IN5270

December 16, 2019

1 IN5270 - Exam preparation

1.1 Topic 1

Want to approximate the function $f(x) = 1 + 2x - x^2$ in the domain $x \in [0, 1]$ by the projection method and by using finite element basis functions.

###Task 1 Single P2 element.

Found by Lagramge polynomials

$$\varphi_i(x) = \prod_{j=0, j\neq i}^{n} \frac{x - x_j}{x_i - x_j}$$

and the resulting φ_i are

$$\varphi_0 = (1 - 2x)(1 - x)$$
$$\varphi_1 = 2x(2 - 2x)$$
$$\varphi_2 = x(2x - 1)$$

We approximate the function f as

$$u = \sum_{i}^{n} c_i \varphi_i$$

and define the residual of f and u as $R = f - u$. Want the residual to be orthogonal to each basis function in V (containing the basis functions), which leads to

$$(R, v) = (f - u, v) = 0$$
$$(f, v) = (u, v)$$

Leads to the linear system $Ac = b$ with

$$A_{i,j} = \int_0^1 \varphi_i \varphi_j dx$$
$$b_i = \int_0^1 f(x) \varphi_i dx$$

The code below solves the system.

[0]:

```python
#@title
import numpy as np
import sympy as sym
from google.colab.output._publish import javascript
url = "https://cdnjs.cloudflare.com/ajax/libs/mathjax/2.7.3/latest.js?
↪config=default"
javascript(url=url)
sym.init_printing(use_unicode=True)


x = sym.Symbol('x')
n = 3
phi = [(1 - 2*x)*(1 - x), 2*x*(2 - 2*x), x*(2*x - 1)]

f = 1 + 2*x-x**2


A = sym.zeros(n, n)
b = sym.zeros(n)

for i in range(n):
    for j in range(n):
        A[i, j] = sym.integrate(phi[i]*phi[j], (x, 0, 1))
    b[i] = sym.integrate(phi[i]*f, (x, 0, 1))
print('A:')
A
```

```
<IPython.core.display.HTML object>
```

```
A:
```

[0]:

$$\begin{bmatrix} \frac{2}{15} & \frac{1}{15} & -\frac{1}{30} \\ \frac{1}{15} & \frac{8}{15} & \frac{1}{15} \\ -\frac{1}{30} & \frac{1}{15} & \frac{2}{15} \end{bmatrix}$$

[0]:

```python
#@title
from google.colab.output._publish import javascript
url = "https://cdnjs.cloudflare.com/ajax/libs/mathjax/2.7.3/latest.js?
↪config=default"
javascript(url=url)
sym.init_printing(use_unicode=True)
print('b:')
b[0:3]
```

```
<IPython.core.display.HTML object>
```

b:

[0]:

$$\left[\frac{11}{60}, \quad \frac{17}{15}, \quad \frac{7}{20} \right]$$

[0]:
```
#@title
b = sym.Matrix([b[0], b[1], b[2]])
Ab = A.col_insert(3, b)
c0, c1, c2 = sym.symbols('c1, c2, c3')
c = sym.solve_linear_system_LU(Ab, [c0, c1, c2])
from google.colab.output._publish import javascript
url = "https://cdnjs.cloudflare.com/ajax/libs/mathjax/2.7.3/latest.js?
 ↪config=default"
javascript(url=url)
sym.init_printing(use_unicode=True)
print('solve Ac=b gives c:')
c
```

<IPython.core.display.HTML object>

solve Ac=b gives c:

[0]:

$$\left\{c_1 : 1, \quad c_2 : \frac{7}{4}, \quad c_3 : 2\right\}$$

1.1.1 Task 2

Use two P1 elements

The P1 basis functions are defined as $\varphi_i =$

$$\begin{array}{cc} 0 & x < x_{i-1} \\ (x - x_{i-1})/h & x_{i-1} \leq x < x_i \\ 1 - (x - x_{i-1})/h & x_i \leq x < x_{i+1} \\ 0 & x \geq x_{i+1} \end{array}$$

They can also be found by the Lagrange polynomials as in task 1. We get:

$\varphi_0 =$

$$\begin{array}{cc} 0 & x < 0 \\ -2x + 1 & 0 \leq x < 0.5 \\ 0 & x \geq 0.5 \end{array}$$

$\varphi_1 =$

$$\begin{array}{cc} 0 & x < 0 \end{array}$$

$$\begin{matrix} 2x & 0 \leq x < 0.5 \\ -2x+2 & 0.5 \leq x < 1 \\ 0 & x \geq 1 \end{matrix}$$

$\varphi_2 =$

$$\begin{matrix} 0 & x < 0.5 \\ 2x-1 & 0.5 \leq x < 1 \\ 0 & x \geq 1 \end{matrix}$$

1.1.2 Task 3

The general function for the basis functions is given above, and each element matrix/vector is found using these formulas (that is also given above):

$$A_{i,j} = \int_0^1 \varphi_i \varphi_j dx$$
$$b_i = \int_0^1 f(x)\varphi_i dx$$

The element matrices and vectors are then assembled into one complete matrix/vector, where each element matrix/vector have one entry overlap. Then we have a linear system $Ac = b$.

1.1.3 Task 4

Question: If we want in addition that the approximation result, when using N equal-sized P1 elements, should attain the same value of $f(x)$ at $x = 0$ and $x = 1$, what are the changes needed in the calculation above?

Answer: We have

- $f(0) = 1$
- $f(1) = 2$.

We add a term to u that leads to correct boundary values:

$$u(x) = B(x) + \sum_{i=0}^{N} c_i \varphi_i$$

More specifically:

$$u(x) = f(0)(1-x) + xf(1) + \sum_{i=0}^{N} c_i \varphi_i$$
$$u(x) = 1 + x + \sum_{i=0}^{N} c_i \varphi_i$$

1.2 Topic 2

We have the 1D Poisson equation:

$$-u_{xx} = 1, 0 < x < 1,$$

and we shall solve it with a finite difference method. On the left boudnary point of $x = 0$ we have the following mixed boudnary condition

$$u_x + Cu = 0,$$

where C is a constant. On the right boundary point of $x = 1$, the Dirichlet boundary condition $u = D$ is valid. We assume that we use a uniform mesh of $N + 1$ points.

1.2.1 Task 1

Question: Discretize the Poisson equation on all the $N-1$ interior points.

Answer: We discretize the double derivative like this:

$$-\frac{u^{n+1} - 2u^n + u^{n-1}}{\Delta x^2} = 1$$

1.2.2 Task 2

Question: Discretize the left boundary condition using appropriate finite differencing.

Answer:

We use the Neumann condition at $x = 0$: $u_x + Cu = 0$, and use a centered difference to find the undefined value of u^{-1}.

$$\frac{u^1 - u^{-1}}{2\Delta x} + Cu^0 = 0$$

$$u^{-1} = u^1 + 2\Delta x C u^0$$

1.2.3 Task 3

Question: Show the details of setting up a linear system $Au = b$ which can be used to find the approximations of $u(x)$ on the mesh points. (There's no need to solve the linear system.)

Answer: We reformulate the last equation as

$$-u^{n-1} + 2u^n - u^{n+1} = \Delta x^2$$

Based on this we make a linear system $Au = b$, as an example with $N = 4$ (we add the boundary conditions into vector b):

$$\begin{bmatrix} 2 & -1 & 0 & 0 \\ -1 & 2 & -1 & 0 \\ 0 & -1 & 2 & -1 \\ 0 & 0 & -1 & 2 \end{bmatrix} \begin{bmatrix} u_0 \\ u_1 \\ u_2 \\ u_3 \end{bmatrix} = \begin{bmatrix} \Delta x^2 + u^1 + 2\Delta x C u^0 \\ \Delta x^2 \\ \Delta x^2 \\ \Delta x^2 + D \end{bmatrix}$$

1.2.4 Task 4

Question: How would you validate that the obtained numerical solutions converge towards the exact solution, when the number of mesh points is increased? What is the expected convergence speed?

Answer: We assume that a measure of the numerical error E is related to the discretization parameters through

$$E = C\Delta x^r,$$

where C is a constant. We expect $r = 2$ because the error term are of order Δx^2. The difference between the exact solution u_e and the numerical u:

$$e_n = u_e(x_n) - u_n$$

,

and we typically use the L_2 norm as a measure of the error:

$$E = ||e_n||_{L_2} = \left(\Delta x \sum_{n=0}^{N} (e_n)^2 \right)^{\frac{1}{2}}$$

We let index i be our iteration index for each time we increase the number of mesh points, and we have

$$E_i = C\Delta x_i^r$$
$$E_{i+1} = C\Delta x_{i+1}^r$$

We divide these two equations:

$$\frac{E_i}{E_{i+1}} = \frac{C\Delta x_i^r}{C\Delta x_{i+1}^r}$$
$$r = \frac{\ln(E_i/E_{i+1})}{\ln(\Delta x_i/\Delta x_{i+1})}$$

1.3 Topic 3

We have the following 1D stationary convection diffusion equation

$$u_x = \varepsilon u_{xx}$$

We will solve it by finite differencing in the domain $0 < x < 1$, where $\varepsilon > 0$ is a given constant and the boundary conditions $u(0) = 0$ and $u(1) = 1$.

1.3.1 Task 1

Question: Show that the analytical solution is

$$u(x) = \frac{1 - e^{x/\epsilon}}{1 - e^{1/\epsilon}}$$

Answer:

We assume a solution of the form $e^{\lambda x}$ and insert into the DE

$$-\epsilon\lambda^2 e^{\lambda x} + \lambda e^{\lambda x} = 0$$

Which gives

$$u(x) = c_1 e^{x/\epsilon} + c_2$$

Using BC's, we find that $c_1 = -c_2$ and $c_1 = 1/(e^{1/\epsilon} - 1)$. Inserting then gives the expected result.

OR insert the function into the DE and see that it is the same.

1.3.2 Task 2

Question: Set up the linear system that solves the discretized equations.

Answer: Using centered difference, the DE is

$$\frac{u^{n+1} - u^{n-1}}{2\Delta x} = \epsilon \frac{u^{n+1} - 2u^n + u^{n-1}}{\Delta x^2}$$

which leads to

$$u^{n+1}(1-\gamma) + u^n(2\gamma) + u^{n-1}(-1-\gamma) = 0, \qquad \gamma = \frac{2\epsilon}{\Delta x}$$

The resulting linear system is then

$$\begin{pmatrix} 1 & 0 & \dots & \dots & \dots & 0 \\ -1-\gamma & 2\gamma & 1-\gamma & 0 & \dots & 0 \\ 0 & -1-\gamma & 2\gamma & 1-\gamma & \dots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & \dots & \dots & -1-\gamma & 2\gamma & 1-\gamma \\ 0 & \dots & \dots & \dots & 0 & 1 \end{pmatrix} \begin{pmatrix} u_0 \\ u_1 \\ \vdots \\ \vdots \\ \vdots \\ u_n \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ \vdots \\ \vdots \\ 0 \\ 1 \end{pmatrix}$$

where $A_{0,0}$ and $A_{n,n}$ have entries 1 since the solutions u_0 and u_n are known.

1.3.3 Task 3

Solve for u^{n+1}, insert and then show that the two sides of the equation is equal. Use boudnary conditions to find C_1 and C_2.

1.3.4 Task 4

1.4 Topic 4

We have the nonlinear diffusion equation in multiple space dimensions:

$$\frac{\partial u}{\partial t} = \nabla \cdot (\alpha(x,t)\nabla u) + f(u)$$

- $x \in \Omega$,
- $t \in (0, T]$,
- $u(x,0) = I(x), x \in \Omega$,
- $\frac{\partial u}{\partial n} = g, x \in \partial\Omega, t \in (0, T]$.

Note that $\frac{\partial u}{\partial n}$ denotes the outward normal derivative on the boudnary $\partial\Omega$, and g is a constant.

1.4.1 Task 1

Question: Use the Crank-Nicolson scheme in time and show the resulting time discrete problem for each time step.

Answer: Crank-Nicolson tine discretization:

$$\frac{\partial u}{\partial t} \approx \frac{u^{n+1} - u^n}{\Delta t} = \frac{1}{2}\left[\nabla(\alpha(x, t_{n+1})\nabla u^{n+1} + f(u^{n+1}) + \nabla(\alpha(x, t_n)\nabla u^n + f(u^n)\right]$$

We isolate u^{n+1}:

$$u^{n+1} = u^n + \frac{\Delta t}{2}\left[\nabla(\alpha(x, t_{n+1})\nabla u^{n+1} + f(u^{n+1}) + \nabla(\alpha(x, t_n)\nabla u^n + f(u^n)\right]$$

1.4.2 Task 2

Question: Formulate Picard iterations to linearize the time discrete problem.

We set

- $u^{n+1,k+1} = u$,
- $u^n = u^{(1)}$,
- $u^{n+1,k} = u^-$,
- $\alpha^{n+1} = \alpha$
- $\alpha^n = \alpha^{(1)}$

$$u = u^{(1)} + \frac{\Delta t}{2}\left[\nabla \cdot (\alpha \nabla u) + f(u^{(1)}) + \nabla \cdot (\alpha^{(1)} \nabla u^{(1)} + f(u^{-})\right]$$

1.4.3 Task 3

Question: Use the Galerkin method to discretize the stationary linear PDE per Picard iteration. Show the details of how to derive the corresponding variational form.

Answer: Residual:

$$R = u - u^{(1)} + \frac{\Delta t}{2}\left[\nabla \cdot (\alpha \nabla u) + f(u^{(1)}) + \nabla \cdot (\alpha^{(1)} \nabla u^{(1)} + f(u^{-})\right]$$

1.4.4 Task 4

Question: Restrict now the spatial domain to the 1D case of $x \in (0,1)$, let α be a constant and choose $f(u) = u^2$. (The boundary conditions are now $u_x = -g$ at $x = 0$ and $u_x = g$ at $x = 1$.) Suppose the 1D spatial domain consists of N equal-sized P1 elements. Carry out the calculation in detail for computing the element matrix and vector for the leftmost P1 element.

1.4.5 Task 5

Question: What is the resulting global linear system Ax = b?

1.5 Topic 5

2D Poisson equation:

$$-\nabla \cdot \nabla u = 2$$

- Defined in the unit square $(x, y) \in [0,1]^2$.
- Homogeneous Neumann condition: $\frac{\partial u}{\partial n} = 0$.

1.5.1 Task 1

Use the Galerkin method, derive the variational form of the above PDE in detail:

Integration by parts:

$$\int \nabla \cdot (\nabla u) v dx = \int_{\Omega} \nabla u \nabla v dx - \int_{\partial\Omega} \frac{\partial u}{\partial n} v dx$$

$$-\int_{\Omega} \nabla u \nabla dx v = 2 v dx.$$

$$(\nabla u, \nabla v) = (2, v)$$

1.5.2 Task 2

Question: What are the degrees of freedom and how many are they in total? How would you number the degrees of freedom, with respect to the rows in a global linear system to be set up?

Answer: The degrees of freedom are the values of u at each node. In each element there will be four (in the bilinear case). The number of global nodes is $(M+1)(N+1)$.

1.5.3 Task 3

Question: Describe in detail how the bilinear basis functions $\hat{\varphi}_0(X,Y), \hat{\varphi}_1(X,Y), \hat{\varphi}_2(X,Y), \hat{\varphi}_3(X,Y)$ are defined in a reference cell $(X,Y) \in [-1,1]^2$. (Hint: Each basis function is of the form $(aX + b) \cdot (cY + d)$ with suitable choices of the a, b, c, d scalar values).

Answer: The basis functions are originally defined in the interval of their respective elements, $[x_L, x_R]$, but we want to map it to a reference cell $[-1,1]$, because then we can compute all integrals over the same domain.

General basis function:

$$\hat{\varphi}_r = \Pi_{s=0, s\neq r}^{d} \frac{X - X_s}{X_r - X_s} \frac{Y - Y_s}{Y_r - Y_s}$$

Basis functions for a cell:

$$\hat{\varphi}_0 = \frac{1}{4}(X-1)(Y-1)$$
$$\hat{\varphi}_1 = -\frac{1}{4}(X+1)(Y-1)$$
$$\hat{\varphi}_2 = \frac{1}{4}(X+1)(Y+1)$$
$$\hat{\varphi}_3 = -\frac{1}{4}(X-1)(Y+1)$$

1.5.4 Task 4

Question: For element number e, how can the physical coordinates (x,y) be mapped from the local coordinates (X,Y) of the reference cell?

Answer: Formula for linear mapping:

$$x = \frac{1}{2}(x_L + x_R) + \frac{1}{2}(x_R - x_L)X$$

$$y = \frac{1}{2}(y_B + y_T) + \frac{1}{2}(y_T - y_B)X$$

1.5.5 Task 5

Question: Compute the element matrix and vector for element number e, with help of the reference cell.

Answer: Gradients of the four basis functions:

$$\nabla\hat{\varphi}_0 = \frac{1}{4}(Y-1, X-1)$$
$$\nabla\hat{\varphi}_1 = -\frac{1}{4}(Y-1, X+1)$$
$$\nabla\hat{\varphi}_2 = \frac{1}{4}(Y+1, X+1)$$
$$\nabla\hat{\varphi}_3 = -\frac{1}{4}(Y+1, X-1)$$

Formula for entries in the element matrix:

$$\int_{\hat{\Omega}^{(r)}} \nabla\hat{\varphi}_i \nabla\hat{\varphi}_j \texttt{det} J dXdY.$$

Formula for entries in the element vector:

$$2\int_{\hat{\Omega}^{(r)}} \nabla\hat{\varphi}_i \texttt{det} J dXdY.$$

We have that J is the Jacobian of the mapping $x(X)$:

$$J = \begin{bmatrix} \frac{\partial x}{\partial X} & \frac{\partial x}{\partial Y} \\ \frac{\partial y}{\partial X} & \frac{\partial y}{\partial Y} \end{bmatrix} = \begin{bmatrix} \frac{1}{2}(x_R - x_L) & 0 \\ 0 & \frac{1}{2}(y_T - y_B) \end{bmatrix}$$

$$\texttt{det} J = \frac{\partial x}{\partial X}\frac{\partial y}{\partial Y} - \frac{\partial x}{\partial Y}\frac{\partial y}{\partial X}$$

The last term is zero, so we have

$$\texttt{det} J = \frac{\partial x}{\partial X}\frac{\partial y}{\partial Y} = \frac{1}{4}(x_R - x_L)(y_T - y_B) = \frac{1}{4}hl$$

,

where h is the the height and l is the length of the elements.

The integrals are

[0]:
```
#@title
import numpy as np
```

```
import matplotlib.pyplot as plt
import sympy as sym
from google.colab.output._publish import javascript
url = "https://cdnjs.cloudflare.com/ajax/libs/mathjax/2.7.3/latest.js?
↪config=default"
javascript(url=url)
sym.init_printing(use_unicode=True)


X, Y, h, l = sym.symbols('X, Y, h, l')

fourth = sym.Rational(1, 4)

J = fourth*h*l

phi0 = fourth*((X-1)*(Y-1))
phi1 = -fourth*((X+1)*(Y-1))
phi2 = fourth*((X+1)*(Y+1))
phi3 = -fourth*((X-1)*(Y+1))

grad0 = fourth*sym.Matrix([Y-1, X-1])
grad1 = -fourth*sym.Matrix([Y-1, X+1])
grad2 = fourth*sym.Matrix([Y+1, X+1])
grad3 = -fourth*sym.Matrix([Y+1, X-1])


def get_matrix_element(f1, f2):
    temp_integral = sym.integrate(f1.dot(f2)*J, (X, -1, 1))
    integral = sym.integrate(temp_integral, (Y, -1, 1))

    return integral

def get_vector_element(f):
    temp_integral = sym.integrate(f*J, (X, -1, 1))
    integral = sym.integrate(temp_integral, (Y, -1, 1))

phis = [phi0, phi1, phi2, phi3]
grads = [grad0, grad1, grad2, grad3]
A = sym.zeros(4, 4)



for i in range(4):
    for j in range(4):
        A[i, j] = get_matrix_element(grads[i], grads[j])
```

```
print('A:')
A
```

<IPython.core.display.HTML object>

A:

[0]:

$$\begin{bmatrix} \frac{hl}{6} & -\frac{hl}{24} & -\frac{hl}{12} & -\frac{hl}{24} \\ -\frac{hl}{24} & \frac{hl}{6} & -\frac{hl}{24} & -\frac{hl}{12} \\ -\frac{hl}{12} & -\frac{hl}{24} & \frac{hl}{6} & -\frac{hl}{24} \\ -\frac{hl}{24} & -\frac{hl}{12} & -\frac{hl}{24} & \frac{hl}{6} \end{bmatrix}$$

[0]:

```
#@title
from google.colab.output._publish import javascript
url = "https://cdnjs.cloudflare.com/ajax/libs/mathjax/2.7.3/latest.js?
↪config=default"
javascript(url=url)

b = sym.zeros(4)

def get_vector_element(f):
    temp_integral = sym.integrate(f*J, (X, -1, 1))
    integral = sym.integrate(temp_integral, (Y, -1, 1))

    return integral

for i in range(4):
    b[i] = get_vector_element(phis[i])

b[0:4]
```

<IPython.core.display.HTML object>

[0]:

$$\left[\frac{hl}{4}, \quad \frac{hl}{4}, \quad \frac{hl}{4}, \quad \frac{hl}{4}\right]$$