Multilevel Methods for HPC Lecture Notes

Jan Van lent

BICS, University of Bath, UK

Friday 23 November 2007

Introduction

- 2 lectures, each $90'$
- mainly introduction to multigrid
- practical session
  - implement Jacobi and Gauss-Seidel for 1D and 2D Poisson with Dirichlet boundary conditions on unit interval/square
  - implement full weighting restriction, (bi)linear interpolation
  - implement multigrid
- goal: explain and illustrate the multigrid method using the problems it was originally designed for

Overview

- model problems
- discretisation
- basic iterative methods
- two-grid method
- multigrid method
- advanced topics

Model Problem 1D

- multigrid originally designed for elliptic PDEs
  since then extended to other classes of problems
- model problems
- 1D Poisson equation
- find u given f s.t.

$$-u_{xx} = f, \qquad x \in [0,1], \quad u, f : [0,1] \longrightarrow \mathbb{R}$$

- Dirichlet boundary conditions

$$u(0) = g(0), \qquad u(1) = g(1)$$

- many applications: e.g. temperature in rod, g specifies temperature at endpoints, f describes sources and sinks of energy

Model Problem 2D

- 2D Poisson equation on unit interval $\Omega = [0,1]^2$

$$-u_{xx} - u_{yy} = f$$

- where

$$u : \Omega \to \mathbb{R} : (x,y) \to u(x,y)$$

- Dirichlet boundary conditions

$$\begin{aligned} u(x,0) &= g(x,0), \quad & u(1,y) &= g(1,y), \\ u(x,1) &= g(x,1), \quad & u(0,y) &= g(0,y) \end{aligned}$$

- or

$$u(x,y) = g(x,y), \qquad (x,y) \in \partial\Omega$$

- application: e.g. temperature in plate with specified temperature on the boundary

Discretisation

- model problems are continuous equations
- solution takes values at infinite number of points
- can only be represented exctly in special cases
- approximate by finite number of values
- discretisation
- many methods: finite differences, finite volumes, finite elements, spectral methods
- here finite differences: simple and sufficient for our purposes
- multigrid can be applied to the other discretisations as well

Discretisation 1D

- 1D Poisson: $-u_{xx} = f$
- interval $[0, 1]$, n subintervals, $\Delta x = \frac{1}{n}$
- grid points: $x_i = i\Delta x$, $i = 0, \ldots, n$
- function values: $f_i = f(x_i)$, $u_i \approx u(x_i)$
- first derivative: $u_x(x) \approx \frac{u(x+\Delta x/2)-u(x-\Delta x/2)}{\Delta x}$
- second derivative: $u_{xx}(x) \approx \frac{u_x(x+\Delta x/2)-u_x(x-\Delta x/2)}{\Delta x}$
- using values at grid ponts: $u_{xx} \approx \frac{u_{i-1}-2u_i+u_{i+1}}{\Delta x^2}$

Discretisation 1D (cont.)

- interval $[0, 1]$, n subintervals, $\Delta x = \frac{1}{n}$
- system of equations:
  $\frac{-u_{i-1}+2u_i-u_{i+1}}{\Delta x^2} = f_i, \quad i = 1, \ldots, n-1$
- boundary conditions: $u_0 = g(0), \qquad u_n = g(1)$
- $(n-1)$ equations, $(n-1)$ unknowns
- discretisation error: $O(\Delta x^2)$

Discretisation 1D (cont.)

- system of equations:
  $\frac{-u_{i-1}+2u_i-u_{i+1}}{\Delta x^2} = f_i, \quad i = 1, \ldots, n-1$
- boundary conditions: $u_0 = g(0), \qquad u_n = g(1)$
- vectors $u, f, g \in \mathbb{R}^{n-1}$

$$u = \begin{bmatrix} u_1 \\ u_2 \\ \vdots \\ u_{n-2} \\ u_{n-1} \end{bmatrix}, \quad f = \begin{bmatrix} f_1 \\ f_2 \\ \vdots \\ f_{n-2} \\ f_{n-1} \end{bmatrix}, \quad g = \frac{1}{\Delta x^2} \begin{bmatrix} g(0) \\ 0 \\ \vdots \\ 0 \\ g(1) \end{bmatrix}$$

Matrix Formulation 1D

- system of equations: $Lu = f$
- matrix

$$L = \frac{1}{\Delta x^2} \begin{bmatrix} 2 & -1 & & \\ -1 & \ddots & \ddots & \\ & \ddots & \ddots & -1 \\ & & -1 & 2 \end{bmatrix} \in \mathbb{R}^{(n-1)\times(n-1)}$$

- symmetric, positive definite, sparse, banded, Toeplitz

Stencil Notation 1D

- system of equations:
  $\frac{-u_{i-1}+2u_i-u_{i+1}}{\Delta x^2} = f_i, \quad i = 1, \ldots, n-1$
- compact notation $Lu = f$
- grid functions: u and f
- stencil notation for linear operator

$$L = \left[\frac{-1}{\Delta x^2} \quad \frac{2}{\Delta x^2} \quad \frac{-1}{\Delta x^2} \right]$$

- meaning

$$(Lu)_i = \frac{-u_{i-1} + 2u_i - u_{i+1}}{\Delta x^2}$$

Discretisation 2D

- 2D Poisson: $-u_{xx} - u_{yy} = f$
- unit square $[0,1]^2$, $\Delta x = \frac{1}{n_x}$, $\Delta y = \frac{1}{n_y}$
- grid points: $(x_i, y_j) = (i\Delta x, j\Delta y)$,
  $i = 0, \ldots, n_x$, $j = 0, \ldots, n_y$
- function values: $f_{i,j} = f(x_i, y_j)$, $u_{i,j} \approx u(x_i, y_j)$
- second derivative:
  $u_{xx}(x,y) = \frac{u(x-\Delta x,y) - 2u(x,y) + u(x+\Delta x,y)}{\Delta x^2}$
- using values at grid points:
  $u_{xx}(x_i, y_j) = \frac{u_{i-1,j} - 2u_{i,j} + u_{i+1,j}}{\Delta x^2}$

Discretisation 2D (cont.)

- system of equations $i = 1, \ldots, n_x - 1, \quad j = 1, \ldots, n_y - 1$
  $\frac{-u_{i-1,j}+2u_{i,j}-u_{i+1,j}}{\Delta x^2} + \frac{-u_{i,j-1}+2u_{i,j}-u_{i,j+1}}{\Delta y^2} = f_{i,j}$
- boundary conditions $i = 0, \ldots, n_x, \quad j = 0, \ldots, n_y$

$$\begin{aligned} u_{i,0} &= g(x_i, 0), & u_{1,j} &= g(1, y_j), \\ u_{i,1} &= g(x_i, 1), & u_{0,j} &= g(0, y_j) \end{aligned}$$

- vectors $u, f, g \in \mathbb{R}^{(n_x-1)(n_y-1)}$, $_{i = 0, \ldots, n_x, \quad j = 0, \ldots, n_y}$

$$u_k = u_{i,j}, \quad f_k = f_{i,j}, \quad g_k = g(x_i, y_j)$$

- where $g(x, y) = 0$ for (x, y) in interior of Ω
- lexicographical ordering of unknowns
  $k = (i-1)(n_y-1) + (j-1)$

Matrix Formulation 2D

- system of equations: $Lu = f$
- matrix

$$L = \begin{bmatrix} T & D & & \\ D & \ddots & \ddots & \\ & \ddots & \ddots & D \\ & & D & T \end{bmatrix}, \; D = \begin{bmatrix} -\frac{1}{\Delta x^2} & & & \\ & \ddots & & \\ & & \ddots & \\ & & & -\frac{1}{\Delta x^2} \end{bmatrix}$$

$$T = \begin{bmatrix} \frac{2}{\Delta x^2} + \frac{2}{\Delta y^2} & -\frac{1}{\Delta y^2} & & \\ -\frac{1}{\Delta y^2} & \ddots & \ddots & \\ & \ddots & \ddots & -\frac{1}{\Delta y^2} \\ & & -\frac{1}{\Delta y^2} & \frac{2}{\Delta x^2} + \frac{2}{\Delta y^2} \end{bmatrix}$$

Tensor Product Notation

- Kronecker product: $A \in \mathbb{R}^{m \times n}$, $B \in \mathbb{R}^{p \times q}$

$$A \otimes B = \begin{bmatrix} a_{11}B & \dots & a_{1n}B \\ \vdots & & \vdots \\ a_{m1}B & \dots & a_{mn}B \end{bmatrix} \in \mathbb{R}^{mp \times nq}$$

- 1D operators: $T_n, I_n \in \mathbb{R}^{n \times n}$

$$T_n = \begin{bmatrix} 2 & -1 & & \\ -1 & \ddots & \ddots & \\ & \ddots & \ddots & -1 \\ & & -1 & 2 \end{bmatrix}, \; I_n = \begin{bmatrix} 1 & & & \\ & \ddots & & \\ & & \ddots & \\ & & & 1 \end{bmatrix}$$

- 2D operator:
$L = \frac{1}{\Delta x^2} T_{n_x - 1} \otimes I_{n_y - 1} + \frac{1}{\Delta y^2} I_{n_x - 1} \otimes T_{n_y - 1}$

Stencil Notation 2D

- system of equations:
  $\frac{-u_{i-1,j}+2u_{i,j}-u_{i+1,j}}{\Delta x^2} + \frac{-u_{i,j-1}+2u_{i,j}-u_{i,j+1}}{\Delta y^2} = f_{i,j}$
- compact notation $Lu = f$
- grid functions: u and f
- stencil notation for linear operator

$$L = \begin{bmatrix} & \frac{-1}{\Delta y^2} & \\ \frac{-1}{\Delta x^2} & \frac{2}{\Delta x^2} + \frac{2}{\Delta y^2} & \frac{-1}{\Delta x^2} \\ & \frac{-1}{\Delta y^2} & \end{bmatrix}$$

- meaning $(Lu)_{i,j} = \frac{-u_{i-1,j}+2u_{i,j}-u_{i+1,j}}{\Delta x^2} + \frac{-u_{i,j-1}+2u_{i,j}-u_{i,j+1}}{\Delta y^2}$

Solvers

- system of equations of order m
- methods for solving, typical space/time complexity
- direct methods
  - dense LU $O(m^2)$, $O(m^3)$
  - band, sparse LU: $O(km)$, $O(k^2m)$, $k \to m$
  - FFT, Toeplitz $O(m)$, $O(m \log m)$
- iterative methods $O(m)$ space, $O(m)$ cost/iterations
- basic iterative methods (Jacobi, Gauss-Seidel): many iterations
- Krylov subspace (CG, GMRES): many iterations, unless good preconditioner
- multilevel methods (multigrid, domain decomposition, wavelets): fixed number of iterations

Multigrid Idea

- use basic iterative methods on many grids
- use calculations on coarser grids to accelerate iteration on fine grid
- interaction between two processes: smoothing and coarse grid correction

Basic Iterative Methods: Jacobi

- explained using model problems
- system of equations $\frac{-u_{i-1}+2u_i-u_{i+1}}{\Delta x^2} = f_i$
- solve for u_i assuming we know u_{i-1}, u_{i+1}
- Jacobi iteration $u_i^{(\nu)} \leftarrow \frac{\Delta x^2}{2}(f_i + \frac{u_{i-1}^{(\nu-1)}+u_{i+1}^{(\nu-1)}}{\Delta x^2})$
- matrix notation $Lu = f$
- matrix splitting $L = L^+ + L^-$, L^+ diagonal of L
- iteration $L^+u^{(\nu)} + L^-u^{(\nu-1)} = f$
- stencil notation $L = \left[\frac{-1}{\Delta x^2} \quad \frac{2}{\Delta x^2} \quad \frac{-1}{\Delta x^2}\right] = \left[\bullet \quad \bullet \quad \bullet\right]$
- iteration $\left[\quad \bullet \quad\right] u^{(\nu)} + \left[\bullet \qquad \bullet\right] u^{(\nu-1)} = f$

Weighted Jacobi

- Jacobi iterate $\tilde{u}_i \leftarrow \frac{\Delta x^2}{2}(f_i + \frac{u_{i-1}^{(\nu-1)} + u_{i+1}^{(\nu-1)}}{\Delta x^2})$
- weighted average with previous iterate $u_i^{(\nu-1)}$

$$u_i^{(\nu)} \leftarrow (1-\omega)\tilde{u}_i + \omega u_i^{(\nu-1)}$$

- for multigrid e.g. $\omega = \frac{2}{3}$

Basic Iterative Methods: Gauss-Seidel

- Jacobi: all values can be updated independently
- GS: use new value as soon as available
- order is important
- lexicographical GS: $i = 1, \ldots, n_{x-1}$
  $$u_i^{(\nu)} \leftarrow f_i \Delta x^2/2 + (u_{i-1}^{(\nu)} + u_{i+1}^{(\nu-1)})/2$$
- stencil notation
  $$[\bullet \;\; \bullet \;\; \quad] \, u^{(\nu)} + [\quad \quad \;\; \bullet] \, u^{(\nu-1)} = f$$
- reverse lexicographical GS: $i = n_{x-1}, \ldots, 1$
  $$u_i^{(\nu)} \leftarrow f_i \Delta x^2/2 + (u_{i-1}^{(\nu-1)} + u_{i+1}^{(\nu)})/2$$
- stencil notation
  $$[\quad \;\; \bullet \;\; \bullet] \, u^{(\nu)} + [\bullet \quad \quad \;\;] \, u^{(\nu-1)} = f$$

Red-Black Gauss-Seidel

- red points: i odd, $i = 1, 3, 5, \ldots n_{x-1}$
  $u_i^{(\nu)} \leftarrow f_i \Delta x^2/2 + (u_{i-1}^{(\nu-1)} + u_{i+1}^{(\nu-1)})/2$
- black points: i even, $i = 2, 4, \ldots, n_{x-2}$
  $u_i^{(\nu)} \leftarrow f_i \Delta x^2/2 + (u_{i-1}^{(\nu)} + u_{i+1}^{(\nu)})/2$
- stencil notation
- red points
  $\begin{bmatrix} & \bullet & \end{bmatrix} u^{(\nu)} + \begin{bmatrix} \bullet & & \bullet \end{bmatrix} u^{(\nu-1)} = f$
- red black
  $\begin{bmatrix} & \bullet & \end{bmatrix} u^{(\nu)} + \begin{bmatrix} \bullet & & \bullet \end{bmatrix} u^{(\nu)} = f$

Jacobi 2D

- system of equations
$$\frac{-u_{i-1,j}+2u_{i,j}-u_{i+1,j}}{\Delta x^2}+\frac{-u_{i,j-1}+2u_{i,j}-u_{i,j+1}}{\Delta y^2}=f_{i,j}$$
- iteration
$$u_{i,j}^{(\nu)}\leftarrow\left(\frac{2}{\Delta x^2}+\frac{2}{\Delta y^2}\right)^{-1}\left(f_{i,j}+\frac{u_{i-1,j}^{(\nu-1)}+u_{i+1,j}^{(\nu-1)}}{\Delta x^2}+\frac{u_{i,j-1}^{(\nu-1)}+u_{i,j+1}^{(\nu-1)}}{\Delta y^2}\right)$$
- stencil notation

$$\begin{bmatrix} & & \\ & \bullet & \\ & & \end{bmatrix} u^{(\nu)} + \begin{bmatrix} & \bullet & \\ \bullet & & \bullet \\ & \bullet & \end{bmatrix} u^{(\nu-1)} = f$$

Lexicographical Gauss-Seidel 2D

- iteration $i = 1, \ldots, n_{x-1},\ j = 1, \ldots, n_{y-1}$

  $$u_{i,j}^{(\nu)} \leftarrow \left(\frac{2}{\Delta x^2} + \frac{2}{\Delta y^2}\right)^{-1}\left(f_{i,j} + \frac{u_{i-1,j}^{(\nu)} + u_{i+1,j}^{(\nu-1)}}{\Delta x^2} + \frac{u_{i,j-1}^{(\nu)} + u_{i,j+1}^{(\nu-1)}}{\Delta y^2}\right)$$

- stencil notation

$$\begin{bmatrix} & & \\ \bullet & \bullet & \\ & \bullet & \end{bmatrix} u^{(\nu)} + \begin{bmatrix} & \bullet & \\ & & \bullet \\ & & \end{bmatrix} u^{(\nu-1)} = f$$

Red-Black Gauss-Seidel 2D

- red points $i = 1, \ldots, n_{x-1}$, $j = 1, \ldots, n_{y-1}$, i+j even
  $$u_{i,j}^{(\nu)} \leftarrow \left(\frac{2}{\Delta x^2} + \frac{2}{\Delta y^2}\right)^{-1}\left(f_{i,j} + \frac{u_{i-1,j}^{(\nu-1)} + u_{i+1,j}^{(\nu-1)}}{\Delta x^2} + \frac{u_{i,j-1}^{(\nu-1)} + u_{i,j+1}^{(\nu-1)}}{\Delta y^2}\right)$$
- black points $i = 1, \ldots, n_{x-1}$, $j = 1, \ldots, n_{y-1}$, i+j odd
  $$u_{i,j}^{(\nu)} \leftarrow \left(\frac{2}{\Delta x^2} + \frac{2}{\Delta y^2}\right)^{-1}\left(f_{i,j} + \frac{u_{i-1,j}^{(\nu)} + u_{i+1,j}^{(\nu)}}{\Delta x^2} + \frac{u_{i,j-1}^{(\nu)} + u_{i,j+1}^{(\nu)}}{\Delta y^2}\right)$$

Red-Black Gauss-Seidel 2D (cont.)

- stencil notation

$$\begin{bmatrix} & & \\ & \bullet & \\ & & \end{bmatrix} u^{(\nu)} + \begin{bmatrix} & \bullet & \\ \bullet & & \bullet \\ & \bullet & \end{bmatrix} u^{(\nu-1)} = f$$

$$\begin{bmatrix} & & \\ & \bullet & \\ & & \end{bmatrix} u^{(\nu)} + \begin{bmatrix} & \bullet & \\ \bullet & & \bullet \\ & \bullet & \end{bmatrix} u^{(\nu)} = f$$

Stationary Linear Methods

- system of equations: $Lu = f$
- exact solution: u
- iteration $u^{(\nu)} \leftarrow Su^{(\nu-1)}$
- linear: S is matrix
- stationary: S does not depend on ν
- error: $e^{(\nu)} = u - u^{(\nu)}$
- error iteration: $e^{(\nu)} = Se^{(\nu-1)} = S^{\nu} e^{(0)}$

Convergence Analysis

- error norm: $\|e^{(\nu)}\| \leq \|S^{\nu}\| \|e^{(0)}\|$
- assymptotically: $\|e^{(\nu)}\| \leq \rho^{\nu} \|e^{(0)}\|, \qquad \nu \longrightarrow \infty$
- convergence factor: $\rho = \lim_{\nu \to \infty} \|S^{\nu}\|^{\frac{1}{\nu}}$
- convergence if $\rho < 1$, the smaller the better
- convergence rate: $R = -\log_{10} \rho$,
  average extra digits of precision per iteration
- estimates:

$$\rho^{(\nu,\mu)} = \sqrt[\mu]{\frac{\|e^{(\nu)}\|}{\|e^{(\nu-\mu)}\|}}$$

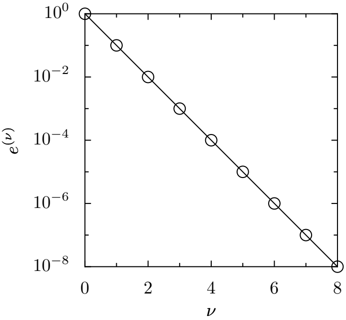
$$R^{(\nu,\mu)} = -\log_{10}(\rho^{(\nu,\mu)})$$

"Ideal" Convergence

- error reduction

$$\|e^{(\nu)}\| \leq \rho \|e^{(\nu)}\|$$

- we aim for $\rho \approx 0.1$, $R \approx 1$



Model Problem Analysis

- Jacobi or Gauss-Seidel
- 1D or 2D Poisson
- convergence factor

$$\rho = 1 - c\Delta x$$

- c some constant
- convergence slows down as $\Delta x \rightarrow 0$

Basic Iterative Methods: Smoothing

- for model problem: $\rho = 1 - O(\Delta x^2)$
- slower as $\Delta x \to 0$
- not practical for realistic problems
- special property
- Jacobi $u_i^{(\nu)} \leftarrow \frac{\Delta x^2}{2}(f_i + \frac{u_{i-1}^{(\nu-1)} + u_{i+1}^{(\nu-1)}}{\Delta x^2})$
- error $e_i^{(\nu)} = \frac{e_{i-1}^{(\nu-1)} + e_{i+1}^{(\nu-1)}}{2}$
- averaging
- oscillatory or high frequency components are reduced quickly
- smooth or low frequency components are reduced slowly

Exploiting Smoothing

- how to exploit smoothing property?
- smooth on fine grid
- smooth error can be represented on coarser grid
- smooth error looks rougher on coarser grid
- intuitively: number of oscillations per grid point
- less work on coarser grid
- idea: use calculation on coarser grid to accelerate iteration on fine grid

Residual Equation

- given approximation v to exact solution u of $Lu = f$
- error $e = u - v$
- satisfies $Le = f - Lv$
- residual $r = f - Lv$
- residual equation $Le = r$
- as hard to solve as $Lu = f$
- but e is smooth, can be represented on coarse grid
- solve residual equation on coarse grid $\bar{L}\bar{u} = \bar{f}$

Coarse Grid Equation

- how to construct this coarse grid eqation?
- transfer residual r to coarse grid: restriction $\bar{f} = Rr$
- coarse solve $\bar{L}\bar{u} = \bar{f}$
- transfer $\bar{u}$ to fine grid: prolongation $e = P\bar{u}$
- coarse matrix construction
  - discretise on coarse grid like on fine grid
  - Galerkin product

Coarse Grid Matrix Example

- $n = 8 = 2\bar{n}$, $\bar{n} = 4$

$$L = \frac{1}{8^2} \begin{bmatrix} 2 & -1 & & & & & \\ -1 & 2 & -1 & & & & \\ & -1 & 2 & -1 & & & \\ & & -1 & 2 & -1 & & \\ & & & -1 & 2 & -1 & \\ & & & & -1 & 2 & -1 \\ & & & & & -1 & 2 \end{bmatrix}$$

$$\bar{L} = \frac{1}{4^2} \begin{bmatrix} 2 & -1 & \\ -1 & 2 & -1 \\ & -1 & 2 \end{bmatrix}$$

Galerkin Product Coarse Grid Matrix

- we would like to solve $Le = r$
- we have $e = P\bar{u}$ and $\bar{f} = Rr$
- substitute e: $LP\bar{u} = r$
- multiply by R: $RLP\bar{u} = Rr = \bar{f}$
- take $\bar{L} = RLP$
- more general than discretisation
- used in AMG
- $\bar{L}$ may not have the same properties as L (sparsity, symmetry)

Two-Grid Algorithm

twog$(u^{(0)}, L, f) \rightarrow u^{(3)}$

- $u^{(1)} \leftarrow \text{smooth}(u^{(0)}, L, f, \mu_1)$
- $r \leftarrow f - Lu^{(1)}$
- $\bar{f} \leftarrow Rr$
- solve $\bar{L}\bar{u} = \bar{f}$
- $e = P\bar{u}$
- $u^{(2)} \leftarrow u^{(1)} + e$
- $u^{(3)} \leftarrow \text{smooth}(u^{(1)}, L, f, \mu_2)$

Smoothing

smooth$(u^{(0)}, L, f, \mu) \rightarrow u^{(\mu)}$

- for $\nu = 1, \ldots, \mu$ solve

$$L^+ u^{(\nu)} + L^- u^{(\nu-1)} = f$$

Restriction 1D

- injection
  - $\bar{f}_i = (Rr)_i = r_{2i}$
  - matrix example ($n = 8 = 2\bar{n}$, $\bar{n} = 4$)

$$R = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix}$$

  - stencil notation $R = \begin{bmatrix} 0 & 1 & 0 \end{bmatrix}$
- full weighting
  - $\bar{f}_i = (Rr)_i = \frac{r_{2i-1} + 2r_{2i} + r_{2i+1}}{4}$
  - matrix example ($n = 8 = 2\bar{n}$, $\bar{n} = 4$)

$$R = \frac{1}{4} \begin{bmatrix} 1 & 2 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 2 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 2 & 1 \end{bmatrix}$$

  - stencil notation $R = \frac{1}{4} \begin{bmatrix} 1 & 2 & 1 \end{bmatrix}$

Restriction 2D

- injection
  - $\bar{f}_{i,j} = (Rr)_{i,j} = r_{2i,2j}$
  - matrix notation using 1D injection $R = R_x \otimes R_y$
  - stencil notation $R = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$
- full weighting
  - $\bar{f}_{i,j} = (Rr)_{i,j} =$

    $$\frac{1}{16}\begin{pmatrix} r_{2i-1,2j-1} + 2r_{2i-1,2j} + r_{2i-1,2j+1} + \\ 2r_{2i,2j-1} + 4r_{2i,2j} + 2r_{2i,2j+1} + \\ r_{2i+1,2j-1} + 2r_{2i+1,2j} + r_{2i+1,2j+1} \end{pmatrix}$$
  - matrix notation using 1D full weighting $R = R_x \otimes R_y$
  - stencil notation

    $$R = \frac{1}{16}\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix} = \frac{1}{4}\begin{bmatrix} 1 & 2 & 1 \end{bmatrix} \otimes \frac{1}{4}\begin{bmatrix} 1 \\ 2 \\ 1 \end{bmatrix}$$

Restriction 2D (cont.)

- half weighting
  - $\bar{f}_{i,j} = (Rr)_{i,j} = \frac{r_{2i-1,2j}+r_{2i,2j-1}+4r_{2i,2j}+r_{2i,2j+1}+2r_{2i+1,2j}}{8}$
  - stencil notation $R = \frac{1}{8}\begin{bmatrix} & 1 & \\ 1 & 4 & 1 \\ & 1 & \end{bmatrix} =$

$$\left(\frac{1}{4}\begin{bmatrix}1 & 2 & 1\end{bmatrix} \otimes \begin{bmatrix}0\\1\\0\end{bmatrix} + \begin{bmatrix}0 & 1 & 0\end{bmatrix} \otimes \frac{1}{4}\begin{bmatrix}1\\2\\1\end{bmatrix}\right)/2$$

Prolongation 1D

- linear interpolation
  $e = P\bar{u}$

$$e_{2i} = \bar{u}_i$$
$$e_{2i+1} = \frac{\bar{u}_i + \bar{u}_{i+1}}{2}$$

- stencil notation
  $P = \left] \tfrac{1}{2} \;\; 1 \;\; \tfrac{1}{2} \right[$

- matrix example
  $(n = 8 = 2\bar{n},\ \bar{n} = 4)$

$$P = \begin{bmatrix} \frac{1}{2} & 0 & 0 \\ 1 & 0 & 0 \\ \frac{1}{2} & \frac{1}{2} & 0 \\ 0 & 1 & 0 \\ 0 & \frac{1}{2} & \frac{1}{2} \\ 0 & 0 & 1 \\ 0 & 0 & \frac{1}{2} \end{bmatrix}$$

Prolongation 2D

- bilinear interpolation $e = P\bar{u}$

$$
\begin{aligned}
e_{2i,2j} &= \bar{u}_{i,j} \\
e_{2i,2j+1} &= \frac{\bar{u}_{i,j} + \bar{u}_{i,j+1}}{2} \\
e_{2i+1,2j} &= \frac{\bar{u}_{i,j} + \bar{u}_{i+1,j}}{2} \\
e_{2i+1,2j+1} &= \frac{\bar{u}_{i,j} + \bar{u}_{i,j+1} + \bar{u}_{i+1,j} + \bar{u}_{i+1,j+1}}{4}
\end{aligned}
$$

- stencil notation

$$
P = \frac{1}{4} \left] \begin{array}{ccc} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{array} \right[
$$

Relation between Restriction and Prolongation

- 1D: linear interpolation and full weighting

$$P = 2R^T$$

- 2D: bilinear interpolation and full weighting

$$P = 4R^T$$

- important, e.g., preserve symmetry on coarse grids

Multigrid Algorithm

- coarse grid eqation $\bar{L}\bar{u} = \bar{f}$
- same structure as original equation $Lu = f$
- apply algorithm recursively $\rightarrow$ multigrid

$\mathrm{mg}(u^{(0)}, L, f) \rightarrow u^{(3)}$

- if coarsest: solve $Lu = f$, otherwise
- $u^{(1)} \leftarrow \mathrm{smooth}(u^{(0)}, L, f, \mu_1)$
- $\bar{f} \leftarrow R(f - Lu^{(1)})$
- $\bar{u} \leftarrow 0$
- γ times: $\bar{u} \leftarrow \mathrm{mg}(\bar{u}, \bar{L}, \bar{f})$
- $u^{(2)} \leftarrow u^{(1)} + P\bar{u}$
- $u^{(3)} \leftarrow \mathrm{smooth}(u^{(1)}, L, f, \mu_2)$

Multigrid Cycles

- V-cycle: $\gamma = 1$
- W-cycle: $\gamma = 2$
- F-cycle: between V and W

Avanced Topics

- Neumann boundary conditions
- full multigrid (FMG)
- multigrid for finite element discretisation
- more general domains
- variable coefficients
- adaptive methods
- nonlinear equations
- anisotropic equations/stretched grids
- multigrid as preconditioner for Krylov subspace methos (CG, GMRES)
- algebraic multigrid (AMG)
- time-dependent problems

References

- *A Multigrid Tutorial*, Briggs et. al
- *Multigrid*, Trottenberg, Oosterlee and Schüller
- *An Introduction to Multigrid Methods*, Wesseling
- *Multigrid Methods and Applications*, Hackbusch
- `mgnet.org`: papers, software, news
- packages with MG component: petsc, trilinos, hypre

Checks for Implementation

- tests where you know what the results should be
  - restriction and interpolation of constants, linear function
  - choose function u, find function f, solve and compare, should be exact for constant, linear or quadratic function, error should behave as $O(\Delta x^2)$, $\Delta x \to 0$ otherwise
  - choose vector u, find vector f, solve and compare, should be exact
- fixed point iteration: iterate Jacobi, GS, MG starting from exact solution, shouldn't change
- check convergence in many norms $\|\cdot\|_1$, $\|\cdot\|_2$, $\|\cdot\|_\infty$, $\|\cdot\|_L$
- plot solution, residual, error
- check boundaries