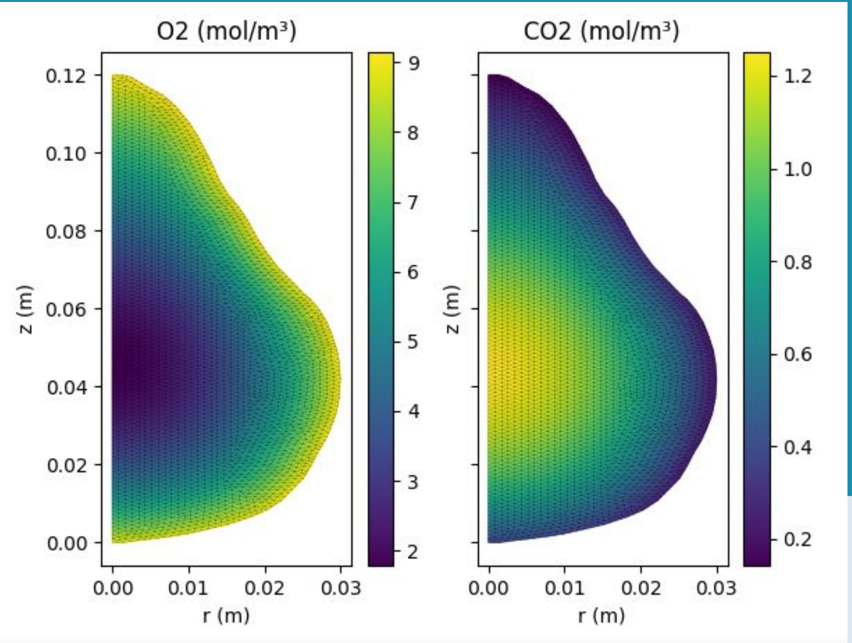
KU LEUVEN

Pear project:
Solving respiration–diffusion system using FEM simulation

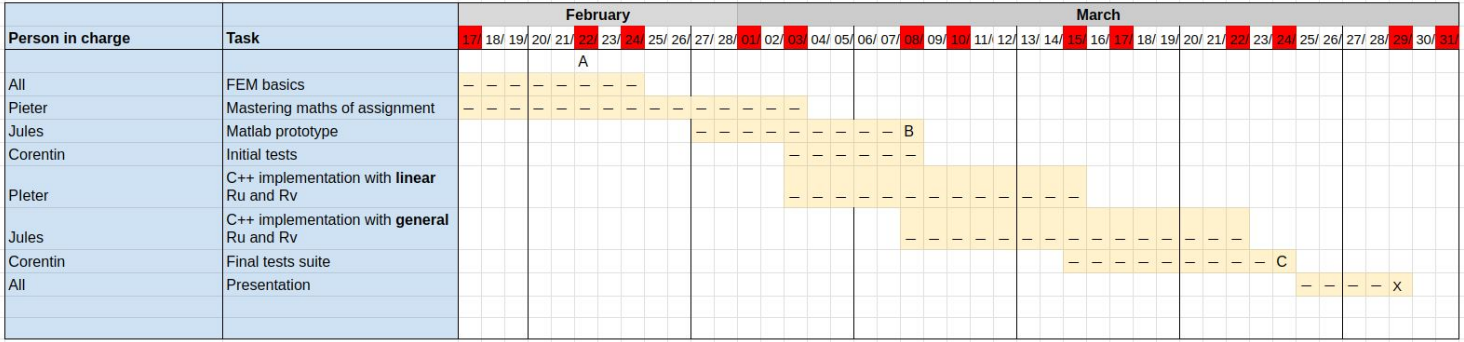


Corentin Bonte, Jules Verbeke
& Pieter Vanslambrouck

Planning

Milestones: Matlab prototype, C++ implementation, test suite

Generally, we followed our planning closely



	Milestones	
A	discuss your planning in session 3 (22/2)	✔
B	Working matlab prototype	✔
C	Compiled C++ solution	✔
D	C++ Test Suite + Pipeline	✔
X	Presentation	

Tools & libraries

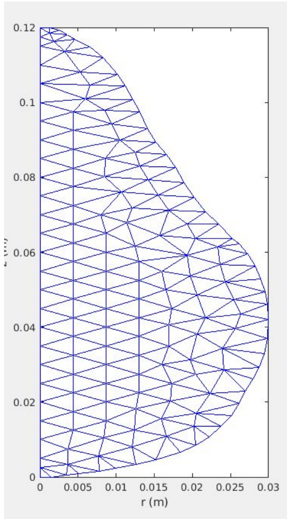
Tools:

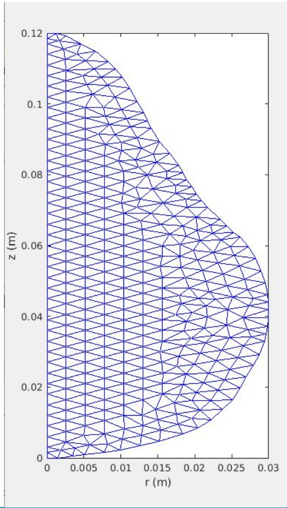
- Git, GitLab (merge requests, issues, test pipeline etc.)
- Matlab, C++, Python
- VS code, GDB, pdb
- Profiling: valgrind & kcachegrind

Libraries:

- Matlab: MESH2D library
- C++: Eigen (Sparse matrix algebra)

MESH2D:
uniform meshes of different granularity



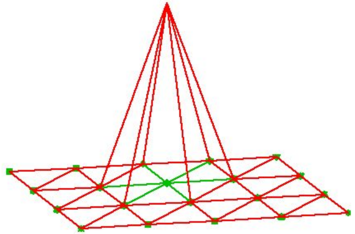


Theory

Respiration-diffusion system:

$$\begin{cases} \nabla \cdot \left(r \begin{pmatrix} \sigma_{u,r} & 0 \\ 0 & \sigma_{u,z} \end{pmatrix} \nabla C_u(r,z) \right) & = r\, R_u(C_u(r,z), C_v(r,z)) \\ \nabla \cdot \left(r \begin{pmatrix} \sigma_{v,r} & 0 \\ 0 & \sigma_{v,z} \end{pmatrix} \nabla C_v(r,z) \right) & = -r\, R_v(C_u(r,z), C_v(r,z)) \end{cases}$$

Chapeau basis function



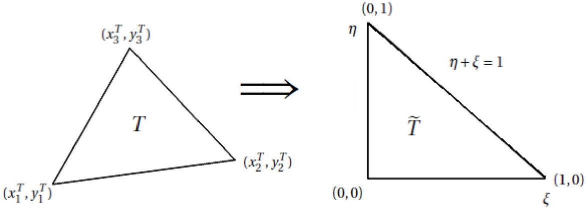
Weak formulation:

$$\int_\Omega \vec{q}_u(r,z) \cdot \nabla \varphi(r,z)\, \mathrm{d}\Omega + \int_\Omega r\, R_u(C_u, C_v)\, \varphi(r,z)\, \mathrm{d}\Omega + \int_\Gamma r\, \varrho_u\, C_u^*(r,z)\, \varphi(r,z)\, \mathrm{d}\Gamma = 0$$

$$\int_\Omega \vec{q}_v(r,z) \cdot \nabla \varphi(r,z)\, \mathrm{d}\Omega - \int_\Omega r\, R_v(C_u, C_v)\, \varphi(r,z)\, \mathrm{d}\Omega + \int_\Gamma r\, \varrho_v\, C_v^*(r,z)\, \varphi(r,z)\, \mathrm{d}\Gamma = 0.$$

$$\begin{pmatrix} K_u & 0 \\ 0 & K_v \end{pmatrix} \begin{pmatrix} \mathbf{c}_u \\ \mathbf{c}_v \end{pmatrix} - \begin{pmatrix} \mathbf{f}_u \\ \mathbf{f}_v \end{pmatrix} + \begin{pmatrix} \mathbf{H}_u(\mathbf{c}_u, \mathbf{c}_v) \\ \mathbf{H}_v(\mathbf{c}_u, \mathbf{c}_v) \end{pmatrix} = \begin{pmatrix} \mathbf{0} \\ \mathbf{0} \end{pmatrix}$$

Coordinate transformation to compute integrals:



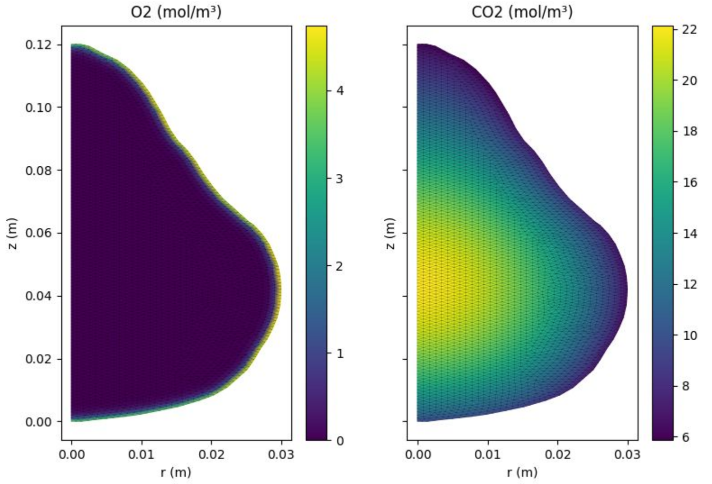
Theory: initial guess

Linear approximations of respiration functions:

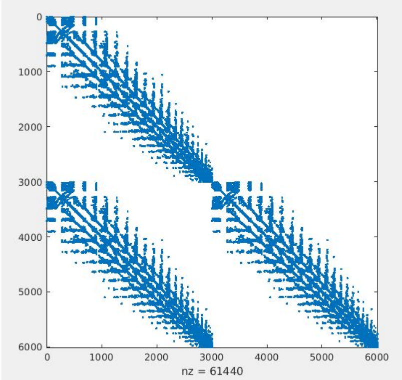
$$R_u(C_u, C_v) = \frac{V_{mu} C_u}{(K_{mu} + C_u)(1 + \frac{C_v}{K_{mv}})} \approx \frac{V_{mu}}{K_{mu}} C_u$$

$$R_v(C_u, C_v) = r_q R_u(C_u, C_v) + \frac{V_{mfv}}{1 + \frac{C_u}{K_{mfu}}} \approx r_q \frac{V_{mu}}{K_{mu}} C_u + V_{mfv}$$

$$\begin{pmatrix} K_u & 0 \\ 0 & K_v \end{pmatrix} \begin{pmatrix} \mathbf{c}_u \\ \mathbf{c}_v \end{pmatrix} - \begin{pmatrix} \mathbf{f}_u \\ \mathbf{f}_v \end{pmatrix} + \begin{pmatrix} \mathbf{H}_u(\mathbf{c}_u, \mathbf{c}_v) \\ \mathbf{H}_v(\mathbf{c}_u, \mathbf{c}_v) \end{pmatrix} = \begin{pmatrix} \mathbf{0} \\ \mathbf{0} \end{pmatrix}$$



K =

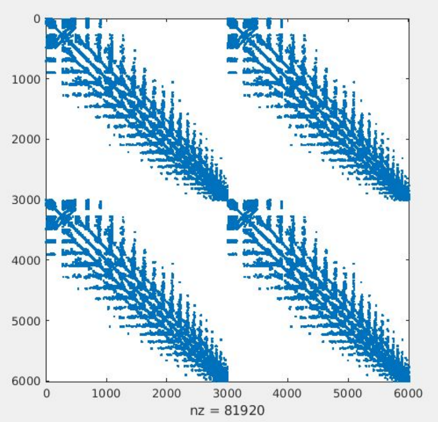


KU LEUVEN

Theory: Newton iterations

Recall, weak formulation leads to non-linear matrix equation:

$$\begin{pmatrix} K_u & 0 \\ 0 & K_v \end{pmatrix} \begin{pmatrix} \mathbf{c}_u \\ \mathbf{c}_v \end{pmatrix} - \begin{pmatrix} \mathbf{f}_u \\ \mathbf{f}_v \end{pmatrix} + \begin{pmatrix} \mathbf{H}_u(\mathbf{c}_u, \mathbf{c}_v) \\ \mathbf{H}_v(\mathbf{c}_u, \mathbf{c}_v) \end{pmatrix} = \begin{pmatrix} \mathbf{0} \\ \mathbf{0} \end{pmatrix}$$

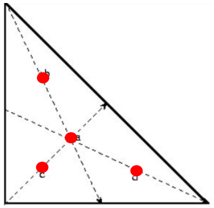
K + J = 

Newton Iteration: Linearization

$$\begin{pmatrix} H_u(c_u, c_v) \\ H_v(c_u, c_v) \end{pmatrix} \approx \begin{pmatrix} H_u(c_u^k, c_v^k) \\ H_v(c_u^k, c_v^k) \end{pmatrix} + J \begin{pmatrix} \Delta c_u \\ \Delta c_v \end{pmatrix}$$

With $J = \begin{pmatrix} J_1 & J_2 \\ J_3 & J_4 \end{pmatrix} = \begin{pmatrix} \frac{\delta H_u}{\delta c_u} & \frac{\delta H_u}{\delta c_v} \\ \frac{\delta H_v}{\delta c_u} & \frac{\delta H_v}{\delta c_v} \end{pmatrix}$

$$\iint_{T_{\mathrm{st}}} g(\xi, \eta)\, \mathrm{d}\xi \mathrm{d}\eta = -\frac{27}{96} \cdot g\left(\frac{1}{3}, \frac{1}{3}\right) + \frac{25}{96}\left[g\left(\frac{1}{5}, \frac{1}{5}\right) + g\left(\frac{1}{5}, \frac{3}{5}\right) + g\left(\frac{3}{5}, \frac{1}{5}\right)\right]$$



Jacobian was determined **analytically** and subsequently approximated using **quadrature** integration rules on standard triangle using **coordinate transformations**.

Theory: Newton iterations

Recall, weak formulation leads to non-linear matrix equation:

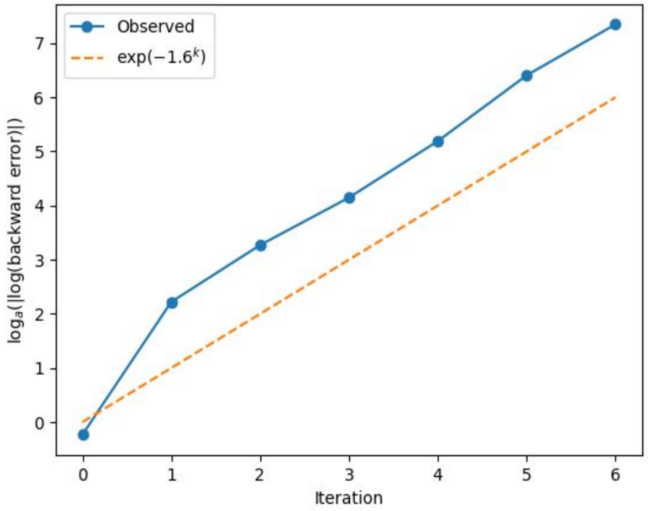
$$\begin{pmatrix} K_u & 0 \\ 0 & K_v \end{pmatrix} \begin{pmatrix} \mathbf{c}_u \\ \mathbf{c}_v \end{pmatrix} - \begin{pmatrix} \mathbf{f}_u \\ \mathbf{f}_v \end{pmatrix} + \begin{pmatrix} \mathbf{H}_u(\mathbf{c}_u, \mathbf{c}_v) \\ \mathbf{H}_v(\mathbf{c}_u, \mathbf{c}_v) \end{pmatrix} = \begin{pmatrix} \mathbf{0} \\ \mathbf{0} \end{pmatrix}$$

Newton Iteration: Linearization

$$\begin{pmatrix} H_u(c_u, c_v) \\ H_v(c_u, c_v) \end{pmatrix} \approx \begin{pmatrix} H_u(c_u^k, c_v^k) \\ H_v(c_u^k, c_v^k) \end{pmatrix} + J \begin{pmatrix} \Delta c_u \\ \Delta c_v \end{pmatrix}$$

With $J = \begin{pmatrix} J_1 & J_2 \\ J_3 & J_4 \end{pmatrix} = \begin{pmatrix} \frac{\delta H_u}{\delta c_u} & \frac{\delta H_u}{\delta c_v} \\ \frac{\delta H_v}{\delta c_u} & \frac{\delta H_v}{\delta c_v} \end{pmatrix}$

Convergence: almost quadratic



Jacobian was determined **analytically** and subsequently approximated using **quadrature** integration rules on standard triangle using **coordinate transformations**.

Testing

Matlab: verification of order of quadrature rules

Automatic Gitlab C++ test pipeline

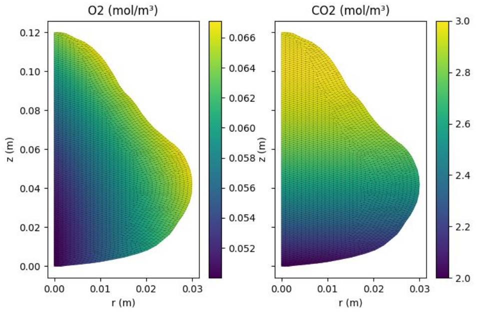
Observed ill-conditioned matrix equations for:

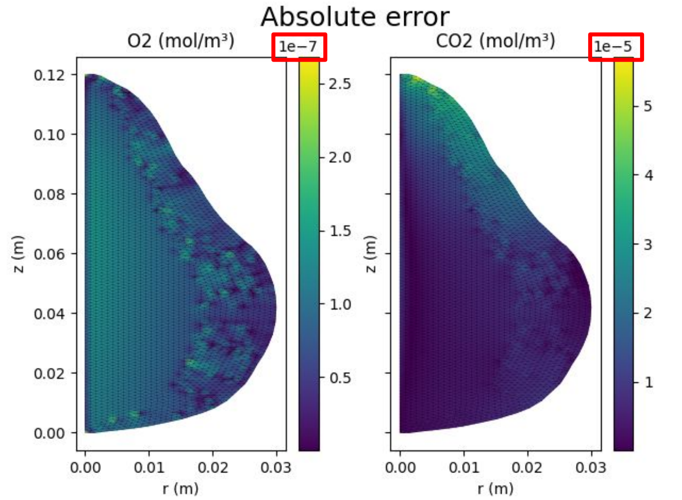
- Newton iterations for rough meshes
- Manufactured solutions initial guess

Status	Pipeline	Triggerer	Stages
passed 00:01:25 11 minutes ago	Small change in theory file #204174 main 15143098 latest		
passed 00:01:28 54 minutes ago	Bugfix in constants #204109 main fb6d649f		
failed 00:00:38 1 hour ago	Remove meshes that don't converge fo... #204095 main f5873506		

Method of manufactured solutions

Cu = 0.05+0.05*sin(10r) + z²
Cv = 2+3*r² + sin(15z)





Choose analytical solution for Cu and Cv

$$\begin{cases} \nabla \cdot \left(r \begin{pmatrix} \sigma_{u,r} & 0 \\ 0 & \sigma_{u,z} \end{pmatrix} \nabla C_u(r,z) \right) = r\, R_u(C_u(r,z), C_v(r,z)) \;\; \color{red}{+\ Qu(r,z)} \\ \nabla \cdot \left(r \begin{pmatrix} \sigma_{v,r} & 0 \\ 0 & \sigma_{v,z} \end{pmatrix} \nabla C_v(r,z) \right) = -r\, R_v(C_u(r,z), C_v(r,z)) \;\; \color{red}{+\ Qv(r,z)} \end{cases}$$

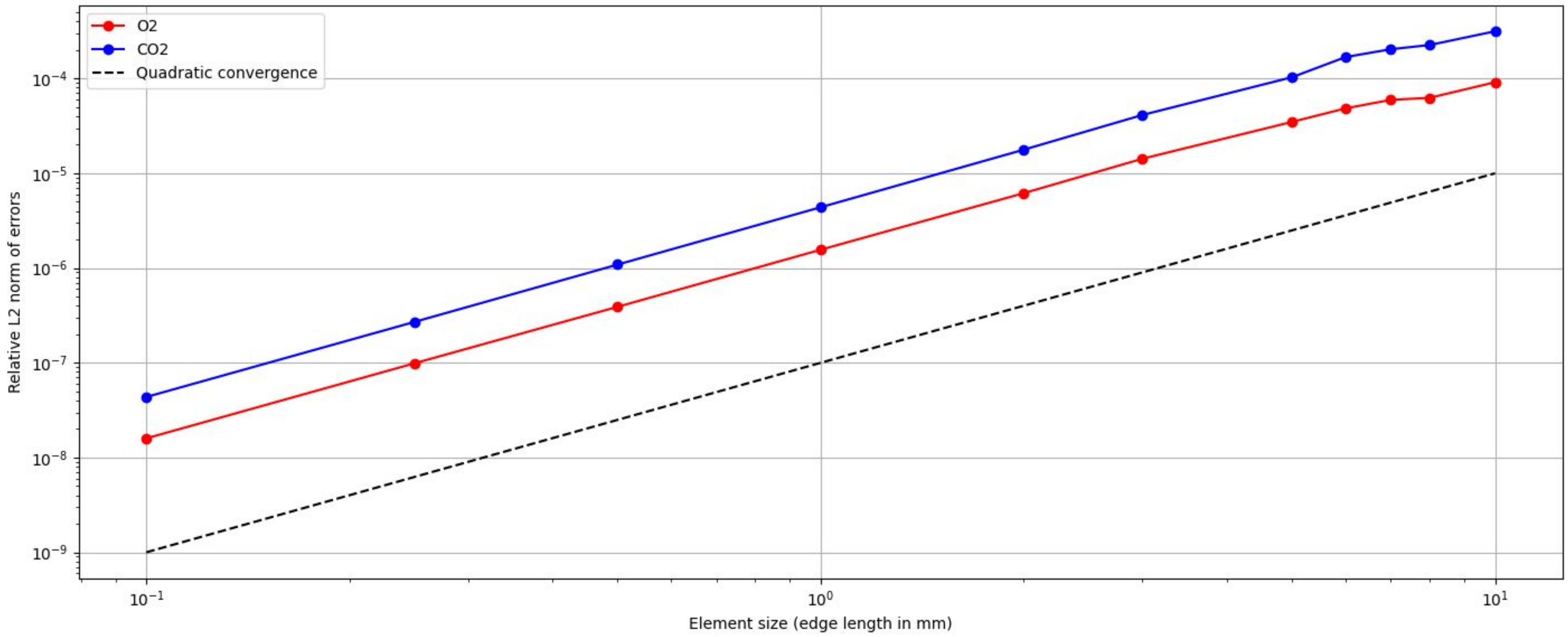
Boundary conditions: enforced by proposed solution

Change contributions of integrals for both initial guess and Newton iterations:

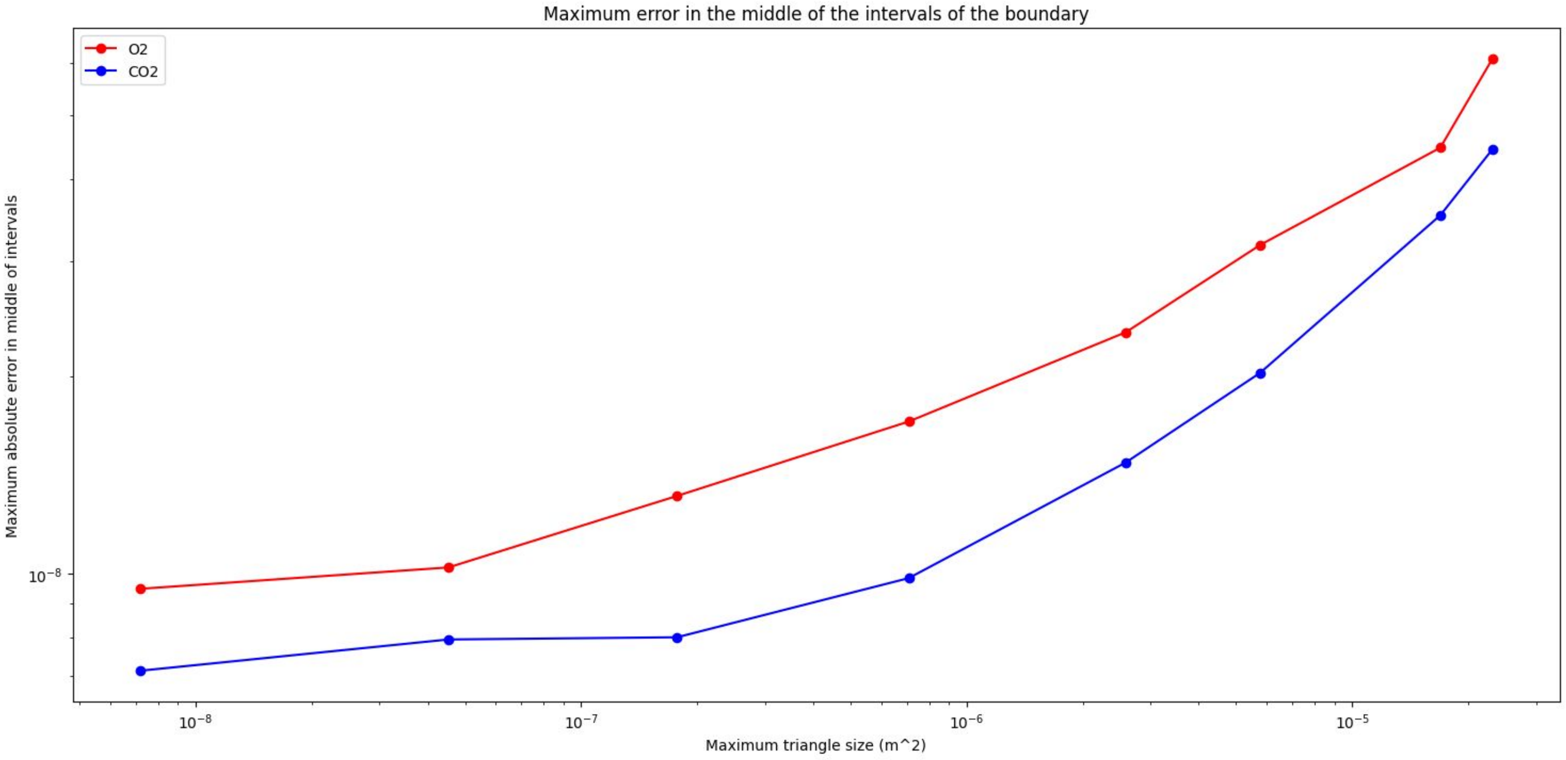
Qu, Qv, boundary conditions

$$\begin{pmatrix} K_u & 0 \\ 0 & K_v \end{pmatrix} \begin{pmatrix} \mathbf{c}_u \\ \mathbf{c}_v \end{pmatrix} - \begin{pmatrix} \mathbf{f}_u \\ \mathbf{f}_v \end{pmatrix} + \begin{pmatrix} \mathbf{H}_u(\mathbf{c}_u, \mathbf{c}_v) \\ \mathbf{H}_v(\mathbf{c}_u, \mathbf{c}_v) \end{pmatrix} = \begin{pmatrix} \mathbf{0} \\ \mathbf{0} \end{pmatrix}$$

Testing: Manufactured Solutions



Testing: boundary conditions



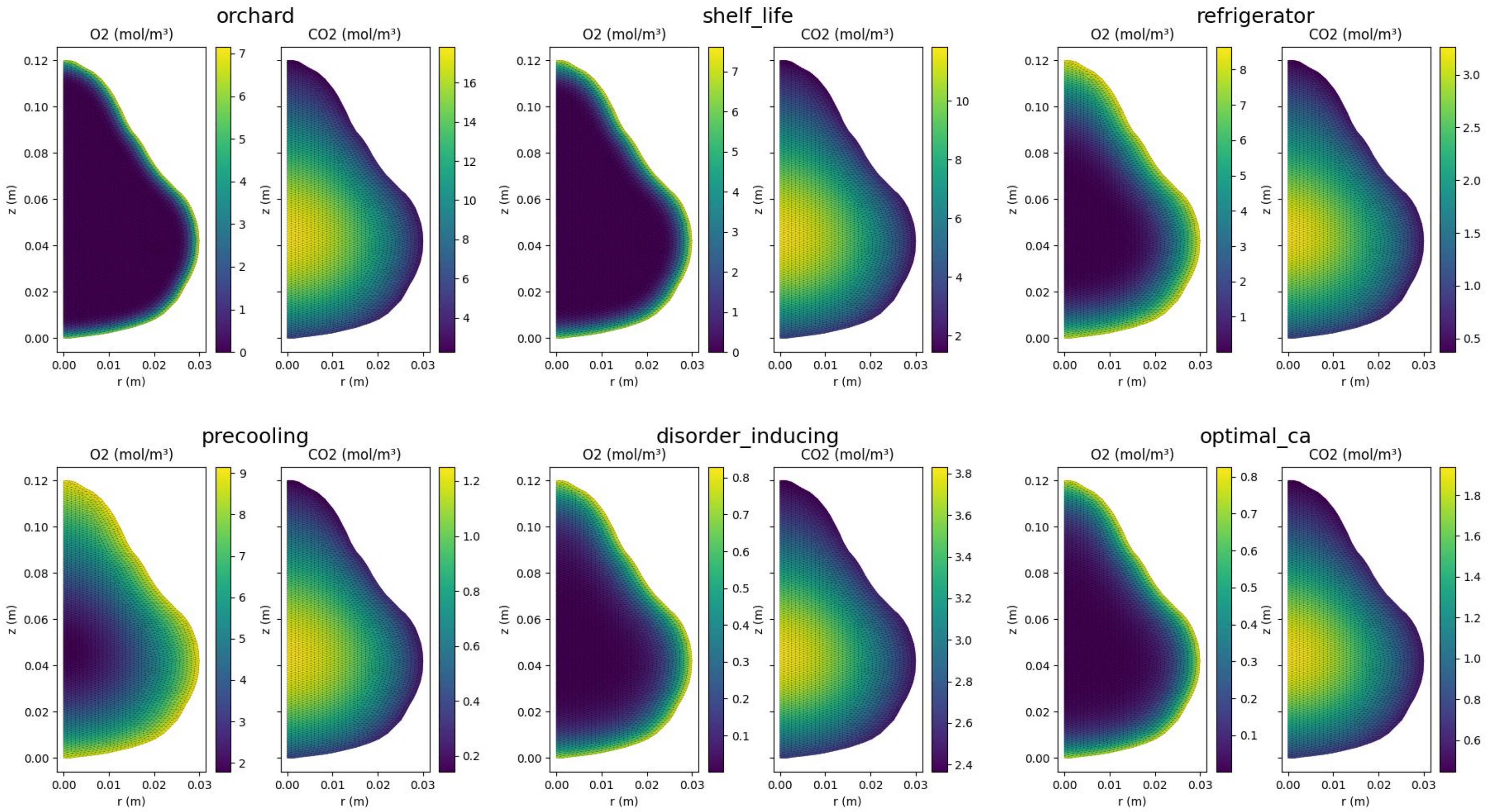
KU LEUVEN

Execution time

Uniform mesh edge length	Matlab Timing (s)	C++ Timing (s)
3 mm	2.230 (σ^2 = 0.0035)	0.0307 (σ^2 = 1.84e-6)
2 mm	5.126 (σ^2 = 0.022)	0.0826 (σ^2 = 1.35e-5)
1 mm	22.951 (σ^2 = 0.639)	0.471 (σ^2 = 4.63e-5)
0.5 mm	145.576 (σ^2 = 52.63)	2.978 (σ^2 = 0.00106)

*Timings done with ‘precooling’ option and 10 newton iterations, averaged over 20 runs

Solving sparse system is bottleneck for finer meshes


orchard
shelf_life
refrigerator
precooling
disorder_inducing
optimal_ca
O2 (mol/m³)
CO2 (mol/m³)
z (m)
r (m)