Numerical Linear algebra

Multigrid methods

Martin van Gijzen and Gerard Sleijpen

December 14, 2016

Iterative methods (review)

The central idea in most iterative methods for solving $\mathbf{Ax} = \mathbf{b}$ is **error correction** using the **residual equation**: $\mathbf{Au} = \mathbf{r}_k$, $\mathbf{u}$ is the error in $\mathbf{x}_k$:

Select $\mathbf{x}_0$. Compute $\mathbf{r}_0 = \mathbf{b} - \mathbf{A}\mathbf{x}_0$

for $k = 0, 1, \ldots, k_{max}$ do %% Outer loop

  stop if $\|\mathbf{r}_k\| \leq$ tol

  $\mathbf{u}_k = \text{Solve}(\mathbf{A}, \mathbf{r}_k)$ %% Inner loop

  $\mathbf{x}_{k+1} = \mathbf{x}_k + \mathbf{u}_k$

  $\mathbf{c}_k = \mathbf{A}\mathbf{u}_k, \quad \mathbf{r}_{k+1} = \mathbf{r}_k - \mathbf{c}_k$

Here 'Solve' is some method that approximately solves the residual equation for $\mathbf{u}$ with $\mathbf{u}_k$ as approximate solution obtained by the method.

Iterative methods (review)

The central idea in most iterative methods for solving $\mathbf{Ax} = \mathbf{b}$ is **error correction** using the **residual equation**: $\mathbf{Au} = \mathbf{r}_k$, $\mathbf{u}$ is the error in $\mathbf{x}_k$:

Select $\mathbf{x}_0$. Compute $\mathbf{r}_0 = \mathbf{b} - \mathbf{A}\mathbf{x}_0$

for $k = 0, 1, \ldots, k_{max}$ do %% Outer loop

  stop if $\|\mathbf{r}_k\| \leq$ tol

  $\mathbf{u}_k = \text{Solve}(\mathbf{A}, \mathbf{r}_k)$ %% Inner loop

  $\mathbf{x}_{k+1} = \mathbf{x}_k + \mathbf{u}_k$

  $\mathbf{c}_k = \mathbf{A}\mathbf{u}_k, \quad \mathbf{r}_{k+1} = \mathbf{r}_k - \mathbf{c}_k$

Examples.

- Iterative refinement: with $\mathbf{A} = \mathbf{LU} + \Delta, \quad \mathbf{LU}\mathbf{u}_k = \mathbf{r}_k$
- Gauss–Seidel: $(\mathbf{D}_A + \mathbf{L}_A)\mathbf{u}_k = \mathbf{r}_k$
- SSOR, ...

Iterative methods (review)

The central idea in most iterative methods for solving $\mathbf{Ax} = \mathbf{b}$ is **error correction** using the **residual equation**: $\mathbf{Au} = \mathbf{r}_k$, $\mathbf{u}$ is the error in $\mathbf{x}_k$:

Select $\mathbf{x}_0$. Compute $\mathbf{r}_0 = \mathbf{b} - \mathbf{A}\mathbf{x}_0$

for $k = 0, 1, \ldots, k_{max}$ do %% Outer loop

  stop if $\|\mathbf{r}_k\| \leq$ tol

  $\mathbf{u}_k = \text{Solve}(\mathbf{A}, \mathbf{r}_k)$ %% Inner loop

  $\mathbf{x}_{k+1} = \mathbf{x}_k + \mathbf{u}_k$

  $\mathbf{c}_k = \mathbf{A}\mathbf{u}_k, \quad \mathbf{r}_{k+1} = \mathbf{r}_k - \mathbf{c}_k$

- Inner loop: use ℓ steps of, say Gauss–Seidel or GMRES, to "Solve" $\mathbf{Au} = \mathbf{r}_k$.
- Inner loop: use, say GMRES, to "Solve" $\mathbf{Au} = \mathbf{r}_k$ to some fixed relative accuracy, as $\|\mathbf{r}_k - \mathbf{Au}_k\| \leq 0.1\|\mathbf{r}_k\|$.

Iterative methods (review)

The central idea in most iterative methods for solving $\mathbf{Ax} = \mathbf{b}$ is **error correction** using the **residual equation**: $\mathbf{Au} = \mathbf{r}_k$, $\mathbf{u}$ is the error in $\mathbf{x}_k$:

Select $\mathbf{x}_0$. Compute $\mathbf{r}_0 = \mathbf{b} - \mathbf{A}\mathbf{x}_0$

for $k = 0, 1, \ldots, k_{max}$ do %% Outer loop

  stop if $\|\mathbf{r}_k\| \leq$ tol

  $\mathbf{u}_k = \text{Solve}(\mathbf{A}, \mathbf{r}_k)$ %% Inner loop

  $\mathbf{x}_{k+1} = \mathbf{x}_k + \mathbf{u}_k \, \alpha_k$

  $\mathbf{c}_k = \mathbf{A}\mathbf{u}_k, \quad \mathbf{r}_{k+1} = \mathbf{r}_k - \mathbf{c}_k \, \alpha_k$

Accelerations. Find 'relaxation' scalars α_k for better updates.

- Richardson: $\alpha_k = \alpha$
- Local minimal residuals: select α_k to minimise $\|\mathbf{r}_{k+1}\|_2$.

Iterative methods (review)

The central idea in most iterative methods for solving $\mathbf{Ax} = \mathbf{b}$ is **error correction** using the **residual equation**: $\mathbf{Au} = \mathbf{r}_k$, $\mathbf{u}$ is the error in $\mathbf{x}_k$:

Select $\mathbf{x}_0$. Compute $\mathbf{r}_0 = \mathbf{b} - \mathbf{A}\mathbf{x}_0$

for $k = 0, 1, \ldots, k_{max}$ do %% Outer loop

  stop if $\|\mathbf{r}_k\| \leq$ tol

  $\mathbf{u}_k = \text{Solve}(\mathbf{A}, \mathbf{r}_k)$ %% Inner loop

  $\mathbf{x}_{k+1} = \mathbf{x}_k + \mathbf{u}_k \, \alpha_k$

  $\mathbf{c}_k = \mathbf{A}\mathbf{u}_k, \quad \mathbf{r}_{k+1} = \mathbf{r}_k - \mathbf{c}_k \, \alpha_k$

Subspace acceleration. Replace '$\mathbf{u}_k = \text{Solve}(\mathbf{A}, \mathbf{r}_k)$' by

$$\mathbf{t} = \text{Solve}(\mathbf{A}, \mathbf{r}_k),\ \mathbf{u}_k = \text{Orth}(\mathbf{U}_{k-1}, \mathbf{t}),\ \mathbf{U}_k = [\mathbf{U}_{k-1}, \mathbf{u}_k]$$

'Orth' orthogonalises (bi-orth., $\mathbf{A}^*\mathbf{A}$-orth. ...) $\mathbf{t}$ against $\mathbf{U}_{k-1}$.

Iterative methods (review)

The central idea in most iterative methods for solving $\mathbf{Ax} = \mathbf{b}$ is **error correction** using the **residual equation**: $\mathbf{Au} = \mathbf{r}_k$, $\mathbf{u}$ is the error in $\mathbf{x}_k$:

Select $\mathbf{x}_0$. Compute $\mathbf{r}_0 = \mathbf{b} - \mathbf{A}\mathbf{x}_0$

for $k = 0, 1, \ldots, k_{max}$ do %% Outer loop

  stop if $\|\mathbf{r}_k\| \leq$ tol

  $\mathbf{u}_k = \text{Solve}(\mathbf{A}, \mathbf{r}_k)$ %% Inner loop

  $\mathbf{x}_{k+1} = \mathbf{x}_k + \mathbf{u}_k \, \alpha_k$

  $\mathbf{c}_k = \mathbf{A}\mathbf{u}_k, \quad \mathbf{r}_{k+1} = \mathbf{r}_k - \mathbf{c}_k \, \alpha_k$

Combine approaches

- Nested GCR with preconditioner

⋮

Iterative methods (review)

The central idea in most iterative methods for solving $\mathbf{Ax} = \mathbf{b}$ is **error correction** using the **residual equation**: $\mathbf{Au} = \mathbf{r}_k$, $\mathbf{u}$ is the error in $\mathbf{x}_k$:

Select $\mathbf{x}_0$. Compute $\mathbf{r}_0 = \mathbf{b} - \mathbf{A}\mathbf{x}_0$

for $k = 0, 1, \ldots, k_{max}$ do %% Outer loop

  stop if $\|\mathbf{r}_k\| \leq$ tol

  $\mathbf{u}_k = \text{Solve}(\mathbf{A}, \mathbf{r}_k)$ %% Inner loop

  $\mathbf{x}_{k+1} = \mathbf{x}_k + \mathbf{u}_k$

  $\mathbf{c}_k = \mathbf{A}\mathbf{u}_k, \quad \mathbf{r}_{k+1} = \mathbf{r}_k - \mathbf{c}_k$

Alternate between two different 'Solves' in consecutive steps.

- Bi-CGSTAB: step $k/2$ of Bi-CG, LMR
- **Multigrid**: 'smooth' (i.e., ℓ steps of a basic method as Gauss--Seidel), apply 'coarse grid correction'.

Program Lecture 14

Multigrid

- Basic preconditioners
- Coarse grid correction
- What we didn’t talk about

Introduction

Multigrid is a solution technique for linear systems from discretised elliptic equations.

Multigrid methods are quite powerful for classes of problems, in particular for Poisson problems the complexity is of optimal order n.

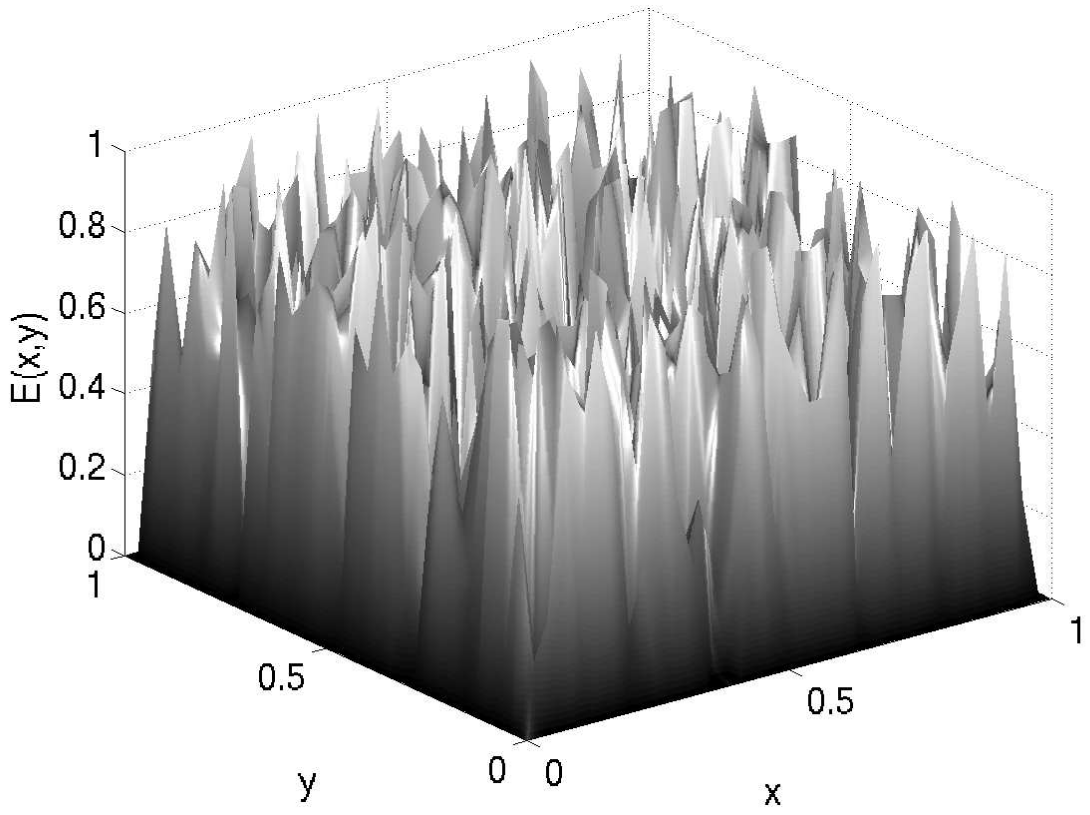
The theory on Multigrid methods is vast. Today we only give a short introduction and the main ideas.

Basic preconditioners

In lecture 10 we discussed some basic preconditioners: Jacobi, Gauss-Seidel, SSOR, and ILU. Although the preconditioners discussed before can considerably reduce the number of iterations, they do normally not reduce the mesh-dependency of the number of iterations.

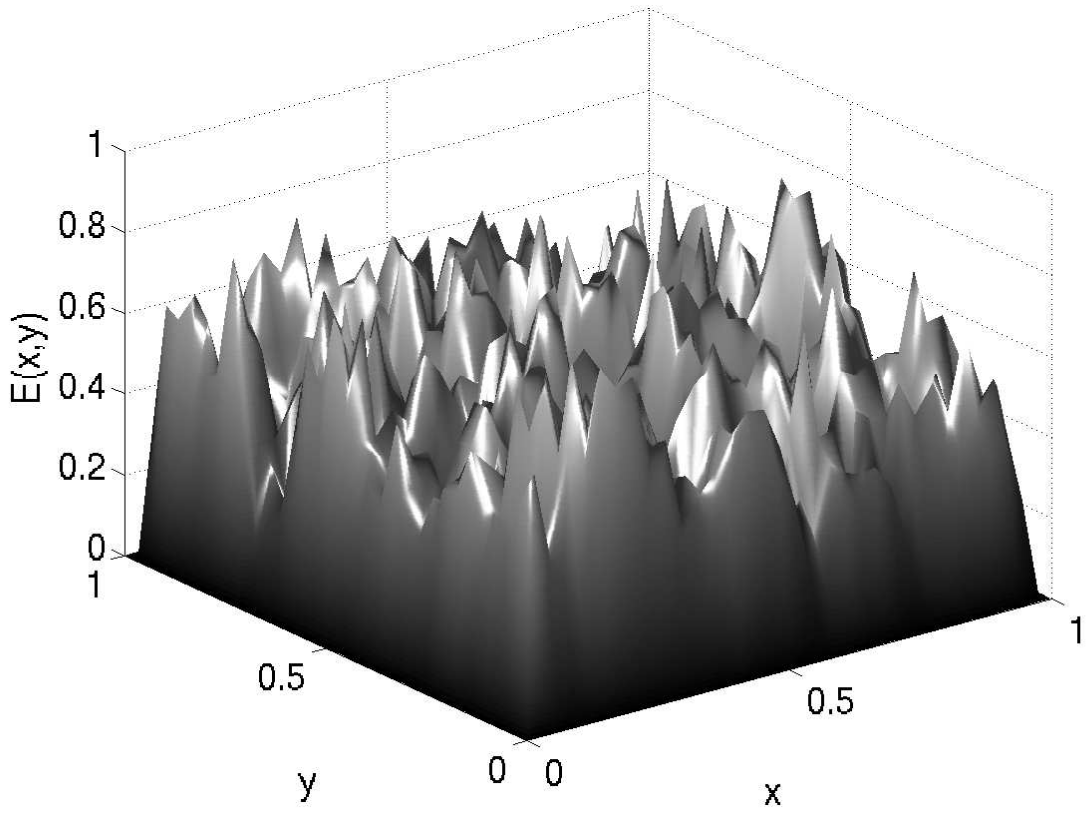
In the next slides we take a closer look at how basic iterative methods reduce the error. From the observations we make, we will develop the idea that is at the basis of one of the fastest techniques: **Multigrid**.

Smoothing Property



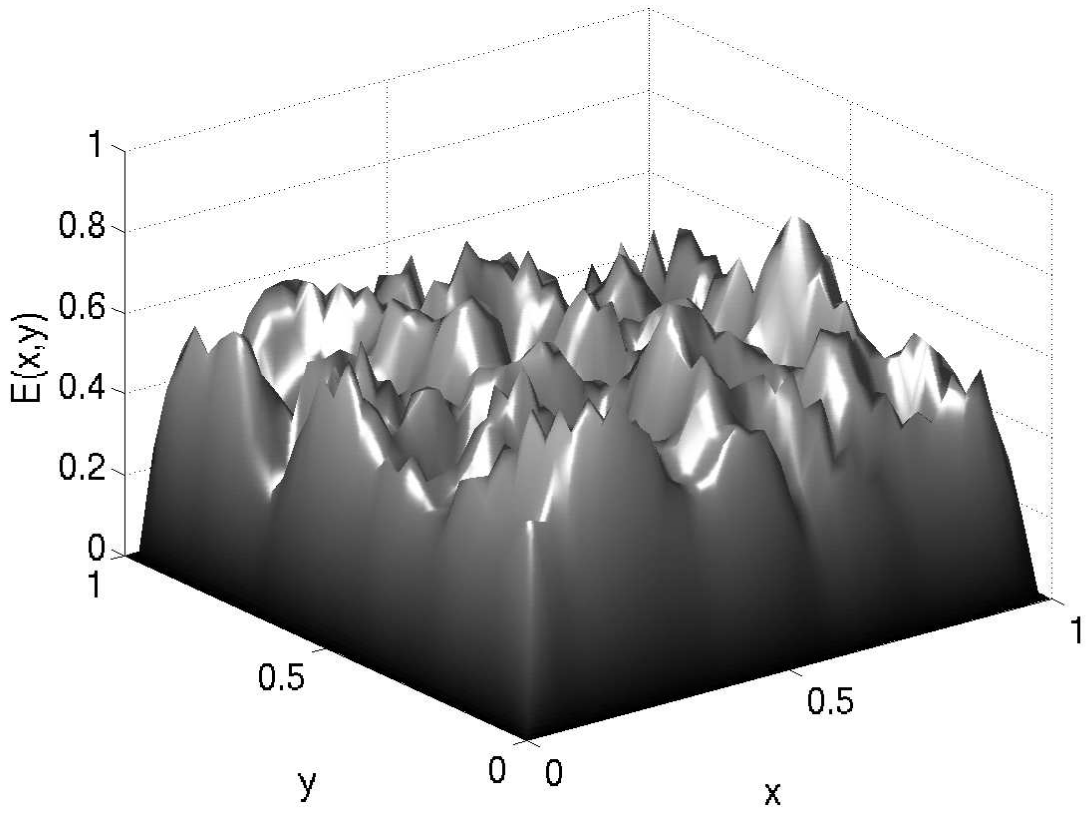
Random initial error

Smoothing Property



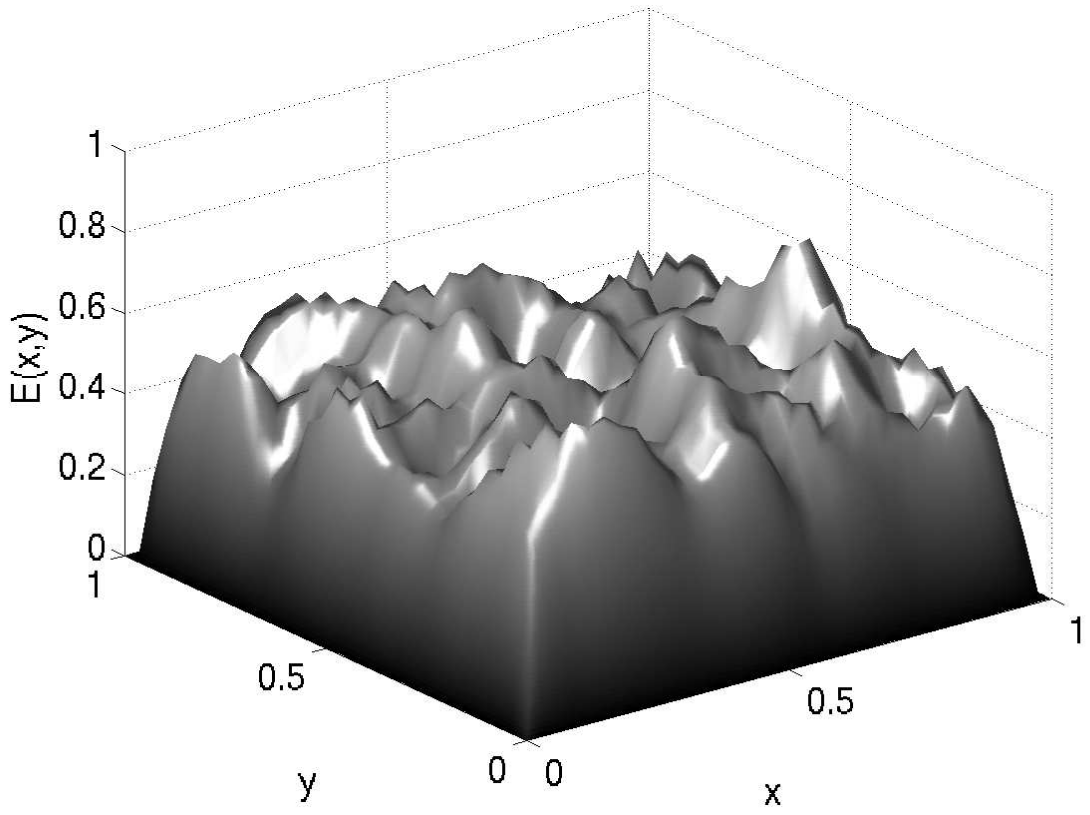
Error after 1 Jacobi iterations

Smoothing Property



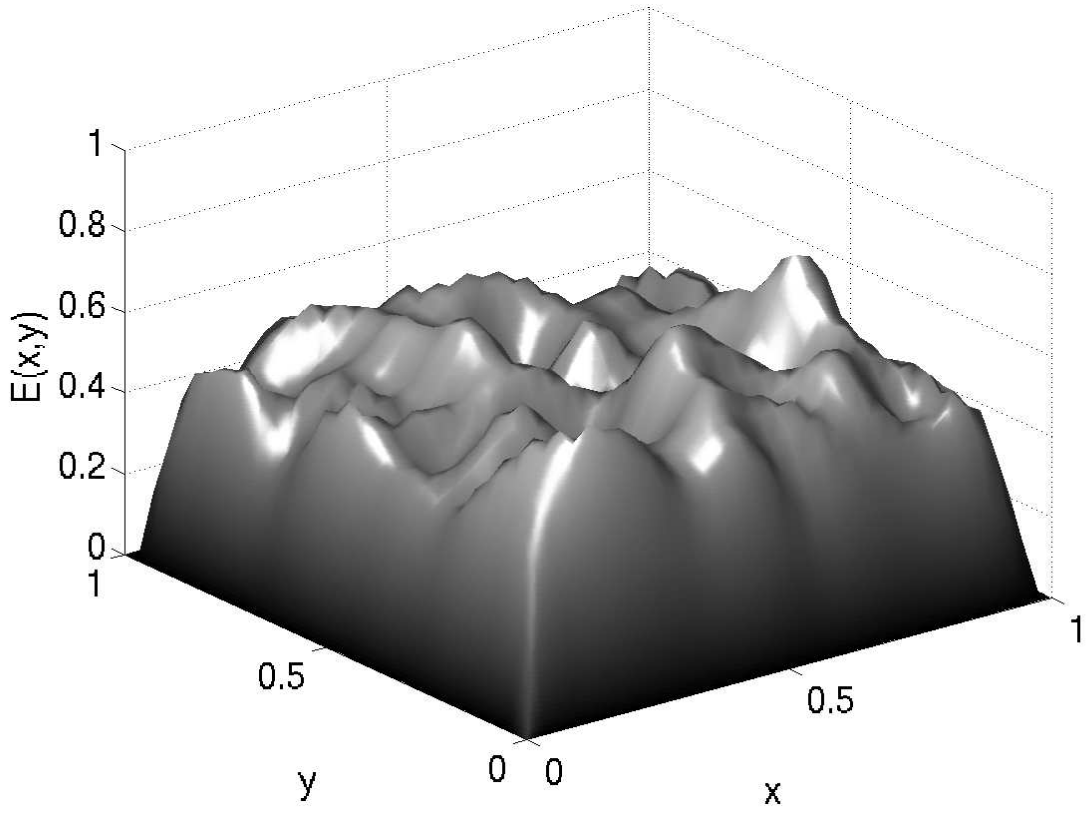
Error after 2 Jacobi iterations

Smoothing Property



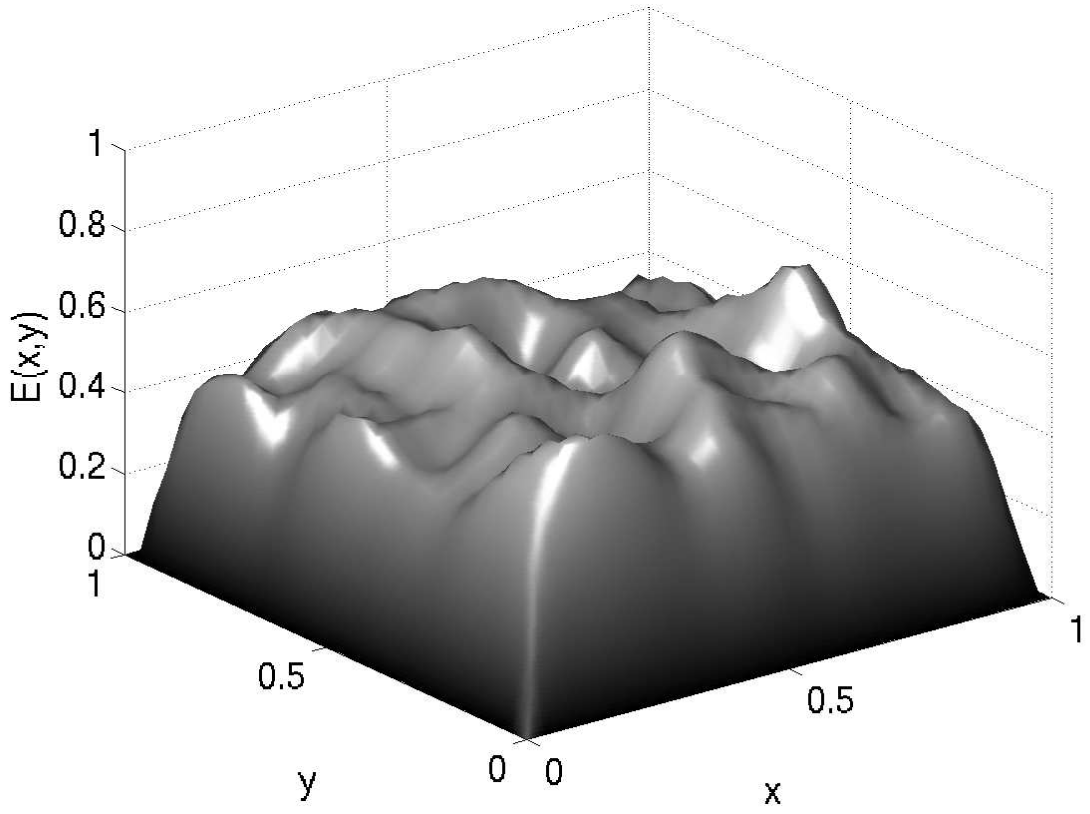
Error after 3 Jacobi iterations

Smoothing Property



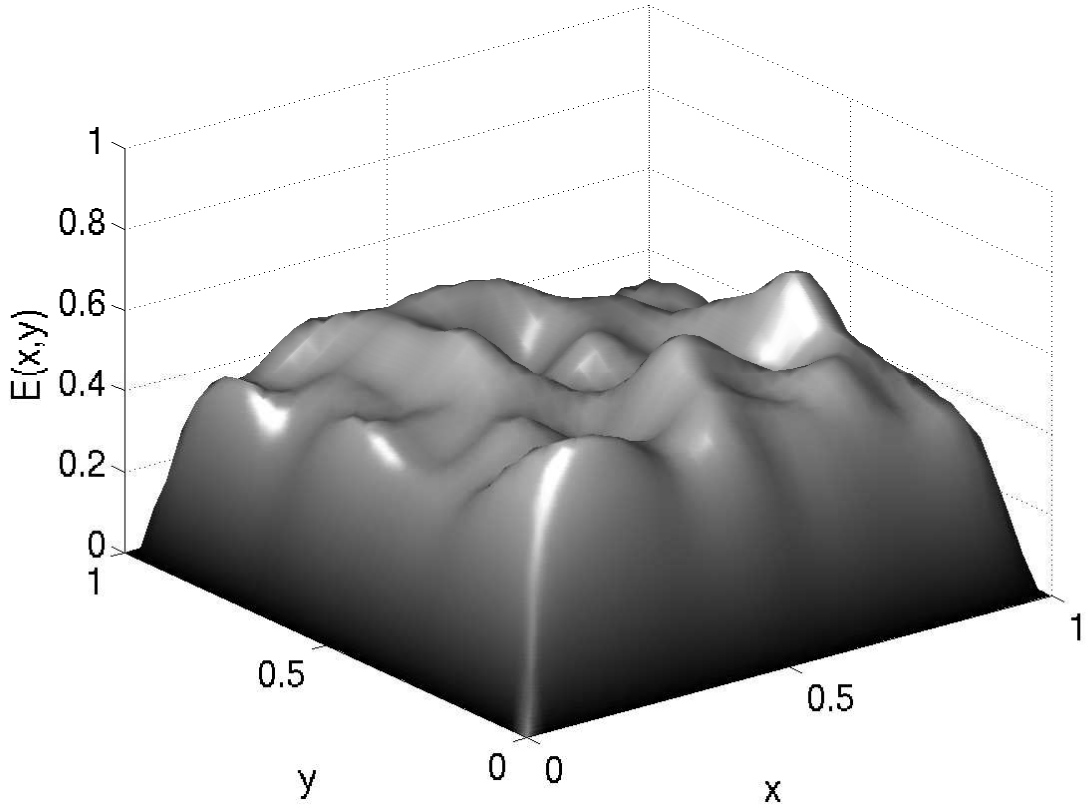
Error after 4 Jacobi iterations

Smoothing Property



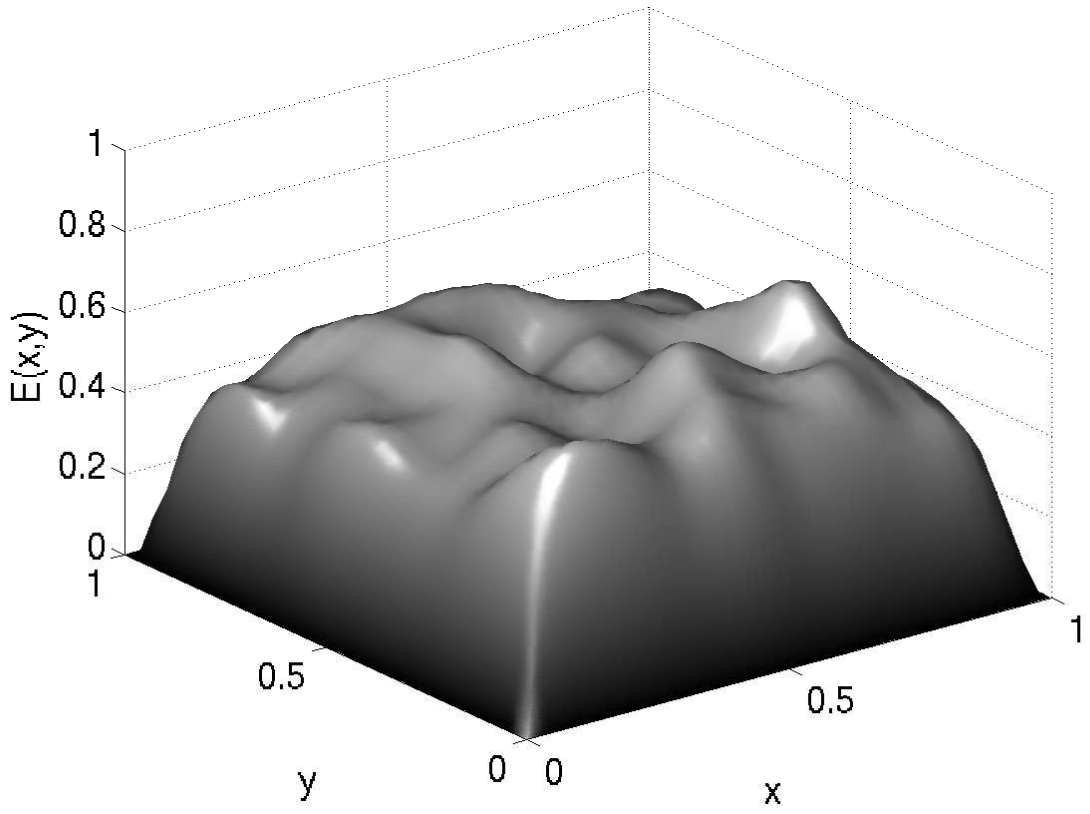
Error after 5 Jacobi iterations

Smoothing Property



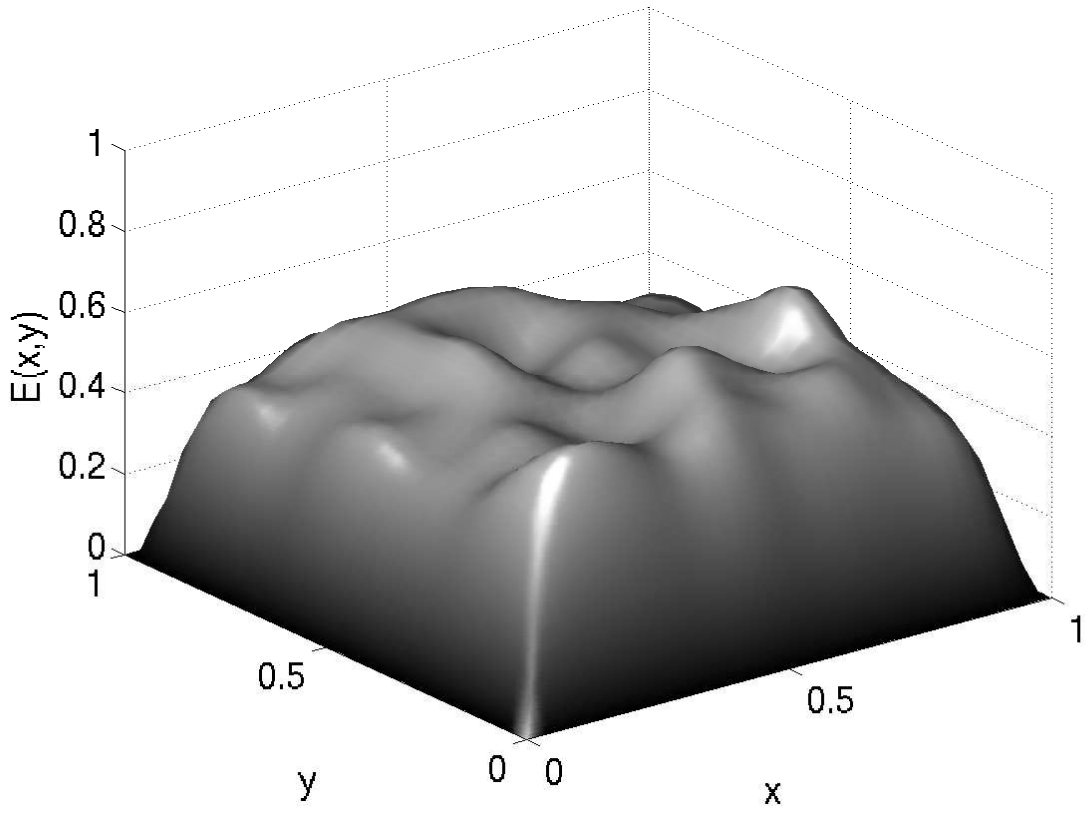
Error after 6 Jacobi iterations

Smoothing Property



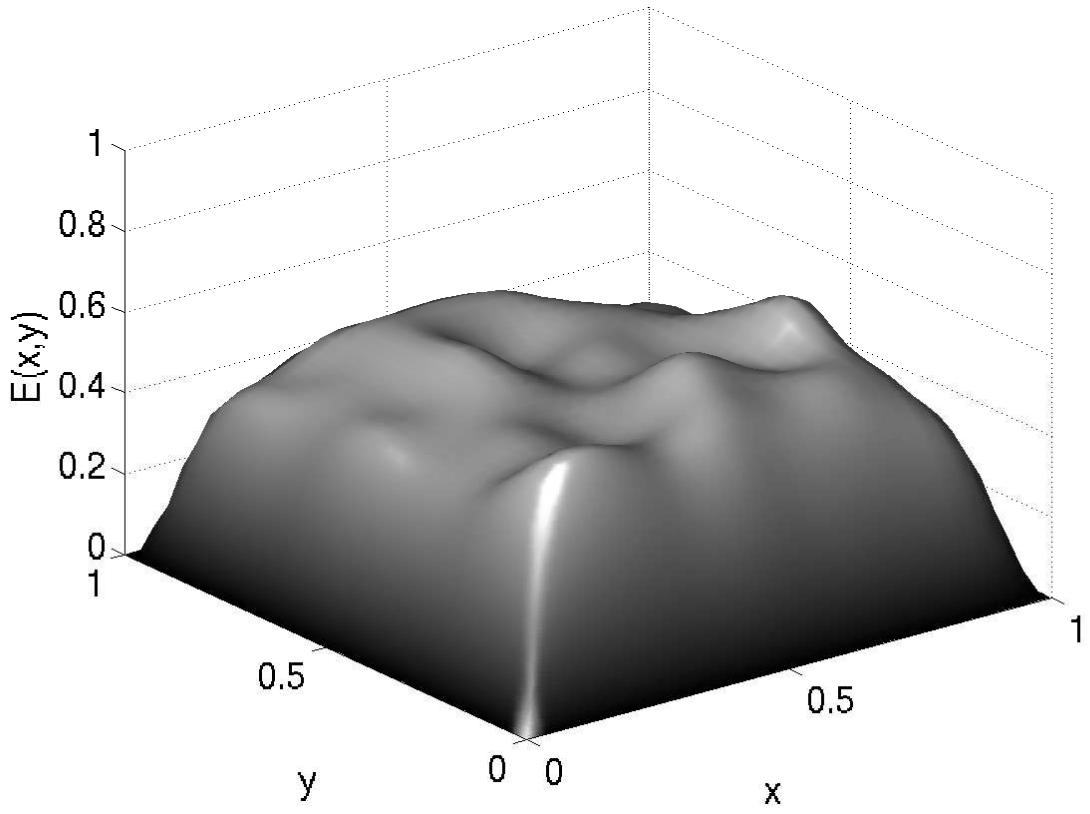
Error after 7 Jacobi iterations

Smoothing Property



Error after 8 Jacobi iterations

Smoothing Property



Error after 10 Jacobi iterations

Smoothing Property

Suppose we apply demped Jacobi with a fixed demping parameter α (i.e., we apply Richardson iteration to the system $\mathbf{D}^{-1}\mathbf{A}\mathbf{x} = \mathbf{D}^{-1}\mathbf{b}$. Here $\mathbf{D} = \mathbf{D}_A$ is the diagonal of $\mathbf{A}$).

If λ is an eigenvalue of the matrix $\mathbf{D}^{-1}\mathbf{A}$, then the component of the error in the direction of the eigenvector associated to λ is multiplied by $1 - \alpha\lambda$: the value of the parameter α determines what components will be demped.

In many applications (as discr. elliptic PDEs), eigenvalues λ close to 0 are associated with 'smooth' eigenvectors, while the eigenvectors with large eigenvalues are highly oscillating.

If the eigenvalues of the matrix $\mathbf{D}^{-1}\mathbf{A}$ are in $(0, \Gamma]$ and $\alpha = 1/\Gamma$, then all components will be demped, but the ones with eigenvalue close to Γ will be be demped most.

If $\frac{1}{\alpha} < \frac{1}{2}\Gamma$, then components with eigenvalues close to Γ get amplified.

Complementarity

- Error after a few Jacobi iterations has some structure. This is also the case for the other basic iterative methods.
- Instead of discarding the method, look to complement of its ‘failings’.

How can we best correct errors
that are slowly reduced by basic iterative method?

Complementarity

- Error after a few Jacobi iterations has some structure. This is also the case for the other basic iterative methods.
- Instead of discarding the method, look to complement of its ‘failings’.

How can we best correct errors
that are slowly reduced by basic iterative method?

- Slow-to-converge errors are smooth
- Smooth vectors can be accurately represented using fewer degrees of freedom

Coarse-Grid Correction

- Smooth vectors can be accurately represented using fewer degrees of freedom
- Idea: transfer job of resolving smooth components to a coarser grid version of the problem
- Need:
  - Complementary process for resolving smooth components of the error on the coarse grid
  - Way to combine the results of the two processes

Multigrid

- **Relaxation** is the name for applying one or a few basic iteration steps.
- Idea is to correct the approximation after relaxation, $\mathbf{x}^{(1)}$, from a coarse-grid version of the problem.
- Need interpolation or **prolongation map**, $\mathbf{P}$, from coarse grid to fine grid.
- Corrected approximation will be $\mathbf{x}^{(2)} = \mathbf{x}^{(1)} + \mathbf{P}\mathbf{u}_c$,
- where $\mathbf{u}_c$ is the solution of the coarse-grid residual problem and satisfies

$$(\mathbf{P}^T\mathbf{A}\mathbf{P})\mathbf{u}_c = \mathbf{P}^T\mathbf{A}(\mathbf{x} - \mathbf{x}^{(1)}) = \mathbf{P}^T\mathbf{r}^{(1)}.$$

Two-grid cycle

Two-grid cycle

Multigrid Components

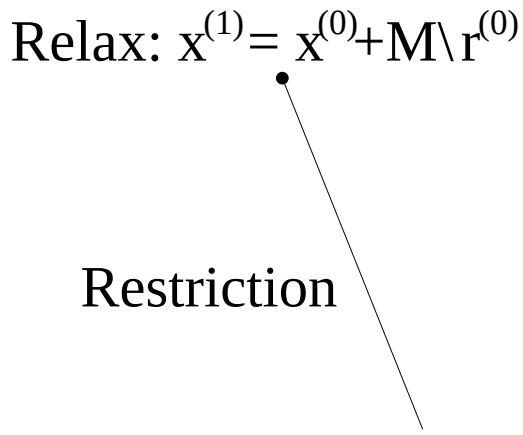
- **Relaxation** Relax: $x^{(1)} = x^{(0)} + M \backslash r^{(0)}$

- Use a **smoothing process** (such as Jacobi or Gauss-Seidel) to eliminate oscillatory errors
- Remaining error satisfies $\mathbf{A}\mathbf{e}^{(1)} = \mathbf{r}^{(1)} = \mathbf{b} - \mathbf{A}\mathbf{x}^{(1)}$

Two-grid cycle

Multigrid Components

- **Relaxation**
- **Restriction**

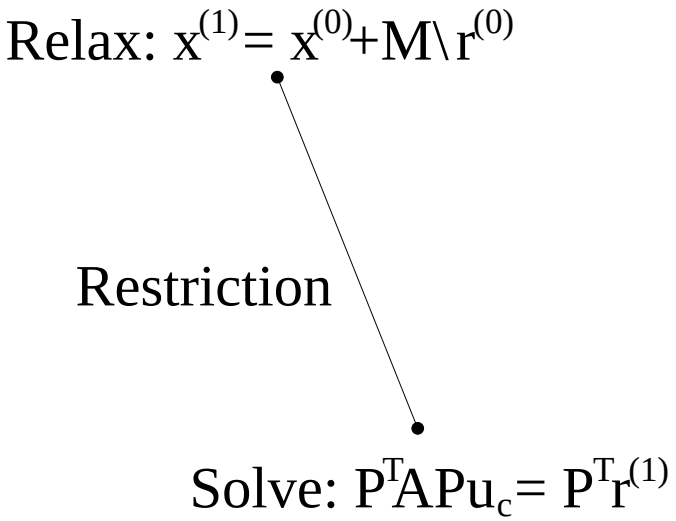


- Transfer residual to coarse grid
- Compute $\mathbf{P}^T \mathbf{r}^{(1)}$

Two-grid cycle

Multigrid Components

- **Relaxation**
- **Restriction**
- **Coarse-Grid Correction**



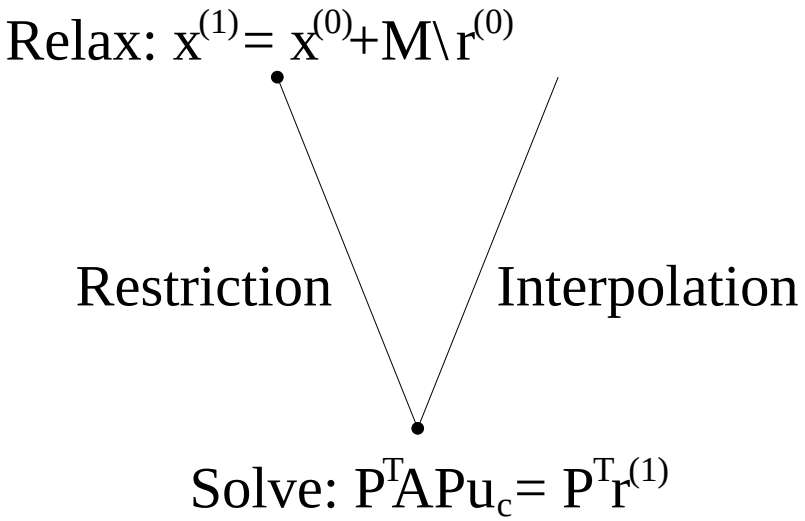
- Use coarse-grid correction to eliminate smooth errors
- Best correction $\mathbf{u}_c$ satisfies

$$\mathbf{P}^T\mathbf{A}\mathbf{P}\mathbf{u}_c = \mathbf{P}^T\mathbf{r}^{(1)}$$

Two-grid cycle

Multigrid Components

- **Relaxation**
- **Restriction**
- **Coarse-Grid Correction**
- **Interpolation**

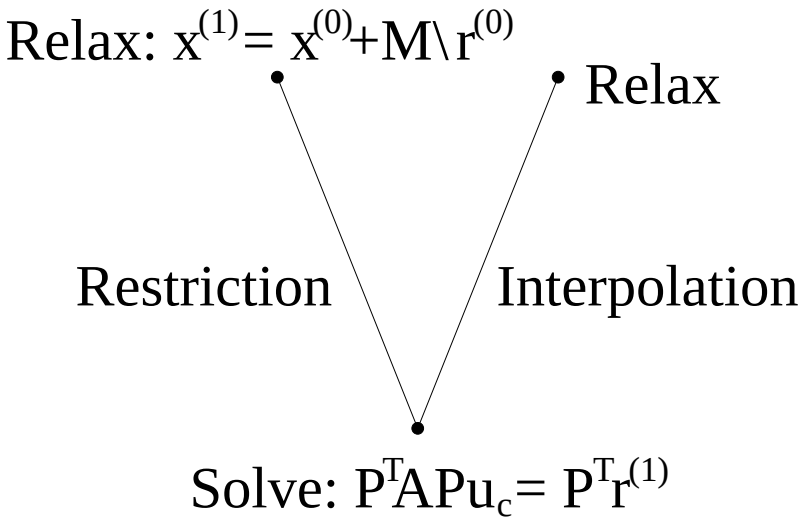


- Transfer correction to fine grid
- Compute $\mathbf{x}^{(2)} = \mathbf{x}^{(1)} + \mathbf{P}\mathbf{u}_c$

Two-grid cycle

Multigrid Components

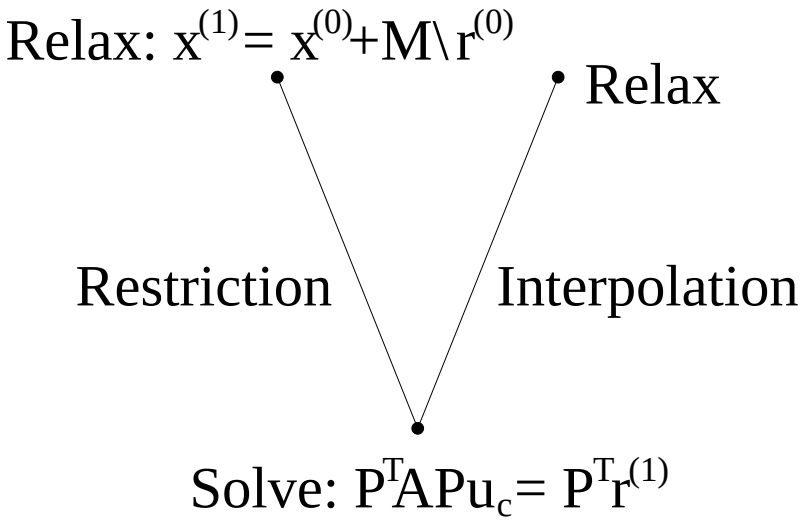
- **Relaxation**
- **Restriction**
- **Coarse-Grid Correction**
- **Interpolation**
- **Relaxation**
- Relax once again to remove oscillatory error introduced in coarse-grid correction



Two-grid cycle

Multigrid Components

- **Relaxation**
- **Restriction**
- **Coarse-Grid Correction**
- **Interpolation**
- **Relaxation**

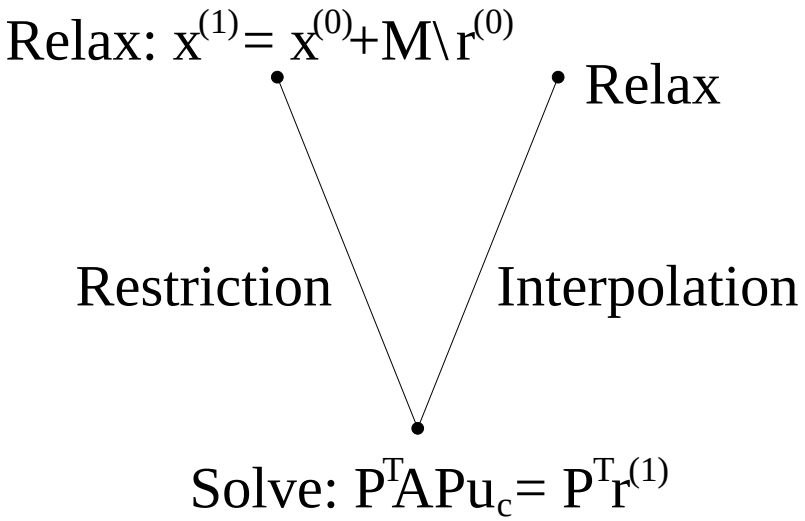


Direct solution of coarse-grid problem isn't practical

Two-grid cycle

Multigrid Components

- **Relaxation**
- **Restriction**
- **Coarse-Grid Correction**
- **Interpolation**
- **Relaxation**



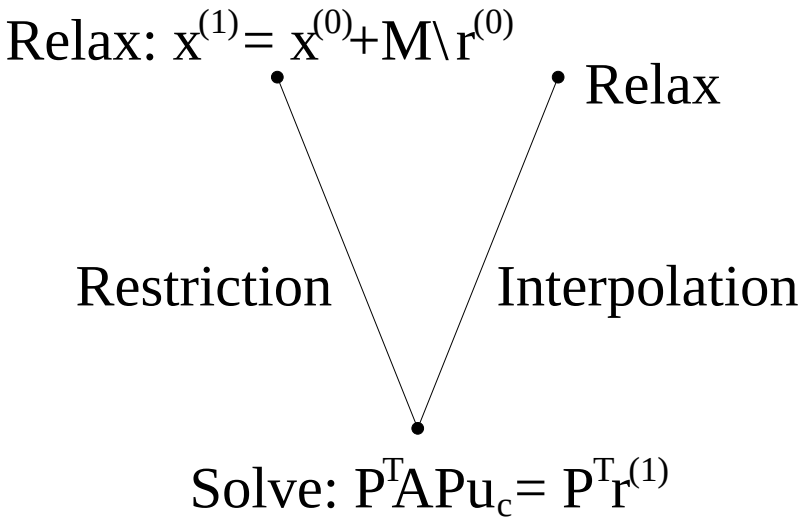
Direct solution of coarse-grid problem isn't practical

Use an iterative method!

Two-grid cycle

Multigrid Components

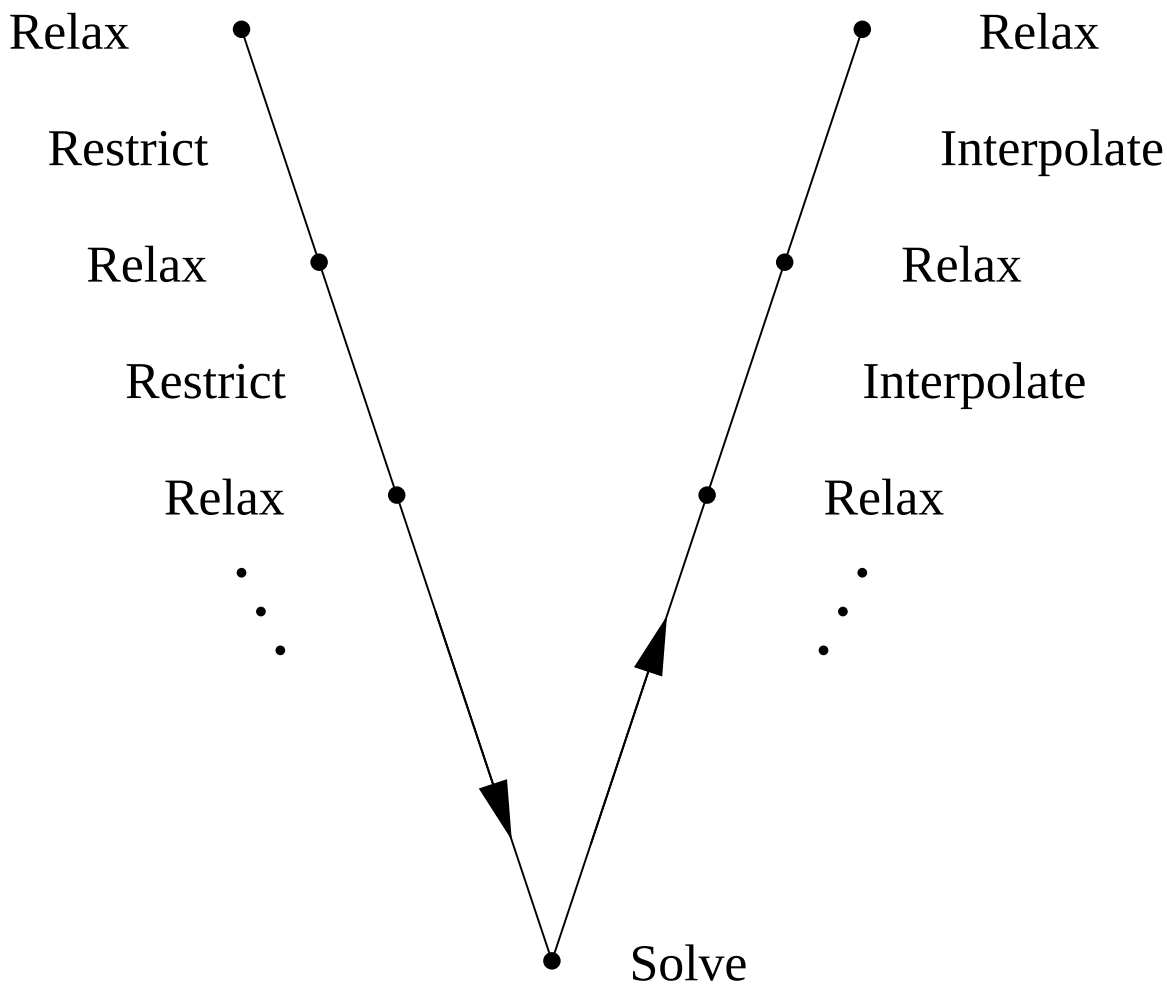
- **Relaxation**
- **Restriction**
- **Coarse-Grid Correction**
- **Interpolation**
- **Relaxation**



Recursion!

Apply same methodology to solve coarse-grid problem

The Multigrid V-cycle



Properties of Effective Cycles

- **Fast convergence**
  - Effective reduction of all error components
  - On each level, coarse-grid correction must effectively reduce exactly those errors that are slow to be reduced by relaxation alone
  - Hierarchy of coarse-grid operators resolves relevant physics at each scale
- **Low iteration cost**
  - Simple relaxation scheme (cheap computations)
  - Sparse coarse-grid operators
  - Sparse interpolation/restriction operations

Properties of Effective Cycles

With n the dimension of $\mathbf{A}\mathbf{x} = \mathbf{b}$ on the finest grid:

- **Fast convergence**

  If $\mathbf{x} - \mathbf{x}_k$ is the error after k multigrid cycles then, typically, for some $\rho \in [0, 1)$, we have

  $$\|\mathbf{x} - \mathbf{x}_{k+1}\|_2 \leq \rho \|\mathbf{x} - \mathbf{x}_k\|_2 \quad \text{for all } k$$

  with ρ independent of n, i.e., of the level of the grid.

  Recall that ρ for methods as Gauss-Jacobi is of the form $1 - \frac{2}{\mathcal{C}(\mathbf{A})}$ and $\mathcal{C}(\mathbf{A}) \sim n$: level dependent, $\rho \uparrow 1$ for $n \to \infty$.

- **Low iteration cost**

Properties of Effective Cycles

With n the dimension of $\mathbf{A}\mathbf{x} = \mathbf{b}$ on the finest grid:

- **Fast convergence**

$$\|\mathbf{x} - \mathbf{x}_{k+1}\|_2 \leq \rho \|\mathbf{x} - \mathbf{x}_k\|_2 \quad \text{for all } k$$

- **Low iteration cost**

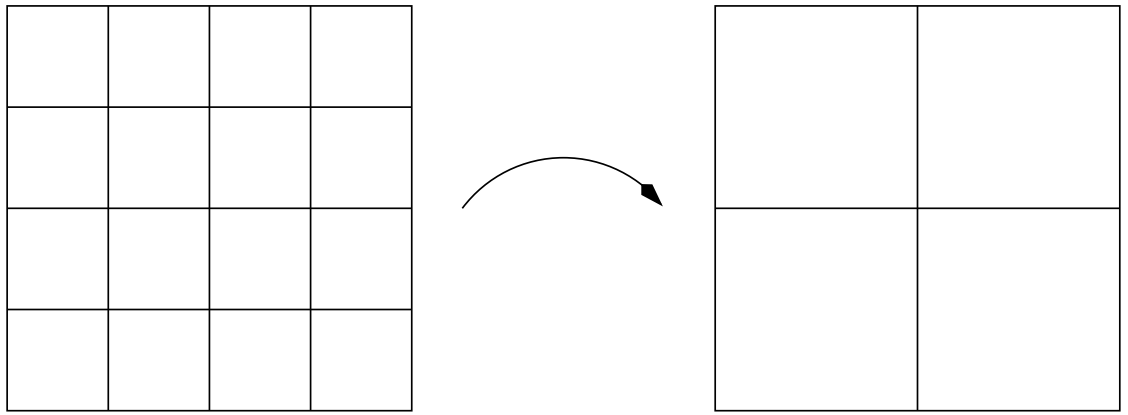
  The costs per cycle are κn where the value of κ depends on the relaxation scheme (as, e.g., the number of relaxation steps) but not on n (the selected number of relaxation steps is independent of the grid level!)

Total costs to have an error $\leq \epsilon$ are $\kappa n \, |\log \epsilon| / |\log \rho| = \mathcal{O}(n)$.

Typically, $\rho \downarrow$ if $\kappa \uparrow$. Challenge: find the optimal balance.

Choosing Coarse Grids

- No *best* strategy to choose coarse grids
- Operator dependent, but also machine dependent
- For structured meshes, often use uniform de-refinement approach



- For unstructured meshes, various weighted independent set algorithms are often used.

What didn’t we talk about?

- How do we choose $\mathbf{P}$?
  - Number of columns
  - Sparsity structure
  - Non-zero values
- Choices depend closely on the properties of the relaxation method

Concluding remarks about Multigrid

Multigrid works well if the problem

- is grid-based. However, matrix-based Multigrid methods (**Algebraic Multigrid**) do exist and are often successful;
- has a smooth solution. An underlying assumption is that the error can be represented on a coarser grid. Multigrid works particularly well for Poisson-type problems. For these problems the number of operations is $\mathcal{O}(n)$.

Multigrid can be used as a separate solver, but is often used as a preconditioner for a Krylov-type method, or for example as building block in a saddle-point preconditioner.