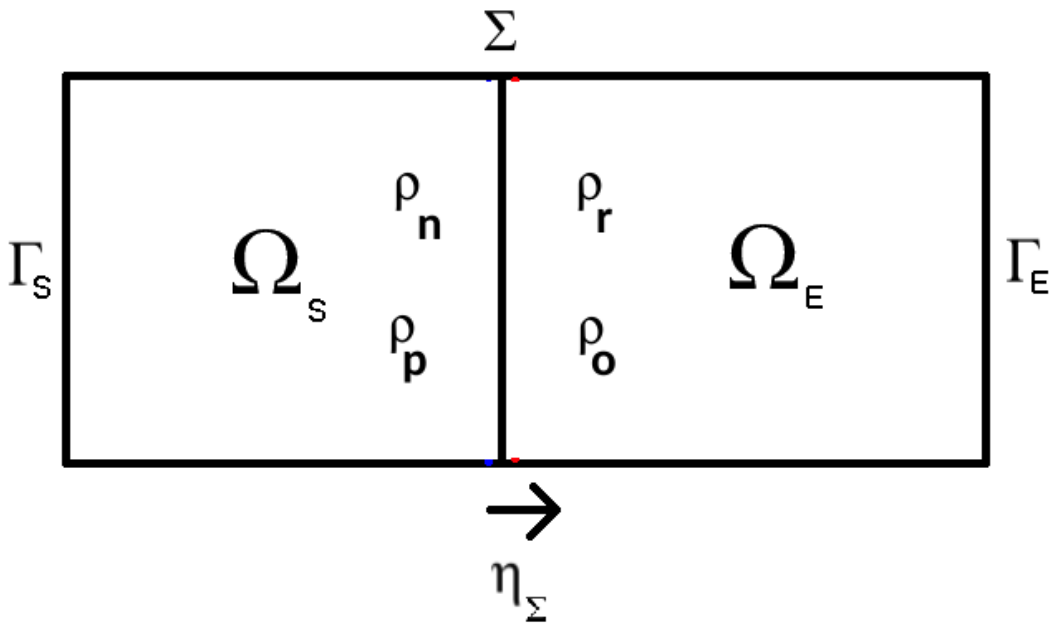
Simulation of solid-liquid solar cells



Solar Cell Domain Decomposition

Numerical Challenge: Simulating systems that are non-linearly coupled through an interface and have different timescales.

Flow-transport with a non-linear reactive interface

Transport in subdomains:

$$\Omega_S \quad \begin{cases} \partial_t \rho_n + J_n \nabla \cdot (\nabla \Phi \rho_n - \nabla \rho_n) = -R(\rho_n, \rho_p) + G(\mathbf{x}) \\ \partial_t \rho_p + J_p \nabla \cdot (- \nabla \Phi \rho_p - \nabla \rho_p) = -R(\rho_n, \rho_p) + G(\mathbf{x}) \end{cases}$$

$$\Omega_E \quad \begin{cases} \partial_t \rho_r + J_r \nabla \cdot (\nabla \Phi \rho_r - \nabla \rho_r) = 0 \\ \partial_t \rho_o + J_o \nabla \cdot (- \nabla \Phi \rho_o - \nabla \rho_p) = 0 \end{cases}$$

$$\Sigma \quad \begin{cases} J_n (\nabla \Phi \rho_n - \nabla \rho_n) \cdot \boldsymbol{\eta} = k_{et} (\rho_n - \rho_n^e) \rho_o \\ J_p (\nabla \Phi \rho_p - \nabla \rho_p) \cdot \boldsymbol{\eta} = k_{ht} (\rho_p - \rho_p^e) \rho_r \\ J_r (\nabla \Phi \rho_r - \nabla \rho_r) \cdot \boldsymbol{\eta} = k_{ht} (\rho_p - \rho_p^e) \rho_r - k_{et} (\rho_n - \rho_n^e) \rho_o \\ J_o (\nabla \Phi \rho_o - \nabla \rho_o) \cdot \boldsymbol{\eta} = -k_{ht} (\rho_p - \rho_p^e) \rho_r + k_{et} (\rho_n - \rho_n^e) \rho_o \end{cases}$$

Flow in entire domain:

$$-\nabla \cdot (\lambda(\mathbf{x}) \nabla \Phi) = \begin{cases} C(\mathbf{x}) - (\rho_n - \rho_p) & \text{in} \quad \Omega_S \\ -(\rho_r - \rho_o) & \text{in} \quad \Omega_E \end{cases}$$

Numerical methods and problems

- **Have**: Flow-transport code that uses a mixed finite element method (MFEM) for flow equation and local discontinuous Galerkin (LDG) method for transport equations.
- **Have:** Code that solves diffusion equations which are non-linearly coupled through interface using LDG in space and time lagging (forward Euler) for linearization.
- **Problem:** Slow convergence to steady state solutions because linearization imposes severe constraint on time step.
- **Need:** Use big time steps.
- **Need:** Non-linear solver which is fast and easy to implement in deal.ii. (Newton-type method?)
- **Need:** Advice on best way to set up data structures.