Waste Form Degradation Model Status Report: ANL Mixed Potential Model, Version 1. Archive

Used Fuel Disposition – Engineered Barrier Systems

Prepared for
U.S. Department of Energy
Used Fuel Disposition Campaign
James Jerden
Kurt Frey
Terry Cruse
William Ebert
Argonne National Laboratory

December 20, 2012
FCRD-UFD-2013-000057

This work was supported by the US Department of Energy, Office of Nuclear Energy. The report was prepared at Argonne National Laboratory as part of the Used Fuel Disposition (UFD) Generic Engineered Barrier Systems (EBS) Evaluations [National Technical Director Peter Swift (SNL)]. This report is submitted for milestone M3FT-13AN0806013 for the work package: FT-13AN080601.

This work was supported by the U.S. Department of Energy, Office of Nuclear Energy, under Contract DE-AC02-06CH11357.

SUMMARY

This work is being performed as part of the DOE NE Used Fuel Disposition (UFD) Campaign's Engineered Barrier Systems (EBS) Evaluations, work package: FT-12AN080601. This document represents the December, 2012 milestone report: M3FT-13AN0806013.

The purpose of this work is to develop and optimize a predictive model for the degradation of used uranium oxide fuel that is based on fundamental electrochemical and thermodynamic principals. This process model will provide source terms for radionuclide release the UFD generic performance assessment model. The approach being followed is to implement an existing, well tested electrochemical corrosion model and then extend it to include specific reactions affecting the fuel degradation processes under the disposal scenarios of interest.

This report presents the details of the first version of our UO_2 fuel matrix degradation model. The model described was produced by implementing the Canadian-mixed potential model for UO_2 fuel dissolution (King and Kolar, 1999, King and Kolar, 2003, Shoesmith et.al., 2003) using the numerical computing environment and programming language MATLAB (release R2012a on 64-bit platform). The intent of the initial version was to reproduce the approach used by King and Kolar, 1999 to gain experience with the model, identify key model parameters to be measured experimentally or determined from evaluation of the literature, and provide a touchstone for future modifications of the code.

The initial version of the MATLAB implementation of mixed potential model, referred to as the ANL-MPM Version 1, was verified by reproducing published results from the Canadian-MPM reports (Jerden et al., 2012). This report provides a detailed discussion of the actual coding that was used in the implementation of ANL-MPM Version 1.

The ANL-MPM Version 1 (MATLAB implementation of the Canadian-MPM described by King and Kolar, 1999) is a 1-dimensional reaction-diffusion model that accounts for the following processes:

- Interfacial redox reaction kinetics influencing oxidative dissolution of the UO_2 matrix.
- Chemical or solubility based dissolution of the fuel matrix.
- Complexation of dissolved uranium by carbonate near the fuel surface and in the bulk solution.
- Production of hydrogen peroxide (which is the dominant fuel oxidant in anoxic repository environments) by alpha-radiolysis.
- Diffusion of reactants and products in the groundwater away from and towards the reacting fuel surface.
- Precipitation and dissolution of a U–bearing corrosion product layer on the fuel surface.
- Adsorption of uranium onto iron oxides.
- Arrhenius-type temperature dependence for all interfacial and bulk reactions.

The ANL-MPM Version 1 will be used as a base-line check for future versions of the ANL-MPM. A working beta version of the ANL-MPM Version 2 has been implemented and is now

being tested and optimized. The ANL-MPM Version 2 accounts for the following processes and conditions (in addition to processes listed above):

- Quantifies the oxidation of dissolved H_2 at the used fuel/solution interface: H_2 concentration to be supplied by other EBS model or user specified).
- Represents the NMPs as a separate domain at the used fuel/solution interface. The "size" of the NMP domain (relative to the fuel) is specified by the user in terms of a surface coverage and is electrically linked with the UO_2 matrix by a user adjustable resistance. This will allow the effects of NMP corrosion and sorption on the catalytic efficiency to be taken into account.
- Quantifies the bulk decomposition of hydrogen peroxide (with temperature dependence).
- Provides option for user to specify temperature and dose profiles of the fuel (profiles can be constant single values or functions).
- Includes Rapid diffusion option to facilitate the calculation of concentrations of species whose diffusion coefficients are sufficiently large that they reach steady state on the order of days (decreases computer time needed for model convergence).

This report focuses on the base-line model ANL-MPM Version 1; for a discussion of the conceptual basis for the ANL-MPM Version 2 see Jerden et al., 2012.

Because the ANL-MPM is based on fundamental principles, it is flexible enough to be applied to the full range of repository environments as well as shorter-term storage scenarios being considered as part of the UFD campaign. On-going experimental work described in Jerden et al., 2012 and Ebert et al., 2012 is focused on providing key model parameter values that are needed to improve predictive accuracy and capabilities of the ANL-MPM.

CONTENTS

FIGURES

TABLES

ACRONYMS

ANL	Argonne National Laboratory
DOE NE	Department of Energy Nuclear Energy
EBS	Engineered Barrier System
GPAM	Generic Performance Assessment Model
GSPM	Generic System Performance Model
MPM	Mixed Potential Model
NMP	Noble Metal bearing Particle
UFD	Used Fuel Disposition

ANL Mixed Potential Model Version 1 Archive

1. INTRODUCTION AND BACKGROUND

This work was performed as part of the DOE NE Used Fuel Disposition (UFD) Campaign's Engineered Barrier Systems (EBS) Evaluations, work package: FT-13AN080601. This document represents the December 20, 2012 milestone report: M3FT-13AN0806013.

1.1. Objectives and Scope

The main purpose of this work is to develop and optimize a predictive model for the degradation of used uranium oxide fuel that is based on fundamental electrochemical and thermodynamic principals. This process model will be integrated with the UFD generic performance assessment model to provide a source term for radionuclide release for a disposal scenario of interest. The approach has been to implement, optimize and extend an existing, well tested electrochemical corrosion model to specific fuel degradation processes and mechanisms of interest. The conceptual context for our modeling work within the larger UFD research campaign is shown in Figure 1.

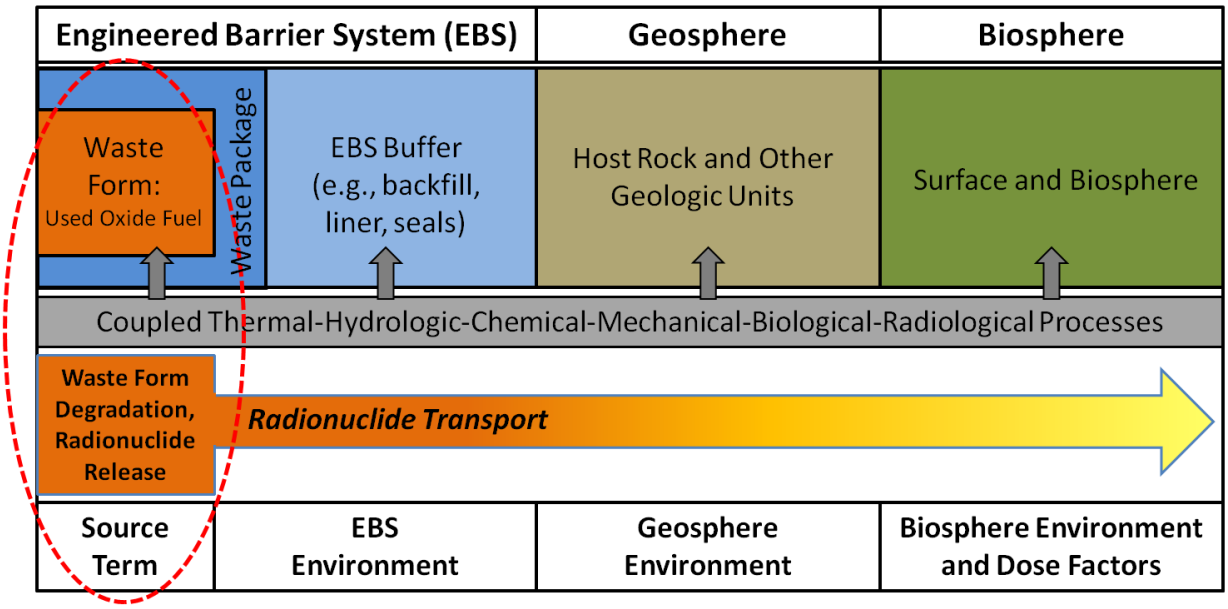

Figure 1. Components of a generic disposal system for used oxide fuel (adapted from Freeze et al. 2010). The red circle identifies the components and associated processes targeted by the experiments described in this report.

This report presents the details of the first version of our UO_2 fuel matrix degradation model. The model described was produced by implementing the Canadian-mixed potential model for UO_2 fuel dissolution (King and Kolar, 1999, King and Kolar, 2003, Shoesmith et.al., 2003) using the numerical computing environment and programming language MATLAB. MATLAB release R2012a, for a 64-bit platform was used to implement the ANL-MPM Version 1.

This work has provided a working, base-line model that is being modified and extended to include additional reactions and factors affecting fuel dissolution. The initial version of the MATLAB implementation of mixed potential model, referred to as the ANL-MPM Version 1, was verified by reproducing published results from the Canadian-MPM reports (Jerden et al., 2012). This report provides a detailed discussion of the actual coding that was used in the implementation of ANL-MPM Version 1. This report thus documents the base-line modeling approach (ANL-MPM) on which future versions will be based.

The ANL-MPM Version 1 (MATLAB implementation of the Canadian-MPM described by King and Kolar, 1999) is a 1-dimensional reaction-diffusion model that accounts for the following processes:

- Interfacial redox reaction kinetics influencing oxidative dissolution of the UO_2 matrix.
- Chemical or solubility based dissolution of the fuel matrix.
- Complexation of dissolved uranium by carbonate near the fuel surface and in the bulk solution.
- Production of hydrogen peroxide (which is the dominant fuel oxidant in anoxic repository environments) by alpha-radiolysis.
- Diffusion of reactants and products in the groundwater away from and towards the reacting fuel surface.
- Precipitation and dissolution of a U–bearing corrosion product layer on the fuel surface.
- Adsorption of uranium onto iron oxides.
- Arrhenius-type temperature dependence for all interfacial and bulk reactions.

Our modeling approach was divided into two stages: (1) implementation of the Canadian-MPM in the programming language MATLAB with only minor changes to the original model. The scripts resulting from this work are referred to as the ANL-MPM Version 1 and are the focus of this report. (2) Extension and optimization of the MPM implemented in MATLAB. This model, which is still under development, is referred to as the ANL-MPM Version 2. The Version 2 of the ANL-MPM quantifies the effects of dissolved hydrogen and the catalytic properties of NMP on fuel matrix dissolution. It has been optimized from a computing standpoint (relative to V1) by using a rapid diffusion approach which significantly decreases the time it takes for the model to converge on a unique solution and facilitates calculations of long-term behavior.

Sensitivity studies of the ANL-MPM Version 1 (presented in Jerden et al., 2012) have been used to identify key model parameters for which values must be determined from literature data, measured experimentally, or calculated with submodels, such as the concentrations of radiolytic products, for application to conditions other than those used in the Canadian-MPM. The focus of our model sensitivity studies and linked experimental work is on demonstrating and improving the applicability of the ANL-MPM to the full range of generic repository concepts being considered as part of the UFD campaign.

Because the ANL-MPM is based on fundamental principles, it is flexible enough to be applied to the full range of repository environments as well as shorter-term storage scenarios being considered as part of the UFD campaign. On-going experimental work described in Jerden et al.,

2012 and Ebert et al., 2012 is focused on providing key model parameter values that are needed to improve predictive accuracy and capabilities of the ANL-MPM.

2. ANL-MPM V1: CONCEPTUAL AND MATHEMATICAL BASIS

2.1. Model Assumptions, Mass-balance Equations and Reactions

The Canadian-MPM, on which the ANL-MPM is based, was developed to predict the corrosion behavior of used fuel inside a failed steel container under anticipated conditions in a granitic repository setting (King and Kolar, 1999, King and Kolar, 2003, Shoesmith et.al., 2003). Since the model is based on fundamental electrochemical and thermodynamic principals it is flexible in its application and can be extended to other repository environments of interest (a focus of on-going work as part of ANL's UFD research). Figure 2 shows the conceptual set-up for both the ANL-MPM Version 1 and the Canadian-MPM on which it is based.

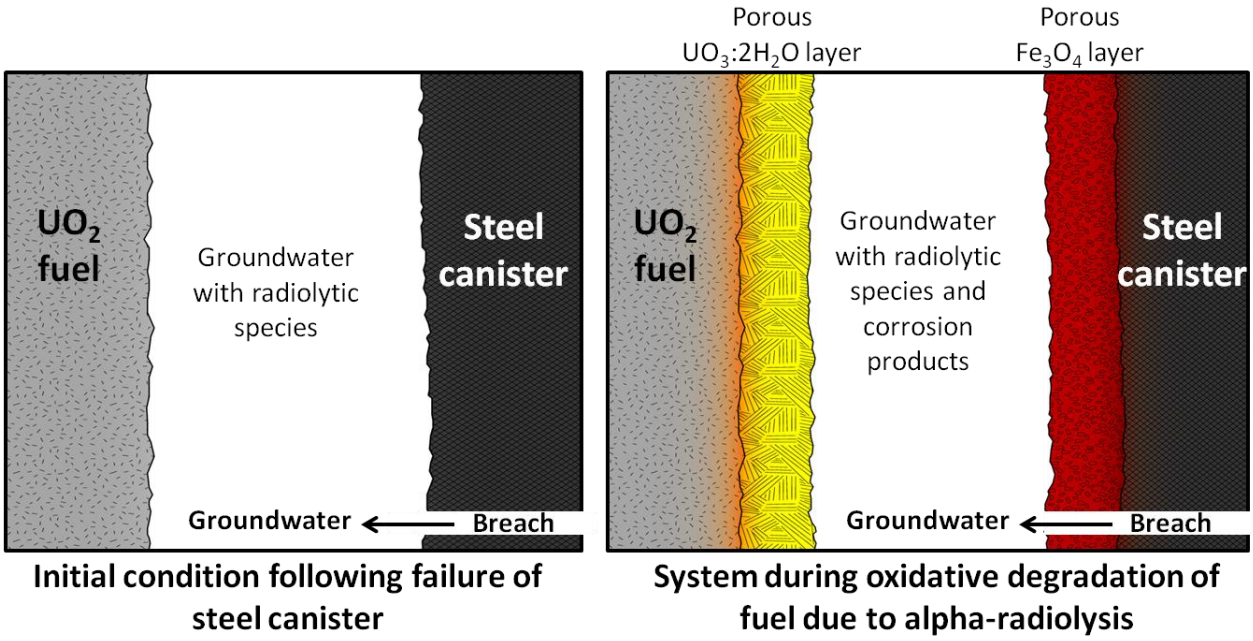

Figure 2. Simplified representation of breached canister system used in the Canadian-MPM (King and Kolar, 1999, King and Kolar, 2003, Shoesmith et.al., 2003) on which the ANL-MPM Version 1 (topic of this report) is based. This is a process model for the oxidative degradation of UO_2 fuel due to alpha radiolysis of groundwater.

The Canadian-MPM and ANL-MPM Version 1 both consist of ten one-dimensional reaction-diffusion equations (Equations 1 - 10), each describing the mass transport, precipitation/dissolution, adsorption/desorption and redox processes of the ten chemical species included in the model. Electrochemical rate expressions are used as boundary conditions for species that participate in the interfacial electrochemical reactions (King and Kolar, 1999, King and Kolar, 2003, Shoesmith et.al., 2003).

Table 1 identifies the chemical species (components) involved in Equations 1 - 10. These mass-balance equations are solved over a non-uniform spatial grid that is bounded by a UO_2 fuel surface on one side and steel surface (representing canister material) on the other. For a given set of conditions these bounding surface undergo oxidative-dissolution and thus represent sources of dissolved uranium and iron in the model.

The reactions that feed into Equations 1 - 10 are written out in Table 2 and shown in a two dimensional representation of the ANL-MPM Version 1 in Figure 4. Reaction-diffusion mass-balance equations on which the Canadian and ANL mixed potential models are based (see Table 1 for symbol identification):

$$\varepsilon\frac{\partial C_{UO_2^{2+}}}{\partial t}=\frac{\partial}{\partial x}\left(\tau_f\varepsilon D_{UO_2^{2+}}\frac{\partial C_{UO_2^{2+}}}{\partial x}\right)-\varepsilon k_8 C_{UO_2^{2+}}(C^{max}_{U(VI)ads}-C_{U(VI)ads})\rho$$
$$+k_{-8}C_{U(VI)ads}\rho-\varepsilon k_5 C_{UO_2^{2+}}C_{Fe^{2+}}-\varepsilon k_1\max(0,C_{UO_2^{2+}}-C^{sat}_{UO_2^{2+}}) \quad \text{(Eq.1)}$$

$$\varepsilon\frac{\partial C_{UO_2(CO_3)_2^{2-}}}{\partial t}=\frac{\partial}{\partial x}\left(\tau_f\varepsilon D_{UO_2(CO_3)_2^{2-}}\frac{\partial C_{UO_2(CO_3)_2^{2-}}}{\partial x}\right)$$
$$-\varepsilon k_9 C_{UO_2(CO_3)_2^{2-}}(C^{max}_{U(VI)ads}-C_{U(VI)ads})\rho+k_{-9}C_{U(VI)ads}\rho-\varepsilon k_6 C_{UO_2(CO_3)_2^{2-}}C_{Fe^{2+}} \quad \text{(Eq.2)}$$
$$-\varepsilon k_2\max(0,C_{UO_2(CO_3)_2^{2-}}-C^{sat}_{UO_2(CO_3)_2^{2-}})+k_{-2}C^{p}_{CO_3^{2-}}\delta(x-x_A)$$

$$\varepsilon\frac{\partial C_{U(VI)ads}}{\partial t}=\varepsilon k_8 C_{UO_2^{2+}}(C^{max}_{U(VI)ads}-C_{U(VI)ads})\rho-k_{-8}C_{U(VI)ads}\rho$$
$$+\varepsilon k_9 C_{UO_2(CO_3)_2^{2-}}(C^{max}_{U(VI)ads}-C_{U(VI)ads})\rho-k_{-9}C_{U(VI)ads}\rho \quad \text{(Eq.3)}$$

$$\varepsilon\frac{\partial C_{(IV)reprecip}}{\partial t}=\varepsilon k_5 C_{UO_2^{2+}}C_{Fe^{2+}}+\varepsilon k_6 C_{UO_2(CO_3)_2^{2-}}C_{Fe^{2+}} \quad \text{(Eq.4)}$$

$$\varepsilon\frac{\partial C_{CO_3^{-2}}}{\partial t}=\frac{\partial}{\partial x}\left(\tau_f\varepsilon D_{CO_3^{-2}}\frac{\partial C_{CO_3^{-2}}}{\partial x}\right)+\varepsilon k_9 C_{UO_2(CO_3)_2^{2-}}(C^{max}_{U(VI)ads}-C_{U(VI)ads})\rho-$$
$$2k_{-9}C_{U(VI)ads}\rho+2\varepsilon k_6 C_{UO_2^{2+}}C_{Fe^{2+}}+2\varepsilon k_2 \quad \text{(Eq.5)}$$
$$\max(0,C_{UO_2(CO_3)_2^{2-}}-C^{sat}_{UO_2(CO_3)_2^{2-}})-2k_{-2}\delta(x-x_A)$$

$$\varepsilon\frac{\partial C_{O_2}}{\partial t}=\frac{\partial}{\partial x}\left(\tau_f\varepsilon D_{O_2}\frac{\partial C_{O_2}}{\partial x}\right)-\varepsilon k_3 C_{O_2}C_{Fe^{2+}} \quad \text{(Eq.6)}$$

$$\varepsilon\frac{\partial C_{H_2O_2}}{\partial t}=\frac{\partial}{\partial x}\left(\tau_f\varepsilon D_{H_2O_2}\frac{\partial C_{H_2O_2}}{\partial x}\right)+\varepsilon G_{H_2O_2}R_D-\varepsilon k_4 C_{H_2O_2}C_{Fe^{2+}} \quad \text{(Eq.7)}$$

$$\varepsilon\frac{\partial C_{Fe^{2+}}}{\partial t}=\frac{\partial}{\partial x}\left(\tau_f\varepsilon D_{Fe^{2+}}\frac{\partial C_{Fe^{2+}}}{\partial x}\right)-4\varepsilon k_3 C_{O_2}C_{Fe^{2+}}-2\varepsilon k_4 C_{H_2O_2}C_{Fe^{2+}}$$
$$-2\varepsilon k_5 C_{UO_2^{2+}}C_{Fe^{2+}}-2\varepsilon k_6 C_{UO_2CO_3}C_{Fe^{2+}}-\varepsilon k_7\max(0,C_{Fe^{2+}}-C^{sat}_{Fe^{2+}}-)+k_{-7}\delta(x-x_B) \quad \text{(Eq.8)}$$

$$\varepsilon\frac{\partial C_A}{\partial t}=\varepsilon k_1\max(0,C_{UO_2^{2+}}-C^{sat}_{UO_2^{2+}})+\varepsilon k_2\max(0,C_{UO_2(CO_3)_2^{2-}}-C^{sat}_{UO_2(CO_3)_2^{2-}})-$$
$$k_2 C^{p}_{CO_3^{-2}}\delta(x-x_A) \quad \text{(Eq.9)}$$

$$\varepsilon\frac{\partial C_B}{\partial t}=\varepsilon k_7\max(0,C_{Fe^{2+}}-C^{sat}_{Fe^{2+}})-k_{-7}\delta(x-x_A) \quad \text{(Eq.10)}$$

Table 1. Description of symbols used in The mass-balance Equations 1 - 10. See Appendix 1 for default parameter values used in the ANL-MPM Version 1, MATLAB scripts.

Symbol	Parameter
k_i	Rate of reaction i (see Table 2 and Figure 3)
ε	Total porosity of corrosion layer
τ_f	Tortuosity factor for corrosion layer
ρ	Density of corrosion layer (mole/m^3)
$C_{UO_2^{2+}}$	Dissolved concentration (mole/m^3)
$D_{UO_2^{2+}}$	Bulk-solution diffusion coefficient (m^2/sec)
$C^{sat}_{UO_2^{2+}}$	Saturation concentration (mole/m^3)
$C_{U(VI)ads}$	U(VI) adsorbed to Fe3O4 layer (mole/g)
$C^{max}_{U(VI)ads}$	Adsorption capacity of UO_2^{2+} on Fe(II) corrosion layer (mole/g)
$C_{UO_2(CO_3)_2^{2-}}$	Dissolved concentration (mole/m^3)
$D_{UO_2(CO_3)_2^{2-}}$	Bulk-solution diffusion coefficient (m^2/sec)
$C^{sat}_{UO_2(CO_3)_2^{2-}}$	Saturation concentration (mole/m^3)
$C_{(IV)reprecip}$	Concentration of U(IV) formed by reduction of U(VI) by iron (mole/m^3)
$C_{CO_3^{-2}}$	Dissolved concentration (mole/m^3)
$D_{CO_3^{-2}}$	Bulk-solution diffusion coefficient (m^2/sec)
C_{O_2}	Dissolved concentration (mole/m^3)

D_{O_2}	Bulk-solution diffusion coefficient (m^2/sec)
$C_{Fe^{2+}}$	Dissolved concentration (mole/m^3)
$D_{Fe^{2+}}$	Bulk-solution diffusion coefficient (m^2/sec)
$C^{sat}_{Fe^{2+}}$	Saturation concentration (mole/m^3)
$C_{H_2O_2}$	Dissolved concentration (mole/m^3)
$D_{H_2O_2}$	Bulk-solution diffusion coefficient (m^2/sec)
$G_{H_2O_2}$	Primary radiolysis yield of H_2O_2 [mol/(J/kg)/m^3]
R_D	Spatial and time-dependent alpha radiation dose [(J/kg)/sec]
C_A	Auxiliary concentration of solid U(VI) (corrosion layer) (mole/m^3)
x_A	Thickness of U(VI) corrosion layer (m)
C_B	Auxiliary concentration of solid Fe(II) (corrosion layer) (mole/m^3)
x_B	Thickness of Fe(II) corrosion layer (m)
δ	Function for dissolution of porous corrosion layer: approximated by finite width profile

*****The reactions that feed into Equations 1 - 10 are shown in Table 2 and Figure 4.

A representation of the non-uniform spatial grid (showing calculation points as lines) over which the reaction-diffusion equations are solved is shown in Figure 3. The actual distance between the bounding surfaces and the number of calculation points used in a simulation can be specified by the user; the default values are 50 mm and 250 points respectively.

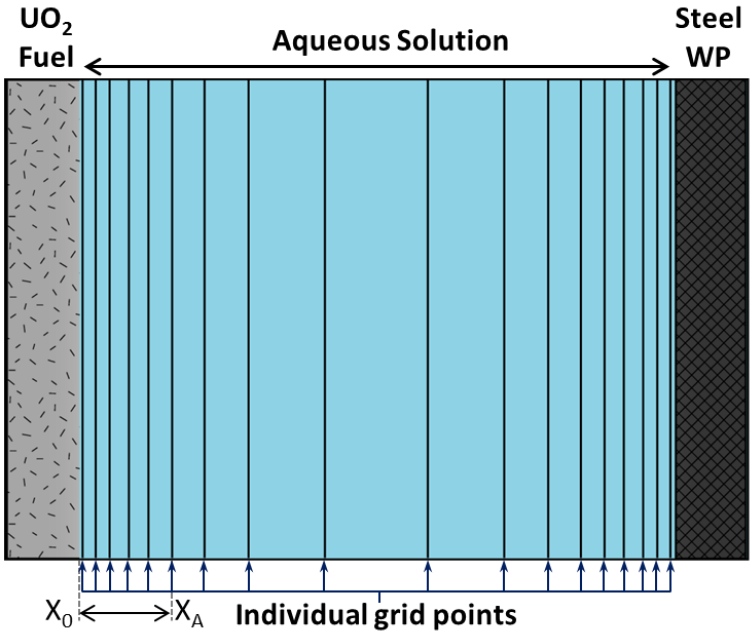

Figure 3. Schematic representation of mixed potential model grid spacing between the used fuel and steel surface boundaries (not all grid intervals are shown). Spacing is logarithmic with finer intervals at the two interfaces. The current implementation of the ANL-MPM

Version 1 contains 250 grid points with a minimum grid spacing of 1 micrometers and a maximum spacing of 1000 micrometers.

Table 2. Surface electrochemical reactions and bulk solution reactions tracked in ANL-MPM Version 1 (symbols map reactions onto Figure 4). For parameter values used to describe each reaction in the MATLAB scripts see Appendices 1 and 2.

Reactions	**Symbols for reactions used in Figure 2**
Anodic reactions on fuel and steel surfaces	
$UO_2 \rightarrow UO_2^{2+} + 2e^-$	**A**
$UO_2 + 2CO_3^{2-} \rightarrow UO_2(CO_3)_2^{2-} + 2e^-$	**B**
$H_2O_2 \rightarrow O_2 + 2H^+ + 2e^-$	**C**
$Fe \rightarrow Fe^{2+} + 2e^-$	**J**
$H_2 \rightarrow 2H^+ + 2e^-$	**L***
Cathodic reactions on fuel and steel surfaces	
$H_2O_2 + 2e^- \rightarrow 2OH^-$	**D, H**
$O_2 + 2H_2O + 4e^- \rightarrow 4OH^-$	**E, I**
$UO_2^{2+} + 2e^- \rightarrow U(IV)_{reprecipitate}$	**F**
$UO_2(CO_3)_2^{2-} + 2e- \rightarrow U(IV)_{reprecipitate} + 2CO_3^{2-}$	**G**
$H_2O + 4e^- \rightarrow ½H_2(aq) + OH^-$	**K**
Homogeneous Bulk Reactions	
$UO_2^{2+} + 2H_2O \rightarrow UO_3{:}2H_2O + 2H^+$	**1**
$UO_2(CO_3)_2^{2-} + 2H_2O \rightarrow UO_3{:}H_2O + 2CO_3^{2-} + 2H^+$	**2**
$UO_3{:}H_2O + 2CO_3^{2-} + 2H^+ \rightarrow UO_2(CO_3)_2^{2-} + 2H_2O$	**-2**
$O_2 + 2H_2O + 4Fe^{2+} \rightarrow 4Fe(III) + 4OH^-$	**3**
$H_2O_2 + 2Fe^{2+} \rightarrow 2Fe(III) + 2OH^-$	**4**
$UO_2^{2+} + Fe^{2+} \rightarrow Fe(III) + U(IV)$	**5**
$UO_2(CO_3)_2^{2-} + Fe^{2+} \rightarrow Fe(III) + U(IV) + 2CO_3^{2-}$	**6**
$Fe^{2+} + Fe(III) + 4H_2O \rightarrow Fe_3O_4 + 8H^+$	**7**
$Fe_3O_4 + 8H^+ \rightarrow Fe^{2+} + Fe(III) + 4H_2O$	**-7**
$UO_2^{2+} \rightarrow (UO_2^{2+})_{ads}$	**8**
$(UO_2^{2+})_{ads} \rightarrow UO_2^{2+}$	**-8**
$UO_2(CO_3)_2^{2-} \rightarrow (UO_2^{2+})_{ads} + CO_3^{2-}$	**9**
$(UO_2^{2+})_{ads} + CO_3^{2-} \rightarrow UO_2(CO_3)_2^{2-}$	**-9**

$UO_2 + 2H_2O \rightarrow U(OH)_4(aq)$	**10**
$H_2O_2 \rightarrow H_2O + 0.5O_2$	**11**

*Hydrogen oxidation reaction is not shown in Figure 4 nor in the scripts in Appendix 2. This reaction (L) was not accounted for in the Canadian-MPM of King and Kolar, 1999. It is included conceptually (as a place holder reaction) in anticipation of extending the ANL-MPM to account for how H_2 oxidation at the fuel surface may counter-act oxidative-dissolution processes driven by the formation of radiolytic oxidants (H_2O_2 in model in ANL-MPM Version 1).

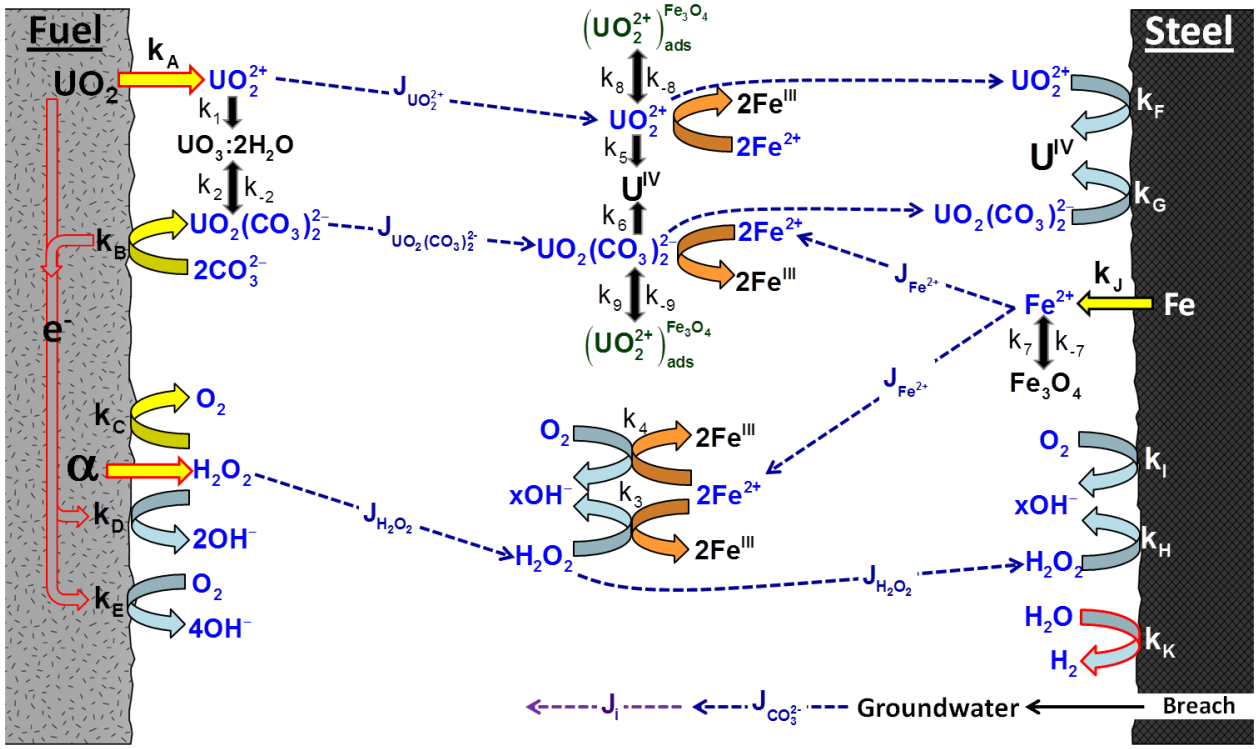

Figure 4. Reaction scheme for ANL-MPM Version 1 and Canadian-MPM of King and Kolar, (1999). Aqueous species are shown in blue, dotted lines represent diffusive fluxes. The "**k**" labels represent rate expressions for the individual half-reactions. Reactions labeled with letters are for heterogeneous (surface) processes and those labeled with numbers describe homogeneous processes. Anodic reactions are noted with yellow or orange arrows and the cathodic reactions are shown in blue arrows. "ads" stands for adsorbed.

The UO_2 and iron bounding surfaces in the model can host porous layers of corrosion phases that form when the dissolved concentration of U(VI) or Fe(II) reach specified saturation thresholds. These corrosion layers influence the rates of diffusion of species towards and away from the bounding surfaces. The only mode of mass transport in this model is by diffusion (no advection). Other key assumptions retained in the ANL implementation of the Canadian-MPM (in V1 and V2) are:

- 1-D model geometry with non-uniform spatial distribution with emphasis on surface reactions at used fuel and steel (Fe) interfaces.
- Uniform dissolution of fuel surface (no localized effects, e.g., grain boundary etching).

- Mass transport by diffusion only.
- System is saturated with groundwater, the supply of groundwater is not limiting.
- Used fuel cladding is excluded from system.
- $U(VI)O_3{:}2H_2O$ and Fe_3O_4 corrosion layers are treated as equivalent porous media with spatially and temporally constant porosity and tortuosity.
- $U(VI)O_3{:}2H_2O$ corrosion layer is assumed to be electrically insulating with electrochemical reactions restricted to areas at base of pores.
- $U(VI)O_3{:}2H_2O$ corrosion layer attenuates alpha dose rate to solution at the fuel surface.
- $U(VI)O_3{:}2H_2O$ corrosion layer may contain alpha-emitting radionuclides (user input).
- Fe_3O_4 is assumed to be the stable corrosion product of carbon steel.
- pH is constant (buffered) throughout system.

The non-electrochemical (chemical) dissolution of the UO_2 matrix (Reaction 10, $UO_2 + 2H_2O \rightarrow U(OH)_4(aq)$, Table 2) is assigned a rate constant of 1x10-17 mole/cm^{2}second (King and Kolar, 1999) and its solubility is modeled using saturation concentration of approximately 1x10-8 moles/L (King and Kolar, 1999). Due to this low saturation concentration and corresponding slow dissolution rate, the chemical dissolution of the fuel matrix does not represent a significant degradation mechanism in the current model.

2.2. Electrochemical Description of Model Bounding Surfaces (UO_2 and Carbon Steel)

In the ANL-MPM Version 1, the rate of mass loss from the used fuel (a quantification of degradation) is directly related to the corrosion current density by Faradays Law (Equation 11). The corrosion current density is defined as the sum of the current densities of the anodic fuel oxidation reactions (Reactions A and B in Table 2, Equation 12 below).

$$\frac{ML^{Fuel}}{time} = \frac{i_{CORR}^{Fuel} MW^{Fuel}}{nF} \qquad \text{(Eq.11)}$$

$$i_{CORR}^{Fuel} = i_A + i_B \qquad \text{(Eq.12)}$$

where ML^{Fuel}/time is the total mass loss rate (grams/m^{2}days) due to oxidative and chemical dissolution, i_{CORR}^{Fuel} is the corrosion current density (amp/m^2), MW^{Fuel} is the molecular weight (grams/mole), n is the number of electrons transferred, F is the Faraday constant C/mole). The corrosion current density is related to the used fuel corrosion potential by the Tafel Equation (Equation3).

$$E_{CORR}^{Fuel} = E_A^0 + \frac{RT}{\alpha_A F} \ln\left(\frac{i_{CORR}}{nF\varepsilon(SA)k_A}\right) \qquad \text{(Eq.13)}$$

where E_{CORR}^{Fuel} is the corrosion potential (Volts), E_A^0 is the standard potential for Reaction A (see Table 1), α_A is the electrical charge transfer coefficient (related to Tafel slope for reaction of

interest), ε is the porosity of the U(VI) corrosion layer covering the used fuel surface (m^3 void/m^3 corrosion phase), (SA) is the reactive surface area of the fuel (m^2), k_A is the rate constant for Reaction A (see Table 1), and R, T, F, n are the ideal gas constant, absolute temperature, Faraday constant and the number of electrons transferred respectively. As implied in Figure 5, the used fuel corrosion potential is also a function (E_0) of the concentrations of species involved in the oxidative dissolution of uranium (see Table 2 for reactions).

$$E_{CORR}^{Fuel}=E_0([CO_3^{2-}],[O_2],[H_2O_2][H_2]) \quad \text{(Eq.14)}$$

The relationships between reaction currents (directly proportional to reaction rates), rate constants, standard potentials and the corrosion potential for individual half-cell reactions at the used fuel surface (Table 2, Figure 4) are derived from the Tafel equations and quantified as follows:

$$i_A=nF\varepsilon k_A \exp\left[\frac{\alpha_A F}{RT}(E_{CORR}^{Fuel}-E_A^0)\right] \quad \text{(Eq.15)}$$

$$i_B=nF\varepsilon k_B \left[CO_3^{2-}\right]^2 \exp\left[\frac{\alpha_B F}{RT}(E_{CORR}^{Fuel}-E_B^0)\right] \quad \text{(Eq.16)}$$

$$i_C=nF\varepsilon k_C \left[H_2O_2\right] \exp\left[\frac{\alpha_C F}{RT}(E_{CORR}^{Fuel}-E_C^0)\right] \quad \text{(Eq.17)}$$

$$-i_D=nF\varepsilon k_D \left[H_2O_2\right] \exp\left[\frac{-\alpha_D F}{RT}(E_{CORR}^{Fuel}-E_D^0)\right] \quad \text{(Eq.18)}$$

$$-i_E=nF\varepsilon k_E \left[O_2\right] \exp\left[\frac{-\alpha_E F}{RT}(E_{CORR}^{Fuel}-E_E^0)\right] \quad \text{(Eq.19)}$$

$$i_L=nF\varepsilon k_L \left[H_2\right] \exp\left[\frac{-\alpha_L F}{RT}(E_{CORR}^{Fuel}-E_L^0)\right] \quad \text{(Eq.20)}$$

where E_{CORR}^{Fuel} is the corrosion potential (Volts), E_A^0 is the standard potential for Reaction A (see Table 2), α_A is the electrical charge transfer coefficient (related to Tafel slope for reaction of interest), ε is the porosity of the U(VI) corrosion layer covering the used fuel surface (m^3 void/m^3 corrosion phase), S is the reactive surface area of the fuel (m^2), k_A is the rate constant for Reaction A (see Table 2), and R, T, F, n are the ideal gas constant, absolute temperature, Faraday's constant and the number of electrons transferred respectively. Note that the equations are written with positive currents for anodic reactions and negative currents for cathodic reactions.

It follows from Equations Eq.21 - Eq.23 that the corrosion current densities for each half cell reaction can also be calculated based on the fluxes of key redox species (see Eq.1 - Eq.10 above for reaction-diffusion relationships):

$$2i_C + i_E = -nF\tau_f \varepsilon D_{O_2} \frac{\partial C_{O_2}(0,t)}{\partial x} \quad \text{(Eq.21)}$$

$$i_C - i_D = -nF\tau_f \varepsilon D_{H_2O_2} \frac{\partial C_{H_2O_2}(0,t)}{\partial x} \quad \text{(Eq.22)}$$

$$i_B = -nF\tau_f \varepsilon D_{UO_2(CO_3)_2^{2-}} \frac{\partial C_{UO_2(CO_3)_2^{2-}}(0,t)}{\partial x} \quad \text{(Eq.23)}$$

$$i_A = -nF\tau_f \varepsilon D_{UO_2^{2+}} \frac{\partial C_{UO_2^{2+}}(0,t)}{\partial x} \quad \text{(Eq.24)}$$

$$i_L = -nF\tau_f \varepsilon D_{H_2} \frac{\partial C_{H_2}(0,t)}{\partial x} \quad \text{(Eq.25)}$$

where τ_f and ε are the tortuosity and porosity of the U(VI) corrosion layer, D is the diffusion coefficient and C is the molar concentration, x is the distance from the used fuel surface (Figure 2) and (0,t) refers to the partial derivative of concentration at x = 0 and time = t.

The fundamental axiom on which kinetic mixed potential theory models (such as the ANL-MPM) is thus quantified by the Equation 26 (see Reactions A - L, Table 2):

$$i_A + i_B + i_L - i_C - i_D - i_E = 0 \quad \text{(Eq.26)}$$

Electrochemical and mass-flux relationships of the types written for the UO_2 surface (Eq.11-Eq.26) can also be written for the iron (steel) surface. See King and Kolar, 1999 for a complete treatment of the steel boundary surface.

2.3. Temperature Dependence of Rates and Key Parameter Values

The temperature dependence of the used fuel degradation rate is captured in the ANL-MPM using Arrhenius relationships for rate constants (Eq.27), saturation concentrations (Eq.28) and diffusion coefficients (Eq.29). A linear temperature dependency is used for standard electrochemical potentials (Eq.30).

$$k_i = k_i(T_r) \exp\left[\frac{\Delta H_i}{R}\left(\frac{1}{T_r} - \frac{1}{T}\right)\right] \quad \text{(Eq.27)}$$

$$C_i^{sat} = C_i^{sat}(T_r)\exp\left[\frac{\Delta H_i^{sat}}{R}\left(\frac{1}{T_r} - \frac{1}{T}\right)\right] \quad \text{(Eq.28)}$$

$$D_i = D_i(T_r)\exp\left[\frac{\Delta H_{D_i}}{R}\left(\frac{1}{T_r} - \frac{1}{T}\right)\right] \quad \text{(Eq.29)}$$

$$E_i^0 = E_i^0(T_r) + \Delta E_i^0(T - T_r) \quad \text{(Eq.30)}$$

where k is a rate constant, T_r is the reference temperature used in determining the activation energy (ΔH) and temperature dependence of the standard potential for a given half-cell reaction (ΔE^0), R is the ideal gas constant, C^{sat} is the molar concentration at which a given corrosion phase precipitates (UO_3:$2H_2O$ for corrosion of fuel surface), and D_i is the diffusion coefficient for component i.

The temperature dependence of the dissolution of the iron corrosion product layer (reaction k_{-7}) is more complicated and is not within the scope of this report since our focus was on UO_2 dissolution. For more details of the temperature dependence of iron corrosion see page 14 of King et al., 1999.

2.4. Treatment of Radiolysis and Generation of Radiolytic Oxidants (H_2O_2 only in ANL-MPM Version 1)

The spatial and temporal dependence of the alpha dose rate ([$\mathbf{R_D(x,t)} = R_{scale}R_{aq}(t)g(x)$], see Table 3, Figure 5 for explanation) is of fundamental significance within the ANL-MPM Version 1 because, at low concentrations of dissolved oxygen, the only oxidant within the system is the hydrogen peroxide produced by alpha radiolysis. Therefore the rate of matrix degradation in anoxic environments (granite repository, deep bore-hole, etc.) is directly proportional to the alpha dose rate.

Calculating the alpha dose rate (and thus H_2O_2 concentration) for corroding UO_2 fuel is complicated by the effects of U(VI) corrosion products (modeled as schoepite, UO_3:$2H_2O$ in ANL-MPM Version 1). A U(VI) corrosion product layer has four effects on the rate of UO_2 degradation:

- Corrosion layer can slow the rate of oxidative dissolution by decreasing the reactive surface area of the fuel (blocking or masking reaction sites).
- Corrosion layer can slow the rate of oxidative dissolution by blocking alpha-particles from interacting with water and producing radiolytic oxidants (decreases total moles H_2O_2 produced near fuel surface). The magnitude of this effect would be proportional to surface coverage of corrosion layer.
- Corrosion layer can slow the rate of oxidative dissolution by slowing the rate of diffusion of oxidants to the fuel surface: U(VI) layer is a tortuous porous mass of crystals.

- Corrosion layer can increase the rate of oxidative dissolution if alpha-emitting radionuclides (e.g., actinides) are incorporated into the U(VI) corrosion crystals or occluded within the mass of corrosion products.

All three of these effects are modeled in the ANL-MPM Version 1 by a radiolysis "sub-routine" that was included within the original Canadian-MPM (King and Kolar, 1999). Because of dependence of the used fuel dissolution rate on the alpha dose rate the radiolysis "sub-routine" is described in some detail. The MATLAB implementation of the radiolysis part of the ANL-MPM Version 1 is given in Appendix 2, lines 165 -209.

In the ANL-MPM Version 1 alpha-particles are assumed to have a constant energy of 5.3MeV and a solution penetration distance (α_{PEN}) of approximately 35 μm. The modeler can set the penetration distance over the range of α_{PEN} = 45μm for ~6.0MeV alpha-particles down to α_{PEN} = 10μm for ~2.3 MeV particles (King and Kolar, 1999). The quantity of hydrogen peroxide produced by alpha-radiolysis per unit of absorbed dose (G_{H2O2}) in both models is assumed to be 1.021E-10 mol/Gy cm^3 (Christensen and Sunder, 2000).

Table 3. Parameters describing radiolysis (generation of H_2O_2) in the ANL-MPM Version 1 (adapted from King and Kolar, 1999).

Parameter	Symbol	Value	Units
G-value for the primary a-radiolysis yield of H_2O_2	G_{H2O2}	1.02E-10	mol/Gy cm^3
Time-dependent alpha dose rate to the solution	$R_{aq}(t)$	From fuel burn-up and history	----
Spatial- and time dependent alpha-radiation dose rate $\mathbf{R_D(x,t) = R_{scale}R_{aq}(t)g(x)}$	$R_D(x,t)$	Calculated within MPM	----
Ratio of dose rate from U(VI) corrosion layer to dose rate from fuel	R_{film}	0 - 1	----
Geometrical factor describing a-radiation field	$g_f(x)$	Based on parameter inputs, see Fig. 5	----
a-particle penetration depth in water	a_{pen}	35	mm
Scaling factor for dose rate: for sensitivity runs	R_{scale}	1	----

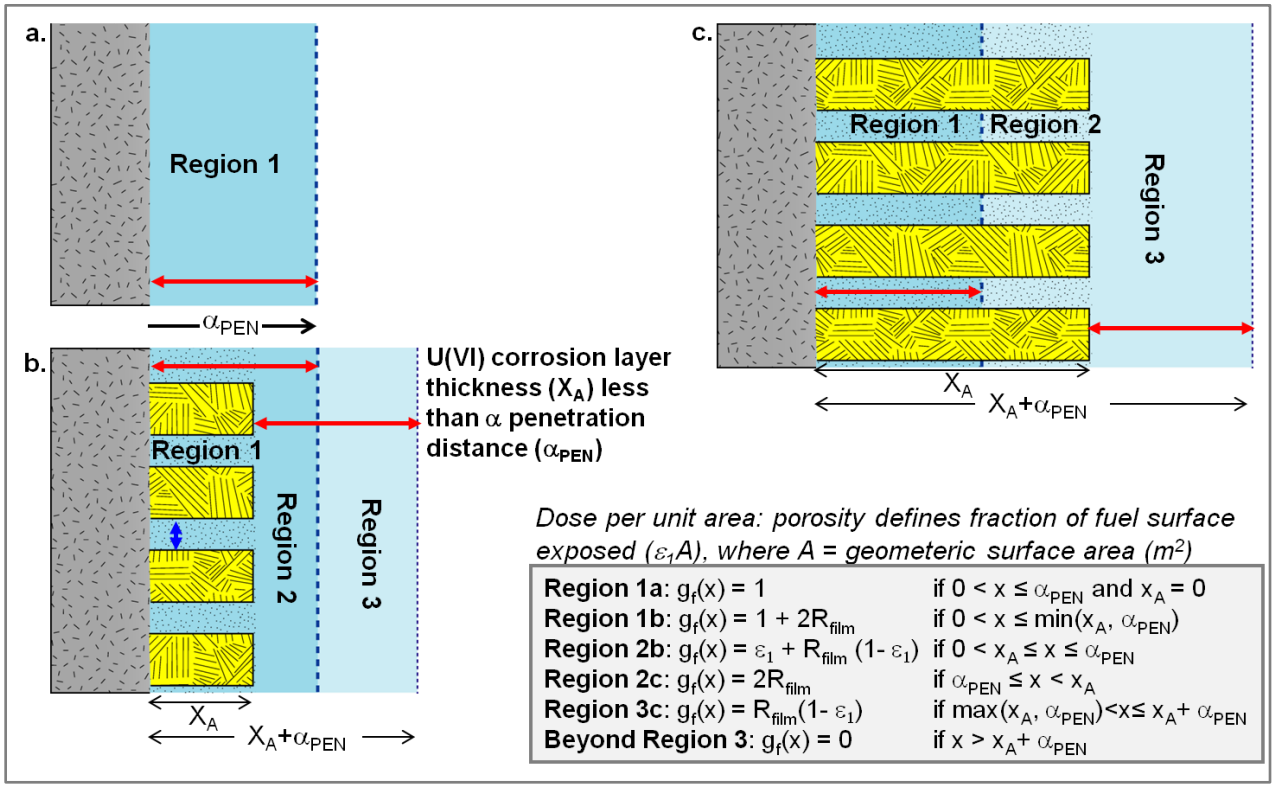

Figure 5. Conceptual rendering of the parallel pore model used for the treatment of radiolysis (H_2O_2 generation) in the ANL-MPM Version 1, as adapted from the Canadian-MPM of King and Kolar, 1999). See Table 3 for an explanation of the relationship used to determine the spatial- and temporal dose rate: $R_D(x,t) = R_{aq}(t)g(x)$. Red lines indicate alpha particle penetration distances for different scenarios.

The relationships shown conceptually in Figure can be summarized as follows:

- The spatial dependence of the a-dose rate and the effects of the precipitate film (corrosion layer) on the effective dose rate are taken into account. A number of regions at the corroded fuel surface need to be considered.
- Precipitated film (corrosion layer) is modeled as medium consisting of parallel pores have bulk porosity e_1 and effective pore cross-sectional surface are of ε_1 A where A is the geometrical surface area.
- Region 1b: $X_A \le \alpha_{pen}$ solution within pores is irradiated by fuel and walls of pores: effective dose per unit area $R_D(x,t) = \varepsilon_1 R_{aq}(x,t)(1+2R_{film})$ [accounts for fact that only pore solution irradiated: $\varepsilon_1 R_{aq}(x,t)$],
- Region 2b: $X_A < x \le \alpha_{pen}$ the solution (beyond film surf) is irradiated by exposed fuel ($\varepsilon_1 A$) and by the surf. of the film A(1- e_1): effective dose per unit area $R_D(x,t) = R_{aq}(x,t)$ $[\varepsilon_1 + R_{film}(1-\varepsilon_1)]$,
- Region 2c: $\alpha_{pen} \le x < X_A$ the solution (beyond a_{pen} within pores) is irradiated only by the pore internal surfaces $A(1-\varepsilon_1)$: effective dose per unit area $R_D(x,t) = \varepsilon_1 R_{aq}(x,t)\, 2R_{film}$,

- Region 3c: $X_A < x \leq (X_A + \alpha_{pen})$ the solution is irradiated by the surface of the porous film of cross-sectional area $A(1- \varepsilon_1)$: the effective dose per unit area $R_D(x,t) = R_{aq}(x,t)\ R_{film}\ (1- \varepsilon_1)$.

3. MATHMATICAL IMPLEMENTATION OF ANL-MPM VERSION 1: SUMMARY

The ANL-MPM Version 1 is based on a set of ordinary differential equations in which concentrations are the state variables (Equations 1 - 10). Based on the relationships shown in Equations 1 - 26, and given a set of initial concentration values for key species at the used fuel surface, a corrosion potential is calculated such that the total current flow at that surface is zero (this is the fundamental axiom of mixed potential theory, Eq.26). The overall rates for all surface reactions are then calculated at that corrosion potential. The rates of the surface reactions control the flux of chemical species from the surface into solution. A species flux from the fuel surface is used to update the concentrations in the solution at the interfacial boundary. The cycle of calculations is repeated for the desired length of time.

The general flow of inputs and calculations involved in the ANL-MPM Version 1 is shown in Figure 6. MATLAB release R2012a, for a 64-bit platform was used to implement the ANL-MPM Version 1. Implementation of the ANL-MPM Version 1 is summarized as follows:

- Several well tested, built-in mathematical tools available in MATLAB were used to facilitate rapid model implementation.
- The time derivatives of the species identified in Equations 1 - 10 were calculated explicitly to reduce the model to a system of ordinary differential equations. The MATLAB environment allows for fully explicit and fully implicit, differencing schemes to be used in solving the reaction-diffusion problems.
- An adaptive time-step algorithm determines the length of a given time step based on the concentration gradients that form in the model during the course of a simulation
- Modeling systems of partial differential equations (Equations 1 - 10) requires discretization in order to calculate approximate derivative values. Appendix 2, which shows the MATLAB code for implementation of the ANL-MPM Version 1, describes the scheme used for discretization of the reaction-diffusion equations in x and t. Algebraic equations are derived from the discretization procedure and solved using standard MATLAB tools. Equations 1 - 10 are each solved at a grid point for each time interval t (by default there are 250 solution points, however, this number can be changed by the user: see Appendix 2, lines 52 to 80).
- The temperature and radioactive dose histories are functions of time that are supplied explicitly as an argument to the MPM and are not a result of the calculations (King and Kolar, 1999, King and Kolar, 2003 and references therein).
- Placeholder values for physical constants that are not explicitly documented in the Canadian-MPM (e.g., reaction rates and diffusivities) were used to implement the model. The relevant physical constants will be updated based on analyses of literature data and from on-going electrochemical experiments.

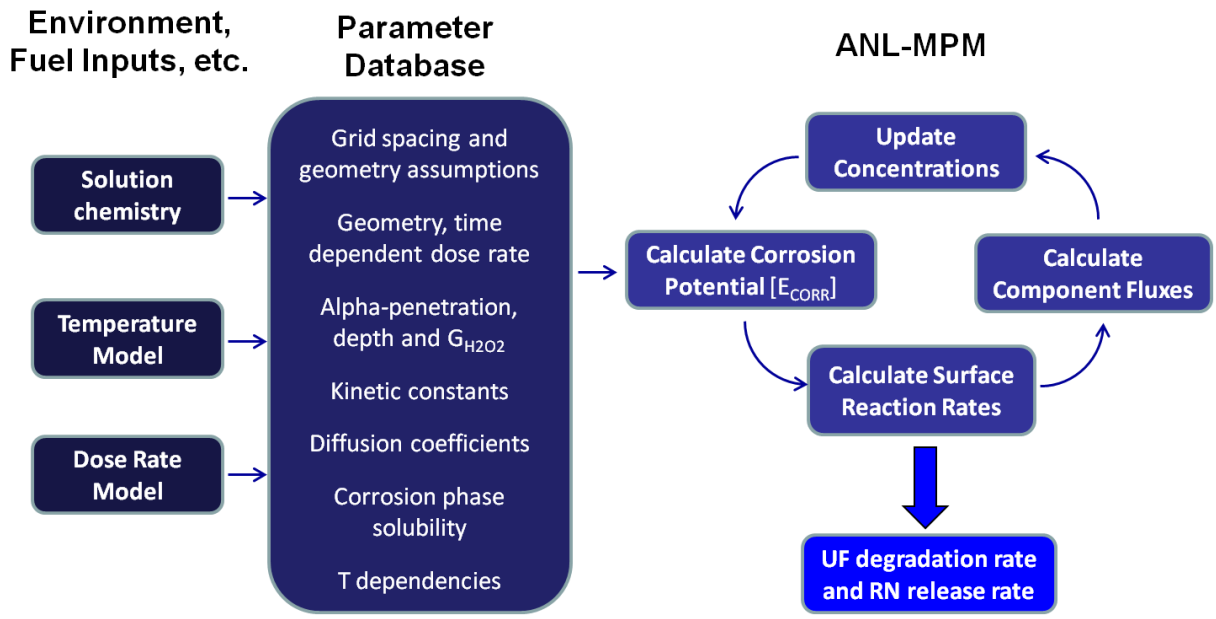

Figure 6. Conceptual drawing for the evolution of the used fuel corrosion potential and interfacial reaction rates with time as calculated by the ANL-MPM Version 1: see Appendix 2 for MATLAB scripts used to implement model.

4. SUMMARY AND FUTURE WORK

The purpose of this work is to develop and optimize a predictive model for the degradation of used uranium oxide fuel that is based on fundamental electrochemical and thermodynamic principals. This process model will be integrated with the UFD generic performance assessment model to provide a source term for radionuclide release for a disposal scenario of interest. The approach has been to implement, optimize and extend an existing, well tested electrochemical corrosion model to specific fuel degradation processes and mechanisms of interest.

Some of the specific multi-year objectives for this project that have been achieved thus far include:

- Implemented, using our own scripts/code (in MATLAB environment), an established and well documented used fuel degradation model (Canadian-mixed potential model) that is based on mixed potential theory (this report, see Appendices 1 and 2 for details on ANL MATLAB scripts).
- Verified our scripting and coding by reproducing published results from the Canadian model (Jerden et al., 2012).
- Performed sensitivity analyses to determine which model parameters and input variables have the strongest impact on the calculated used fuel degradation rate (Jerden et al., 2012).
- Completed a critical review of the sources of all model parameters and input variables to determine which values need further investigation through literature review or experimental studies. This review also identified which variables must be provided by other process models (Jerden et al., 2012).
- Extended the base-case model to quantify the role of dissolved hydrogen in protecting used fuel from oxidative dissolution by lowering the electrochemical potential at the fuel surface (Jerden et al., 2012).
- Developed a plan to extend the base-case model to account for the catalytic effects of fission product alloy phase (noble metal particles) on reactions affecting UO_2 dissolution, such as the kinetic balance of H_2 oxidation and H_2O_2 reduction (Jerden et al., 2012).

This work has provided a working, base-line model that is being modified and extended to include additional reactions and factors affecting fuel dissolution. The initial version of the MATLAB implementation of mixed potential model, referred to as the ANL-MPM Version 1, was verified by reproducing published results from the Canadian-MPM reports (Jerden et al., 2012). This report provides a detailed discussion of the actual coding that was used in the implementation of ANL-MPM Version 1. This report thus documents the base-line ANL-MPM on which future versions will be based.

This report presents the details of the first version of our UO_2 fuel matrix degradation model. The model described was produced by implementing the Canadian-mixed potential model for UO_2 fuel dissolution (King and Kolar, 1999, King and Kolar, 2003, Shoesmith et.al., 2003) using the numerical computing environment and programming language MATLAB (release R2012a on 64-bit platform). The intent of the initial version was to reproduce the approach used by King and Kolar, 1999 to gain experience with the model, identify key model parameters to be

measured experimentally or determined from evaluation of the literature, and provide a touchstone for future modifications of the code.

The initial version of the MATLAB implementation of mixed potential model, referred to as the ANL-MPM Version 1, was verified by reproducing published results from the Canadian-MPM reports (Jerden et al., 2012). This report provides a detailed discussion of the actual coding that was used in the implementation of ANL-MPM Version 1.

The ANL-MPM Version 1 (MATLAB implementation of the Canadian-MPM described by King and Kolar, 1999) is a 1-dimensional reaction-diffusion model that accounts for the following processes:

- Interfacial redox reaction kinetics influencing oxidative dissolution of the UO_2 matrix.
- Chemical or solubility based dissolution of the fuel matrix.
- Complexation of dissolved uranium by carbonate near the fuel surface and in the bulk solution.
- Production of hydrogen peroxide (which is the dominant fuel oxidant in anoxic repository environments) by alpha-radiolysis.
- Diffusion of reactants and products in the groundwater away from and towards the reacting fuel surface.
- Precipitation and dissolution of a U–bearing corrosion product layer on the fuel surface.
- Adsorption of uranium onto iron oxides.
- Arrhenius-type temperature dependence for all interfacial and bulk reactions.

The ANL-MPM Version 1 will be used as a base-line check for future versions of the ANL-MPM. A working beta version of the ANL-MPM Version 2 has been implemented and is now being tested and optimized. The ANL-MPM Version 2 accounts for the following processes and conditions (in addition to processes listed above):

- Quantifies the oxidation of dissolved H_2 at the used fuel/solution interface: H_2 concentration to be supplied by other EBS model or user specified).
- Represents the NMPs as a separate domain at the used fuel/solution interface. The "size" of the NMP domain (relative to the fuel) is specified by the user in terms of a surface coverage and is electrically linked with the UO_2 matrix by a user adjustable resistance. This will allow the effects of NMP corrosion and sorption on the catalytic efficiency to be taken into account.
- Quantifies the bulk decomposition of hydrogen peroxide (with temperature dependence).
- Provides option for user to specify temperature and dose profiles of the fuel (profiles can be constant single values or functions).
- Includes Rapid diffusion option to facilitate the calculation of concentrations of species whose diffusion coefficients are sufficiently large that they reach steady state on the order of days (decreases computer time needed for model convergence).

This report focuses on the base-line model ANL-MPM Version 1; for a discussion of the conceptual basis for the ANL-MPM Version 2 see Jerden et al., 2012.

Because the ANL-MPM is based on fundamental principles, it is flexible enough to be applied to the full range of repository environments as well as shorter-term storage scenarios being considered as part of the UFD campaign. On-going experimental work described in Jerden et al., 2012 and Ebert et al., 2012 is focused on providing key model parameter values that are needed to improve predictive accuracy and capabilities of the ANL-MPM.

5. REFERENCES CITED AND CONSULTED

Christensen, H. and Sunder, S. (2000). "Current State of Knowledge of Water Radiolysis Effects on Spent Nuclear Fuel Corrosion." *Nuclear Technology, 131*, (1), 102-123.

Ebert, W. L., Cruse, T. A., and Jerden J., (2012), "Electrochemical Experiments Supporting Oxide Fuel Corrosion Model" U.S. Department of Energy Used Fuel Disposition Campaign, Argonne National Laboratory Milestone Report FCRD-UFD-2012-000201

Jerden, J., Fortner, J., Cruse, T., Cunnane, J. (2011). *Repository Science/Waste Form Degradation & Radionuclide Mobilization*, Prepared for U.S. Department of Energy Used Fuel Disposition Campaign, Argonne National Laboratory Milestone Report M41UF030701 for the work package: FTAN11UF0307, July, 2011.

Jerden, J., Fortner, J., Cruse, T., Frey, K., Ebert, W., (2012). *Experimental Plan for ANL Electrochemical Corrosion Studies*, Prepared for U.S. Department of Energy Used Fuel Disposition Campaign, Argonne National Laboratory Milestone Report FCRD-USED-2012-000, January 6, 2012

Jerden, J., Frey, K., Cruse, T., and Ebert, W. (2012a). *Waste Form Degradation Model Status Report: Electrochemical Model for Used Fuel Matrix Degradation Rate*. FCRD-UFD-2012-000169.

King, F. and Kolar, M. (1999) "Mathematical Implementation of the Mixed-Potential Model of Fuel Dissolution Model Version MPM-V1.0", Ontario Hydro, Nuclear Waste Management Division Report No. 06819-REP-01200-10005 R00.

King, F. and Kolar, M. (2002) "Validation of the Mixed-Potential Model for Used Fuel Dissolution Against Experimental Data", Ontario Hydro, Nuclear Waste Management Division Report No. 06819-REP-01200-10077-R00.

King, F. and Kolar, M. (2003) "The Mixed-Potential Model for UO2 Dissolution MPM Versions V1.3 and V1.4", Ontario Hydro, Nuclear Waste Management Division Report No. 06819-REP-01200-10104 R00.

Sassani et al., 2012 *"Integration of EBS Models with Generic Disposal System Models"*, U.S. Department of Energy, Used Fuel Disposition Campaign milestone report: M2FT-12SN0806062, September, 7 2012

Shoesmith, D.W., S. Sunder, M. G. Bailey, and G. J. Wallace (1989). "The Corrosion of Nuclear Fuel (UO_2) in Oxygenated Solutions." *Corrosion Science, 29,* 1115-1128.

Shoesmith, D.W., S. Sunder, and J. C. Tait (1998). "Validation of an Electrochemical Model for the Oxidative Dissolution of Used CANDU Fuel." *Journal of Nuclear Materials, 257,* 89-98.

Shoesmith, D.W. (2000). "Fuel Corrosion Processes Under Waste Disposal Conditions." *Journal of Nuclear Materials, 282,* 1-31.

Shoesmith, D.W., M. Kolar, and F. King (2003). "A Mixed-Potential Model to Predict Fuel (Uranium Dioxide) Corrosion Within a Failed Nuclear Waste Container" *Corrosion, 59,* 802-816.

Shoesmith, D. W. (2007). *Used Fuel and Uranium Dioxide Dissolution Studies – A Review*. NWMO TR-2007-03, July 2007 Nuclear Waste Management Organization, 22 St. Clair Avenue East, 6th Floor, Toronto, Ontario M4T 2S3, Canada.

Shoesmith, D. W. (2008). *The Role of Dissolved Hydrogen on the Corrosion/Dissolution of Spent Nuclear Fuel*, NWMO TR-2008-19, November 2008 Nuclear Waste Management Organization, 22 St. Clair Avenue East, 6th Floor, Toronto, Ontario M4T 2S3, Canada.

Sunder, S., N.H. Miller, D.W. Shoesmith, (2004). "Corrosion of uranium dioxide in hydrogen peroxide solutions" *Corrosion Science, 46,* 1095–1111.

UFD R&D Roadmap (2011). *Used Fuel Disposition Campaign Disposal Research and Development Roadmap*. FCRD-USED-2011-000065 REV0, March 2011.

Appendix 1

Input parameter summary for ANL-MPM Version 1 for MATLAB (release R2012a). Black text is copied directly from MATLAB scripts, green text are comments in the code.

```
F = 96487;  % C/mol
R = 8.314;  % J/mol/K
L = 0.05; % m distance fuel - steel
nPts = 250; Number of grid points
t = 1*365*24*60*60; % s time of simulation

D(MPM_data.UO2)  = 0.5e-09*exp(15000*dT);   % m^2/s
D(MPM_data.UCO3) = 0.5e-09*exp(15000*dT);   % m^2/s
D(MPM_data.Uads) = 0.1e-10*exp(15000*dT);   % m^2/s
D(MPM_data.Usus) = 0.1e-09*exp(15000*dT);   % m^2/s
D(MPM_data.CO3)  = 1.7e-09*exp(15000*dT);   % m^2/s
D(MPM_data.O2)   = 1.7e-09*exp(15000*dT);   % m^2/s
D(MPM_data.H2O2) = 1.7e-09*exp(15000*dT);   % m^2/s
D(MPM_data.Fe2)  = 0.5e-09*exp(15000*dT);   % m^2/s
D(MPM_data.UO3)  = 0.1e-10*exp(15000*dT);   % m^2/s
D(MPM_data.FeO)  = 0.1e-10*exp(15000*dT);   % m^2/s
D(MPM_data.H2)   = 4.5e-09*exp(15000*dT);   % m^2/s

M(MPM_data.UO2)  = 0.302;   % kg/mol
M(MPM_data.UCO3) = 0.422;   % kg/mol
M(MPM_data.Uads) = 0.302;   % kg/mol
M(MPM_data.Usus) = 0.270;   % kg/mol
M(MPM_data.CO3)  = 0.060;   % kg/mol
M(MPM_data.O2)   = 0.032;   % kg/mol
M(MPM_data.H2O2) = 0.034;   % kg/mol
M(MPM_data.Fe2)  = 0.056;   % kg/mol
M(MPM_data.UO3)  = 0.322;   % kg/mol
M(MPM_data.FeO)  = 0.232;   % kg/mol
M(MPM_data.H2)   = 0.002;   % kg/mol

rho(1) = 4980;   % kg/m^3; U(VI) corrosion layer (Schoepite)(ID = 1)
rho(2) = 1000;   % kg/m^3; Bulk water layer                 (ID = 2)
rho(3) = 5173;   % kg/m^3; Iron oxide corrosion layer       (ID = 3)

epsl(1) = 0.45;   % U(VI) corrosion layer (Schoepite)
epsl(2) = 1.00;   % Water layer
epsl(3) = 0.10;   % Iron oxide corrosion layer

tau(1) = 0.10;   % U(VI) corrosion layer (Schoepite) (ID = 1)
tau(2) = 1.00;   % Bulk water layer           (ID = 2)
tau(3) = 0.10;   % Iron oxide corrosion layer (ID = 3)

G    = 1.021e-4;   % generation value in mol/(J/kg)/m^3
penD = 3.5e-5;     % alpha particle penetration distance in m
Rf   = 1.0;        % Factor for dose from corrosion film
dR = 0.02 % Constant dose rate (J/kg)/s

T = 298.15 % constant temperature in K

Cs(MPM_data.UO2,:)   = 3.20e-2*exp(6e4*dT);          % mol/m^3
```

```
        Cs(MPM_data.UCO3,:)  = 4.67e-2*exp(6e4*dT)*...
                                 (conc(MPM_data.CO3)^1.34);    % mol/m^3
        Cs(MPM_data.Uads,:)  = 5.0e-3*(rho(3)*epsl(3));        % mol/m^3
        Cs(MPM_data.Fe2,:)   = 1.00e-2*exp(6e4*dT);            % mol/m^3

        Cinit(MPM_data.CO3)  = 1.0e-6; % mol/m^3
        Cinit(MPM_data.O2)   = 1.0e-6; % mol/m^3

        % One entry per half reaction at fuel surface
        %   cMat:  number of electrons generated
        %   kMat:  reaction rate constant (mol/m^3/s; or appropriate)
        %   ectc:  electrochemical charge transfer coefficient
        %   oMat:  reaction order in concentrations
        %   sMat:  stoichiometry matrix
        %   Ezero: standard potential (Vsce)

        %*********************************%
        % Fuel 1: Reaction A              %
        % Fuel -> UO2(2+) + 2e(-)         %
        %*********************************%
        cMat(1,1)  =  2;
        kMat(1,1)  =  5.0e-8*exp(6.0e4*dT);
        ectc(1,1)  =  0.96;
        Ezero(1,1) =  0.169 - 0.000248*(T-298);

        %*******************************************%
        % Fuel 2: Reaction B                        %
        % Fuel + 2CO3(2-) -> UO2(CO3)2(2-) + 2e(-)  %
        %*******************************************%
        cMat(2,1)  =  2;
        kMat(2,1)  =  1.43e-12 *exp(6.0e4*dT);
        ectc(2,1)  =  0.82;
        Ezero(2,1) = -0.173 + 0.002100*(T-298);

        %*********************************%
        % Fuel 3: Reaction C              %
        % H2O2 -> O2 + 2H(+) + 2e(-)      %
        %*********************************%
        cMat(3,1)  =  2;
        kMat(3,1)  =  7.4e-8   *exp(6.0e4*dT);
        ectc(3,1)  =  0.41;
        Ezero(3,1) = -0.121 - 0.000993*(T-298);

        %*********************************%
        % Fuel 4: Reaction D              %
        % H2O2 + 2e(-) -> 2OH(-)          %
        %*********************************%
        cMat(4,1)  = -2;
        kMat(4,1)  =  1.2e-12  *exp(6.0e4*dT);
        ectc(4,1)  = -0.41;
        Ezero(4,1) =  0.973 - 0.000698*(T-298);

        %*********************************%
        % Fuel 5: Reaction E              %
        % O2 + 2H2O + 4e(-) -> 4OH(-)     %
        %*********************************%
```

```
        cMat(5,1)  = -4;
        kMat(5,1)  =  1.4e-12  *exp(6.0e4*dT);
        ectc(5,1)  = -0.50;
        Ezero(5,1) =  0.426 - 0.000123*(T-298);

        %*********************************%
        % Container 1: Reaction F          %
        % UO2(2+) + 2e(-) -> UO2           %
        %*********************************%
        cMat(1,1)  = -2;
        kMat(1,1)  =  1.00e-9  *exp(6.0e4*dT);
        ectc(1,1)  = -0.50;
        Ezero(1,1) =  0.169 - 0.000248*(T-298);

        %*****************************************%
        % Container 2: Reaction G                  %
        % UO2(CO3)2(2-) + 2e(-) -> UO2 + 2CO3(2-)  %
        %*****************************************%
        cMat(2,1)  = -2;
        kMat(2,1)  =  1.00e-10 *exp(6.0e4*dT);
        ectc(2,1)  = -0.50;
        Ezero(2,1) = -0.173 + 0.002100*(T-298);

        %*********************************%
        % Container 3: Reaction H          %
        % H2O2 + 2e(-) -> 2OH(-)           %
        %*********************************%
        cMat(3,1)  = -2;
        kMat(3,1)  =  1.6e-14  *exp(6.0e4*dT);
        ectc(3,1)  = -0.38;
        Ezero(3,1) =  0.973 - 0.000698*(T-298);

        %*********************************%
        % Container 4: Reaction I          %
        % O2 + 2H2O + 4e(-) -> 4OH(-)      %
        %*********************************%
        cMat(4,1)  = -4;
        kMat(4,1)  =  3.2e-12  *exp(6.0e4*dT);
        ectc(4,1)  = -0.42;
        Ezero(4,1) =  0.426 - 0.000123*(T-298);

        %*********************************%
        % Container 5: Reaction J          %
        % Container -> Fe(2+) + 2e(-)      %
        %*********************************%
        cMat(5,1)  =  2;
        kMat(5,1)  =  2.2e-5*exp(10.4e4*dT);
        ectc(5,1)  =  1.08;
        Ezero(5,1) = -0.650 + 0.000680*(T-298);

        %*********************************%
        % Container 6: Reaction K          %
        % H2O + e(-) -> (1/2)H2 + OH(-)    %
        %*********************************%
        cMat(6,1)  = -1;
        kMat(6,1)  =  1.2e-7   *exp(5.21e4*dT);
```

```
163         ectc(6,1)  = -0.48;
164         Ezero(6,1) = -0.802 - 0.001900*(T-298);
165
166         %************************************%
167         % Bulk 1: Reaction 1                 %
168         % UO2(2+) + 2OH(-) + H2O -> UO3*2H2O  %
169         %************************************%
170         kMat(1,1)  =   1e-3*exp(6.0e4*dT);
171
172         %****************************************************%
173         % Bulk 2: Reaction 2f                                %
174         % UO2(CO3)2(2-) + 2OH(-) + H2O -> UO3*2H2O + 2CO3(2-)  %
175         %****************************************************%
176         kMat(2,1)  =   1e-4*exp(6.0e4*dT);
177
178         %****************************************************%
179         % Bulk 3: Reaction 2r                                %
180         % UO3*2H2O + 2CO3(2-) -> UO2(CO3)2(2-) + 2OH(-) + H2O  %
181         %****************************************************%
182         kMat(3,1)  =   6.3e-12*exp(6.0e4*dT);
183
184         %*****************************************%
185         % Bulk 4: Reaction 3                     %
186         % O2 + 4Fe(2+) + 8OH(-) -> 4H2O + 2Fe2O3  %
187         %*****************************************%
188         kMat(4,1)  =   5.9e-1*exp(6.0e4*dT);
189
190         %*****************************************%
191         % Bulk 5: Reaction 4                     %
192         % H2O2 + 2Fe(2+) + 4OH(-) -> 3H2O + Fe2O3  %
193         %*****************************************%
194         kMat(5,1)  =   6.9e-2*exp(4.2e4*dT);
195
196         %************************************************%
197         % Bulk 6: Reaction 5                             %
198         % UO2(2+) + 2Fe(2+) + 6OH(-) -> UO2 + 3H2O + Fe2O3  %
199         %************************************************%
200         kMat(6,1)  =   1.0e-2*exp(6.0e4*dT);
201
202         %*********************************************************************%
203         % Bulk 7: Reaction 6                                                  %
204         % UO2(CO3)2(2-) + 2Fe(2+) + 6OH(-) -> UO2 + 2CO3(2-) + 3H2O + Fe2O3 %
205         %*********************************************************************%
206         kMat(7,1)  =   1.0e-3*exp(6.0e4*dT);
207
208         %************************************************%
209         % Bulk 8: Reaction 7f                            %
210         % Fe(2+) + 2OH(-) -> (1/3)Fe3O4 + (2/3)H2O + (1/3)H2  %
211         %************************************************%
212         kMat(8,1)  =   1.0e-3*exp(6.0e4*dT);
213
214         %************************************************%
215         % Bulk 9: Reaction 7r                            %
216         % (1/3)Fe3O4 + (2/3)H2O + (1/3)H2 -> Fe(2+) + 2OH(-)  %
217         %************************************************%
```

```
218         kMat(9,1)  = 0;
219
220         %***********************************%
221         % Bulk 10: Reaction 8f (adsorption) %
222         % UO2(2+) -> UO2(2+)                %
223         %***********************************%
224         kMat(10,1)  =   1e-6*exp(0.0*dT);
225
226         %***********************************%
227         % Bulk 11: Reaction 8r (desorption) %
228         % UO2(2+) -> UO2(2+)                %
229         %***********************************%
230         kMat(11,1)  =   1e-9*exp(0.0*dT);
231
232         %*************************************%
233         % Bulk 12: Reaction 9f (adsorption)   %
234         % UO2(CO3)2(2-) -> UO2(2+) + 2CO3(2-) %
235         %*************************************%
236         kMat(12,1)  =   1e-9*exp(0.0*dT);
237
238         %*************************************%
239         % Bulk 13: Reaction 9r (desorption)    %
240         % UO2(2+) + 2CO3(2-) -> UO2(CO3)2(2-)  %
241         %*************************************%
242         kMat(13,1)  =   1e-9*exp(0.0*dT);
```

Appendix 2

MATLAB scripts for implementation of the ANL-MPM Version 1 (MATLAB release R2012a). Black text is active code, green text are comments in the code. The ANL-MPM Version 1 for MATLAB consists of five individual files that must be in the MATLAB operational folder to run the model. The code is presented in the following order below:

The **MPM_data** file is a data container that holds information on simulation time steps, grid spacing, temperature, dose rate, H_2O_2 generation, species stoichiometry, saturation concentrations, diffusion coefficients, porosity, tortuosity of corrosion layer etc.

The **MPM_react** file defines the reactions and calculates the rates of reactions and corrosion potentials at the fuel surface (and container surface if included). Probably the most important file since the corrosion potential is the defining characteristic of the model. This file is called recursively when calculating the corrosion potential. This file also defines the reactions and calculates the rates of reactions that occur in the space between the fuel surface and end of domain (end of domain can be iron canister or bulk solution).

The **MPM_main** file is a top level function that calls all the other files needed to run the ANL-MPM Version 1.

The **MPM_odefun** file provides input arguments to the MATLAB ordinary differential equation solver.

The **MPM_odestat** file provides the status function and reshape of matrix for the MATLAB ordinary differential equation solver.

The **MPM_output** file writes data to a comma separated value file.

This file: **MPM_data**, is a data container that holds information on simulation time steps, grid spacing, temperature, dose rate, H_2O_2 generation, species stoichiometry, saturation concentrations, diffusion coefficients, porosity, tortuosity of corrosion layer etc.

```
%Start MATLAB script

classdef MPM_data < uint16[I]

  enumeration
    % Enumeration of component list (members of MATLAB class here defined)
    UO2      (1)   % UO2 (2+)        (aqueous)
    UCO3     (2)   % UO2(CO3)2 (2-)  (aqueous)
    Uads     (3)   % UO2 (2+)        (adsorbed on solid iron corrosion)
    Usus     (4)   % UO2              (homogeneous solid suspension)
    CO3      (5)   % CO3 (2-)        (aqueous)
    O2       (6)   % O2               (aqueous)
    H2O2     (7)   % H2O2             (aqueous)
    Fe2      (8)   % Fe (2+)         (aqueous)
    UO3      (9)   % UO3*2H2O        (solid uranium corrosion product)
    FeO     (10)   % Fe3O4           (solid iron corrosion product)
    H2      (11)   % H2               (aqueous)

    % Components not tracked by code: produced in bulk / at surfaces;
    %       (XX)   % Fe2O3           (homogeneous solid suspension)
    %       (XX)   % H(+)            (aqueous)
    %       (XX)   % OH(-)           (aqueous)
end

  methods(Static)[II]

    % Faraday's constant
    function F = constF()
      F = 96487;  % C/mol
    end

    % Gas constant
    function R = constR()
      R = 8.314;  % J/mol/K
    end

    % Number of components
    function nCmps = cmpList()
      [~,compNames] = enumeration('MPM_data');
      nCmps = length(compNames);
    end
```

[I] ***classdef***: begins the class definition. There are many different data types, or classes, in MATLAB. You can build matrices and arrays of floating-point and integer data, characters and strings, and logical true and false states.

[II] ***static methods*** are associated with a class, but not with specific instances of that class. These methods do not perform operations on individual objects of a class and, therefore, do not require an instance of the class as an input argument, like ordinary methods. Static methods are useful when you do not want to first create an instance of the class before executing some code. For example, you might want to set up the MATLAB environment or use the static method to calculate data needed to create class instances.

```
47      % Dimension of environment, distance from fuel surf to steel surf
48      function L = cellLen()
49        L = 0.05; % m
50      end
51
52      % Number of grid points used
53      function nPts = gridPts()
54        nPts = 250;
55      end
56
57      % Grid subdivisions
58      function lmat = ordVec()
59        % Retrieve constants
60        cellLen = MPM_data.cellLen();
61        gridPts = MPM_data.gridPts();
62
63        % Spaceing parameter
64        sp = 0;
65
66        % Symmetric, logspace grid
67        hp = floor(gridPts/2);
68        df = 2*(10^sp)/cellLen;
69        x1 = (logspace(-3,sp,hp)-1e-3)/df; x1(1) = 0;
70        IL = x1(end)-x1(end-1);
71        if(hp == gridPts/2)
72          mf = (x1(end)-IL/2)/x1(end);
73          x1 = x1*mf;
74          lmat = [x1,cellLen-fliplr(x1)];
75        else
76          mp = x1(end); mf = (x1(end)-IL)/x1(end);
77          x1 = x1*mf;
78          lmat = [x1,mp,cellLen-fliplr(x1)];
79        end
80      end
81
82      % Duration of simulation
83      function t = maxTime()
84        t = 1*365*24*60*60; % s
85      end
86
87      % Diffusion coefficients
88      %  T (K): temperature; scalar
89      function D = valD(T)
90        % Retrieve constants
91        R     = MPM_data.constR();
92        nCmps = MPM_data.cmpList();
93        dT    = (1/298-1/T)/R;
94
95        % Default to 1e-9 m^2/s
96        D = zeros(nCmps,1) + 1e-9;
97
98        % Data
99        D(MPM_data.UO2)  = 0.5e-09*exp(15000*dT);   % m^2/s
100       D(MPM_data.UCO3) = 0.5e-09*exp(15000*dT);   % m^2/s
101       D(MPM_data.Uads) = 0.1e-10*exp(15000*dT);   % m^2/s
```

```
102         D(MPM_data.Usus) = 0.1e-09*exp(15000*dT);   % m^2/s
103         D(MPM_data.CO3)  = 1.7e-09*exp(15000*dT);   % m^2/s
104         D(MPM_data.O2)   = 1.7e-09*exp(15000*dT);   % m^2/s
105         D(MPM_data.H2O2) = 1.7e-09*exp(15000*dT);   % m^2/s
106         D(MPM_data.Fe2)  = 0.5e-09*exp(15000*dT);   % m^2/s
107         D(MPM_data.UO3)  = 0.1e-10*exp(15000*dT);   % m^2/s
108         D(MPM_data.FeO)  = 0.1e-10*exp(15000*dT);   % m^2/s
109         D(MPM_data.H2)   = 4.5e-09*exp(15000*dT);   % m^2/s
110       end
111
112       % Molecular weights
113       function M = molWT()
114         % Retrieve constants
115         nCmps = MPM_data.cmpList();
116
117         % Default to 0.018 kg/mol
118         M = zeros(nCmps,1) + 0.018; % kg/mol
119
120         % Data
121         M(MPM_data.UO2)  = 0.302;   % kg/mol
122         M(MPM_data.UCO3) = 0.422;   % kg/mol
123         M(MPM_data.Uads) = 0.302;   % kg/mol
124         M(MPM_data.Usus) = 0.270;   % kg/mol
125         M(MPM_data.CO3)  = 0.060;   % kg/mol
126         M(MPM_data.O2)   = 0.032;   % kg/mol
127         M(MPM_data.H2O2) = 0.034;   % kg/mol
128         M(MPM_data.Fe2)  = 0.056;   % kg/mol
129         M(MPM_data.UO3)  = 0.322;   % kg/mol
130         M(MPM_data.FeO)  = 0.232;   % kg/mol
131         M(MPM_data.H2)   = 0.002;   % kg/mol
132       end
133
134       % Porosities
135       function epsl = poro()
136         % Data
137         epsl(1) = 0.45;   % U(VI) corrosion layer (Schoepite)
138         epsl(2) = 1.00;   % Water layer
139         epsl(3) = 0.10;   % Iron oxide corrosion layer
140       end
141
142       % Tortuosities
143       function tau = tort()
144         tau(1) = 0.10;   % U(VI) corrosion layer (Schoepite)  (ID = 1)
145         tau(2) = 1.00;   % Bulk water layer                   (ID = 2)
146         tau(3) = 0.10;   % Iron oxide corrosion layer         (ID = 3)
147       end
148
149       % Layer densities
150       function rho = dens()
151         rho(1) = 4980;   % kg/m^3; U(VI) corrosion layer (Schoepite)(ID = 1)
152         rho(2) = 1000;   % kg/m^3; Bulk water layer                 (ID = 2)
153         rho(3) = 5173;   % kg/m^3; Iron oxide corrosion layer       (ID = 3)
154       end
155
156       % Region identification
```

```
%  conc (M): concentrations; matrix
function [corr0,corrL] = regID(conc)
  % Retrieve constants
  MW    = MPM_data.molWT();
  epsl  = MPM_data.poro();
  rho   = MPM_data.dens();
  lmat  = MPM_data.ordVec();

  % Calculate corrosion products mass per square meter of surface area
   %U(VI) corrosion layer (Schoepite)
  tot0 = trapz(lmat,conc(MPM_data.UO3,:))*MW(MPM_data.UO3);[III] % kg/m^2
   % Iron oxide corrosion layer
  totL = trapz(lmat,conc(MPM_data.FeO,:))*MW(MPM_data.FeO); % kg/m^2

  % Determine layer thickness
   %U(VI) corrosion layer (Schoepite)
  corr0 = tot0/(rho(1)*(1-epsl(1)+eps));  % m
   % Iron oxide corrosion layer
  corrL = totL/(rho(3)*(1-epsl(3)+eps));  % m
end

% ANL-MPM V 1 Radiolysis model for peroxide generation
 %  t(s):    time; scalar
 %  conc(M): concentrations; matrix
function [rGen,dR] = alphRad(t,conc)
  % Retrieve constants
  lmat     = MPM_data.ordVec();
  corr0    = MPM_data.regID(conc);
  epsl     = MPM_data.poro();

  % Radiolysis data
  G    = 1.021e-4;   % generation value in mol/(J/kg)/m^3
  penD = 3.5e-5;     % alpha particle penetration distance in m
  Rf   = 1.0;[IV]      % Factor for dose from corrosion film

  % Empirical correlation of dose history from King and Kolar, 1999 p.74
  %xdat = [3.092e7  1.615e8  3.113e9  2.708e10 3.134e11 3.156e13];
  %ydat = [1.929e-2 1.313e-2 1.929e-2 1.051e-2 3.590e-3 6.504e-5];
  %dR = interp1(xdat,ydat,t,'linear','extrap');  % (J/kg)/s
  dR = 0.02 % Constant dose rate (J/kg)/s

  % Form factor calc, accounts effect of corrosion layer geometry on dose
  FF    = zeros(1,length(lmat));
  r1 = (lmat >=     0)&(lmat <=       penD);
  r2 = (lmat >= corr0)&(lmat <= corr0+penD);
  r3 = (lmat >      0)&(lmat <  corr0     );

  FF(r1) = FF(r1) + (  epsl(1));
  FF(r2) = FF(r2) + (1-epsl(1))*Rf;
  FF(r3) = FF(r3) + (  epsl(1))*Rf*2;
```

[III] ***Z = trapz(X,Y)*** computes the integral of Y with respect to X using trapezoidal integration. Inputs X and Y can be complex.

[IV] ***Rf*** is the dose ratio of U(VI) corrosion layer / fuel, based on assumption of incorporation of alpha emitting radionuclides into corrosion layer. Rf = 1.0 indicates dose from corrosion layer equivalent to dose from fuel (dR).

```
207
208         % Rate expression for peroxide generation
209         rGen = G*dR*FF;  % mol/m^3/s
210       end
211
212       % Temperature
213        %  t (s): time; scalar
214       function T = temp(t)
215         % Empirical correlation of T history from King and Kolar, 1999 p.39
216         %log_sec = min(log(t+3e5),29.5);
217         %a = [21.9  41.3    890    72.61  -488   ];
218         %b = [20.27 18.43    44.53 25.72    29.92];
219         %c = [ 1.46   3.267  35.98  4.027   11.41];
220         %T = sum(a.*exp(-((log_sec-b)./c).^2));      % K
221         T = 298.15 % constant temperature in K
222       end
223
224       % Saturation concentrations
225       %  T (K):    temperature; scalar
226       %  conc (M): concentrations; matrix
227       function Cs = satVals(T,conc)
228         % Retrieve constants
229         R           = MPM_data.constR();
230         nCmps       = MPM_data.cmpList();
231         nPts        = MPM_data.gridPts();
232         rho         = MPM_data.dens();
233         epsl        = MPM_data.poro();
234         dT          = (1/298-1/T)/R;
235
236         % Default to 10000 mol/m^3
237         Cs = zeros(nCmps,nPts) + 1e4;
238
239         % Data
240         Cs(MPM_data.UO2,:)   = 3.20e-2*exp(6e4*dT);          % mol/m^3
241         Cs(MPM_data.UCO3,:)  = 4.67e-2*exp(6e4*dT)*...
242                                (conc(MPM_data.CO3)^1.34);   % mol/m^3
243         Cs(MPM_data.Uads,:)  = 5.0e-3*(rho(3)*epsl(3));      % mol/m^3
244         Cs(MPM_data.Fe2,:)   = 1.00e-2*exp(6e4*dT);          % mol/m^3
245       end
246
247       % Initial concentrations
248       function Cinit = initVals()
249         % Retrieve constants
250         nCmps = MPM_data.cmpList();
251
252         % Default to 0.0 mol/m^3
253         Cinit = zeros(nCmps,1);
254
255         % Data
256         Cinit(MPM_data.CO3)  = 1.0e-6; % mol/m^3
257         Cinit(MPM_data.O2)   = 1.0e-6; % mol/m^3
258         Cinit(MPM_data.H2O2) = 0;      % mol/m^3
259         Cinit(MPM_data.UO3) = 0.5e+0;
260       end
261     end
```

```
262 end
```

This file: **MPM_react** defines the reactions and calculates the rates of reactions and corrosion potentials at the fuel surface (and container surface if included). Probably the most important file since the corrosion potential is the defining characteristic of the model. This file is called recursively when calculating the corrosion potential. This file also defines the reactions and calculates the rates of reactions that occur in the space between the fuel surface and end of domain (end of domain can be iron canister or bulk solution).

```
%Start MATLAB script

function [Rvec,sMat,oFun,dFun,Ecorr][V] = MPM_react(rSet,t,conc,Ecorr)

  % Initialize parameters
  [nCmps,nPts] = size(conc);
  sMat         = zeros(nCmps,0);
  oFun         = 0;
  dFun         = 0;

  % Retrieve constants
  F               = MPM_data.constF();
  R               = MPM_data.constR();
  T               = MPM_data.temp(t);

  % Calculate temperature dependence
  dT   = (1/298-1/T)/R;

  % Calculate reaction set
  switch(rSet)
    % Fuel surface reactions
    case 1
      % Calculate corrosion potential (recursive)
      if(nargin<4)[VI]
        Ecorr = 0; dEcorr = 1;
        [~,~,oFun,dFun] = MPM_react(rSet,t,conc,Ecorr);
        while(abs(dEcorr) > 1e-3 && abs(Ecorr) < 10)[VII]
          dEcorr = min(oFun/dFun,5);[VIII]
          oFunTest = 2*abs(oFun); dFunTest = 0; EcorrNew = 0;
          while(abs(oFunTest)>abs(oFun))

            EcorrNew  = Ecorr - dEcorr;
            [~,~,oFunTest,dFunTest] = MPM_react(rSet,t,conc,EcorrNew);
            dEcorr   = dEcorr/2;
          end
          Ecorr = EcorrNew;
          oFun = oFunTest; dFun = dFunTest;
        end
      end

```

[V] ***Rvec***: vector of reaction rates, ***sMat***: stoichiometry matrix, ***oFun***: objective function, ***dFun***: derivative function, ***Ecorr***: corrosion potential (Vsce)
[VI] ***nargin<4*** returns the number of input arguments passed in the call to the currently executing function.
[VII] ***abs(X)*** returns an array Y such that each element of Y is the absolute value of the corresponding element of X.
[VIII] ***C = min(A)*** returns the smallest elements along different dimensions of an array.

```
%**********************************%
% Start Fuel Reactions             %
%**********************************%

% One entry per half reaction at fuel surface
%   cMat:  number of electrons generated
%   kMat:  reaction rate constant (mol/m^3/s; or appropriate)
%   ectc:  electrochemical charge transfer coefficient
%   oMat:  reaction order in concentrations
%   sMat:  stoichiometry matrix
%   Ezero: standard potential (Vsce)

%**********************************%
% Fuel 1: Reaction A               %
% Fuel -> UO2(2+) + 2e(-)          %
%**********************************%

oMat                    = zeros(nCmps,1);
sMatAdd                 = zeros(nCmps,1);

cMat(1,1)  =  2;
kMat(1,1)  =  5.0e-8   *exp(6.0e4*dT);
ectc(1,1)  =  0.96;
Ezero(1,1) =  0.169 - 0.000248*(T-298);

sMatAdd(MPM_data.UO2)   =  1;
sMat                    = [sMat,sMatAdd];
cDep(1,1)               = prod(conc.^oMat);

%******************************************%
% Fuel 2: Reaction B                       %
% Fuel + 2CO3(2-) -> UO2(CO3)2(2-) + 2e(-)   %
%******************************************%

oMat                    = zeros(nCmps,1);
sMatAdd                 = zeros(nCmps,1);

cMat(2,1)  =  2;
kMat(2,1)  =  1.43e-12 *exp(6.0e4*dT);
ectc(2,1)  =  0.82;
Ezero(2,1) = -0.173 + 0.002100*(T-298);

sMatAdd(MPM_data.CO3)   = -2;
sMatAdd(MPM_data.UCO3)  =  1;
sMat                    = [sMat,sMatAdd];
oMat(MPM_data.CO3)      =  0.66;
cDep(2,1)               = prod(conc.^oMat);

%**********************************%
% Fuel 3: Reaction C               %
% H2O2 -> O2 + 2H(+) + 2e(-)       %
%**********************************%

oMat                    = zeros(nCmps,1);
sMatAdd                 = zeros(nCmps,1);
```

```
365
366         cMat(3,1)  =  2;
367         kMat(3,1)  =  7.4e-8   *exp(6.0e4*dT);
368         ectc(3,1)  =  0.41;
369         Ezero(3,1) = -0.121 - 0.000993*(T-298);
370
371
372         sMatAdd(MPM_data.H2O2)  = -1;
373         sMatAdd(MPM_data.O2)    =  1;
374         sMat                    = [sMat,sMatAdd];
375         oMat(MPM_data.H2O2)     =  1;
376         cDep(3,1)               = prod(conc.^oMat);
377
378         %*********************************%
379         % Fuel 4: Reaction D              %
380         % H2O2 + 2e(-) -> 2OH(-)          %
381         %*********************************%
382
383         oMat                    = zeros(nCmps,1);
384         sMatAdd                 = zeros(nCmps,1);
385
386         cMat(4,1)  = -2;
387         kMat(4,1)  =  1.2e-12  *exp(6.0e4*dT);
388         ectc(4,1)  = -0.41;
389         Ezero(4,1) =  0.973 - 0.000698*(T-298);
390
391         sMatAdd(MPM_data.H2O2) = -1;
392         sMat                   = [sMat,sMatAdd];
393         oMat(MPM_data.H2O2)    =  1;
394         cDep(4,1)              = prod(conc.^oMat);
395
396         %*********************************%
397         % Fuel 5: Reaction E              %
398         % O2 + 2H2O + 4e(-) -> 4OH(-)     %
399         %*********************************%
400
401         oMat                    = zeros(nCmps,1);
402         sMatAdd                 = zeros(nCmps,1);
403
404         cMat(5,1)  = -4;
405         kMat(5,1)  =  1.4e-12  *exp(6.0e4*dT);
406         ectc(5,1)  = -0.50;
407         Ezero(5,1) =  0.426 - 0.000123*(T-298);
408
409         sMatAdd(MPM_data.O2)    = -1;
410         sMat                    = [sMat,sMatAdd];
411         oMat(MPM_data.O2)       =  1;
412         cDep(5,1)               = prod(conc.^oMat);
413
414         %*********************************%
415         % End Fuel Reactions              %
416         %*********************************%
417
418         % Vector of reaction rates, objective function, derivative
```

```
      Rvec =           kMat.*cDep.*exp(ectc*F/R/T.*(Ecorr-Ezero));
      oFun = sum(cMat.*kMat.*cDep.*exp(ectc*F/R/T.*(Ecorr-Ezero)));
      dFun = sum(cMat.*ectc*F/R/T.*Rvec);

    % Container surface reactions
    case 2
      % Calculate corrosion potential (recursive)
      if(nargin<4)
        Ecorr = 9.5; dEcorr = 1;
        [~,~,oFun,dFun] = MPM_react(rSet,t,conc,Ecorr);
        while(abs(dEcorr) > 1e-3 && abs(Ecorr) < 10)
          dEcorr = min(oFun/dFun,5);
          oFunTest = 2*abs(oFun); dFunTest = 0; EcorrNew = 0;
          while(abs(oFunTest)>abs(oFun))
            EcorrNew  = Ecorr - dEcorr;
            [~,~,oFunTest,dFunTest] = MPM_react(rSet,t,conc,EcorrNew);
            dEcorr   = dEcorr/2;
          end
          Ecorr = EcorrNew;
          oFun = oFunTest; dFun = dFunTest;
        end
      end

      %**********************************%
      % Start Container Reactions        %
      %**********************************%

      % One entry per half reaction at container surface
      %   cMat:  number of electrons generated
      %   kMat:  reaction rate constant (mol/m^3/s; or appropriate)
      %   ectc:  electrochemical charge transfer coefficient
      %   oMat:  reaction order in concentrations
      %   sMat:  stoichiometry matrix
      %   Ezero: standard potential (Vsce)

      %**********************************%
      % Container 1: Reaction F          %
      % UO2(2+) + 2e(-) -> UO2           %
      %**********************************%

      oMat                    = zeros(nCmps,1);
      sMatAdd                 = zeros(nCmps,1);

      cMat(1,1)  = -2;
      kMat(1,1)  =  1.00e-9  *exp(6.0e4*dT);
      ectc(1,1)  = -0.50;
      Ezero(1,1) =  0.169 - 0.000248*(T-298);

      sMatAdd(MPM_data.UO2)   = -1;
      sMatAdd(MPM_data.Usus)  =  1;
      sMat                    = [sMat,sMatAdd];
      oMat(MPM_data.UO2)      =  1;
      cDep(1,1)               = prod(conc.^oMat);

      %**************************************%
```

```
474         % Container 2: Reaction G                  %
475         % UO2(CO3)2(2-) + 2e(-) -> UO2 + 2CO3(2-)   %
476         %*****************************************%
477
478         oMat                    = zeros(nCmps,1);
479         sMatAdd                 = zeros(nCmps,1);
480
481         cMat(2,1)  = -2;
482         kMat(2,1)  =  1.00e-10 *exp(6.0e4*dT);
483         ectc(2,1)  = -0.50;
484         Ezero(2,1) = -0.173 + 0.002100*(T-298);
485
486         sMatAdd(MPM_data.CO3)   =  2;
487         sMatAdd(MPM_data.Usus)  =  1;
488         sMatAdd(MPM_data.UCO3)  = -1;
489         sMat                    = [sMat,sMatAdd];
490         oMat(MPM_data.UCO3)     =  1;
491         cDep(2,1)               = prod(conc.^oMat);
492
493         %*********************************%
494         % Container 3: Reaction H          %
495         % H2O2 + 2e(-) -> 2OH(-)           %
496         %*********************************%
497
498         oMat                    = zeros(nCmps,1);
499         sMatAdd                 = zeros(nCmps,1);
500
501         cMat(3,1)  = -2;
502         kMat(3,1)  =  1.6e-14  *exp(6.0e4*dT);
503         ectc(3,1)  = -0.38;
504         Ezero(3,1) =  0.973 - 0.000698*(T-298);
505
506         sMatAdd(MPM_data.H2O2) = -1;
507         sMat                   = [sMat,sMatAdd];
508         oMat(MPM_data.H2O2)    =  1;
509         cDep(3,1)              = prod(conc.^oMat);
510
511         %*********************************%
512         % Container 4: Reaction I          %
513         % O2 + 2H2O + 4e(-) -> 4OH(-)      %
514         %*********************************%
515
516         oMat                    = zeros(nCmps,1);
517         sMatAdd                 = zeros(nCmps,1);
518
519         cMat(4,1)  = -4;
520         kMat(4,1)  =  3.2e-12  *exp(6.0e4*dT);
521         ectc(4,1)  = -0.42;
522         Ezero(4,1) =  0.426 - 0.000123*(T-298);
523
524         sMatAdd(MPM_data.O2)    = -1;
525         sMat                    = [sMat,sMatAdd];
526         oMat(MPM_data.O2)       =  1;
527         cDep(4,1)               = prod(conc.^oMat);
```

```
528
529         %*********************************%
530         % Container 5: Reaction J          %
531         % Container -> Fe(2+) + 2e(-)      %
532         %*********************************%
533
534         oMat                    = zeros(nCmps,1);
535         sMatAdd                 = zeros(nCmps,1);
536
537         cMat(5,1)  =  2;
538
539         % kMat(5,1)  =  2.2e-5*exp(10.4e4*dT);
540         kMat(5,1)  =  0.0;  % Turns off iron reactions
541
542         ectc(5,1)  =  1.08;
543         Ezero(5,1) = -0.650 + 0.000680*(T-298);
544
545         sMatAdd(MPM_data.Fe2)   =  1;
546         sMat                    = [sMat,sMatAdd];
547         cDep(5,1)               = prod(conc.^oMat);
548
549         %*********************************%
550         % Container 6: Reaction K          %
551         % H2O + e(-) -> (1/2)H2 + OH(-)    %
552         %*********************************%
553
554         oMat                    = zeros(nCmps,1);
555         sMatAdd                 = zeros(nCmps,1);
556
557         cMat(6,1)  = -1;
558         kMat(6,1)  =  1.2e-7   *exp(5.21e4*dT);
559         ectc(6,1)  = -0.48;
560         Ezero(6,1) = -0.802 - 0.001900*(T-298);
561
562         sMatAdd(MPM_data.H2)    =  1;
563         sMat                    = [sMat,sMatAdd];
564         cDep(6,1)               = prod(conc.^oMat);
565
566         %*********************************%
567         % End Container Reactions          %
568         %*********************************%
569
570         % Vector of reaction rates, objective function, derivative
571         Rvec =           kMat.*cDep.*exp(ectc*F/R/T.*(Ecorr-Ezero));
572         oFun = sum(cMat.*kMat.*cDep.*exp(ectc*F/R/T.*(Ecorr-Ezero)));
573         dFun = sum(cMat.*ectc*F/R/T.*Rvec);
574
575       % Bulk reactions
576       case 3
577         % Corrosion layer thicknesses
578         [~,corrL] = MPM_data.regID(conc);
579         lmat      = MPM_data.ordVec();
580         cellL     = MPM_data.cellLen();
581         cLL       = (lmat > (cellL-corrL));
```

```
582
583         % Saturation concentrations
584         Csat                  = MPM_data.satVals(T,conc);
585         Csat(MPM_data.Uads,:) = Csat(MPM_data.Uads,:).*cLL;
586
587         % Sub and supersaturation concentrations
588         concSub = max(Csat - conc,0);
589         concSup = max(conc - Csat,0);
590
591         %*********************************%
592         % Start Bulk Reactions            %
593         %*********************************%
594
595         % One entry per bulk reaction
596         %   kMat:     reaction rate constant (mol/m^3/s; or appropriate)
597         %   oMat:     reaction order in concentrations
598         %   oMatSub:  reaction order in subsaturation
599         %   oMatSup:  reaction order in supersaturation
600         %   sMat:     stoichiometry matrix
601
602         %*************************************%
603         % Bulk 1: Reaction 1                  %
604         % UO2(2+) + 2OH(-) + H2O -> UO3*2H2O  %
605         %*************************************%
606
607         oMat                  = zeros(nCmps,1);
608         oMatSub               = zeros(nCmps,1);
609         oMatSup               = zeros(nCmps,1);
610         sMatAdd               = zeros(nCmps,1);
611
612         kMat(1,1)  =   1e-3*exp(6.0e4*dT);
613
614         oMatSup(MPM_data.UO2) =  1;
615         sMatAdd(MPM_data.UO2) = -1;
616         sMatAdd(MPM_data.UO3) =  1;
617         sMat                  = [sMat,sMatAdd];
618         cDep(1,:)             = prod(conc.^   repmat(oMat,   1,nPts),1).*...
619                                 prod(concSub.^repmat(oMatSub,1,nPts),1).*...
620                                 prod(concSup.^repmat(oMatSup,1,nPts),1);
621
622         %*****************************************************%
623         % Bulk 2: Reaction 2f                                 %
624         % UO2(CO3)2(2-) + 2OH(-) + H2O -> UO3*2H2O + 2CO3(2-) %
625         %*****************************************************%
626
627         oMat                  = zeros(nCmps,1);
628         oMatSub               = zeros(nCmps,1);
629         oMatSup               = zeros(nCmps,1);
630         sMatAdd               = zeros(nCmps,1);
631
632         kMat(2,1)  =   1e-4*exp(6.0e4*dT);
633
634         oMatSup(MPM_data.UCO3) =  1;
635         sMatAdd(MPM_data.UCO3) = -1;
```

```
636         sMatAdd(MPM_data.CO3)   =  2;
637         sMatAdd(MPM_data.UO3)   =  1;
638         sMat                    = [sMat,sMatAdd];
639         cDep(2,:)               = prod(conc.^   repmat(oMat,   1,nPts),1).*...
640                                   prod(concSub.^repmat(oMatSub,1,nPts),1).*...
641                                   prod(concSup.^repmat(oMatSup,1,nPts),1);
642
643         %*****************************************************%
644         % Bulk 3: Reaction 2r                                 %
645         % UO3*2H2O + 2CO3(2-) -> UO2(CO3)2(2-) + 2OH(-) + H2O  %
646         %*****************************************************%
647
648         oMat                    = zeros(nCmps,1);
649         oMatSub                 = zeros(nCmps,1);
650         oMatSup                 = zeros(nCmps,1);
651         sMatAdd                 = zeros(nCmps,1);
652
653         kMat(3,1)  =   6.3e-12*exp(6.0e4*dT);
654
655         oMat(MPM_data.UO3)      =  1;
656         sMatAdd(MPM_data.UCO3)  =  1;
657         sMatAdd(MPM_data.CO3)   = -2;
658         sMatAdd(MPM_data.UO3)   = -1;
659         sMat                    = [sMat,sMatAdd];
660         cDep(3,:)               = prod(conc.^   repmat(oMat,   1,nPts),1).*...
661                                   prod(concSub.^repmat(oMatSub,1,nPts),1).*...
662                                   prod(concSup.^repmat(oMatSup,1,nPts),1);
663
664         %*****************************************%
665         % Bulk 4: Reaction 3                     %
666         % O2 + 4Fe(2+) + 8OH(-) -> 4H2O + 2Fe2O3  %
667         %*****************************************%
668
669         oMat                    = zeros(nCmps,1);
670         oMatSub                 = zeros(nCmps,1);
671         oMatSup                 = zeros(nCmps,1);
672         sMatAdd                 = zeros(nCmps,1);
673
674         kMat(4,1)  =   5.9e-1*exp(6.0e4*dT);
675
676         oMat(MPM_data.O2)      =  1;
677         oMat(MPM_data.Fe2)     =  1;
678         sMatAdd(MPM_data.O2)   = -1;
679         sMatAdd(MPM_data.Fe2)  = -4;
680         sMat                   = [sMat,sMatAdd];
681         cDep(4,:)              = prod(conc.^   repmat(oMat,   1,nPts),1).*...
682                                  prod(concSub.^repmat(oMatSub,1,nPts),1).*...
683                                  prod(concSup.^repmat(oMatSup,1,nPts),1);
684
685         %*****************************************%
686         % Bulk 5: Reaction 4                     %
687         % H2O2 + 2Fe(2+) + 4OH(-) -> 3H2O + Fe2O3  %
688         %*****************************************%
689
690         oMat                    = zeros(nCmps,1);
```

```
691         oMatSub                 = zeros(nCmps,1);
692         oMatSup                 = zeros(nCmps,1);
693         sMatAdd                 = zeros(nCmps,1);
694
695         kMat(5,1)   =    6.9e-2*exp(4.2e4*dT);
696
697         oMat(MPM_data.H2O2)     =  1;
698         oMat(MPM_data.Fe2)      =  1;
699         sMatAdd(MPM_data.H2O2) = -1;
700         sMatAdd(MPM_data.Fe2)  = -2;
701         sMat                    = [sMat,sMatAdd];
702         cDep(5,:)               = prod(conc.^   repmat(oMat,   1,nPts),1).*...
703                                   prod(concSub.^repmat(oMatSub,1,nPts),1).*...
704                                   prod(concSup.^repmat(oMatSup,1,nPts),1);
705
706         %*************************************************%
707         % Bulk 6: Reaction 5                              %
708         % UO2(2+) + 2Fe(2+) + 6OH(-) -> UO2 + 3H2O + Fe2O3  %
709         %*************************************************%
710
711         oMat                    = zeros(nCmps,1);
712         oMatSub                 = zeros(nCmps,1);
713         oMatSup                 = zeros(nCmps,1);
714         sMatAdd                 = zeros(nCmps,1);
715
716         kMat(6,1)   =    1.0e-2*exp(6.0e4*dT);
717
718         oMat(MPM_data.UO2)      =  1;
719         oMat(MPM_data.Fe2)      =  1;
720         sMatAdd(MPM_data.UO2)  = -1;
721         sMatAdd(MPM_data.Fe2)  = -1;
722         sMatAdd(MPM_data.Usus) =  1;
723         sMat                    = [sMat,sMatAdd];
724         cDep(6,:)               = prod(conc.^   repmat(oMat,   1,nPts),1).*...
725                                   prod(concSub.^repmat(oMatSub,1,nPts),1).*...
726                                   prod(concSup.^repmat(oMatSup,1,nPts),1);
727
728         %*****************************************************************%
729         % Bulk 7: Reaction 6                                              %
730         % UO2(CO3)2(2-) + 2Fe(2+) + 6OH(-) -> UO2 + 2CO3(2-) + 3H2O + Fe2O3 %
731         %*****************************************************************%
732
733         oMat                    = zeros(nCmps,1);
734         oMatSub                 = zeros(nCmps,1);
735         oMatSup                 = zeros(nCmps,1);
736         sMatAdd                 = zeros(nCmps,1);
737
738         kMat(7,1)   =    1.0e-3*exp(6.0e4*dT);
739
740         oMat(MPM_data.UCO3)     =  1;
741         oMat(MPM_data.Fe2)      =  1;
742         sMatAdd(MPM_data.UCO3) = -1;
743         sMatAdd(MPM_data.Fe2)  = -2;
744         sMatAdd(MPM_data.Usus) =  1;
745         sMatAdd(MPM_data.CO3)  =  2;
```

```
746         sMat                    = [sMat,sMatAdd];
747         cDep(7,:)               = prod(conc.^   repmat(oMat,   1,nPts),1).*...
748                                   prod(concSub.^repmat(oMatSub,1,nPts),1).*...
749                                   prod(concSup.^repmat(oMatSup,1,nPts),1);
750
751         %*****************************************************%
752         % Bulk 8: Reaction 7f                                 %
753         % Fe(2+) + 2OH(-) -> (1/3)Fe3O4 + (2/3)H2O + (1/3)H2  %
754         %*****************************************************%
755
756         oMat                    = zeros(nCmps,1);
757         oMatSub                 = zeros(nCmps,1);
758         oMatSup                 = zeros(nCmps,1);
759         sMatAdd                 = zeros(nCmps,1);
760
761         kMat(8,1)   =   1.0e-3*exp(6.0e4*dT);
762
763         oMatSup(MPM_data.Fe2) =  1;
764         sMatAdd(MPM_data.Fe2) = -1;
765         sMatAdd(MPM_data.FeO) =  1/3;
766         sMatAdd(MPM_data.H2)  =  1/3;
767         sMat                  = [sMat,sMatAdd];
768         cDep(8,:)             = prod(conc.^   repmat(oMat,   1,nPts),1).*...
769                                 prod(concSub.^repmat(oMatSub,1,nPts),1).*...
770                                 prod(concSup.^repmat(oMatSup,1,nPts),1);
771
772         %*****************************************************%
773         % Bulk 9: Reaction 7r                                 %
774         % (1/3)Fe3O4 + (2/3)H2O + (1/3)H2 -> Fe(2+) + 2OH(-)  %
775         %*****************************************************%
776
777         oMat                    = zeros(nCmps,1);
778         oMatSub                 = zeros(nCmps,1);
779         oMatSup                 = zeros(nCmps,1);
780         sMatAdd                 = zeros(nCmps,1);
781
782         % Depreciated - Unreasonably fast
783         %vct1 = 10^(-8.794 - 1254/T - 3725/T*(Ecorr_contain+0.241));
784         %vct2 = 10^( 4.575 - 3970/T - 1753/T*(Ecorr_contain+0.241));
785         %vct3 = 10^(-5.772 - 2298/T -  876/T*(Ecorr_contain+0.241));
786         %kMat(9,1)  =   (vct3 + (vct2*vct1)/(vct2+vct1))*1e4;
787
788         kMat(9,1)  = 0;
789
790         oMat(MPM_data.FeO)    =  1;
791         sMatAdd(MPM_data.Fe2) =  1;
792         sMatAdd(MPM_data.FeO) = -1/3;
793         sMatAdd(MPM_data.H2)  = -1/3;
794         sMat                  = [sMat,sMatAdd];
795         cDep(9,:)             = prod(conc.^   repmat(oMat,   1,nPts),1).*...
796                                 prod(concSub.^repmat(oMatSub,1,nPts),1).*...
797                                 prod(concSup.^repmat(oMatSup,1,nPts),1);
798
799         %*********************************%
800         % Bulk 10: Reaction 8f (adsorption) %
```

```
801         % UO2(2+) -> UO2(2+)                %
802         %***********************************%
803
804         oMat                    = zeros(nCmps,1);
805         oMatSub                 = zeros(nCmps,1);
806         oMatSup                 = zeros(nCmps,1);
807         sMatAdd                 = zeros(nCmps,1);
808
809         kMat(10,1)  =   1e-6*exp(0.0*dT);
810
811         oMat(MPM_data.UO2)      =  1;
812         oMatSub(MPM_data.Uads)  =  1;
813         sMatAdd(MPM_data.UO2)   = -1;
814         sMatAdd(MPM_data.Uads)  =  1;
815         sMat                    = [sMat,sMatAdd];
816         cDep(10,:)              = prod(conc.^   repmat(oMat,   1,nPts),1).*...
817                                   prod(concSub.^repmat(oMatSub,1,nPts),1).*...
818                                   prod(concSup.^repmat(oMatSup,1,nPts),1);
819
820         %***********************************%
821         % Bulk 11: Reaction 8r (desorption) %
822         % UO2(2+) -> UO2(2+)                %
823         %***********************************%
824
825         oMat                    = zeros(nCmps,1);
826         oMatSub                 = zeros(nCmps,1);
827         oMatSup                 = zeros(nCmps,1);
828         sMatAdd                 = zeros(nCmps,1);
829
830         % ANL: Slowed down desorption
831         %kMat(11,1)  =   1e-6*exp(0.0*dT);
832         kMat(11,1)  =   1e-9*exp(0.0*dT);
833
834         oMat(MPM_data.Uads)     =  1;
835         sMatAdd(MPM_data.UO2)   =  1;
836         sMatAdd(MPM_data.Uads)  = -1;
837         sMat                    = [sMat,sMatAdd];
838         cDep(11,:)              = prod(conc.^   repmat(oMat,   1,nPts),1).*...
839                                   prod(concSub.^repmat(oMatSub,1,nPts),1).*...
840                                   prod(concSup.^repmat(oMatSup,1,nPts),1);
841
842         %***********************************%
843         % Bulk 12: Reaction 9f (adsorption)   %
844         % UO2(CO3)2(2-) -> UO2(2+) + 2CO3(2-) %
845         %***********************************%
846
847         oMat                    = zeros(nCmps,1);
848         oMatSub                 = zeros(nCmps,1);
849         oMatSup                 = zeros(nCmps,1);
850         sMatAdd                 = zeros(nCmps,1);
851
852         kMat(12,1)  =   1e-9*exp(0.0*dT);
853
854         oMat(MPM_data.UCO3)     =  1;
855         oMatSub(MPM_data.Uads)  =  1;
```

```
856         sMatAdd(MPM_data.UCO3) = -1;
857         sMatAdd(MPM_data.Uads) =  1;
858         sMatAdd(MPM_data.CO3)  =  2;
859         sMat                   = [sMat,sMatAdd];
860         cDep(12,:)             = prod(conc.^   repmat(oMat,   1,nPts),1).*...
861                                  prod(concSub.^repmat(oMatSub,1,nPts),1).*...
862                                  prod(concSup.^repmat(oMatSup,1,nPts),1);
863
864         %************************************%
865         % Bulk 13: Reaction 9r (desorption)    %
866         % UO2(2+) + 2CO3(2-) -> UO2(CO3)2(2-)  %
867         %************************************%
868
869         oMat                   = zeros(nCmps,1);
870         oMatSub                = zeros(nCmps,1);
871         oMatSup                = zeros(nCmps,1);
872         sMatAdd                = zeros(nCmps,1);
873
874         % ANL: Changed to complex carbonate
875         % kMat(13,1)  =   1e-6*exp(0.0*dT);
876         kMat(13,1)  =   1e-9*exp(0.0*dT);
877
878         oMat(MPM_data.Uads)    =  1;
879         oMat(MPM_data.CO3)     =  1;
880         sMatAdd(MPM_data.UCO3) =  1;
881         sMatAdd(MPM_data.Uads) = -1;
882         sMatAdd(MPM_data.CO3)  = -2;
883         sMat                   = [sMat,sMatAdd];
884         cDep(13,:)             = prod(conc.^   repmat(oMat,   1,nPts),1).*...
885                                  prod(concSub.^repmat(oMatSub,1,nPts),1).*...
886                                  prod(concSup.^repmat(oMatSup,1,nPts),1);
887
888         %*********************************%
889         % End Bulk Reactions               %
890         %*********************************%
891
892         % Matrix of reation rates
893         Rvec = repmat(kMat,1,nPts).*cDep;
894
895       end
896
897     return
898
899
```

This file: **MPM_main**, is a top level function that calls all the other files needed to run the ANL-MPM V 1.

```
%Start MATLAB script

function [] = MPM_main()

  % Retrieves number of components
  nCmps = MPM_data.cmpList();

  % Simulation Parameters
  cellLen = MPM_data.cellLen();
  tmax    = MPM_data.maxTime();
  gridPts = MPM_data.gridPts();
  lmat    = MPM_data.ordVec();

  % Define initial conditions
  Cinit = MPM_data.initVals();
  X0    = repmat(Cinit,1,gridPts);

  % Group ordinary differential equation (ode) parameters
  params.cellLen = cellLen;
  params.gridPts = gridPts;
  params.nCmps   = nCmps;
  params.lmat    = lmat;

  % Set ordinary differential equation (ode) options
  opts = odeset('OutputFcn',@MPM_odestat,'NonNegative',1:(nCmps*gridPts));[IX]

  % User feedback
  clc;
  fprintf(['Running ','\n']);

  % Performs simulation
  [tvec,Xout] = ode15s(@MPM_odefun,[0.01 tmax],X0,opts,params);[X]

  % Reshapes output matrices
  tLen = length(tvec);
  Xmat = zeros(tLen,gridPts,nCmps);
  for k1 = 1:nCmps
    Xmat(:,:,k1) = Xout(:,k1:nCmps:nCmps*gridPts);
  end

  % Create output report file
  fprintf(['Writing Results','\n']);
  MPM_output(tvec,lmat,Xmat);
```

[IX] The ***odeset*** function lets the user adjust the integration parameters of the relevant ordinary differential equation solvers.

[X] ***[tvec,Xout] = ode15s(@MPM_odefun, [0.01 tmax], X0,opts,params)*** integrates the system of differential equations y′ = f(t,y) from time t0 to tf with initial conditions y0. Default integration parameters are replaced by property values specified in options, an argument created with the odeset function. The MATLAB ode15s solver is preferred for stiff problem types.

```
946
947     return
```

This file: **MPM_odefun** provides input arguments to the MATLAB ordinary differential equation solver.

```
%Start MATLAB script

function [dt] = MPM_odefun(t,X,params)[XI]

  % Unpack parameters
  nCmps   = params.nCmps;
  gridPts = params.gridPts;
  lmat    = params.lmat;

  % Reshape inputs/outputs
  X   = reshape(X,nCmps,gridPts);
  dt  = zeros(nCmps,gridPts);

  % Define finite difference ranges
  r1 = 1;              % Fuel boundary
  r2 = 2:gridPts-1;    % Bulk
  r3 = gridPts;        % Container boundary

  % Interval size to the right
  intVR          = zeros(1,gridPts);
  intVR(1:end-1) = lmat(2:end)-lmat(1:end-1);
  intVR(end)     = intVR(end-1);
  intVR          = repmat(intVR,nCmps,1);

  % Temperature, Diffusivity, Radiolysis
  T              = MPM_data.temp(t);
  D              = MPM_data.valD(T);
  rGen           = MPM_data.alphRad(t,X);

  % Initial Concentrations, Corrosion, Porosity
  Cinit          = MPM_data.initVals();
  [corr0,corrL]  = MPM_data.regID(X);
  epsl           = MPM_data.poro();

  % Reaction rates
  [Rvec0,   sMat0]    = MPM_react(1,t,X(:,1));
  [RvecL,   sMatL]    = MPM_react(2,t,X(:,end));
  [RvecBulk,sMatBulk] = MPM_react(3,t,X(:,:));

  % Flux bounday conditions
  dx0  = -epsl(2)*(sMat0*Rvec0)./(D+eps);
  dxL  =  epsl(2)*(sMatL*RvecL)./(D+eps);

  % Fixed boundary conditions
  dx0([3,4,8:11])   = 0;
  dxL(MPM_data.CO3) = (Cinit(MPM_data.CO3)-X(MPM_data.CO3,r3))/intVR(end);

  % Bulk reactions
```

[XI] ***odefun***: A MATLAB function handle that evaluates the right side of the differential equations. Provides input arguments to MATLAB ordinary differential equation solvers.

```
999     rRates = sMatBulk*RvecBulk;
1000
1001    % Second derivatives
1002    dx2     = zeros(nCmps,gridPts);
1003    dx2(:,r1) = ((X(:,r1+1)-X(:,r1))./intVR(:,r1) - ...
1004                  dx0                              )./ ...
1005                 (0.5*intVR(:,r1)                    );
1006    dx2(:,r2) = ((X(:,r2+1)-X(:,r2))./intVR(:,r2) - ...
1007                 (X(:,r2)-X(:,r2-1))./intVR(:,r2-1))./ ...
1008                 (0.5*intVR(:,r2) + 0.5*intVR(:,r2-1));
1009    dx2(:,r3) = (dxL                                - ...
1010                 (X(:,r3)-X(:,r3-1))./intVR(:,r3-1))./ ...
1011                 (                  0.5*intVR(:,r3-1));
1012
1013    % Diffusion contributions
1014    dt = dt + repmat(D,1,gridPts).*dx2;
1015
1016    % Reaction contributions
1017    dt = dt + rRates;
1018
1019    % Radiolysis Contributions
1020    dt(MPM_data.H2O2,:) = dt(MPM_data.H2O2,:) + rGen;
1021
1022    % Reshape outputs
1023    dt = reshape(dt,gridPts*nCmps,1);
1024
1025  return
1026
1027
```

This file: **MPM_odestat** provides the status function and reshape of matrix for the MATLAB ordinary differential equation solver.

```
%Start MATLAB script

function [status] = MPM_odestat(t,X,flag,~)

  % Continue evaluation
  status = 0;

  % Unpack parameters
  nCmps   = MPM_data.cmpList();
  gridPts = MPM_data.gridPts();

  % Reshape inputs/outputs
  if(isempty(flag))[XII]
    X   = reshape(X(:,1),nCmps,gridPts);[XIII]
  end

  % Output the time to the screen
  if(isempty(flag))
    clc;
    fprintf(['Running ','\n']);
    fprintf('Time (sec) = %12.5E \n',t(end));
  end

return
```

[XII] Determines whether array flag is empty. ***TF = isempty(A)*** returns logical 1 (true) if A is an empty array and logical 0 (false) otherwise. An empty array has at least one dimension of size zero.

[XIII] ***X = reshape(A,m,n)*** returns the m-by-n matrix B whose elements are taken column-wise from A. An error results if A does not have m*n elements.

This file: **MPM_output** writes data to a comma separated value file.

```
%Start MATLAB script

function [] = MPM_output(tvec,lmat,Xmat)

  % Retrieves number of components
  nCmps = MPM_data.cmpList();
  nTims = length(tvec);
  [~,cmpNames] = enumeration('MPM_data');

  % Unique file ID and file name
  numsec =  int64(now()*24*60)-2010*365.25*24*60;
  outfilename = strcat('MPM_run_', num2str(numsec),'.csv');

  % Write date information
  ofid = fopen(outfilename, 'a');
  fprintf(ofid,[datestr(now()),'\n\n']);
  fclose(ofid);

  % Write simulation times
  ofid = fopen(outfilename, 'a');
  fprintf(ofid,'Time (s),');
  fclose(ofid);
  dlmwrite(outfilename, tvec', '-append');

  % Write species concentrations
  for k1 = 1:nCmps
    ofid = fopen(outfilename, 'a');
    fprintf(ofid,['\n','x (m)',',',cmpNames{k1},' (mol/m^3),']);
    fprintf(ofid,'\n');
    fclose(ofid);
    dlmwrite(outfilename, [lmat',Xmat(:,:,k1)'], '-append');
  end

  % Write Diffusivities
  ofid = fopen(outfilename, 'a');
  fprintf(ofid,'\n');
  fprintf(ofid,'Diffusivities (m^2/s)\n');
  for k1 = 1:nCmps
    fprintf(ofid,[cmpNames{k1},',']);
    for k2 = 1:nTims
      eDif = MPM_data.valD(MPM_data.temp(tvec(k2)));
      fprintf(ofid,[num2str(eDif(k1)),',']);
    end
    fprintf(ofid,'\n');
  end
  fclose(ofid);

  % Write Saturation Concentrations
  ofid = fopen(outfilename, 'a');
  fprintf(ofid,'\n');
  fprintf(ofid,'Saturation (mol/m^3)\n');
  for k1 = 1:nCmps
    fprintf(ofid,[cmpNames{k1},',']);
```

```
    for k2 = 1:nTims
      tempK = MPM_data.temp(tvec(k2));
      eCs   = MPM_data.satVals(tempK,shiftdim(Xmat(k2,:,:))');
      fprintf(ofid,[num2str(eCs(k1,end)),',']);
    end
    fprintf(ofid,'\n');
  end
  fclose(ofid);

  % Write temperature information
  ofid = fopen(outfilename, 'a');
  fprintf(ofid,'\n');
  fprintf(ofid,'Other Time Series Data\n');
  fprintf(ofid,['Temp (K)',',']);
  for k1 = 1:nTims
    fprintf(ofid,[num2str(MPM_data.temp(tvec(k1))),',']);
  end
  fprintf(ofid,'\n');
  fclose(ofid);

  % Write radiolysis information
  ofid = fopen(outfilename, 'a');
  fprintf(ofid,['Rad (Gy)',',']);
  for k1 = 1:nTims
    [~,dR] = MPM_data.alphRad(tvec(k1),shiftdim(Xmat(k1,:,:))');
    fprintf(ofid,[num2str(dR),',']);
  end
  fprintf(ofid,'\n');
  fclose(ofid);

  % Write corrosion layer thickness information
  ofid = fopen(outfilename, 'a');
  fprintf(ofid,['Corrosion 1 (m)',',']);
  for k1 = 1:nTims
    [corr0,~] = MPM_data.regID(shiftdim(Xmat(k1,:,:))');
    fprintf(ofid,[num2str(corr0),',']);
  end
  fprintf(ofid,'\n');
  fprintf(ofid,['Corrosion 2 (m)',',']);
  for k1 = 1:nTims
    [~,corrL] = MPM_data.regID(shiftdim(Xmat(k1,:,:))');
    fprintf(ofid,[num2str(corrL),',']);
  end
  fprintf(ofid,'\n');
  fclose(ofid);

  % Write corrosion potentials
  ofid = fopen(outfilename, 'a');
  fprintf(ofid,['Surf 1 Potential (V)',',']);
  for k1 = 1:nTims
    [~,~,~,~,eEp] = MPM_react(1,tvec(k1),shiftdim(Xmat(k1,1,:)));
    fprintf(ofid,[num2str(eEp),',']);
  end
  fprintf(ofid,'\n');
  fprintf(ofid,['Surf 2 Potential (V)',',']);
  for k1 = 1:nTims
```

```
      [~,~,~,~,eEp] = MPM_react(2,tvec(k1),shiftdim(Xmat(k1,end,:)));
      fprintf(ofid,[num2str(eEp),',']);
    end
    fprintf(ofid,'\n');
    fclose(ofid);

    % Write region densities
    ofid = fopen(outfilename, 'a');
    fprintf(ofid,'\n');
    eRho = MPM_data.dens();
    fprintf(ofid,'Density (kg/m^3)\n');
    fprintf(ofid,['Layer 1',',',num2str(eRho(1)),'\n']);
    fprintf(ofid,['Bulk   ',',',num2str(eRho(2)),'\n']);
    fprintf(ofid,['Layer 2',',',num2str(eRho(3)),'\n']);
    fclose(ofid);

    % Write region porosities
    ofid = fopen(outfilename, 'a');
    fprintf(ofid,'\n');
    ePor = MPM_data.poro();
    fprintf(ofid,'Porosities\n');
    fprintf(ofid,['Layer 1',',',num2str(ePor(1)),'\n']);
    fprintf(ofid,['Bulk   ',',',num2str(ePor(2)),'\n']);
    fprintf(ofid,['Layer 2',',',num2str(ePor(3)),'\n']);
    fclose(ofid);

    % Write region tortuosities
    ofid = fopen(outfilename, 'a');
    fprintf(ofid,'\n');
    eTor = MPM_data.tort();
    fprintf(ofid,'Tortuosities\n');
    fprintf(ofid,['Layer 1',',',num2str(eTor(1)),'\n']);
    fprintf(ofid,['Bulk   ',',',num2str(eTor(2)),'\n']);
    fprintf(ofid,['Layer 2',',',num2str(eTor(3)),'\n']);
    fclose(ofid);

return
```