one of the following operating systems:

Windows 95/98/ME/NT/2000/XP

you can possibly skip at least Part I of this manual and exploit the pre-compiled binaries. These as well as instructions on how to set them up, can be found at Chapter 4.

Note: There is no guarantee for this approach to work. If it doesn't, don't give up! Have a closer look through this guide notably appendix A.

Part I
Installation

Chapter 1

Want to have a free crevice corrosion testing?

1.1 To whom this guide is addressed and how it is organized

There is little, if any, material in this guide that is presented here exclusively. You could even say with Montaigne that we “merely gathered here a big bunch of other men’s flowers, having furnished nothing of my own but the strip to hold them together.” However, a neatly printed manual is arguably preferable over loosely scattered readme files by some people, and those people may acknowledge the effort.

This CREVICER Simulator – Installation and Getting Started is intended to be a first step towards a more complete CREVICER documentation (with the other parts, hopefully, to be written by others). The target audience is the end-user who is not familiar to the CREVICER in general. It is our hope, that someday there will be an accompanying CREVICER Programmer’s Guide.

We kindly ask you to help us refine this document by submitting corrections, improvements, and more. We will be more than happy to include those into future versions of this manual, of course not without giving credit to the authors.

While we intend to continuously update this document at least for the foreseeable future, supposedly we will not be able to produce a new one for any single release of CREVICER. While we are watching the mailing lists, it would help if developers adding new functionality would send us a short note.

1.2 Yet another Crevice Corrosion Simulator?

Did you ever want to look at the inside of crevice corrosion and see the current, potential, and chemical distributions? Do you want to compare the numerical predictions with the your experimental results of crevice corrosion in the laboratory? Do you want to learn about how to implement finite element method into the corrosion computational modeling? Or do you just want to have fun with the C⁺⁺ code for corrosion science and learn the Object Oriented Design (OOD)? If any of these questions applies, this CREVICER simulator is just for you.

CREVICER aims to be an open, user-supported, user-extensible platform.

Open: The project is not restricted to a given cadre of developers. Anyone who feels he or she is able to contribute is most welcome. The code (including documentation) is copyrighted under the terms of the GPL (GNU Public License).

The GPL is often misunderstood. In simple terms it states that you can copy and freely distribute the program(s) so licensed. You can modify them if you like. You are even allowed to charge as much money for the distribution of the modified or original program as you want. However, you must distribute it complete with the entire source code and it must retain the original copyrights. In short:

“You can do anything with the software except make it non-free.”

The full text of the GNU Public License (GPL) can be obtained from <http://www.gnu.org/copyleft/gpl.html>.

CREVICER is user accessible and documented from the beginning. It is our goal to build a basic engine to which new functionalities, e.g. corrosion inhibitor release from coating, cladding cathodic protection, and etc. can be added.

1.3 Copyright claim

CREVICER 1.0 Copyright (C) 2002 Robert G. Kelly, University of Virginia

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License Version 2 as published by the Free Software Foundation.

This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License at the Appendix C; or, if not, write to the Free Software Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA 02111-1307 USA.

Contact information:

Robert G. Kelly

Associate Professor

Center for Electrochemical Science and Engineering

Department of Materials Science and Engineering

University of Virginia

116 Engineer's Way

Charlottesville, VA 22904

Phone: 434.982.5783

Fax: 434.982.5799

Email: rgkelly@virginia.edu

Website: <http://www.virginia.edu/%7Eecese/people/kelly.html>

1.4 System requirements

The system requirements for CREVICER are not extravagant. A decent PIII/800 or something in that range will be sufficient; given you have a proper computational speed. On the other hand, any modern UNIX-type workstation will handle CREVICER as well.

To install the executable code and support files, you will need around 2.0 MB of free disk space. In case you want/have to compile the program yourself you will need additional about 20 MB for the source code and for temporary files created during compilation. This does not yet include the development environment, which possibly may have to be installed under Windows yet, and which amounts to additional around 150 MB, depending on the installed packages.

1.5 Which version should I use?

When we decided to open the CREVICER source code in 2002, a new version numbering was applied starting from version CREVICER 1.0. If you happen to read the CREVICER 1.0 and 2.0 mentioned in these dissertation and theses by Kevin Stewart, Lisa DeJong, and Jason Lee, please be alert to the difference and these are the old version numbering before this official release version.

This version has included most of the work from 1994 to 2001, including the work by Kevin Stewart, Lisa DeJong, Jason Lee, and Changqing Lin. Francisco J. Presuel-Moreno added the gap size in the GUI in 2002, which was also included in this release.

Concerning the CREVICER source code there exist two branches, a stable one and a developmental branch. Even version numbers like 1.6, 1.8, and 1.0 refer to stable releases, while odd numbers like 1.7, 1.9, and so on refer to developmental releases. The policy is to only do bug fixes in the even versions, while new features are generally added to odd-numbered versions which, after all things have stabilized, will become the next stable release with a version number calculated by adding 0.1.

Note that the stable version is usually somewhat outdated, in that it does not reflect the state of development CREVICER has reached. Given that the recent developmental versions on the other hands may contain bugs (e.g. undocumented features), we recommend using the “latest official (stable) release” for the average user. This is the latest version named at

<http://www.virginia.edu/%7Ecese/research/crevicer/newsandevents.html>.

Usually this is also the version which the binary distributions available at

<http://www.virginia.edu/%7Ecese/research/crevicer/download.html>

are based on. If not otherwise stated, all procedures in this “Installation and Getting Started” will be based on these packages with version 1.0.

1.6 CREVICER models

From Kevin Stewart, one of the goals was to develop a general, adaptable model that could represent the two processes that are common to all forms of crevice corrosion: generation and transport. By performing the modeling in this fashion, which stresses the commonalities among different systems of crevice corrosion, it is possible to greatly shorten the development time of models that are specific to a particular system.

The following sections on CREVICER models were extracted from the Ph.D. dissertation of Kevin Stewart, *Intermediate Attack in Crevice Corrosion by Cathodic Focusing*, University of Virginia (1999).

General Aspects of Crevice Corrosion

In order to make a code which is suitable to model crevice corrosion for a wide variety of material/environment systems it is necessary to examine the factors which all instances of crevice corrosion have in common. At the broadest level, each instance of crevice corrosion is governed by generation and transport. The reaction rate at any interface is determined by the local conditions. The local chemistry, the sum of all the quantities of all the species present (C_i, \dots, C_j), the local electrochemical potential, E , and the temperature and absolute pressure (T and P) are sufficient to determine the reaction rate on any given interface. Understanding crevice corrosion is a matter of predicting how the local conditions, principally chemistry and potential, evolve in space and time and thus change the corrosion rate.

At steady state, the generation of chemical species and electrical charge at an interface is exactly balanced by their transport away from the interface. There are two methods of transporting chemical species, flux due to a gradient in electrochemical potential and flux due to physical flow of solution. The former is often broken into diffusion, flux due to an activity gradient, and migration, flux due to the interaction of charged ions with an electrical field. Flux due to the physical flow of solution is called convection. Electrical flux is always carried by ions in aqueous solution. It can be visualized by treating the charge flow as current through a medium of varying electrical resistivity.

In an occluded region, the crevice former retards transport and allows the reaction products from the interface to accumulate. The former acts by creating a long, thin path over which species and charge must move. For the transport of a fixed quantity of ions, a reduction in the cross-sectional area available for transport increases the required flux proportionately. Additionally, because the fluxes of both chemical species and electrical charge are determined by the gradients of the appropriate field variables, activity and

electrostatic potential, an increase in path length causes a proportionately larger total change between one end of the path and the other. Another retarding effect of a crevice on transport is that drag from the crevice walls dramatically reduces convective flow as the crevice narrows.

These change in local conditions caused by the retarding effect of the crevice former on transport affect the reaction rate, and if the entire process spirals out of control, crevice corrosion occurs. The general description of the preceding two paragraphs holds for all types of crevice corrosion. It is only the specific parameters such as the transport coefficients diffusivity and mobility (D_i and u_i), reaction rate as a function of chemistry and potential and any homogeneous chemical reactions such as hydrolysis or precipitation that make any material/environment combination unique.

A general model for crevice corrosion needs to solve the following two equations at every point inside the crevice:

$$\frac{\partial C_i}{\partial t} = -\nabla \cdot J_i(C_i \dots C_j, T, P) + R_i(E, C_i, \dots C_j, T, P) \quad (1)$$

$$\frac{\partial \phi_{el}}{\partial t} = -\nabla \cdot J_{el}(C_i \dots C_j, T, P) + R_{el}(E, C_i, \dots C_j, T, P) \quad (2)$$

Solution of these equations gives the concentration and electrostatic potential fields from which the electrochemical reaction rates can be generated. The generality for such a model comes from its ability to use different functional dependencies for the fluxes, (J_i , J_{el}), and generation rates, (R_i , R_{el}), for different materials. For example, the passive films on stainless steels are relatively unaffected by high levels of caustic whereas aluminum alloys begin to dissolve rapidly in extremely alkaline environments.

All types of crevice corrosion share the need to solve equations 1 and 2 and only differ in the exact effects of chemistry, potential, temperature and pressure on corrosion rate, homogeneous chemical reactions, and transport. The problem is one of how to represent the underlying equations and data in a fashion that allows the specifics to be readily changed without affecting the rest of the code. One approach for programming such a model is Object Oriented Design (OOD).

Object Oriented Design

One goal of Stewart's initial work was to create a model that was flexible enough to handle the variety of conditions that must be expressed in simulating crevice corrosion and was extendable to include future modeling efforts. To accomplish this, it was decided to use Object Oriented Design (OOD). OOD is a programming methodology, which is characterized by two main attributes: encapsulation and inheritance.

An object is piece of code meant to represent a physical entity or idea. It includes all of the data and all of the methods that operate on that data. An example, shown in Figure 1.1(a) would be an object representing a child's ball. 'Mass' and 'velocity' might be data properties of the object. 'Accelerate' would be a method property, included in

the 'ball' object, which uses the 'mass' property and the 'force' and 'time' inputs to change the 'velocity' property.

Encapsulation is accomplished by hiding the internal functionality of the object away from other objects or code. Objects interact with each other only in defined ways, and the necessary information is passed overtly through formal calls rather than accessed directly from one object to another. This approach allows individual objects to be modified without causing unforeseen errors in other parts of the program. Continuing the 'ball' example, the initial form of the 'ball' object might use Newton's First Law to calculate velocity changes in the 'accelerate' method. If the object is properly encapsulated, rewriting the 'accelerate' method to account for relativistic effects will correctly and uniformly affect the entire program. Figure 1.1(b) shows this more clearly. The encapsulated code can be changed and verified quickly and with little chance of introducing errors, while the non-encapsulated code is more difficult to modify and runs significant risks of introducing new errors elsewhere in the code when it is changed.

Inheritance allows an object to inherit the properties of a parent class. Additional properties can be added or inherited ones overridden. This characteristic is useful because it allows the reuse of code. An example drawn from the dissertation is the TASpecies class and its children classes, such as TClm as shown in Figure 1.1(c). For any chemical species, we are interested in the same types of properties. Physical properties such as charge number, z , and transport properties such as mobility, u , and diffusivity, D must be defined for all chemical species. Inheritance allows a derived property, such as D , to be easily and accurately applied to all child classes derived from TASpecies. Figure 1.1(c) shows the default behavior of TASpecies as using the Nernst-Einstein equation to calculate D from u and temperature, T . A child class, such as TClm, must give new values for identifying physical traits such as z and u , but automatically inherits the method of calculating D from u . This property saves code, but more importantly inheritance makes it easier to debug the overall code, since the method of calculating D must only be verified at one point rather than many. It should be pointed out that this inheritance in no way limits the ability of a child class to redefine any inherited property. If it were desirable to redefine D for a particular child class it could be done.

Encapsulation and inheritance allow the type of model to be created. Figure 1.1 (d) is a high level representation of the code developed in this project for the modeling of crevice systems. Conceptually, there are four areas that have to be addressed. By encapsulating them, each can be modified or improved without affecting the others. The central elements are the forms of the equations governing the accumulation of electrical charge and chemical species given in section above. Two separate types of information, specific to each crevice system, are required to produce the coefficients for those equations. The first is the physical geometry and construction of the crevice. This information defines how deep and narrow the modeled crevice is as well as the metal it is constructed of and the electrolyte which fills it. The second type of information concerns how materials and chemical species act under different conditions. Corrosion rates at different potentials and the effects of ionic strength on diffusion coefficients are examples. Finally, the last important conceptual area is the mathematical method, which

solves the specific equations for the chemical and electrical fields inside the crevice. In the program CREVICER which was developed for this dissertation the finite element method (FEM) was used for these calculations and a short discussion of the method is included in the next section.

Finite Element Method

Many possible methods can be used to solve differential equations. The most accurate techniques are analytical ones. They produce a solution in the form of an equation, which can be solved at every point inside the parameter space to give the exact solution. Unfortunately, analytic solutions are only available for a very limited number of geometries and boundary conditions. Typically the cases for which they are available are very simple and not representative of “real-world” crevices, though experimental cases have been constructed to fit the available analytic solutions. Numerical approaches are more general and can be applied to any well-defined geometry. Numerical solutions use computer algorithms to produce results that are approximately correct at a finite number of points. However, for many types of problems, numerical solutions are the only possible ones and they dominate in the fields of fluid flow, heat transfer, and stress analysis.

There are numerous methodologies for arriving at numerical solutions to differential equations as was discussed. Three principal ones are Finite Difference (FD), the Boundary Element Method (BEM) and the Finite Element Method (FEM). Each has advantages and disadvantages. FD is easy to code, but is computationally inefficient. BEM and FEM are both more complicated to code initially, but use computational resources more efficiently once implemented. BEM is more suited to solving differential equations over areas that have a low surface to volume ratio, while FEM better applied to volumes with large surface to volume ratios. For this thesis, FEM was the mathematical technique used to solve the differential equations governing the transfer of chemical and electrical flux. The method allows the grid to be adapted to maximize the resolution in areas of high gradients. The basic reference text used was Allaire.

Another important issue in solving the relevant differential equations is the dimensionality of the mathematical solution. The physical situation being modeled has three spatial dimensions and also varies in time. Limitations on the amount of computational power available constrain the number of dimensions that can be practically modeled, since all numerical methods depend on solving matrices of equations. The size of these matrices varies linearly with number of points or nodes where the equation is solved. The number of computations, and thus the time, required to solve the matrix varies as the square of the number of nodes - in O notation solving matrices is $O(n^2)$. The number of nodes depends on r , the resolution desired along an axis, and the dimensionality. In one-dimensional problems the resolution directly determines the number of nodes. If the equation is to be solved with a resolution of ten points along an axis, then only ten nodes are required. For two- and three- dimensional problems, however, the number of nodes required scales as the square and cube respectively of the

resolution. The amount of time required to solve a matrix for a fixed resolution varies as r^2 for one-dimension, r^4 for two dimensions, and r^6 for three dimensions. For his thesis, the solution is performed for two independent spatial dimensions and the temporal dimension and allowances are made to account for the effect of charges in a dependent third spatial dimension. This compromise choice of dimensionality provides a good match to the physical shapes of crevices, which are by definition much larger in two dimensions than they are in the third and also allows a higher r to be solved in the available time than if three independent spatial dimensions were used. The method used for reflecting dependent changes in the third dimension allows CREVICER to reflect gross changes in that dimension, such as an opening crack, but does not permit the effects of surface roughness to be included.

The following subsections address the implementation of FEM for the spatial and temporal dimensions and the limitations of the method for both.

Typically, CREVICER is run in Crank-Nicolson mode which is a combination of the explicit and implicit techniques. This reduces some of the problems with stability that are discussed in the next section.

Applications of FEM to Crevice Corrosion

CREVICER uses FEM to model both electrical potential and chemical concentration fields. Each case can be represented in the canonical form by using the relation that the time rate of change is the negative of the divergence of the appropriate flux plus the rate of generation. This is done first for electrical charge. The electrical flux is equal to the conductivity times the gradient of electrostatic potential, as shown in

$$J_{el} = \kappa \nabla \phi \quad (3)$$

Equation below.

Where:

J_{el}	is the electrical flux [C/m ² -s]
κ	is the conductivity [(Ω -m) ⁻¹]
$\nabla \phi$	is the gradient of electrostatic potential in solution [V/m]

The rate of accumulation of charge at each point is found by taking the divergence of this flux and adding the rate of generation of electrical charge. Since charge is only generated at electrochemical interfaces, the rate of generation is the product of the reacting area and the rate divided by the volume of interest.

$$\frac{\partial \rho_{el}}{\partial t} = -\kappa(x, y) \nabla^2 \phi + \frac{J_{el} A(x, y)}{h(x, y) A(x, y)} \quad (4)$$

Where:

Δ_{el}	is the charge density [C/m ³]
---------------	---

$h(x,y)$ is height at point (x,y) [m]
 A is area [m^2]

By canceling terms and multiplying through by h , the Equation above is placed in the proper form for solution by FEM. Table 1.1 shows the correspondence between the terms for FEM and the physical parameters of the crevice.

$$h(x,y) \frac{\partial Q}{\partial t} = -\kappa h(x,y) \nabla^2 \phi + J_{el}(x,y) \quad (5)$$

Table 1.1 The coefficients for use in FEM can be related to physical parameters for use in calculating electrostatic potential fields.

FEM Coefficient	Physical Parameters
K_t	$h(x,y)$
$K_x = K_y$	$-\kappa(x,y) h(x,y)$
$M_x = M_y$	0
P	0
Q	$J_{el}(x,y)$

A similar analysis can be performed for the case of chemical concentration fields. The starting point is the flux equation using the gradient of concentration for diffusivity and ignoring convection for the reasons discussed in section 2.3.1 (Refer to Stewart Dissertation).

$$J_i = -D_i \nabla C_i - z_i F u_i C_i \nabla \phi$$

The rate of accumulation of chemical species at each point is found by taking the divergence of this flux and adding the rate of generation of chemical species. Because charge is only generated at electrochemical interfaces, the rate of generation is the product of the reacting area and the rate divided by the volume of interest.

$$\frac{\partial C_i}{\partial t} = D_i(x,y) \nabla^2 C_i + z_i F u_i(x,y) \nabla \phi \nabla C_i + \frac{J_i(x,y) A(x,y)}{A(x,y) h(x,y)}$$

By canceling terms and multiplying through by h , Equation above is placed in the proper form for solution by FEM.

$$h(x,y)\frac{\partial C_i}{\partial t} = h(x,y)D_i(x,y)\nabla^2 C_i + h(x,y)z_i F u_i(x,y)\nabla \phi + J_i(x,y)$$

Table 1.2 shows the correspondence between the terms for FEM and the physical parameters of the crevice.

Table 1.2 The coefficients for use in FEM can be related to physical parameters for use in calculating chemical concentration fields.

FEM Coefficient	Physical Parameters
K_t	H
$K_x = K_y$	$h(x,y)D_i(x,y)$
$M_x = M_y$	$h(x,y)z_i F u_i(x,y) \text{ grad } \phi$
P	0
Q	$J_i(x,y)$

Program Structure and Simplifications used in CREVICER

CREVICER is a two-dimensional FEM code for the solution of time-varying partial differential equations used in modeling the chemical concentration and potential fields of occluded regions. This section will address the basic structure of the program and the simplifications used within it. Any text marked **(Bold)** refers to a data structure or procedure in CREVICER and is included to more directly tie this subsection to the actual code. Any item with a “T” prefix is an object.

Program structure

Figure 1.2(a) shows the major information flows within CREVICER. The diagram is complemented by Figure 1.2(b) which illustrates the objects and their inter-relationships. The **Setup** procedure uses the input node and element files to create the two data structures needed to define a crevice: the nodes and elements arrays. The nodes array contains all of the nodes in the crevice. Each node (**NodeInfo**) consists of an **x**, **y**, **z** position for the node, the conditions at the current time step (**chem**) and the conditions at the previous time step (**oldchem**). The z coordinate is not used by CREVICER, but is included for future functionality. The elements array stores all of the elements that form the crevice. Each element (**TSolutionVolume**) lists the **i**, **j** and **k** nodes that define the element, the height (**h**) of the element, and the material of which it is constructed (**mat**), and the materials along each of the edges if needed (**ijmat**, **ikmat**, **jkmat**).

The program then consists of two loops: one for activities that happen within a time period, and another for the series of time periods being modeled. The inner loop includes an additional iterative step of variable length that is used to find the potential field.

The inner loop finds the chemical concentration or electrical field at the next time period for each field variable (concentration or potential) in turn. In order to find m field variables at n nodes the loops solves an n by n matrix m times. This approach is faster than solving an mn by mn matrix once by a factor of $1/m$. The conditions of charge neutrality and chemical equilibrium are enforced after the transport of all the species for each time period.

The matrix of equations is developed using the FEM equations. The mode of the solver, steady-state vs explicit, implicit or mixed is controlled by \mathbf{K}_t , θ (**theta**) and Δt (**time**). The matrix is constructed element-by-element. Three equations are generated by each element, one for each node. The entries into the matrix depend on the \mathbf{x} , \mathbf{y} positions of the nodes, the values of the field variable (in **oldchem**), and \mathbf{h} .

Equations for chemical species also depend on z_i , D_i , and u_i . These parameters are encapsulated in **TAllSpecies**, a container class for all possible species (refer to Figure 1.2 b). This object in turn accesses the values for a particular species through the particular **TASpecies** which defines it. The heterogeneous reaction rates are queried from the element's **mat** (which is a **TMaterial**). A material sums the reaction rates from the various **TReaction**'s that are included in it. At a minimum a **TMaterial** should have one anodic and one cathodic **TReaction**. This separation of the reactions on a surface into different groups not only provides better fidelity to the physical situation, it also allows chemical fluxes even when the electrical flux is small or zero. This ability is vital if the crevice will be depolarized to E_{corr} . Information on E , C_i, \dots, C_j , T and P is passed to each of the objects when a request is made for any parameter. The end of this subsection lists which of the parameters are used, the unused parameters are passed for future use.

The procedure for electrical transport is similar. The transport parameters are replaced by κ (**kappa**) which is calculated based on the local **chem**. Generation rates are found from the appropriate **TMaterial** which in turn uses the **TReaction**. One difference is that because electrical processes equilibrate much more rapidly than chemical ones an additional loop is performed only on the electrical parameter to find the proper potential distribution for each time step.

Chemical and charge equilibria are enforced once all the transport steps have occurred. These routines are hard-coded as procedures into **TAllspecies**. Once equilibrium is restored output occurs and the new **chem** for each node becomes its new **oldchem** and the process restarts.

Figure 1.2(b) is a view of the code that emphasizes objects. The **Main()** routine initializes the Elements and the Nodes with the geometry (**Elements** and **Nodes**) as well as the boundary and initial conditions. **Elements** is an array of **TSolutionVolume**. **Nodes**

is an array of **NodeInfos**. Each **Nodeinfo** contain two **TChemistries** one for the current and one for the next time period. The **Main()** procedure also controls the use of **Solve()**. **Solve()** operates on each species in turn. It builds an array of simultaneous equations and solves it to find the value of the field variables at the next time step.

Solve() requests from each **TSolutionVolume** in **Elements** the values of the coefficients in the canonical equation (**K_x**, **K_y**, **M_x**, **M_y**, **P** and **Q**). These parameters are generated from the appropriate transport coefficients and generation rates.

TAllSpecies is a container class for **TASpecies**. It passes through information on conditions such as **E**, **C_i ... C_j**, **T** and **P** to the correct **TASpecies** and receives and returns the appropriate parameter, such as **D**, **u** or **z**.

Each **TMaterial** performs a similar function for the **TReactions** it contains. The local conditions are passed to each **TReaction** and the resulting fluxes of chemical species and electrical current (**J_i** and **J_e**) are summed and returned.

More information about the structure of the code can be obtained by reading the comments in the program itself. The individual files are listed in Appendices A - S. As a further aid the next subsection will address all the simplifications used in the code.

Simplifications in CREVICER

Several types of simplifications have been made in CREVICER. The physical geometry of a crevice is approximated, the governing equations are simplified and several of the terms ignore the full functional dependency on the local conditions. The format used in this subsection is name of the concept or term being simplified, the relevant sections of Stewart dissertation where the problem is discussed and the approach taken towards to problem in CREVICER.

Governing Equations

Convection:

Relevant Sections: 2.3.1.1.1 (Refer to Stewart dissertation)

Approach Used: CREVICER ignores convection. Small crevice gaps are unlikely to experience much flow unless a driving mechanism like crack pumping in corrosion fatigue is present.

Diffusion:

Relevant Sections: 2.3.1.1.2, 2.3.2.1

Approach Used: CREVICER calculates diffusion based on the gradient of concentration rather than the gradient of activity. This is a valid assumption if the gradient of the activity coefficient with respect to chemistry is zero. **D** is determined separately for each element according to the method described in 3.5.2.2.

Migration:

Relevant Sections: 2.3.1.1.3

Approach Used: Migration is calculated based on ϕ_s , the potential in solution. It ignores the diffusion potential. u is determined separately for each element according to the method described in 3.5.2.2.

Potential:

Relevant Sections: 2.3.1.4, 2.3.1.5, 2.3.1.6, 2.3.2.4

Approach Used: The potential field is found from Ohm's law. It ignores the diffusion potential. κ is determined separately for each element according to the method described in 3.5.2.2.

Terms

z:

Relevant Sections: None

Approach Used: z is a constant for each chemical species and is stored in the appropriate **TASpecies**. This is not an approximation.

u:

Relevant Sections: 2.3.2.2, 2.3.3.2.1

Approach Used: The mobility for each species used in CREVICER is a constant stored in the appropriate **TASpecies**. E , C_i, \dots, C_j , T and P are passed when u is called but no accounting is made for the effects of those variables or the derived effects of viscosity and tortuosity.

D:

Relevant Sections: 2.3.2.2, 2.3.3.2.1

Approach Used: The diffusivity for each species used in CREVICER is calculated from the Nernst-Einstein equation. D is proportionate to the absolute temperature. No accounting is made for the effects of P the chemistry or the derived effects of viscosity and tortuosity.

a:

Relevant Sections: 2.1.2.1, 2.3.1.1.2, 2.3.2.1

Approach Used: The single ion activity is not used in CREVICER. In the future it could be used for diffusive flux or to modify the transport coefficients. It is here that concentrated solution effects should eventually be accounted for. The function call passes E , C_i, \dots, C_j , T and P .

T:

Relevant Sections: 2.3.3.1.1

Approach Used: Temperature is only used in CREVICER in calculating D from u according to the Nernst-Einstein relationship. It is consistently exchanged between objects to allow for future functionality.

P:

Relevant Sections: 2.3.3.1.2

Approach Used: Pressure is not used in CREVICER. It is consistently exchanged between objects to allow for future functionality.

Heterogeneous and Homogeneous Reactions

J_{el}:

Relevant Sections: 2.1.2, 2.1.3, 2.1.4, 2.3.3.2.3, 2.3.3.3.4

Approach Used: Electrical flux is determined in CREVICER by summing the electrical fluxes from each of the component reactions on a **TMaterial**. This summation is done with the convention that anodic currents are positive and cathodic ones negative. The function call passes E, C_i,...,C_j, T and P. Some reactions in CREVICER use C_i to adjust the polarization curve. The polarization curve itself can be programmed in any functional format. CREVICER has employed constant, Tafel and polynomial representations for the polarization curve.

J_i:

Relevant Sections: 2.1.2, 2.1.3, 2.1.4, 2.3.2.3.1, 2.3.3.3.4

Approach Used: Chemical flux is determined in CREVICER by summing the fluxes from each of the component reactions on a **TMaterial**. This summation is done with the convention that dissolution reactions are positive and plating ones negative. The function call passes E, C_i,...,C_j, T and P. Some reactions in CREVICER use C_i to adjust the polarization curve. The polarization curve itself can be programmed in any functional format. CREVICER has employed constant, Tafel and polynomial representations for the polarization curve.

Homogeneous Chemical Reactions:

Relevant Sections: 2.3.2.3.2, 2.3.3.2.4

Approach Used: Homogeneous chemical reactions including hydrolysis can be programmed into CREVICER. Chromium hydrolysis is already implemented. It uses only the mononuclear species and assumes that hydrolysis is fast compared to transport.

Derived Parameters

pH:

Relevant Sections: None

Approach Used: CREVICER reports the pH of a chemistry for use in chemically sensitive polarization curves or for output purposes. The code assumes the dilute solution approximation and returns the negative of the log of the H⁺ concentration.

I:

Relevant Sections: 2.3.2.3.2.1

Approach Used: CREVICER reports the ionic strength of a chemistry for use in

chemically sensitive polarization curves or for output purposes. The code uses Equation 83 given in 2.3.2.3.2.1.

η :

Relevant Sections: 2.3.2.2.2

Approach Used: Viscosity is not used in CREVICER, nor is it even a defined parameter. If a general method of calculating the viscosity were developed it would have to be implemented as a new object.

Tortuosity:

Relevant Sections: 2.3.2.2.3

Approach Used: Tortuosity caused by precipitation is not used in CREVICER, nor is it even a defined parameter. If a general method of calculating the porosity were developed it would have to be implemented as a new object.

Ball Object

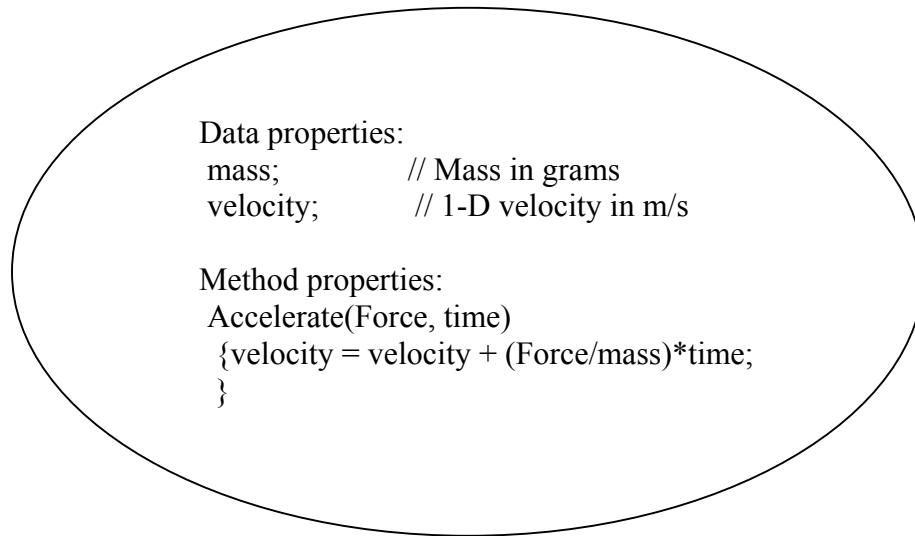


Figure 1.1(a) An object is a logical grouping of data elements and methods that reflect a physical item or idea. In this case a ball is represented by its 'mass' and 'velocity'. The 'accelerate' method uses the inputs of 'force' and 'time' to change the velocity based on the mass.

Encapsulated	Not Encapsulated
<pre>ThrowToFirst(Force, time) {ball->accelerate(Force,time); speed = ball->velocity; }</pre>	<pre>ThrowToFirst(Force, time) {speed =(Force / (ball->mass/sqrt(1-(speed/c)^2)) * time; }</pre>
<pre>ThrowToSecond(Force, time) {ball->accelerate(Force,time); speed = ball->velocity; }</pre>	<pre>ThrowToSecond(Force, time) {speed =(Force / ball->mass) * time; }</pre>

Figure 1.1(b) Encapsulation hides the functionality of an object. In the code on the left the 'ball' object is properly encapsulated. If the programmer wishes to treat the ball in relativistic rather than a Newtonian fashion, only the 'accelerate' parameter of the ball object need be changed. Procedures which depend on the speed of the ball will automatically and properly reflect the change. In the code on the right however, each procedure must be individually modified and this leaves significant room for errors to occur. 'ThrowFirst' now reflects the Lorentz transformation, but 'ThrowToSecond' does not. Not only might a procedure be missed, but each must be debugged and verified, instead of verifying only the 'accelerate' method.

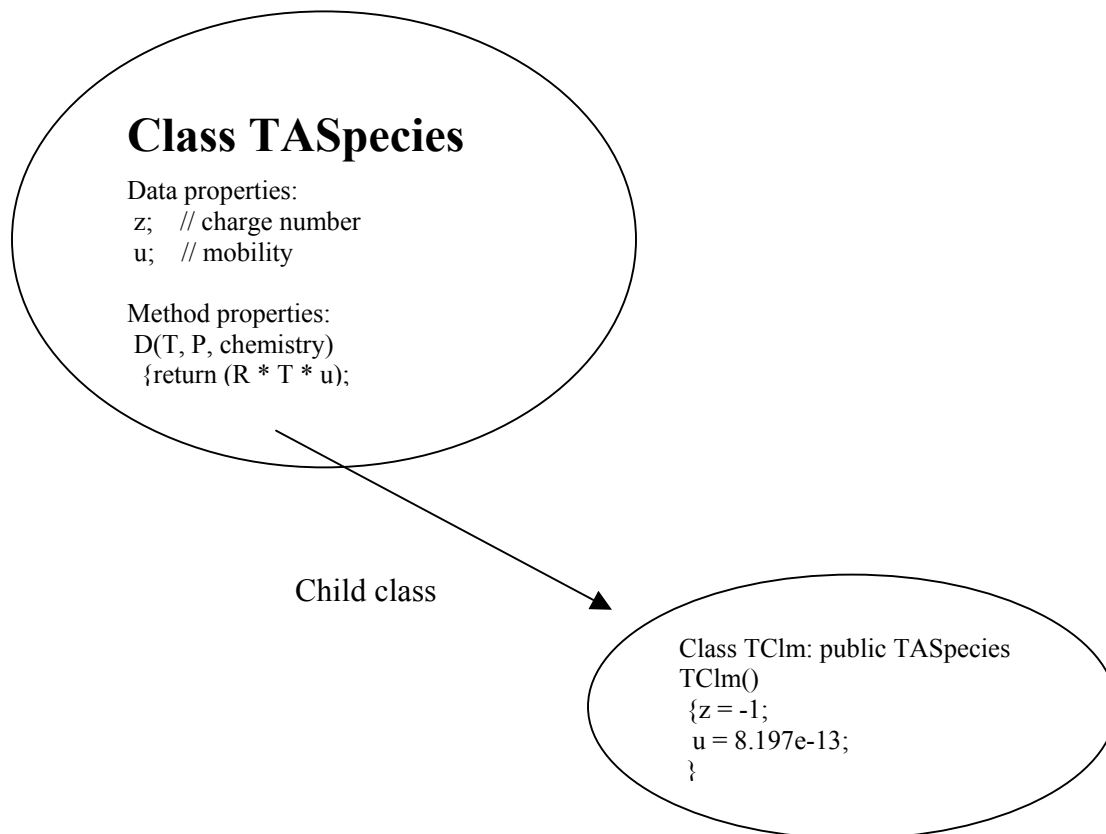


Figure 1.1(c) Inheritance allows code to be reused very efficiently. A parent class, such as TAspecies, can be written once to provide generic functionality needed by all of its children classes. In this case, one such method shared by all chemical species is how diffusivity, D, is calculated from mobility, temperature and the gas constant. A child class need only define its unique properties, such as the relevant mobility. The method of calculating D is inherited from TAspecies. In addition to saving code, inheritance increases accuracy, since methods like D need only be defined once, at the top level in TAspecies rather than individually in each child class.

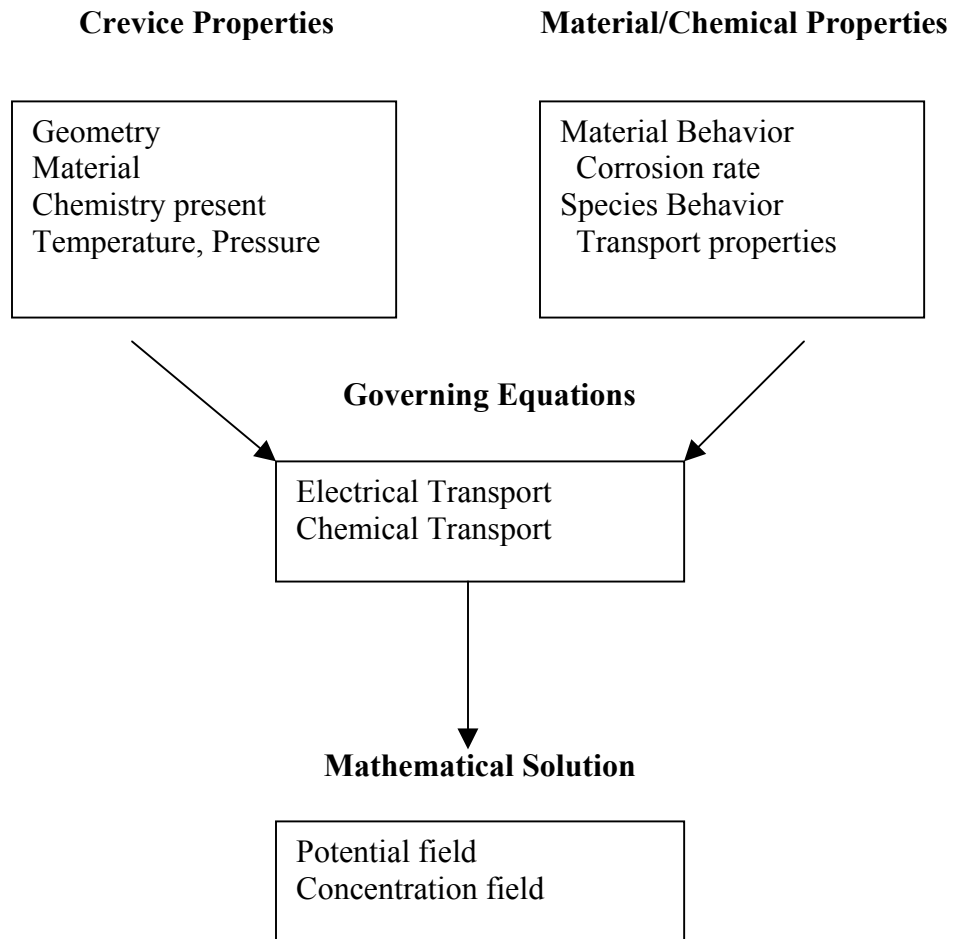


Figure 1.1(d) CREVICER consists of four main conceptual areas which are encapsulated by OOD. The governing equations are central and common to all possible crevices. The coefficients in these equations are determined by the crevice-specific geometry and the material-specific behavior. Finally, the solution of the governing equations for the crevice's concentration and electrical fields is accomplished separately. This approach allows any area to be refined independently of the others.

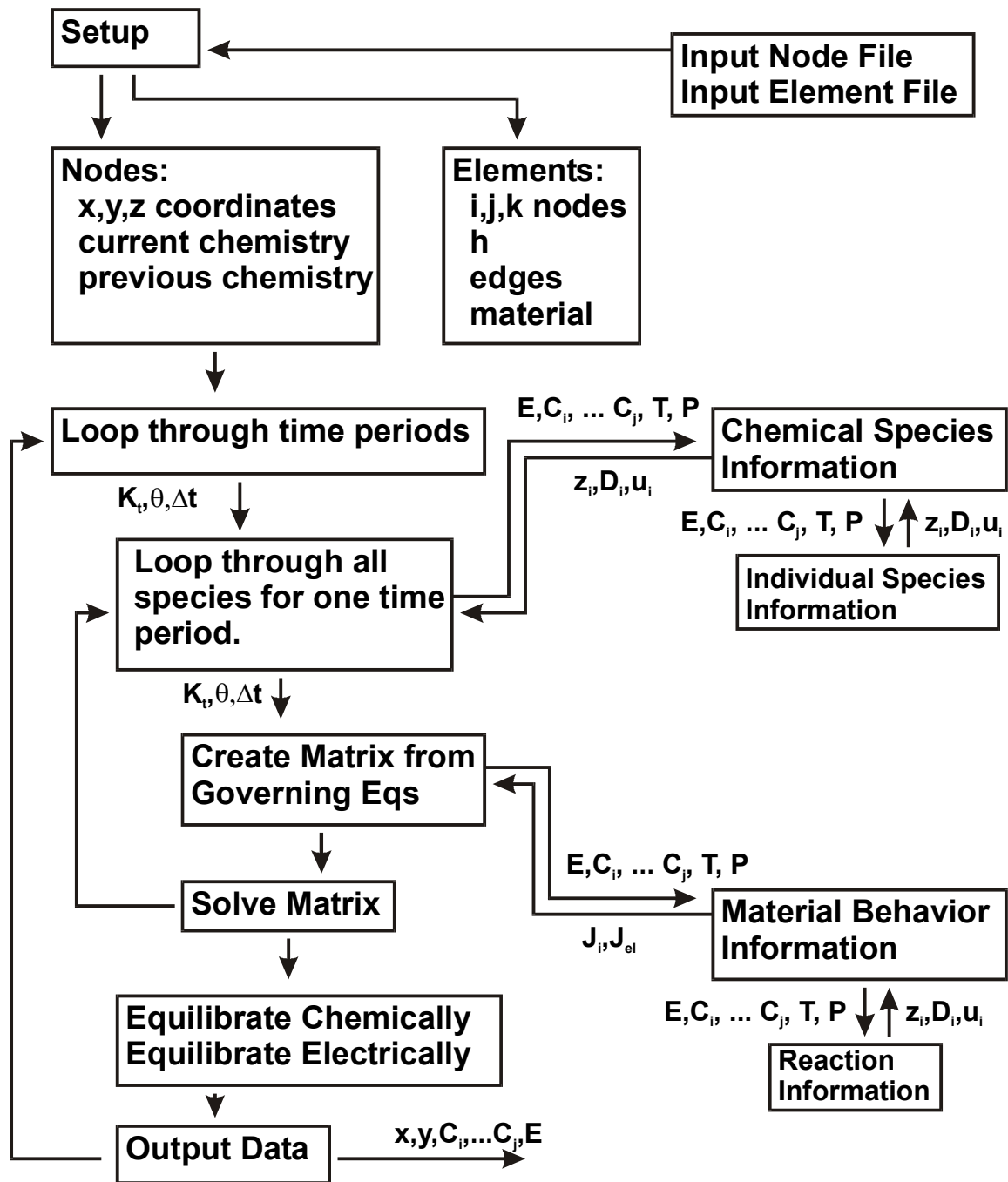


Figure 1.2 (a) Information in CREVICER flows according to the arrows. Labels on individual arrows list the major information exchanged. The node and element files are used by the Setup procedure to create the node and element arrays. The code then performs two loops. The inner loop iterates through the potential and individual species creating and solving the appropriate matrices. Information on the behavior of individual chemical species and material is used to create the matrices. K_t and θ are used to set the method (explicit/implicit/Crank-Nicolson) for the temporal step. Δt also is used in the matrices. The outer loop enforces charge and chemical equilibria and performs output for each time step.

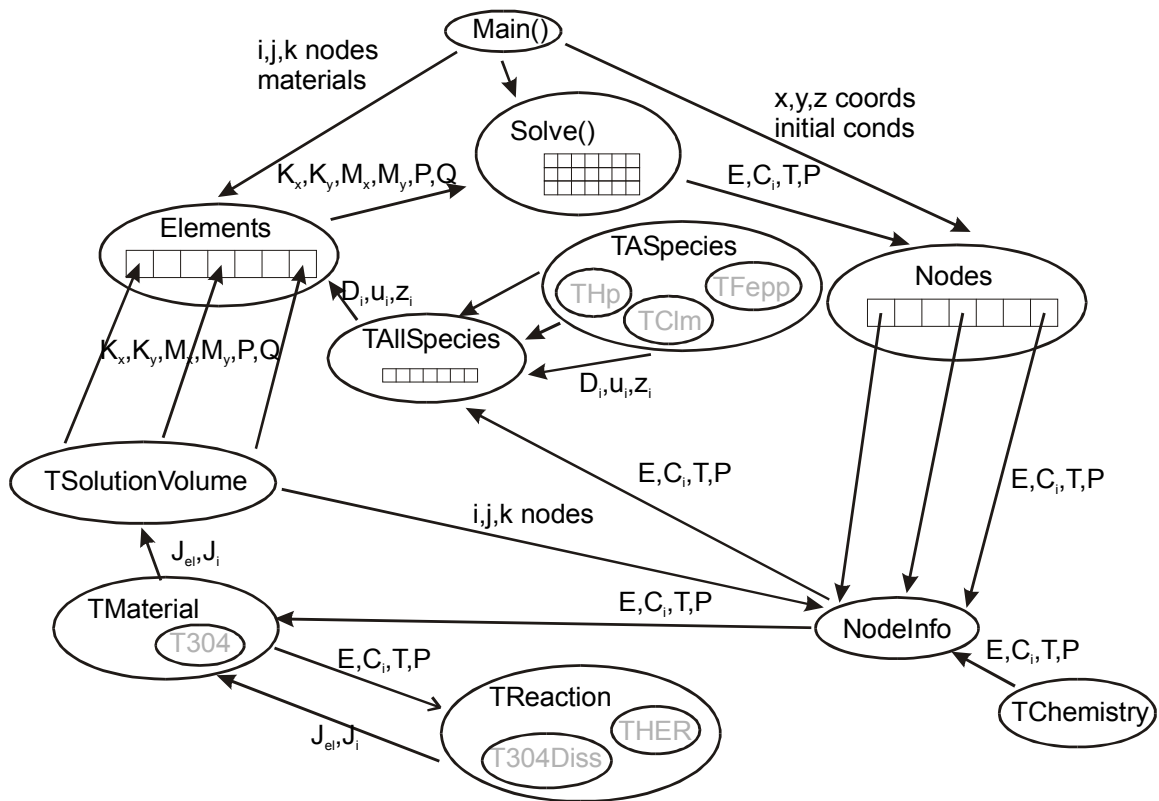


Figure 1.2 (b) Object dependencies and handshaking for the various objects in CREVICER. Circled items are objects and the labeled arrows represent the information flows.

This section has covered the general structure needed by code that models crevice corrosion, the suitability of OOD for use in such a code, the basis and limitations of FEM for solving such a model and presented how the physical parameters are translated into the form needed for solution by FEM. It concludes by presenting the structure and approximations of the CREVICER code.

Chapter 2

Building the crevice: Compiling the program

This central chapter describes how to build CREVICER on Windows systems. You may not want to go through that potentially troublesome process, but skip this chapter instead and straightly go to the next one in which how to use the executable is described. However, there may be good reason for at least trying to build the simulator:

Compiling CREVICER is not a task for novice users. Thus, if you're a beginner (we all were once) on a platform which binaries are available for, we recommend postponing this task and just starting with the binary distribution (executable file) to get you playing.

In this chapter, we describe compiling for one operating systems only, Windows, and for only one compiler, the Microsoft Visual C++ 6.0 compiler. It is expandable to Linux system which is left to the interested developers.

2.1 Getting a development environment under Windows

Contrary to Linux/Unix systems, Windows usually comes without any development tools. Thus, you first have to install a development environment. On Windows, in a sense, before building the crevice you will have to build the plant for building crevices. You need have a Microsoft Visual Studio including the C++ 6.0 or higher version installed in your Windows system.

2.2 Download the CREVICER source code

The following supposes you are on a Windows (95/98/Me/NT/2000/XP) system. To begin with, the CREVICER build process is based on three packages that you have to downloaded from

<http://www.virginia.edu/%7Ecese/research/crevicer/download.html>

as follows:

- [SourceCrevicerBase.zip](#)
- [SourceCrevicerGui.zip](#)
- [SourceCrevicerSolver.zip](#)

to a drive of your choice. Windows XP includes a program for unpacking *.zip files. If you are working under an older version of Windows, we suggest getting

Winzip from

<http://www.winzip.com/>.

For a free alternative, you may consider unzip from Info-ZIP,

<http://www.info-zip.org/pub/infozip/>

Extract the files named above.

After unzipping these three packages, you will find a new folder

`/usr/code/`, including three new folders

CREVICERBASE

CREVICERGUI

CREVICERSOLVER

Now you can check the file information in each folder.

1) CREVICERBASE folder:

This package includes the following files,
Parameters.txt, includes the initial value of input parameters
ELVAR350, includes the element information
NOVAR350, includes the node information

2) CREVICERSOLVER folder

Open the C⁺⁺ project in `/usr/local/code/CREVICERSOLVER/` folder. If you see the above files as shown in Figure 2.1, go to next step.

3) CREVICERGUI folder

Open the C⁺⁺ project in `/usr/local/code/CREVICERGUI/` folder. If you see the above files as shown in Figure 2.2, you already have a correct download.

2.3 Compiling CREVICER under Windows

Open the C⁺⁺ project Allin.dsw in `/usr/code/CREVICERSOLVER/`, compiling will generate the ALLin.exe in the `/usr/code/CREVICERSOLVER/debug` directory. Copy it exe file to `/usr/code/CREVICERBASE`.

Open the C⁺⁺ project ALV2GUILin.dsw in `/usr/code/CREVICERGUI/`, compiling will generate the ALV2GUILin.exe in the `/usr/code/CREVICERGUI/debug` directory. Copy it exe file to `/usr/code/CREVICERBASE`.

2.4 Running the CREVICERGUI or CREVICERSOLVER

Now you should find CREVICERSOLVER executable file (Allin.exe), CREVICERGUI executable file (ALV2GUIIn.exe), and initial data file under /usr/code/ CREVICERBASE. You can run the CREVICERGUI to start the simulation.

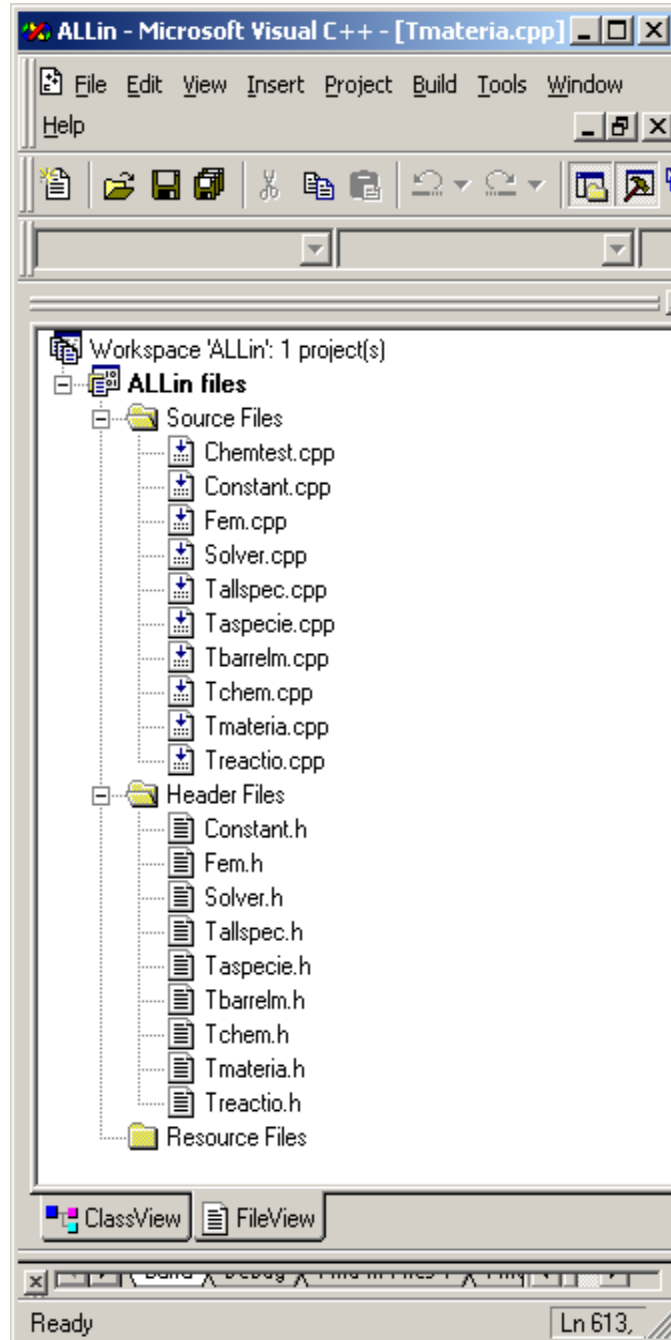


Figure 2.1 File directory of CREVICESOLER

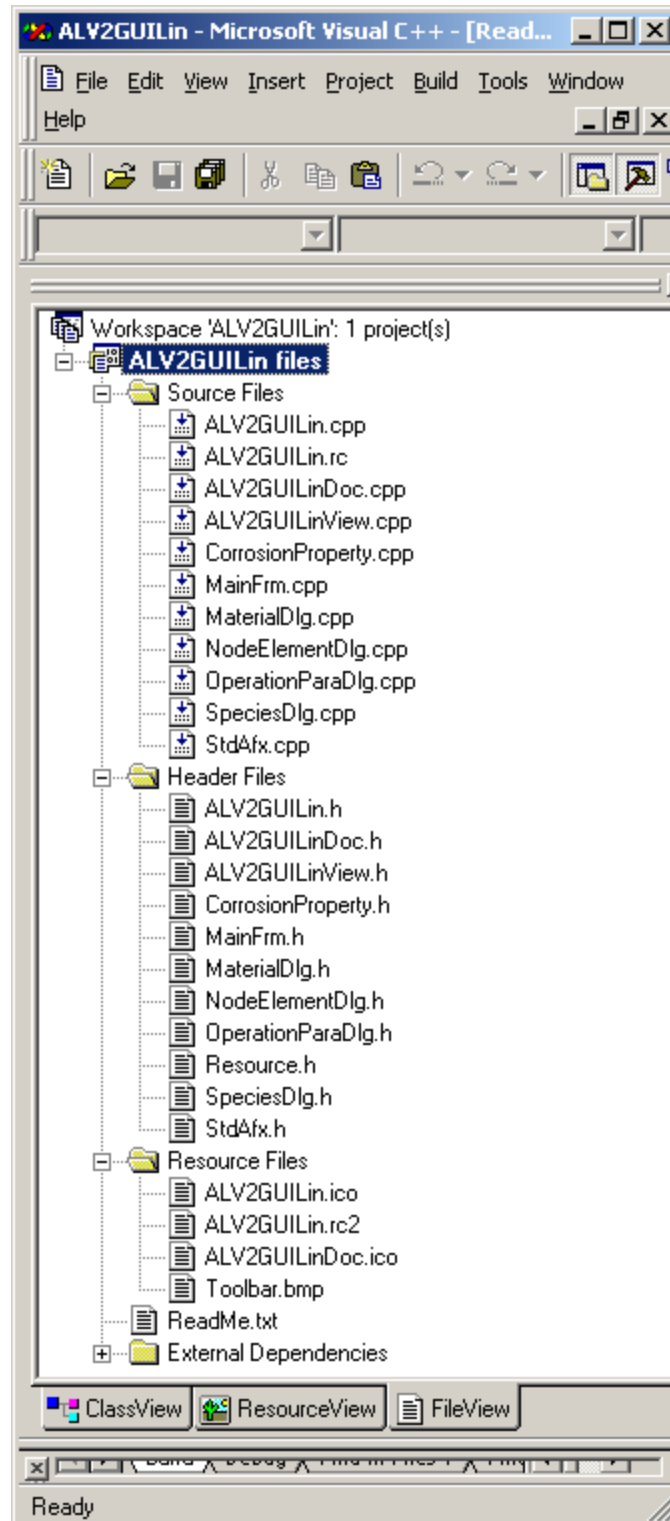


Figure 2.2 File directory of CREVICERGUI

Chapter 3

Installing CREVICER

You can skip this section if you built CREVICER along the lines described in the previous Chapter. If you did not and you're jumping in here, your first step will consist in installing the binaries. At present, there are pre-compiled binaries available for Windows system.

3.1 Installing the binary distribution on a Windows system

The following supposes you are on a Windows (95/98/Me/NT/2000/XP) system. Installing the binaries is quite simple. Go to

<http://www.virginia.edu/%7Ecese/research/crevicer/download.html>

and download one package: [BinaryCrevicer.zip](#)

to a drive of your choice. Windows XP includes a program for unpacking *.zip files. If you are working under an older version of Windows, we suggest getting Winzip from

<http://www.winzip.com/>.

For a free alternative, you may consider unzip from Info-ZIP,

<http://www.info-zip.org/pub/infozip/>

Extract the files named above.

3.2 Installing documentation

Most of the packages named above include the complete CREVICER documentation including a .pdf version of the CREVICER Simulator - Installation and Getting Started using Adobe's Acrobat Reader being available from <http://www.adobe.com/acrobat> and other design documentation.

Part II

Playing with CREVICER

Chapter 4

Crevice corrosion parameters and menus

4.1 Starting the GUI

Here is an overall procedure on how to use the GUI.

- 1). Execute the ALV2GUILin.exe in the /node/CREVICERBASE directory.
- 2). First click the Setting button to select node and element files, materials, species, and operation conditions. The settings command will read the last used simulation parameters (parameters.txt), offer users a dialog to select the new simulation parameters, and generate a new *parameters.txt* file.
- 3). Click the Run button to run CREVICERSOLVER. This command will call the ALLin.exe, which reads the new *parameters.txt* file, to run simulation. After simulation, a message box appears. Click OK.
- 4). After Simulation, several result files will be written to the directory (e.g. elements.txt, Potential1.txt, H+_1.txt, etc.). More information about every step is given in the case study at section 4.3.3.

4.2 Implementation of functions of GUI

The function writes the user-selected parameters to a text file named *paratmers.txt*, which has the following format:

parameters.txt

```
-----  
Node and Element:  
5  
novar350          !File name for nodes, 5 for novar350  
5  
elvar350          !File name for elements, 5 for elvar350  
10  
395  
1  
1  
Nickel            !Material name for crevice corrosion substrate  
Operation Conditions:  
-0.2              !Potential at tip (volt)  
0                 !Variable  
0.6               !Potential at mouth  
298               !Temperature at tip (K)  
1                 !Fixed  
298               !Temperature at mouth (k)  
100000            !Pressure at tip (Pa)  
1                 !Fixed
```

```

100000          !Pressure at mouth (Pa)
Add Species:
:Species 1
11
H+              !Species 1 name
428            !Concentration for species 1, (mol/m3)
0              !Fixed or variable, 0-variable, 1-fixed
428            !BC for species 1
1              !BC is fixed
:Species 2
15
Ni++           !Species 2 name
0              !Concentration for species 1, (mol/m3)
0              !Fixed or variable, 0-variable, 1-fixed
0              !BC for species 2
1              !BC is fixed-1, variable-0
:Species 3
17
O2             !Species 4 name
0.6           !Concentration for species 1, (mol/m3)
0             !Fixed or variable, 0-variable, 1-fixed
0.44         !BC for species 4
1            !BC is fixed-1, variable-0
:Species 4
19
SO4--         !Species 3 name
428          !Concentration for species 1, (mol/m3)
0            !Fixed or variable, 0-variable, 1-fixed
214         !BC for species 3
1            !BC is fixed-1, variable-0
&

```

When the user clicks the Setting button, GUI will read the *parameters.txt* file used in last simulation. After the user selects the new parameters and clicks the OK button, a new *parameters.txt* file will be written by the GUI and then read by the CREVICERSOLVER in the exact same format and sequence.

The Setting dialog contains a PropertySheet control, which includes four tabs (PropertyPage control): Node and Element, Select Material, Add Species, and Operation Conditions.

Node and Element Tab:

When the Node and Element tab is clicked, two combo boxes offer filenames for node and elements. It offers 6 options for element files:

- ELVAR350
- EL500
- EL2X300
- EL3X400
- ELROD
- ELROD500

It also offers 6 corresponding options for node files:

- NOVAR350
- NO500
- NO2X300
- NO2X400
- NOROD
- NOROD500

For the program to run correctly, the selected node file and element file need to be in the project directory. Now only the ELVAR350 and NOVAR350 are included in the open source code.

In addition, it also includes several options for the crevice gap size.

Select Material Tab: (μm)

- 5
- 14
- 25
- 35
- 50
- 75
- 93
- 106
- 130
- 153
- 395

When the Select Material tab is clicked, the combo box offers users to select from 7 kinds of materials:

- Ag
- Nickel
- Gold
- 304Mat
- SAF2205

FOOMaterial
LisaMat

These seven materials have been defined in the CREVICERSOLVER. When new material types are defined in the CREVICER, they can also be added to the GUI. Currently only Nickel is open to the public.

Add Species Tab:

It offers users four combo boxes to add four species. Future public releases will allow users to add any number of species they want. For each species, the user can specify if the concentration is fixed or variable, the initial concentration, the boundary condition (i.e., the concentration at the crevice mouth), and if the boundary condition is fixed or variable.

Operation Conditions Tab:

It lets users specify the operation conditions: potential, temperature, pressure, and their boundary conditions.

4.3 Case study

4.3.1. Physical phenomena –crevice corrosion

Why crevice corrosion?

One of the materials scientist's interests is the corrosion that occurs to a material, e.g. steel or alloy, surface beneath a faster. The crevice corrosion is related to the size of the micro-scale gap that exists between the two objects.

What challenge do we have to study crevice corrosion?

In order to confirm computer simulations we developed, we needed a test model with rigorously define, micron-scale geometry. DeJong, Lee, and Kelly produced the practical size crevice by nickel with microfabrication technique available in semiconductor industry. They uncovered new relationships that shape corrosion. It set the stage for analyzing the process itself. Now we can trace the mechanisms as well as the magnitude of crevice corrosion under clearly controlled circumstances.

Figure 1 shows the schematic of a crevice formed by a former and substrate. The critical parameters of crevice include the gap of crevice (g), and the length of crevice (l).

The gap of crevice in Lee's experiments ranges from 10 μm to 600 μm . The length is 7 mm. The crevice substrate is nickel 200.

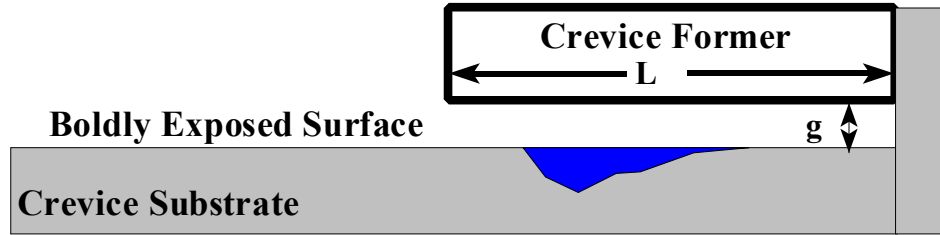


Figure 4.1 A schematic of a crevice formed by a former and substrate.

4.3.2. Crevice corrosion experiments

The following sections on CREVICER models were extracted from the M.S. thesis of Jason Lee, Investigation of Crevice Corrosion Using Computational Modeling and Microfabrication Techniques, University of Virginia (2001).

4.3.2.1 Crevice assembly

Patterned silicon and Ni200 substrates were affixed to a 3 x 3 inch piece of Plexiglas using Dow Corning vacuum grease. The grease held the substrate in place and electrically isolated the back of the substrate. A crevice former was placed with the patterned side down onto the substrate area. Two optical microscopes, one with a top-down view and one with a side-view were used to align the crevice former over the substrate area. The goal was to match the photoresist boundaries of both the former and the substrate so that a gap of continuous height was created throughout the crevice. A smaller piece of Plexiglas was placed on top of the former and held down with nylon screws that connected the two pieces of Plexiglas.

An electrical connection to the substrate was made by connecting six inches of 0.1 mm diameter Premion (99.998% pure) platinum wire to the pattern-defined substrate contact patch using conductive adhesive. The entire assembly was vertically aligned in the corrosion-testing cell. Only the very bottom of the crevice former was allowed to touch the surface of the acid solution. Capillary action drew the solution up into the crevice. This position was maintained throughout the experiment to prevent current from being able to leak out from the sides of the crevice. The platinum wire was connected to the working lead of the potentiostat. A platinum-niobium mesh and a saturated calomel electrode were used as the counter and the reference electrode respectively. Before each test, the open circuit potential of the crevice was allowed to stabilize. Stabilization was defined as having occurred when the open circuit potential varied by only ± 1 mV over a 5-minute period. Stabilization took between 30 minutes and 4 hours. A schematic of the corrosion-testing cell can be seen in Figure 2.

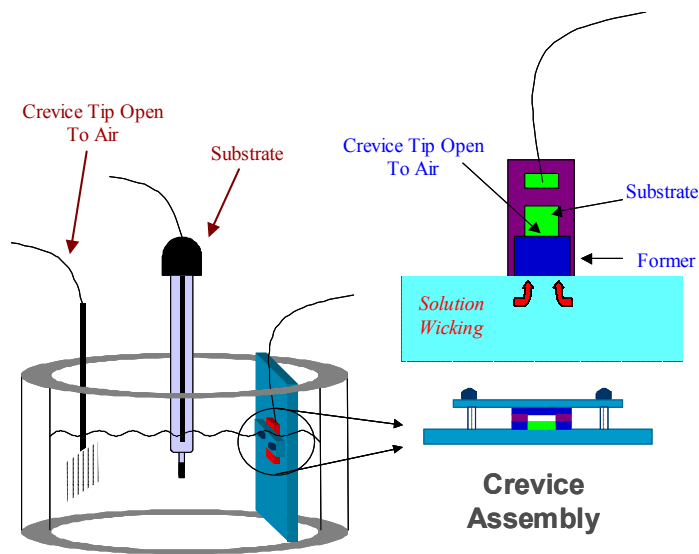


Figure 4.2 A schematic of the crevice assembly. Only the crevice mouth was placed into the solution. The solution wicked up the entire crevice length due to capillary action.

4.3.2.2 Potentiodynamic scans

Both electroplated wafers and polished Ni200 samples were used in polarization scans. The Ni200 samples were able to be loaded directly into a standard flat-cell, whereas the wafers had to be strengthened or they would break during placement into the cell. The un-patterned, but electroplated, wafers had a 6-inch piece of nickel ribbon attached to the back using conductive adhesive to allow connection to the working lead on the potentiostat. Epoxy was used to attach a 1 x 1 inch piece of Plexiglas to the backside of the wafer. A 1-cm² knife-edge gasket was used to define the electrode area exposed to the electrolyte. The attached nickel ribbon was connected to the working lead, a platinum-niobium mesh was connected to the counter lead, and a calomel electrode was used as a reference electrode. Before each test, the open circuit potential was allowed to stabilize. Scans were performed over the range of -0.1 V vs. open circuit potential to 1.0 V (SCE) with a scan rate of 2 mV/sec. Solutions containing 0.5 M or 0.01 M H₂SO₄ with varying NiSO₄ concentrations (between zero and 3.5 M saturation) were used. After the scan was completed, the Ni200 samples were polished to 1200 grit for reuse while the wafers were rotated to an un-corroded region and run again.

Figure 3 shows the experimentally obtained polarization curve from substrate NI-2 with its double bump shape. An E_{crit} of 0.244 V (SCE) was determined from the data to be the beginning of the active nose (corrosion). This curve was fit mathematically to a series of high order polynomial, exponential, and sigmoidal equations using SigmaPlot 2000™ (the overlaid curve in Figure 3). The equations used were:

10th Order Polynomial

$$y = y_0 + ax + bx^2 + cx^3 + dx^4 + ex^5 + gx^6 + hx^7 + ix^8 + jx^9 + kx^{10} + lx^{11} \quad (1)$$

Modified 2 Parameter Exponential Decay

$$y = ae^{\left(\frac{b}{x+c}\right)} \quad (2)$$

3 Parameter Sigmoidal

$$y = \frac{a}{1 + e^{\frac{-(x-x_0)}{b}}} \quad (3)$$

Where:

y is the current density in [A/m²]
 x is the potential in [V vs. SCE]
 $y_0, x_0, a, b, c, d, e, g, h, i, j, k, l$ are fit parameters

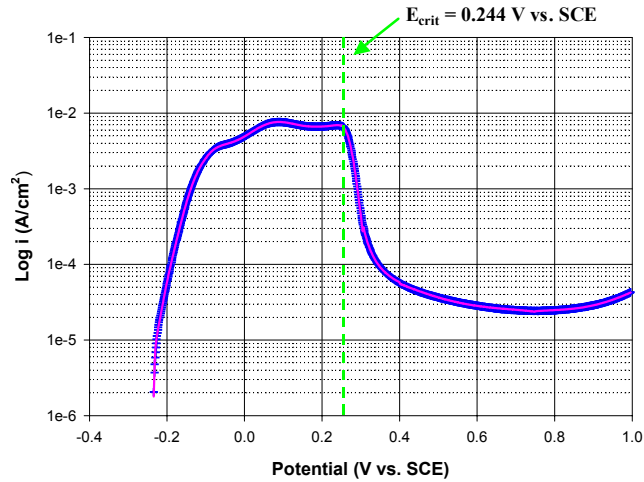


Figure 4.3 Measured electrochemical boundary condition of Ni200 in 0.5 H₂SO₄ (blue) and the corresponding mathematical fit (pink). E_{crit} was determined to be 0.244 V (SCE).

Each region that was fit had to have an R² value > 0.999 to qualify. The fit parameters were coded into CREVICERv2 to define the electrochemical boundary condition and can be found in Table (1).

Table 4.1 Fit parameters used to mathematically describe the polarization curve

Potential Range (V vs. SCE)	> 0.750	0.400 0.750	0.306 0.400	0.250 0.306	-0.100 0.250	-0.235* -0.100
Equation Type (Order)	Poly (1)	Poly (1)	Modified Exponential Decay	Sigmoidal	Poly (2)	Poly (3)
y0	-0.0007	0.0003	-	-	0.0049	0.0938
x0	-	-	-	0.2726	-	-
A	0.0029	-0.0013	2.57358e-5	0.0076	0.034	-3.7514
B	-0.0037	0.0019	0.1126	-0.0098	0.2811	61.1780
C	0.0016	-0.0009	-0.2617	-	-1.6702	453.6162
D	-	-	-	-	-40.2348	874.5814
E	-	-	-	-	100.9172	-7526.3694
G	-	-	-	-	1628.2576	-35029.2485
H	-	-	-	-	-1058.3561	101206.0435
I	-	-	-	-	-38833.3098	1075895.0779
J	-	-	-	-	-73116.8588	2834226.9535
K	-	-	-	-	1166188.391	2585682.6718
L	-	-	-	-	-2152169.6822	-

*Below -0.235 V(SCE), which is E_{corr} , the current density was set to 0.

4.3.2.3 Equipment

All electrochemical experiments were performed using either an EG&G Versastat or an EG&G model 273 potentiostat. A Dell 333 MHz computer controlled the Versastat and the 273. Scribner Associates CorrWare v2.3f electrochemical analysis software was used to run the experiments.

Solution conductivity measurements were made using a YSI Model 3200 conductivity meter with a YSI Model 3252 conductivity cell with a cell constant $K = 1.0 \text{ cm}^{-1}$. Solution pH measurements were taken with a Corning pH/ion analyzer model 350. The analyzer was calibrated using standard, buffered solutions with pH values of 2.0, 3.0, 4.0, and 6.0. Electrolyte surface tension values were taken using a Kruss Tensiometer model K-12 using the platinum plate technique. Electrolyte density measurements were taken by measuring the mass of 1 mL (cm^3) with a Denver Instrument Precision Balance M220D. The electrolyte was drawn using an Eppendorf pipette to ensure accurate solution volume.

4.3.2.4 Crevice assembly experiments

Crevice experiments were performed in 0.5 M H_2SO_4 solution. After each experiment, the cell was emptied, rinsed with de-ionized water, and refilled with fresh acid solution. Formers with 7.3, 28.0, 74.3, and 86.5 μm gap sizes were used. Diced wafer pieces 301.5 μm thick were used in conjunction with the 86.5 μm formers to create extra large gaps of $\sim 400 \mu\text{m}$. The wafer pieces were placed along either side of the substrate electrode and the former was placed on top of these pieces so that the

former's SU-8-covered feet rested on top of the wafer pieces. The SU-8 patterned on the substrates also added $\sim 7 \mu\text{m}$ to the total crevice gap, resulting in final crevice gaps of ~ 14 , 35 , and $93 \mu\text{m}$. Substrates with SU-8-50 spun on them ($\sim 79 \mu\text{m}$) were combined with the $74.3 \mu\text{m}$ formers to create crevice gaps of $\sim 153 \mu\text{m}$. Experiments were run for 1, 5, 10, and 30 minutes with potentiostatic hold potentials in the range of 0.5 to 0.6 V (SCE).

Metallography was performed on a 1 x 1 inch sample of Ni200 plating which had been polished to 1200 grit. The surface was etched using equal parts of concentrated acetic acid, concentrated nitric acid, and distilled water. A Zeiss 510 LSM (laser scanning microscope) and a FEI 200 FIB (focused-ion beam) were used to examine the attack morphology.

4.3.3 CREVICER computational experiments

CREVICER was used to model the initial potential and current distributions down the length of the crevice. The potentiodynamic scan of Ni200 in 0.5 H_2SO_4 described in Section 4.3.3.2 was mathematically fitted and coded into CREVICER. Crevices with gaps of 14, 35, 93, 153, and 395 μm were chosen for direct comparison with results from the microfabricated crevice experiments. The potentiostatic hold potential at the crevice mouth was +600 mV. The pseudo-electrical time step was set to 10^{-7} .

The following steps are summarized.

- A) run the GUI, refer to Figure 4
- B) change setting, refer to Figures 5, 6, 7, 8
- C) run, refer to Figures 9, 10, 11
- D) check calculation result, refer to Figures 12, 13, 14, 16, 16
- E) repeat calculation in other conditions
- F) summary

The above steps are discussed below with the examples.

A) Running GUI shows as shown in Figure 4:

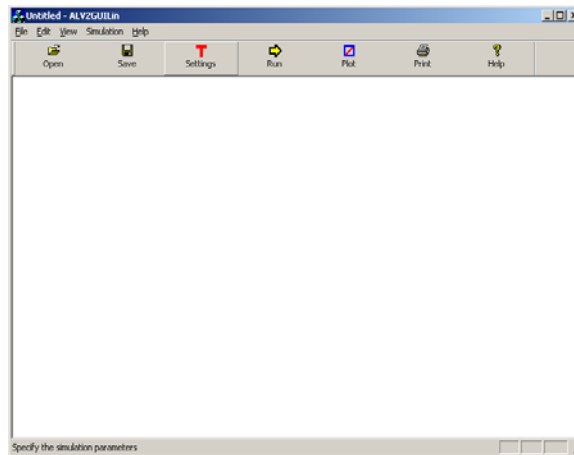


Figure 4.4 The window of CREVICER GUI

B) Changing the setting shows

B1. Figure 5 shows how to select nodes, elements, and crevice gap sizes.

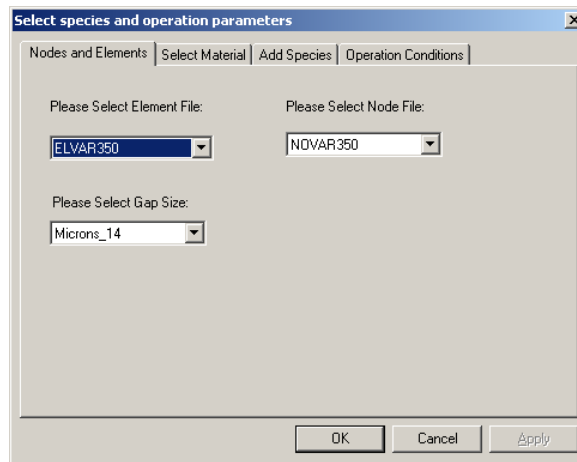


Figure 4.5 The window of CREVICER setting – nodes, elements, and crevice gap sizes

B2. Figure 6 shows how to select materials

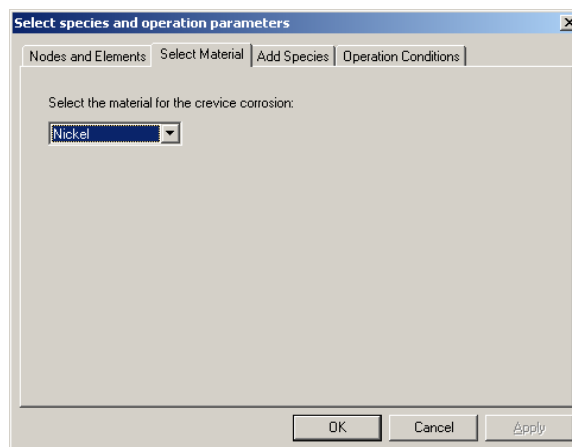


Figure 4.6 The window of CREVICER setting – select material

B3. Add species

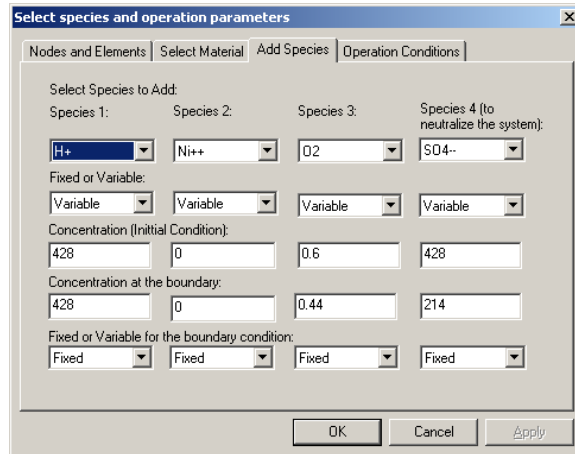


Figure 4.7 The window of CREVICER setting – add species

Figure 7 shows how to change the information for each chemical species. In the first line user can choose four species. The second line defines the initial concentrations of selected each species in the crevice while the third line defines these concentration as variable or fixed parameters. The fourth and fifth lines define the concentration at the crevice mouth and define it fixed or variable parameters, respectively.

B4. Operation conditions

In the first line user can define the initial value for potential, temperature, and pressure in the crevice as shown in Figure 8. The second line defines the above three parameters as fixed or varied one. The third line defines the fixed boundary condition for potential, temperature, and pressure at the crevice mouth.

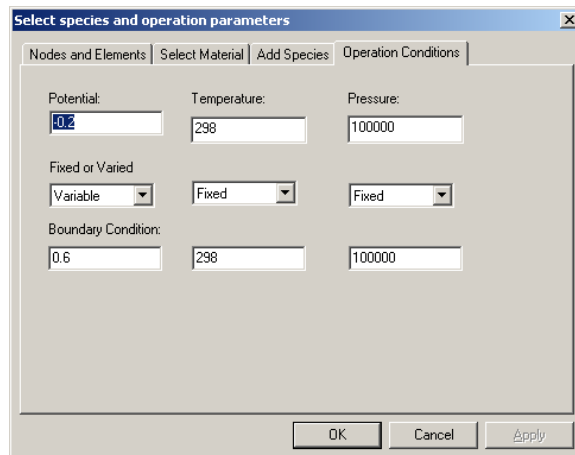


Figure 4.8 The window of CREVICER setting – operation conditions

C. Once you finish changing the setting, click OK to close the setting window. You return to the main window. Now you choose button “Run”. The GUI will run the ALLin.exe, which will read the file <parameters.txt>, carry out the simulation. Refer to the Section 4.2 for the detail information about the parameters.txt file.

C1. When the ALLin.exe starts, you will see Figure 9:

```

C:\Documents and Settings\hw7s\My Documents\Wang\Publish\ALV2GUIin-calculate location\ALLin...
Reading parameters from the input file ...
nTotalSpeciesAdded=4
File name for node:  novar350
File name for element:  elvar350
Selected material for crevice corrosion: Nickel
Electrical Potential:  -0.2
Operation Temperature:  298
Operation Pressure:    100000
Add Specie  .. 11  H+  Conc: 428  Fixed? 0
Add Specie  .. 15  Ni++ Conc: 0    Fixed? 0
Add Specie  .. 17  O2  Conc: 0.55 Fixed? 0
Add Specie  .. 19  SO4-- Conc: 214 Fixed? 0
read files now
Nodes 248
Using boundary conditions for Test1
to main
Time period 1 found
Starting ELECTRICAL
Rt= 0.000153 Rx= 0.00281373 Ry= 0.00281373 Mx= 0 My= 0 electrical
Entering SOLVE
Just back from SOLVE
residual = 2.34892
b0.txt
Entering SOLVE
-
  
```

Figure 4.9 The computational process - beginning

C2. When the ALLin.exe approaches the end of computation, you will see Figure 10:

```

C:\Documents and Settings\hw7s\My Documents\Wang\Publish\ALV2GUIin-calculate location\ALLin...
Just back from SOLVE
residual = 0.00253382
b30.txt
Entering SOLVE
Just back from SOLVE
residual = 0.00241487
b31.txt
Entering SOLVE
Just back from SOLVE
residual = 0.00231846
b32.txt
Entering SOLVE
Just back from SOLVE
residual = 0.00223111
b33.txt
Entering SOLVE
Just back from SOLVE
residual = 0.00215714
b34.txt
Entering SOLVE
Just back from SOLVE
residual = 0.00209478
b35.txt
Entering SOLVE
Just back from SOLVE
residual = 0.00203968
b36.txt
Entering SOLVE
Just back from SOLVE
residual = 0.00199064
Time period 1 electrical found
PotentialI.txt
Starting Ni++
Entering SOLVE
Time period 1 Ni++ found
Ni1.txt
Starting H+
Entering SOLVE
-
  
```

Figure 4.10 The computational process - end

C3. When the simulation is over, you will see a small window to indicate “Simulations Completed” as shown in Figure 11.



Figure 4.11 Finish the simulation

Now click the OK button in the small window. The simulation is over.

D. Analyze the simulation results in these output data files.

In the directory `c:/user/local/CREVICERBASE/`, you can find seven output files. They are `Elements.txt`, `Potential1.txt`, `H+_1.txt`, `N++_1.txt`, `NiFlux_1.txt`, `O2_1.txt`, `SO4-_1.txt` as shown in Figure 12. Detail specifications are provided below.

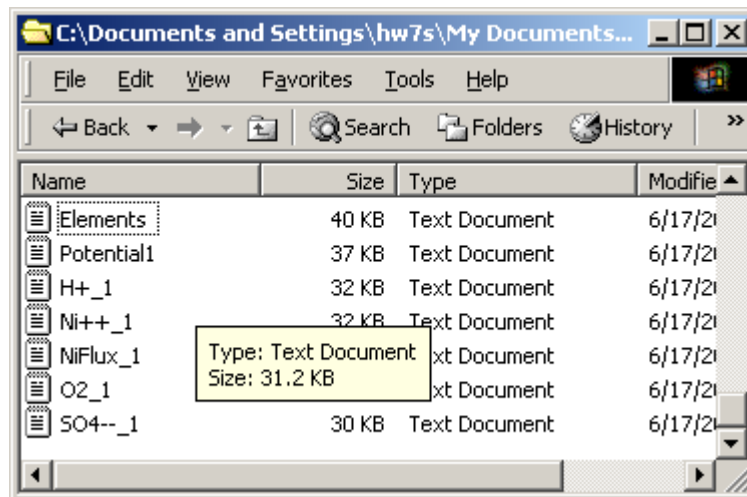


Figure 4.12 Output data files

Elements.txt includes the total element number, their connectivity (a, b, c), position in the crevice length, and crevice gap, e.g. 9, 6, 7, 752, 2.12183e-007, 0.000153 as shown in

Figure 13. The code is `outelem << (elnumber-1) << " " << nodes[i-1]->iNodeId << " "`
`<< nodes[j-1]->iNodeId << " " << nodes[k-1]->iNodeId << " " << avgY << " " <<`
`temp_gap << endl;`

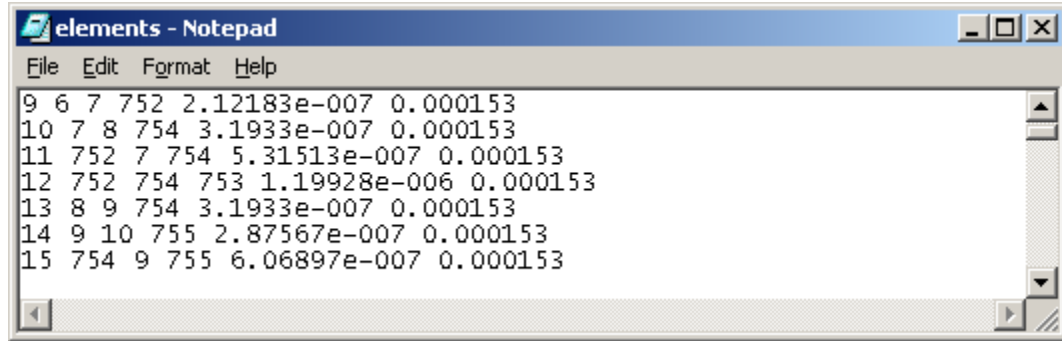


Figure 4.13 Output data file – elements.txt

Potential1.txt includes the potential distribution along the crevice length. Each line gives the relative node number, the node distance, the distance in the crevice from the mouth, the corresponding potential value, and potential value in the previous iteration time as shown in Figure 14. After copying the third and fourth column data to Excel, you can learn the distribution of potential along the crevice.

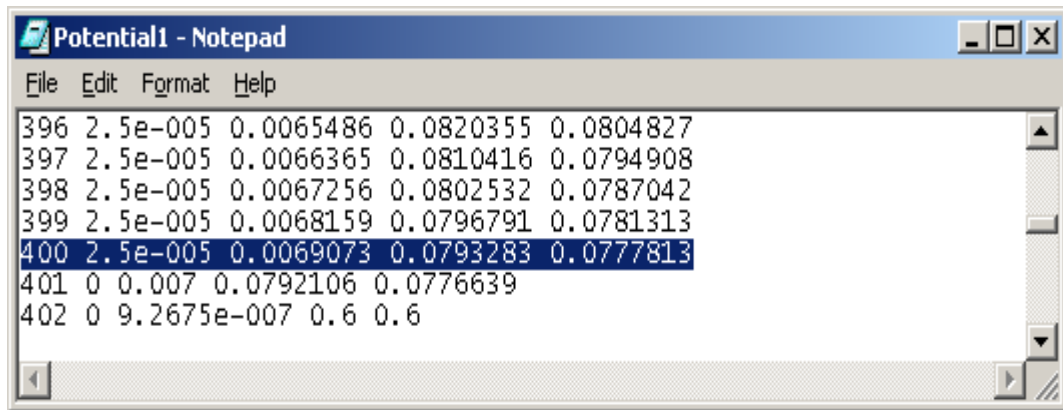
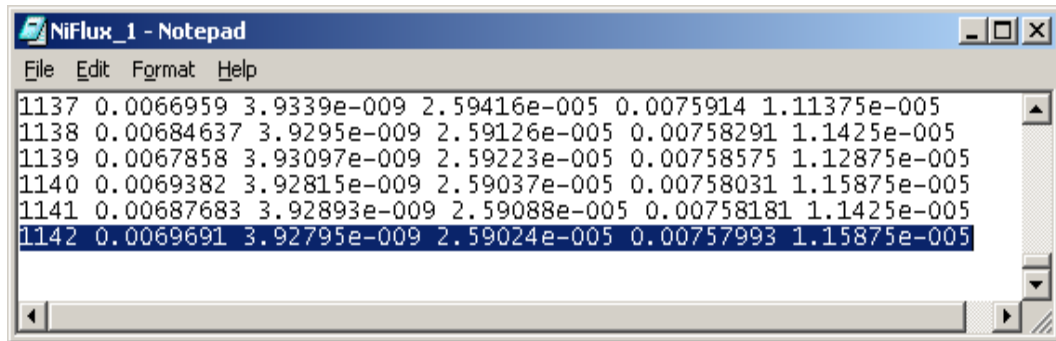


Figure 4.14 Output data file – potential1.txt

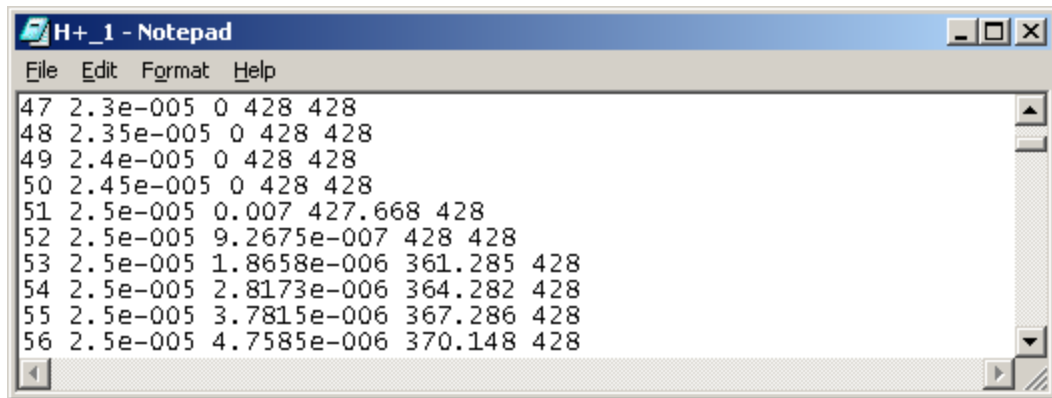
NiFlux_1.txt includes the current density of across section long the crevice. Each includes the element number, the distance from the crevice mouth, the nickel flux, the penetration depth, the current density, and the element area, respectively as shown in Figure 15. Copying these second and fifth column data to Excel, you can learn the distribution of current density in crevice. The code is `outflux << count3 << " " << avgy`
`<< " " << Niflux[count3] << " " << pendepth << " " << newi << " " << el_area << "`
`" << endl;`



```
File Edit Format Help
1137 0.0066959 3.9339e-009 2.59416e-005 0.0075914 1.11375e-005
1138 0.00684637 3.9295e-009 2.59126e-005 0.00758291 1.1425e-005
1139 0.0067858 3.93097e-009 2.59223e-005 0.00758575 1.12875e-005
1140 0.0069382 3.92815e-009 2.59037e-005 0.00758031 1.15875e-005
1141 0.00687683 3.92893e-009 2.59088e-005 0.00758181 1.1425e-005
1142 0.0069691 3.92795e-009 2.59024e-005 0.00757993 1.15875e-005
```

Figure 4.15 Output data file – NiFlux_1.txt

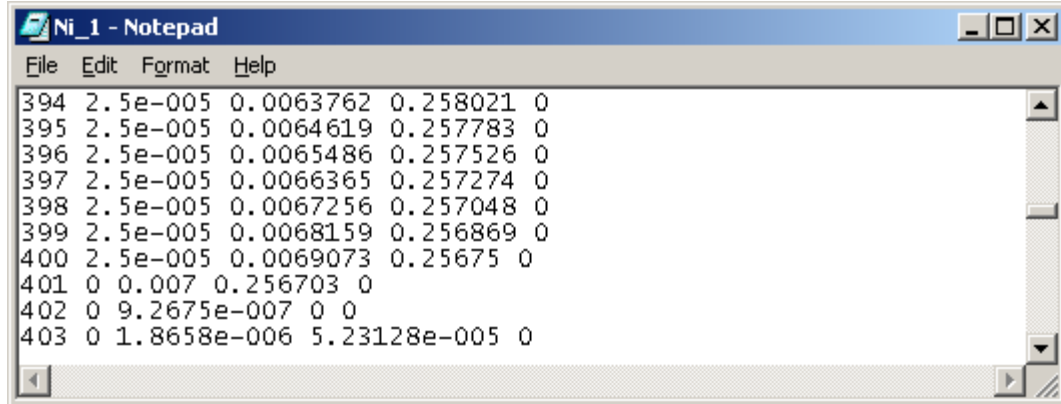
H+ _1.txt includes the hydrogen concentration in the crevice. The Figure 16-19 have the same format. Each line gives the node number (count), the node distance (nodes[count]->dbX), the distance into the crevice from the mouth (nodes[count]->dbY), the new concentration in the node (dbnew), and the old concentration in the node (dbold), respectively. See the code in the Solver.cpp “concout<<count<<" "<<nodes[count]->dbX<<" "<<nodes[count]->dbY<<" "<<dbnew<<" "<<dbold<<endl;”



```
File Edit Format Help
47 2.3e-005 0 428 428
48 2.35e-005 0 428 428
49 2.4e-005 0 428 428
50 2.45e-005 0 428 428
51 2.5e-005 0.007 427.668 428
52 2.5e-005 9.2675e-007 428 428
53 2.5e-005 1.8658e-006 361.285 428
54 2.5e-005 2.8173e-006 364.282 428
55 2.5e-005 3.7815e-006 367.286 428
56 2.5e-005 4.7585e-006 370.148 428
```

Figure 4.16 Output data file H+_1.txt

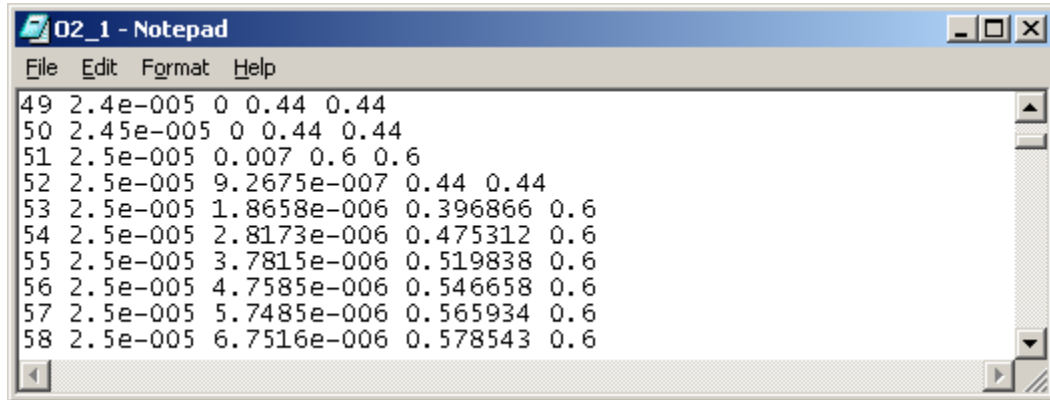
Ni⁺⁺_1.txt includes the nickel concentration in the crevice.



```
File Edit Format Help
394 2.5e-005 0.0063762 0.258021 0
395 2.5e-005 0.0064619 0.257783 0
396 2.5e-005 0.0065486 0.257526 0
397 2.5e-005 0.0066365 0.257274 0
398 2.5e-005 0.0067256 0.257048 0
399 2.5e-005 0.0068159 0.256869 0
400 2.5e-005 0.0069073 0.25675 0
401 0 0.007 0.256703 0
402 0 9.2675e-007 0 0
403 0 1.8658e-006 5.23128e-005 0
```

Figure 4.17 Output data file Ni⁺⁺_1.txt

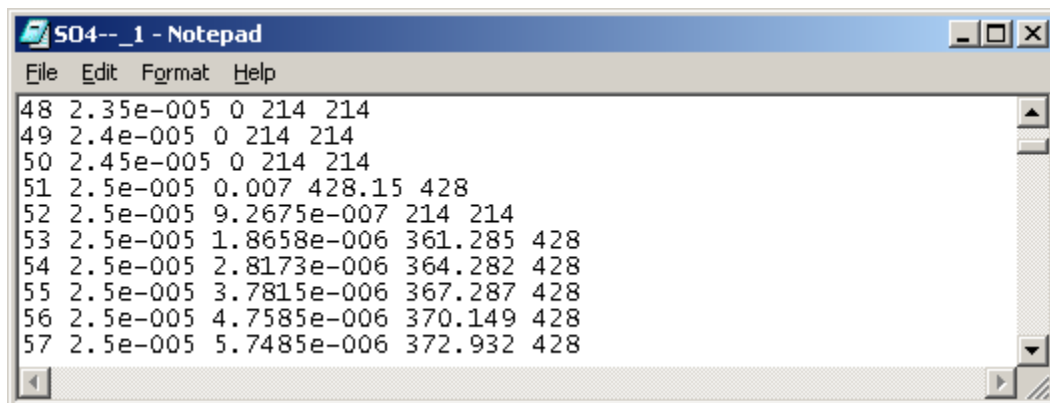
O₂_1.txt includes the oxygen concentration in the crevice.



```
File Edit Format Help
49 2.4e-005 0 0.44 0.44
50 2.45e-005 0 0.44 0.44
51 2.5e-005 0.007 0.6 0.6
52 2.5e-005 9.2675e-007 0.44 0.44
53 2.5e-005 1.8658e-006 0.396866 0.6
54 2.5e-005 2.8173e-006 0.475312 0.6
55 2.5e-005 3.7815e-006 0.519838 0.6
56 2.5e-005 4.7585e-006 0.546658 0.6
57 2.5e-005 5.7485e-006 0.565934 0.6
58 2.5e-005 6.7516e-006 0.578543 0.6
```

Figure 4.18 Output data file H_1.txt

SO₄⁻⁻_1.txt includes the SO₄²⁻ concentration in the crevice.



```
File Edit Format Help
48 2.35e-005 0 214 214
49 2.4e-005 0 214 214
50 2.45e-005 0 214 214
51 2.5e-005 0.007 428.15 428
52 2.5e-005 9.2675e-007 214 214
53 2.5e-005 1.8658e-006 361.285 428
54 2.5e-005 2.8173e-006 364.282 428
55 2.5e-005 3.7815e-006 367.287 428
56 2.5e-005 4.7585e-006 370.149 428
57 2.5e-005 5.7485e-006 372.932 428
```

Figure 4.19 Output data file SO₄⁻⁻_1.txt

E. Repeat the above simulation in the gap size of 14 to 395 μm .

F. Simulation summary

Table 2 list the results from model runs with crevice gaps of 14, 35, 93, 153, and 395 μm . The crevice mouth was held at 0.6 V (SCE). The initial potential and current distributions were the outputs of interest (Figure 20a and b). The plots indicate that x_{crit} moved deeper into the crevice as the gap increased. With a 395 μm gap size, the potential did not drop below E_{crit} (0.244 V (SCE)) in the 7 mm crevice, therefore, active corrosion did not take place within the crevice. This effect is illustrated in Figure 17b by the very low current density observed down the length of the crevice.

Figures 20a and b are plots of x_{crit}^2 vs. gap and x_{crit} vs. gap respectively. The x_{crit}^2 vs. gap plot is linear below gap sizes of $\sim 100 \mu\text{m}$ with an $R^2 = 0.998$. The same region on the x_{crit} vs. gap has a slightly lower R^2 value of 0.983. The scaling factor of DeJong's Skinny boundary condition, which has a much thinner active nose ($\sim 75 \text{ mV}$) than the one used in this model ($\sim 400 \text{ mV}$), was plotted on the same graphs to contrast the difference between the two boundary conditions. The x_{crit}^2 vs. gap plot for the Skinny is even more linear below gap sizes of $\sim 100 \mu\text{m}$ with an $R^2 = 0.9996$. Again, the same region on the x_{crit} vs. gap for the Skinny boundary condition has a slightly lower R^2 value of 0.9875 indicating less linearity.

Table 4.2 Results of crevice holds modeled by CREVICER

Gap (μm)	Hold Potential (V vs SCE)	X_{pass} (mm)	X_{pass}^2 (mm^2)
5	0.6	0.55	0.30
14	0.6	0.83	0.69
25	0.6	1.12	1.26
35	0.6	1.33	1.76
35	0.525	0.86	0.74
35	0.5	0.77	0.59
50	0.6	1.59	2.52
75	0.6	1.98	3.90
93	0.6	2.26	5.12
106	0.6	2.49	6.20
130	0.6	2.89	8.35
153	0.6	3.54	12.51
395	0.6	N/A	N/A

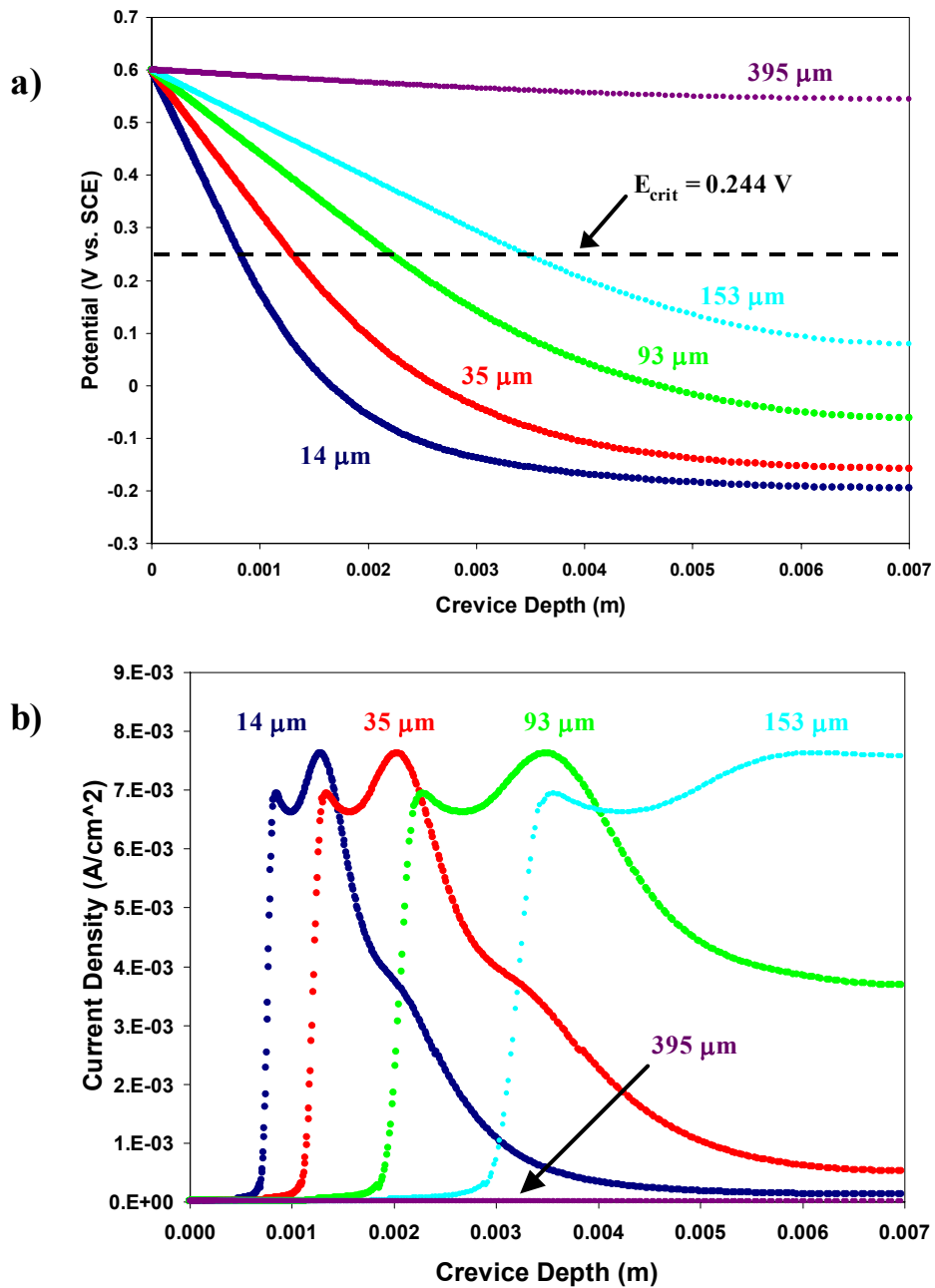


Figure 4.20 (a) Potential distributions from CREVICER for gaps ranging from 14 – 395 μm . The 395 μm gap near reach E_{crit} and passivated at the onset. (b) Corresponding current distributions for each gap size, notice that the 395 μm gap exhibits very low current indicating passive corrosion only.

Chapter 5

Future Improvements to the CREVICER

Before we visualize the future improvements to the CREVICER, let's take a quick look on the developments of CREVICER in the last eight years. According these revisits, the improvements are proposed. Wang wrote this review in November 2001, which did not include the progress in 2002.

5.1 Kevin's wish (1994-1999)

He created the CREVICER to simulate the crevice corrosion. The code is a 2-D(+), temporal FEM code for the generation and transport of electrochemical species inside occluded sites. Its strength includes Object Oriented Design (OOD). He used CREVICER to test Critical Crevice Solution Model (CCS), Critical Potential Drop Model (IR), and Cathodic Focusing (CF) model. He also examined the proposed Intermediate Attack (IA) Model using CREVICER simulator.

He gave some suggestions on the future work on CREVICER as follows.

Code improvement

- GUI accessible for a wide user.
- Improve speed by updating the solver routine, routine for finding the potential distribution in the crevice at the each time period.
- Improve stability by adjusting its own parameters, such as the time step or mesh size.

Fundamental model improvement

- Add other concentration factors. This can even destroy the underlying basis for using gradients of concentration rather than activity in diffusion.
- Consider the time delay caused by removing the protective oxide while CREVICER assumes that behaviors instantly change to reflect new electrochemical conditions. This approach does not fully reflect the physical behavior of real system.
- Recast the mathematical solution technique to allow for the boundaries of the crevice to move based on the corrosion rate. This would allow for a single model to investigate the shape evolution of a crevice.

5.2 DeJong's wish (1997-1999)

She applied the microfabricated nickel crevices for experimental verification of CREVICER and utilized the microfabricated nickel crevices in conjunction with CREVICER to investigate the scaling laws that govern crevice corrosion. Modeling results indicated that x_{crit}^2/G is the correct scaling factor for the system. The above studies are "informative," much work is needed for conclusive quantification of the scaling laws. She also tried to develop CREVICER into a use-friendly tool for crevice corrosion

modeling. Based on the Kevin's work, she improved the CREVICER in the following components.

- A new routine for solving the matrix of equations is used and it increases the speed significantly.
- Expand the applicability of the model to aluminum alloys.
- Add the aluminum hydrolysis reaction.
- GUI is developed. (Visual representation of the crevice geometry and the FEM mesh.

She basically finished the future work on code improvement suggested by Kevin. However, the fundamental model improvements were not available. She gave the following suggestions on the future work.

- Precipitation of a salt film inside the crevice.
- Evolution of the morphology of the crevice.
- Homogeneous reactions other than chromium and aluminum hydrolysis.
- Allow the concentration gradients to extend outside the crevice mouth.
- Continued development of GUI.
- Internet-accessible.

5.3 Lee's wish (1999-2001)

Lee continued the study of the scaling laws of crevice corrosion using the microfabricated nickel crevices in conjunction with CREVICER. His achievements includes:

- Improvements were made to these fabrication techniques that allowed direct comparison with results obtained from CREVICER. E.g. the thickness of metal on the electrode was increased by an order of magnitude to allow for longer experiments to be performed.
- Examined the competing forces of natural convection due to density differences and surface tension during active corrosion. With the decrease of gap size, the role of natural convection decreased, while the surface tension of the crevice solution increased.
- CREVICER was used to predict the initial potential distribution and attack profiles of crevice corrosion experiments that mirrored the ones performed experimentally.

The future work he envisioned includes:

- Perform real time experiment: evolution of chemistry changes within an actively corroding crevice.
- Improve the solution conductivity method. The CREVICER used the dilute solution theory of conductivity, which is not sufficient in accounting for the decrease in conductivity at high solute concentrations.
- Add the viscosity into the calculation of conductivity.

5.4 Wang' wish for new developments (2001-2003)

Wang reviewed the code improvement and fundamental model development of CREVICER based on the earlier achievements and suggestions as follows:

A) Code improvement

- Chang Lin's continued work on GUI. An important progress in GUI development, the first time to connect the GUI and calculation.
- A team is needed to have a professional GUI software development evaluation and setup practical and tangible targets in short-term and long-term.
- A team will work together to accomplish the commercial software development based on the development goals.

B) Fundamental model improvement—two directions are found as follows

Include other time dependents

- Evolution of potential, current, and chemistry changes within an actively corroding crevice.
- Evolution of the morphology of the crevice, the precipitation of a salt film inside the crevice.
- Consider the time delay caused by removing the protective oxide while CREVICER assumes that behaviors instantly change to reflect new electrochemical conditions. This approach does not fully reflect the physical behavior of real system.
- Recast the mathematical solution technique to allow for the boundaries of the crevice to move based on the corrosion rate. This would allow for a single model to investigate the shape evolution of a crevice.

Modify the physical and chemical parameters to fit the reality better

- Allow the concentration gradients to extend outside the crevice mouth.
- Improve the solution conductivity method. The CREVICER used the dilute solution theory of conductivity, which is not sufficient in accounting for the decrease in conductivity at high solute concentrations.
- Add the viscosity into the calculation of conductivity.
- Homogeneous reactions other than chromium and aluminum hydrolysis.

Chapter 6 Reference

Publications

K.C. Stewart, Ph.D. Dissertation, Intermediate Attack in Crevice Corrosion by Cathodic Focusing, University of Virginia, Charlottesville, VA, 1999

L.A. DeJong, M.S. Thesis, Investigation of Crevice Corrosion Scaling Laws Using Microfabrication Techniques and Modeling, University of Virginia, Charlottesville, VA, 1999

J. S. Lee, M.S. Thesis, Investigation of Crevice Corrosion Using Computational Modeling and Microfabrication Techniques, University of Virginia, Charlottesville, VA, 2001

R. G. Kelly, K. C. Stewart, *Passivity of Metals and Semiconductors VIII*, M. B. Ives, B. R. MacDougall, J. A. Bardwell, eds. PV 99-42, The Electrochemical Society, Princeton, p. 546-554 (1999)

L. A. DeJong, R. G. Kelly, "The Demonstration of the Microfabrication of Rigorously Defined Crevices for the Investigation of Crevice Corrosion Scaling Laws," in *Critical Factors in Localized Corrosion III*, R. G. Kelly, G. S. Frankel, P. M. Natishan, R. C. Newman, eds., PV 98-17, The Electrochemical Society, Inc., Pennington, NJ, pp. 678-89 (1999)

J. S. Lee, M. L. Reed, R. G. Kelly, in *Corrosion and Corrosion Protection*, J. D. Sinclair, E. Kalman, M. W. Kendig, W. Plieth, W. H. Smyrl, PV 2001- 22, The Electrochemical Society, Princeton, pp. 279-90 (2001)

C. DiBona, S. Ockman, and M. Stone, *Open sources: voices from the open source revolution*, Beijing; Sebastopol, O'Reilly Publisher (1999)

Websites

Flightgear, an open-source, multi-platform, cooperative flight simulator development project, <http://www.flightgear.org/>

Free Software Foundation, <http://www.fsf.org/>

GNU Project Web Server, <http://www.gnu.org/>

Mozilla, an open-source web browser, <http://www.mozilla.org/>

Open Source Initiative (OSI), <http://www.opensource.org/>

Part III
Appendices

Appendix A

Missed approach: If anything refuses to work

In the following section, we tried to sort out some problems according to operating system, but if you encounter a problem, it may be a wise idea to look beyond “your” operating system - just in case.

A.1 CREVICER problem reports

The best place to look for help is generally the mailing lists, specifically the CREVICER mailing list. Instructions for subscription can be found at

<http://www.virginia.edu/%7Ecese/research/crevicer/contactinformation.html>

There are numerous developers and users reading the lists, so questions are generally answered. However, messages of the type,

“CREVICER does not compile on my system. What shall I do?”

are hard to answer without any further detail given. Here are some things to consider including in your message when you report a problem:

Operating system: (/Windows 98SE. . .)

Computer: (Pentium III, 1GHz. . .)

A.2 General problems

CREVICER runs SOOO slow.

Check if your computer meets the requirements.

A.3 Potential problems under Windows

The executable refuses to run.

You may have tried to start the executable directly by invoking it within a MS-DOS shell. Another cause of grief might be that you did not download the most recent versions of the base package files (initial parameter, node and element file) required by CREVICERGUI and CREVICERSOLVER, or you did not download any of them at all. Have a close look at this. For more details, check Chapter 3. In principle, it should be possible to compile CREVICER with the project files provided with the source code.

Appendix B

Finishing: Some further thoughts before leaving the crevice

B.1 A not so short history of CREVICER

The crevice computational project goes back to 1994 resulting in a NSF proposal written by Robert G. Kelly and Michael L. Reed. The original proposal is still available from the CREVICER web site and can be found under

<http://www.virginia.edu/%7Ecese/research/crevicer/designdocumentation.html>

This proposal has produced one Ph.D. student, three M.S. students. After seven years from 1994 to 2001, the CREVICER possesses its mature shape.

Actual coding started in the 1994. At that time, programming was mainly performed and coordinated by Kevin Stewart, who was a Ph.D. student with Professor Kelly at University of Virginia. Early code ran under Windows 95/NT/2000. This was found to be quite an ambitious project as it involved entirely from scratch.

At the meantime, Lisa DeJong joined the team in 1997. At the beginning, she worked on the experimental crevice work using microfabrication technique (Refer to thesis). According to experimental work, she upgraded the FEM solver code by a better algorithm with 10 times increases in computational speed.

In the summer of 1999, two other team members Koehl and Switzer developed the first GUI for the initial CREVICER. However, the connection between the GUI and Crevice solver was missed. Although the applicable GUI was not successfully accomplished, it is a good try.

Since fall 1999 Jason S. Lee continued the crevice experimental work for a better crevice dimension by improved microfabrication technique. Some cases studies were added to the CREVICER code.

It was Dr. Changqing Lin, a postdoc with Professor Kelly, who continued the GUI development in the August of 2001. His idea was as simple as it was powerful: develop the first useful GUI to access the CREVICERSOLVER. After three months development, his GUI works for a simple case, tested in Jason's experimental result. However, most fundamental issues, such as more adjustable parameters in GUI, still remain. A fully satisfactory GUI for CREVICER is expected.

This new released CREVICER code - still largely being based on the original GUI by Dr. Lin was further edited by Hongwei (David) Wang, a postdoctoral research associate with Robert Kelly. The edited code was released in July 2002, which is the first released version. He edited the most of the early work and published the first released CREVICER version 1.0.

This is by no means an exhaustive history and most likely some people who have made important contributions have been left out. Besides the above-named contributions there was a lot of work done concerning the boundary work by: Kelvin Cooper and others. A more comprehensive list of contributors can be found in Appendix B2. Also, the CREVICER website contains a detailed history worth reading of all of the notable development milestones.

B.2 Those, who did the work

Did you enjoy the crevice corrosion simulation? In case you did, don't forget those who devoted hundreds of hours to that project. All of this work is done at the Center for Electrochemical Science and Engineering until it is released to public. Now, you can join the development team voluntarily and subscribe to the CREVICER mailing lists and contribute your thoughts there. Instructions to do so can be found at

<http://www.virginia.edu/%7Ecese/research/crevicer/contactinformation.html>

The following names the people who did the job (this information was essentially taken from the file Acknowledgement accompanying the code). We will keep the future contributor's name here also.

Robert G. Kelly <rgkelly@virginia.edu>
<http://www.virginia.edu/%7Ecese/people/kelly.html>
Professor Kelly is the director of this project since its conception in 1994.

Professor M. L. Reed <reed@virginia.edu>
<http://www.ee.virginia.edu/people/faculty/reed.html>
He has contributed in the microfabrication of practical crevice corrosion samples.

Kevin C. Stewart
Dr. Kevin wrote the framework code that is CREVICER. His Ph.D. dissertation entitled by Intermediate Attack in Crevice Corrosion by Cathodic Focusing provides the solid foundation for this code. He worked on this project during 1994-1999. Currently he is President of Stewart Technologies in Richmond, Virginia.

Lisa A. DeJong
She was the second developer who improved the solver of CREVICER and also initiated crevice corrosion experiments using microfabrication techniques. She also added the aluminum hydrolysis reactions to the code. Her thesis is entitled by Investigation of Crevice Corrosion Scaling Laws Using Microfabrication Techniques and Modeling. She worked on this project from 1997-1999. She is currently working on Wall Street.

Jason S. Lee <jason.lee@nrlssc.navy.mil>

Jason was the third developer in Professor Kelly group at UVa. He extended the experimental work in particular, allowing direct comparisons of measurement and modeling of crevice corrosion for the first time on crevices of practical dimensions. His thesis is entitled by Investigation of Crevice Corrosion Using Computational Modeling and Microfabrication Techniques. Currently he is a research scientist in the Naval Research Laboratory in Stennis Space Center in Mississippi.

Dr. Changqing Lin <changqinglin@netscape.net>

Although he worked on the project for a short time, he successfully developed the first functional GUI for CREVICER. The simple GUI only allows changes to a small portion of the parameters of the typical crevice corrosion systems. It is a good start for a fully functional GUI for CREVICER. CREVICER version 1.0 contains his GUI. The new GUI of CREVICER will be developed soon.

Dr. Hongwei (David) Wang <hw7s@virginia.edu >

He is a postdoctoral research associate with Rob Kelly since October 2001. He optimized the work by Dr. Lin and has added more FEM meshes, allowing CREVICER to be applied to a wider variety of conditions. He prepared the manual for the CREVICER 1.0 released in 2002 summer. Currently he is working on the extension of CREVICER 1.0 to the inhibitor release, transport, and corrosion inhibition in coating systems. His work will be added to the future CREVICER.

Dr. Francisco J. Presuel-Moreno <fp8t@virginia.edu >

He is a postdoctoral research associate with Rob Kelly since Spring 2002. He is working on the extension of CREVICER to the throwing power of cladding system and hydrogen cracking. His work will be also added to the future CREVICER.

B.3 What remains to be done

If you read (and, maybe, followed) this guide up to this point you may probably agree: CREVICER, even in its present state, is not at all for the gamer. It is already a crevice corrosion simulator which tests several selectable crevice corrosion conditions. According this work, the scaling laws of crevice corrosion theory has been further tested (refer to the paper by Jason and Kelly in 2001).

CREVICER needs - and gets - further development. There are several fields where CREVICER needs basic improvement and development. A first direction is adding a flexible FEM meshing tool. Another task is further implementation of the menu system. A lot of options at present set during compile time should be finally moved into menu entries. Finally, CREVICER will expand to more crevice corrosion conditions, including inhibitor in coating, cladding, hydrogen cracking, and etc. There are already people working in all of these directions. If you're working in the similar research field and think you can contribute, you are invited to join with us to develop the CREVICER.

B.4 Acknowledgements

Obviously this document could not have been written without all those contributors mentioned above making CREVICER a reality.

First, Changqing Lin wrote an early version of the guide for CREVICERGUI.

Secondly I gained a lot of help and support from Jason S. Lee. Maybe, without Jason's answers I would have never been able to tame different versions of the CREVICER, FEM, and experimental data. Further contributions came from Kevin Cooper for the Excel calculation of crevice corrosion.

Special thanks also include:

Front photo, courtesy of the Corrosion Laboratories at Kennedy Space Center

Manual writing, courtesies of www.flightgear.org, www.opensource.org, www.fsf.org,

Finally I would like to say special thanks to Jeannie Reese, office manager of CESE, who edited the web site of CREVICER.

Appendix C

GNU General Public License

C.1 GNU general public license

GNU GENERAL PUBLIC LICENSE Version 2, June 1991

Copyright (C) 1989, 1991 Free Software Foundation, Inc.
59 Temple Place, Suite 330, Boston, MA 02111-1307 USA
Everyone is permitted to copy and distribute verbatim copies
of this license document, but changing it is not allowed.

Preamble

The licenses for most software are designed to take away your freedom to share and change it. By contrast, the GNU General Public License is intended to guarantee your freedom to share and change free software--to make sure the software is free for all its users. This General Public License applies to most of the Free Software Foundation's software and to any other program whose authors commit to using it. (Some other Free Software Foundation software is covered by the GNU Library General Public License instead.) You can apply it to your programs, too.

When we speak of free software, we are referring to freedom, not price. Our General Public Licenses are designed to make sure that you have the freedom to distribute copies of free software (and charge for this service if you wish), that you receive source code or can get it if you want it, that you can change the software or use pieces of it in new free programs; and that you know you can do these things.

To protect your rights, we need to make restrictions that forbid anyone to deny you these rights or to ask you to surrender the rights. These restrictions translate to certain responsibilities for you if you distribute copies of the software, or if you modify it.

For example, if you distribute copies of such a program, whether gratis or for a fee, you must give the recipients all the rights that you have. You must make sure that they, too, receive or can get the source code. And you must show them these terms so they know their rights.

We protect your rights with two steps: (1) copyright the software, and (2) offer you this license which gives you legal permission to copy, distribute and/or modify the software.

Also, for each author's protection and ours, we want to make certain that everyone understands that there is no warranty for this free software. If the software is modified by someone else and passed on, we want its recipients to know that what they have is not the original, so that any problems introduced by others will not reflect on the original authors' reputations.

Finally, any free program is threatened constantly by software patents. We wish to avoid the danger that redistributors of a free program will individually obtain patent licenses, in effect making the program proprietary. To prevent this, we have made it clear that any patent must be licensed for everyone's free use or not licensed at all.

The precise terms and conditions for copying, distribution and modification follow.

GNU GENERAL PUBLIC LICENSE TERMS AND CONDITIONS FOR COPYING, DISTRIBUTION AND MODIFICATION

0. This License applies to any program or other work which contains a notice placed by the copyright holder saying it may be distributed under the terms of this General Public License. The "Program", below, refers to any such program or work, and a "work based on the Program" means either the Program or any derivative work under copyright law: that is to say, a work containing the Program or a portion of it, either verbatim or with modifications and/or translated into another language. (Hereinafter, translation is included without limitation in the term "modification".) Each licensee is addressed as "you".

Activities other than copying, distribution and modification are not covered by this License; they are outside its scope. The act of running the Program is not restricted, and the output from the Program is covered only if its contents constitute a work based on the Program (independent of having been made by running the Program). Whether that is true depends on what the Program does.

1. You may copy and distribute verbatim copies of the Program's source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice and disclaimer of warranty; keep intact all the notices that refer to this License and to the absence of any warranty; and give any other recipients of the Program a copy of this License along with the Program.

You may charge a fee for the physical act of transferring a copy, and you may at your option offer warranty protection in exchange for a fee.

2. You may modify your copy or copies of the Program or any portion of it, thus forming a work based on the Program, and copy and distribute such modifications or

work under the terms of Section 1 above, provided that you also meet all of these conditions:

a) You must cause the modified files to carry prominent notices stating that you changed the files and the date of any change.

b) You must cause any work that you distribute or publish, that in whole or in part contains or is derived from the Program or any part thereof, to be licensed as a whole at no charge to all third parties under the terms of this License.

c) If the modified program normally reads commands interactively when run, you must cause it, when started running for such interactive use in the most ordinary way, to print or display an announcement including an appropriate copyright notice and a notice that there is no warranty (or else, saying that you provide a warranty) and that users may redistribute the program under these conditions, and telling the user how to view a copy of this License. (Exception: if the Program itself is interactive but does not normally print such an announcement, your work based on the Program is not required to print an announcement.)

These requirements apply to the modified work as a whole. If identifiable sections of that work are not derived from the Program, and can be reasonably considered independent and separate works in themselves, then this License, and its terms, do not apply to those sections when you distribute them as separate works. But when you distribute the same sections as part of a whole which is a work based on the Program, the distribution of the whole must be on the terms of this License, whose permissions for other licensees extend to the entire whole, and thus to each and every part regardless of who wrote it.

Thus, it is not the intent of this section to claim rights or contest your rights to work written entirely by you; rather, the intent is to exercise the right to control the distribution of derivative or collective works based on the Program.

In addition, mere aggregation of another work not based on the Program with the Program (or with a work based on the Program) on a volume of a storage or distribution medium does not bring the other work under the scope of this License.

3. You may copy and distribute the Program (or a work based on it, under Section 2) in object code or executable form under the terms of Sections 1 and 2 above provided that you also do one of the following:

a) Accompany it with the complete corresponding machine-readable source code, which must be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange; or,

b) Accompany it with a written offer, valid for at least three years, to give any third party, for a charge no more than your cost of physically performing source

distribution, a complete machine-readable copy of the corresponding source code, to be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange; or,

c) Accompany it with the information you received as to the offer to distribute corresponding source code. (This alternative is allowed only for noncommercial distribution and only if you received the program in object code or executable form with such an offer, in accord with Subsection b above.)

The source code for a work means the preferred form of the work for making modifications to it. For an executable work, complete source code means all the source code for all modules it contains, plus any associated interface definition files, plus the scripts used to control compilation and installation of the executable. However, as a special exception, the source code distributed need not include anything that is normally distributed (in either source or binary form) with the major components (compiler, kernel, and so on) of the operating system on which the executable runs, unless that component itself accompanies the executable.

If distribution of executable or object code is made by offering access to copy from a designated place, then offering equivalent access to copy the source code from the same place counts as distribution of the source code, even though third parties are not compelled to copy the source along with the object code.

4. You may not copy, modify, sublicense, or distribute the Program except as expressly provided under this License. Any attempt otherwise to copy, modify, sublicense or distribute the Program is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

5. You are not required to accept this License, since you have not signed it. However, nothing else grants you permission to modify or distribute the Program or its derivative works. These actions are prohibited by law if you do not accept this License. Therefore, by modifying or distributing the Program (or any work based on the Program), you indicate your acceptance of this License to do so, and all its terms and conditions for copying, distributing or modifying the Program or works based on it.

6. Each time you redistribute the Program (or any work based on the Program), the recipient automatically receives a license from the original licensor to copy, distribute or modify the Program subject to these terms and conditions. You may not impose any further restrictions on the recipients' exercise of the rights granted herein. You are not responsible for enforcing compliance by third parties to this License.

7. If, as a consequence of a court judgment or allegation of patent infringement or for any other reason (not limited to patent issues), conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this

License, they do not excuse you from the conditions of this License. If you cannot distribute so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not distribute the Program at all. For example, if a patent license would not permit royalty-free redistribution of the Program by all those who receive copies directly or indirectly through you, then the only way you could satisfy both it and this License would be to refrain entirely from distribution of the Program.

If any portion of this section is held invalid or unenforceable under any particular circumstance, the balance of the section is intended to apply and the section as a whole is intended to apply in other circumstances.

It is not the purpose of this section to induce you to infringe any patents or other property right claims or to contest validity of any such claims; this section has the sole purpose of protecting the integrity of the free software distribution system, which is implemented by public license practices. Many people have made generous contributions to the wide range of software distributed through that system in reliance on consistent application of that system; it is up to the author/donor to decide if he or she is willing to distribute software through any other system and a licensee cannot impose that choice.

This section is intended to make thoroughly clear what is believed to be a consequence of the rest of this License.

8. If the distribution and/or use of the Program is restricted in certain countries either by patents or by copyrighted interfaces, the original copyright holder who places the Program under this License may add an explicit geographical distribution limitation excluding those countries, so that distribution is permitted only in or among countries not thus excluded. In such case, this License incorporates the limitation as if written in the body of this License.

9. The Free Software Foundation may publish revised and/or new versions of the General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

Each version is given a distinguishing version number. If the Program specifies a version number of this License which applies to it and "any later version", you have the option of following the terms and conditions either of that version or of any later version published by the Free Software Foundation. If the Program does not specify a version number of this License, you may choose any version ever published by the Free Software Foundation.

10. If you wish to incorporate parts of the Program into other free programs whose distribution conditions are different, write to the author to ask for permission. For software which is copyrighted by the Free Software Foundation, write to the Free Software Foundation; we sometimes make exceptions for this. Our decision will be

guided by the two goals of preserving the free status of all derivatives of our free software and of promoting the sharing and reuse of software generally.

NO WARRANTY

11. BECAUSE THE PROGRAM IS LICENSED FREE OF CHARGE, THERE IS NO WARRANTY FOR THE PROGRAM, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE PROGRAM "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE PROGRAM IS WITH YOU. SHOULD THE PROGRAM PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

12. IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MAY MODIFY AND/OR REDISTRIBUTE THE PROGRAM AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE PROGRAM (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE PROGRAM TO OPERATE WITH ANY OTHER PROGRAMS), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

END OF TERMS AND CONDITIONS

C.2 How to apply these terms to your new programs

If you develop a new program, and you want it to be of the greatest possible use to the public, the best way to achieve this is to make it free software which everyone can redistribute and change under these terms.

To do so, attach the following notices to the program. It is safest to attach them to the start of each source file to most effectively convey the exclusion of warranty; and each file should have at least the "copyright" line and a pointer to where the full notice is found.

```
<one line to give the program's name and a brief idea of what it does.>  
Copyright (C) <year> <name of author>
```

This program is free software; you can redistribute it and/or modify

it under the terms of the GNU General Public License as published by the Free Software Foundation; either version 2 of the License, or (at your option) any later version.

This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with this program; if not, write to the Free Software Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA 02111-1307 USA

Also add information on how to contact you by electronic and paper mail.

If the program is interactive, make it output a short notice like this when it starts in an interactive mode:

```
Gnomovision version 69, Copyright (C) year name of author
Gnomovision comes with ABSOLUTELY NO WARRANTY; for details type `show
w'.
```

```
This is free software, and you are welcome to redistribute it
under certain conditions; type `show c' for details.
```

The hypothetical commands `show w' and `show c' should show the appropriate parts of the General Public License. Of course, the commands you use may be called something other than `show w' and `show c'; they could even be mouse-clicks or menu items--whatever suits your program.

You should also get your employer (if you work as a programmer) or your school, if any, to sign a "copyright disclaimer" for the program, if necessary. Here is a sample; alter the names:

```
Yoyodyne, Inc., hereby disclaims all copyright interest in the program
`Gnomovision' (which makes passes at compilers) written by James Hacker.
```

```
<signature of Ty Coon>, 1 April 1989
Ty Coon, President of Vice
```

This General Public License does not permit incorporating your program into proprietary programs. If your program is a subroutine library, you may consider it more useful to permit linking proprietary applications with the library. If this is what you want to do, use the GNU Library General Public License instead of this License.