Otto-von-Guericke Universität Magdeburg
Institut für Analysis und Numerik
Prof. Dr. Thomas Richter
carolin.mehlmann@ovgu.de

thomas.richter@ovgu.de

16.07.2018

MathCore: Discretization & Solution

Today, we

- implement a simple finite element discretization for the Laplace Equation
- we approximate the problem

$$-\Delta u = f \text{ in } \Omega, \quad u = 0 \text{ on } \partial\Omega$$

- we consider a 2d- and a 3d-domain

$$\Omega = (0,1)^2, \quad \Omega = (0,1)^3.$$

- Slides: `https://www.math.uni-magdeburg.de/~richter/mathcore/finiteelements.pdf`

Matlab-Template

- Download and unzip `https://www.math.uni-magdeburg.de/~richter/mathcore/template1.zip`
- Open Matlab and look at the files. A lot is already prepared

  createmesh.m This function gets two arguments, `DIM` for the spatial dimension and `M`, where $M-1$ is the number of inner mesh points in every direction. The mesh has a total number of $N = (M-1)^{DIM}$ inner points (x_i, y_j) (or (x_i, y_j, z_k) in 3d). The indices i and j run from 1 to $M-1$. The mesh size is $h = 1/M$ so that $x_i = ih$. The mesh is stored as a $N \times DIM$ matrix so that `mesh(:,1)` is the vector of x-coordinates. The sorting of the unknowns is lexicographic, x being the innermost index, then y, then z (in 3d) as outer index. Use the function, e.g. `createmesh(2,3)`, and have a look at the output.

  setrhs.m This function takes the mesh and returns the right hand side vector b. Right hand side functions f are integrated with the trapecoidal rule, such that (in 2d and 3d)

$$b_{ij} = h^2 f(x_i, y_j), \quad b_{ijk} = h^3 f(x_i, y_j, z_k).$$

  At the moment the right hand side is given as $f(x,y) = 2\pi^2 \sin(\pi x)\sin(\pi y)$ (likewise in 3d). This is the right hand side to the solution $u(x,y) = \sin(\pi x)\sin(\pi y)$, i.e. $f = -\Delta u$.

  plotsolution.m This functions gets a name for the plot, the mesh and a data vector to plot.

- Furthermore you have some empty templates to be finished by you.

1. In `assemblematrix.m` implement the finite element matrix for the Laplace problem for the discretization with piecewise linear functions on a uniform triangular mesh. This is the 5-point stencil

in 2d and 7-point stencil in 3d

$$\begin{array}{ccc} & & -1 \\ S_h^{2d} = \begin{bmatrix} & -1 & \\ -1 & 4 & -1 \\ & -1 & \end{bmatrix}, \quad S_h^{3d} = & h & \begin{bmatrix} & -1 & \iddots \\ -1 & 6 & -1 \\ \iddots & -1 & \end{bmatrix} \\ & -1 & \end{array}$$

Use `sparse-matrices` and try to avoid loops. (Hint: look at the matlab functions `eye` and `kronecker`)

With `full(A)` you can print a sparse matrix and with `spy(A)` you can visualize the sparsity pattern of the matrix.

2. Finish the function `start.m`. It gets two parameters, `DIM` and `M`. It should:

   1. Set up the mesh
   2. Set up the matrix `A`
   3. Set up the right hand side `b`
   4. Solve the problem with `x = A\b`
   5. Plot the solution

   Start the program in 2d and 3d with different values of `M`.

3. We want to compute the finite element error

$$\|u - u_h\|_\infty$$

   Look at `exactsolution.m` and `setrhs.m`. The function `computeerror.m` (everything is already done) computes the error between a finite element solution and the exact solution and prints out the maximum norm.

   a) Modify `start.m`: after solving the problem in `start.m` add a call to `computeerror.m` to compute the error (in between steps 4 and 5)

   b) Run the problem and try to reduce the error as much as possible. Does the error go to zero with $h \to 0$? If not, right hand side, matrix or exact solution are wrong.

   c) Check that you get the theoretical order of convergence

$$\|u - u_h\|_\infty = \mathcal{O}(h^2)$$

   (Note: $h = 1/M$ and thus $\|u - u_h\|_\infty = \mathcal{O}(1/M^2)$)

4. We want to solve the Laplace problem $-\Delta u = f$ such that the solution is given by

$$u(x, y) = \sin(\pi x)\sin(\pi y)\exp(5x), \quad u(x, y, z) = \sin(\pi x)\sin(\pi y)\sin(\pi z)\exp(5x).$$

   a) Compute the corresponding right hand side via

$$f = -\Delta u.$$

b) Implement the right hand f side in `setrhs.m` and implement the exact solution u in `exactsolution.m`

c) Run the problem and try to reduce the error as much as possible. Does the error go to zero with $h \to 0$? If not, right hand side, matrix or exact solution are wrong.

Try to reach and error of 10^{-3} in 2d and 3d.