A Fast Parallel Poisson Solver on Irregular Domains Applied to Beam Dynamic Simulations

Yves Ineichen*,**
Andreas Adelmann**
Peter Arbenz*

*Federal Institute of Technology
Department of Computer Science
Universitaetsstrasse 6, CH-8092 Zuerich, Switzerland

**Paul Scherrer Institute
Accelerator Modelling and Advanced Simulations
CH-5234 Villigen, Switzerland

31st August 2009

Outline

Outline

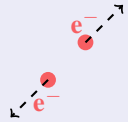
Self Force Calculation

Self Forces 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 act as forces on to the particles!



Express the Coulomb potential ϕ in terms of charge densities ρ (proportional to the particle density)

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

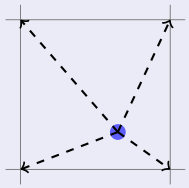
$$\mathbf{E} = -\nabla \phi.$$

The magnetic field can be calculated from the electric field (Lorentz transformation).

Self Force Calculation

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
- typically faster $\mathcal{O}(n \log n)$ than Particle-Particle method $\mathcal{O}(n^2)$



- Finite difference scheme leading to a set of linear equations

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

  b denotes the charge densities on the mesh
- Integrated into code tracking relativistic particles in time

OPAL in a Nutshell

OPAL is a tool for charged-particle optics in large accelerator structures and beam lines including 3D space charge

Some of the features

- OPAL is built from the ground up as a parallel application exemplifying the fact that HPC (High Performance Computing) is the third leg of science, complementing theory and the experiment
- OPAL runs on your laptop as well as on the largest HPC clusters
- OPAL uses the MAD language with extensions
- OPAL (and all other used frameworks) are written in C++ using OO-techniques, hence OPAL is very easy to extend.
- Documentation is taken very seriously at both levels: source code and user manual (`http://amas.web.psi.ch/docs/index.html`)

Wednesday 13:30 A. Adelmann: OPAL Design, Implementation and Application

Motivation

State of the art space charge calculation as implemented in OPAL

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

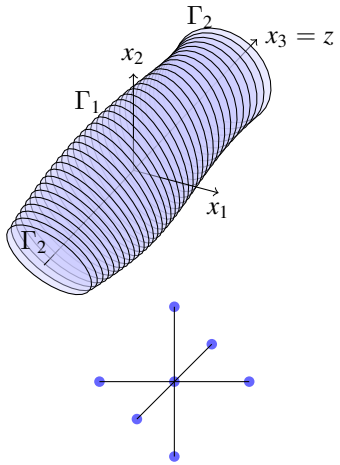
A New Iterative Solver

- solve anisotropic electrostatic Poisson PDE with an iterative solver
- reuse information available from previous time steps
- achieving good parallel efficiency
- **irregular** domain with "exact" boundary conditions
- easy to specify boundary surface
- P. McCorquodale, P. Colella, D. P. Grote, J.-L. Vay, J. Comp. Phys., 2004

Outline

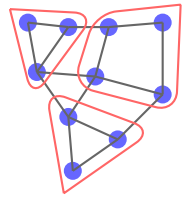
Solver in a Nutshell

- second order finite difference scheme
- standard 7 point stencil (3D) on Cartesian grid
- preconditioned CG iterative solver
- algebraic multigrid preconditioner (using smoothed aggregation)



AMG Parameters

- "decoupled" aggregation scheme: aggregates of size $3 \times 3 \times 3$
  - each processor aggregate its portion of the grid
  - many aggregates near inter-processor boundaries with non-optimal size
  - number of vertices is substantially reduced in every coarsening step



clustering vertices into aggregates

- Chebyshev polynomial pre and postsmoothers perform well for parallel solvers (M. Adams, M. Brezina, J. Hu, R. Tuminaro, J. Comp. Phys., 2003)
- LU based direct coarse level solver

AMG performance critically depends on choice of parameters!

Implementation (1/2)

For preconditioner setup and iterative solver we used TRILINOS:

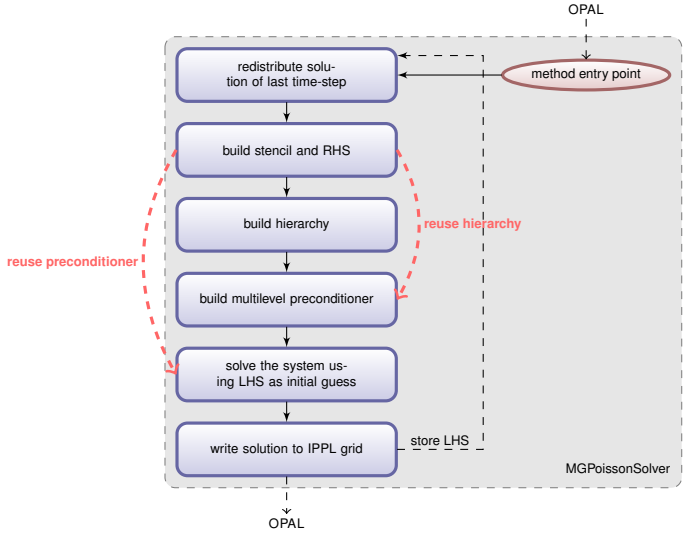
- EPETRA: distributed matrices and vectors
- AMESOS: direct coarse level solver
- AZTECOO: iterative solver
- ML: smoothed aggregation based AMG preconditioner

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

- parallel fields
- particle representation
- operators on fields

Implementation (2/2)

Integrating the Solver in OPAL



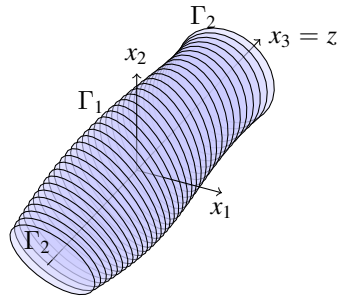
Outline

Boundary Conditions

Boundary Problem

$$\nabla^2\phi = -\frac{\rho}{\epsilon_0}, \text{ in } \Omega \subset \mathbb{R}^3,$$
$$\phi = 0, \text{ on } \Gamma_1$$
$$\frac{\partial\phi}{\partial\vec{n}} + \frac{1}{d}\phi = 0, \text{ on } \Gamma_2$$

- $\Omega \subset \mathbb{R}^3$: simply connected computational domain
- ϵ_0: the dielectric constant
- $\Gamma = \Gamma_1 \cup \Gamma_2$: boundary of Ω
- d: distance of bunch centroid to the boundary



Γ_1 is the surface of an

1. elliptic beam-pipe
2. arbitrary beam-pipe element

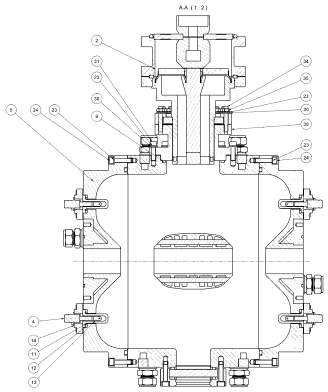
Using Real Beam-Pipe Geometries

Components

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

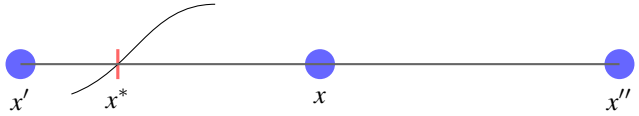
Motivation

- more accurate simulation of space-charges



Super Buncher

Extrapolation at Boundary



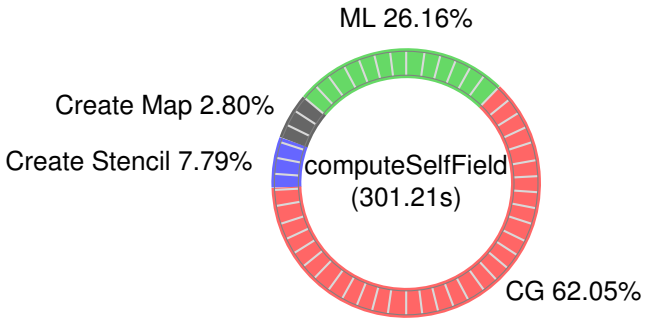
1. *Constant extrapolation*: $u(x') = u(x^*)$ and $x^* \in \Gamma_1$
2. *Linear extrapolation*: $u(x')$ is obtained by means of $u(x)$ and $u(x^*)$
3. *Quadratic extrapolation* (Shortley-Weller approximation): $u(x')$ is obtained by quadratic interpolation of $u(x)$, $u(x'')$, and $u(x^*)$
   → non-symmetric stencil

Outline

Environment

Buin: Cray XT4 cluster at the CSCS in Manno (Switzerland)

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



Validation of the Solver

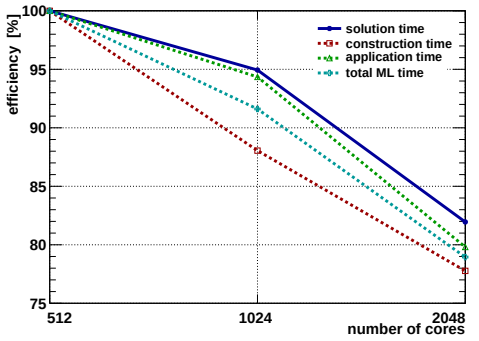
For validation purposes we defined an along the z axis axi-symmetric potential function and calculated the analytical solution.

h	$\|\|e_h\|\|_2$	r	$\|\|e_h\|\|_\infty$	r
$1/64$	2.162×10^{-3}	—	7.647×10^{-3}	—
$1/128$	1.240×10^{-3}	0.80	4.153×10^{-3}	0.88
$1/64$	2.460×10^{-5}	—	6.020×10^{-5}	—
$1/128$	6.226×10^{-6}	1.98	1.437×10^{-5}	2.07
$1/64$	5.581×10^{-6}	—	1.689×10^{-5}	—
$1/128$	1.384×10^{-7}	2.01	4.550×10^{-6}	1.89

The convergence rate r is defined by

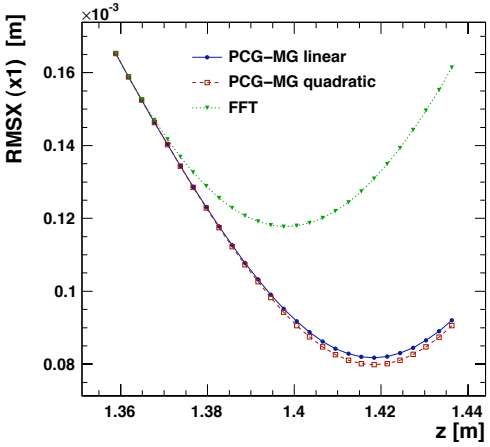
$$r = \log_2 \left(\frac{||e_{2h}||}{||e_h||} \right)$$

Parallel Efficiency



- obtained for a tube embedded in a $1024 \times 1024 \times 1024$ grid
- construction phase is performing the worst with an efficiency of 73%
- influence of problem size on the low performance of the aggregation in ML

Impact on Physics of OPAL Simulations



- shift of the beam size minimum (waist) towards larger z values
- a smaller minimum $\longrightarrow$ self forces are larger when considering the beam pipe
- beam pipe radius is an important optimization quantity

Outline

Summary

- smoothed aggregation based algebraic Multigrid preconditioned CG
- implemented 3 extrapolation schemes at boundary intersection
- non-symmetric equations resulting from quadratic boundary treatment converge well with PCG
- elliptic and arbitrary domains based on real geometries
- reducing time to solution (20 and 40%) by reusing hierarchy or preconditioner
- compared to FFT more flexibilities for only a small performance loss
- attaining good parallel efficiency: 73% for the worst performing phase
- considerable impact on physics (e.g. for narrow beam pipes)

http://arxiv.org/abs/0907.4863 and submitted to JCP

Further Work

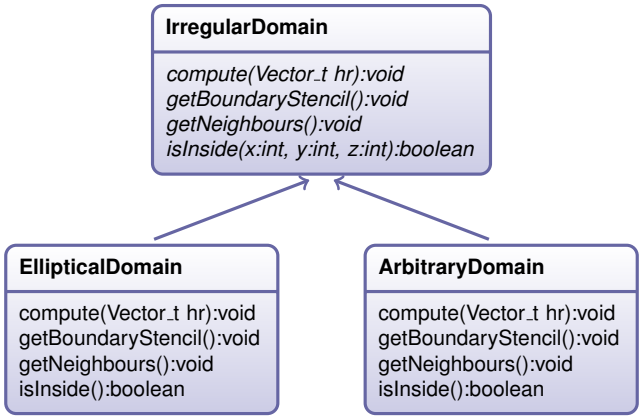
- validation of arbitrary domains against complex geometries
- adaptive mesh refinement (AMR)
- overcome Trilinos global index 32 bit integer size limitation

Backup

Backup

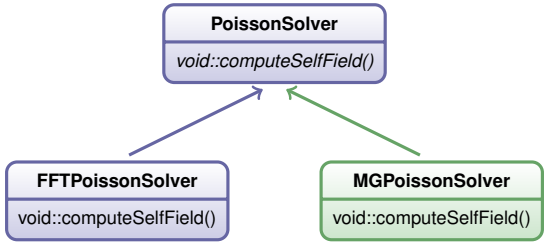
Implementation (1/3)

Class Diagram



Implementation (2/4)

Integration in OPAL I: Class Hierarchy



Implementation (3/3)

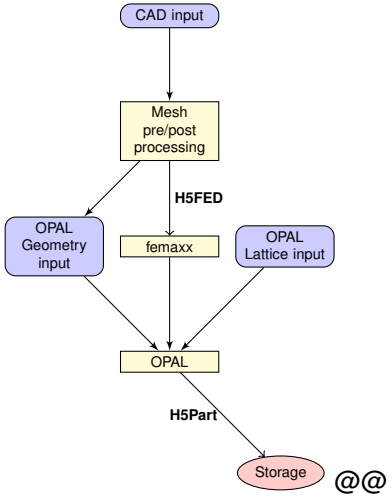
Interface between IPPL and EPETRA

IPPL to EPETRA Map

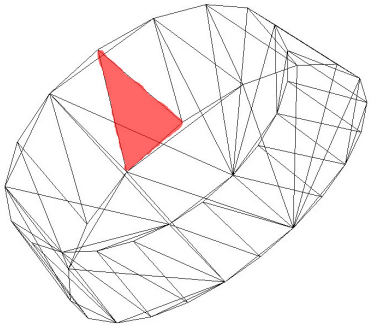
```
1:  procedure IPPLToMap3D(localidx)
2:  idx ← 0
3:  for all localidx.x do
4:     for all localidx.y do
5:        for all localidx.z do
6:           MyGlobalElements[idx] ← bp→getIdx(x,y,z)
7:           idx ← idx + 1
8:        end for
9:     end for
10: end for
11: return  new Epetra_Map(-1, NumMyElements,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

Implementation

Importing geometries in OPAL



@@

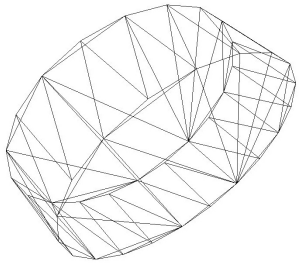


Efficient Intersection of Grid-Lines with Triangular Surface Mesh (T. Moeller, B. Trumbore, 1997)

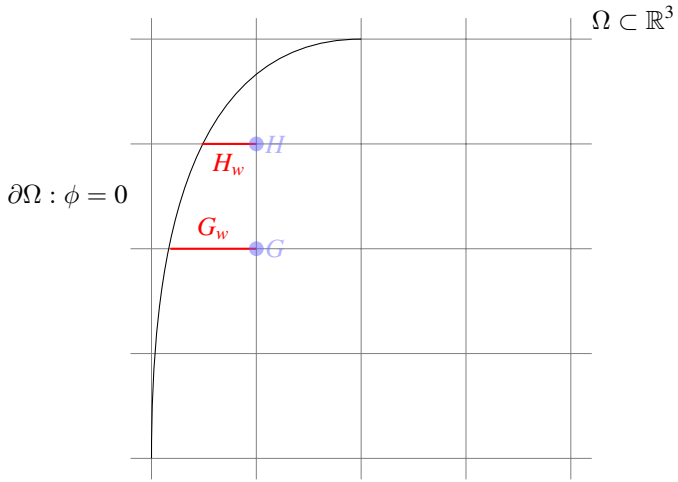
Implementation (2/2)

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: $O(n_x n_y)$



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:

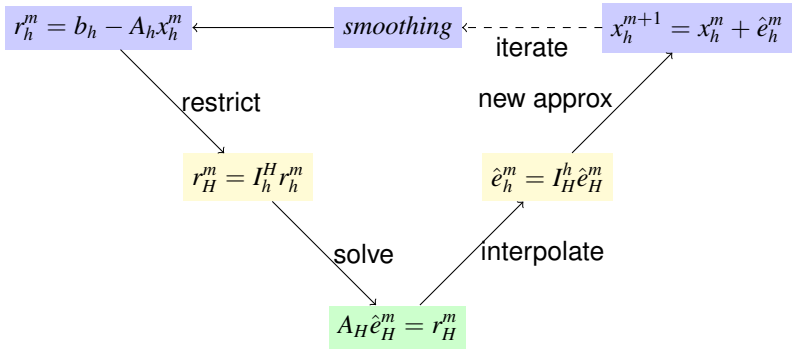
1. apply smoother
2. restrict to a grid with considerably fewer grid points (coarse)
3. solve
4. interpolate back to the fine grid
5. compute a new approximation

Multigrid Theory (2/2)

The Two-Grid: Smoothed Coarse Grid Correction

The discretized system is solved by a Two-Grid:

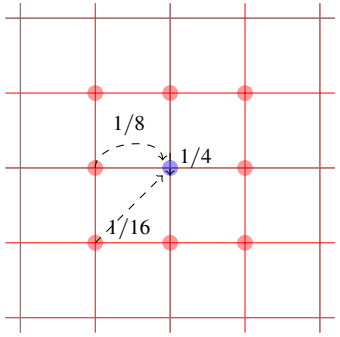
$$A\mathbf{x} = \mathbf{b}$$
$$e_h^m = x_h - x_h^m,\ r_h^m = b_h - A_h x_h^m$$
$$r_h^m = A_h e_h^m$$



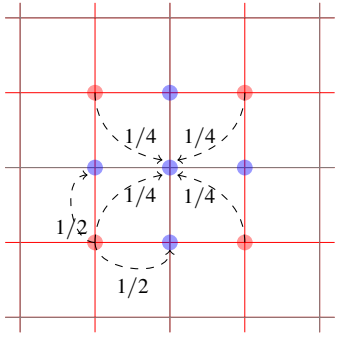
Grid Operators

Geometric Multigrid

restriction

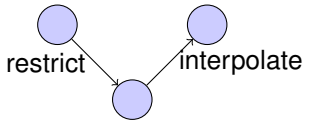


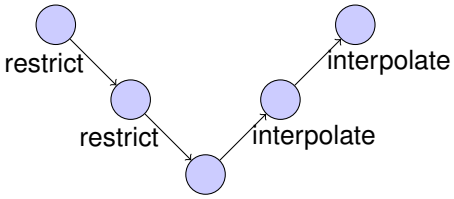
bilinear interpolation



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

Multigrid Algorithm

Mutligrid V-Cycle Algorithm

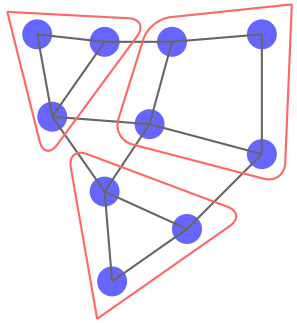
1: **procedure** MultiGridSolve(A_l, b_l, x_l, l)
2: **if** l = maxLevel-1 **then**
3: DirectSolve $A_l\mathbf{x}_l = \mathbf{b}_l$
4: **else**
5: $\mathbf{x}_l \leftarrow S_l^{pre}(A_l, \mathbf{b}_l, 0)$
6: $\mathbf{r}_l \leftarrow \mathbf{b}_l$ - $A_l\mathbf{x}_l$ {calculate residual}
7: $\mathbf{b}_{l+1} \leftarrow R_l\mathbf{r}_l$ {Restriction}
8: $\mathbf{v}_{l+1} \leftarrow \mathbf{0}$
9: MultiGridSolve(A_{l+1}, $\mathbf{b}_{l+1}$, $\mathbf{v}_{l+1}$, $l+1$)
10: $\mathbf{x}_l \leftarrow \mathbf{x}_l + P_l\mathbf{v}_{l+1}$ {coarse grid correction}
11: $\mathbf{x}_l \leftarrow S_l^{post}(A_l, \mathbf{b}_l, \mathbf{x}_l)$
12: **end if**
13: **end procedure**

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

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

4. improve robustness by smoothing the tentative prolongation operator



clustering vertices into aggregates

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:

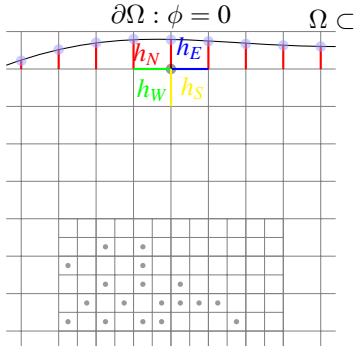
$$(h_w^{-1} + h_s^{-1} + h_e^{-1} + h_n^{-1})u_p - h_n^{-1}u_n - h_w^{-1}u_w - h_s^{-1}u_s - h_e^{-1}\underbrace{u_e}_{=0} = f_p$$

Properties

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

Discretization: Irregular Domains (2/2)

Shortley-Weller approximation



$$2\begin{bmatrix} \frac{a}{h_W(h_W+h_E)} & -\frac{b}{h_N(h_N+h_S)} - \frac{a}{h_w h_E} - \frac{b}{h_S h_N} - \frac{b}{h_S(h_N+h_S)} & \frac{a}{h_E(h_W+h_E)} \end{bmatrix}_h$$

Properties

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