Technische Universität München
Center for Mathematical Sciences, M17

Dr. Anne-Céline Boulanger, Konstantin Pieper
www-m17.ma.tum.de/Lehrstuhl/LehreSoSe14LabCourseOptPDE

Summer 2014

Lab course: Optimization with elliptic PDEs: Sheet 0

Exercise 0.1 (From a continuous to a discrete problem — Linear setting): We consider a linear elliptic boundary value problem on the square domain $\Omega = (-1,1) \times (-1,1)$

$$\begin{cases} -\Delta u + u = q & \text{in } \Omega, \\ \partial_n u = 0 & \text{on } \partial\Omega, \end{cases} \tag{1}$$

where the source term q is given by

$$q(x_1, x_2) = \sin\left(\tfrac{\pi}{2} x_1\right) \sin\left(\tfrac{\pi}{2} x_2\right).$$

To solve the example (and each of the following), you should use the following recipe:

On a sheet of paper:

1. Continuous weak form. State the weak form of problem (1) as

   $$\text{Find } u \in V \text{ such that } a(u, \varphi) = f(\varphi) \quad \text{for all } \varphi \in V.$$

   Find the appropriate bilinear form a, linear form f and space V.

2. Discrete weak form. We denote by $V_h \subset H^1(\Omega)$ the space of linear finite elements on a triangulation T_h of Ω and denote by $(\varphi_i)_{i=1,\dots,N} \subset V_h$ the nodal basis defined by $\varphi_i(x_j) = \delta_{ij}$ for $x_j \in \mathcal{N}(T_h)$ (the nodes of the mesh).

   Write the discrete weak form of problem (1) for the discrete solution $u_h \in V_h$.

3. Matrix formulation. The discrete solution $u_h = \sum_{i=1}^{N} \mathbf{u}_i \varphi_i$ is represented by the nodal basis vector $\mathbf{u} \in \mathbb{R}^N$.

   Introducing an appropriate matrix $\mathbf{A}$ and vector $\mathbf{b}$, write the discrete weak formulation as the linear system

   $$\mathbf{A}\mathbf{u} = \mathbf{b}.$$

   What are the entries of $\mathbf{A}$ and $\mathbf{b}$?

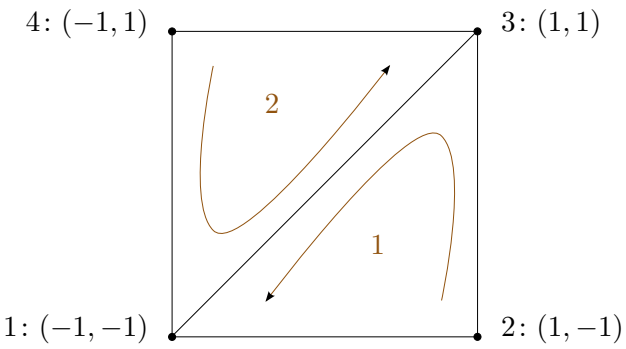

Figure 1: The initial triangulation/mesh of the square $\Omega = (-1, 1) \times (-1, 1)$.

In the code: Create a directory for your files and copy the directory `FEMlib` there. For the examples, you can copy the files `Ex_0_*.m` and base your work on them.

1. Add the library `FEMlib` to the search path.

   ```
   addpath 'FEMlib'
   ```

2. Create first the initial mesh of Ω using the code:

   ```
   node = [-1,-1; 1,-1; 1,1; -1,1];
   elem = [2,3,1; 4,1,3];
   boundary = Boundary('gamma', [1,2; 2,3; 3,4; 4,1]);
   mesh0 = Discretization(node, elem, boundary);
   ```

   Try to figure out how the code relates to Figure 1.

3. Refine it with `mesh = PreRefine(mesh0, refine)`. The parameter `refine` controls the discretization level of the mesh. Try values between 1 and 9.

4. Assemble the matrix using `A = AssembleMatrix(mesh, @a)`. The form `a` shall be prescribed in a function with signature

   `function a = a(u, phi, data)`.

   Assemble the load vector using `b = AssembleIntegralVector(mesh, @f)`. The linear functional `f` shall be prescribed in a function with signature

   `function f = f(phi, data)`.

   The vectors `u` and `phi` contain the values of the ansatz and test function in the quadrature points. Access the values of the function `phi` with `phi.m`, its x_1-derivative with `phi.x` and its x_2-derivative with `phi.y` (the same for `u`). `data.point.x` and `data.point.y` contain the x_1 and the x_2 values at the quadrature points.

   The functions shall return the pointwise values in the quadrature points of the linear/bilinear form for specific ansatz and test functions, e.g., for the first example

   ```
   a = u.x.*phi.x + u.y.*phi.y + u.m.*phi.m;
   f = sin(pi/2 * data.point.x).*sin(pi/2 * data.point.y).*phi.m;
   ```

5. Compute u_h by solving `u = A\b`. Visualize u_h using the functions `visu` or `visusimple`. Compare with Figure 2.

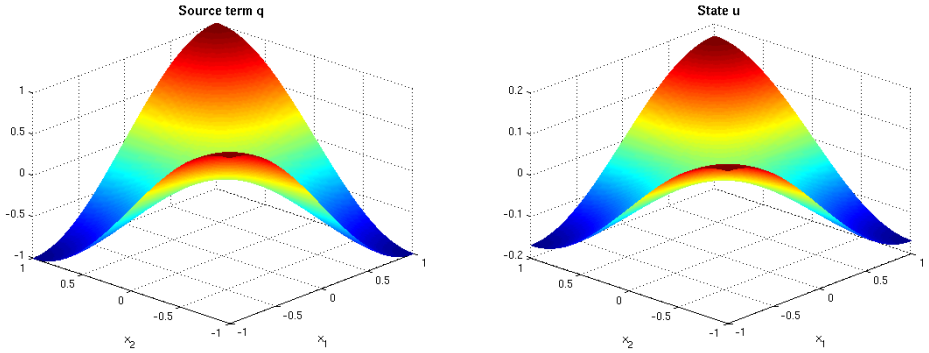

Figure 2: $-\Delta u + u = q$ with zero Neumann data. q is a sine function.

Here are some other test cases that you can try.

1. Solve again (1) but with the source term $q(x_1, x_2) = \chi_{\left\{\sqrt{x_1^2+x_2^2} \leq 0.5\right\}}$. Compare with Figure 3.

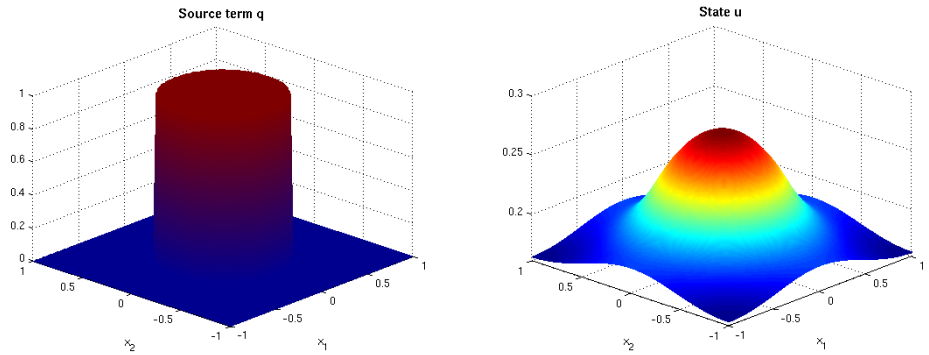

Figure 3: $-\Delta u + u = q$ with zero Neumann data. q is an indicator function.

2. Solve

$$\begin{cases} -\Delta u + c(x)u = q & \text{in } \Omega, \\ \partial_n u = 0 & \text{on } \partial\Omega, \end{cases} \tag{2}$$

where the coefficient c is given by $c(x_1, x_2) = \sin(x_1)$ and the source term is $q = \sin(\frac{\pi}{2}x_1)\sin(\frac{\pi}{2}x_2)$. Compare with Figure 4.

3. Solve

$$\begin{cases} -\nabla \cdot (g(x)\nabla u) + u = q & \text{in } \Omega, \\ \partial_n u = 0 & \text{on } \partial\Omega, \end{cases} \tag{3}$$

where the diffusion coefficient is given by $g(x_1, x_2) = 1 - (1 + \exp(5 - 20(x_1^2 + x_2^2)))^{-1}$ and the source is $q = \sin(\frac{\pi}{2}x_1)\sin(\frac{\pi}{2}x_2)$. Compare with Figure 5.

4. Now, we consider the case of Robin boundary conditions, i.e., for $\sigma \neq 0$,

$$\begin{cases} -\Delta u = 0 & \text{in } \Omega, \\ \partial_n u + \sigma u = q & \text{on } \partial\Omega. \end{cases} \tag{4}$$

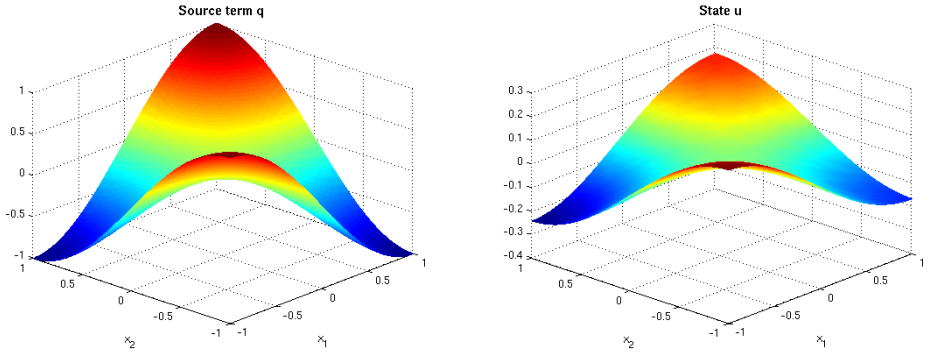

Figure 4: $-\Delta u + \sin(x_1)u = q$ with zero Neumann data. q is a sine function.

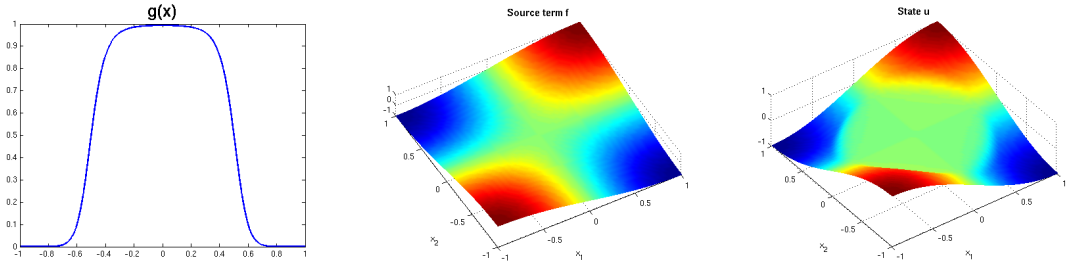

Figure 5: $-\nabla \cdot (g(x)\nabla u) + u = q$ with zero Neumann data. q is a sine function.

Since integrals over the boundary appear in the weak formulation, use additionally the functions `AssembleBoundaryMatrix` and `AssembleBoundaryIntegralVector` to build the corresponding matrix and vector.

For $\sigma \equiv 1$ and $q(x_1, x_2) = \sin(x_1)$ compare with Figure 6.

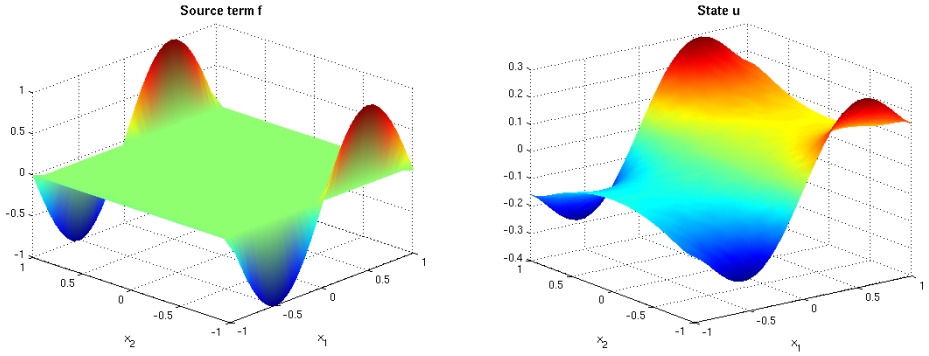

Figure 6: Case $-\Delta u = 0$ with Robin boundary conditions $\partial_n u + u = \sin(x_1)$.

Verify the correctness of your implementation. Consider again the problem (1). It is easy to check that the exact solution is given by the formula.

$$u^{\text{ref}}(x_1, x_2) = \frac{1}{1 + \pi^2/2} \sin\left(\tfrac{\pi}{2} x_1\right) \sin\left(\tfrac{\pi}{2} x_2\right)$$

Solve the problem (1) on a sequence of iteratively refined meshes. You can use the command

```
mesh = Refine(mesh);
```

to refine the mesh globally one more time. On each mesh, evaluate the L^2 error between the numerical and the exact solution $\|u^{\text{ref}} - u_h\|_{L^2(\Omega)}$. You can do this with the following code:

```
uref = @(x,y) 1/(1+pi^2/2) sin(pi/2*x).*sin(pi/2*y);
err = ComputeError(mesh, uh, uref)
```

Store the error and the number of nodes (degrees of freedom) from each refinement level each in a vector. Plot the error against the number of "dof's" in a loglog-plot. You should observe quadratic convergence.

Exercise 0.2 (The L^2 projection): The L^2 projection is the projection of a function $u_d \in L^2(\Omega)$ onto the finite element space $V_h \subset L^2(\Omega)$, which can be formulated as the minimisation problem,

$$u_h = \pi_h u_d = \operatorname*{argmin}_{\tilde{u}_h \in V_h} \frac{1}{2} \|\tilde{u}_h - u_d\|^2_{L^2(\Omega)}. \tag{5}$$

1. Show that the optimality condition for the unique solution of (5) is given by

$$(u_h, \varphi) = (u_d, \varphi), \quad \forall \varphi \in V_h. \tag{6}$$

2. Write (6) in the matrix form

$$\mathbf{M}\mathbf{u} = \mathbf{b}.$$

3. Compute the L^2 projection on V_h for the functions $u_d(x_1, x_2) = x_1^2$ and $u_d(x_1, x_2) = \chi_{\{\, x_1 \leq 0 \,\}}$ on $\Omega = (-1, 1) \times (-1, 1)$ with the mesh from Ex 0.1 on refinement levels $\{\, 1, 2, 3, 4, 5, 6 \,\}$.

   Plot u_h and $I_h u_d$ and compare the result, where $I_h u_d$ denotes the nodal interpolation of u_d

   Hint: Use the vectors `mesh.node(:,1)` *and* `mesh.node(:,2)` *to compute the nodal interpolation.*

Exercise 0.3 (From a continuous to a discrete problem — Semi-linear setting): *This section should be done before session 2 but after session 1.*

We consider a nonlinear (semilinear) elliptic problem. In this example, we add to the linear Laplace operator a cubic term.

$$\begin{cases} -\Delta u + \sigma u^3 = q & \text{in } \Omega, \\ \partial_n u = 0 & \text{on } \partial\Omega. \end{cases} \tag{7}$$

1. Write in the continuous and discrete cases the weak formulation of the state equation

   $$a(u)(\varphi) = (q, \varphi) \quad \text{for all } \varphi \in V.$$

   Since the form a is not linear in u we cannot set this problem in a matrix formulation.

2. The idea is now to use Newton's method on the residual

   $$r(u)(\cdot) = (q, \cdot) - a(u)(\cdot). \tag{8}$$

   Starting at some initial guess u_0, we solve for the unknown update $\delta u \in V$ the *linear* equation

   $$a'_u(u_0)(\delta u, \varphi) = r(u_0)(\varphi) \quad \text{for all } \varphi \in V. \tag{9}$$

   Note, that $a'_u(u_0)(\cdot, \cdot)$ is again a bilinear form.

   Transfer (9) to the discrete setting in terms of the update $\delta u_h \in V_h$.

   In matrix form this is equivalent to $\mathbf{A}\,\mathbf{du} = \mathbf{r}$. Give the expressions of the entries of $\mathbf{A}$ and $\mathbf{r}$.

   Use again the functions the functions `AssembleIntegralVector` with the linear forms `a` and `f` to build the vector `r` as well as `AssembleMatrix` with the bilinear form `a_u` to build the matrix $\mathbf{A}$. In contrast to the linear setting the forms `a` and `a_u` depend on the current value of `u` (i.e., in the first iterate, the function u_0). You can pass additional functions to the forms by creating first a data container:

   ```
   data = EmptyData()
   ```

   Then you can set additional FE-functions in the `node` structure and additional parameters in the `param` structure:

   ```
   data.node.u = zeros(N,1)
   data.param.sigma = 0.1
   ```

   These values are then passed to the forms in the third argument

   ```
   A = AssembleMatrix(mesh, @a_u, data)
   function a = a_u(du, phi, localdata)
       u = localdata.u;
       sigma = localdata.sigma;
       a = ..
   end
   ```

   Note, that the nodal functions have to be transformed and evaluated in the quadrature points, therefore `localdata.node.u` *is not the same object as* `data.u`.

Then apply the update $u_1 = u_0 + \delta u$. Repeat the process as necessary to generate a sequence of u_i's. You can base the stopping criterion on a sufficiently small norm of the vector $\mathbf{r} \in \mathbb{R}^N$.

3. Try your code for several σ and $q = \chi_{\{|x| \leq 0.5\}}$. Use $u_0 = 0$ as an initial guess. Compare with Figure 7.

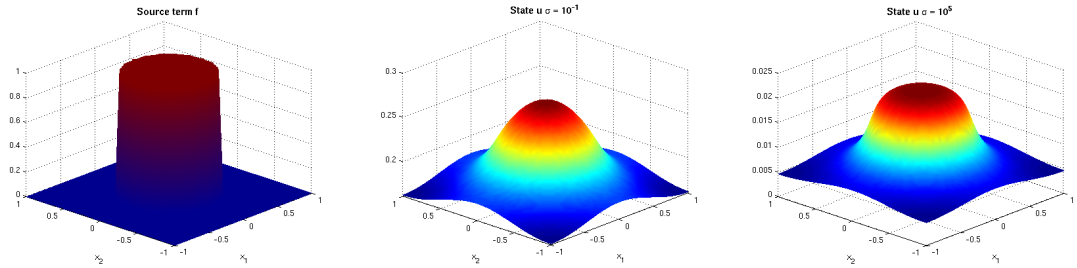

Figure 7: Semilinear case for $\sigma = 0.1$ and $\sigma = 10^6$.