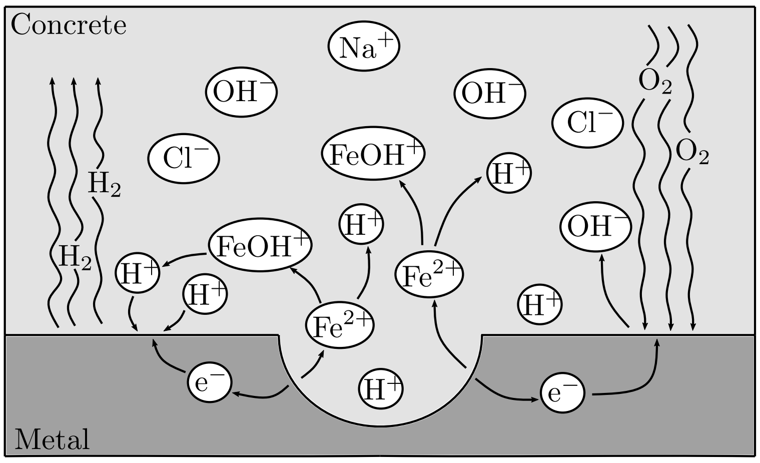

Figure 1: Overview of the system simulated using the presented C++ code

ConcreteCorrosion: A C++ code to simulate natural corrosion of rebar within concrete.

Tim Hageman[a,*], Emilio Martínez-Pañeda[a], Carmen Andrade[b]

[a] *Department of Engineering Science, University of Oxford, Oxford OX1 3PJ, UK*
[b] *International Center of Numerical Methods in Engineering (CIMNE), Madrid 28010, Spain*

Abstract

Documentation that accompanies the *C++* code ConcreteCorrosion, available from here. This documentation explains how to compile the code, the usage of the implemented finite element framework, and highlight the main files. The code allows the simulation of a corrosion pit with a supporting passivated surface, applied to corrosion of rebar within concrete. Specifically, it allows for an in-depth investigation of the locations of the cathodic and anodic reactions, under the requirement that the sum of these reaction currents acting on the complete rebar needs to be zero. If using (part of) this code, please cite T Hageman, C Andrade, and E Martinez-Pañeda. *Corrosion of metal reinforcements within concrete and localisation of supporting reactions under natural conditions.* Acta Electrochemica [1].

Keywords: Corrosion, Electrochemistry, natural corrosion, Numerical simulation

*Corresponding author
Email address: `tim.hageman@eng.ox.ac.uk` (Tim Hageman)

Contents

1 Introduction **1**
- 1.1 Installation . . . 2
  - 1.1.1 Required external libraries . . . 2
  - 1.1.2 Compiling . . . 2
- 1.2 Basic usage . . . 2
- 1.3 Input Files . . . 2
  - 1.3.1 Mesh . . . 3
  - 1.3.2 Chemical species . . . 3
  - 1.3.3 Bulk Reactions . . . 4
  - 1.3.4 Surface Reactions . . . 4
  - 1.3.5 Degrees of freedom . . . 5
  - 1.3.6 Solver settings . . . 5
  - 1.3.7 Time discretisation options . . . 5
  - 1.3.8 Outputs . . . 6
  - 1.3.9 Physics models . . . 6
  - 1.3.10 Visualisation . . . 7

2 Mesh Generation **7**
- 2.1 Mesh file format . . . 9

3 Summary of included models **9**
- 3.1 Nernst-Planck equations . . . 9
- 3.2 Surface reactions & charge conservation . . . 10
- 3.3 Boundary conditions . . . 10

4 Output files and post-processing **11**
- 4.1 Output files . . . 11
- 4.2 Post-processing . . . 12

5 References **14**

1. Introduction

While reinforcements within concrete play a crucial role in bearing loads and enhancing the structural performance of buildings and infrastructure, they are inherently vulnerable to corrosion when exposed to aggressive environmental conditions. This susceptibility can lead to the development of internal stresses within the concrete matrix [2, 3], as the expansive nature of corrosion products exerts pressure on the surrounding material. As corrosion progresses, it contributes to the formation of local fractures [4–6], which can compromise the load-carrying capacity of the structure. Additionally, the bond between the steel reinforcement and the adjacent concrete may deteriorate—a process known as de-bonding—which further diminishes the structural integrity [7–10]. Initially, the concrete's internal environment is highly alkaline due to the presence of calcium hydroxide in the pore solution, which facilitates the formation of a protective passivation layer on the steel surface [2, 11]. This layer effectively prevents corrosion under normal conditions. However, in the presence of chloride ions—often introduced through deicing salts, marine environments, or contaminated aggregates—the passivation film can be disrupted, allowing for the initiation of localized pitting corrosion [12–15].

Once a corrosion pit has formed, the chemical environment within the pit changes drastically. The hydrolysis of metal cations results in acidification of the local environment, further accelerating the corrosion process. Even if the original source of chloride ions is later removed, repassivation is unlikely due to the persistently low pH conditions inside the pit [16–20]. This localized breakdown of the protective layer means that anodic and cathodic reactions occur in spatially separated regions on the steel surface, with the anodic sites (where metal dissolution occurs) being particularly confined and aggressive. Surrounding these active corrosion zones, cathodic reactions consume the electrons generated by the anodic dissolution, thereby maintaining the electrochemical circuit and influencing the steel's overall potential [21–25]. Over time, the progressive growth of corrosion pits reduces the effective cross-sectional area of the reinforcing bars, thereby compromising their mechanical performance and increasing the likelihood of structural failure. Furthermore, the accumulation and expansion of corrosion products within the concrete create significant internal tensile stresses. These stresses induce cracking and spalling of the concrete cover, which not only weakens the structure but also exposes more steel surface to corrosive agents. As a result, the durability and service life of reinforced concrete structures are substantially diminished [26].

Therefore, it is crucial to establish the changes in the local environment, the range over which these changes are induced, and how long these changes are being sustained by the presence of a corrosion pit under realistic circumstances. This code focusses on these aspects, assuming a constant corrosion pit geometry, and resolving how the corrosion currents change over time. It also incorporates the requirement that these corrosion currents are supported by hydrogen and oxygen evolution reactions. Here, we present the C++ implementation of a new computational model to estimate the corrosion of reinforcements within concrete, under the requirement that the corrosion current is fully supported by hydrogen and oxygen evolution reactions. This requirement emulates natural corrosion, where the rebar is fully isolated from any external current and potential sources. As this creates a direct link between the corrosion rate and the cathodic reaction rates, this model allows for in-depth analysis of the processes causing corrosion in natural conditions. We furthermore do not impose any specific reaction areas, rather letting the computational model resolve these by itself, to study which areas of the rebar partake in the electrochemical reactions.

These two important and novel model features allow us to gain new physical insight and answer the pressing questions formulated above. In the remainder of this documentation, we will first describe how to install and run the simulation code, followed by a description of the mesh generation and the input files. We will then describe the models implemented in the code, and finally describe the output files generated during the simulation. For a full description of our model, we refer to T Hageman, C Andrade, and E Martinez-Pañeda. *Corrosion of metal reinforcements within concrete and localisation of supporting reactions under natural conditions.* Acta Electrochemica [1].

1.1. *Installation*

1.1.1. *Required external libraries*

The simulation code has been developed to work directly in ubuntu, or within windows using WSL [27]. A script is included which installs all required libraries, *Install_Libs.sh*. While it can be directly run to install all libraries at once, it is recommended that the contents is run line-by-line by copying it into the terminal. This allows for easier debugging in case of errors. The script installs the following libraries:
Intel OneMKL [28] is used for its linear matrix solvers (and used by PETSc).
Eigen [29] is used for small matrix and vector mutliplications.
RapidJSON [30] is used for reading the input files.
VTK [31] is used for visualisation during simulations.
Highfive [32] is used to write output files in the HDF5 format.
PETSc [33] is used for its implementation of parrallel (MPI) matrices and vectors (and their associated solvers).

Most of the above libraries are easy to install. However, for PETSc care needs to be taken to enable support for C++, link to the Intel MKL library, and enable support for HDF5. This is enabled through the command:

Install_Libs.sh

```
PETSC_ARCH=optimised
./configure PETSC_ARCH=optimised --with-scalapack-include=$MKLROOT/include --with-
    scalapack-lib="-L$MKLROOT/lib/intel64 -lmkl_scalapack_lp64 -
    lmkl_blacs_intelmpi_lp64" --with-mpi-dir=$I_MPI_ROOT --with-mkl=$MKLROOT --with-
    blas-lapack-dir=$MKLROOT --with-mkl_pardiso-dir=$MKLROOT --with-mkl_cpardiso-dir=$
    MKLROOT --download-hdf5 --download-mumps --with-clanguage=cxx --with-debugging=0 --
    COPTFLAGS='-O3' --FOPTFLAGS='-O3' --CXXOPTFLAGS='-O3'
```

In addition to the environmental variables set by OneMKL and PETSc ($PETSC_DIR, $ PETSC_ARCH, $MKLROOT), an additional environmental variable is used in the cmake files to locate the other libraries, $LIBRARY_DIR, which should point to the parent folder that contains the eigen, rapidjson, and Highfive folders.

1.1.2. *Compiling*

The code makes uses of cmake to create appropriate makefiles. To compile the code, first run cmake using *cmake .* from the base directory, after which the code itself can be compiled using *make* (or *make -j 56* to compile in parrallel).

1.2. *Basic usage*

Once compiled, the code can be run using *./ConcreteCorrosion <inputfile>* to run on a single core, or *mpirun -n <number of cores> ./ConcreteCorrosion <inputfile>* to run in parallel. Example input files are included within the folder *TestCases/Electrochemistry*. Running the code using the example file will print the progress of the simulation to the terminal, and open several plots showing the state of the system, with an example shown in Fig. 2. Note that while the simulation considers a three-dimensional domain, plots are only produced on 2D edges of the domain. The code will also save the results to a file, which can be visualised using the *PlotData.m* matlab script.

1.3. *Input Files*

The input file contains the following sections:

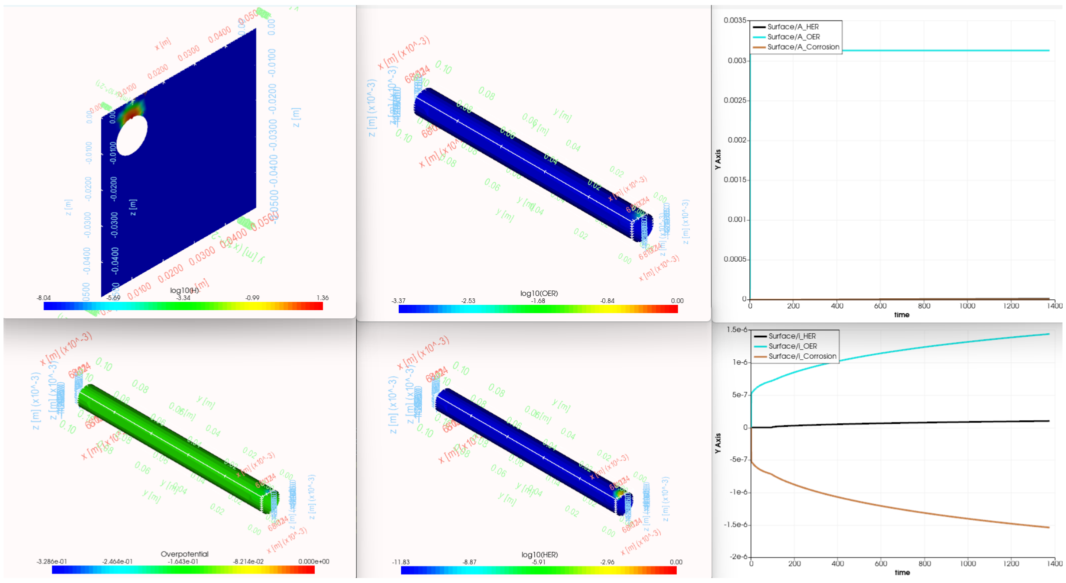

Figure 2: Example of the plots produced during the simulation. The top left shows the H^+ concentration, the top middle the oxygen evolution reaction current, top right the areas involved in the surface reactions. The bottom left shows the overpotential, middle the hydrogen evolution reaction rate, and the bottom right shows the reaction rates for each surface reaction.

1.3.1. Mesh

TestCases/ElectroChemistry/Test_ElectroChem.json

```
    "mesh":{
        "dim":3,
        "file":"TestCases/ElectroChemistry/Mesh/Beam_32c.h5",
        "ipcount1D": 3
    },
```

which defines the file to use as mesh (see also Section 2), indicates that the mesh is 3D, and sets the number of integration points per dimension to 3 (e.g. 3x3x3 integration points for volume elements, 3x3 for surface elements).

1.3.2. Chemical species

Next, the chemical species are defined via:

TestCases/ElectroChemistry/Test_ElectroChem.json

```
    "properties":{
        "Species":{
            "Types": ["H", "OH", "Fe", "FeOH", "Na", "Cl", "O2"],
            "UseChemPotentials": false,
            "H":{
                "D": 9.3e-9,
                "z": 1,
                "C0": 1.0e-8,
                "Aqueous": true
            },
```

First defining the names of the chemical species present, and then defining their diffusivity (D), ionic charge (z, this can be set to zero), the initial concentration (C0), and whether the diffusivity and concentration should be corrected for the water saturation (Aqueous). Note that the amount of degrees of freedom is automatically adapted to match the number of species declared here. As such, extra species can be freely added or removed within needing to update the simulation code itself.

1.3.3. Bulk Reactions

Chemical reactions occurring within the bulk are defined by:

TestCases/ElectroChemistry/Test_ElectroChem.json
```
        "VolumeReactions":{
            "Reactions": ["Auto-Ionisation","Corrosion1","Corrosion2"],
            "Auto-Ionisation":{
                "Type": "Equilibrium",
                "K": 1.0e-14,
                "k_dummy": 1.0e7,
                "Species_In":[],
                "n_In":[],
                "Species_Out":["H","OH"],
                "n_Out":[1,1],
                "C_ref": 1.0e3,
                "Lumped": true
            },
            "Corrosion1":{
                "Type": "Dynamic",
                "k": [1.0e1, 1.0e1],
                "Species_In": ["Fe"],
                "n_In": [1],
                "Species_Out": ["FeOH","H"],
                "n_Out": [1,1],
                "C_ref": 1.0e3,
                "Lumped": true
            },
```

First defining names for all reactions that are considered in the bulk (this list can be expanded to include mode reactions), and then defining the parameters for each individual reaction. This includes defining whether the reaction is assumed to be either solved as a dynamic or equilibrium reaction, the species involved in the reaction (this can be left empty if water or other unresolved species are involved), and their respective stoichiometric coefficients. For equilibrium reactions, this furthermore requires setting the equilibrium constant (K), and a dummy rate k_dummy to be chosen high enough to enforce equilibrium. For dynamic reactions, the forward and backward rates k are defined. Finally, a flag used to indicate whether a lumped integration scheme should be used to stabilize the reactions is included [34].

1.3.4. Surface Reactions

Surface reactions are defined in a similar way as bulk reactions:

TestCases/ElectroChemistry/Test_ElectroChem.json
```
        "SurfaceReactions":{
            "Reactions": ["HER","OER","Corrosion"],
            "HER":{
                "Reaction": "2H+ + 2e- <-> H2",
                "Type": "Electrochemical",
                "i0": [1.0e-2, 0.0],
                "alpha": 0.5,
                "E_eq": 0.0,
                "electrons_In": 2,
                "Species_In":["H"],
                "n_In":[2],
                "Species_Out":[],
                "n_Out":[],
                "C_ref": 1.0e3,
                "Surface": "Cathode"
            },
```

Defining the type of surface reaction, the reference reaction currents i0, the charge transfer coefficient alpha, the equilibrium potential E_eq, the species and electrons involved as input and output, and the surface this reaction occurs on (used by later modules).

1.3.5. Degrees of freedom

This section defines the degree of freedom names (these should include the species declared earlier), using ePot to declare the electrolyte potential, and Em to declare the metal potential:

TestCases/ElectroChemistry/Test_ElectroChem.json

```
    "Dofs":{
        "DofNames": ["H", "OH", "Fe", "FeOH", "Na", "Cl", "O2", "ePot","Em"],
        "DofStep":  [0,   0,    0,    0,      0,     0,    0,     0,     0]
    },
```

While it is possible to declare that species are resolved in a staggered manner (by setting DofStep to different valeus), this will cause convergence issues and is not recommended.

1.3.6. Solver settings

The solver settings are defined in the following section:

TestCases/ElectroChemistry/Test_ElectroChem.json

```
    "nonlinSolver":{
        "max_it": 30,
        "max_outer_it": 1,
        "convCrit": [1.0e-3, 1.0e10, 1.0e-6],
        "tiny": [1.0e-3, 1.0e-99, 1.0e-14],
        "linesearch": true,
        "linesearchLims": [0.25, 1],
        "LinSolver":{
            "Type": "Pardiso",
            "IOptionIdx": [],
            "IOptionVals": []
        },
        "Initialization":{
            "Type": "ElectroChemistry",
            "InteriorGroup": "Interior"
        }
    },
```

Defining the maximum amount of Newton-Raphson iterations (max_it), the convergence criteria used (convCrit, with the three components corresponding to a criterion based on the norm of concentrations, forces, and energy respectively), whether a line search algorith is used, and which linear solver is used to solve the coupled system of equations (either *Pardiso*, *MUMPS*, or *GMRES*). Finally, this section declares that the degrees of freedom are initialised consistent with the C0 declared in the species section.

1.3.7. Time discretisation options

This section declares the time discretisation options:

TestCases/ElectroChemistry/Test_ElectroChem.json

```
    "TimeSolver":{
        "dt": 1.0,
        "tmax": 2419200.0,
        "outputN": 10,
        "dtGrow": 1.05,
        "dtMax": 3600.0
    },
```

detailing the initial time increment dt, the rate at which this time increment increases (dtGrow), the maximum time increment (dtMax), and the total duration of the simulation (tmax). This also declares how often output files should be saved, through the outputN parameter.

1.3.8. Outputs

The output section defines the variables to be written to the output files for later post-processing/visualisation:

TestCases/ElectroChemistry/Test_ElectroChem.json

```
"Outputs":{
    "OutputMeshOnlyOnce": true,
    "SaveFolder":"Results",
    "ElementGroups": ["Interior","Pit","Bar"],
    "Interior":{
        "Nodes": ["H", "OH", "Fe", "FeOH", "Na", "Cl", "O2", "ePot"],
        "IP": []
    },
    "Pit":{
        "Nodes": ["H", "OH", "Fe", "FeOH", "Na", "Cl", "O2", "ePot", "HER", "OER", "
            Corrosion"],
        "IP": []
    },
    "Bar":{
        "Nodes": ["H", "OH", "Fe", "FeOH", "Na", "Cl", "O2", "ePot", "HER", "OER", "
            Corrosion"],
        "IP": []
    }
},
```

1.3.9. Physics models

This section declares the models and boundary conditions to be used (see also Section 3 for more details):

TestCases/ElectroChemistry/Test_ElectroChem.json

```
"Models":{
    "Names":["Surface","NernstPlanck","Boundary1","Boundary2"],
    "NernstPlanck":{
        "_Comment": "Diffusion-electromigration",
        "Name": "Electrochemistry/NernstPlanck",
        "ElementGroup_C": "Interior",
        "ElementGroup_E": "Interior",
        "Porosity": 0.05,
        "PorosityFactor": 1.5,
        "Sw": 1.0,
        "S_irr": 0.2,
        "SatFactor": 2.0,
        "Stabilisation": "None"
    },
    "Surface":{
        "_Comment": "Electrochemical surface reactions",
        "Name": "Electrochemistry/ElectroSurface",
        "AreaGroups_C": ["Pit", "Bar"],
        "AreaGroups_E": ["Pit", "Bar"],
        "SurfaceReactions": [["Cathode","Anode"],["Cathode"]],
        "ChargeConservation": true,
        "E_m": 0.0,
        "ActiveCurrent_Threshold": 1.0e-5
    },
    "Boundary1":{
        "Name": "Electrochemistry/EchemConstraints",
        "NodeGroup_C":"Left",
        "NodeGroup_E":"Left",
        "Species": ["H", "OH", "Fe", "FeOH", "Na", "Cl", "O2"],
        "ePot": 0.0
    },
    "Boundary2":{
        "Name": "Electrochemistry/EchemConstraints",
        "NodeGroup_C":"Top",
        "NodeGroup_E":"Top",
        "Species": ["H", "OH", "Fe", "FeOH", "Na", "Cl", "O2"],
        "ePot": 0.0
    }
},
```

with the Name field detailing which physics model to use.

For the NernstPlanck model, it also requires the element groups for the concentrations and electrolyte potential to be defined, the porosity of the concrete to be set (Porosity), the tortuity coefficient (PorosityFactor), the saturation and saturation limits (Sw and S_irr respectively), the exponent used for the saturation-diffusion relation (SatFactor), and if any numerical stabilisation is used.

For the Surface reactions, this requires setting the element groups that represent the surface. Additionally, for each area indicated as surface, SurfaceReactions needs to be set to define which reactions occur on this surface (using the same names as used when these reactions were first defined). This model also allows toggling the charge conservation on and off, and sets a threshold value (solely used for post-processing) to indicate when a surface reaction is considered to be negligible (ActiveCurrent_Threshold). This threshold is used to calculate the areas associated with each reaction, but is not used to impose any constraints on the reaction rates themselves.

For the boundary condition models, this requires setting which species are imposed by the boundary condition, and what the electric potential at the boundary is.

1.3.10. Visualisation

Finally, during the simulation, optional plots are produced as defined in visualisation:

TestCases/ElectroChemistry/Test_ElectroChem.json

```
    "Visualisation":{
        "Plots": ["O2","H","OER","HER","OPot","plot","plot2","plot3"],
        "O2":{
            "Type": "Surf_3D",
            "Dof": "O2",
            "ElementGroup": "Symmetry",
            "Frequency": "step"
        },
```

Indicating the degree of freedom to plot, the surface where to plot this for, and how often the plots are updated. Note that, for computational efficiency, the plots produced during the simulation are non-interactive.

2. Mesh Generation

Meshes used for the simulations are generated in MATLAB before running the simulation itself. AN example of mesh generation is given in *TestCases/ElectroChemistry/Mesh/Beam.m*, which generates the mesh used for all the results in our paper.

The initial geometry is defined as a .stl file, and imported into MATLAB via

TestCases/ElectroChemistry/Mesh/Beam.m

```
    model = createpde(1);
    importGeometry(model,"Geo.stl");
```

This geometry is then used to create a basic mesh, defining the maximum element sizes in the pit, and on the surface of the rebar:

TestCases/ElectroChemistry/Mesh/Beam.m

```
    model = generateMesh(model,'Hgrad',1.2,"Hmax",dxMax,"Hface",{7,dxPit,6,dxBar});
```

To make this mesh suitable for finite element simulations, separate element groups are extracted for the interior elements and the boundaries:

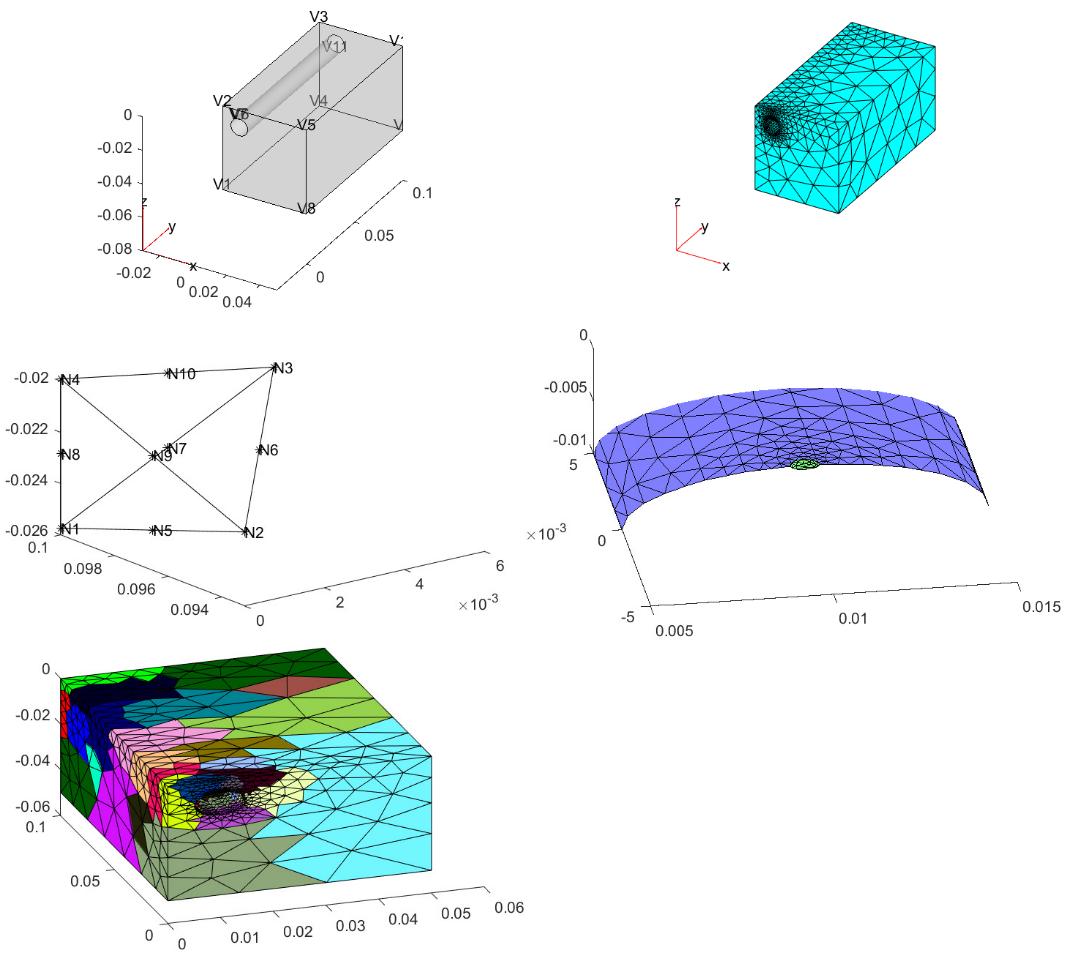

Figure 3: Example of a mesh generated using the MATLAB script *Beam.m*. The figure shows the geometry (top left), generated mesh (top right) and a close-up of the mesh near the corrosion pit (middle right), the node numbering for volume elements (middle left), and how the mesh is partitioned and distributed between cpu cores (bottom).

TestCases/ElectroChemistry/Mesh/Beam.m

```
        Elementgroups{1}.Name = "Interior";
        Elementgroups{1}.Type = "T3_10B";
        Elementgroups{1}.Elements = model.Elements(:,findElements(model,'region','Cell',1))
            ';
```

Finally, this mesh is partitioned to be compatible with the required number of cpu cores, and saved to a file:

TestCases/ElectroChemistry/Mesh/Beam.m

```
    [Nodes, Nodegroups, Elementgroups, coreData] = partition(Nodes, Nodegroups,
        Elementgroups, nCores);

    nexttile
    PlotPartition(Nodes,Elementgroups,coreData);

    drawnow();
    saveToHDF(fname, Nodes, Nodegroups, Elementgroups, coreData)
```

Running this MATLAB script also plots the mesh, producing the figure shown in Fig. 3.

2.1. *Mesh file format*

The saved output file has the following entries:

- NCores: number of CPU cores this mesh is generated for
- nodes: 3xn matrix containing the coordinates of all the nodes
- nodegroupnames: Names for all node groups used
- nodegroups/<name>: Vector containing the indices of all nodes associated with this groups
- elementgroupnames: Names for all element groups used
- elementgrouptypes: Types of all element groups used, using *T3_ 10B* to indicate tetrahedral, quadratic, Bernstein volume elements, and *T2_ 6B* to indicate triangular, quadratic, Bernstein surface elements
- elementgroups/<name>: mxn matrix containing the nodes m associated with each element n.
- <core>/noderange: Indicates the node ownership range of this cpu core.
- <core>/ghosts: Indicates the nodes which are not owned by the core, but are required to be synced with neighbouring cores.
- <core>/nodegrouprange: Vector indicating the start and end for each nodegroup owned by this cpu core
- <core>/Haselems: Vector indicating whether this core contains any elements for each elementgroup
- <core>/elemrange: Vector indicating the start and end for each elementgroup owned by this cpu core

3. Summary of included models

3.1. *Nernst-Planck equations*

This model is implemented in *Models/Electrochemistry/NernstPlanck.cpp.* It implements the conservation of chemical species using the Nernst-Planck equation:

$$S^*_\pi \phi \dot{C}_\pi + \boldsymbol{\nabla} \cdot \left(-D^{\text{eff}}_\pi \boldsymbol{\nabla} C_\pi\right) + \frac{z_\pi F}{RT} \boldsymbol{\nabla} \cdot \left(-D^{\text{eff}}_\pi C_\pi \boldsymbol{\nabla} \varphi\right) + \phi S^* R_\pi = 0 \tag{1}$$

where S^* is the water saturation, ϕ the porosity, C_π is the concentration of species π, D^{eff}_π is the effective diffusivity of species π, z_π is the ionic charge of species π, F is Faraday's constant, R is the universal gas constant, T is the temperature, and φ is the electric potential. The term R_π represents the reaction rate of species π. The effective diffusivity is defined in terms of the saturation and porosity as [35, 36]:

$$D^{\text{eff}}_\pi = \phi^{3/2} D_\pi \left(\frac{S_{\text{w}} - S_{\text{irr}}}{1 - S_{\text{irr}}}\right)^2 \tag{2}$$

where the exponents 3/2 and 2 correspond to the factors set in the input file for *PorosityFactor* and *SatFactor* respectively.

Eq. (1) is combined with the electroneutrality condition:

$$\sum_\pi \phi z_\pi C_\pi = 0 \tag{3}$$

with this condition allowing the electrolyte potential φ to be obtained, implicitly taking into account the local diffusivity and ion concentrations (and thus the local conductivity) [37].

For the volume reactions included in the R_π term, this considers dynamic reactions and equilibrium reactions, with their implementation performed in *Models/ElectroChemistry/Reactions.cpp*. For the dynamic reactions, the reaction rates are defined as:

$$R = k_f \prod_{\pi_i} \frac{C_{\pi_i}}{C_{\text{ref}}} - k_b \prod_{\pi_j} \frac{C_{\pi_j}}{C_{\text{ref}}} \tag{4}$$

where k_f and k_b are the forward and backward reaction rates, π_i and π_j are the species involved in the forward and backward reactions respectively, and C_{ref} is a reference concentration. Based on this rate, the reaction term is then obtained as:

$$R_\pi = \frac{R}{n_\pi} \tag{5}$$

where n_π is the stoichiometric coefficient of species π in the reaction.

For equilibrium reactions, the reaction rate is defined as:

$$R = k_{\text{dummy}} \left(K \prod_{\pi_i} \frac{C_{\pi_i}}{C_{\text{ref}}} - \prod_{\pi_j} \frac{C_{\pi_j}}{C_{\text{ref}}} \right) \tag{6}$$

where K is the equilibrium constant, and k_{dummy} is a dummy rate used to enforce equilibrium. The reaction term is then obtained in the same way as for dynamic reactions, using Eq. (5).

3.2. Surface reactions & charge conservation

The surface reactions and charge conservation condition are implemented in *Models/Electrochemistry/-SurfaceReactions.cpp*. These surface reactions are defined through the Butler-Volmer equation, giving the reaction current as:

$$i = i_0 \exp\left(\frac{\alpha z F \eta}{RT}\right) K \prod_{\pi_i} \frac{C_{\pi_i}}{C_{\text{ref}}} - i_0' \exp\left(-\frac{(1-\alpha) z F \eta}{RT}\right) \prod_{\pi_j} \frac{C_{\pi_j}}{C_{\text{ref}}} \tag{7}$$

with i_0 and i_0' the reference reaction currents, α the charge transfer coefficient, z the number of electrons involved in the reaction, and η the overpotential. From this reaction current, the species flux at the metal surface is given by:

$$\nu_\pi = \frac{i\, n_\pi}{z_\pi F} \tag{8}$$

Finally, this model also implements the charge conservation condition, which is used to ensure that the sum of all anodic and cathodic reactions on the surface is zero. This is done by solving the following equation:

$$\int_\Gamma \sum_\pi i_\pi \, \mathrm{d}\Gamma = 0 \tag{9}$$

with this constraint being imposed to allow the metal potential to change over time.

3.3. Boundary conditions

The final model included is responsible for adding the boundary conditions, implemented in *Models/-Electrochemistry/EChemConstraints.cpp*. This model adds boundary conditions based on the C0 values set in the input file to the surface the boundary condition is applied to. The set boundary conditions are eliminated from the tangent matrix and force vectors through allocation matrices $\boldsymbol{C}_{\text{con}}$ and $\boldsymbol{C}_{\text{uncon}}$, reordering the system into a constrained and unconstrained part. This allows the constrained system to be solved as:

$$\boldsymbol{C}_{\text{uncon}}^T \boldsymbol{K} \boldsymbol{C}_{\text{uncon}} \mathbf{y} = -\left(\boldsymbol{C}_{\text{uncon}}^T \boldsymbol{f} + \boldsymbol{C}_{\text{uncon}}^T \boldsymbol{K} \boldsymbol{C}_{\text{con}} \mathbf{c} \right) \tag{10}$$

with the values of the boundary constraints contained in the vector $\mathbf{c}$. After solving, the state vector is then incremented through:

$$\mathbf{x}^{\text{new}} = \mathbf{x}^{\text{old}} + \boldsymbol{C}_{\text{uncon}} \mathbf{y} + \boldsymbol{C}_{\text{con}} \mathbf{c} \tag{11}$$

4. Output files and post-processing

4.1. Output files

The simulations will output three kinds of files to the output folder, a mesh file, a series of results files, and a temporal data file, all of which use the HDF5 file format. The mesh file contains the mesh used for saving the outputs, structured as (where ElementGroup is either Pit/Bar/Interior):

- <ElementGroup>/X: 3xn matrix containing the X coordinates of the output data points for each of the elements (6xn for volume elements)
- <ElementGroup>/Y: Contains the Y coordinates of the output data points
- <ElementGroup>/Z: Contains the Z coordinates of the output data points
- <ElementGroup>/Xip: Contains the X coordinates of the integration points
- <ElementGroup>/Yip: Contains the Y coordinates of the integration points
- <ElementGroup>/Zip: Contains the Z coordinates of the integration points

The result files contain the accompanying simulation results at these coordinates, with the output file structured as:

- time: The time at which the results are saved
- <Pit/Bar/Interior>/<Species>: The concentration of the species at the output points
- <Pit/Bar/Interior>/ePot: The electrolyte potential at the output points
- <Pit/Bar>/HER: The hydrogen evolution reaction rate at the output points
- <Pit/Bar>/OER: The oxygen evolution reaction rate at the output points
- <Pit/Bar>/Corrosion: The corrosion rate at the output points

Finally, the temporal data file (TimeData.hdf5) contains the following information: TimeDataTypes: A vector containing the names of all time data saved, and TimeData: A matrix of size $n \times t$ containing temporal information saved at each time increment t.T he columns of this matrix represent the following time data:

- time
- time increment
- index of the time step
- total hydrogen reaction current
- area over which this hydrogen reaction occurs
- total oxygen reaction current
- area over which this current occurs
- Total corrosion current
- area over which this corrosion current occurs
- the electric potential of the metal.

4.2. Post-processing

An example of how to further process the output files is given in the MATLAB file *TestCases/ElectroChemistry/PlotData_EChem.m*. This file first asks for the names of the output files:

TestCases/ElectroChemistry/PlotData_EChem.m

```
meshFile = "../../Results/mesh_0.hdf5";
dataFile = "../../Results/results_100.hdf5";
timeDataFile = "../../Results/TimeData.hdf5";
```

After which it opens the output files, and reads the contents saved in them:

TestCases/ElectroChemistry/PlotData_EChem.m

```
        t = h5read(dataFile,"/time")

        for i=1:length(DataNamesNodes)
            X{i} = h5read(meshFile,GroupName{i}+"X");
            Y{i} = h5read(meshFile,GroupName{i}+"Y");
            Z{i} = h5read(meshFile,GroupName{i}+"Z");

            Data = h5read(dataFile,GroupName{i}+DataNamesNodes{i});
            D{i} = Data;
        end
        for i=1:length(DataNamesIPs)
            Xip = h5read(meshFile,GroupName{i}+"Xip");
            Yip = h5read(meshFile,GroupName{i}+"Yip");
            Zip = h5read(meshFile,GroupName{i}+"Zip");

            DIP{i} = h5read(dataFile,GroupName{i}+DataNamesIPs{i});
        end
```

The nodal data is then plotted via:

TestCases/ElectroChemistry/PlotData_EChem.m

```
for i=1:8
    if (individualfigures)
        figure
    else
        nexttile
    end
    PlotNodeData(X{i}, Y{i}, Z{i}, D{i});
    title(DataNamesNodes{i});
    view(0,0)
end
```

Plotting the concentrations of each species, and the electrolyte potential throughout the domain. This results in the Figure shown in Fig. 4.

The second part of this post-processing script plots the data at the surface of the metal rebar:

TestCases/ElectroChemistry/PlotData_EChem.m

```
for i=9:12
    if (individualfigures)
        figure
    else
        nexttile
    end
    PlotNodeData(X{i}, Y{i}, Z{i}, D{i});
    hold on
    PlotNodeData(X{i+4}, Y{i+4}, Z{i+4}, D{i+4});
    title(DataNamesNodes{i});
    view(0,90)
    axis equal
end
```

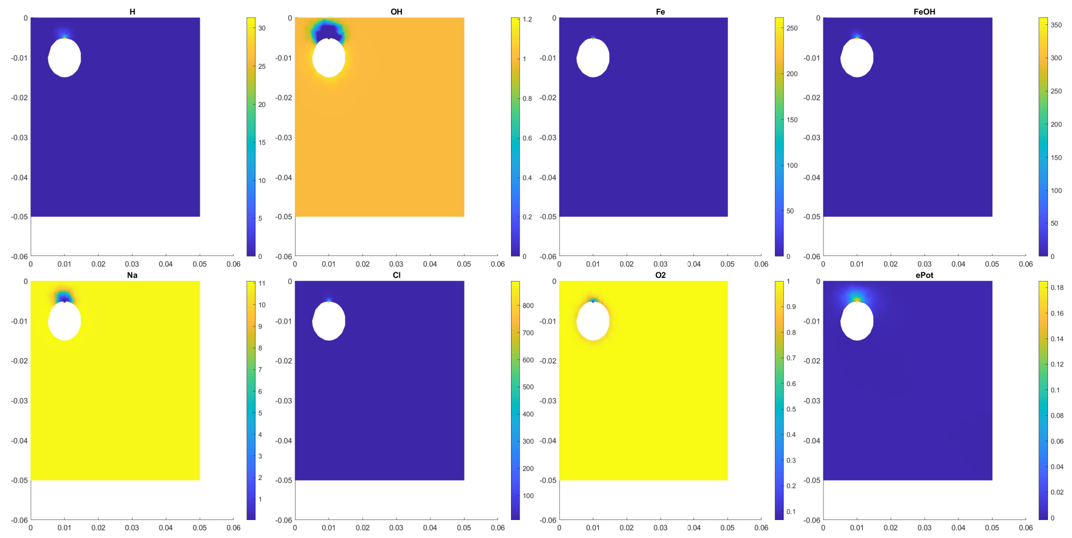

Figure 4: Example of the post-processing results, showing the concentrations of all species, and the electrolyte potential.

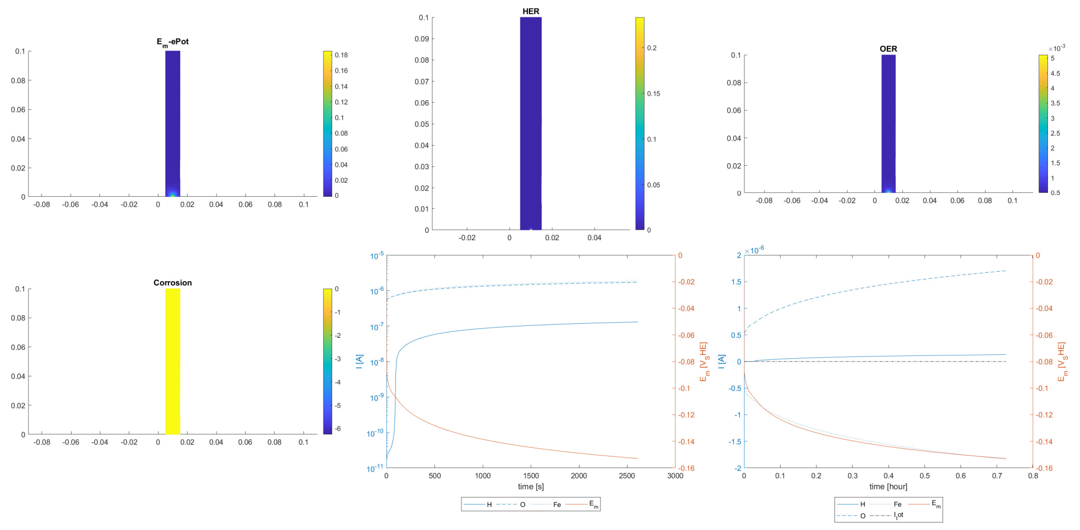

Figure 5: Example of the post-processing results, showing the surface reaction currents, and the total currents occurring throughout the domain.

Producing the results shown in Fig. 5. This shows the surface reaction currents, and the total currents occurring throughout the domain.

5. References

[1] T. Hageman, C. Andrade, E. Martínez-Pañeda, Corrosion of metal reinforcements within concrete and localisation of supporting reactions under natural conditions, Electrochimica Acta 461 (2023) 142624. doi:10.1016/j.electacta.2025.146203.

[2] C. L. Page, K. W. J. Treadaway, Aspects of the electrochemistry of steel in concrete, Nature 297 (5862) (1982) 109–115. doi:10.1038/297109a0.

[3] C. Arya, F. K. Ofori-Darko, Influence of crack frequency on reinforcement corrosion in concrete, Cement and Concrete Research 26 (3) (1996) 345–353. doi:10.1016/S0008-8846(96)85022-8.

[4] I. Khan, R. François, A. Castel, Prediction of reinforcement corrosion using corrosion induced cracks width in corroded reinforced concrete beams, Cement and Concrete Research 56 (2014) 84–96. doi:10.1016/j.cemconres.2013.11.006.

[5] E. Korec, M. Jirásek, H. S. Wong, E. Martínez-Pañeda, A phase-field chemo-mechanical model for corrosion-induced cracking in reinforced concrete, Construction and Building Materials 393 (2023) 131964. doi:10.1016/j.conbuildmat.2023.131964.

[6] E. Korec, M. Jirásek, H. S. Wong, E. Martínez-Pañeda, Unravelling the interplay between steel rebar corrosion rate and corrosion-induced cracking of reinforced concrete, Cement and Concrete Research 186 (2024) 107647. doi:10.1016/j.cemconres.2024.107647.

[7] C. Fang, K. Lundgren, L. Chen, C. Zhu, Corrosion influence on bond in reinforced concrete, Cement and Concrete Research 34 (11) (2004) 2159–2167. doi:10.1016/j.cemconres.2004.04.006.

[8] H.-S. Lee, T. Noguchi, F. Tomosawa, Evaluation of the bond properties between concrete and reinforcement as a function of the degree of reinforcement corrosion, Cement and Concrete Research 32 (8) (2002) 1313–1318. doi:10.1016/S0008-8846(02)00783-4.

[9] Y. Zhao, J. Yu, W. Jin, Damage analysis and cracking model of reinforced concrete structures with rebar corrosion, Corrosion Science 53 (10) (2011) 3388–3397. doi:10.1016/j.corsci.2011.06.018.

[10] A. Michel, A. O. S. Solgaard, B. J. Pease, M. R. Geiker, H. Stang, J. F. Olesen, Experimental investigation of the relation between damage at the concrete-steel interface and initiation of reinforcement corrosion in plain and fibre reinforced concrete, Corrosion Science 77 (2013) 308–321. doi:10.1016/j.corsci.2013.08.019.

[11] U. Angst, B. Elsener, C. K. Larsen, Ø. Vennesland, Chloride induced reinforcement corrosion: Rate limiting step of early pitting corrosion, Electrochimica Acta 56 (17) (2011) 5877–5889. doi:10.1016/j.electacta.2011.04.124.

[12] K. Wang, M. Salasi, S. Bakhtiari, M. Iannuzzi, On the Critical Factors for Estimating the Pit Stability Product under a Salt Film, Journal of The Electrochemical Society 168 (6) (2021) 061506. doi:10.1149/1945-7111/ac0aab.

[13] U. M. Angst, Predicting the time to corrosion initiation in reinforced concrete structures exposed to chlorides, Cement and Concrete Research 115 (2019) 559–567. doi:10.1016/j.cemconres.2018.08.007.

[14] D. D. Macdonald, Y. Zhu, J. Yang, J. Qiu, G. R. Engelhardt, A. Sagüés, L. Sun, Z. Xiong, Corrosion of rebar in concrete. Part IV. On the theoretical basis of the chloride threshold, Corrosion Science 185 (2021) 109460. doi:10.1016/j.corsci.2021.109460.

[15] H. Wang, E. H. Han, Simulation of metastable corrosion pit development under mechanical stress, Electrochimica Acta 90 (2013) 128–134. doi:10.1016/J.ELECTACTA.2012.11.056.

[16] M. Pourbaix, J. Van Muylder, E. D. Verink, A. Pourbaix, études électrochimiques concernant les corrosions en cellules occluses C.C.O. et protection contre ces corrosions: Corrosion par piqûres, corrosion caverneuse, corrosion fissurante sous tension, corrosion sélective d'alliages: Applications au fer et aux aciers, au cuivre, aux cupronickels et aux laitons, Journal of Electroanalytical Chemistry and Interfacial Electrochemistry 62 (1) (1975) 219–229. doi:10.1016/0022-0728(75)80039-8.

[17] C. D. Taylor, P. Lu, J. Saal, G. S. Frankel, J. R. Scully, Integrated computational materials engineering of corrosion resistant alloys, npj Materials Degradation 2 (1) (2018) 1–10. doi:10.1038/s41529-018-0027-4.

[18] T. Li, J. Wu, X. Guo, A. M. Panindre, G. S. Frankel, Activation energy of metal dissolution in local pit environments, Corrosion Science 193 (2021) 109901. doi:10.1016/j.corsci.2021.109901.

[19] A. Turnbull, The solution composition and electrode potential in pits, crevices and cracks, Corrosion Science 23 (8) (1983) 833–870. doi:10.1016/0010-938X(83)90014-8.

[20] H. W. Pickering, R. P. Frankenthal, On the Mechanism of Localized Corrosion of Iron and Stainless Steel: I . Electrochemical Studies, Journal of The Electrochemical Society 119 (10) (1972) 1297. doi:10.1149/1.2403982.

[21] T. P. Hoar, The production and breakdown of the passivity of metals, Corrosion Science 7 (6) (1967) 341–355. doi:10.1016/S0010-938X(67)80023-4.

[22] J. R. Galvele, Transport processes in passivity breakdown—II. Full hydrolysis of the metal ions, Corrosion Science 21 (8) (1981) 551–579. doi:10.1016/0010-938X(81)90009-3.

[23] G. S. Frankel, T. Li, J. R. Scully, Perspective—Localized Corrosion: Passive Film Breakdown vs Pit Growth Stability, Journal of The Electrochemical Society 164 (4) (2017) C180. doi:10.1149/2.1381704jes.

[24] G. S. Frankel, The growth of 2-D pits in thin film aluminum, Corrosion Science 30 (12) (1990) 1203–1218. doi:10.1016/0010-938X(90)90199-F.

[25] V. A. Nguyen, R. C. Newman, A comprehensive modelling and experimental approach to study the diffusion-controlled dissolution in pitting corrosion, Corrosion Science 186 (December 2020) (2021) 109461. doi:10.1016/j.corsci.2021.109461.

[26] E. Korec, M. Jirásek, H. S. Wong, E. Martínez-Pañeda, Phase-field chemo-mechanical modelling of corrosion-induced cracking in reinforced concrete subjected to non-uniform chloride-induced corrosion, Theoretical and Applied Fracture Mechanics 129 (2024) 104233.

[27] Microsoft, Windows Subsystem for Linux Documentation, https://learn.microsoft.com/en-us/windows/wsl/.

[28] Accelerate Fast Math with Intel® oneAPI Math Kernel Library, https://www.intel.com/content/www/us/en/developer/tools/oneapi/oneml
[29] Libeigen / eigen · GitLab, https://gitlab.com/libeigen/eigen (Apr. 2025).
[30] Tencent/rapidjson, Tencent (Apr. 2025).
[31] W. J. Schroeder, K. Martin, W. E. Lorensen, L. S. Avila, K. W. Martin, B. Lorensen, The Visualization Toolkit: An Object-Oriented Approach to 3D Graphics ; [Visualize Data in 3D - Medical, Engineering or Scientific ; Build Your Own Applications with C++, Tcl, Java or Python ; Includes Source Code for VTK (Supports UNIX, Windows and Mac)], 4th Edition, Kitware, Inc, Clifton Park, NY, 2006.
[32] A. Devresse, N. Cornu, L. Grosheintz-Laval, O. Awile, T. de Geus, F. Pereira, M. Wolf, HighFive Contributors, HighFive - Header-only C++ HDF5 interface, Zenodo (Dec. 2024). doi:10.5281/ZENODO.10679422.
[33] S. Balay, S. Abhyankar, M. Adams, S. Benson, J. Brown, P. Brune, K. Buschelman, E. Constantinescu, L. Dalcin, A. Dener, V. Eijkhout, J. Faibussowitsch, W. Gropp, V. Hapla, T. Isaac, P. Jolivet, D. Karpeev, D. Kaushik, M. Knepley, F. Kong, S. Kruger, D. May, L. McInnes, R. Mills, L. Mitchell, T. Munson, J. Roman, K. Rupp, P. Sanan, J. Sarich, B. Smith, S. Zampini, H. Zhang, J. Zhang, PETSc/TAO Users Manual (Rev. 3.20), Tech. Rep. ANL–21/39 Rev. 3-20, 2205494, 185712 (Nov. 2023). doi:10.2172/2205494.
[34] T. Hageman, E. Martínez-Pañeda, Stabilising Effects of Lumped Integration Schemes for the Simulation of Metal-Electrolyte Reactions, Journal of The Electrochemical Society 170 (2) (2023) 021511. doi:10.1149/1945-7111/ACB971.
[35] D. M. Tartakovsky, M. Dentz, Diffusion in Porous Media: Phenomena and Mechanisms, Transport in Porous Media 130 (1) (2019) 105–127. doi:10.1007/S11242-019-01262-6/FIGURES/6.
[36] B. Ghanbarian, A. G. Hunt, R. P. Ewing, M. Sahimi, Tortuosity in Porous Media: A Critical Review, Soil Science Society of America Journal 77 (5) (2013) 1461–1477. doi:10.2136/SSSAJ2012.0435.
[37] S. W. Feldberg, On the dilemma of the use of the electroneutrality constraint in electrochemical calculations, Electrochemistry Communications 2 (7) (2000) 453–456. doi:10.1016/S1388-2481(00)00055-2.