Driving metric mesh adaptivity in a nonlinear minimization scheme

Iztok Bajc, Frédéric Hecht, Slobodan Žumer

Fakulteta za matematiko in fiziko
Univerza v Ljubljani
Slovenija

Laboratoire Jacques-Louis Lions
Université Pierre et Marie Curie
Paris 6, France

Inštitut Jožef Stefan
Ljubljana
Slovenija

Introduction

Introduction

Motivations:

Introduction

Motivations:

- Tendence to make simulations more and more **realistic**.

Introduction

Motivations:

- Tendence to make simulations more and more **realistic**.
- **Nonlinear problems** increasingly important.

Introduction

Motivations:

- Tendence to make simulations more and more **realistic**.
- **Nonlinear problems** increasingly important.
- In general not easy to solve. $\longrightarrow$ **Iterative methods** needed.

Introduction

Motivations:

- Tendence to make simulations more and more **realistic**.
- **Nonlinear problems** increasingly important.
- In general not easy to solve. $\longrightarrow$ **Iterative methods** needed.

- If we use **finite elements**:

Introduction

Motivations:

- Tendence to make simulations more and more **realistic**.
- **Nonlinear problems** increasingly important.
- In general not easy to solve. $\longrightarrow$ **Iterative methods** needed.

- If we use **finite elements**:

  - **large** problems

Introduction

Motivations:

- Tendence to make simulations more and more **realistic**.
- **Nonlinear problems** increasingly important.
- In general not easy to solve. $\longrightarrow$ **Iterative methods** needed.

- If we use **finite elements**:

  - **large** problems
  - **unknown** proceeding of iterates in configuration space

Introduction

Motivations:

- Tendence to make simulations more and more **realistic**.
- **Nonlinear problems** increasingly important.
- In general not easy to solve. $\longrightarrow$ **Iterative methods** needed.

- If we use **finite elements**:

  - **large** problems
  - **unknown** proceeding of iterates in configuration space $\longrightarrow$

Mesh adaptivity

welcome

Introduction

Motivations:

- Tendence to make simulations more and more **realistic**.
- **Nonlinear problems** increasingly important.
- In general not easy to solve. → **Iterative methods** needed.

- If we use **finite elements**:

  - **large** problems
  - **unknown** proceeding of iterates in configuration space
  - spatially **unpredictable** and perhaps very **small details**

→ **Mesh adaptivity** <u>welcome</u>

Introduction

Motivations:

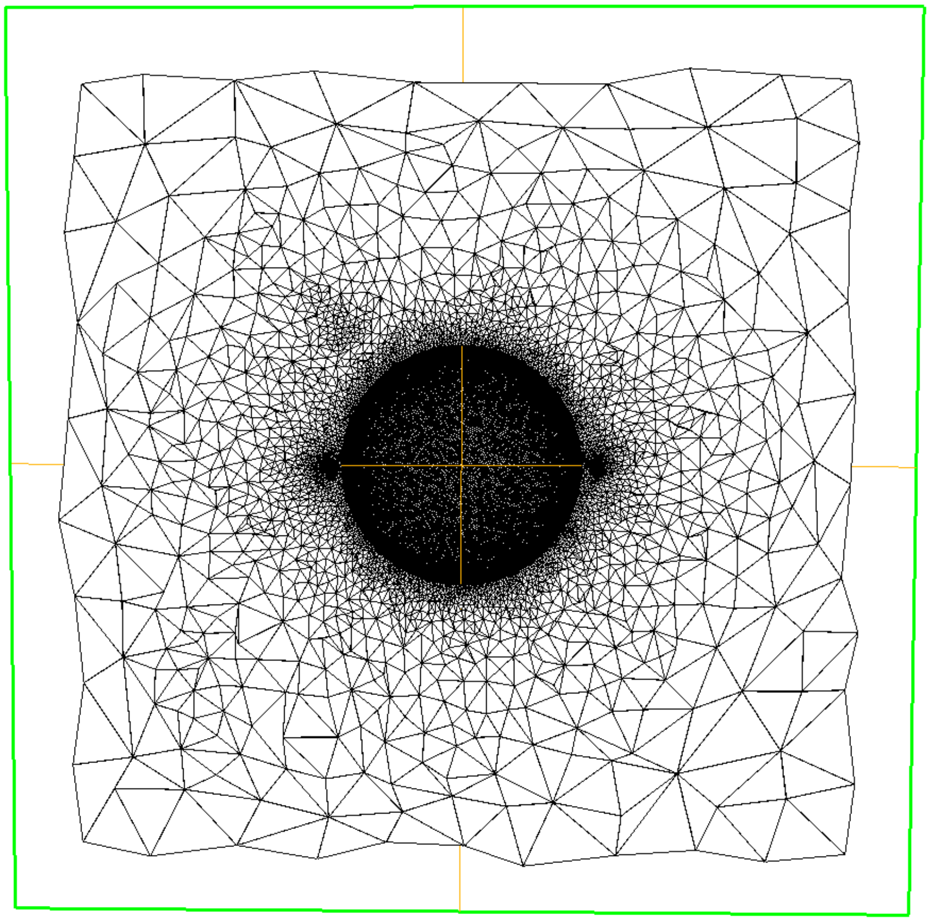
- Tendence to make simulations more and more **realistic**.
- **Nonlinear problems** increasingly important.
- In general not easy to solve. → **Iterative methods** needed.

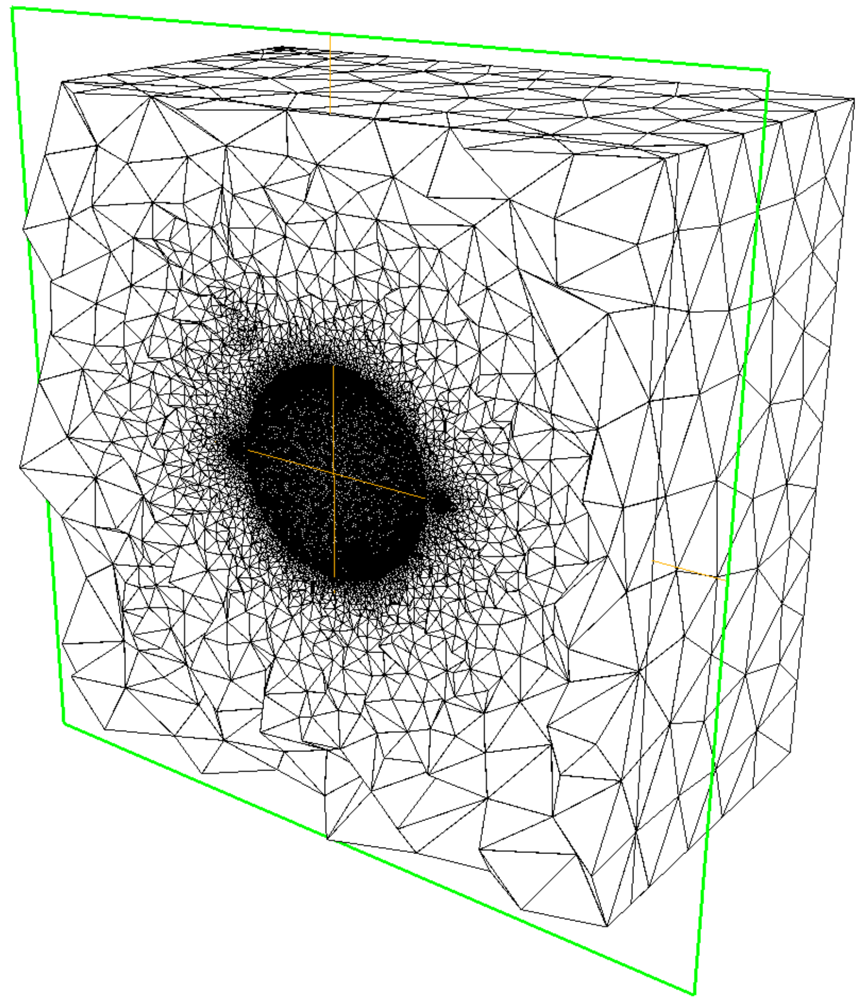
- If we use **finite elements**:

  - **large** problems
  - **unknown** proceeding of iterates in configuration space
  - spatially **unpredictable** and perhaps very **small details**

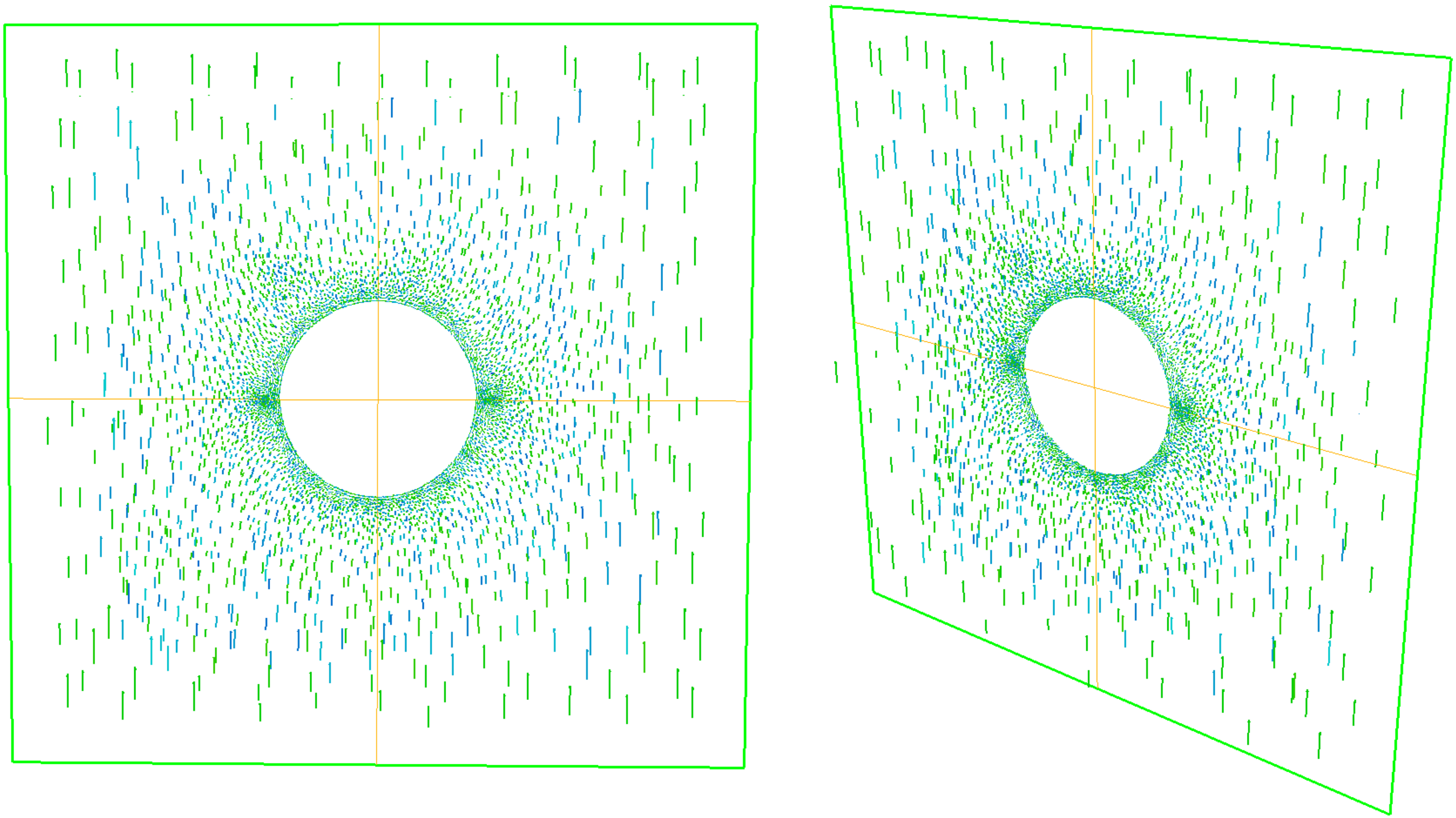
→ **Mesh adaptivity** welcome or even required

Example – confined liquid crystals

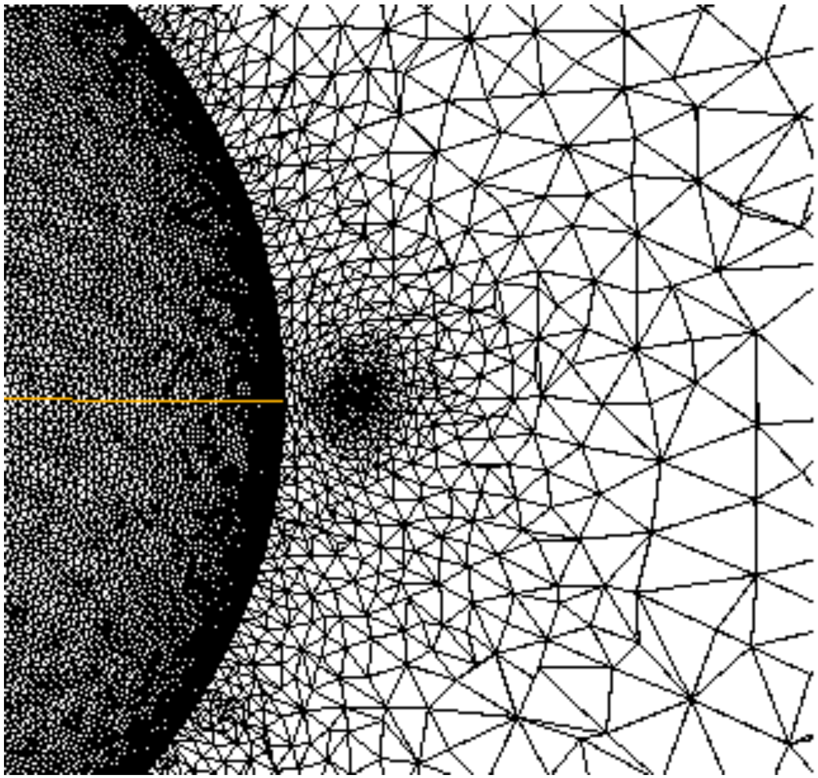


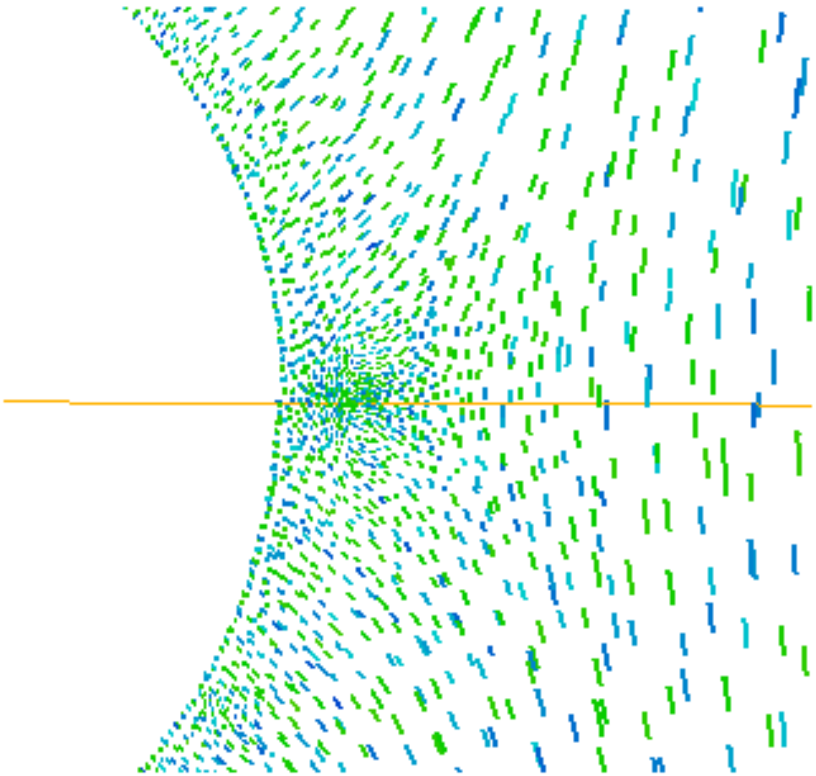


Example – confined liquid crystals



Example – confined liquid crystals





Newton iteration of tensor fields

Newton iteration of tensor fields

$F(Q)$ min: $\boxed{\delta F(Q) = F'(Q)\delta Q = 0}$ ***Euler-Lagrange equations***

Newton iteration of tensor fields

$F(Q)$ min: $\boxed{\delta F(Q) = F'(Q)\delta Q = 0}$ ***Euler-Lagrange equations***

$$\delta F(Q) = \delta \int_{\Omega} f(Q, \nabla Q) dV$$

$$= \int_{\Omega} \left(\frac{\partial f}{\partial Q} - \frac{\partial}{\partial \vec{r}} \frac{\partial f}{\partial(\nabla Q)} \right) \delta Q dV + \int_{\partial\Omega} \frac{\partial f}{\partial(\nabla Q)} \cdot \vec{\nu} \delta Q dV$$

$$= \int_{\Omega} L \nabla Q_{ij} \cdot \nabla \varphi_{ij} + \left(AQ_{ij} + BQ_{ik}Q_{kj} + CQ_{ik}Q_{kl}Q_{lj} \right) \varphi_{ij} dV - W \int_{\partial\Omega} \left(Q_{ij} - Q_{ij}{}^{0} \right) \varphi_{ij} dA$$

Newton iteration of tensor fields

$$F(Q) \text{ min:} \quad \boxed{\delta F(Q) = F'(Q)\delta Q = 0}$$

Euler-Lagrange equations

$$\delta F(Q) = \delta \int_{\Omega} f(Q, \nabla Q) dV$$

$$= \int_{\Omega} \left(\frac{\partial f}{\partial Q} - \frac{\partial}{\partial \vec{r}} \frac{\partial f}{\partial (\nabla Q)} \right) \delta Q dV + \int_{\partial\Omega} \frac{\partial f}{\partial (\nabla Q)} \cdot \vec{\nu} \delta Q dV$$

$$L \nabla \mathrm{Q}_{\mathrm{ij}} \cdot \vec{\nu} = -\mathrm{W} \left(\mathrm{Q}_{\mathrm{ij}} - \mathrm{Q}_{\mathrm{ij}}^{\ 0} \right)$$

$$= \int_{\Omega} L \nabla Q_{ij} \cdot \nabla \varphi_{ij} + \left(A Q_{ij} + B Q_{ik} Q_{kj} + C Q_{ik} Q_{kl} Q_{lj} \right) \varphi_{ij} dV - W \int_{\partial\Omega} \left(Q_{ij} - Q_{ij}^{\ 0} \right) \varphi_{ij} dA$$

Newton iteration of tensor fields

$F(Q)$ min: $\boxed{\delta F(Q) = F'(Q)\delta Q = 0}$ ***Euler-Lagrange equations***

$$\delta F(Q) = \delta \int_{\Omega} f(Q, \nabla Q) dV$$

$$= \int_{\Omega} \left(\frac{\partial f}{\partial Q} - \frac{\partial}{\partial \vec{r}} \frac{\partial f}{\partial (\nabla Q)} \right) \delta Q dV + \int_{\partial\Omega} \frac{\partial f}{\partial (\nabla Q)} \cdot \vec{\nu} \delta Q dV$$

$$L \nabla \mathrm{Q}_{\mathrm{ij}} \cdot \vec{\nu} = -\mathrm{W}\left(\mathrm{Q}_{\mathrm{ij}} - \mathrm{Q}_{\mathrm{ij}}^{0}\right)$$

$$= \int_{\Omega} L \nabla Q_{ij} \cdot \nabla \varphi_{ij} + \left(A Q_{ij} + B Q_{ik} Q_{kj} + C Q_{ik} Q_{kl} Q_{lj} \right) \varphi_{ij} dV - W \int_{\partial\Omega} \left(Q_{ij} - Q_{ij}^{0} \right) \varphi_{ij} dA$$

$$\boxed{F''(Q_k) v_k \varphi = -F'(Q_k) \varphi}$$

Newton iteration equation

(φ - *test functions*)

$$Q_{k+1} = Q_k + v_k$$

(next iteration step)

Tasks

Tasks

Two tasks:

Tasks

Two tasks:

- Coupling: **mesh adaptivity** + **nonlinear minimization**.

Tasks

Two tasks:

- Coupling: **mesh adaptivity** + **nonlinear minimization**.
- Role played by various **parameters**.

Tasks

Two tasks:

- Coupling: **mesh adaptivity** + **nonlinear minimization**.
- Role played by various **parameters**.

- Not much adressed until now [1].

[1] I. Danaila, F. Hecht, *A finite element method with mesh adaptivity for computing vortex states in fast-rotating Bose-Einstein condensates.* Jour. of Computat. Phys., **229** (2010) 6946-6960.

Tasks

Two tasks:

- Coupling: **mesh adaptivity** + **nonlinear minimization**.
- Role played by various **parameters**.

- Not much adressed until now [1].
- In particular not for **metric** mesh adaptivity.

[1] I. Danaila, F. Hecht, *A finite element method with mesh adaptivity for computing vortex states in fast-rotating Bose-Einstein condensates.* Jour. of Computat. Phys., **229** (2010) 6946-6960.

Tasks

Two tasks:

- Coupling: **mesh adaptivity** + **nonlinear minimization**.
- Role played by various **parameters**.

- Not much adressed until now [1].
- In particular not for **metric** mesh adaptivity.
- Surely **not completely clarified**.

[1] I. Danaila, F. Hecht, *A finite element method with mesh adaptivity for computing vortex states in fast-rotating Bose-Einstein condensates.* Jour. of Computat. Phys., **229** (2010) 6946-6960.

Metric mesh adaptivity in 3D

Tools:

Parameters

`mshmet:` calculates metric

`scFields, hmin, hmax, errm`

`mmg3d:` with this metric remeshes previous mesh

`metric M, hmin, hmax, errm, hgrad`

Main scheme:

Main scheme:

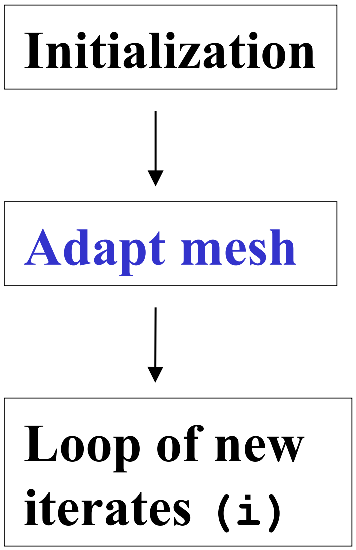
$\boxed{\textbf{Initialization}}$

Main scheme:

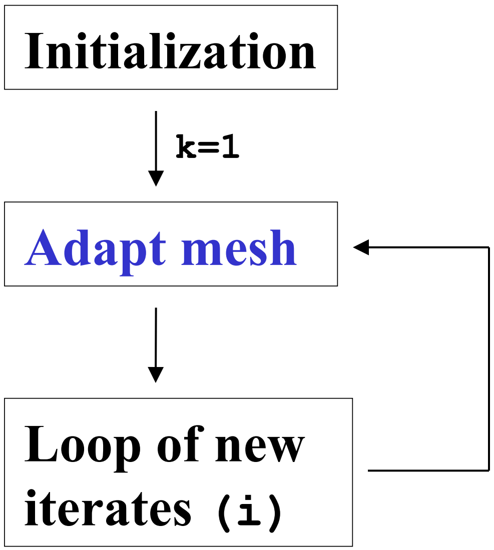
Initialization

Adapt mesh

Main scheme:

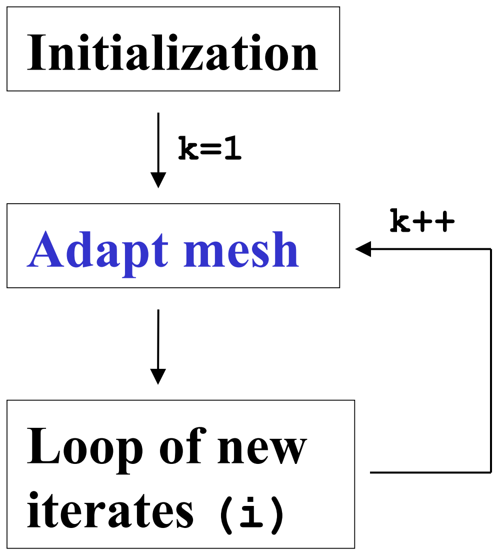


Main scheme:



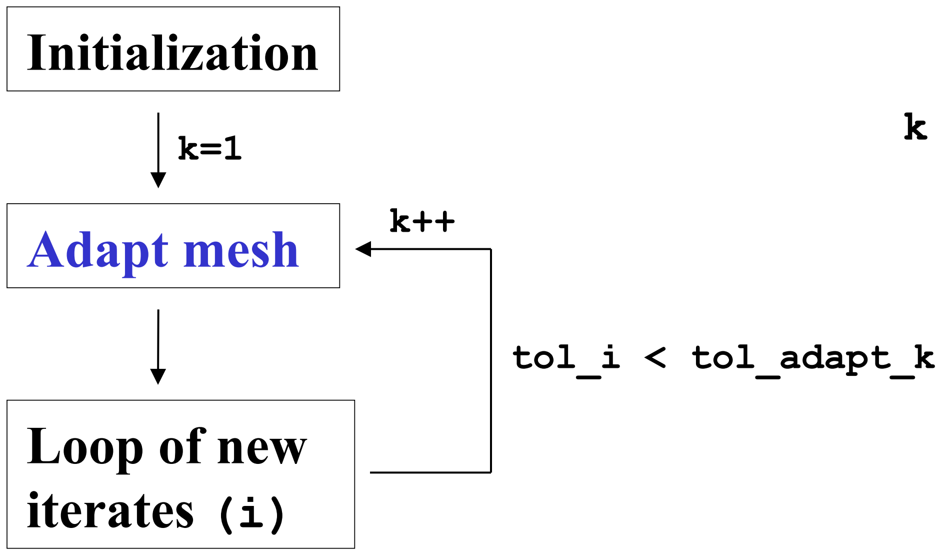
k ... mesh adaptivity index

Main scheme:



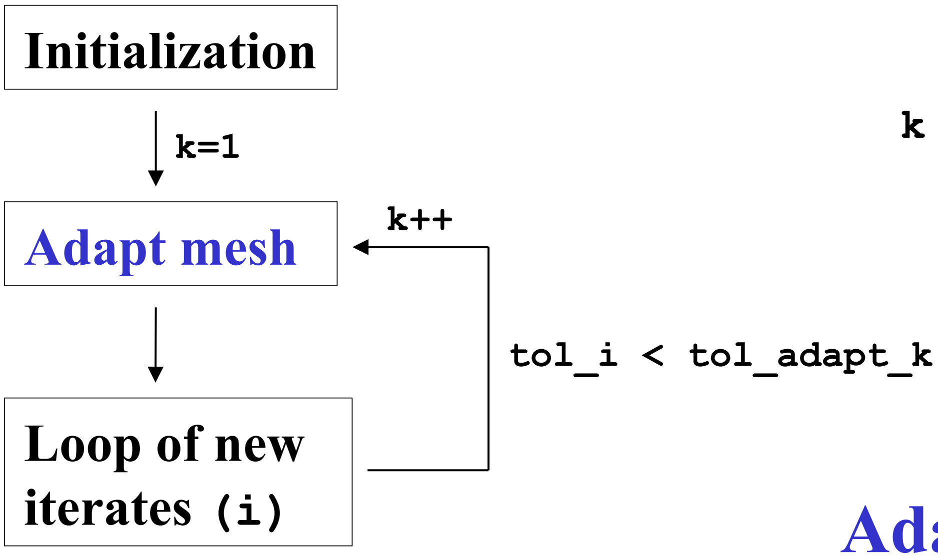
k ... mesh adaptivity index

Main scheme:



k ... mesh adaptivity index

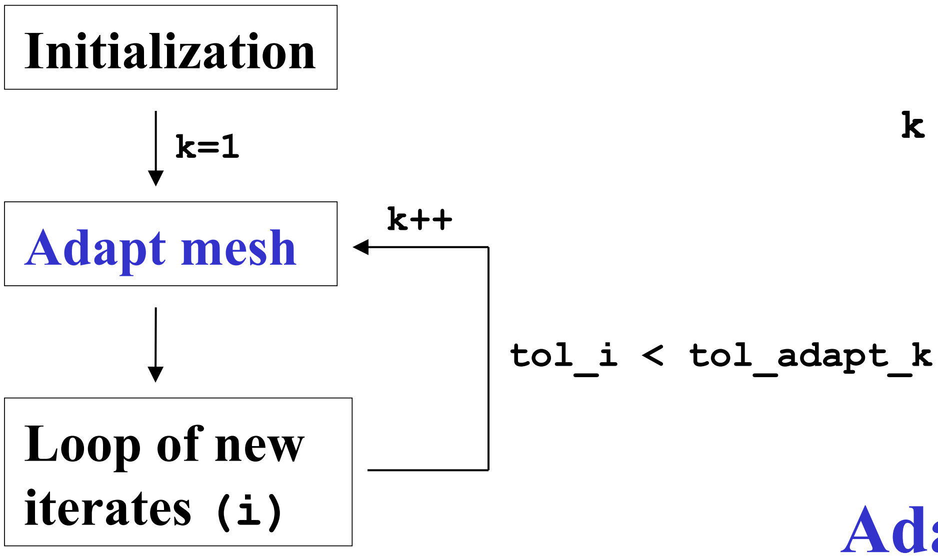
Main scheme:



k ... mesh adaptivity index

Adapt mesh:

Main scheme:

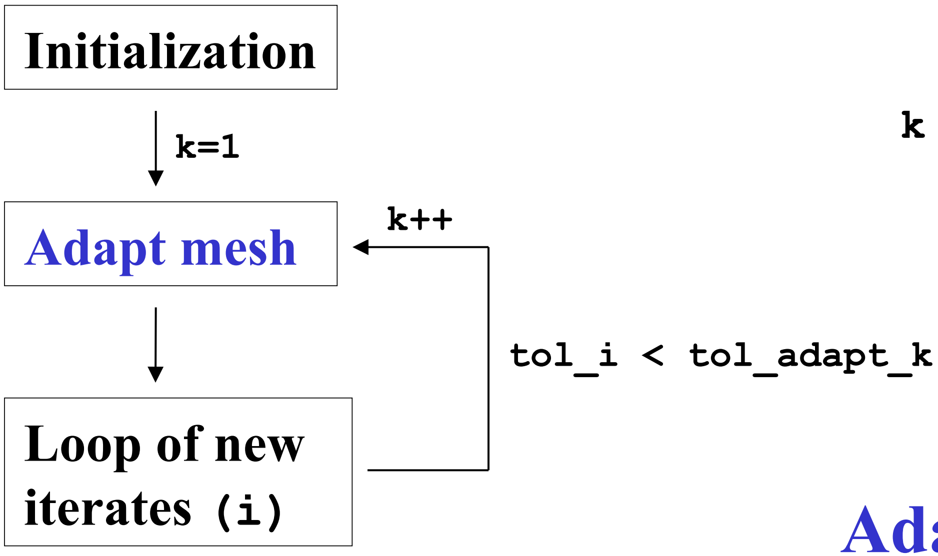


`k` ... mesh adaptivity index

Adapt mesh:

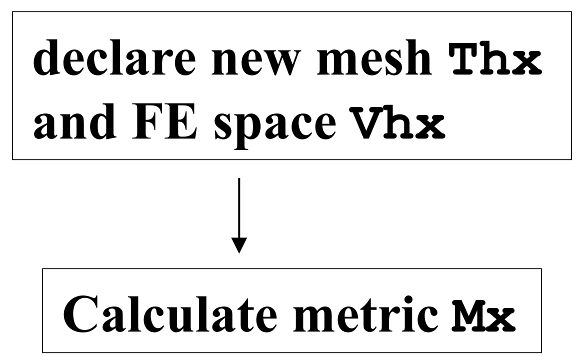
declare new mesh `Thx` and FE space `Vhx`

Main scheme:

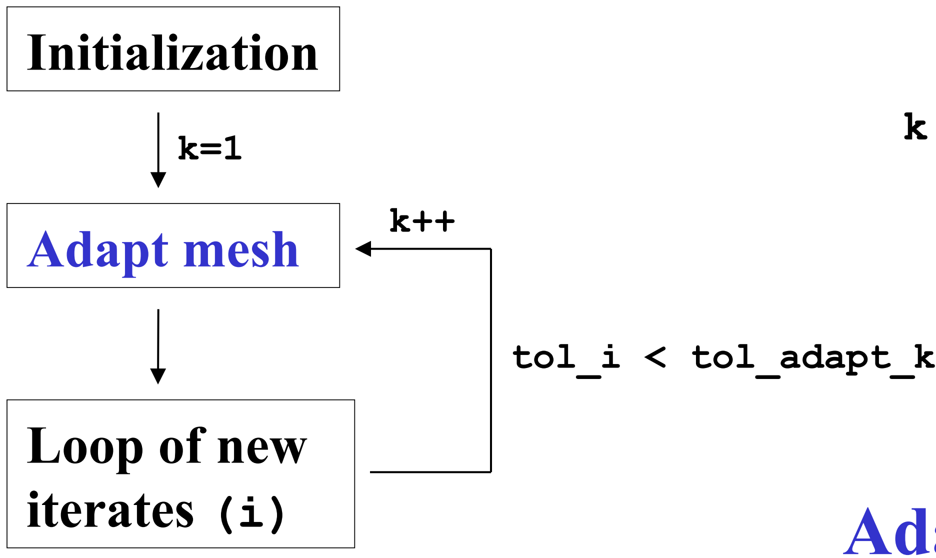


k ... mesh adaptivity index

Adapt mesh:

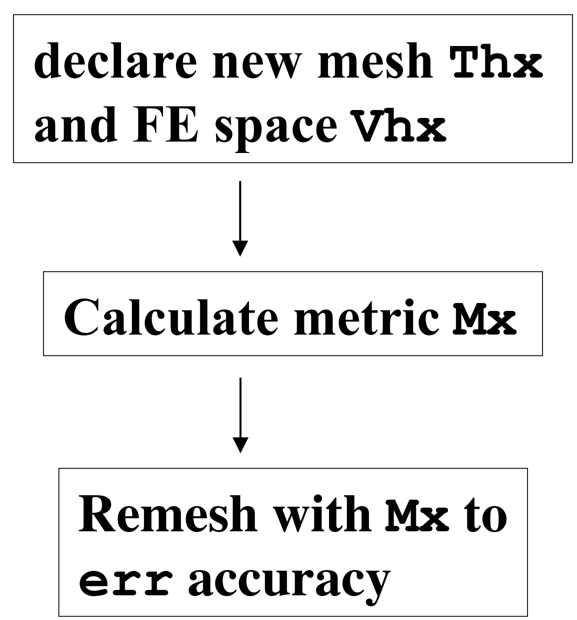


Main scheme:

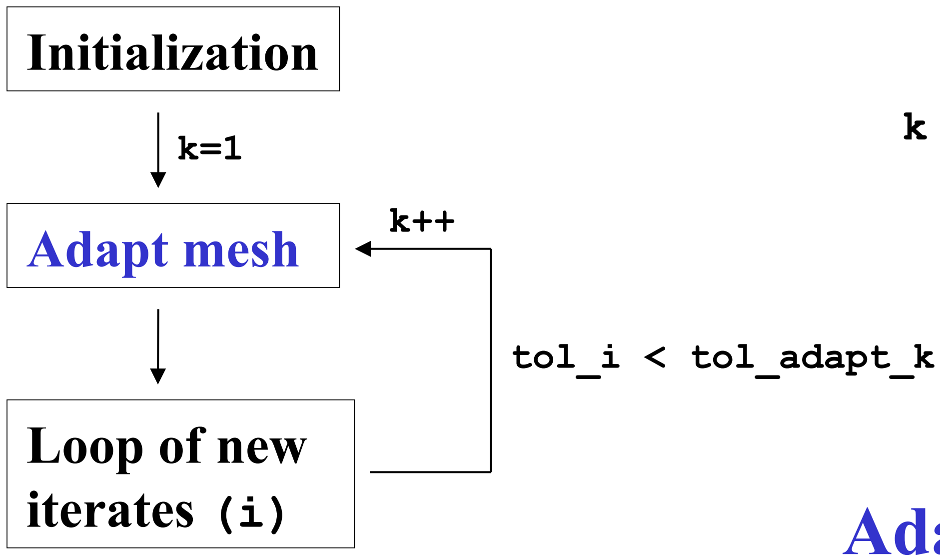


`k` ... mesh adaptivity index

Adapt mesh:

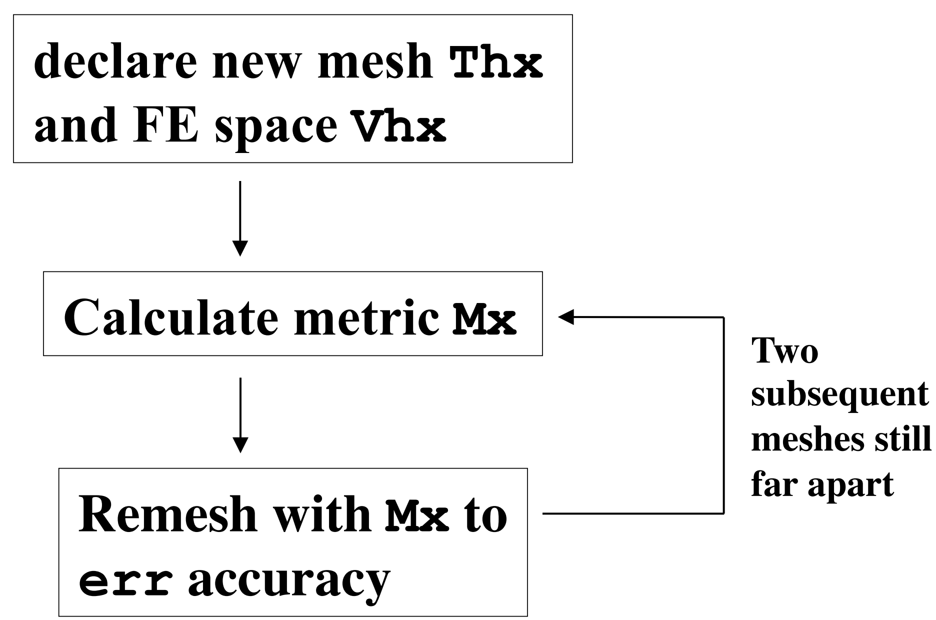


Main scheme:

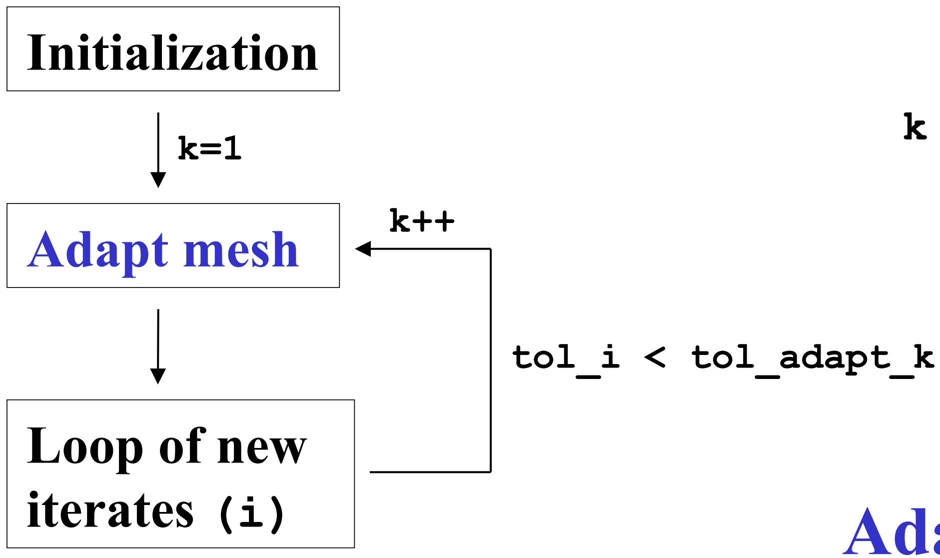


k ... mesh adaptivity index

Adapt mesh:

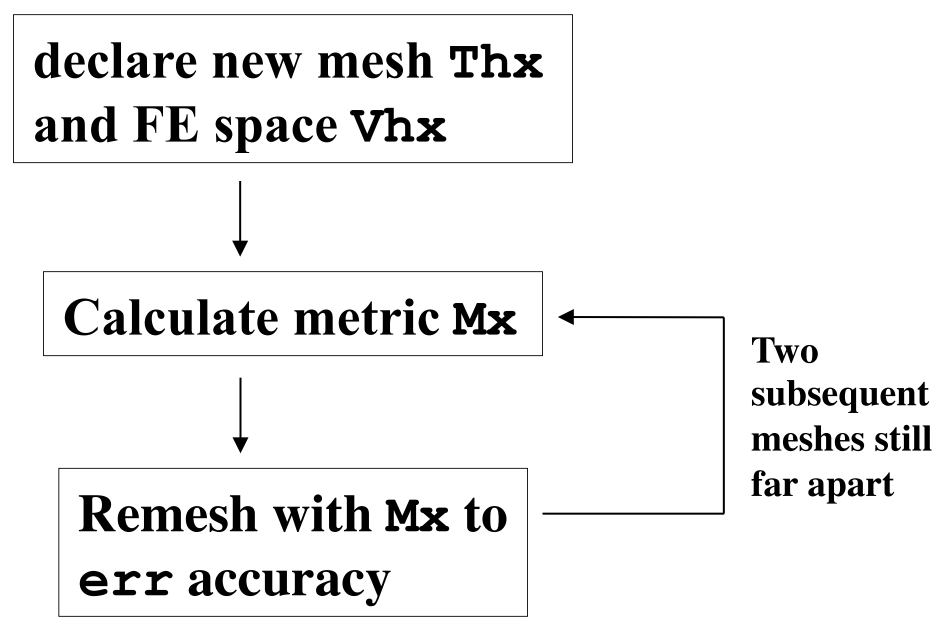


Main scheme:



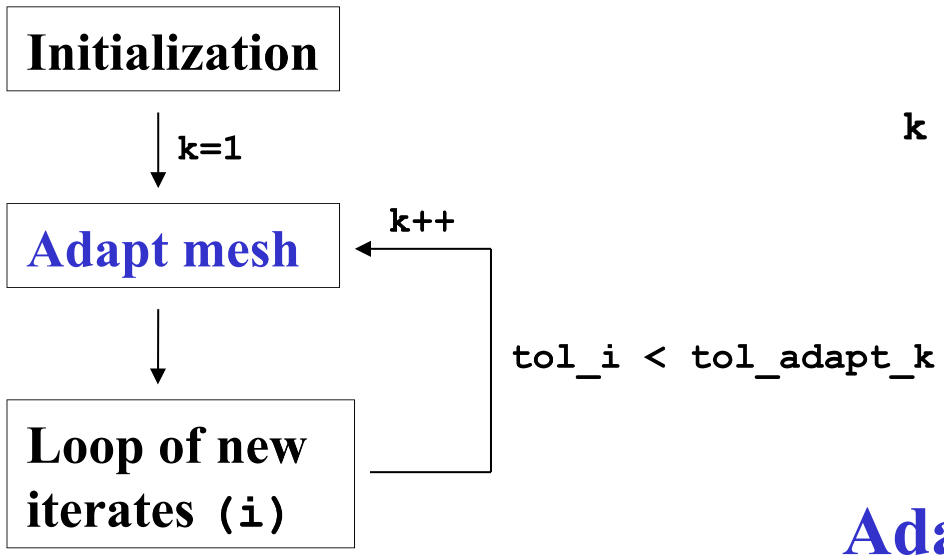
k ... mesh adaptivity index

Adapt mesh:



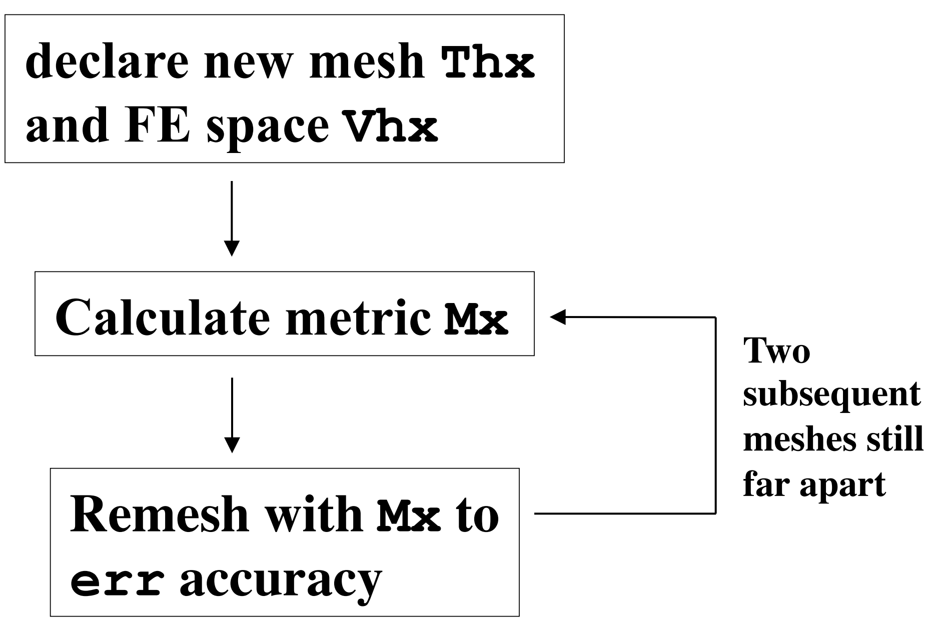
Questions:

Main scheme:



`k` ... mesh adaptivity index

Adapt mesh:



Questions:

1) **How many** mesh adaptations to do?

Main scheme:

Initialization

`k=1`

Adapt mesh

Loop of new iterates `(i)`

`k++`

`tol_i < tol_adapt_k`

`k` ... mesh adaptivity index

Adapt mesh:

declare new mesh `Thx` and FE space `Vhx`

Calculate metric `Mx`

Remesh with `Mx` to `err` accuracy

Two subsequent meshes still far apart

Questions:

1) **How many** mesh adaptations to do?
2) At what **tolerances** to trigger new mesh adaptivity?

Main scheme:

Initialization

↓ `k=1`

Adapt mesh

↓

Loop of new iterates `(i)`

`k++` — `tol_i < tol_adapt_k`

`k` ... mesh adaptivity index

Questions:

1) **How many** mesh adaptations to do?

2) At what **tolerances** to trigger new mesh adaptivity?

3) To what **accuracy** compute new meshes?

Adapt mesh:

declare new mesh `Thx` and FE space `Vhx`

↓

Calculate metric `Mx`

↓

Remesh with `Mx` to `err` accuracy

Two subsequent meshes still far apart

Main scheme:

Initialization

k=1

Adapt mesh

k++

tol_i < tol_adapt_k

Loop of new iterates (i)

k ... mesh adaptivity index

Adapt mesh:

declare new mesh `Thx` and FE space `Vhx`

Calculate metric `Mx`

Remesh with `Mx` to `err` accuracy

Two subsequent meshes still far apart

Questions:

1) **How many** mesh adaptations to do?
2) At what **tolerances** to trigger new mesh adaptivity?
3) To what **accuracy** compute new meshes?

Three (sets of) free parameters!

Temporary answers:

Temporary answers:

1) Use a **fixed** nb of mesh adaptations.

Temporary answers:

1) Use a **fixed** nb of mesh adaptations.
2) Trigger new mesh adaptivity at a predefined, **fixed sequences of tolerances**.

Temporary answers:

1) Use a **fixed** nb of mesh adaptations.
2) Trigger new mesh adaptivity at a predefined, **fixed sequences of tolerances**.
3) Compute new meshes at predefined, **fixed sequences of accuracies**.

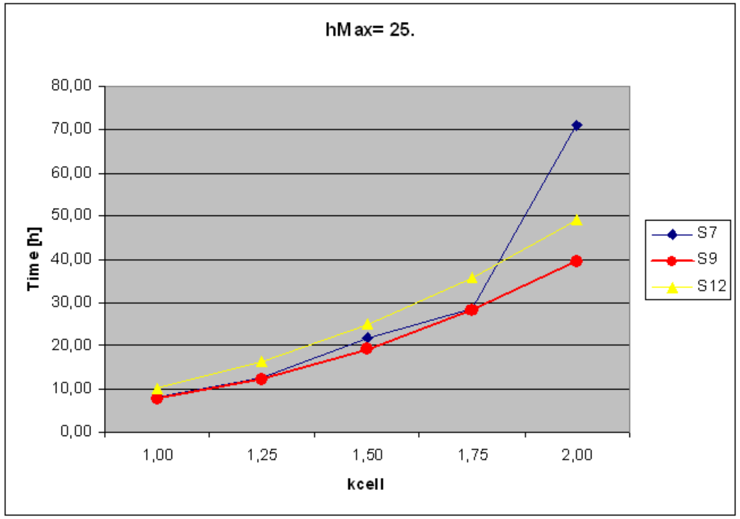
Temporary answers:

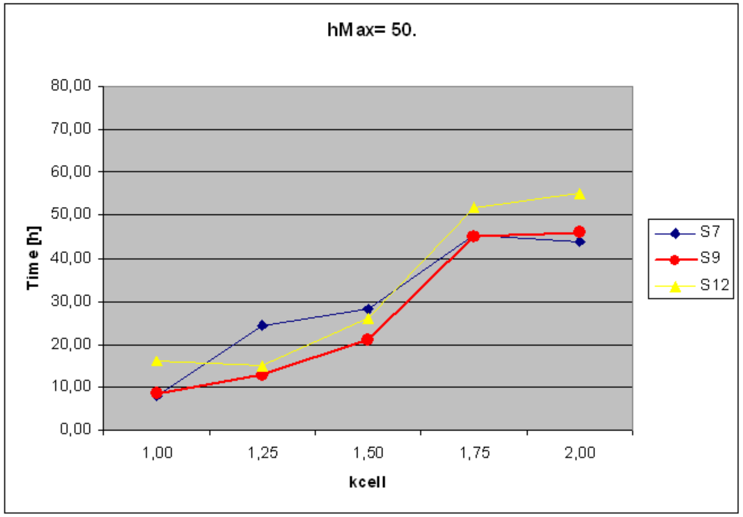
1) Use a **fixed** nb of mesh adaptations.
2) Trigger new mesh adaptivity at a predefined, **fixed sequences of tolerances**.
3) Compute new meshes at predefined, **fixed sequences of accuracies**.

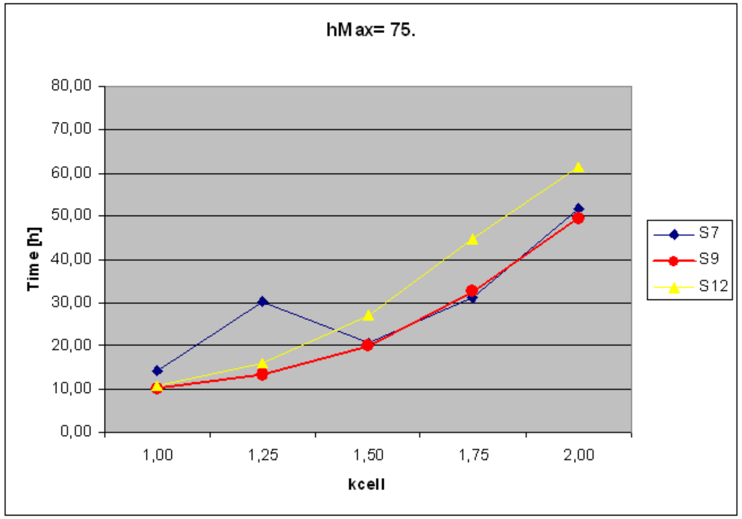
Table 1.
Three sequences (arrays) used in calculations:

Mesh adaptation	**S7** tolAdapt	errm	**S9** tolAdapt	errm	**S12** tolAdapt	errm
0.	0.5e-4	0.020	0.5e-4	0.020	0.5e-4	0.020
1.	0.5e-3	0.020	0.5e-3	0.020	0.5e-3	0.020
2.	0.5e-3	0.015	0.5e-3	0.015	0.5e-3	0.015
3.	0.5e-4	0.015	1.0e-4	0.020	1.0e-4	0.020
4.	0.5e-5	0.015	1.0e-4	0.015	1.0e-4	0.015
5.	0.5e-5	0.010	0.5e-4	0.015	0.5e-4	0.020
6.	1.0e-6	0.015	1.0e-5	0.015	0.5e-4	0.015
7.	1.0e-6	0.010	0.5e-5	0.015	1.0e-5	0.015
8.			0.5e-5	0.010	1.0e-5	0.015
9.			1.0e-6	0.010	0.5e-5	0.015
10.					0.5e-5	0.010
11.					1.0e-6	0.015
12.					1.0e-6	0.010

Results and conclusions







hmax=25

kcell/seq	S7	S9	S12
1,00	8,17	7,83	10,08
1,25	12,42	12,33	16,25
1,50	21,75	19,25	25,00
1,75	28,50	28,17	35,67
2,00	71,00	39,50	49,08

hmax=50

kcell/seq	S7	S9	S12
1,00	8,00	8,50	16,00
1,25	24,37	13,00	15,00
1,50	28,00	21,00	26,00
1,75	45,33	45,12	51,88
2,00	44,00	46,00	55,00

hmax=75

kcell/seq	S7	S9	S12
1,00	14,28	10,12	10,80
1,25	30,25	13,33	16,07
1,50	20,67	20,17	27,12
1,75	31,12	32,50	44,83
2,00	51,78	49,38	61,28

Main nonlinear minimization scheme

```
// MAIN SCHEME:
Main(Sh, f, tolAdapt, errm)
{
  Initialize(Th, Qh; Sh, f);  // Initialization of Th and Qh.
  NbOfAdapt= length(tolAdapt);    // Total nb of adaptations.

  int k= 0;                   // Adaptation index is initialized.
  Qh= Calculate_Nematic_Structure(Th, tolAdapt_k);
  while (++k < NbOfAdapt)  {
     Th= Adapt_Mesh(Th, Qh, errm_k);
     Qh= Calculate_Nematic_Structure(Th, tolAdapt_k);
  }
  return Th, Qh;
}
```

Mesh adaptivity procedure

```
// MESH ADAPTATION:
Adapt_Mesh(Th, Qh; hmax, errm_k)   // Other possible parameters:
{                                  //  hmin, hgrad (here fixed).
   mesh3 Thx= Th; // Declares and initializes new mesh variable.
   scFields= {Qh, S, DF};  // Scalar fields for metric calculus.

   for (j=1; j<=NAdaptIter; ++j)  {
      fespace Vhx(Thx, P13d);           // Declares new FE space.
      Vhx M= mshmet(Thx, scFields, hmin, hmax, errm_k);//Metric.
      Thx= mmg3d5ljll(Thx, M, hmin, hmax, hgrad);  // Remeshing.
      if (meshes close enough)  break;   // Loop-exit condition.
   }
   return Th=Thx, Qh;
}
```