Numerical simulation using finite elements

Weak formulation

We consider the heat equation

$$\partial_t u = \Delta u.$$

on Ω.

To apply the finite element method we first pose the problem in a **weak formulation**.

Multiply by a test function φ and integrate over the domain. Then we have

$$\int_\Omega (\partial_t u)\varphi \, dx = \int_\Omega \Delta u \varphi \, dx$$

and thus

$$\int_\Omega (\partial_t u)\varphi \, dx = -\int_\Omega \nabla u \cdot \nabla \varphi \, dx \tag{1}$$

where we have assumed that $\varphi|_{\partial\Omega} = 0$.

Problem statement: Find u such that (1) is satisfied for all φ.

Discretization

We start with a **triangulation** with vertices $v_1, \ldots, v_{n_v}$ and n_T triangles.

- Each triangle is specified by a 3-tuple of vertices.

The basis function φ_i associated to the vertex v_i is piecewise linear on each triangle.

- For a triangle (v_i, v_j, v_k) we have $\varphi_i(x, y) = a + bx + cy$.

The constants a, b, and c are determined by imposing the **interpolation condition**

$$\begin{bmatrix} 1 & x_i & y_i \\ 1 & x_j & y_j \\ 1 & x_k & y_k \end{bmatrix} \begin{bmatrix} a \\ b \\ c \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}$$

The function φ_i is zero inside each triangle that does not contain v_i as a vertex.

Basis functions

With $\eta_0 = (1,1,1)^T$, $\eta_1 = (x_i, x_j, x_k)^T$, $\eta_2 = (y_i, y_j, y_k)^T$ we have

$$\begin{bmatrix} a \\ b \\ c \end{bmatrix} = \frac{1}{\eta_0 \cdot (\eta_1 \times \eta_2)} \begin{bmatrix} (\eta_1 \times \eta_2)^T \\ (\eta_2 \times \eta_0)^T \\ (\eta_0 \times \eta_1)^T \end{bmatrix} \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}$$
$$= \frac{1}{\eta_0 \cdot (\eta_1 \times \eta_2)} \begin{bmatrix} (\eta_1 \times \eta_2)_1 \\ (\eta_2 \times \eta_0)_1 \\ (\eta_0 \times \eta_1)_1 \end{bmatrix} = \frac{1}{\eta_0 \cdot (\eta_1 \times \eta_2)} \begin{bmatrix} x_j y_k - x_k y_j \\ y_j - y_k \\ x_k - x_j \end{bmatrix}$$

with

$$\eta_0 \cdot (\eta_1 \times \eta_2) = x_i y_j - x_i y_k - x_j y_i + x_j y_k + x_k y_i - x_k y_j .$$

Finite elements

Now, expanding $u = \sum_i u_i \varphi_i$ and testing with φ_j gives

$$\sum_i (\partial_t u_i) \underbrace{\int_\Omega \varphi_i \varphi_j \, dx}_{M_{ji}} = \sum_i u_i \underbrace{\left(- \int_\Omega \nabla \varphi_i \cdot \nabla \varphi_j \, dx \right)}_{A_{ji}}.$$

In **matrix notation**, with $U = [u_1, \ldots, u_{n_v}]$, we write this as

$$M \partial_t U = AU.$$

Matrix A

We have

$$\begin{aligned} A_{ij} &= -\int_\Omega \nabla\varphi_i \cdot \nabla\varphi_j \, dx \\ &= -\sum_k V_k \left[(\partial_x\varphi_i)(\partial_x\varphi_j) + (\partial_y\varphi_i)(\partial_y\varphi_j)\right]|_{T_k} \\ &= -\sum_{\text{triangle k includes i and j}} V_k(b_{ik}b_{jk} + c_{ik}c_{jk}), \end{aligned}$$

where b_{ik} and c_{ik} are the corresponding constants of the basis function i on triangle k and V_k is the area of triangle k.

Area of a triangle with vertices (x_1, y_1), (x_2, y_2), (x_3, y_3)

$$V = \frac{1}{2}|(x_1 - x_3)(y_2 - y_1) - (x_1 - x_2)(y_3 - y_1)|$$

Mass lumping

To solve

$$M\partial_t U = AU.$$

would require us to invert the matrix M.

We approximate the integral by a quadrature rule

$$M_{ij} = \int_\Omega \varphi_i \varphi_j \, dx \approx H_i \varphi_i(x_i)\varphi_j(x_i) = H_i \delta_{ij},$$

where $H_i =$ sum of the area of all triangles that include vertex i.

Diagonal matrix is easy to invert.

This approach is called **mass lumping** (second order accurate).

Numerical method

After inverting the diagonal matrix we have $\partial_t U = \mathrm{diag}(\frac{1}{H_1}, \ldots, \frac{1}{H_{n_v}})AU := BU$

Matrix assembly

```
B = 0 # initialize matrix to zero
for i=0:n_v-1 # iterate over all vertices
    H=0 # area of adjacent triangles
    for k=0:n_T-1
        if triangle k includes vertex i
            H += area of triangle k

    if vertex is _not_ on the boundary
        for j=0:n_v-1
            for k=0:n_T-1
                if triangle k includes vertices i and j
                    B[j + n_v*i] -= (area of triangle k)
                                    * (bik bjk + cik cjk)
        B[j + n_v*i] /= H
```

Not very fast, **but we have to do this only once**.

Numerical method

Applying the explicit Euler scheme to

$$\partial_t U = BU$$

gives

$$\frac{U^{n+1} - U^n}{\Delta t} = BU^n.$$

or

$$U^{n+1} = U^n + \Delta t B U^n.$$

This is our numerical scheme.

Visualization

Output file

```
# vtk DataFile Version 2.0
mesh with triangles
ASCII

DATASET POLYDATA
POINTS 25 double    ## 25 points
0 0 0               ## x y z
0.1 0 0
0.2 0 0
...

POLYGONS 32 128     ## 32 triangles, 4*32=128 data points
3 0 1 5             ## 3 data points (i.e. a triangle) id1 id2 id3
3 1 6 5
3 1 2 6
...
```

Output file

```
POINT_DATA 25               ## 25 points
SCALARS value double 1      ## scalar field with name 'value'
LOOKUP_TABLE default        ##   type double and 1 element per line
0                           ## value
0.2323                      ## value
1.434                       ## value
...
```

Comments starting with ## are added for clearity; they should not be in the file.

- Note that this does not apply to the first line.

Paraview

We will use $\boxed{\text{Paraview}}$ to visualize the results.

```
$ sudo apt install paraview
```

- File $\rightarrow$ Open and select the .vtk file.
- Apply
- Switch between 'Surface', 'Surface width edges', and 'Wireframe'.

Contour plot

- Filters $\rightarrow$ Search $\rightarrow$ Contour (or the button in the toolbar).
- Add values in 'Value Range'
- Hit 'Apply'

The 'eye symbol' in the 'Pipeline browser' can be used to show or hide filters.