S. Mishra
L. Scarabosio
J. Sukys

Spring Term 2014

Numerical Methods for Partial Differential Equations

ETH Zürich
D-MATH

Homework Problem Sheet 2

Introduction. The first problem of this problem sheet concerns the Finite Difference discretization of the Poisson equation on a 2-dimensional domain.
The other two problems address the discretization of one-dimensional problems using the Finite-Element method.

Problem 2.1 Finite Differences for 2D Poisson Equation

In this problem we consider the Finite Differences discretization of the Poisson problem on the unit square:

$$\begin{aligned} -\Delta u &= f \quad \text{in } \Omega := (0,1)^2, \\ u &= 0 \quad \text{on } \partial\Omega, \end{aligned} \tag{2.1.1}$$

for a bounded and continuous function $f \in \mathcal{C}^0(\overline{\Omega})$.

We consider a regular tensor product grid with meshwidth $h := (N+1)^{-1}$ and we assume a lexikographic numbering of the interior vertices of the mesh as depicted in Fig.2.1.

We first consider the 5-point stencil finite difference scheme for the operator $-\Delta$ described by the 5-points stencil shown in Fig. 2.2.

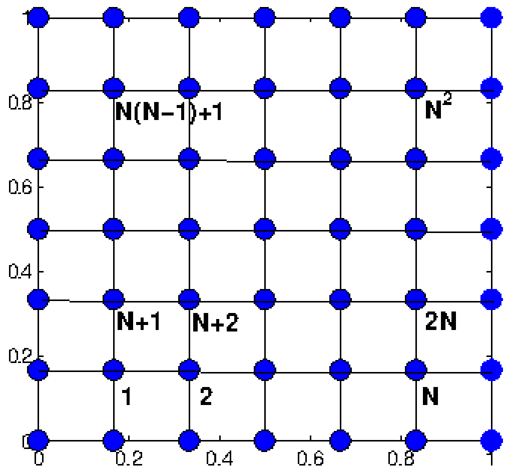

Figure 2.1: Lexikographic numbering of vertices of the equidistant tensor product mesh.

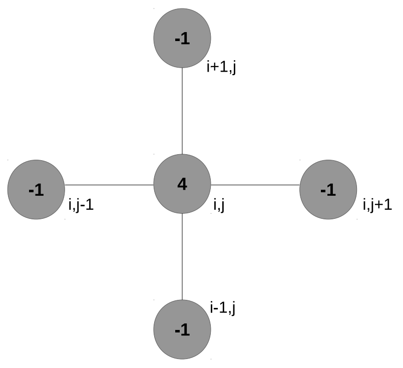

Figure 2.2: 5-point stencil for subproblems (2.1a)-(2.1f).

(2.1a) Write the system

$$\boldsymbol{A}\boldsymbol{u} = \boldsymbol{L} \tag{2.1.2}$$

corresponding to the discretization of (2.1.1) using the stencil in Fig. 2.2, specifying the matrix $\boldsymbol{A}$ and the vectors $\boldsymbol{L}$ and $\boldsymbol{u}$.

(2.1b) Write a function

```
A = PoissonMatrix2D(N)
```

to construct the matrix $\boldsymbol{A}$ in (2.1.2), where N denotes the number of interior grid points along one dimension.

(2.1c) Write a function

```
L = PoissonRHS2D(FHandle,N)
```

to build the vector $\boldsymbol{L}$ in (2.1.2). Here `FHandle` is a function handle to the function f in (2.1.1) and N the number of interior grid points.

(2.1d) Write a function

```
uh = PoissonSolve2D(FHandle,N)
```

to solve the system (2.1.2), with `FHandle` and N as in the previous subproblems.

(2.1e) Plot the discrete solution that you get from subproblem (2.1d) for $f(x, y) = 8\pi^2 \sin(2\pi x) \sin(2\pi y)$ and $N = 50$.

(2.1f) Investigate the convergence of the scheme in the L^∞-norm for $N = 10 \cdot 2^r$, $r = 1, \ldots, 6$. As right-handside f, consider the function $f(x, y) = 8\pi^2 \sin(2\pi x) \sin(2\pi y)$.
Which rate do you observe?

HINT: The exact solution is $u(x, y) = \sin(2\pi x) \sin(2\pi y)$.

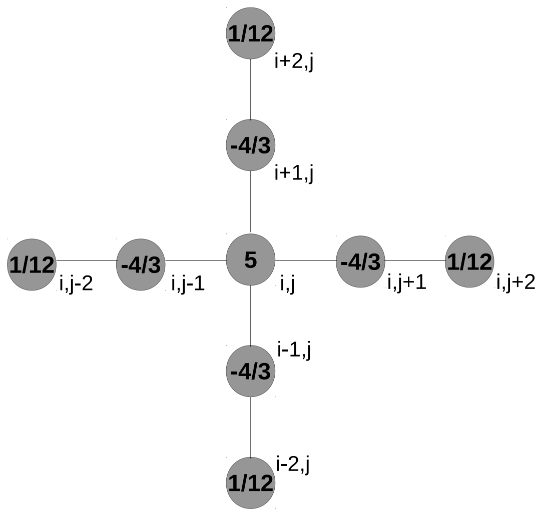

Figure 2.3: 9-point stencil for subproblems (2.1g)-(2.1j).

At this point, we are unsatisfied about the performance of the scheme that we consider, and we want then to adopt an higher order Finite Difference scheme.
To this aim, we consider the so-called *Collatz Merhstellenverfahren*, described by the 9-point stencil depicted in Fig. 2.3. More precisely, this stencil is applied to the inner nodes, i.e. the nodes with distance at least $2h$ from the boundary; for the first layer of nodes, distant h from the boundary, the 5-point stencil of Fig. 2.2 is used.

(2.1g) Describe the matrix $\boldsymbol{A}_M$ resulting from the discretization of (2.1.1) using the stencil 2.3.

(2.1h) Write a function

```
A = PoissonMehrstellen(N)
```

to implement the matrix $\boldsymbol{A}_M$ from subproblem (2.1g). Here N denotes again the number of interior grid points along one dimension.

(2.1i) Write a function

```
uh = PoissonSolveMehrstellen(FHandle,N)
```

to solve the system (2.1.1) using the *Mehrstellenverfahren*. The arguments are the same as in subproblem (2.1d).

(2.1j) Investigate the convergence of the *Mehrstellenverfahren* in the L^∞-norm for $N = 10 \cdot 2^r$, $r = 1, \ldots, 6$. Consider the same right-handside as in subproblem (2.1f).
Which rate of convergence do you observe?

Listing 2.1: Testcalls for Problem 2.1

```
UHandle = @(x) sin(2*pi*x(:,1)).*sin(2*pi*x(:,2));
FHandle = @(x) 8.*pi^2.*UHandle(x);

fprintf('\n\n##PoissonSolve2D:')
PoissonSolve2D(FHandle,5)'

fprintf('\n\n##PoissonSolveMehrstellen:')
PoissonSolveMehrstellen(FHandle,5)'
```

Listing 2.2: Output for Testcalls for Problem 2.1

```
>>test_call

##PoissonSolve2D:
ans =

  Columns 1 through 15

    0.8225    0.8225    0.0000   -0.8225   -0.8225    0.8225
       0.8225    0.0000   -0.8225   -0.8225    0.0000    0.0000
       -0.0000   -0.0000   -0.0000

  Columns 16 through 25

   -0.8225   -0.8225   -0.0000    0.8225    0.8225   -0.8225
      -0.8225   -0.0000    0.8225    0.8225

##PoissonSolveMehrstellen:
ans =

  Columns 1 through 15

    0.8177    0.8128    0.0000   -0.8128   -0.8177    0.8128
       0.7888    0.0000   -0.7888   -0.8128    0.0000    0.0000
       0.0000   -0.0000   -0.0000

  Columns 16 through 25

   -0.8128   -0.7888   -0.0000    0.7888    0.8128   -0.8177
      -0.8128   -0.0000    0.8128    0.8177
```

Problem 2.2 Linear Finite Elements in 1D (Core problem)

In class the Galerkin discretization of a 2-point boundary value problem by means of trial and test spaces of merely continuous piecewise linear functions was discussed. The Galerkin matrix for a linear variational problem was derived in detail. In this problem we practise the crucial steps for the slightly modified linear variational problem

$$u \in H_0^1([a,b]) : \quad \int_a^b \frac{\mathrm{d}u}{\mathrm{d}x}(x)\,\frac{\mathrm{d}v}{\mathrm{d}x}(x) + c\,u(x)v(x)\,\mathrm{d}x = \int_a^b g(x)v(x)\;\mathrm{d}x, \quad \forall v \in H_0^1([a,b]), \tag{2.2.1}$$

where $c > 0$, $-\infty < a < b < \infty$, $g \in \mathcal{C}^0([a,b])$. Please note that both trial and test functions vanish at the endpoints of the interval, as indicated by the subscript "0" in the symbol for the function space.

(2.2a) Derive the Galerkin matrix for (2.2.1), when using the trial and test space $\mathcal{S}_{1,0}^0(\mathcal{M})$ of continuous, piecewise linear functions on an *equidistant* mesh $\mathcal{M}$ with $N \in \mathbb{N}$ interior nodes. The standard basis of tent functions is to be used.

(2.2b) To obtain the right-hand side vector of the linear system arising from the Galerkin discretization of (2.2.1) as described in subproblem (2.2a), one relies on the composite trapezoidal rule on $\mathcal{M} = \{x_0 = a, x_1, \ldots, x_N, x_{N+1} = b\}$ for numerical quadrature:

$$\int_a^b f(x)\,\mathrm{d}x \approx f(a)\frac{h}{2} + h\sum_{i=1}^{N} f(x_i) + f(b)\frac{h}{2}, \tag{2.2.2}$$

with h the meshsize.

Implement an efficient function

```
u = linfegalerkinsol(a,b,c,g,N)
```

that computes the values of the Galerkin solution $u_N \in \mathcal{S}_{1,0}^0(\mathcal{M})$ at the nodes of the mesh $\mathcal{M}$ and returns them in the row vector `u`. The arguments `a`, `b`, `c` supply the domain $\Omega = [a,b]$, and the coefficient $c > 0$, whereas `g` is a function handle to the source function g. The argument `N` passes the number of interior nodes of the equidistant mesh.

(2.2c) State and justify the asymptotic computational complexity of `linfegalerkinsol` in terms of the problem size parameter N.

(2.2d) Plot the Galerkin solution u_N for $\Omega := [-\pi,\pi]$, $c = 1$, $g(x) = \sin(x)$, and $N = 50, 100, 200$. To validate your code compare u_N with the exact analytic solution $u(x) = \frac{1}{2}\sin(x)$.

(2.2e) Extend your above implementation of `linfegalerkinsol` to

```
u = linfegalerkinsolDirichlet(a,b,c,g,N,ua,ub),
```

where the optional arguments `ua`, `ub` may be used to specify *boundary values* for the solution u of (2.2.1). This means that now we seek to solve (2.2.1) under the constraints $u(a) = u_a$, $u(b) = u_b$.
HINT: Use the offset function technique to arrive at a modified right-hand side of the linear system of equations that incorporates the values u_a and u_b.

(2.2f) Plot the Galerkin solution u_N for $\Omega := [-\pi,\pi]$, $c = 1$, $g(x) = \cos(x)$, $u_a = u_b = -\frac{1}{2}$ and $N = 50$. To validate your code compare u_N with the exact analytic solution $u(x) = \frac{1}{2}\cos(x)$.

(2.2g) Investigate the convergence of the finite element discretization developed in (2.2b) in the L^∞-norm using $N = 10 \cdot 2^r$ degrees of freedom, for $r = 1, 2, \dots, 9$. This sequence of N-values should also be used for the following sub-problems. Use the test case discussed in (2.2d).

The exact evaluation of the L^∞-norm is hardly ever possible. Thus we have to resign ourselves to evaluating it only approximately. To do so, we rely on a mesh $\widetilde{\mathcal{M}}$ obtained by splitting each cell of the finite element mesh $\mathcal{M}$ into four smaller cells of equal size. Then sample the modulus of the error on all vertices of the finer mesh $\widetilde{\mathcal{M}}$ and find the maximal value.

HINT: For $N = 10$, the error should be around 0.016.

(2.2h) Investigate the convergence in the L^2-norm

$$\|e\|_{L^2} := \left(\int_0^1 |e(x)|^2 \, \mathrm{d}x \right)^{\frac{1}{2}}, \quad e \in \mathcal{C}^0_{\mathrm{pw}}([0,1]). \tag{2.2.3}$$

To evaluate this norm approximately, use *composite Gaussian quadrature* on the mesh $\mathcal{M}$ with two points per mesh cell.

HINT: The nodes and weights for 2-point Gaussian quadrature on $[-1,1]$ are $\zeta_1 = -\frac{1}{3}\sqrt{3}$, $\zeta_2 = \frac{1}{3}\sqrt{3}$, $\omega_1 = 1$, $\omega_2 = 1$. This quadrature rule has to be transformed to all mesh cells (x_{j-1}, x_j). For $N = 10$, the error should be around 0.012.

(2.2i) Investigate the convergence in the energy seminorm

$$|e|_{H^1} := \left(\int_0^1 \left| \frac{\mathrm{d}e}{\mathrm{d}x}(x) \right|^2 \right)^{\frac{1}{2}}, \quad u \in \mathcal{C}^1_{\mathrm{pw}}([0,1]). \tag{2.2.4}$$

Again, use composite 2-point Gaussian quadrature on $\mathcal{M}$ to approximate the integral.

HINT: For $N = 10$, the error should be around 0.150.

Listing 2.3: Testcalls for Problem 2.2

```
a=-pi;
b=pi;
c=1;
N=10;

fprintf('\n\n##linfegalerkinsol:')
linfegalerkinsol(a,b,c,@(x)(sin(x)),N)

fprintf('\n\n##linfegalerkinsolDirichlet:')
linfegalerkinsolDirichlet(a,b,c,@(x)(cos(x)),N,-1/2,-1/2)
```

Listing 2.4: Output for Testcalls for Problem 2.2

```
>> test_call

##linfegalerkinsol:
ans =

         0
   -0.2816
   -0.4737
   -0.5155
   -0.3936
   -0.1467
    0.1467
    0.3936
    0.5155
    0.4737
    0.2816
         0

##linfegalerkinsolDirichlet:
ans =

   -0.5000
   -0.4264
   -0.2097
    0.0780
    0.3434
    0.5015
    0.5015
    0.3434
    0.0780
   -0.2097
   -0.4264
   -0.5000
```

Problem 2.3 $L^2(0,1)$-Orthogonal Projection onto Linear Finite Element Space

In this problem we deal with a very simple *linear* variational problem that does not even involve derivatives. For discretization we employ linear finite elements. A careful re-examination of this section is recommended. You will be asked to implement the method and perform a numerical study of its convergence.

The variational problem reads: seek $u \in H^1([0,1])$:

$$\int_0^1 u(x)v(x)\,\mathrm{d}x = \int_0^1 f(x)v(x)\,\mathrm{d}x \quad \forall v \in V := H^1([0,1]). \tag{2.3.1}$$

Remark: Variational problems of this kind are encountered when computing the $L^2(0,1)$-orthogonal projection of f onto subspaces.

To begin with, we consider Galerkin discretization with the subspace $V_N = \mathcal{S}_1^0(\mathcal{M})$ of piecewise linear continuous functions on a general (not necessarily uniform) mesh $\mathcal{M}$ of $[0,1]$. In the following, use tent functions as a basis. Note that the values of the solution in the boundary points $x = 0$, $x = 1$ are **not** fixed. In fact, this would not make any sense.

(2.3a) What is the Galerkin matrix for this problem? Write a function

```
A = galmatrix_tent(mesh)
```

which takes the mesh points (including the boundary points) as input in the row vector `mesh`, computes the Galerkin matrix, and returns it in the sparse matrix `A`.

HINT: The cell sizes h_i must be taken into account.

(2.3b) Write a function

```
L = rhs_tent(mesh,f)
```

which takes as input the same `mesh` vector and a function handle for f, computes the right-hand side vector, and returns it in the column vector `L`. Use the composite trapezoidal rule for numerical quadrature.

HINT: Assume that `f` can take vector arguments to compute f at each mesh point quickly.

(2.3c) Write a function

```
U = l2proj_tent(mesh,f)
```

that solves (2.3.1) approximately based on linear finite element Galerkin discretization. The arguments `mesh` and `f` are the same as before. The column vector `U` should contain the value of the solution at each mesh point.

(2.3d) Investigate the convergence of the method from Problem 2.3 with equidistant mesh points for $f(x) = \sqrt{x}$ in the L^∞- and L^2-norms. Compute these norms by sampling on a finer mesh or using composite 2-point Gauss quadrature on $\mathcal{M}$, as in (2.3a) and (2.3b). Use $N = 10 \cdot 2^r$ degrees of freedom, for $r = 1, 2, \ldots, 9$. Which is the rate of convergence? Give an explanation about the result that you get.

HINT: The exact solution is $u(x) = \sqrt{x}$. For $N = 10$, the L^∞-error should be around 0.222, and the L^2-error should be around 0.059.

HINT: Note that $f'(x)$ is large for x close to zero and even singular at $x = 0$, for a different f you would observe other results.

Listing 2.5: Testcalls for Problem 2.3

```
fprintf('\n\n##galmatrix_tent:')
full(galmatrix_tent([0; 0.1; 0.7; 1]))

fprintf('\n\n##rhs_tent:')
rhs_tent([0; 0.1; 0.7; 1], @(x) x)'

fprintf('\n\n##l2proj_tent:')
l2proj_tent([0; 0.1; 0.7; 1], @(x) x)'
```

Listing 2.6: Output for Testcalls for Problem 2.3

```
>> test_call

```

```
##galmatrix_tent:
ans =

    0.0333    0.0167         0         0
    0.0167    0.2333    0.1000         0
         0    0.1000    0.3000    0.0500
         0         0    0.0500    0.1000

##rhs_tent:
ans =

         0    0.0350    0.3150    0.1500

##l2proj_tent:
ans =

    0.1386   -0.2771    0.9735    1.0133
```

Published on March 17.

To be submitted on March 31.

Last modified on March 24, 2014