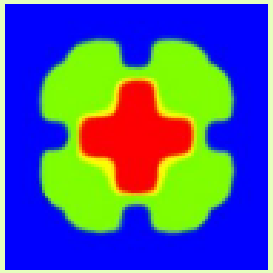
Short Course on Partial Differential Equations with deal.II

Play Time

Daniel Arndt

IWR, Universität Heidelberg



April 09 – 13, 2018

Goals

Goals of this course

- Learn about the software library `deal.II`
- understand practical aspects of finite element software
- use the library `deal.II` for own computations
- Build, document, and present a software project
- solve nonlinear, time-dependent and coupled PDEs
- use advanced tools in software development (IDEs, debuggers, ...)
- Not:
  - Teach the Finite Element Method (see 11MMAV0277 Numerical Methods for Partial Differential Equations)
  - Learn how to program in C++ (but I am happy to help!)

Topics

Topics

- basics of FEM, structure of FEM codes, algorithmic aspects
- modern tools for software development (IDEs, debuggers, . . .)
- some C++ topics (templates, . . .) used in large software projects
- iterative solvers and preconditioners
- coupled PDEs, block systems
- nonlinear problems
- time discretization
- parallel computations
- software engineering practices

Interacting

For you

- Join the mailing lists
- Ask questions
- Just read and learn
- Become a contributor
- Smallest changes are welcome! (find a typo? Documentation of a function lacking? Implement a small feature?)
- Cite `deal.II` if you use it

Play Time - step-1 I

1. Create an image of an L-shape domain (add another function to step-1).
2. Refine the mesh in 1) adaptively around the re-entrant corner.
3. Create a helper function that takes a Triangulation and outputs the following information: number of levels, number of cells, number of active cells. Test this with the (now three) meshes.
4. Output the mesh as an svg file instead of eps. Open it in a browser to display it.
5. Create a 3d cylinder and refine it 3 times (globally). Try outputting to gnuplot format too.

Play Time - step-1 II

6. Bonus: create a 3d unit cube and create a loop that in each step a) refines globally once, b) outputs the number of active cells, c) the amount of memory in megabytes required to store this mesh (look for a function called memory consumption()). Do this for 6 global refinements first. How many refinements can you fit into memory of your computer (typing free in the terminal tells you how much memory you have)?

Play Time - step-2 I

1. How does the pattern change if you increase the polynomial degree from 1 to 2 or to 3?
2. How does the pattern change if you use a globally refined (say 3 times) unit square?
3. How many entries per row do you expect for a Q1 element (assuming four cells are around each vertex)? Check that this is true for the mesh in b) (look for row length and output them for each row).
   How does that change with a different mesh?
   Can you construct a mesh (without hanging nodes) that has a row with more entries?
4. Print all entries for row 42 for the original renumbered sparsity pattern.
5. Are these patterns symmetric? Why/why not?

Play Time - step-2 II

6. Compute and output statistics like the the number of unknowns, bandwidth of the sparsity pattern, average number of entries per row, and fill ratio.
7. Bonus: figure out a way to write the sparsity pattern into a file that you can either read in using Matlab or write it as a .PPM image file.