A NOVEL MASSIVE PARALLEL POSSION SOLVER FOR BEAM DYNAMICS

A. Adelmann, C. Pflaum [1]

[1] University Erlangen

We present a novel parallel Poisson solver based on semi-unstructured grids, a finite element discretization (FEM) of the three dimensional computational domain. Multigrid is used for solving the resulting linear system of equations. The code is based on expression templates and written in C++. Results for a benchmarking problem are presented.

INTRODUCTION AND MOTIVATION

Many problems in beam dynamics are concerned with calculation of the inter-particles forces for millions of particles. One particular hard example is described in this annual report, see [1]. In order to estimate the interparticle forces in the electrostatic approximation with intrinsic open boundary conditions we have naively to compute the electric field at every particle's position by summing (in continuum) over the contributions from all other $N-1$ particles: using $\vec{q}_{ij} = \vec{q}_i - \vec{q}_j$ and $q_{ij} = |\vec{q}_i - \vec{q}_j|$. This is exact up to numerics, as no spatial discretization is made, but at the price of the highest computational effort, namely $\mathcal{O}(N^2)$. In order to avoid a numerical breakdown at small interparticle distances, a softening parameter ϵ was included:

$$\vec{E}_i = \frac{1}{4\pi\epsilon_0}\sum_{j\neq i} e_j \frac{\vec{q}_{ij}}{(q_{ij}^2+\epsilon^2)^{3/2}}.$$

POISSON SOLVER

Another way to formulate the same problem is to calculate the electric field $\vec{E}$ via the gradient of the scalar potential $\vec{E} = \nabla\phi$. The scalar potential is obtained by solving a Poisson problem. The boundary conditions of this Poisson problem are Dirichlet boundary conditions and, in some applications, additionally periodic boundary conditions. In this paper, for reasons of simplicity, we restrict ourselves to pure Dirichlet boundary conditions, since the additional periodic boundary condition are of no numerical difficulty. Such a Poisson problem can be described as follows:

$$-\triangle u = f \quad \text{on } \Omega, \tag{1}$$
$$u = 0 \quad \text{on } \partial\Omega, \tag{2}$$

where $\Omega \subset]0,1[^3$ is a bounded domain.

In case of the pure rectangular domain $\Omega =]0,1[^3$, one can apply trilinear finite elements on the discretization grid

$$\Omega_h = \{(ih, jh, kh) \mid i,j,k = 0,1,\cdots,N = 1/h\},$$

where $N = 2^n, n \in \mathcal{N}$, is the solution u_h of the difference equation

$$27U_h(x,y,z) - \sum_{l,m,n\in\{-1,0,1\}} U_h(x+lh, y+mh, z+nh) \tag{3}$$
$$= F_h(x,y,z) \quad \text{for all } (x,y,z) \in \overset{\circ}{\Omega}_h \tag{4}$$
$$U_h(x,y,z) = 0 \quad \text{for all } (x,y,z) \in \Omega_h \backslash \overset{\circ}{\Omega}_h, \tag{5}$$

$\overset{\circ}{\Omega}_h = \Omega_h \cap]0,1[^2$ denotes the interior grid points and $F_h(x,y,z)$ is a suitable scaled local average of the right hand side f. For simplicity, let us denote

$$L_h(U_h)(x,y,z) = -27U_h(x,y,z) + \tag{6}$$
$$\sum_{l,m,n\in\{-1,0,1\}} U_h(x+lh, y+mh, z+nk). \tag{7}$$

the discrete Laplace operator.

SEMI-UNSTRUCTURED GRID

The use of such a structured grid Ω_h has several advantages in comparison to a pure unstructured grid. One of them is the small storage requirement, since the discretization stencil is a fixed stencil independent of the grid point. Other advantages are the superconvergence of the gradient and the natural construction of coarse grids. To be able to discretize more general domains, we apply so called semi-unstructured or embedded structured grids. These grids consist of a large structured grid in the interior of the domain and an unstructured grid, which is only contained in boundary cells. A detailed description of semi-unstructured grids for general domains in 2D and 3D is given in [2]. Here, we describe only the main properties of semi-unstructured grids. A semi-unstructured grid generation is based on the structured grid Ω_h, and leads to the following objects:

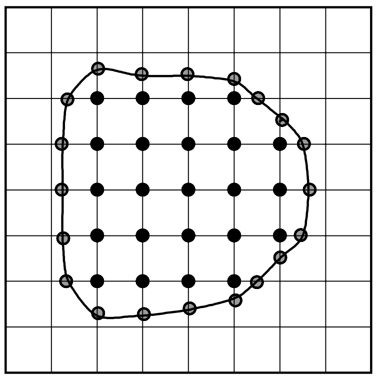

Fig. 1: Semi-unstructured grid.

- All cells $[lh,(l+1)h]\times[nh,(n+1)h]\times[mh,(m+1)h]$, $l,n,m = 0,\cdots N-1$ are classified in
  - interior cells,

  - boundary cells,
  - exterior cells.

  The boundary of Ω cuts the boundary cells. This cut is approximated by triangles for every boundary cell. The union of all these triangles and all interior cells is the *discretization domain* Ω_h.

- The semi-unstructured grid is the set of *nodal points*

$$\mathcal{N}_h \quad := \quad \mathcal{N}_{h,i} \cup \partial\mathcal{N}_h,$$

  where $\partial\mathcal{N}_h$ are the boundary nodal points and $\mathcal{N}_{h,i} \subset \Omega_h$ are the interior nodal points. The boundary nodal points are constructed in such a way that every boundary nodal point $p \in \mathcal{N}_h$ is contained in the interior of an edge of a boundary cell.

To obtain a finite element discretization on the grid $\mathcal{N}_h$ let V_h be the space of linear elements.

Several advantages of the structured grid Ω_h still remain for the semi-unstructured grid $\mathcal{N}_h$. One of them is the low storage requirement, since the discretization stencils of the structured grid are constant. Another is the natural construction of coarse grids up to a very coarse grid. Such constructions are important for obtaining an optimal multilevel iterative solver. Furthermore, the structured grid inside of the domain leads to a local super-convergence of the gradient.

The Gradient Operator

To obtain an accurate approximation of the gradient of u, we apply a finite element approach using a weak formulation. To explain this approximation let V_h be the finite element space. Then, define the approximation of the gradient $\delta_h(u) \in V_h$ as follows

$$\int_\Omega \frac{\partial u}{\partial x} v_h = \int_\Omega \delta_h(u)\, v_h \, dx \tag{8}$$

for all $v_h \in V_h$.

Particle Grid-Interpolation

Let V_h be the finite element space, the space of piecewise linear functions on the discretized grid Ω_h. The operator $\mathcal{I}$ is defined by:

$$\begin{aligned} &\mathcal{I} : \Omega_h \mapsto \Re \\ &\mathcal{I}(q) = \int_\Omega \sum_{i=1}^{N} \delta(x_i - x) v_{q,h}(x) d(x) \end{aligned} \tag{9}$$

where $v_{q,h} \in V_h$ are finite element base functions to $q \in \Omega_h$. $x_i \in \Omega$ represents one particle with unit charge in real space.

EFFICIENT PARALLELIZATION

An automatic parallelization of a code can only be achieved if the code is implemented in a suitable language. Such a language can be provided by expression templates in C++. Using expression templates, one can implement operators like +,-, ... in such a way that expressions like

```
u = a+b+c;
```

are evaluated in an efficient way for vectors a,b,c. The main idea of this concept is to implement the operator + such that a+b does not return the resulting vector, but a template object which is able to evaluate a+b efficiently for every component of the vector. This idea was originally proposed in [3], and allows C++ to achieve the same performance on vector and matrix expressions as with Fortran.

The expression template concept can also be extended to solvers for finite element discretizations. Of course, in this case, one needs a more complicated data structure for storing the vectors on a discretization grid. Let us call such vectors **Variable**. Then, the Jacobi iteration and the Gauss-Seidel iteration

Jacobi iteration for Poisson's equation

```
void Smoother(Variable& u, Variable& f, Variable& r)
{
  r = f + Laplace_FE(u);
  u = u - r ω;
}
```

Gauss-Seidel iteration for Poisson's equation

```
void Smoother(Variable& u, Variable& f, Variable& r)
{
  u = u - Laplace_FE(u) - f;
}
```

Fig. 2: One Jacobi and Gauss-Seidel iteration

for Poisson's equation can be implemented almost identical to the mathematical formulation. A very nice feature of expression templates not only for pedagogical reasons. Here Laplace_FE(u) means the discrete Laplace operator L_h (6) for a finite element discretization.

THE MULTIGRID SOLVER

A multigrid algorithm [4] to solve the linear system of equations resulting from the (FEM) discretization of Ω and the corresponding operator (6) is based on a sequence of fine and coarse grids

$$\Omega_{h_1} \subset \Omega_{h_2} \subset \Omega_{h_2} \subset \cdots \subset \Omega_{h_n} \tag{10}$$

and restriction and prolongation operators

$$\begin{aligned} R_{h_i} &: \Omega_{h_{i+1}} \rightarrow \Omega_{h_i} \\ P_{h_i} &: \Omega_{h_{i-1}} \rightarrow \Omega_{h_i}. \end{aligned} \tag{11}$$

```
Multigrid Algorithm
void MG(Variable& u, Variable& f, Variable& r,
int level) {
   if($level > 1$) {
      Smoother(u,f,r); // pre-smoothing
      // restriction
      r = Laplace_FE(u) - f;
      f = Restriction_FE(r);
      u.Level_down();
      u = 0.0;
      // coarse-grid correction
      MG(u,f,r,level-1);
      u = u + Prolongation_FE(u);
      f.Level_up();
      Smoother(u,f,r); // post-smoothing
   }
   else { // smoothing on coarsest grid
      Smoother(u,f,r);
   }
}
```

Fig. 3: Multigrid algorithm

Restriction and prolongation has to be applied to FEM-spaces and to the differential operators from fine to coarse grid (and vice versa). Depending on the grid and the operator additional structures must be provided.

In case of the uniform structured grid Ω_h one can geometrically construct the coarse grid as follows:

$$\Omega_{h_i} := \Omega_{h2^{n-i}}.$$

In case of an unstructured grid, the construction of coarse grids is a non-trivial task, since one has to apply an algebraic coarsening. But since $\mathcal{N}_{h,i}$ is a semi-unstructured grid and since the boundary conditions are Dirichlet and periodic boundary conditions, we can construct coarse grids as follows:

$$\Omega_{h_i} := \Omega_{h2^{n-i}} \cap \mathcal{N}_{h,i}.$$

For the implementation of the multigrid algorithm, the concept of activity of points (arising in the context of expression templates) has to be extended to the concept of activity of levels. Expressions are evaluated only on the grid Ω_{h_i}, where i is the active level of this expression. In order to change the activity level operators like *Level_down()* or *Level_up()* are implemented and used as shown in Fig 3.

PERFORMANCE RESULTS

For the Poisson solver the type of simulations described in [1] is very demanding. First of all, the computational domain Ω is very large *and* almost completely filled with simulation particles (protons and electrons). Second, the number of macro particles (or simulation particles) is huge (many times 10^6) and the number of time steps is large as well.

Performance results of the parallel Poisson Solver and the parallel grid generators is shown in Table 1 for a toy Poisson problem where $\Omega = \mathcal{S}^3$ (sphere). We show in Table 1 the scalability of the grid generator and the solver. A method is said to be scalable, if the time (T) times the number of processors used (P) divided by problem size (M) remains bounded as P and M get increased. The data in Table 1 is given for the grid generation (in column 3) and for one multigrid iteration (in column 5) with an Gauss-Seidel smoother. Table 1 shows excellent scalability with respect to the problem size M which is equivalent to say we can handle in the order of 10^{11} **macro particles** in a simulation with reasonable computing time. For this scaling study we use the Seaborg (IBM SP-3) computer at NERSC.

P	M	T_gP/M	T	TP/M
8	625,464	3.5e-3	3.1	3.9e-5
32	306,080	8.5e-3	0.78	8.1e-5
248	4,751,744	5.90e-3	1.2	6.2e-5
248	36,998,619	7.50e-3	7.7	5.1e-5
960	23,312,735	4.85e-3	4	1.64e-4
2025	405,242,845	6.60e-3	10.7	5.3e-5
4075	7,166,171,845	8.76e-3	160	9.9e-5

Tab. 1: Scalability of the parallel grid generator T_gP/M and the Poisson solver showing also T, the time in seconds for one Multigrid step

CONCLUSION

A novel massive parallel Poisson solver Kernel is presented. Scalability of the grid generator and the solver is demonstrated up to **4075** processors. Next steps are the integration of the solver in beam dynamic simulation programs and improve the sustained performance.

ACKNOWLEDGMENTS

A.A acknowledges the used resources of the NERSC, which is supported by the Office of Science of the U.S. Department of Energy under Contract No. DE-AC03-76SF00098.

REFERENCES

[1] Andreas Adelmann and M. A. Furman , PARSEC: Parallel Self-consistent 3D Electron-Cloud Simulation in Arbitrary External Fields, PSI Annual Report 2003.

[2] Christoph Pflaum, Semi-unstructured grids, Computing, 67(2):141-166, 2001

[3] T. Veldhuizen, Expression templates, C++ Report, 7(5):26-31, 1995

[4] U. Trottenberg, C.W. Oosterlee, A. Schüller, Multigrid, Academic Press 2001s, ISBN 0-12-701070-X