Prof. Dr. Karsten Urban
Mazen Ali

Institute for Numerical Mathematics
Ulm University
Summer Term 2018

Numerical Finance – Sheet 9

(due 25.06.2018)

Exercise 1: General θ-schemes for finite elements

To understand how the Black-Scholes equation can be discretized by means of the finite elements method, we consider more generally the following parabolic model problem:

$$\begin{cases} \dfrac{\partial u}{\partial t}(t,x) - \dfrac{\partial^2 u}{\partial x^2}(t,x) = f(t,x), \quad \forall t \in (0,1],\ \forall x \in (0,1), \\ u(0,x) = 0, \quad \forall x \in [0,1], \\ u(t,0) = u(t,1) = 0, \quad \forall t \in (0,1]. \end{cases} \tag{1}$$

Based on a one-dimensional grid $\{0 = x_0, x_1, x_2, \ldots, x_N, x_{N+1} = 1\}$, we define the finite element space

$$S_h = \operatorname{span}\{\psi_i :\ 1 \le i \le N\}, \quad \psi_i := \begin{cases} \frac{x - x_{i-1}}{x_i - x_{i-1}}, & x_{i-1} \le x \le x_i, \\ \frac{x_{i+1} - x}{x_{i+1} - x_i}, & x_i \le x \le x_{i+1}, \\ 0, & \text{else.} \end{cases}$$

The standard procedure to discretize (1) consists of first discretizing the problem in the space variable x, and afterwards one discretizes the resulting *semi-discrete* problem in the time variable t.

a) To derive the semi-discrete problem, we make the following separation ansatz

$$u_h(t,x) := \sum_{i=1}^{N} u_i(t)\psi_i(x), \quad \forall t \in [0,1],\ \forall x \in [0,1],$$

where u_i for $i = 1, \ldots, N$ are assumed to be continuously differentiable. Show that by multiplying by ψ_j $(j = 1, \ldots, N)$, integrating over x from 0 to 1 and finally replacing u in (1) by u_h, we obtain the semi-discrete problem

$$\begin{cases} M_h \underline{u}'(t) + A_h \underline{u}(t) = b_h(t), \quad t \in (0,1], \\ \underline{u}(0) = 0. \end{cases} \tag{2}$$

where $\underline{u}(t) := (u_1(t), \ldots, u_N(t))^T$, $\underline{u}'(t) := (u_1'(t), \ldots, u_N'(t))^T$,
$b_h(t) := \left(\int_0^1 f(t,x)\psi_1(x)\,dx, \ldots, \int_0^1 f(t,x)\psi_N(x)\,dx\right)^T$, $M_h = \left(\int_0^1 \psi_i(x)\psi_j(x)\,dx\right)_{i,j=1,\ldots,N}$ and
$A_h = \left(\int_0^1 \psi_i'(x)\psi_j'(x)\,dx\right)_{i,j=1,\ldots,N}$.

b) Obviously, (2) is an ordinary differential equation $\underline{y}'(t) = F(t, \underline{y}(t))$ where $F : [0,1] \times \mathbb{R}^{n-1} \to \mathbb{R}^N$ is supposed to be Lipschitz continuous. So, if we discretize the time interval $[0,1]$ by an equidistant grid $\{0 = t_0, t_1, \ldots, t_{m-1}, t_m = 1\}$ with mesh size Δt, a *full discretization* of (1) is obtained by the so called θ-scheme

$$\frac{1}{\Delta t}\left(\underline{y}(t_\nu) - \underline{y}(t_{\nu-1})\right) = \theta F(t_\nu, \underline{y}(t_\nu)) + (1-\theta)F(t_{\nu-1}, \underline{y}(t_{\nu-1})), \quad \nu = 1, \ldots, m,$$

for $\theta \in [0,1]$.
Show that the full discretization of (1) is

$$\begin{cases} [M_h + \theta \Delta t A_h]\, \underline{u}(t_\nu) = [M_h - \Delta t(1-\theta)A_h]\, \underline{u}(t_{\nu-1}) + \Delta t\, [\theta b_h(t_\nu) + (1-\theta) b_h(t_{\nu-1})] \\ \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \text{for } \nu = 1, \ldots, m \\ \qquad\qquad \underline{u}(t_0) = 0. \end{cases} \tag{3}$$

Programming Exercise 1: Elliptic FEM 1D (21 points)

Solve the 1D Poisson problem

$$\begin{aligned} -u_{xx} &= f(x) \qquad \text{on } \Omega = (0,1), \\ u(0) = g_0,\ u(1) &= g_1. \end{aligned}$$

using linear Lagrangian Finite Elements (the "hat basis"). On the homepage, you find header files for the following classes:

- `EllipticFEM1D`: Contains the finite element structures, i.e. the 1D mesh, the solution vector, information about the Dirichlet boundaries etc. Also has a reference to a class that provides the PDE which we want to solve.
- `PDE1D`: The (abstract) class for a PDE problem which we want to solve. This is nothing but an interface, specifying what a PDE problem class should look like. There can never be an object of this class, but we can derive different classes for different PDEs from this one. The abstract function `assemble_on_element` has to be implemented in each of these derived classes, so that it is guaranteed that this function always exists.
- `Poisson1D`: One example of a concrete PDE problem. This class provides the assembly of the Laplace operator on a given finite element (here in 1D specified by the interval boundaries x_i, x_{i+1}). In requires function pointers describing the parameters of the PDE: f for the right hand side and g for the Dirichlet boundaries (with $g(0) = g_0$, $g(1) = g_1$).

Moreover, you find a file `test_fem1d.cpp` that uses the above classes to set up a Poisson problem and solve this PDE.

Implement the missing function implementations and solve the problem for $f(x) = 1$, $g_0 = g_1 = 0$, as well as for $f(x) = 1$, $g(x) = -\frac{1}{2}x$ (with solution $u(x) = -\frac{1}{2}x^2$). Plot your solutions u.

Hints:

- ***Dirichlet boundary conditions:*** In order to homogenize the problem, we choose a function $g \in S_h$ as $g = g_0\psi_0 + g_1\psi_{N+1}$. Then $g(0) = g_0$, $g(1) = g_1$. The homogeneous equation system $Lw = f - Lg$ then reads in the discrete space

$$\sum_{j=1}^{N} u_j a(\psi_j, \psi_i) = (f, \psi_i)_0 - [g_0 a(\psi_0, \psi_i) - g_1 a(\psi_{N+1}, \psi_i)], \qquad i = 1, \ldots, N,$$

  and $u_h := \sum_{i=j}^{N} u_j \psi_j + g$ is a solution of the original boundary value problem. Note that most additional terms $g_0 a(\psi_0, \psi_i)$, $g_1 a(\psi_{N+1}, \psi_i)$ on the right hand side are zero. Which ones are not?

- ***Element-wise assembly:*** Each entry $A_{i,j}$, $i,j = 1,\dots,N$, in the stiffness matrix A_h can be decomposed into the contributions on the individual elements (intervals in 1D):

$$A_{i,j} = \int_0^1 \nabla\psi_i(x)\cdot\nabla\psi_j(x)dx = \sum_{k=0}^{N}\int_{x_k}^{x_{k+1}} \nabla\psi_i(x)\cdot\nabla\psi_j(x)dx =: \sum_{k=0}^{N} A_{i,j}^{(k)}.$$

  As $A_{i,j}^{(k)} \neq 0$ only for $i,j \in \{k, k+1\}$, we have to assemble on each interval only the 2×2-matrix $A^{(k)} := \begin{pmatrix} A_{k,k}^{(k)} & A_{k,k+1}^{(k)} \\ A_{k+1,k}^{(k)} & A_{k+1,k+1}^{(k)} \end{pmatrix}$, which can be done "by hand" (cf. lecture).
  Therefore, we can assemble A_h by once looping over all intervals, calculating the individual contributions $A^{(k)}$ and adding them to the corresponding entries in the global stiffness matrix A_h.

- ***Assembly of right-hand side:*** As for the stiffness matrix, we perform an element-wise assembly for each entry:

$$(f,\psi_i)_0 = \int_0^1 f(x)\psi_i(x)dx = \sum_{k=0}^{N}\int_{x_k}^{x_{k+1}} f(x)\psi_i(x)dx =: \sum_{k=0}^{N} b_i^{(k)}, \qquad i = 1,\dots,N.$$

  We use the trapezoidal rule to approximate the integrals: $b_i^{(k)} = \frac{f(x_k)\psi_i(x_k)+f(x_{k+1})\psi_i(x_{k+1})}{2}(x_{k+1}-x_k)$.
  Again, on each interval, we only have to compute the vector $b^{(k)} := \left(b_k^{(k)}, b_{k+1}^{(k)}\right)^T$, which is easily done using the nodal structure of our basis.

- ***Data structures:*** The linear system itself is only solved for the unknown coefficients u_j, $j = 1,\dots,N$ (the so-called *degrees of freedom*). However, we want a solution representation for **all** mesh points $\{x_0,\dots,x_{N+1}\}$, including the boundary. The easiest solution (which avoids unnecessary copy operations or separate structures for inner and boundary values) is to just assemble a system of size $(N+1)\times(N+1)$ and fill the rows and columns corresponding to Dirichlet boundary points with zeros.
  For our cg-solver, this causes no problem at all. Note, however, that one usually sets the diagonal entries $A_{i,i} = 1$ in the Dirichlet rows/columns, so that the matrix has full rank. This is important for some solvers and most preconditioners.

- ***General note:*** The provided structures are supposed to help you. They are sufficient for the solution of the exercises. However, if you prefer adding auxiliary functions, operators or variables – feel free to do so (even if we will probably ask you why).
  The material includes classes for dense and sparse matrices as well as the necessary operators. Of course, you can also use your own classes from Sheet 8.

Programming Exercise 2: Parabolic FEM 1D (21 points)

Solve the 1D heat equation

$$\begin{aligned} u_t - u_{xx} &= f(x) && \text{on } \Omega = (0,1), \\ u(t,0) = g_0,\ u(t,1) &= g_1, && \forall\, t \in [0,T], \\ u(0,x) &= u_0(x). \end{aligned}$$

using linear Finite Elements in space. Again, you find a header file `SimpleParabolicFEM1D` on the homepage. Complete the missing parts, so that you can solve a general θ-scheme.

You can assume that the elliptic operator part (here $-u_{xx}$) as well as the right hand side f are time independent. What would you have to change in the given structure in order to solve a general parabolic PDE with time-dependent $A = A(t,u)$ and $f = f(t,x)$?

Use your implementation to solve the heat equation for $f(x) = 1$, $g_0 = g_1 = u_0 = 0$ as well as for $f(x) = g(x) = 1$, $u_0(x) = x(1-x)+1$. Plot your solutions $u = u(t,x)$ in a 3D plot.

Hints:

- ***Mass matrix assembly:*** For the θ-scheme we now have to assemble not only the stiffness matrix A_h, but also the so-called *mass matrix* M_h. This matrix only depends on the given basis (and not on any problem-dependent parts), so this assembly can be done completely inside a function `SimpleParabolicFEM1D::assemble_M()`. Again, the integrals can be computed by hand.

Programming Exercise 3*: Black Scholes FEM 1D (6* points)

For the Black-Scholes Equation, we additionally need time-dependent boundary conditions. As they enter the right hand side of the equation system at each time step, we would like to be able to perform time-dependent matrix assemblies.
In order to do so, we now change the structure from Programming Exercise 2 in the following way:

- We extend the class `PDE1D` by a member variable `current_time` and now call it `TimedepPDE1D`. We also need a function `set_time()` to change that variable.

- The FEM model class is now called `ParabolicFEM1D` and has an object of type `TimedepPDE1D` as member. Note that it looks almost exactly like the `SimpleParabolicFEM1D`. The only difference is that we can now call

  ```
  pde1d.set_time(t);
  assemble_A();
  ```

  in each time step inside the θ-scheme and obtain assembled variables $A = A_h(t)$ and $F = b_h(t)$.

- Our PDE problem is now called `TimedepPoisson1D` and is derived from `TimedepPDE1D`. Note that all that has changed are the signatures of the internal functions pointers for f and g – these functions now take two parameters (time and space). They are called with the member variable `current_time` and the argument `x`.

You can now copy everything you have done in Programming Exercise 2 into `TimedepParabolicFEM1D` and `TimedepPoisson1D`. The only thing you have to adapt are the functions

- `TimedepParabolicFEM1D::solve()` : As $A_h(t)$ and $b_h(t)$ are now time-dependent, you have to modify the θ-scheme as indicated above to realize the time-dependent assembly.

- `TimedepPoisson1D::assemble_on_element(...)` : Replace each former call `f(x)` by the call `f(current_time,x)`.

Compile and run the test program `test_black_scholes.cpp` that sets up the problem for the European Put from Sheet 8. Plot the solution in a 3D plot.