

# Project Mathematical Engineering



**Taoufik Benyahia, Michiel De Koninck**

*Faculty of Engineering Science*

*Mathematical Engineering*

*09/04/2019*

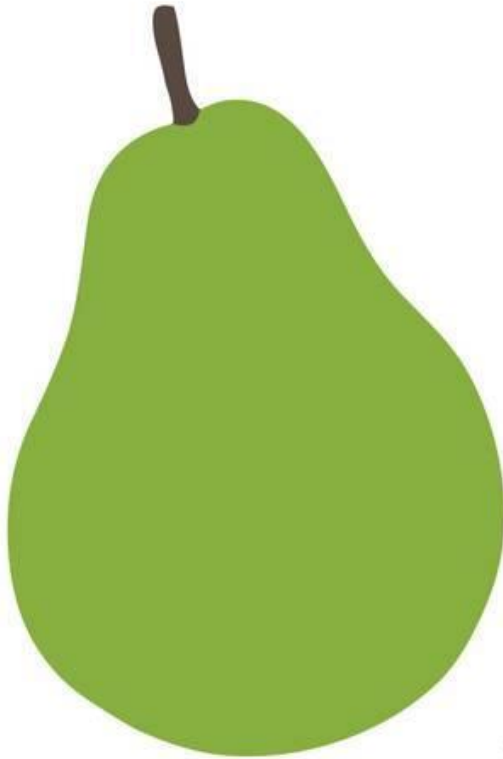
# Pear Project: Contents

---

1. Mathematical Problem
2. Description of FE method
3. Construction of linear system
4. Demonstration of result for linear system
5. Solving of nonlinear system
6. Demonstration of results of nonlinear problem



# 1. Mathematical Problem



- Challenge: Simulate and predict internal gas concentrations/distributions
- Study the effect of controlled storage conditions on local oxygen and carbon dioxide concentrations

# 1. Mathematical Problem: Model

- **Model** used for system:
  - Non-linear **reaction-diffusion equations**: in stationary regime
  - On **two-dimensional bounded** spatial domain
  - u: oxygen    v:carbon-dioxide

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

# 1. Mathematical Problem: Boundary Conditions

## • Boundary Conditions:

- Internal Boundary:  $(C_u^*(r, z), C_v^*(r, z)) = (0, 0)$
- External Boundary:  $(C_u^*(r, z), C_v^*(r, z)) = (C_u(r, z) - C_{uamb}, C_v(r, z) - C_{vamb})$

Ambient  
concentrations



$$\begin{cases} -\vec{n}(r, z) \cdot \begin{pmatrix} \sigma_{u,r} & 0 \\ 0 & \sigma_{u,z} \end{pmatrix} \nabla C_u(r, z) \\ -\vec{n}(r, z) \cdot \begin{pmatrix} \sigma_{v,r} & 0 \\ 0 & \sigma_{v,z} \end{pmatrix} \nabla C_v(r, z) \end{cases} = \begin{cases} \varrho_u C_u^*(r, z) \\ \varrho_v C_v^*(r, z) \end{cases} \quad \text{for } (r, z) \in \Gamma,$$

# 1. Mathematical Problem: Respiration kinetics

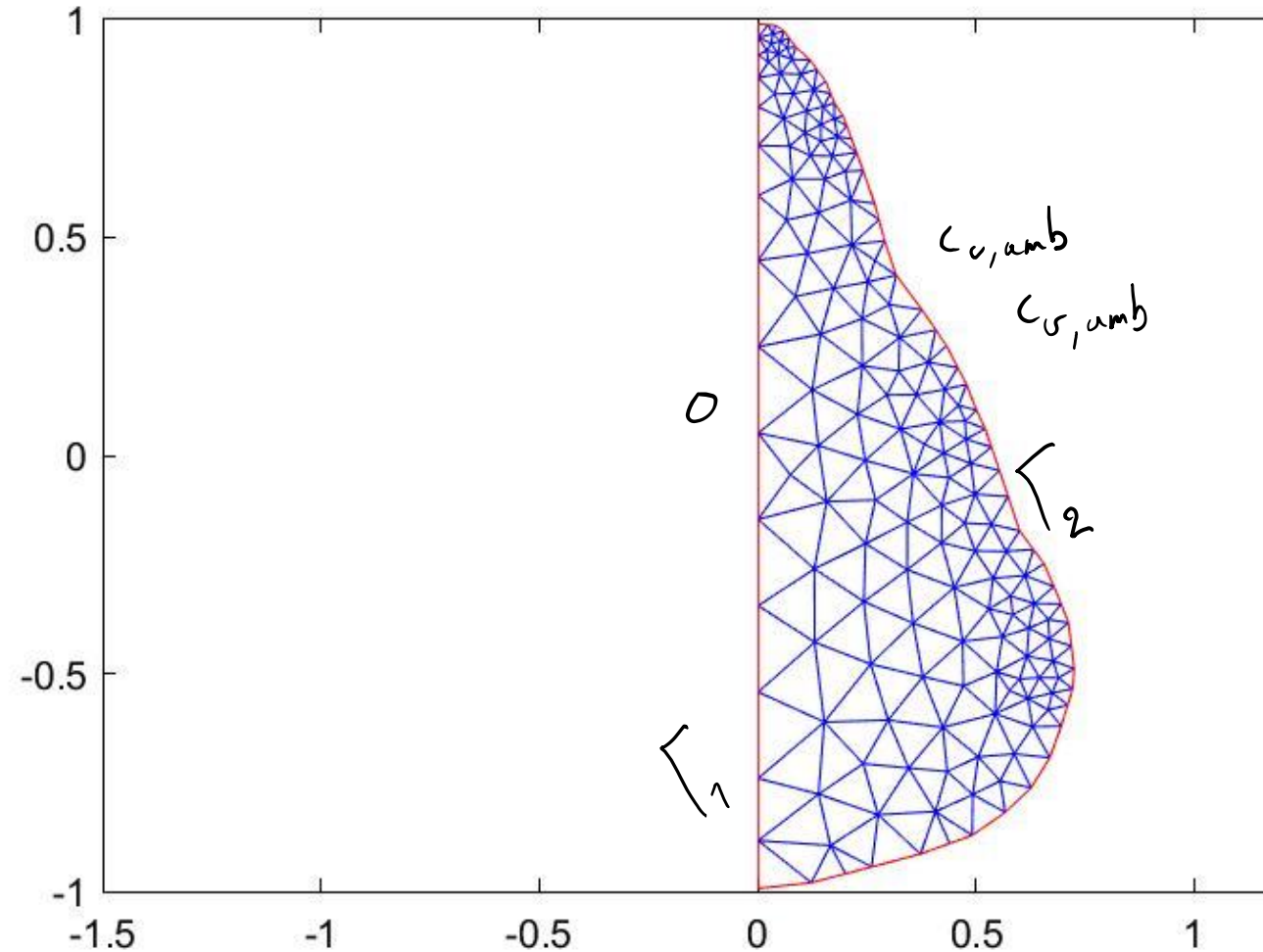
- **Respiration Kinetics:**

Describe the **evolution** of the system through **gas exchange** with the environment

$$\left\{ \begin{array}{l} R_u(C_u, C_v) = \frac{V_{mu}C_u}{(K_{mu} + C_u) \left(1 + \frac{C_v}{K_{mv}}\right)} \\ R_v(C_u, C_v) = r_q R_u(C_u, C_v) + \frac{V_{mfv}}{1 + \frac{C_u}{K_{mfu}}} \end{array} \right.$$

## 2. Description of FE-method

- Domain is split up into triangles consisting of  $M$  nodes
- We want to approximate the solutions at the nodes of our mesh by use of linear basisfunctions
- In every triangle we use three basisfunctions to represent our solution.
- On every edge we use two basisfunctions to represent our solution.



## 2. Description of FE-method

$$\vec{q}_u(r, z) = r \begin{pmatrix} \sigma_{u,r} & 0 \\ 0 & \sigma_{u,z} \end{pmatrix} \nabla C_u(r, z)$$

$$C_u^M(r, z) := \sum_{j=1}^M c_j \varphi_j(r, z),$$

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

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

$\Omega$  : Internal  
 $\Gamma$  : Edge

We solve these integrals by looping over all triangles/edges. The integral becomes a sum of integrals.



## 2. Description of FE-method

- For example start from:  $\int_{\Omega} \vec{q}_u(r, z) \cdot \nabla \varphi(r, z) d\Omega$

- Each triangle is mapped onto standard triangle:  
with  $\mathcal{L} = x_1^T \hat{N}_1 + x_2^T \hat{N}_2 + x_3^T \hat{N}_3$

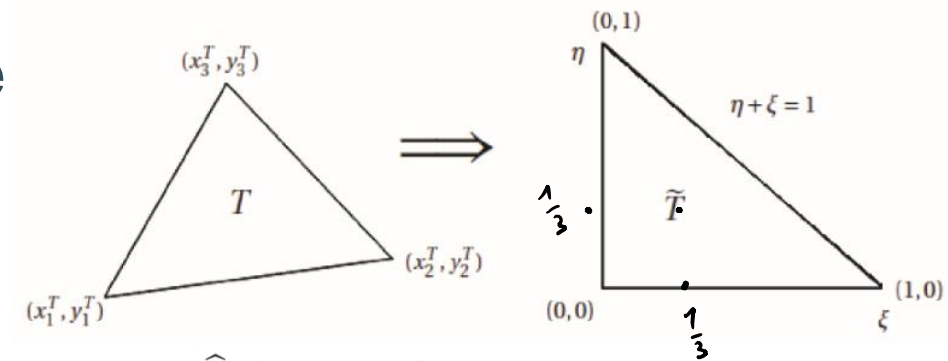
- With derivatives of basisfunctions in each triangle

with:  $|J| = \begin{vmatrix} \frac{\partial x}{\partial \xi} & \frac{\partial x}{\partial \eta} \\ \frac{\partial y}{\partial \xi} & \frac{\partial y}{\partial \eta} \end{vmatrix} = \begin{vmatrix} x_2^T - x_1^T & x_3^T - x_1^T \\ y_2^T - y_1^T & y_3^T - y_1^T \end{vmatrix}$

$$\vec{q}_u(r, z) = r \begin{pmatrix} \sigma_{u,r} & 0 \\ 0 & \sigma_{u,z} \end{pmatrix} \nabla C_u(r, z)$$

$$C_u^M(r, z) := \sum_{j=1}^M c_j \varphi_j(r, z),$$

$$\begin{aligned} \frac{\partial N_1}{\partial x} &= \frac{1}{|J|} (y_2^T - y_3^T) & \frac{\partial N_1}{\partial y} &= \frac{1}{|J|} (x_3^T - x_2^T) \\ \frac{\partial N_2}{\partial x} &= \frac{1}{|J|} (y_3^T - y_1^T) & \frac{\partial N_2}{\partial y} &= -\frac{1}{|J|} (x_3^T - x_1^T) \\ \frac{\partial N_3}{\partial x} &= -\frac{1}{|J|} (y_2^T - y_1^T) & \frac{\partial N_3}{\partial y} &= \frac{1}{|J|} (x_2^T - x_1^T) \end{aligned}$$



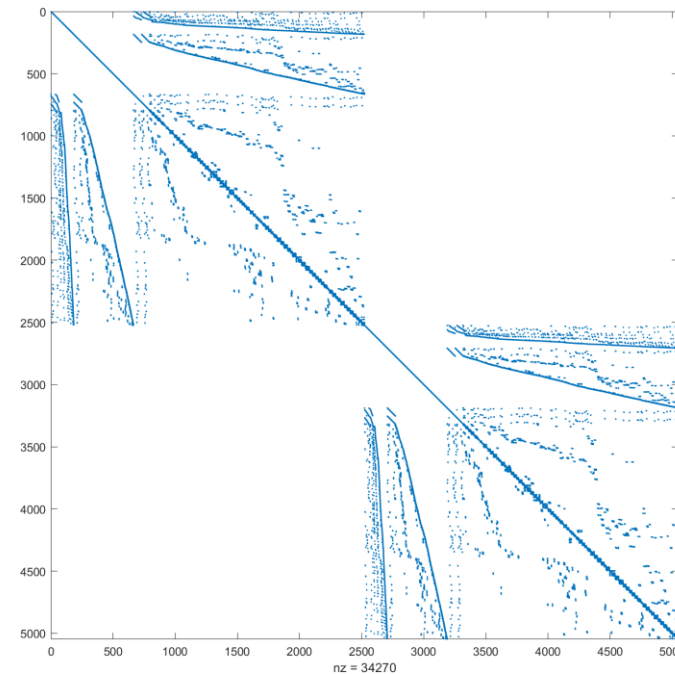
$$\hat{N}_1(\xi, \eta) = 1 - \xi - \eta$$

$$\hat{N}_2(\xi, \eta) = \xi$$

$$\hat{N}_3(\xi, \eta) = \eta$$

# 3. Construction of Linear System

- Using this technique we fill the matrix of the linear system (K):  
-> 9 adjustments (combinations of 3 nodes) are made to  $K_u$  and  $K_v$  for every triangle handled.
- Sparsity pattern is obtained:



### 3. Construction of Linear System

- Integral over edge:  $\int_{\Gamma \rightarrow \Gamma_2} r \varrho_u C_u^*(r, z) \varphi(r, z) d\Gamma$

$$\hat{N}_1(\xi, \eta) = \xi$$

$$\hat{N}_2(\xi, \eta) = 1 - \xi$$

- Map edge onto standard line

- Two basisfunctions instead of three:  $\mathcal{U} = x_1^T \hat{N}_1 + x_2^T \hat{N}_2$

- Integral for one edge translates to:  $\int_0^1 r \varrho_u C_u^*(\xi, \eta) \hat{N}(\xi, \eta) d\xi$

- This integral can be computed (mupad)

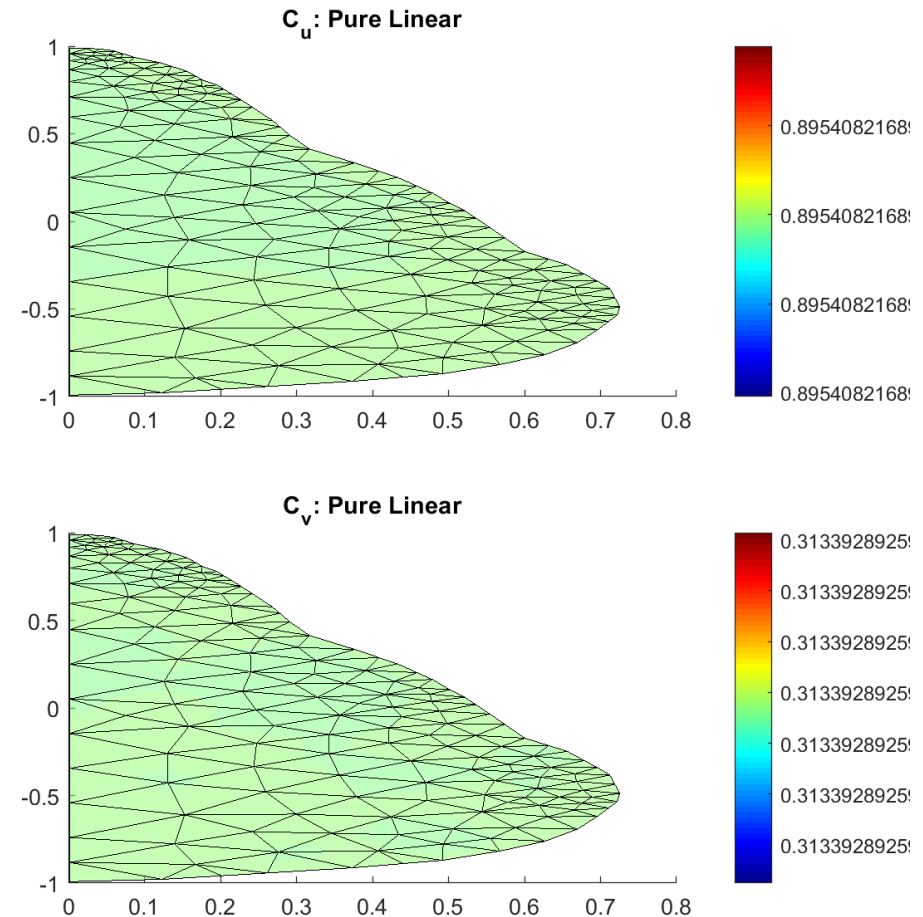
- For every edge: two adjustments are made to  $F_u$  and  $F_v$  as well as to  $K_u$  and  $K_v$

- If we now calculate  $K^*c = F$  (ignore  $H$ ) then we find:  $c = [C_{uamb} \dots C_{vamb} \dots]$  (steady state)

### 3. Demonstration of Result for Linear System

- Initial plots give expected result  
->values near the ambient O2 and CO2 concentration

Figure generated with data from  
FORTRAN

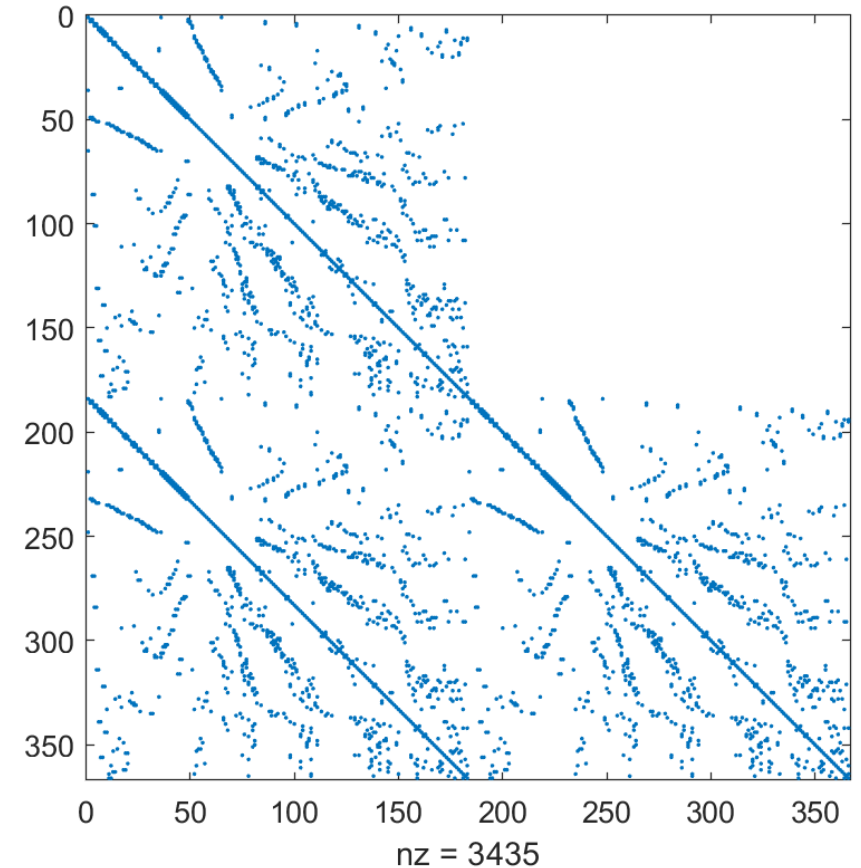


### 3. Construction of Linear System: Linearization

- Goal: find initial values ( $c_{u0}, c_{v0}$ ) to start newton iteration
- Linearize  $H_u$  and  $H_v$  around  $C_u = 0$  (no oxygen within pear, because it is turned into  $CO_2$  really fast)
  - > This leads to a simple expression for  $R_u$  and  $R_v$
- Contribution to the K-matrix and f-vector after which system can be solved again

# 3. Construction of Linear System: Linearization

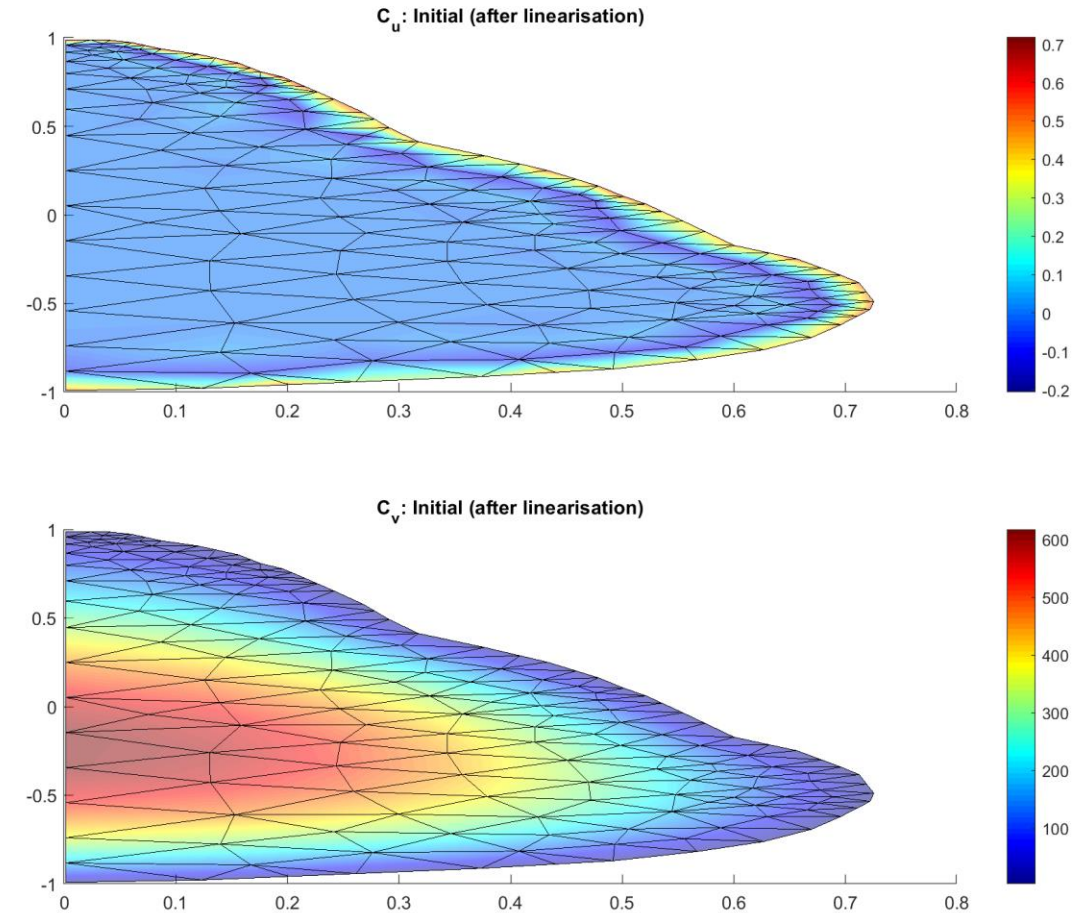
- The linearization causes changes to the K-matrix; its form becomes:
- Using this we can solve the linear system:  $c_0 = K \backslash f$ . In FORTRAN, this is solved by using a pivot-version of the Gauss algorithm.



# 4. Results for initial values after linearisation

- Using FORTRAN and MATLAB we get the same results.
- Both concentrations show expected evolution pattern.
- O2 concentration distribution is too heavily concentrated near the edges

Figure generated with data from FORTRAN



# 5. Solving Nonlinear System

- We use the Newton-Raphson Iterative Algorithm

$$F(c_u, c_v) = \begin{pmatrix} K_u & 0 \\ 0 & K_v \end{pmatrix} \begin{pmatrix} c_u \\ c_v \end{pmatrix} - \begin{pmatrix} f_u \\ f_v \end{pmatrix} + \begin{pmatrix} H_u(c_u, c_v) \\ H_v(c_u, c_v) \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$$

$$J(c_u, c_v) = \begin{bmatrix} K_u + \frac{\partial H_u}{\partial c_u} & \frac{\partial H_u}{\partial c_v} \\ \frac{\partial H_v}{\partial c_u} & K_v + \frac{\partial H_v}{\partial c_v} \end{bmatrix}$$

- $c_{k+1} = c_k - J^{-1}(c_k) \cdot F(c_k)$



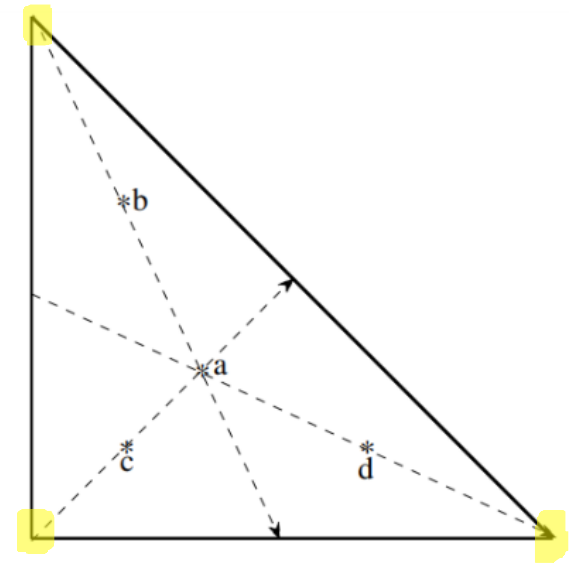
# 5. Solving Nonlinear System

- Used Gaussian (degree 3 precision) quadrature formula to approximate integral

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

$$C_u = C_{u_1} (1 - \xi - \eta) + C_{u_2} \xi + C_{u_3} \eta$$
$$C_v = C_{v_1} (1 - \xi - \eta) + C_{v_2} \xi + C_{v_3} \eta$$

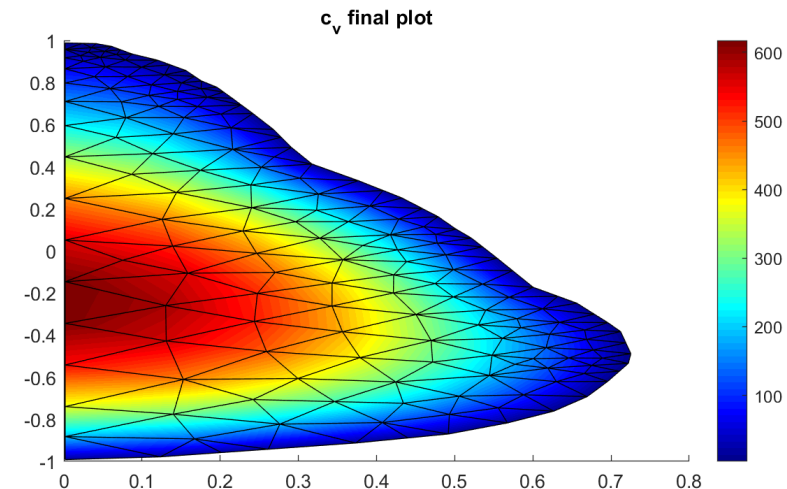
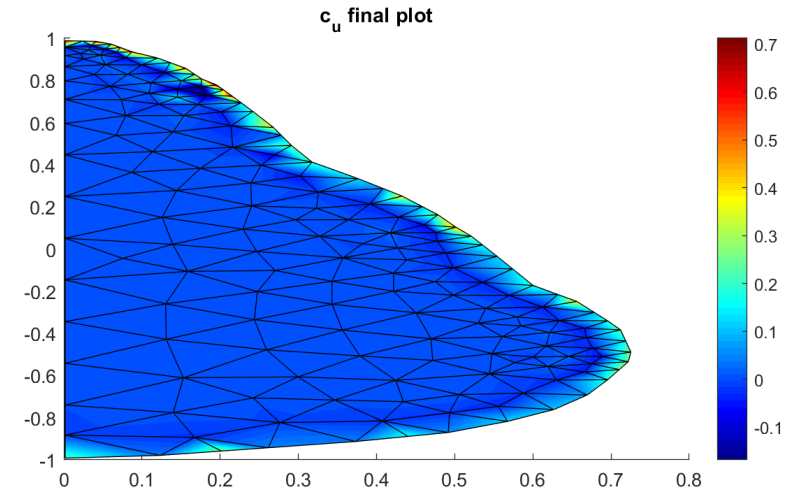
 = known values in vertices from previous iteration



# 6. Demonstration of results of nonlinear problem

- In FORTRAN: used MINPACK library to solve nonlinear system of equations (jacobian approximated by forward differences)
- In MATLAB: Calculated Jacobian analytically, result checked by calculating via finite differences function
- The distribution of the CO<sub>2</sub> concentration is as we would expect it.
- The distribution of the O<sub>2</sub> has not changed much from the initial values.

Figure generated with data from MATLAB



# Timing comparison: Fortran vs Matlab

- Using a maximum of 100 iterations for both ways of solving the non-linear system, we got following timings for the same pear:
  - FORTRAN: 0.3948s (average)
  - MATLAB: 5.3906s (average)

# Conclusion

- Problems:
  - Not completely correct initial concentration (linearisation assumptions?)
  - Convergence to wrong solutions when solving non-linear system
- Insights
  - More detailed understanding of Finite Element Method
  - Use of quadrature rules in practice
  - Structured MATLAB-programming
  - Structured FORTRAN-programming and working with library (MINPACK)
  - FORTRAN is much faster than MATLAB (also because of efficient solver)