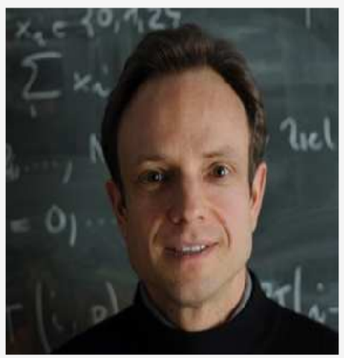
ETH Lecture 401-0674-00L Numerical Methods for Partial Differential Equations

Homework Problem Video Tutorial
Problem 6-2: Implicit Timestepping for Parabolic IBVP

Prof. R. Hiptmair, SAM, ETH Zurich

Date: April 19, 2019



Non-dimensional heat equation

$$\begin{aligned} &\frac{\partial u}{\partial t} - \Delta u = 0 && \text{in } \Omega \times [0,T]\,, && \text{(evolution equation)} \\ &-\mathbf{grad}\, u \cdot n = cu && \text{on } \partial\Omega \times [0,T]\,, && \text{(boundary conditions)} \\ &u(\mathbf{x},0) = u_0(\mathbf{x}) && \text{in } \Omega\,, && \text{(initial conditions)} \end{aligned} \tag{6.2.1}$$

(*) $c > 0$

↳ convective cooling b.c.

a, Spatial variational formulation

Test with $v \in H^1(\Omega)$ & integrate & i.b.p.

$u : [0,T] \longrightarrow H^1(\Omega)$:

$$\int_\Omega \dot{u}\, v \,\mathrm{d}\boldsymbol{x} + \int_\Omega \mathbf{grad}\, u \cdot \mathbf{grad}\, v \,\mathrm{d}\boldsymbol{x} - \int_{\partial\Omega} v\, \mathbf{grad}\, u \cdot \boldsymbol{n} \,\mathrm{d}\boldsymbol{x} = 0$$

$$\Rightarrow \underbrace{\int_\Omega \dot{u}\, v \,\mathrm{d}\boldsymbol{x}}_{m(\dot{u},v)} + \underbrace{\int_\Omega \mathbf{grad}\, u \cdot \mathbf{grad}\, v \,\mathrm{d}\boldsymbol{x} + \int_{\partial\Omega} cu\, v \,\mathrm{d}\boldsymbol{x}}_{a(u,v)} = \underbrace{0}_{\ell(v)}\,, \tag{6.2.3}$$

(*) $\forall v \in H^1(\Omega)$

$$m(\dot{u},v) + a(u,v) = \ell(t)(v)$$

m: s.p.d. $\quad$ a: s.p.d.

b, Argue why the total thermal energy [heat capacity $\mathcal{S} \equiv 1$]

"Heat content": $E(t) := \int_\Omega u(\mathbf{x},t) \,\mathrm{d}\boldsymbol{x}\,,$ (6.2.4)

decreases with time, if $u_0(\mathbf{x}) > 0$ for all $\mathbf{x} \in \Omega$.

(2) "Physical intuition": $u(x,t) \geq 0 \quad \forall (t,x)$

[$\Leftrightarrow$ Maximum principle for parabolic evolutions]

$$\frac{d}{dt}E(t) = \frac{d}{dt}\int_\Omega u(x,t)\,dx$$

$$= \int_\Omega \dot{u}(x,t)\,dx$$

[Use variational formulation with $v \equiv 1$]

$$= -c\int_{\partial\Omega} \underbrace{u(x,t)}_{\geq 0}\,dS(x) \leq 0$$

If $u_{|\partial\Omega} > 0 \Rightarrow t \to E(t)$ decrease.

c, Relevant bilinear forms

$$\mathsf{a}(u,v) := \int_\Omega \mathbf{grad}\, u \cdot \mathbf{grad}\, v \,\mathrm{d}\boldsymbol{x} + \int_{\partial\Omega} cuv \,\mathrm{d}\boldsymbol{x},$$

$$\mathsf{m}(u,v) := \int_\Omega u(\boldsymbol{x})v(\boldsymbol{x})\,\mathrm{d}\boldsymbol{x}.$$

$S_1^0(\mathcal{M})$ Galerkin FEM

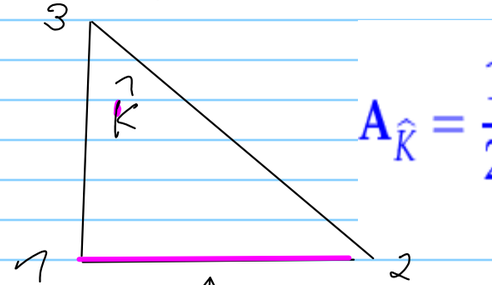
Here: Element mass matrix ($\Leftrightarrow \mathsf{m}(\cdot,\cdot)$) on unit triangle $\hat{K}$,

LSF $\hat{=}$ barycentric coordinate functions $\lambda_1, \lambda_2, \lambda_3$

$$M_K = \left[\int_K \lambda_i \lambda_j \,dx\right]_{i,j=1}^3$$

$$\mathbf{M}_{\hat{K}} = \frac{1}{24}\begin{pmatrix} 2 & 1 & 1 \\ 1 & 2 & 1 \\ 1 & 1 & 2 \end{pmatrix}. \tag{6.2.6}$$

d, $a(\cdot,\cdot) - \hat{K}$ element matrix



$$\mathbf{A}_{\hat{K}} = \frac{1}{2}\begin{pmatrix} 2 & -1 & -1 \\ -1 & 1 & 0 \\ -1 & 0 & 1 \end{pmatrix} + \frac{c}{6}\begin{pmatrix} 2 & 1 & 0 \\ 1 & 2 & 0 \\ 0 & 0 & 0 \end{pmatrix}.$$

edge $e \subset \partial\Omega$; "Δ-element matrix" ; edge contribution

f, Implement a LEHRFEM++-based C++ function (in the file `sdirkmethodoflines.cc`)

```
std::pair<Eigen::SparseMatrix<double>, Eigen::SparseMatrix<double>>
  assembleGalerkinMatrices(const lf::assemble::DofHandler &dofh,
    double c);
```

that assembles the $\mathcal{S}_1^0(\mathcal{M})$-Galerkin matrices for the bilinear forms $\mathsf{a}(\cdot,\cdot)$ and $\mathsf{m}(\cdot,\cdot)$ found in Sub-problem (6-2.a). The argument `dofh` is to provide a hybrid mesh $\mathcal{M}$ and information about the numbering of local and global basis functions. The coefficient $c > 0$ is given through `c`.

Your implementation can rely on the LEHRFEM++ library class `lf::uscalfe::LinearFELaplaceElementMatrix` and, of course, on `lf::assemble::assembleMatrix` see [Lecture → Code 2.7.4.23], which even accepts a co-dimension argument! You will also have to implement you own slightly extended version of the class `LinFEMassEdgeMatProvider` from [Lecture → Code 2.7.4.39].

An alternative is the use of `lf::uscalfe::ReactionDiffusionElementMatrixProvider` with suitable coefficient functions, see [Lecture → Ex. 2.8.3.28].

③ Implicit 2-stage RK-SSM : SDIRK-2

$$\begin{array}{c|cc} c & \mathcal{A} \\ \hline & b^T \end{array} \;\hat{=}\; \begin{array}{c|cc} \lambda & \lambda & 0 \\ 1 & 1-\lambda & \lambda \\ \hline & 1-\lambda & \lambda \end{array}, \quad \lambda := 1 - \tfrac{1}{2}\sqrt{2}\,. \tag{6.2.8}$$

Definition 6.2.6.31. General Runge-Kutta method →

For coefficients $b_i, a_{ij} \in \mathbb{R}$, $c_i := \sum_{j=1}^{s} a_{ij}$, $i,j = 1,\ldots,s$, $s \in \mathbb{N}$, the discrete evolution $\Psi^{s,t}$ of an s-stage Runge-Kutta single step method (RK-SSM) for the ODE $\dot{\mathbf{u}} = \mathbf{f}(t,\mathbf{u})$, is defined by

$$\mathbf{k}_i := \mathbf{f}(t + c_i\tau, \mathbf{u} + \tau \sum_{j=1}^{s} a_{ij}\mathbf{k}_j)\,, \quad i = 1,\ldots,s \quad , \quad \Psi^{t,t+\tau}\mathbf{u} := \mathbf{u} + \tau \sum_{i=1}^{s} b_i \mathbf{k}_i\,.$$

The $\mathbf{k}_i \in V_0$ are called increments.

↳ discrete evolution operator

g, SDIRK-2 for $\dot{y} = -\gamma y =: f(y)$ [scalar linear ODE]

Increment eqn. :

$$\kappa_1 = -\gamma y - \gamma\tau\lambda\kappa_1\,,$$
$$\kappa_2 = -\gamma y - \gamma\tau(1-\lambda)\kappa_1 - \gamma\tau\lambda\kappa_2$$

≈ 2×2 LSE

$$\triangleright \quad \kappa_1 = \frac{-\gamma y}{1+\gamma\tau\lambda}, \qquad \kappa_2 = \frac{-\gamma y(1+2\gamma\tau\lambda-\gamma\tau)}{(1+\gamma\tau\lambda)^2} \tag{6.2.9}$$

$$y^{(j+1)} = y^{(j)} + \tau(1-\lambda)\kappa_1 + \tau\lambda\kappa_2 \tag{6.2.10}$$

$$= y^{(j)}\left(1 - \frac{\gamma\tau(1+\gamma\tau\lambda^2)}{(1+\gamma\tau\lambda)^2}\right). \tag{6.2.11}$$

h, "Measuring" order of SDIRK-2 RK-SSM

⇕

= rate of alg. cvg.

Model problem : $\dot{y} = -\gamma$ on $[0,2]$, $y_0 = 1$

ℓ	m_ℓ #timesteps	error ε_ℓ	estimated rate
0	10	0.000446558	
1	20	0.000110501	2.01478
2	40	2.74922e-05	2.00697
⋮	80	6.85696e-06	2.00338
	160	1.71226e-06	2.00167
	320	4.2782e-07	2.00083
	640	1.06924e-07	2.00041
	1280	2.67273e-08	2.00021
	2560	6.68136e-09	2.0001
	5120	1.67028e-09	2.00005

↓ reduction by factor ≈ 4

Alg. cvg.

$$\varepsilon_\ell \approx C\, m_\ell^{-\alpha}$$

$$\log \varepsilon_\ell \approx \log C - \alpha \log m_\ell$$

$$\log \varepsilon_\ell/\varepsilon_{\ell-1} \approx -\alpha \log \underbrace{m_\ell/m_{\ell-1}}_{2}$$

$$\alpha \approx -\frac{\log \varepsilon_\ell/\varepsilon_{\ell-1}}{\log 2}$$

g, Show : SDIRK-2 is $L(\pi)$-stable

$$S(-\tau\gamma) := \left(1 - \frac{\gamma\tau(1+\gamma\tau\lambda^2)}{(1+\gamma\tau\lambda)^2}\right) \Leftrightarrow S(z) = 1 - \frac{-z(1-z\lambda^2)}{(1-z\lambda)^2}\,. \tag{6.2.16}$$

↳ Stability function

(4)

$L(\Pi)$-stable $\Leftrightarrow$ $|S(z)| < 1 \quad \forall z < \infty$ ✓

$\lim_{z \to -\infty} S(z) = 0$ ✓

⏸

7.1 SDIRK-2 for MOL-ODE

$$\mathbf{M}\left\{\frac{d}{dt}\vec{\mu}(t)\right\} + \mathbf{A}\vec{\mu}(t) = \vec{\varphi}(t) \quad \Leftrightarrow \quad \dot{\vec{\mu}} = \underbrace{\mathbf{M}^{-1}(\vec{\varphi}(t) - \mathbf{A}\vec{\mu}(t))}_{=\mathbf{f}(t,\vec{\mu})} . \tag{7.1.4.4}$$

apply RK-SSM formulas

Increment equations LSE

⏸

$$\text{(7.1.7.7)} \quad \Leftrightarrow \quad (\mathbf{I}_s \otimes \mathbf{M} + \tau \mathfrak{A} \otimes \mathbf{A}) \begin{bmatrix} \vec{\kappa}_1 \\ \vdots \\ \vec{\kappa}_s \end{bmatrix} = \begin{bmatrix} \vec{\varphi}(t_j + c_1\tau) - \mathbf{A}\vec{\mu}^{(j)} \\ \vdots \\ \vec{\varphi}(t_j + c_s\tau) - \mathbf{A}\vec{\mu}^{(j)} \end{bmatrix} . \tag{6.2.7.9}$$

↑ Kronecker product of matrices

$$\begin{aligned} \mathbf{M}\vec{\kappa}_1 + \tau\lambda\mathbf{A}\vec{\kappa}_1 &= -\mathbf{A}\vec{\mu}^{(j)} , \\ \mathbf{M}\vec{\kappa}_2 + \tau(1-\lambda)\mathbf{A}\vec{\kappa}_1 + \tau\lambda\mathbf{A}\vec{\kappa}_2 &= -\mathbf{A}\vec{\mu}^{(j)} . \end{aligned} \tag{6.2.19}$$

$\hat{=}$ A staggered LSE

k, Compute, $v_h \in S_1^0(\mathcal{M})$

$$\int_\Omega v_h \, dx = \sum_K \sum_{\ell=1}^{3} v_h(a_K^\ell) \int_K \lambda_\ell \, dx$$

↑ entries of coefficient vector $\hat{=}$ of v_h

$= \frac{1}{3}|K|$

C++11 code 6.2.22: Sub-problem (6-2.k), function `thermalEnergy()` ↪ GITLAB

```
double thermalEnergy(const lf::assemble::DofHandler &dofh,
                     const Eigen::VectorXd &temperature_vec) {
  double thermal_energy = 0.0;
  /* SOLUTION_BEGIN */
  auto mesh_p = dofh.Mesh();  // pointer to mesh

  // The thermal energy of the system is defined as the integral of the
  // temperature solution over the domain. It is computed for linear
     lagrangian
  // finite elements using the trapezoidal rule by summing up the
     contribution
  // of that quadrature rule over each triangle
  double thermal_energy_loc;
  for (const lf::mesh::Entity &tria : mesh_p->Entities(0)) {
    thermal_energy_loc = 0.0;
    // Compute the area of the triangle
    const double area = lf::geometry::Volume(*(tria.Geometry()));
    // Obtain the global indices of the nodal degrees of freedom
    for (const lf::assemble::gdof_idx_t &g_idx :
         dofh.GlobalDofIndices(tria)) {
      thermal_energy_loc += temperature_vec[g_idx];
    }
    // Sum local contribution of quadrature rule
    thermal_energy += thermal_energy_loc * area / 3.0;
  }
  /* SOLUTION_END */
  return thermal_energy;
}  // thermalEnergy
```

(5)

(a)

In the file `sdirkmethodoflines.cc` implement a C++ function

```
std::pair<Eigen::VectorXd, Eigen::VectorXd>
  solveTemperatureEvolution(const lf::assemble::DofHandler &dofh,
  unsigned int m, double cool_coeff,
  Eigen::VectorXd initial_temperature_vec)
```

[→ $c \in \mathbb{R}$]

that carries out m equal timesteps of the SDIRK-2 in order to solve (6.2.1) over the time interval $[0,1]$. The argument `initial_temperature_vec` provides the initial conditions for the method-of-lines ODE, *cf.* [Lecture → Eq. (6.2.4.4)], the parameter `cool_coeff` the coefficient $c \in \mathbb{R}$. The discretization in space relies on $\mathcal{S}_1^0(\mathcal{M})$-FEM.

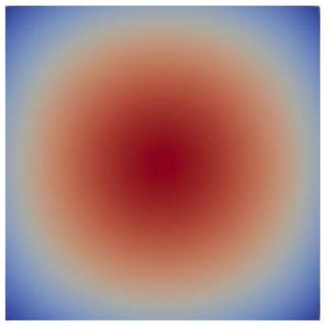
The function should return the basis expansion ocefficient vector of the solution at final time $T = 1$ and the "energies" (6.2.25) for times τj, $j = 0, \ldots, m$, where $\tau := 1/m$ is the constant timestep size.

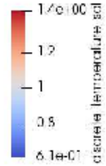
In every time step we have to solve

$$\begin{aligned} \mathbf{M}\vec{\kappa}_1 + \tau\lambda\mathbf{A}\vec{\kappa}_1 &= -\mathbf{A}\vec{\mu}^{(j)}, \\ \mathbf{M}\vec{\kappa}_2 + \tau(1-\lambda)\mathbf{A}\vec{\kappa}_1 + \tau\lambda\mathbf{A}\vec{\kappa}_2 &= -\mathbf{A}\vec{\mu}^{(j)}. \end{aligned} \tag{6.2.19}$$

▷ Two $N \times N$ LSEs, $N := \dim \mathcal{S}_1^0(\mathcal{M})$ with the same system matrix $\mathbf{M} + \tau\lambda\mathbf{A}$: [the same for all timesteps] ↑ fixed

▷ Assemble $\mathbf{A}$, $\mathbf{M}$ & LU-factorize $\mathbf{M} + \tau\lambda\mathbf{A}$ before entering the timestepping loop.





Δ μ at $T = 1$

$\mu_0 \equiv 1$

▷ Material has cooled close to 293.