Prof. Dr. Gerhard Zumbusch
Wissenschaftliches Rechnen I – WS 2022/23

Lineare Finite Elemente

Aufgabe 8

Lösen sie das Poissonproblem auf einem triangulierten Gebiet mit linearen Finiten Elementen und gemischten Randbedingungen. Ein kurzes Matlab Programm dafür ist angegeben. Erklären sie das Programm und bringen sie diese oder eine davon inspirierte Version zum Laufen.

- Gebiet: Einheitsquadrat, regulär in rechtwincklige Dreiecke zerlegt. $-\Delta u = 1$ mit Dirichlet 0 Randbedingungen.

The unknown state variable $U(x, y)$ is assumed to satisfy Laplace's equation:
$-U_{xx}(x, y) - U_{yy}(x, y) = F(x, y)$ in Ω
with Dirichlet boundary conditions $U(x, y) = U_D(x, y)$ on Γ_D and Neumann boundary conditions on the outward normal derivative: $Un(x, y) = G(x, y)$ on Γ_N
If Γ designates the boundary of the region Ω, then we presume that $\Gamma = \Gamma_D + \Gamma_N$
The code uses piecewise linear basis functions for triangular elements. Author: Jochen Alberty, Carsten Carstensen, Stefan Funken.

```
function fem_50 ( )
  clear
  load coordinates.dat;
  load elements3.dat;
  eval ( 'load neumann.dat;', 'neumann=[];' );
  load dirichlet.dat;
  A = sparse ( size(coordinates,1), size(coordinates,1) );
  b = sparse ( size(coordinates,1), 1 );
  for j = 1 : size(elements3,1)
    A(elements3(j,:),elements3(j,:)) = A(elements3(j,:),elements3(j,:)) ...
      + stima3(coordinates(elements3(j,:),:));
  end
  for j = 1 : size(elements3,1)
    b(elements3(j,:)) = b(elements3(j,:)) ...
      + det( [1,1,1; coordinates(elements3(j,:),:)'] ) * ...
      f(sum(coordinates(elements3(j,:),:))/3)/6;
  end
  if ( ~isempty(neumann) )
    for j = 1 : size(neumann,1)
      b(neumann(j,:)) = b(neumann(j,:)) + ...
        norm(coordinates(neumann(j,1),:) - coordinates(neumann(j,2),:)) * ...
        g(sum(coordinates(neumann(j,:),:))/2)/2;
    end
  end
  u = sparse ( size(coordinates,1), 1 );
  BoundNodes = unique ( dirichlet );
  u(BoundNodes) = u_d ( coordinates(BoundNodes,:) );
  b = b - A * u;
  FreeNodes = setdiff ( 1:size(coordinates,1), BoundNodes );
  u(FreeNodes) = A(FreeNodes,FreeNodes) \ b(FreeNodes);
  show ( elements3, coordinates, full ( u ) );
  return
end

function M = stima3 ( vertices )
  d = size ( vertices, 2 );
  D_eta = [ ones(1,d+1); vertices' ] \ [ zeros(1,d); eye(d) ];
  M = det ( [ ones(1,d+1); vertices' ] ) * D_eta * D_eta' / prod ( 1:d );
  return
end

function show (elements3, coordinates, u)
  trisurf ( elements3, coordinates(:,1), coordinates(:,2), u' );
  view ( -67.5, 30 );
```

```
  title ( 'Solution to the Problem' )
  return
end
```

Ergänzen Sie dazu noch folgende Dateien (Beispieldaten!):

- coordinates.dat, the spatial coordinates of the nodes.

  ```
  0.0000  0.0000
  0.0833  0.0000
  ...
  ```

- dirichlet.dat, the edges of the region along which Dirichlet boundary conditions are specified. Edges are denoted by pairs of nodes, and the nodes should be listed in their counterclockwise order around the region.

  ```
  1 2
  2 3
  ...
  ```

- elements3.dat, defines the triangular elements by listing the indices of their vertices. The vertices should be listed in counterclockwise order.

  ```
  1 2 3
  2 3 5
  ...
  ```

- neumann.dat, the edges of the region along which Neumann boundary conditions are specified. Edges are denoted by pairs of nodes, and the nodes should be listed in their counterclockwise order around the region.

  ```
  5 6
  6 7
  ...
  ```

- f.m, evaluates the right hand side of Laplace's equation.

  ```
  function value = f ( u )
    value = 2.0 * pi * pi * sin ( pi * u(:,1) ) .* sin ( pi * u(:,2) );
  ```

- g.m, evaluates the Neumann boundary conditions.

  ```
  function value = g ( u )
    value = zeros ( size ( u, 1 ), 1 );
  ```

- u_d.m, evaluates the Dirichlet boundary conditions.

  ```
  function value = u_d ( u )
    value = 2.0 * pi * pi * sin ( pi * u(:,1) ) .* sin ( pi * u(:,2) );
  ```