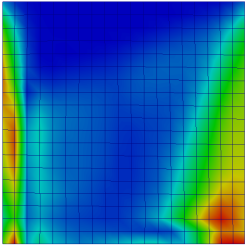
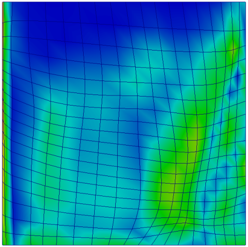
Output-based R-refinement and the Use of Geometric Multigrid for Truncation Error Estimation in CFD.

Mateusz Gugala, Jens-Dominik Müller

Queen Mary University of London,
School of Engineering and Material Science



September, 2015



Contents

Introduction / background

- Part of AboutFlow project
  Adjoint based optimisation for unsteady and industrial flows.
- My project goals:
  - goal: reduce adjoint-based optimisation run time
  - method: use **mesh adaptation** and one-shot approach.
  - Start optimisation with low-fidelity solutions, go towards high-fidelity when converging to optimum.
- Working with an in-house code developed at QMUL:
  - Finite Volume, vertex centred
  - Steady/Unsteady, Inviscid/Laminar/Turbulent (SA model)
  - **Geometric multigrid**, Explicit/Implicit (JT-KIRK[1]) solver
  - 2^{nd} spatial accuracy, $ROE/AUSM_{+}^{up}$ flux
  - Discrete adjoint solver (*Tapenade*[2])

[1] Shenren Xu, Jens-Dominik Müller, Stabilisation of the discrete steady adjoint solver using Jacobian-Trained Krylov-Implicit-Runge-Kutta (JT-KIRK) algorithm, (in revision)

[2] AD tool developed at Inria http://www-sop.inria.fr/tropics/

Mesh adaptation - intro

Mesh adaptation and error estimation allows to increase reliability of numerical simulation. Main aspects in mesh adaptation:

- Error estimation
    - discretisation error (δU)
    - truncation error (δR)
    - output error (δL)
- Adaptation indicator evaluation (scalar, metric)
    - feature-based: uses some form of discretisation error
    - truncation-based: direct use of some form of truncation error
    - **output-based: e.g. adjoint-weighted truncation error**
- Adaptation method/algorithm:
    - **r-refinement: relocate mesh nodes, keep mesh size constant**
    - h-refinement: refine mesh by subdividing cells/edges
    - p-refienemnt: changing order of discretisation polynomial

Mesh adaptation - intro

Mesh adaptation and error estimation allows to increase reliability of numerical simulation. Main aspects in mesh adaptation:

- Error estimation
    - discretisation error (δU)
    - truncation error (δR)
    - output error (δL)
- Adaptation indicator evaluation (scalar, metric)
    - feature-based: uses some form of discretisation error
    - truncation-based: direct use of some form of truncation error
    - **output-based: e.g. adjoint-weighted truncation error**
- Adaptation method/algorithm:
    - **r-refinement: relocate mesh nodes, keep mesh size constant**
    - h-refinement: refine mesh by subdividing cells/edges
    - p-refienemnt: changing order of discretisation polynomial

Mesh adaptation - intro

Mesh adaptation and error estimation allows to increase reliability of numerical simulation. Main aspects in mesh adaptation:

- Error estimation
  - discretisation error (δU)
  - truncation error (δR)
  - output error (δL)
- Adaptation indicator evaluation (scalar, metric)
  - feature-based: uses some form of discretisation error
  - truncation-based: direct use of some form of truncation error
  - **output-based: e.g. adjoint-weighted truncation error**
- Adaptation method/algorithm:
  - **r-refinement: relocate mesh nodes, keep mesh size constant**
  - h-refinement: refine mesh by subdividing cells/edges
  - p-refienemnt: changing order of discretisation polynomial

Mesh adaptation - intro

Mesh adaptation and error estimation allows to increase reliability of numerical simulation. Main aspects in mesh adaptation:

- Error estimation
  - discretisation error (δU)
  - truncation error (δR)
  - output error (δL)
- Adaptation indicator evaluation (scalar, metric)
  - feature-based: uses some form of discretisation error
  - truncation-based: direct use of some form of truncation error
  - **output-based: e.g. adjoint-weighted truncation error**
- Adaptation method/algorithm:
  - **r-refinement: relocate mesh nodes, keep mesh size constant**
  - h-refinement: refine mesh by subdividing cells/edges
  - p-refienemnt: changing order of discretisation polynomial

Adaptation and Multigrid

Output-based error estimation - math

Discretisation error (δU_h):

$$\delta U_h = U - U_h \tag{1}$$

Truncation Error (δR_h):

$$R(U) - R(U_h) = \left.\frac{\partial R}{\partial U}\right|_{U_h} (U - U_h) + \dots \tag{2}$$

Output error (δJ_h):

$$J(U) - J(U_h) = \left.\frac{\partial J}{\partial U}\right|_{U_h} (U - U_h) + \dots \tag{3}$$

Replacing δU_h in (3) with discretisation error in (2):

$$\delta J_h \approx \left.\frac{\partial J}{\partial U}\right|_{U_h} \left.\frac{\partial R}{\partial U}^{-1}\right|_{U_h} \delta R_h = - \left. v_h^T \right|_{U_h} \delta R_h \tag{4}$$

Output-based error estimation - math

Discretisation error (δU_h):

$$\delta U_h = U - U_h \tag{1}$$

Truncation Error (δR_h):

$$R(U) - R(U_h) = \left.\frac{\partial R}{\partial U}\right|_{U_h} (U - U_h) + \dots \tag{2}$$

Output error (δJ_h):

$$J(U) - J(U_h) = \left.\frac{\partial J}{\partial U}\right|_{U_h} (U - U_h) + \dots \tag{3}$$

Replacing δU_h in (3) with discretisation error in (2):

$$\delta J_h \approx \left.\frac{\partial J}{\partial U}\right|_{U_h} \left.\frac{\partial R}{\partial U}^{-1}\right|_{U_h} \delta R_h = -\left. v_h^T\right|_{U_h} \delta R_h \tag{4}$$

Output-based error estimation - math

Discretisation error (δU_h):

$$\delta U_h = U - U_h \tag{1}$$

Truncation Error (δR_h):

$$R(U) - R(U_h) = \left.\frac{\partial R}{\partial U}\right|_{U_h} (U - U_h) + \ldots \tag{2}$$

Output error (δJ_h):

$$J(U) - J(U_h) = \left.\frac{\partial J}{\partial U}\right|_{U_h} (U - U_h) + \ldots \tag{3}$$

Replacing δU_h in (3) with discretisation error in (2):

$$\delta J_h \approx \left.\frac{\partial J}{\partial U}\right|_{U_h} \left.\frac{\partial R}{\partial U}^{-1}\right|_{U_h} \delta R_h = - \left.v_h^T\right|_{U_h} \delta R_h \tag{4}$$

Output-based error estimation - math

Discretisation error (δU_h):

$$\delta U_h = U - U_h \quad (1)$$

Truncation Error (δR_h):

$$R(U) - R(U_h) = \left.\frac{\partial R}{\partial U}\right|_{U_h} (U - U_h) + \dots \quad (2)$$

Output error (δJ_h):

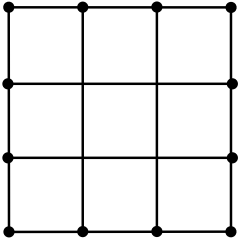
$$J(U) - J(U_h) = \left.\frac{\partial J}{\partial U}\right|_{U_h} (U - U_h) + \dots \quad (3)$$

Replacing δU_h in (3) with discretisation error in (2):

$$\delta J_h \approx \left.\frac{\partial J}{\partial U}\right|_{U_h} \left.\frac{\partial R}{\partial U}^{-1}\right|_{U_h} \delta R_h = - \left. v_h^T \right|_{U_h} \delta R_h \quad (4)$$

Error estimation with multigrid - procedure

Fine mesh (h)

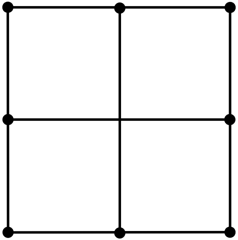


$U_h \xrightarrow{I_h^H} U_H^h$

$R_h^H \xleftarrow{I_H^h} R_H(U_H^h)$

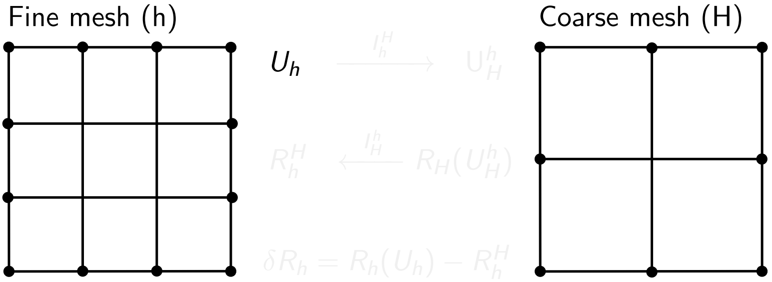
$\delta R_h = R_h(U_h) - R_h^H$

Coarse mesh (H)



1. Solve primal
2. Restrict primal solution to coarse mesh
3. Calculate residual on coarse mesh
4. Prolong residual to fine mesh
5. Calculate truncation error (Note: $R_h(U_h) = 0$ at convergence)

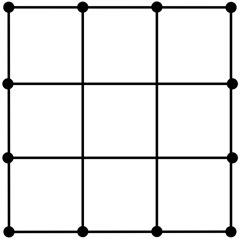
Error estimation with multigrid - procedure



1. Solve primal
2. Restrict primal solution to coarse mesh
3. Calculate residual on coarse mesh
4. Prolong residual to fine mesh
5. Calculate truncation error (Note: $R_h(U_h) = 0$ at convergence)

Error estimation with multigrid - procedure

Fine mesh (h)

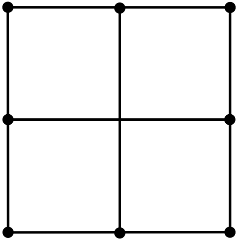


$$U_h \xrightarrow{I_h^H} U_H^h$$

$$R_h^H \xleftarrow{I_H^h} R_H(U_H^h)$$

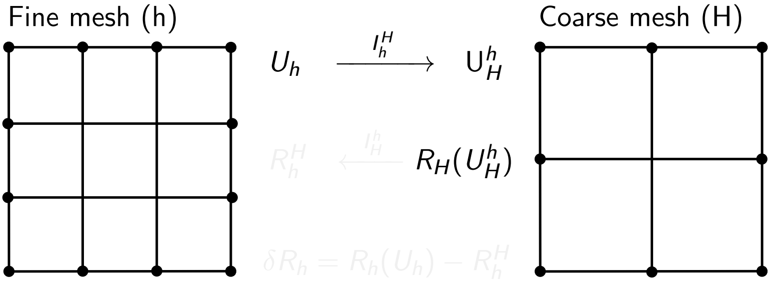
$$\delta R_h = R_h(U_h) - R_h^H$$

Coarse mesh (H)



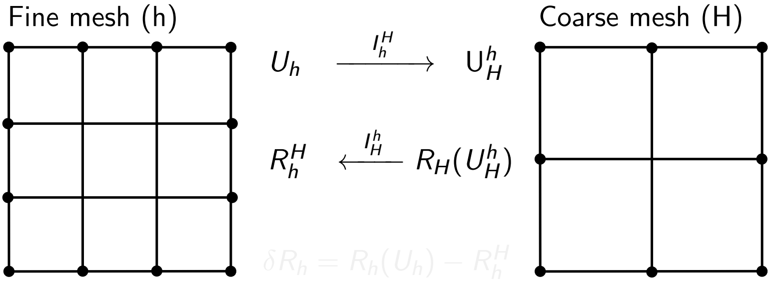
1. Solve primal
2. Restrict primal solution to coarse mesh
3. Calculate residual on coarse mesh
4. Prolong residual to fine mesh
5. Calculate truncation error (Note: $R_h(U_h) = 0$ at convergence)

Error estimation with multigrid - procedure



1. Solve primal
2. Restrict primal solution to coarse mesh
3. Calculate residual on coarse mesh
4. Prolong residual to fine mesh
5. Calculate truncation error (Note: $R_h(U_h) = 0$ at convergence)

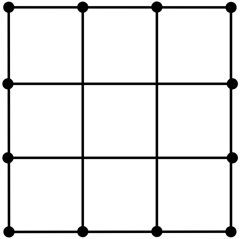
Error estimation with multigrid - procedure



1. Solve primal
2. Restrict primal solution to coarse mesh
3. Calculate residual on coarse mesh
4. Prolong residual to fine mesh
5. Calculate truncation error (Note: $R_h(U_h) = 0$ at convergence)

Error estimation with multigrid - procedure

Fine mesh (h)

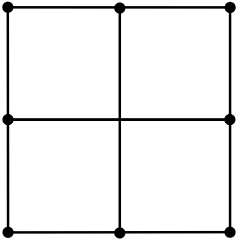


$$U_h \xrightarrow{I_h^H} \mathrm{U}_H^h$$

$$R_h^H \xleftarrow{I_H^h} R_H(U_H^h)$$

$$\delta R_h = R_h(U_h) - R_h^H$$

Coarse mesh (H)



1. Solve primal
2. Restrict primal solution to coarse mesh
3. Calculate residual on coarse mesh
4. Prolong residual to fine mesh
5. Calculate truncation error (Note: $R_h(U_h) = 0$ at convergence)

Adaptation sensor/indicator

Output error estimation:

$$\delta J_h \approx - v_h^T\Big|_{U_h} \delta R_h \tag{5}$$

Solve adjoint on fine mesh:

$$\frac{\partial R}{\partial U}^T\Bigg|_{U_h} v_h = - \frac{\partial J}{\partial U}^T\Bigg|_{U_h} \tag{6}$$

Adaptation indicator (adjoint-weighted truncation error):

$$\mathrm{I}_s = \left| \, v_h^T|_{U_h} \delta R_h \, \right| \tag{7}$$

Adaptation sensor/indicator

Output error estimation:

$$\delta J_h \approx - v_h^T\Big|_{U_h} \delta R_h \tag{5}$$

Solve adjoint on fine mesh:

$$\frac{\partial R}{\partial U}^T\Bigg|_{U_h} v_h = - \frac{\partial J}{\partial U}^T\Bigg|_{U_h} \tag{6}$$

Adaptation indicator (adjoint-weighted truncation error):

$$\mathrm{I}_s = \left| \, v_h^T\big|_{U_h} \delta R_h \, \right| \tag{7}$$

Adaptation sensor/indicator

Output error estimation:

$$\delta J_h \approx - \left. v_h^T \right|_{U_h} \delta R_h \qquad (5)$$

Solve adjoint on fine mesh:

$$\left. \frac{\partial R}{\partial U}^T \right|_{U_h} v_h = - \left. \frac{\partial J}{\partial U}^T \right|_{U_h} \qquad (6)$$

Adaptation indicator (adjoint-weighted truncation error):

$$\mathrm{I}_s = \left| \; v_h^T |_{U_h} \delta R_h \; \right| \qquad (7)$$

Testing

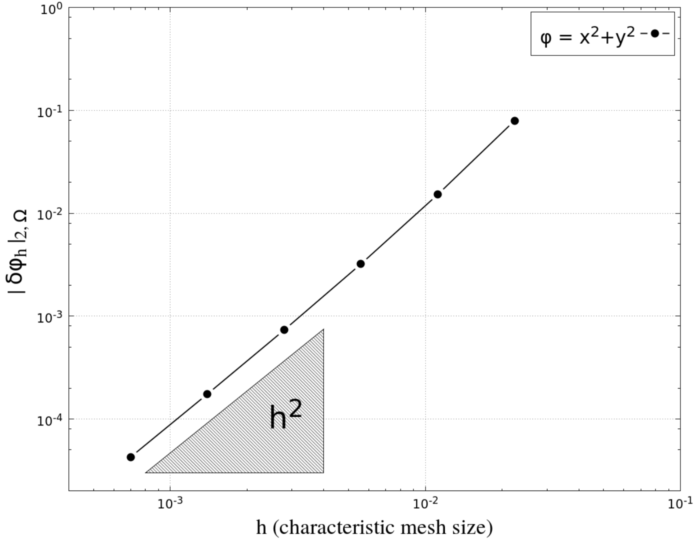
Interpolation operators

Order of accuracy of transfer operators should be at least consistent with discretisation accuracy of the solver to avoid loosing information when transferring it between meshes.

- Test restriction operator: I_h^H and prolongation operator: I_H^h
- Accuracy test (square domain 20x20 nodes): restrict constant/linear/quadratic fields to coarse mesh and prolong back to fine, check errors.
  - Constant field $\phi(x, y) = const$: PASSED (error ~ 0)
  - Linear field $\phi(x, y) = x + y$: PASSED (error ~ 0)
  - Quadratic field $\phi(x, y) = x^2 + y^2$:
    PASSED (error converging with $2^{nd} O$ slope)

$$\left|\phi_{|_h} - \phi_h^H\right|_2 \tag{8}$$

Interpolation operators



Manufactured solution (M.S.) - intro

1. Define 'made up' solution: $U = U(X),\ X = [x, y, z]$
2. Define continuous system of equations, e.g Euler's: $R(U)$
3. Calc. sources arising from 'made up' sol.: $R(U(X)) = f(X)$
4. Solve discrete system, subject to Dirichlet BC's:
5. $lim_{h \to 0}(\delta U_h) = 0,\ \delta U_h = U(X)|_h - U_h$

$$R_h(U_h) - f(X)|_h = 0, \quad U(X)|_{\partial\Omega_h}$$

$$f(X)_h$$

$$U(X)_{\partial\Omega_h}$$

Manufactured solution (M.S.) - intro

1. Define 'made up' solution: $U = U(X),\ X = [x, y, z]$
2. Define continuous system of equations, e.g Euler's: $R(U)$
3. Calc. sources arising from 'made up' sol.: $R(U(X)) = f(X)$
4. Solve discrete system, subject to Dirichlet BC's:
5. $lim_{h \to 0}(\delta U_h) = 0,\ \delta U_h = U(X)|_h - U_h$

$$R_h(U_h) - f(X)|_h = 0, \quad U(X)|_{\partial\Omega_h}$$

$$f(X)_h$$

$$U(X)_{\partial\Omega_h}$$

Manufactured solution (M.S.) - intro

1. Define 'made up' solution: $U = U(X), \; X = [x, y, z]$
2. Define continuous system of equations, e.g Euler's: $R(U)$
3. Calc. sources arising from 'made up' sol.: $R(U(X)) = f(X)$
4. Solve discrete system, subject to Dirichlet BC's:
5. $lim_{h \to 0}(\delta U_h) = 0, \; \delta U_h = U(X)|_h - U_h$

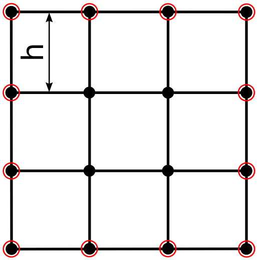
$$R_h(U_h) - f(X)|_h = 0, \quad U(X)|_{\partial\Omega_h}$$

$$f(X)_h$$

$$U(X)_{\partial\Omega_h}$$

Manufactured solution (M.S.) - intro

1. Define 'made up' solution: $U = U(X),\ X = [x, y, z]$
2. Define continuous system of equations, e.g Euler's: $R(U)$
3. Calc. sources arising from 'made up' sol.: $R(U(X)) = f(X)$
4. Solve discrete system, subject to Dirichlet BC's:
5. $lim_{h\to 0}(\delta U_h) = 0,\ \delta U_h = U(X)|_h - U_h$



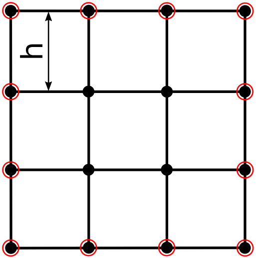
$$R_h(U_h) - f(X)|_h = 0, \quad U(X)|_{\partial\Omega_h}$$

● $f(X)_h$

○ $U(X)_{\partial\Omega_h}$

Manufactured solution (M.S.) - intro

1. Define 'made up' solution: $U = U(X),\ X = [x, y, z]$
2. Define continuous system of equations, e.g Euler's: $R(U)$
3. Calc. sources arising from 'made up' sol.: $R(U(X)) = f(X)$
4. Solve discrete system, subject to Dirichlet BC's:
5. $lim_{h\to 0}(\delta U_h) = 0,\ \delta U_h = U(X)|_h - U_h$



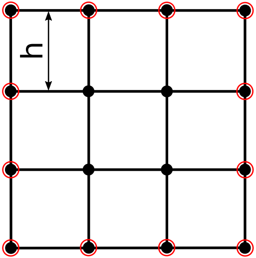
$$R_h(U_h) - f(X)|_h = 0, \quad U(X)|_{\partial\Omega_h}$$

- $f(X)_h$

○ $U(X)_{\partial\Omega_h}$

Manufactured solution (M.S.) - intro

1. Define 'made up' solution: $U = U(X),\ X = [x, y, z]$
2. Define continuous system of equations, e.g Euler's: $R(U)$
3. Calc. sources arising from 'made up' sol.: $R(U(X)) = f(X)$
4. Solve discrete system, subject to Dirichlet BC's:
5. $lim_{h \to 0}(\delta U_h) = 0,\ \delta U_h = U(X)|_h - U_h$

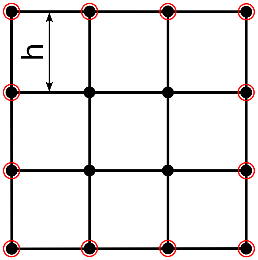


$$R_h(U_h) - f(X)|_h = 0, \quad U(X)|_{\partial\Omega_h}$$

- ● $f(X)_h$
- ○ $U(X)_{\partial\Omega_h}$

Manufactured solution (M.S.) - intro

1. Define 'made up' solution: $U = U(X),\ X = [x, y, z]$
2. Define continuous system of equations, e.g Euler's: $R(U)$
3. Calc. sources arising from 'made up' sol.: $R(U(X)) = f(X)$
4. Solve discrete system, subject to Dirichlet BC's:
5. $lim_{h \to 0}(\delta U_h) = 0,\ \delta U_h = \ U(X)|_h - U_h$



$$R_h(U_h) - \ f(X)|_h = 0, \ \ U(X)|_{\partial\Omega_h}$$

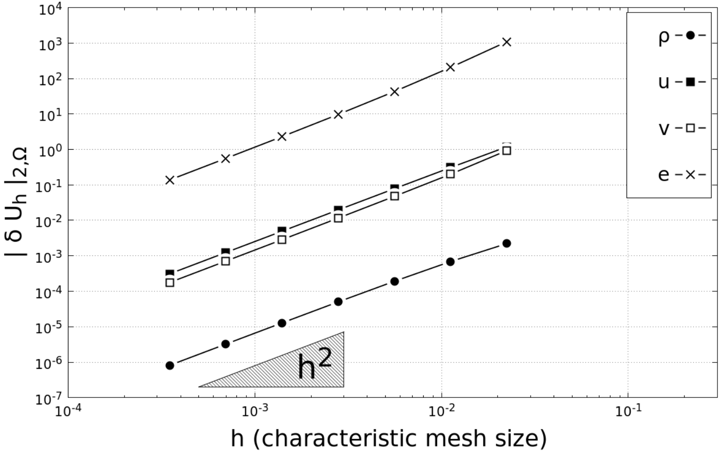
- $f(X)_h$
- ○ $U(X)_{\partial\Omega_h}$

Manufactured solution (M.S.) - test

1. Use set of 7 square meshes, regular quad cells. (10x10 $\div$ 640x640 nodes)
2. Solver tests:
   - δU_h: Confirm accuracy of discretisation scheme to be $2^{nd} O$.
   - δJ_h: Check convergence of cost function error, (drag and lift force integrated over sides and bottom boundaries)
3. Truncation error estimation test:
   - δR_h: Check convergence of estimated truncation error

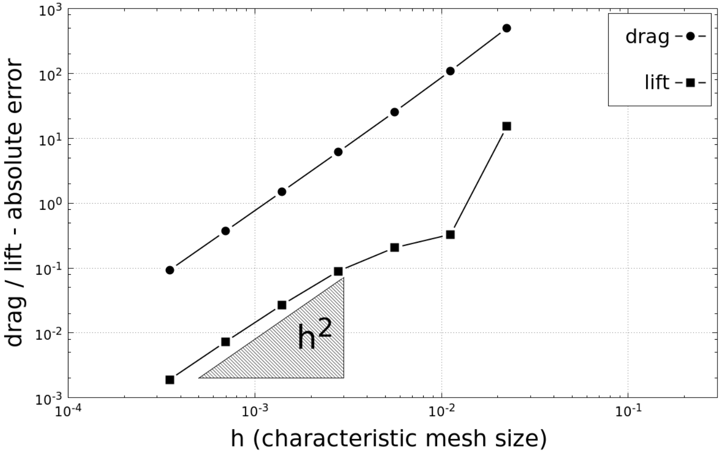
Discretisation error

- 2^{nd} order accuracy of the flow solver confirmed



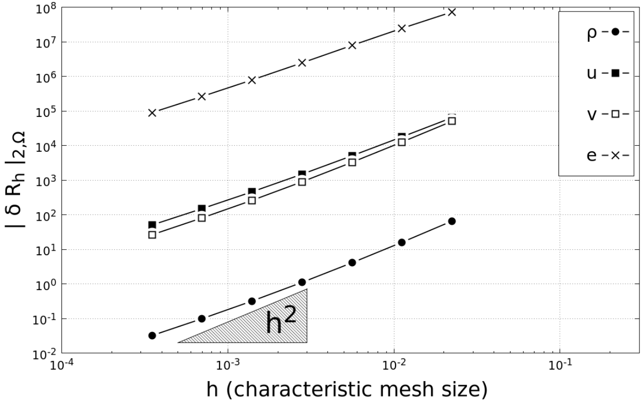
Cost function error

- Cost function converging with 2^{nd} order slope.



Estimated truncation error

- Estimated truncation error is converging (proposed method seems valid).
- As 'h' approaches 0, the slope will become 1^{st} order due to 1^{st} order accurate integration of residuals at boundaries.



Refinement

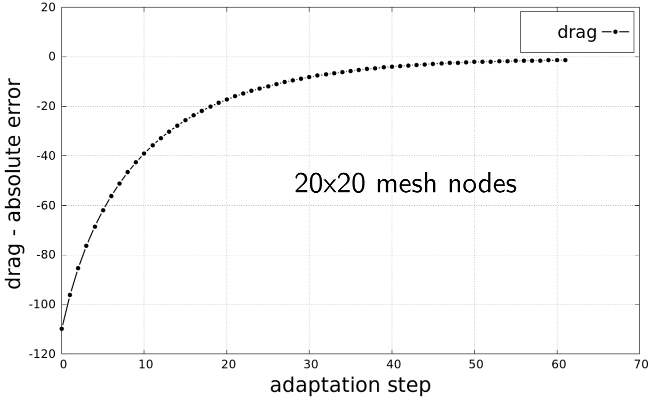
R-refinement for square domain (M.S.)

1. Solve flow: $R_h(U_h) = 0$
2. Solve adjoint: $R_h(v_h) = 0$
3. Estimate truncation error: δR_h
4. Get output-based adaptation indicator (scalar):
   $\mathrm{I}_s = \left| \; v_h^T|_{U_h} \delta R_h \; \right|$
5. Calculate gradient of scalar indicator field: $\nabla \mathrm{I}_s$
6. Mesh deformation using linear elasticity with body Force source term:
   $\nabla \sigma = f, \; f = \nabla \mathrm{I}_s$
7. Repeat until smallest edge length or volume is below specified thresholds

Note: mesh movement at domain boundary is allowed only in its tangent direction

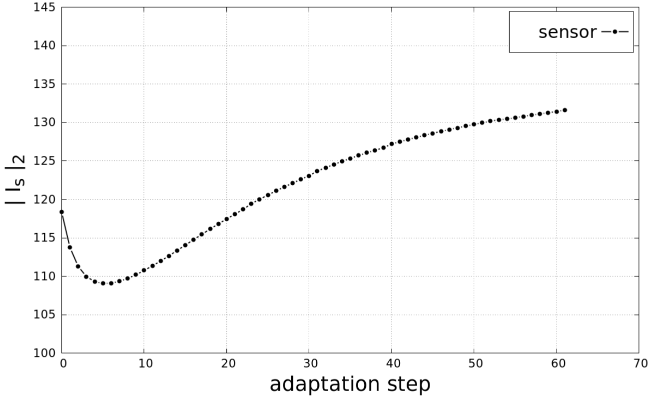
Cost function convergence

- Cost function converge towards exact value. Final error (-1.1)
- Achieved cost function error level corresponds to uniform 200x200 nodes mesh (100 times more degrees of freedom)



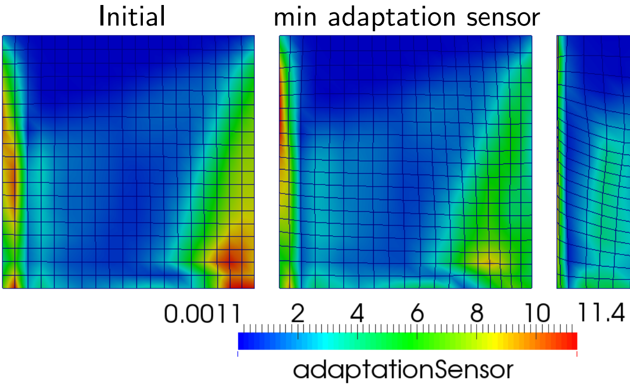
Adaptation sensor convergence

- Adaptation sensor norm rises after 5 adapt. steps
- Reason for this behaviour not known at the moment - to be investigated



Adaptation sensor contour plots

- Interior: adaptation sensor converging towards even distribution
- Boundaries: increase in adaptation sensor values at some boundary nodes



Summary

Conclusion / Future work

Summary:

1. The manufactured solution tests shows that the proposed method is estimating truncation error correctly.
2. When using geometric multigrid, the truncation error estimation is almost free.
3. When running adjoint-based optimisation we can obtain a robust output-based adaptation sensor.
4. Question to answer: why adaptation sensor norm is not decreasing?

Future work:

1. Perform r-refinement with 2D-naca and 3D-FEV test cases
2. Perform h-refinement with 2D-naca and 3D-FEV test cases
3. Go towards multi-fidelity optimisation case (combine on-shot approach and mesh refinement)

Industrial partners:

University partners:

Acknowledgement

"This project has received funding from the European Union's Seventh Framework Programme for research, technological development and demonstration under grant agreement no [317006]".

Research funded by the European Commission