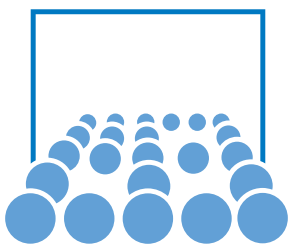
Algorithms of Scientific Computing

Hierarchical Methods and Sparse Grids

Tobias Neckel, Dirk Pflüger

Technische Universität München

Summer Term 2010



Part IX

Multigrid Methods

Multigrid Methods

Up to now we have considered

- Hierarchical bases (and algorithms) to represent functions with low costs as well as possible for
  - interpolation and quadrature,
  - the solution of a system of linear equations stemming from a PDE discretization

We haven't considered so far

- The solution of the system of linear equations
- There, hierarchical methods play a very important role
- Especially *multigrid methods* allow efficient iterative methods to solve linear system for important classes of discretized problems
- We will now consider multigrid (MG) methods
- But before, we have to look at classical iteration methods (esp. at their rates of convergence) to be able to evaluate multigrids

System of Linear Equations

- Aim: solve linear system

$$Ax = b \text{ with } A \in \mathbb{R}^{n \times n} \text{ and } x, b \in \mathbb{R}^n$$

- We assume that n is so large that a direct solution (with Gaussian elimination, e.g.) is too expensive regarding time or space
- As usual, we assume that A is sufficiently well-behaved
  - Typically desired properties: invertible, symmetric, with non-zero entries on diagonal, ...

Iterative Solvers

- Iterative solvers compute sequence of approximations

$$x^0, x^1, x^2, \ldots$$

- Converges against solution x of $Ax = b$
- In practice: stop method after finite number of steps; take iterate as approximation of x
- An iterative method will make compromise as well as possible between two requirements
  - x^{i+1} should be cheap to be computed out of $x^0, \ldots, x^i$ $\Rightarrow$ in the majority of cases: x^{i+1} is function only of x^i (and, of course, of A and b) – just consider storage space
  - Convergence should be as fast as possible: accurate results after as few steps as possible

Residual and Error

- For further considerations, the following notions are helpful:
  - The *residual* after i steps is defined as $r^i := b - Ax^i$
  - The *error* is $e^i := x^i - x$
- Residual easy to compute, but we would like to know error (would directly provide access to exact solution)
- Both magnitudes are related via equation $r^i = -Ae^i$
- This suggests to compute estimate of error out of residual:
  - Apply (cheap) approximation of $-A^{-1}$ on residual

  (Outlook for hierarchical methods: approximation can be obtained cheaply on coarser grid)

Linear Iterative Methods

Linear method to solve linear systems

- Can be written as

$$x^{i+1} = Mx^i + Nb \tag{1}$$

- M and N are $n \times n$ matrices
- M and N depend only on A, not on b or x^i
- Linear iterative methods are popular for several reasons (their analysis is relatively easy, e.g.)

Linear Iterative Methods: Consistency

Minimal requirement at iterative method:

- Exact solution has to be fixed point of iteration
- If we provide solution as x^0 it should not be destroyed
- Thus, for all b and a with $Ax = b$ it has to hold

$$x \stackrel{!}{=} Mx + Nb = (M + NA)x$$

$$\Rightarrow$$

$$M + NA = I \tag{2}$$

(as we can choose b and thus x arbitrarily)

- This requirement is called *consistency*

Linear Iterative Methods: Convergence

- With the consistency requirement $M + NA = I$ (2), we rewrite the iteration scheme $x^{i+1} = Mx^i + Nb$ (1) as

$$\begin{aligned} x^{i+1} &= (I - NA)x^i + Nb \\ &= x^i - N(Ax^i - b) \\ &= x^i + Nr^i \end{aligned}$$

- For the error, this results to

$$\begin{aligned} e^{i+1} = x^{i+1} - x &= x^i - x + Nr^i \\ &= e^i - NAe^i = Me^i \end{aligned}$$

Linear Iterative Methods: Convergence (2)

- Therefore, speed of convergence of iteration depends on M, more accurately on the norm of M:

$$\|e^{i+1}\| \leq \|M\| \cdot \|e^i\|$$

$\Rightarrow$ $\|M\|$ should be smaller than 1; the closer to 0 the better

First try

- We therefore choose $M := 0$
- The iterative method solves the linear system in the first step:

$$x^1 = x$$

- As expected, there is no free lunch, as

$$N = (I - M)A^{-1} = A^{-1}$$

- We would have to solve our linear system to compute $x^1 \ldots$

Jacobi Method

- We obtain feasible methods by the decomposition

$$A =: D - E - F$$

  where
  - D contains the diagonal of A,
  - $-E$ the (strictly) lower triangular part, and
  - $-F$ the (strictly) upper triangular part
- As an example, we look at the *Jacobi method*

A short remark:

- For sparse grids, we have mentioned the conjugated gradient method (CG)
- It is not a linear iterative method, but behaves rather similar

Jacobi Method (2)

- The Jacobi method chooses

$$N := D^{-1},$$

thus

$$M = I - D^{-1}A$$

- This results in the following algorithm:

  Compute x^{i+1} out of x^i:

  Compute residual $r^i := b - Ax^i$

  for $k = 1, \ldots, n$:

  $$x_k^{i+1} = x_k^i + \frac{1}{a_{k,k}} r_k^i$$

  endfor

- Computing r^i is, of course, a loop over all components, too
- But the matrix could be as well provided as a procedure allowing to compute matrix-vector products $x \mapsto Ax$
- Additionally, we only need knowledge about diagonal entries $a_{k,k}$

Jacobi Method (3)

- Unfortunately, we do not have convergence for arbitrary A
- In practice, one often introduces a damping factor $0 < \alpha \leq 1$ for the modification $D^{-1} r^i$ to obtain convergence
- If α is too small this goes at the expense of speed
- In the algorithm this looks like

$$\begin{aligned} &\dots \\ &x_k^{i+1} = x_k^i + \alpha \frac{1}{a_{k,k}} r_k^i \\ &\dots \end{aligned}$$

- For the matrices we obtain

$$N(\alpha) = \alpha D^{-1}$$

and

$$M(\alpha) = I - \alpha D^{-1} A$$

Speed of Convergence

How many iterations do we need to perform to obtain sufficiently small convergence error?

- We can obtain propositions about speed of convergence from equation

$$e^{i+1} = Me^i$$

- To simplify things we assume
  - M has a full set of eigen vectors $\eta_1, \ldots, \eta_n$ for real-valued eigen values $\lambda_1, \ldots, \lambda_n$
  - In reality, this does not always hold; but the concepts from the simplified case remain mainly the same

Speed of Convergence (2)

- We then can write e^0 as linear combination of eigen vectors

$$e^0 := \sum_{k=1}^{n} \beta_k \eta_k.$$

- Iterating (applying M) multiplies each component with corresponding λ_i

$$e^i = \sum_{k=1}^{n} \lambda_k^i \beta_k \eta_k.$$

- Typically, some of the eigen values are close to 0
  - $\Rightarrow$ The corresponding components decay very fast: at first, convergence makes a lot of progress

Speed of Convergence (3)

- Unfortunately, there are eigen values with absolute value just slightly below 1
  - The corresponding components of the error are hardly reduced
  - They dominate after few iterations the whole progress which becomes very slow after the first progress
- This effect is widely independent of x^0:
  - Error contains almost always components of all eigen vectors
  - If not: introduced at the latest by rounding errors

Speed of Convergence (4)

- Be λ_n the eigen value with absolute value closest to 1, thus

$$\delta := 1 - |\lambda_n|$$

  (unfortunately) close to 0
- Then it is for sufficiently many iterations

$$\frac{\|e^{i+1}\|}{\|e^i\|} \approx 1 - \delta.$$

- Number of iterations to obtain

$$\|e^{i+n_{it}}\| < \epsilon \|e^i\|$$

  for given $0 < \epsilon < 1$:

$$n_{it} \approx \frac{\ln \epsilon}{\ln(1-\delta)} \approx \frac{-\ln \epsilon}{\delta}$$

  (Expansion at $\delta = 0$ leads to $\ln(1-\delta) \doteq -\delta$)

Speed of Convergence (5)

- Unfortunately, we obtain for the discretization of PDEs on a grid with mesh-width h typically $\delta \sim h^{\gamma}$ for some $\gamma > 0$

$\Rightarrow$ The number of steps grows with $h^{-\gamma}$

- Even if the cost per step is proportional to number of unknowns, the overall effort grows disproportionally high

Speed of Convergence, Example

Example: damped Jacobi

- We solve linear system for discretized one-dimensional Poisson equation
  - Poisson equation $-u'' = f$
  - Stencil $\frac{1}{h^2}[-1\ 2\ -1]$

  on grid with $m-1$ inner grid points (zero on boundary)
- We solve linear system with damped Jacobi

$$x^{i+1} := x^i + \alpha D^{-1} r^i \quad \text{and} \quad 0 < \alpha \leq 1$$

Speed of Convergence, Example (2)

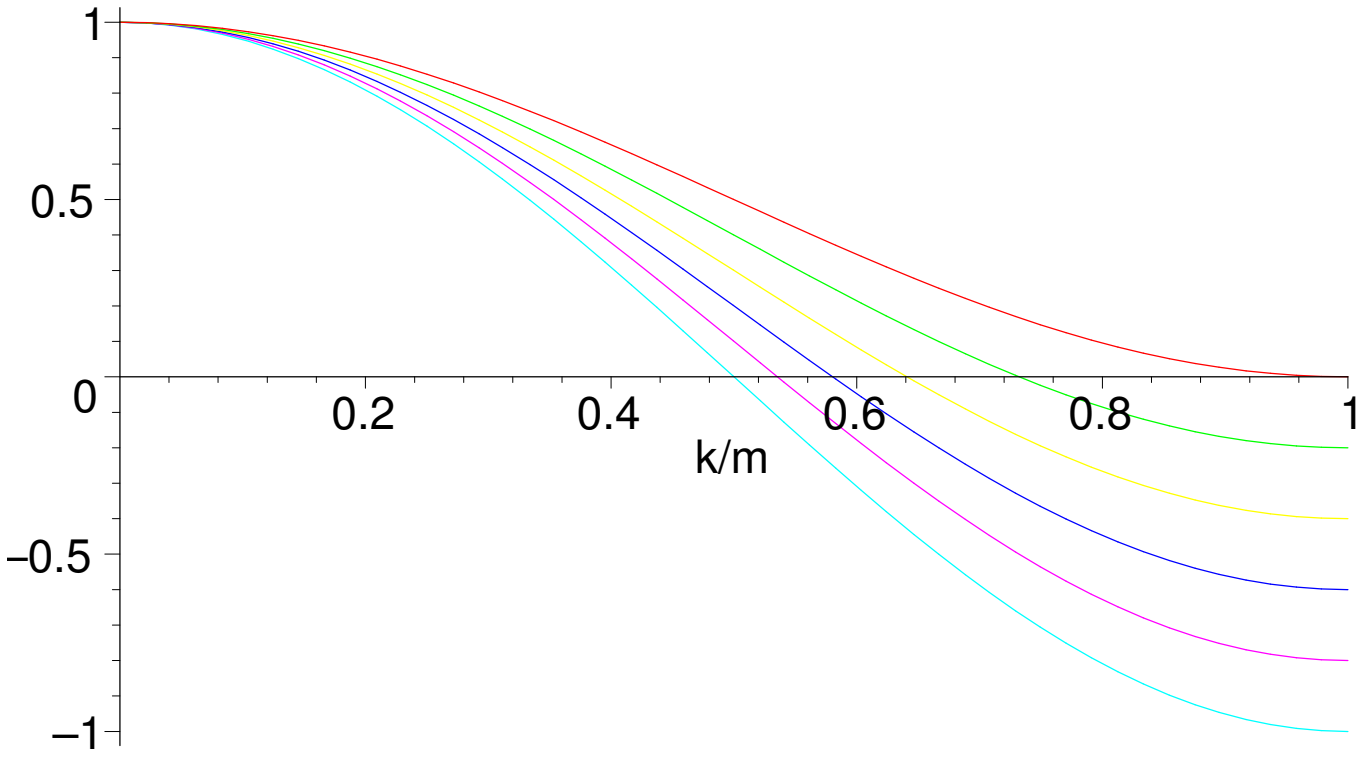
- The iteration matrix $M^{Jac} := I - \alpha D^{-1} A$ has as eigen vectors the discrete sine oscillations

$$\eta^k := \left(\sin\left(\frac{ik\pi}{m} \right) \right)_{1 \leq i < m} \in \mathbb{R}^{m-1}$$

- The corresponding eigen values are

$$\lambda_k := \alpha \cos\left(\frac{k\pi}{m} \right) + 1 - \alpha = 1 - 2\alpha \sin^2\left(\frac{k\pi}{2m} \right)$$

Speed of Convergence, Example (3)



Eigen values of M^{jac} for different values of $\alpha \in [0.5, 1]$

Speed of Convergence, Example (4)

- In the diagram, we can observe eigen values for low-frequency error components (k/m small)
- They are practically independent of α and very close to 1
- Closest to one is

$$\lambda_1 := 1 - 2\alpha \sin^2\left(\frac{\pi}{2m}\right) \doteq 1 - \frac{\alpha\pi^2}{2m^2}.$$

- At the right end of the diagram: eigen values to high-frequency error components ($k/m \approx 1$)
- We can adjust damping via α
  - $\alpha > 0.5$ results in oscillations (negative eigen values)
  - Convergence deteriorates with $\alpha \to 1$
- For all $0 < \alpha \leq 1$, the convergence rate is determined by λ_1

$\Rightarrow$ $\delta \in \mathcal{O}(m^{-2})$: half the mesh-width results in four times as many iterations for given error reduction

Why is Jacobi so Slow?

Summarizing the main observations

- For α suitably chosen, we can damp high-frequency error components ($k/m \to 1$) very well
- For all values of α, the low-frequency error components remain almost undamped

- We could have assumed that as low-frequency error terms produce only very small residuals
- Thus, residuum is not a very well-suited to construct estimate of error
- Remark: Related methods, such as Gauss-Seidel, therefore behave similarly)

Why is Jacobi so Slow? (2)

- Further search for reasons of this problem lead to observation
  - Different error components (frequencies) have completely different relations between error e^i and residual $r^i = -Ae^i$
- This can be expressed as
  - A has a large condition number

$$\kappa(A) = \frac{\max\limits_{\|x\|=1} \|Ax\|}{\min\limits_{\|x\|=1} \|Ax\|}$$

(not so large that we get problems with accuracy of solution, but large enough to make iterative solution annoyingly slow)

Why is Jacobi so Slow? (3)

- Where does large condition number come from?
  - Not a property of our problem (solve a DE), but of discretization!
- For same problem, discretizations can be provided leading to arbitrary well-conditioned coefficient matrices
- Taking hierarchical basis (for a suitable scaling) leads to matrices with significantly better condition

- In the following, we proceed otherwise
- We kind of try to remedy the gaucheness during discretization with as little computational effort as possible
- The original problem with its properties will play an important role:
  - (Geometric) multigrid methods as treated here cannot be considered independent of problem

Multiple Grids

- A_h: coefficient matrix of example problem for mesh width $h = 1/m$
- Condition number of A_h is in $\mathcal{O}(h^{-2}) \Rightarrow$ gets worse with $h \to 0$
- But need small h to keep discretization error low

$\Rightarrow$ Take solutions of coarser grids for solution on current grid

- We will have to deal with family of linear systems on interval $[0, 1]$:

$$A_h x_h = b_h$$

for mesh-width $h = 2^{-l}$ as index for discretization level $l = l_{\min}, \ldots, l_{\max}$

Multiple Grids (2)

- Solutions on different grids represented by vectors of different length
- To be able to compare them, we decompose h-grid into
  - *coarse grid points*, which also exist on $2h$-grid
  - *fine grid points*, which don't
- The *prolongation* operator

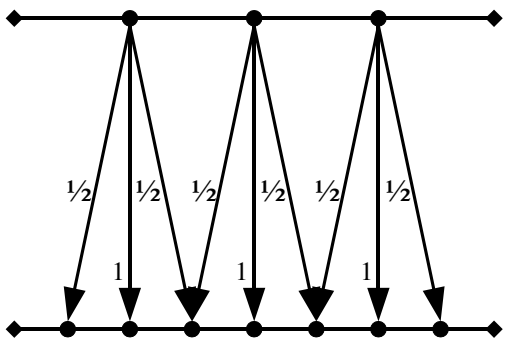
$$I_{2h}^{h} : \mathbb{R}^{1/(2h)-1} \rightarrow \mathbb{R}^{1/h-1},$$

  maps a u_{2h} on a $2h$-grid to a u_h on the h-grid
- The u_h
  - takes the values of u_{2h} at the coarse grid points and
  - interpolates the values at fine grid points linear out of coarse grid points (or boundary values, resp.)

Prolongation Operator

- As matrix and picture it looks as follows

$$I_{2h}^{h} = \frac{1}{2} \begin{pmatrix} 1 & & & \\ 2 & & & \\ 1 & 1 & & \\ & 2 & \ddots & \\ & 1 & \ddots & 1 \\ & & \ddots & 2 \\ & & & 1 \end{pmatrix}$$



Multiple Grids (cont.)

- Compare x_h with $I_{2h}^h x_{2h}$:
  - The difference will be small and mainly high-frequency
  - The low-frequency parts of the exact solution can be represented well on $2h$-grid
- Therefore straightforward to use $I_{2h}^h x_{2h}$ as initial solution for iteration on h-grid, so that contributions of resistant error modes small:

  Solve for $h = 2^{-l_{min}}$ linear system $A_h x_h = b_h$.
  for $l = l_{min} + 1, \ldots, l_{max}$:
  Iterate with $h = 2^{-l}$ and $x_h^0 := I_{2h}^h x_{2h}$
  the linear system $A_h x_h = b_h$ sufficiently often,
  call the result (afflicted with remaining error) x_h
  endfor

Multigrids

- Idea goes in right direction, but not far enough for most problems:
  - $I_{2h}^{h} x_{2h}$ contains – even if $A_{2h} x_{2h} = b_{2h}$ solved exactly – low-frequency error components, too

  $\Rightarrow$ Still many iterations on fine grid necessary if very accurate solution desired
- Thus, modify scheme so that coarse grids can be used multiple times to compute suitable correction
- Main idea:
  - After some iterations on fine grid, solve auxiliary equation which combines error and residual,

$$r^i = -Ae^i \,,$$

approximately for e^i on coarse grid

Multigrids (2)

- We need another operator, the *restriction*

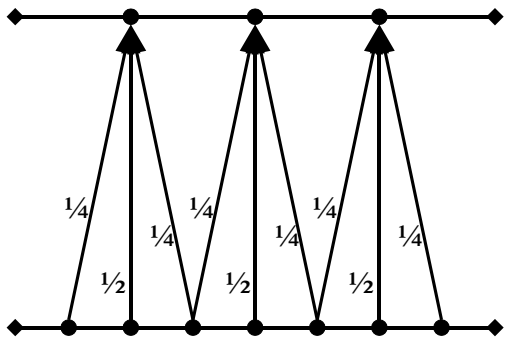
$$R_h^{2h} : \mathbb{R}^{1/h-1} \rightarrow \mathbb{R}^{1/(2h)-1},$$

- Maps right-hand side of linear system on h-grid onto right-hand side on $2h$-grid
  (Haven't needed that so far, as we assumed that b_h is known for all h))

Multigrids (3)

- Restriction can be obtained by simply omitting every second grid point
- Other choice: *weighted restriction*

$$R_h^{2h} := \frac{1}{4} \begin{pmatrix} 1 & 2 & 1 & & & & \\ & & 1 & 2 & 1 & & \\ & & & \ddots & \ddots & \ddots & \\ & & & & 1 & 2 & 1 \end{pmatrix}$$



- Remark: The property $R_h^{2h} = c \cdot \left(I_{2h}^h\right)^T$ (provided by weighted restriction) is sometimes helpful, especially concerning the analysis of such schemes

Multigrids (4)

- We now have assembled everything for *coarse grid correction*:
  - Compute on fine grid residual $r_h = b_h - A_h x_h$ for current iterated x_h
  - Transport to coarse grid: $r_{2h} = R_h^{2h} r_h$
  - Solve (approximately) $A_{2h}(-e_{2h}) = r_{2h}$
  - Transport correction to fine grid and apply to current iterate: $x_h^{new} := x_h + I_{2h}^h(-e_{2h})$
- Additionally, we have to treat the high frequencies
  - For example, by some steps of a damped Jacobi method which we call *smoother*
  - Doesn't necessarily reduce error, but smooths them as high frequencies are eliminated

Multigrids (5)

- In the example problem, the eigen values of the iteration matrix suggest to damp as high-frequency error components oscillate so much that they are practically not reduced
- With, e.g., $\alpha = 1/2$ we can overcome this
- On $2h$-grid we typically don't solve linear system exactly
- Instead: apply idea of coarse grid correction recursively
- Recursion stops if h is so large that obtaining exact solution is cheap or – for $h = 1/2$, e.g. – trivial
- We thus assume that m is power of two
- As we compute $-e_{2h}$ iteratively, we need initial value; zero vector is best choice due to several reasons...

Multigrid Algorithm

- We have two yet unknown parameters in our algorithm:
- The numbers $\nu_1, \nu_2, \mu \in \mathbb{N}$ denote the number of smoothing steps before and after the coarse grid correction, and the number of recursive calls within coarse grid correction:

$mg(x_h, b_h, \nu_1, \nu_2, \mu)$:

if $h = 2^{-l_{min}}$

solve $A_h x_h = b_h$ exactly

else

Pre-smoothing:

Apply ν_1 smoothing steps to $A_h x_h = b_h$

Coarse grid correction:

for $k = 1, \ldots, \mu$:

$x_h := x_h + I_{2h}^h(mg(0_{2h}, R_h^{2h}(b_h - A_h x_h), \nu_1, \nu_2, \mu))$

Post-smoothing:

Apply ν_2 smoothing steps to $A_h x_h = b_h$

end if

Effort/Cost

How does it look like for the effort?

- Consider the application of I_{2h}^{h} and R_{h}^{2h} to be about as expensive as one smoothing steps
- This is realistic: we have to evaluate a local stencil for the fine grid

$\Rightarrow$ The effort with N grid points is about

$$C \cdot M \cdot (\nu_1 + \nu_2 + 2\mu)$$

plus effort for μ coarse grid corrections

- C is cost per grid point (small, constant number of operations)

Effort/Cost (2)

Cost for coarse grid points

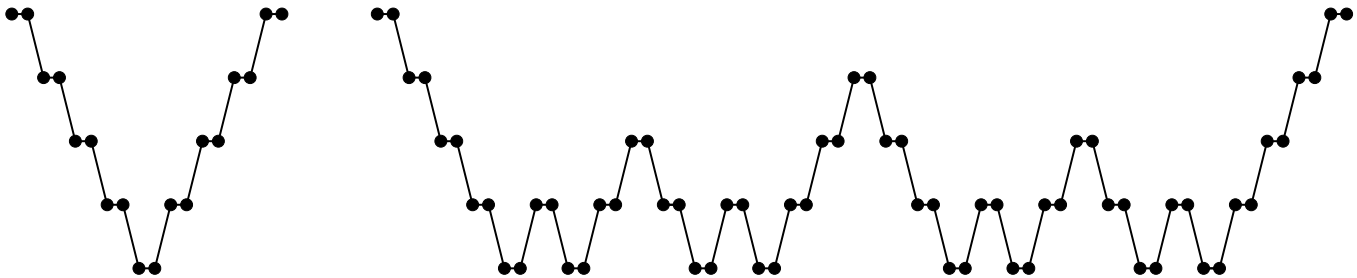
- One-dimensional model problem (as considered):
  - Smoothness and grid transfer costs for $2h$-grid are half as high as for h-grid
  - For $\mu = 1$ the total cost is about

$$1 + 1/2 + 1/4 + \ldots = 2$$

    times the cost on fine grid, i.e., proportional to number of grid points (on finest grid)
  - For $\mu = 2$ the cost per grid point grows logarithmically in M
- Problem transferred to two or three spatial dimensions:
  - Calculation looks even much better:
  - On all coarse grids together there are just $1/3$ ($2d$), or $1/7$ ($3d$) respectively, as many grid points as on finest grid
  - There, even for $\mu < 4$ and $\mu = 8$ we obtain overall effort proportional to number of unknowns on fine grid

Effort/Cost (3)

- Typical choices: $\mu = 1$ (V-cycle), and $\mu = 2$ (W-cycle)



Effort/Cost (4)

- We do not recompute, but just notice (and verify experimentally?) the other part of efficiency considerations:
  - Convergence rate (for $\nu_1 = \nu_1 = \mu = 1$, e.g.) is bounded by 1 independent of fine grid mesh-width h
- Typical multigrid convergence rates for well-behaved problems are about $1/2$ and smaller

$\Rightarrow$ We need only few steps to obtain small error

Hierarchical Methods...

Summary

Hierarchical are beneficial in many settings

- Can allow to reduce cost
- Can allow to "compress" functions (represent with few degrees of freedom)
- Can allow to estimate errors
- Can provide "level of detail" (coarse partial solutions)
- Can allow to compute coarse approximations
- Can be used to define refinement criteria
- Can speed up the solution of linear systems
- ...