1. **Problem description**

For 1D solid-liquid problem, specie M transport between two domains (left domain 1 and right domain 2).

1. Diffusion in both domains can be described by Fick’s law, Specie M has different diffusivities in different materials.
2. Dirichlet boundary conditions are applied at most left and most right.
3. At the interface consider the following
   1. Fluxes are matched from both domains
   2. The reaction is

With the first order reaction rate assuming a quasi-steady-state

1. **Model in 1-D** 
   1. **Governing equation**

2

1

0

In domain 1 ()

|  |  |
| --- | --- |
|  | (1) |

In domain 2 ()

|  |  |
| --- | --- |
|  | (2) |

Where D1 =4, D2=2.

* 1. **Boundary condition**

|  |  |
| --- | --- |
|  | (3) |
|  | (4) |

* 1. **At the interface**

|  |  |
| --- | --- |
| or | (5) |

Also due to the reactions, it should satisfy

|  |  |
| --- | --- |
| Or | (6) |

Where .

* 1. **Analytic solution:**

|  |  |
| --- | --- |
|  | (7) |



1. **MOOSE way** 
   1. **Overview**

MOOSE works on “kernel” which represent a piece of physics (one or more operators or terms in a PDE, like )

|  |  |  |
| --- | --- | --- |
| Equation | Kernel | Remark |
| (1) | MatDiffusion | MOOSE provide |
| (2) | MatDiffusion | MOOSE provide |
| (3) | DirichletBC | MOOSE provide |
| (4) | DirichletBC | MOOSE provide |
| (5) | InterfaceDiffusion | MOOSE provide |
| (6) | InterfaceReaction | User defined |

As we can notice, for this simple question MOOSE can provide most of the kernels except for InterfaceReaction Kernel ( Eq.(6)).

* 1. **Details for InterfaceReaction Kernel**

Similar to InterfaceDiffusion kernel [[[1]](#footnote-1)], in order to write a kernel to operate the variable (u1 for master, u2 for neighbor) for InterfaceReaction, we need to get the Residual [[[2]](#footnote-2)] (computeQpResidual) and Jacobian [2][[[3]](#footnote-3)] (computeQpJacobian) for master domain and neighbor domain respectively.

Residual (computeQpResidual)

|  |  |
| --- | --- |
| Residual for element (master domain) |  |
| Residual for neighbor |  |

Where is the test function of master domain, is the test function of neighbor domain. The negative sign in element (master domain), think of a NeumannBC.

Jacobian (computeQpJacobian)

|  |  |  |  |
| --- | --- | --- | --- |
| Kernel code | | Understanding and explanation | |
| computeQpJacobian()  case Moose:: | Jac= | Domain | Get derivative w.r.t. |
| ElementElement |  | master | u1 |
| NeighborNeighbor |  | neighbor | u2 |
| NeighborElement |  | neighbor | u1 |
| ElementNeighbor |  | master | u2 |

Where is the shape function of master domain, is the shape function of neighbor domain.

* 1. **Result and discussion**

The code [[[4]](#footnote-4)] has been implemented to 1-D geometry giving the results agree with the analytic solution (ElementL2Error is at the order of E-17).

Jacobian is analyzed by the method provided by MOOSE [[[5]](#footnote-5)], the result shows the No errors detected. :-).

1. http://mooseframework.org/wiki/MooseSystems/InterfaceKernels/ [↑](#footnote-ref-1)
2. http://mooseframework.org/wiki/MooseTraining/FEM/NumericalImplementation/ [↑](#footnote-ref-2)
3. http://mooseframework.org/moose/application\_development/jacobian\_definition.html [↑](#footnote-ref-3)
4. https://github.com/wesleyzzz/InterfaceReaction1.git [↑](#footnote-ref-4)
5. http://mooseframework.org/wiki/JacobianDebugging/ [↑](#footnote-ref-5)