

A Parallel Multigrid Solver for Beam Dynamics

Yves Ineichen (ETH Zurich)

Master Thesis

Supervisor: Andreas Adelmann (PSI)

Supervising Professor: Peter Arbenz (ETH)

14th August 2008



Outline

- 1 Motivation
- 2 Solver
- 3 Boundary Conditions
- 4 Results
- 5 Summary

Outline

- 1 Motivation
- 2 Solver
- 3 Boundary Conditions
- 4 Results
- 5 Summary

Motivation

Space-Charge Calculation in OPAL I

Space-Charge in the Electrostatic Approximation

Whenever we have a number of moving charged particles:

- electric fields caused by Coulomb repulsion are present
- magnetic fields arising from the moving particles

Both effects acts as **forces** on to the particles!

Motivation

Space-Charge Calculation in OPAL I

Space-Charge in the Electrostatic Approximation

Whenever we have a number of moving charged particles:

- electric fields caused by Coulomb repulsion are present
- magnetic fields arising from the moving particles

Both effects acts as **forces** on to the particles!

Express the Coulomb potential ϕ in terms of charge densities ρ (proportional to the particle density). The electric field can be expressed by

$$\mathbf{E} = -\nabla\phi.$$

The arising Poisson's equation for the electrostatic potential has the form

$$\nabla^2\phi = -\frac{\rho}{\varepsilon_0}.$$

The Magnetic field can be calculated from the electric field by the Lorentz transformation.

Motivation

Space-Charge Calculation in OPAL II

Particle-in-cell (PIC) Method in N-body Simulations

- interpolate individual particle charges to the grid
- solve the Poisson equation on the mesh in a Lorentz frame
- values of the potential at the location of individual particles is interpolated from the potential at the grid points
- typically faster $\mathcal{O}(n \log n)$ than Particle-Particle method $\mathcal{O}(n^2)$

Motivation

Space-Charge Calculation in OPAL II

Particle-in-cell (PIC) Method in N-body Simulations

- interpolate individual particle charges to the grid
- solve the Poisson equation on the mesh in a Lorentz frame
- values of the potential at the location of individual particles is interpolated from the potential at the grid points
- typically faster $\mathcal{O}(n \log n)$ than Particle-Particle method $\mathcal{O}(n^2)$

We apply a second order finite difference scheme which leads to a set of linear equations

$$\mathbf{Ax} = \mathbf{b},$$

where \mathbf{b} denotes the charge densities on the mesh.

Motivation

Space-Charge Calculation in OPAL III

Current space-charge calculation in OPAL

- FFT based direct solver: convolution with Green's function
- rectangular domain with open and periodic boundary conditions

Motivation

Space-Charge Calculation in OPAL III

Current space-charge calculation in OPAL

- FFT based direct solver: convolution with Green's function
- rectangular domain with open and periodic boundary conditions

Goals: improvements

- solve anisotropic electrostatic Poisson PDE with an iterative solver
- irregular domain with “exact” boundary conditions

Motivation

Space-Charge Calculation in OPAL III

Current space-charge calculation in OPAL

- FFT based direct solver: convolution with Green's function
- rectangular domain with open and periodic boundary conditions

Goals: improvements

- solve anisotropic electrostatic Poisson PDE with an iterative solver
 - irregular domain with “exact” boundary conditions
- 1 implement a preconditioned iterative solver
 - 2 incorporate this solver in OPAL
 - 3 handle the irregular boundary points
 - 4 investigate and implement ways to enable OPAL to use exact beam pipe geometries

Outline

- 1 Motivation
- 2 Solver
- 3 Boundary Conditions
- 4 Results
- 5 Summary

Multigrid Algorithm

Multigrid V-Cycle Algorithm

```
1: procedure MultiGridSolve( $A_l, b_l, x_l, l$ )
2: if  $l = \text{maxLevel}-1$  then
3:   DirectSolve  $A_l x_l = b_l$ 
4: else
5:    $x_l \leftarrow S_l^{\text{pre}}(A_l, b_l, 0)$ 
6:    $r_l \leftarrow b_l - A_l x_l$  {calculate residual}
7:    $b_{l+1} \leftarrow R_l r_l$  {Restriction}
8:    $v_{l+1} \leftarrow 0$ 
9:   MultiGridSolve( $A_{l+1}, b_{l+1}, v_{l+1}, l+1$ )
10:   $x_l \leftarrow x_l + P_l v_{l+1}$  {coarse grid correction}
11:   $x_l \leftarrow S_l^{\text{post}}(A_l, b_l, x_l)$ 
12: end if
13: end procedure
```

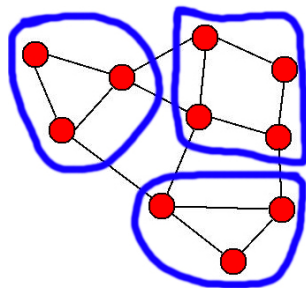
Prolongation Operator (1/2)

Smoothed Aggregation: The Grid Transfer Operator

- 1 discretization matrix A_l is converted into a graph G_l
- 2 assign each vertex of G_l is assigned to one aggregate
- 3 the tentative prolongation operator matrix is formed
 - matrix rows correspond to vertices
 - matrix columns to aggregates

$$p_{i,j} = \begin{cases} 1 & \text{if } i^{\text{th}} \text{ vertex in } j^{\text{th}} \text{ aggregate} \\ 0 & \text{otherwise} \end{cases}$$

- 4 improve robustness by smoothing the tentative prolongation operator



clustering vertices into aggregates

Prolongation Operator (2/2)

Smoothed Aggregation: Parameters

We employed a “decoupled” aggregation scheme:

- aggregates of size 3^d in d dimensions ($3 \times 3 \times 3 = 27$)
- each processor aggregate its portion of the grid
- many aggregates near inter-processor boundaries with non-optimal size
- number of processors determine number of aggregates: coarse problem may still be to big
- number of vertices is substantially reduced in every coarsening step
- typical operator complexity for decoupled aggregation is 1.5

Smoothing Operator and Coarse Level Solver

A Chebyshev polynomial smoother is used as pre- and postsmoother:

- polynomial smoothers perform well for parallel solvers (R. Tuminaro and C. Tong, 2000)
- profit of architecture optimized matrix vector products
- polynomial smoothers are easy to parallelize

Smoothing Operator and Coarse Level Solver

A Chebyshev polynomial smoother is used as pre- and postsmoother:

- polynomial smoothers perform well for parallel solvers (R. Tuminaro and C. Tong, 2000)
- profit of architecture optimized matrix vector products
- polynomial smoothers are easy to parallelize

We chose to use a LU based solver as direct coarse level solver

Implementation (1/3)

For preconditioner setup and iterative solver we used TRILINOS:

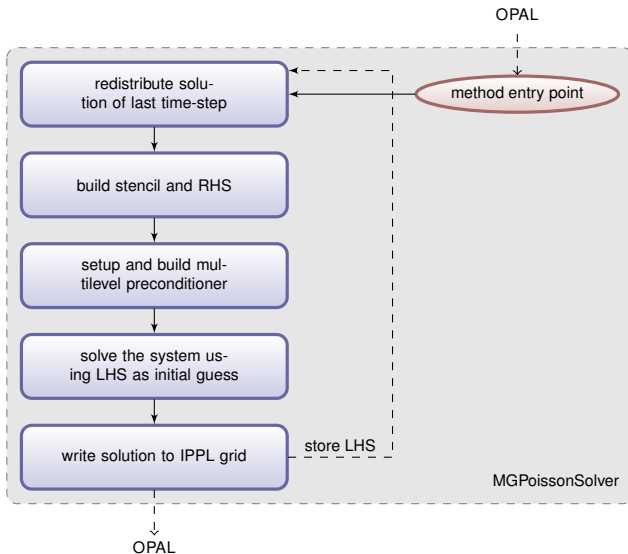
- ML: smoothed aggregation based AMG preconditioner
- EPETRA: distributed matrices and vectors
- AMESOS: direct coarse level solver
- AZTECOO: iterative solver

OPAL in conjunction with Independent Parallel Particle Layer (IPPL) offers:

- parallel fields
- particle representation
- operators on fields

Implementation (2/3)

Integration in OPAL II: Solver



Implementation (3/3)

Interface between IPPL and EPETRA

IPPL to EPETRA Map

```
1: procedure IPPLToMap3D(localidx)
2:   idx  $\leftarrow$  0
3:   for all localidx.x do
4:     for all localidx.y do
5:       for all localidx.z do
6:         MyGlobalElements[idx]  $\leftarrow$  bp $\rightarrow$ getIdx(x,y,z)
7:         idx  $\leftarrow$  idx + 1
8:       end for
9:     end for
10:  end for
11:  return new Epetra_Map(-1, NumMyElements, &MyGlobalElements[0], 0,
    Comm)
12: end procedure
```

Outline

- 1 Motivation
- 2 Solver
- 3 Boundary Conditions**
- 4 Results
- 5 Summary

Boundary Condition Problem

Boundary Problem

Lets denote $\Omega \subset \mathbb{R}^3$ the simply connected computational domain and $\Gamma = \Gamma_1 \cup \Gamma_2$, the boundary of Ω ($\Gamma_1 \cap \Gamma_2 = \emptyset$). As mentioned we solve:

$$\nabla^2 \phi = -\frac{\rho}{\epsilon_0}, \text{ in } \Omega \subset \mathbb{R}^3,$$

$$\phi = 0, \text{ on } \Gamma_1$$

$$\frac{\partial \phi}{\partial \vec{n}} + \frac{1}{d} \phi = 0, \text{ on } \Gamma_2,$$

with ϵ_0 denotes the dielectric constant and d is the distance of the bunches centroid to the boundary.

Boundary Condition Problem

Boundary Problem

Lets denote $\Omega \subset \mathbb{R}^3$ the simply connected computational domain and $\Gamma = \Gamma_1 \cup \Gamma_2$, the boundary of Ω ($\Gamma_1 \cap \Gamma_2 = \emptyset$). As mentioned we solve:

$$\nabla^2 \phi = -\frac{\rho}{\epsilon_0}, \text{ in } \Omega \subset \mathbb{R}^3,$$

$$\phi = 0, \text{ on } \Gamma_1$$

$$\frac{\partial \phi}{\partial \vec{n}} + \frac{1}{d} \phi = 0, \text{ on } \Gamma_2,$$

with ϵ_0 denotes the dielectric constant and d is the distance of the bunches centroid to the boundary.

- ① Γ_1 is the surface of an elliptic beam-pipe
- ② Γ_1 is the surface of an arbitrary beam-pipe element

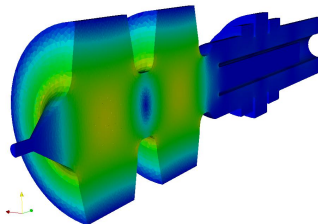
Using Real Beam-Pipe Geometries

Components

- arbitrary bounded domains are specified in files
- OPAL imports triangulated surface mesh
- efficient intersection of grid with surface mesh
- discretization approach

Motivation

- more accurate simulation of space-charges



2 Frequency LEG Cavity (Based on the design of J.-Y. Raguin (PSI))

Discretization: Irregular Domains (1/2)

$O(h)$ Approach

The key idea of this approach is to only consider grid points inside the domain neglecting the distance to the domain boundary:

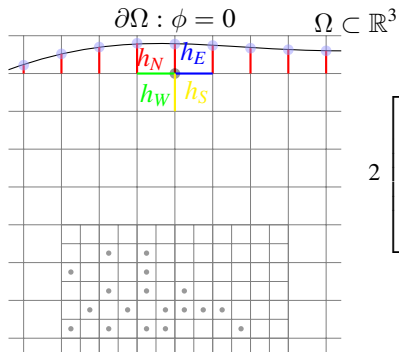
$$(h_w^{-1} + h_s^{-1} + h_e^{-1} + h_n^{-1})u_p - h_n^{-1}u_n - h_w^{-1}u_w - h_s^{-1}u_s - h_e^{-1}\underbrace{u_e}_{=0} = f_p$$

Properties

- the resulting discretization matrix is symmetric
- $O(h)$ accurate

Discretization: Irregular Domains (2/2)

Shortley-Weller approximation



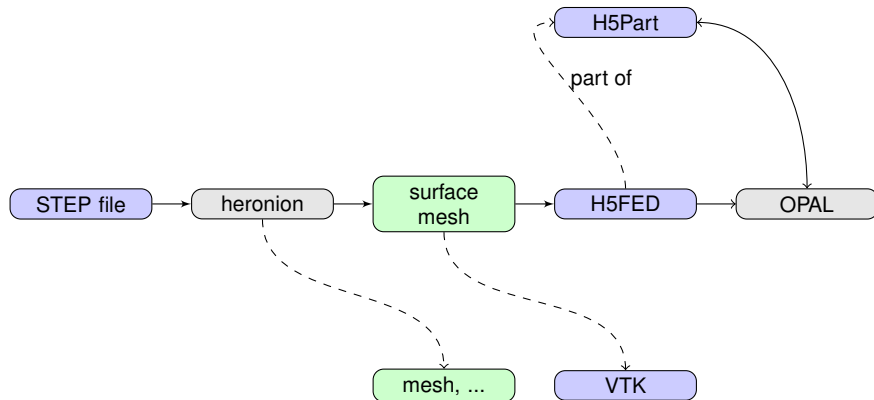
$$2 \left[\begin{array}{ccc} \frac{a}{h_W(h_W + h_E)} & -\frac{\frac{b}{h_N(h_N + h_S)}}{\frac{a}{h_W h_E} - \frac{b}{h_S h_N}} & \frac{a}{h_E(h_W + h_E)} \end{array} \right]_h$$

Properties

- the resulting discretization matrix is non-symmetric for boundary points
- $O(h^2)$ accurate

Implementation (1/2)

Importing geometries in OPAL

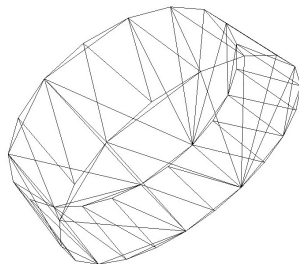


in collaboration with B. Oswald and A. Gsell (AIT)

Implementation (2/2)

Setup Phase

- extended HERONION to dump H5Fed surface mesh
- OPAL imports H5Fed files (serial): m triangles and v vertices
- efficient intersection of grid-lines with triangular surface mesh (T. Moeller and B. Trumbore (1997)):
 - arbitrary domain: $O(m(n_x + n_y + local_z))$
 - elliptic domain: $O(n_x + n_y)$
- building index table
 - arbitrary domain: $O(n_x n_y local_z)$
 - elliptic domain: $O(n_x n_y)$



Outline

- 1 Motivation
- 2 Solver
- 3 Boundary Conditions
- 4 Results**
- 5 Summary

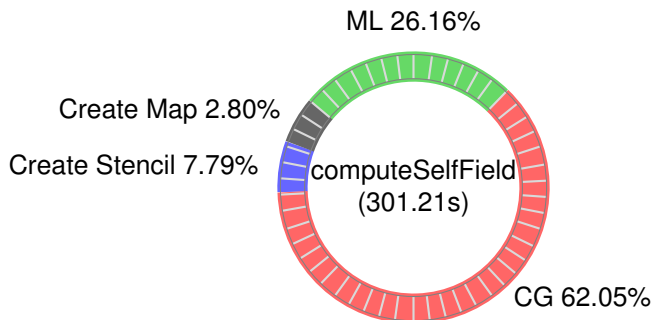
Buin: Cray XT4 cluster at the CSCS in Manno

- 468 AMD dual core Opteron at 2.6 GHz
- 936 GB DDR RAM
- 30 TB Disk
- 7.6 GB/s interconnect bandwidth

Environment

Buin: Cray XT4 cluster at the CSCS in Manno

- 468 AMD dual core Opteron at 2.6 GHz
- 936 GB DDR RAM
- 30 TB Disk
- 7.6 GB/s interconnect bandwidth



Validation

For validation purposes we defined a to the z axis, rotation symmetric potential function and calculated the analytical solution.

Arbitrary domains: tested with a triangulated surface mesh of a cylinder with the same potential function.

Validation

For validation purposes we defined a to the z axis, rotation symmetric potential function and calculated the analytical solution.

Arbitrary domains: tested with a triangulated surface mesh of a cylinder with the same potential function.

Results: L_2 norms of the error:

grid size	elliptic	arbitrary
$20 \times 20 \times 20$	0.1158	0.1145
$40 \times 40 \times 40$	0.0846	0.0838
$80 \times 80 \times 80$	0.0765	0.0755

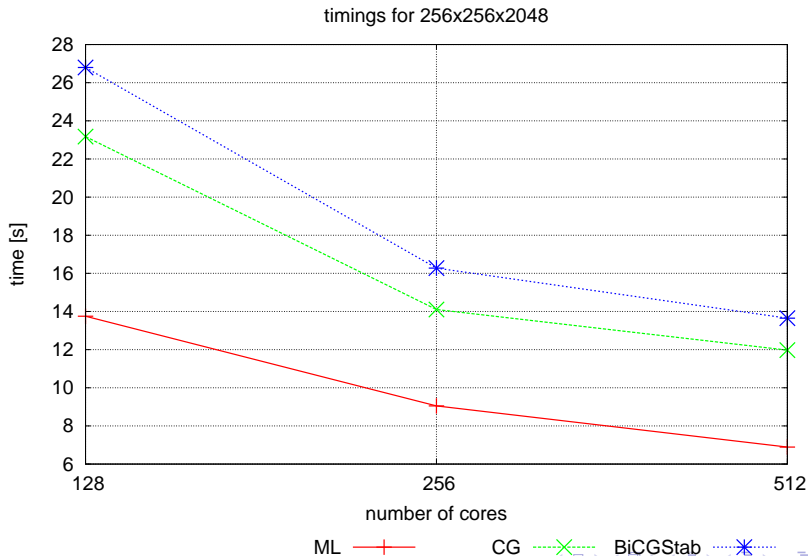
Convergence

On a $256 \times 256 \times 2048$ grid:

discretization	number of nodes	iter. for CG	iter. for BiCGStab
$O(h)$	128	12	-
$O(h)$	256	12	-
$O(h)$	512	11	-
SW	128	17	10
SW	256	17	10
SW	512	18	11

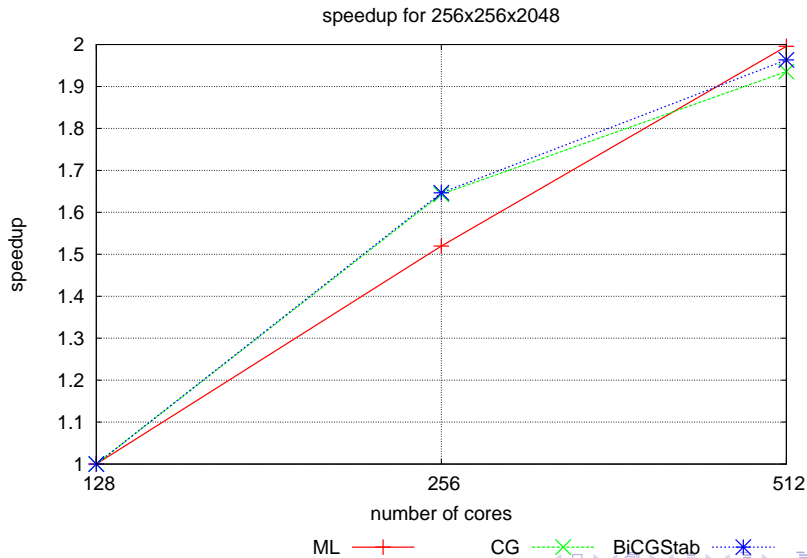
Parallel Efficiency (1/2)

Timings



Parallel Efficiency (2/2)

Speedup



Outline

- 1 Motivation
- 2 Solver
- 3 Boundary Conditions
- 4 Results
- 5 Summary

Improved Space-Charge Solver:

- is discretized with finite differences
- smoothed aggregation based algebraic Multigrid preconditioner
- fully integrated into OPAL
- reuse solutions of previous time-steps

Improved Space-Charge Solver:

- is discretized with finite differences
- smoothed aggregation based algebraic Multigrid preconditioner
- fully integrated into OPAL
- reuse solutions of previous time-steps

Improved Boundary Conditions:

- Shortley-Weller approximation
- elliptic beam-pipe domain
- arbitrary domains based on real geometries of beam-pipe elements
- beam-pipe geometries can be processed and imported

Problems

- OPAL on Buin cluster in Manno: stand-alone solver
- ACML

Problems and Further Work

Problems

- OPAL on Buin cluster in Manno: stand-alone solver
- ACML

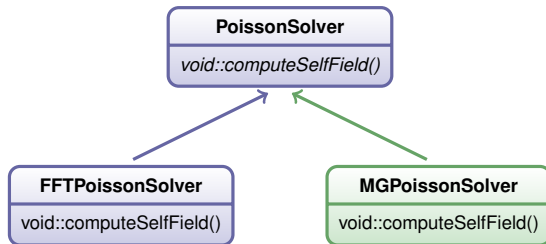
Further Work

- parallelize in x and y direction
- validation of arbitrary domains against complex geometries
- investigate in non-symmetry solver
- improve aggregation process
- adaptive mesh refinement (AMR)

Backup

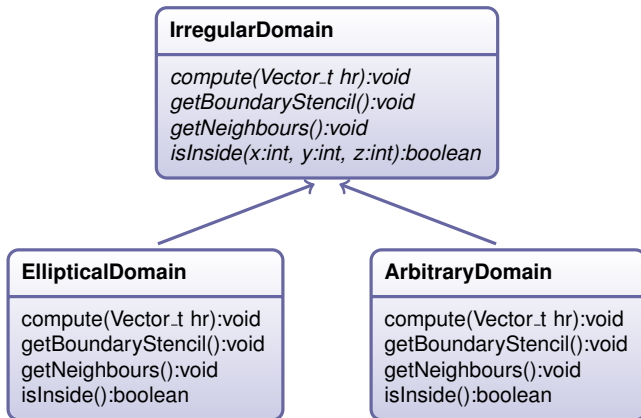
Implementation (2/4)

Integration in OPAL I: Class Hierarchy

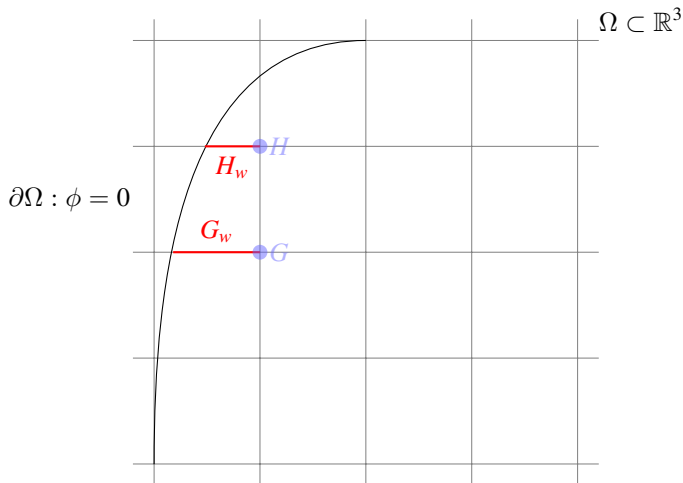


Implementation (1/3)

Class Diagram



SW: non-symmetries



Grid Operators

AMG: smoothed aggregation

Operate on directly on (linear sparse) algebraic equations:

$$\sum_j a_{ij}^h x_j^h = b_i^h$$

- replace "grid" with "variables"
- coarse level equations are generated without the use of any geometry
- no coarse level grids have to be generated or stored
- good preconditioner: works on all error components (in contrast to level-one preconditioner)

SA restrict operator:

$$I_H^h = (I_h - \omega D_h^{-1} A_h^f) \hat{I}_H^h$$

Multigrid Theory (1/2)

Motivation

Important Observations

- Some classical iterative methods (i.e. Gauss Seidel) have a smoothing effect on the error of any approximation for discrete elliptic problems.
- A smooth error can be well approximated on a coarse grid. This coarse grid has considerably fewer grid points and is therefore cheaper to solve.

From this two observations a Two-Grid can be deduced:

- 1 apply smoother
- 2 restrict to a grid with considerably fewer grid points (coarse)
- 3 solve
- 4 interpolate back to the fine grid
- 5 compute a new approximation

Multigrid Theory (2/2)

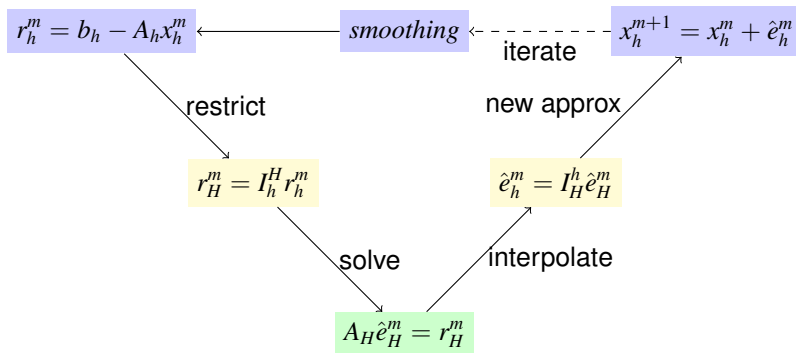
The Two-Grid: Smoothed Coarse Grid Correction

The discretized system is solved by a Two-Grid:

$$A\mathbf{x} = \mathbf{b}$$

$$e_h^m = x_h - x_h^m, r_h^m = b_h - A_h x_h^m$$

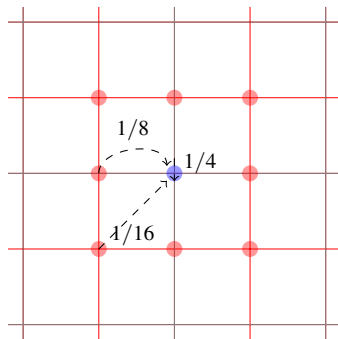
$$r_h^m = A_h e_h^m$$



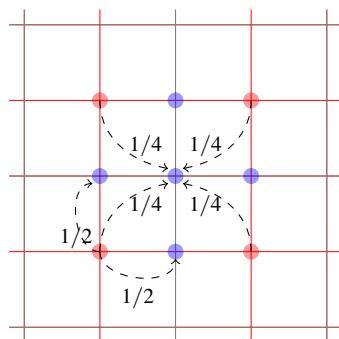
Grid Operators

Geometric Multigrid

restriction

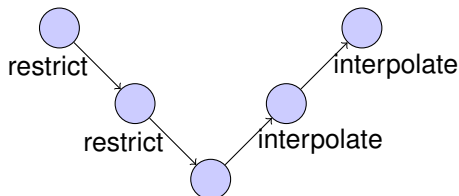
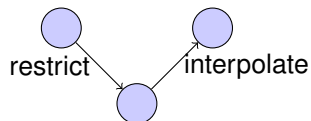


bilinear interpolation



Multigrid

from Two-Grid to Multigrid



Depending on how the recursion is coded, some variants of the V-cycle can be produced.

- grid-independence convergence
- iterative solver: reuse information
- $\mathcal{O}(n)$ algorithm

Anisotropy is handled in the discretized problem