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

Numerical Methods for Partial Differential Equations

Prof. R. Hiptmair, SAM, ETH Zurich

Spring Term 2020

Q & A Session May 18

HW 2-13

In attempt to solve the problem 2.13 b) and then understand the solution I have faced the following misunderstanding: In the problem we should implement the following general formula for the element matrix:

$$(\mathbf{A}_K)_{i,j} = \frac{1}{\sharp\mathcal{V}(K)} \sum_{\ell=1}^{\sharp\mathcal{V}(K)} (\mathbf{I} + \boldsymbol{d}(\boldsymbol{\Phi}(\widehat{\boldsymbol{m}}_\ell)\boldsymbol{d}(\boldsymbol{\Phi}(\widehat{\boldsymbol{m}}_\ell))^\top)(\mathrm{D}\boldsymbol{\Phi}(\widehat{\boldsymbol{m}}_\ell))^{-\top} \mathbf{grad}\, \widehat{b}^i(\widehat{\boldsymbol{m}}_\ell)) \cdot ((\mathrm{D}\boldsymbol{\Phi}(\widehat{\boldsymbol{m}}_\ell))^{-\top} \mathbf{grad}\, \widehat{b}^j(\widehat{\boldsymbol{m}}_\ell))\, |\det \mathrm{D}\boldsymbol{\Phi}(\widehat{\boldsymbol{m}}_\ell)|\,,$$

The code of the solution includes the following for the gradients of basis functions calculation:

```
Eigen::Matrix<double, 2, 3> gradients_ref(2, 3);
gradients_ref << -1, 1, 0, -1, 0, 1;
```

```
Eigen::MatrixXd gradients_param(2, 3);
gradients_param = JinvT.block(0, 2 * i, 2, 2) * gradients_ref;
```

I do not fully understand how this code implements the formula these gradients are calculated with? (I see that they should implement here DΦ^{-T}grad(b^{i}))

Variational problem: $\forall w \in S^0_1(\mathcal{M})$

$$\int_\Omega (\mathbf{I} + \boldsymbol{d}(\boldsymbol{x})\boldsymbol{d}(\boldsymbol{x})^\top)\, \mathbf{grad}\, u \cdot \mathbf{grad}\, v\, \mathrm{d}\boldsymbol{x} + \int_{\partial\Omega} w^2(\boldsymbol{x}) u(\boldsymbol{x})\, v(\boldsymbol{x})\, \mathrm{d}S(\boldsymbol{x}) = \int_\Omega \frac{v(\boldsymbol{x})}{1 + w(\boldsymbol{x})^2}\, \mathrm{d}\boldsymbol{x} \quad \forall v \in H^1(\Omega)\,, \tag{2.13.1}$$

2-13_01_ec

$d : \Omega \to \mathbb{R}^2$ continuous on $\overline{\Omega}$

$w : \Omega \to \mathbb{R}$

Edge midpoint quadrature

$$\int_K \varphi(\boldsymbol{x})\,\mathrm{d}\boldsymbol{x} \approx \frac{|K|}{\sharp\mathcal{V}(K)} \sum_{\ell=1}^{\sharp\mathcal{V}(K)} \varphi(\boldsymbol{m}_K^\ell)\,, \quad K \in \mathcal{M}\,. \tag{2.13.2}$$

2-13_03_eq:el

C++11 code 2.13.3: Sub-problem (2-13.b): Implementation of `Eval()` member function of AnisotropicDiffusionElementMatrixProvider *for triangles* ↪ GITHUB

```
// I. OBTAIN COORDINATES OF THE MIDPOINTS
// I.i Hard-code the midpoints of the edges on the reference triangle
Eigen::Matrix<double, 2, 3> midpoints_ref(2, 3);
midpoints_ref << 0.5, 0.5, 0, 0, 0.5, 0.5;
// I.ii Obtain the midpoints of the parametrized triangle
auto midpoints_param = cell_geometry->Global(midpoints_ref);

// II. COMPUTE LOCAL INTEGRATION DATA
// II.i Compute the inverse of the tranposed Jacobian
//      (J^T)^{-1} = J*(J^T*J)^{-1}
// of the parametrization map at the midpoints of the reference triangle
const Eigen::MatrixXd JinvT(
    cell_geometry->JacobianInverseGramian(midpoints_ref));
// II.ii Compute the integration element
//      integration_element(x) = sqrt(det(J^T*J))
// where J is the Jacobian of the parametrization map
const Eigen::VectorXd integration_element(
    cell_geometry->IntegrationElement(midpoints_ref));
// II.iii Hard-code the gradients of the reference basis functions
Eigen::Matrix<double, 2, 3> gradients_ref(2, 3);
gradients_ref << -1, 1, 0, -1, 0, 1;

// III. PERFORM NUMERICAL QUADRATURE
element_matrix = Eigen::MatrixXd::Zero(3, 3);
for (int i = 0; i < 3; i++) {  // for each local degree of freedom
  // III.i Evaluate the diffusion tensor
  Eigen::Vector2d anisotropy_vec =
      anisotropy_vec_field_(midpoints_param.col(i));
  Eigen::Matrix2d diffusion_tensor =
      Eigen::Matrix2d::Identity(2, 2) +
      anisotropy_vec * anisotropy_vec.transpose();
  // III. ii Compute gradients of the global basis functions
  Eigen::MatrixXd gradients_param(2, 3);
  gradients_param = JinvT.block(0, 2 * i, 2, 2) * gradients_ref;
  // III. iii Compute local contribution to the element matrix
  for (int j = 0; j < 3; j++) {
    for (int k = 0; k < 3; k++) {
      double integrand = (gradients_param.col(j).transpose() *
                          diffusion_tensor * gradients_param.col(k));
      element_matrix(j, k) += integration_element(i) * integrand;
    }
  }
}
element_matrix *= 1. / 6.;
break;
```

2-13_05_cpp:ademp

Δ Only for triangles

} grad $\mathcal{F}_\ell$, $\ell = 1,2,3$

→ Loop over midpoints

Δ grad $\mathcal{F}_\ell$, $\ell = 1,2,3$

[13:40]

$$(\mathbf{A}_K)_{i,j} = \frac{1}{\sharp\mathcal{V}(K)} \sum_{\ell=1}^{\sharp\mathcal{V}(K)} \left(\mathbf{I} + d(\boldsymbol{\Phi}(\widehat{\boldsymbol{m}}_\ell) d(\boldsymbol{\Phi}(\widehat{\boldsymbol{m}}_\ell))^\top\right) (\mathrm{D}\boldsymbol{\Phi}(\widehat{\boldsymbol{m}}_\ell))^{-\top} \mathbf{grad}\, \widehat{b}^i(\widehat{\boldsymbol{m}}_\ell)) \cdot ((\mathrm{D}\boldsymbol{\Phi}(\widehat{\boldsymbol{m}}_\ell))^{-\top} \mathbf{grad}\, \widehat{b}^j(\widehat{\boldsymbol{m}}_\ell))\, |\det \mathrm{D}\boldsymbol{\Phi}(\widehat{\boldsymbol{m}}_\ell)|\,,$$

Lemma 2.8.3.10. Transformation formula for gradients

For differentiable $u : K \mapsto \mathbb{R}$ and any diffeomorphism $\boldsymbol{\Phi} : \widehat{K} \mapsto K$ we have

2_8_31_lem:Gtr

$$(\mathbf{grad}_{\widehat{\boldsymbol{x}}}(\boldsymbol{\Phi}^* u))(\widehat{\boldsymbol{x}}) = (\mathrm{D}\boldsymbol{\Phi}(\widehat{\boldsymbol{x}}))^\top \underbrace{(\mathbf{grad}_{\boldsymbol{x}} u)(\boldsymbol{\Phi}(\widehat{\boldsymbol{x}}))}_{=\boldsymbol{\Phi}^*(\mathbf{grad}\, u)(\widehat{\boldsymbol{x}})} \quad \forall \widehat{\boldsymbol{x}} \in \widehat{K}\,. \tag{2.8.3.11}$$

Lecture 8.2.2: Characteristics

(Conservation law (scalar, 1D)): $\partial_t u + \partial_x f(u) = 0$

Characteristic: $\Gamma := (\gamma(\tau), \tau)$

$$\frac{d\gamma}{d\tau}(\tau) = f'(u(\gamma(\tau), \tau))$$

Definition 8.2.2.3. Characteristic curve for one-dimensional scalar conservation law

A curve $\Gamma := (\gamma(\tau), \tau) : [0, T] \mapsto \mathbb{R}\times]0, T[$ in the (x, t)-plane is a characteristic curve for the conservation law (8.2.2.1), if

8_2_18_def:char1

$$\frac{d}{d\tau}\gamma(\tau) = f'(u(\gamma(\tau), \tau)) \,, \quad 0 \le \tau \le T \,, \tag{8.2.2.4}$$

where u is a continuously differentiable solution of (8.2.2.1).

Lemma 8.2.2.6. Classical solutions and characteristic curves

8_2_21_lem:solcha

Smooth solutions of (8.2.2.1) *are constant along characteristic curves.*

$$\frac{d}{d\tau}u(\gamma(\tau), \tau) \overset{\text{chain rule}}{=} \frac{\partial u}{\partial x}(\gamma(\tau), \tau)\frac{d}{d\tau}\gamma(\tau) + \frac{\partial u}{\partial t}(\gamma(\tau), \tau)$$

$$\overset{(8.2.2.4)}{=} \frac{\partial u}{\partial x}(\gamma(\tau), \tau) \cdot f'(u(\gamma(\tau), \tau)) + \frac{\partial u}{\partial t}(\gamma(\tau), \tau)$$

8_2_22_proofsolch

$$\overset{\text{chain rule}}{=} \left(\frac{\partial}{\partial x} f(u)\right)(\gamma(\tau), \tau) + \frac{\partial u}{\partial t}(\gamma(\tau), \tau) = 0 \,.$$

Application of chain rule:

$$u : \mathbb{R}^2 \to \mathbb{R} : u = u(x, t)$$

$$\psi : \mathbb{R} \to \mathbb{R}^2 : \tau \mapsto \begin{bmatrix} \gamma(\tau) \\ \tau \end{bmatrix}$$

$$\frac{d}{d\tau} u(\psi(\tau)) = Du(\psi(\tau))\frac{d\psi}{d\tau}(\tau)$$

$$= \begin{bmatrix} \frac{\partial u}{\partial x}(\psi(\tau)) & \frac{\partial u}{\partial t}(\psi(\tau)) \end{bmatrix} \begin{bmatrix} \frac{d\gamma}{d\tau} \\ 1 \end{bmatrix}$$

$$f : \mathbb{R} \longrightarrow \mathbb{R}, \quad f = f(u)$$

$$u : \mathbb{R}^2 \longrightarrow \mathbb{R}, \quad u = u(x,t)$$

$$\frac{\partial}{\partial x} f(u(x,t)) = f'(u(x,t)) \frac{\partial u}{\partial x}(x,t)$$