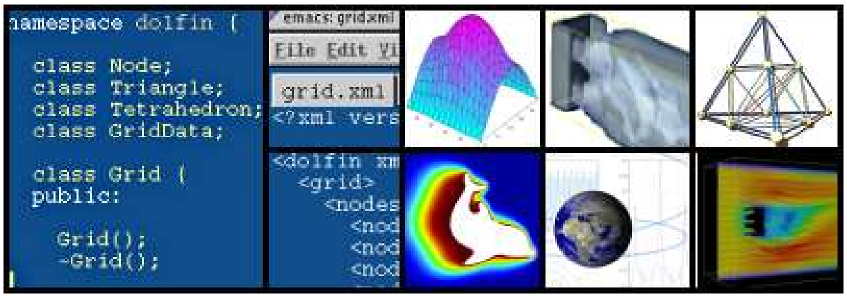
PDE Project Course 03/04

Suggestions for projects



Anders Logg
Dept. Computational Mathematics
Chalmers University of Technology
Email: `logg@math.chalmers.se`

General guidelines

This document contains a list of suggestions for projects. These are only suggestions, so feel free to design your own project! Think of this list as an inspiration, and perhaps a hint on the expected level of your projects.

Concerning grades, the projects are divided into two parts: basic level and advanced level. Basic level means grade 3 and advanced level means grade 4 or 5. However, advanced level is no guarantee for grade 4 or 5. It is also required that your report and your presentation match the level of your project. It is also possible to receive a higher grade even if you only complete the basic level, if you deliver an excellent report and an excellent presentation.

Have fun!
Anders

1 Chemical reactions

Simulate the following system of chemical reactions, where the substances A and B react to form C: $A + B \to C$.

Consider a beaker containing a solution of A with given concentration. To this beaker, we add a drop of B every second until finally A has "completely" reacted with B. Try to find a suitable reaction to simulate in a chemistry book. Maybe the reaction you want to simulate is instead given by $2A+3B \to 4C$, or perhaps $5A + 2B + C \to 2C$?

Model this as a system of reaction–diffusion equations, where $u_1(x, t)$ and $u_2(x, t)$ are the two concentrations to be determined.

Advanced

Consider one of the following extensions:

- Add a magnetic stirrer to increase the mixing of the substances in the beaker. Model this by adding appropriate convection terms to the system.
- Implement your solver as a module in DOLFIN.
- Implement adaptive time-stepping, and investigate the efficiency of your solver compared to your original non-adaptive solver.

References

1. *Applied Mathematics: Body and Soul*, by Eriksson, Estep, and Johnson, Springer Verlag 2003.
2. *Computational Differential Equations*, by Eriksson, Estep, Hansbo, and Johnson. Studentlitteratur 1996.
3. Some suitable book on chemistry.

2 Pattern formation

Read the article on pattern formation by Pearson, and try to repeat (some of) the results presented in the article by implementing a solver for the given system of reaction–diffusion equations.

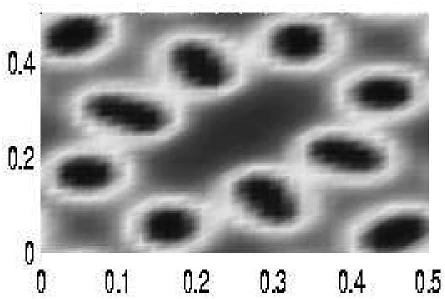


Figure 1: Example of a pattern obtained in a simulation of the reaction–diffusion system.

Advanced

Consider one of the following extensions:

- Implement your solver as a module in DOLFIN.
- Implement adaptive time-stepping, and compare the efficiency of your solver to your original non-adaptive solver.

References

1. *Applied Mathematics: Body and Soul*, by Eriksson, Estep, and Johnson, Springer Verlag 2003.
2. *Computational Differential Equations*, by Eriksson, Estep, Hansbo, and Johnson. Studentlitteratur 1996.
3. *Complex Patterns in a Simple System*, by John E. Pearson. Science Vol. 261 (1993) pp. 189-192.

3 Poisson's equation

Solve Poisson's equation in 2D and evaluate different types of error estimates (energy norm or L_2). Study the convergence with respect to the mesh size h. How sharp are the error estimates?

Advanced

Consider one of the following extensions:

- Implement higher-order elements (quadratic) and study again the convergence with respect to h.
- Do the computations in 3D.
- Use the error estimates to do adaptivity.

References

1. *Applied Mathematics: Body and Soul*, by Eriksson, Estep, and Johnson, Springer Verlag 2003.
2. *Computational Differential Equations*, by Eriksson, Estep, Hansbo, and Johnson. Studentlitteratur 1996.

4 The Navier-Stokes equations

Implement a solver for the Navier-Stokes equations in 2D or 3D.

Advanced

Nothing extra is needed for advanced level.

References

1. *Applied Mathematics: Body and Soul*, by Eriksson, Estep, and Johnson, Springer Verlag 2003.
2. *Computational Differential Equations* by Eriksson, Estep, Hansbo, and Johnson. Studentlitteratur 1996.
3. *Adaptive finite element methods for turbulent flow* by Johan Hoffman. Chalmers Finite Element Center Preprint 2002–08, available at `http://www.phi.chalmers.se/preprints/`.

5 Iterative solvers for linear systems

Implement GMRES and the conjugate gradient method for sparse linear systems. Study the convergence when the sparse matrix A is given by a discretization of Poisson's equation.

Advanced

Implement the solver as a C++ class in DOLFIN. Compare the performance to the existing solver. Maybe you can do better? Implement a module for Poisson's equation in DOLFIN that makes use of your new solver.

References

1. *Applied Mathematics: Body and Soul*, by Eriksson, Estep, and Johnson, Springer Verlag 2003.
2. *Computational Differential Equations* by Eriksson, Estep, Hansbo, and Johnson. Studentlitteratur 1996.
3. *Applied Numerical Linear Algebra* by James W. Demmel. SIAM 1997.

6 Iterative solvers for eigenvalue problems

Implement an iterative solver (some version of the Lanczos algorithm) for eigenvalue problems. Study the convergence when the sparse matrix A is given by a discretization of Poisson's equation.

Advanced

Implement the solver as a C++ class in DOLFIN. Implement a module for Helmholtz' equation in DOLFIN that makes use of your new solver.

References

1. *Applied Mathematics: Body and Soul*, by Eriksson, Estep, and Johnson, Springer Verlag 2003.

2. *Computational Differential Equations* by Eriksson, Estep, Hansbo, and Johnson. Studentlitteratur 1996.

3. *Applied Numerical Linear Algebra* by James W. Demmel. SIAM 1997.

7 Mesh generation

Implement your own mesh generator for triangular meshes in 2D. Investigate different existing algorithms and pick the one you like best, or construct your own algorithm.

Advanced

Implement a mesh generator for tetrahedral meshes in 3D.

References

1. *Applied Mathematics: Body and Soul*, by Eriksson, Estep, and Johnson, Springer Verlag 2003.
2. *Computational Differential Equations* by Eriksson, Estep, Hansbo, and Johnson. Studentlitteratur 1996.
3. `http://www.google.com`. Keywords: mesh generation, grid generation, Delauney, Voronoi, advancing front, quad-tree, oct-tree.

8 Mesh refinement

Implement adaptive mesh refinement for triangular meshes in 2D. Investigate different existing algorithms and pick the one you like best, or construct your own algorithm.

Advanced

Implement adaptive mesh refinement for tetrahedral meshes in 3D.

References

1. *Applied Mathematics: Body and Soul*, by Eriksson, Estep, and Johnson, Springer Verlag 2003.

2. *Computational Differential Equations* by Eriksson, Estep, Hansbo, and Johnson. Studentlitteratur 1996.

3. `http://www.google.com`. Keywords: mesh refinement, grid refinement, bisection, octasection.

9 The Lorenz system

Implement a solver for the Lorenz system. Use the solver to investigate the *computability* of the Lorenz system. How far is the solution computable? Use the cG(1) method with adaptive time-stepping.

Advanced

Consider one of the following extensions:

- Solve the dual problem and compute stability factors. Make predictions concerning the computatibility of the Lorenz system from the growth of the stability factors, and try to verify your predictions with experiments.
- Implement a higher-order method, such as cG(2) or cG(3), and see if you can reach further than you did with the cG(1) method.

References

1. *Applied Mathematics: Body and Soul*, by Eriksson, Estep, and Johnson, Springer Verlag 2003.
2. *Computational Differential Equations* by Eriksson, Estep, Hansbo and Johnson. Studentlitteratur 1996.
3. *Multi-Adaptive Galerkin Methods for ODEs I: Theory & Algorithms* by Anders Logg. Chalmers Finite Element Center Preprint 2001–09, available at `http://www.phi.chalmers.se/preprints/`.
4. *Multi-Adaptive Galerkin Methods for ODEs II: Applications* by Anders Logg. Chalmers Finite Element Center Preprint 2001–10, available at `http://www.phi.chalmers.se/preprints/`.

10 Mechanical systems

Simulate a complex mechanical system, given by (a large number of) point masses, springs and dampers. This can be used to simulate a wide class of physical systems, such as a bungy-jumper, a bridge, the Solar system or perhaps a cow...

Solve the resulting system of ODEs using the cG(1) method.

Advanced

Consider one of the following extensions:

- Solve the dual problem and compute stability factors. Interpret the stability factors in terms of the computatibility of the system.
- Implement adaptive time-stepping for your system and investigate the efficiency of your solver compared to your original non-adaptive solver.

References

1. *Applied Mathematics: Body and Soul*, by Eriksson, Estep, and Johnson, Springer Verlag 2003.
2. *Computational Differential Equations* by Eriksson, Estep, Hansbo and Johnson. Studentlitteratur 1996.