J.M. Burgerscentrum

Research School for Fluid Mechanics

Computational Fluid Dynamics II
Practical exercises

Prof.dr.ir. F.J. Vermolen and Prof.dr.ir. C. Vuik

2023

Delft University of Technology
Faculty of Electrical Engineering, Mathematics and Computer Science
Delft Institute of Applied Mathematics

Introduction

In this note we treat some exercises that are meant as illustration for the theory treated in the course "CFD 2".

The exercises are planned for several days. Of course one is free to finish the exercises of the previous day. The exercises are ordered in increasing degree of difficulty and it is not necessary to make all of them. Exercises marked with a star (*) are only meant for those who have time left. In general they are more difficult than the other ones.

The following extra manuals are meant for the exercises:

- When we refer to `editor` and use `edit`, the editor of the linux system is meant.
  You may use for example kwrite or you may open your favorite editor by pressing the left icon on the control panel and choose `utilities` and later on `editors`.

- The SEPRAN manuals can all be found at the following Internet address:

  `http://homepage.tudelft.nl/d2b4e/burgers/burg.html`

  press `Practical work` to find the manuals.

Alternatively they can also be found on the local system in the following way:

```
sepman intro        shows the introduction
sepman um           shows the user's manual
sepman sp           shows the manual standard problems
sepman exams        shows the examples manual
```

Schedule

- Monday: Exercises 1, 2 and 3
- Tuesday: Exercises 4 and 5
- Wednesday: Exercises 14, 15, and 17 (Matlab Exercises)
- Thursday: Exercises 6 and 7
- Friday: Exercises 8, 9 and 16

The exercises 10-13 are meant for those that have time left. These exercises require much more effort and have more an investigating character.

All these exercises have been preprogrammed. In your directory you will find 2 subdirectories: sepran and matlab. In sepran you will find the sepran files, in matlab the Matlab files (if you do not have access to Matlab you can also use octave).

`http://homepage.tudelft.nl/d2b4e/wi211/matlab-clones.html`

It is wise to create a new directory, for example workdir in which you run the sepran exercises and update the files. Make a copy of the files corresponding to a specific exercise from the sepran directory to this new directory. In this way the original files are kept undisturbed and may be reused. For example in case of Exercise 1:

```
mkdir workdir
cp sepran/exercise1c.msh workdir
cp sepran/exercise1d.prb workdir
cd workdir
```

After that you may run SEPRAN.

In case of Exercises 14, 15, and 17 you should proceed as follows. After login in go to the *main window* and give the following commands:

```
cd matlab
matlab (or octave)
```

After that the exercises 14, 15, and 17 can be made. Matlab may be left by the command `quit`. Logging out is done by the command `exit` in the *main window*.

Exercise 1 heat conduction in a plate with a hole

This exercise is meant to solve a simple temperature problem in order to get acquainted with SEPRAN as well as the notions mesh generation, computational part and postprocessing. Besides that, the notion of boundary element is introduced.
In this exercise we consider a square plate with a hole like sketched in Figure 1. In this plate we want to solve a stationary heat conduction problem,

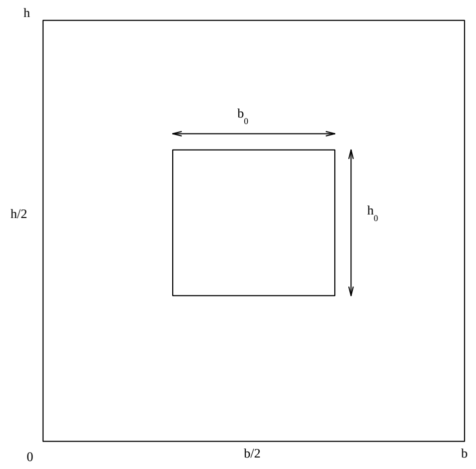

Figure 1: Square plate with hole

which is described by the two-dimensional Poisson equation

$$-div k \nabla T \;=\; \phi. \tag{1}$$

T(x,y) denotes the unknown temperature , k the heat conduction coefficient and ϕ the production of heat per unit of volume and time.
In order to solve this problem uniquely, it is necessary to prescribe boundary conditions on the whole boundary. Types of boundary conditions that can be applied are for example:

- Temperature T prescribed at the wall.
- Heat flux prescribed at the wall.
  $q_n = 0$: insulated wall

$q_n = q_0$: prescribed heat flux
with q_n normal component of the heat flux $\mathbf{q} = k\nabla T$.

- Heat transfer relation of Newton
  $q_n = \alpha(T_w - T_\infty)$
  with α the heat transfer coefficient,
  T_w, T_∞: the temperature on the wall and far away from the wall respectively.

We assume that in the case of the plate in Figure 1, the left and right-hand wall are kept at a constant temperature T = 0. The lower wall is kept at temperature T = 1. The heat production ϕ is assumed to be constant.
On the boundary of the hole we assume that $q_n = 0$ (insulated) or that $q_n = q_0$ (constant heat flux).
In first instance we assume that the upper wall is insulated (q_n=0) and in second instance we use the heat transfer relation $\alpha T + k\nabla T \cdot \mathbf{n} = 0$.

Exercises:

1a Verify that the temperature with the prescribed boundary conditions is symmetric with respect to the line x = b/2.
What is the correct boundary condition on this line motivated by the symmetry of the solution?

1b Derive the weak formulation for the heat conduction problem. If you have a choice, use the most general of the boundary conditions including the symmetry boundary conditions.
Why is it not necessary to use the symmetry boundary condition explicitly in the finite element method?

1c In order to solve the temperature equation, half the region is subdivided into elements , where the symmetry axis is part of the boundary. Subdivide the region in triangular elements with the aid of the SEPRAN grid generator SEPMESH. Use coarse with unit = 0.05 and as sub-grid generator GENERAL We choose the following data:
h = 1
b = 1
$h_0 = b_0 = 0.4$

In order to run this exercise we apply the next commands.

```
mkdir workdir
                             (workdir is your own work directory,
                              you may choose the name freely)
cp sepran/exercise1c.msh  workdir
cd workdir
sepmesh exercise1c.msh       (Make the mesh)
sepview sepplot.001          (View the mesh)
```

You can inspect the mesh file with the editor:

```
 kwrite exercise1c.msh
```

sepmesh creates a file meshoutput, which will be used by the subsequent programs.
Make a sketch of the boundary, using the input file and the plot shown by sepview. Indicate the user points and the curves in this sketch (reverse engineering).

1d Compute the temperature and the heat flux vector **q** by SEPCOMP using the following data:
k = 1
$\phi = 1$
The hole and the upper walls are insulated.

The procedure is as follows:

```
cp ../sepran/exercise1d.prb .
sepcomp exercise1d.prb       (Solve problem)
```

You can inspect the input file with the editor:

```
 kwrite exercise1d.prb
```

Program sepcomp solves the problem prints the temperature and plot the isotherms and make a vector plot of **q**.
You can inspect the plots by sepview:

```
  sepview sepplot.001          (View the plots)
```

Interpret the results. Are the boundary conditions satisfied?

1e Perform the same computations, with a coarseness of 0.025.

1f Instead of an insulated upper wall we use the mixed boundary condition $\alpha T + k\nabla T \cdot \mathbf{n} = 0$, with $\alpha = 500$.
Repeat the computations.

In this case we need to define more boundaries than in the first case. Therefore two new files are needed:

```
 exercise1f.msh
 exercise1f.prb
```

First run sepmesh and then sepcomp with these files and view the results.

```
sepmesh exercise1f.msh
sepcomp exercise1f.prb
sepview sepplot.001
```

Explain the boundary condition at the upper wall. Is this boundary condition satisfied?

1g Repeat the computations of Exercise 1f, now with T=0 on the right wall and a heat flux $q_n = 1$ through the hole. Try some combinations of physical data yourself. Also try some other configurations.

Also in this case the mesh is adapted since more boundaries are needed to define the boundary conditions. These new files are:

```
 exercise1g.msh
 exercise1g.prb
```

Explain the results.

After finishing the computations all files can be removed from your work directory,

Exercise 2 Memory usage and computational time for direct methods

The goal of this exercise is the study the memory usage and the computational time of a direct method as function of the number of elements and the numbering used.

For that reason we solve the Poisson equation on the rectangle (0,5) × (0,1).

$$-\Delta T \;=\; 1. \tag{2}$$

As boundary conditions we choose $T = 0$ on the whole boundary.

Exercises:

2a Generate a mesh by sepmesh with 250 elements in horizontal direction and 50 elements in vertical direction. Use the submesh generator SQUARE. Use bi-linear rectangular elements. The number of elements is defined by using coarse(unit=0.05).

The input file for sepmesh is

```
exercise2.msh
```

2b Solve the Poisson equation with a direct method and the matrix stored as symmetrical matrix. Inspect the number of entries of the matrix and the computation time required for building the equations and solving the linear system of equations. In order to measure the memory usage and the computational time we put two extra lines in the input file:

```
set output on
set time on
```

The adapted input file is called exercise2b.prb

These lines generate a large quantity of extra output. The number of entries in the large matrix is given by the line:

```
the large matrix contains .... real entries.
```

The computational time is given (accumulated) in the lines:

```
time is ... seconds in subroutine ....
```

In order to measure the computational time in the matrix building we subtract the computational time before the call of build from the one after the call of build. It concerns the lines

```
 time is ... seconds in subroutine buildbf
```

The line

```
 time is ... seconds in subroutine solvelbf
```

gives the time after the linear solver.
Carry out the same exercise, but now with the matrix stored as non-symmetrical matrix.

2c In exercise 2b we implicitly used the fact that SEPRAN renumbers the nodes. To investigate the effect of the renumbering, we switch of the renumbering. This is done by putting the lines [1]

```
start
   renumber not
end
```

in the input file for sepcomp before the part

```
problem
.
.
end
```

The adapted input file is called

[1]if you put a # before renumber not, the command is ignored

`exercise2c.prb`

Repeat exercise 2b.
Try to estimate the memory usage if a band method and a horizontal is chosen. Do the same for a vertical numbering. Compare these with the printed values. Note that SEPRAN uses a profile method.

2d In order to find a relation between the computational time and memory usage as function of the number of points we subdivide the region again with SEPMESH, however, now by doubling the mesh in both directions, using coarse(unit=0.025) and again using coarse(unit=0.0125). Repeat exercise 2b for these subdivisions.
if you use unit = 0.00625 the non-renumbered case fails because the amount of declared memory is too small.

Exercise 3* Writing a user element

Remark: this exercise is meant for those students that want to write an element subroutine themselves. It is not part of the standard exercises.

The goal of this exercise is practicing the writing of your own standard element. For that reason we reuse exercise 1, however, in this case we use element type 1 for the inner region and element types 2 and 3 respectively for the boundary elements. Due to this choice it is necessary to add your own subroutine ELEM to program SEPCOMP.

Exercises:

3a Derive the Galerkin equations corresponding to the weak formulation of 1b. Compute the element matrix and element vector using linear triangles and the also the linear boundary element. Assume that the triangle may have an arbitrary shape and also that boundary elements may be positioned anywhere in the domain.

3b Program subroutine ELEM using the results of 3a. Use the users manual Section 4.2. Repeat the exercises 1c, 1d and 1f using your own element.

Exercise 4 Application of iterative methods

The goal of this exercise is to solve a simple three-dimensional problem and to compare the effect of direct solvers with the effect of iterative solvers.

For that reason we compute the temperature in a straight pipe. The lower wall and the upper wall of the pipe consist of parallel circles with radius 1. The outer wall is perpendicular to these circles and has height 5. The temperature in the pipe satisfies the Laplace equation

$$-div k \nabla T \; = \; 0. \tag{3}$$

The pipe is insulated on the upper wall and on the under wall we have a constant temperature T_0. On the outer wall we assume the heat transfer relation: $q_n = \alpha(T_w - T_\infty)$
To solve this problem, we can use the axi-symmetry or we can solve the problem as a three-dimensional one.
First we carry out the axi-symmetrical approach.

Exercises:

4a Choose $k = 1$ and $\alpha = 0.5$. Solve the heat conduction problem with 20 elements in the r direction and 100 elements in the z direction. Record the number of nodes in the mesh and the number of elements of the matrix as well as the computation time required to build the matrix and to solve the system of equations .
The corresponding files are:

```
exercise4a.msh
exercise4a.prb
```

4b Repeat the exercise with 40×200, 80×400 elements and 160×800 elements.

4c Now repeat the exercises 4a and 4b with an iterative solver (CG) and an accuracy of 10^{-3}. Compare the results.
The corresponding file is:

```
exercise4c.prb
```

4d We will now repeat the exercise with a real three dimensional mesh. Although this is senseless for this exercise, in practice many problems are purely three-dimensional.
Subdivide the lower surface into triangular elements by the submesh generator GENERAL. Use 40 elements along the circle and 50 in z-direction.
The upper face must be constructed by translation of the lower face.
The side wall must be generated by the generator PIPESURFACE.
To generate the 3D grid the volume generator PIPE is used.
Carry out the same exercises as in 4a with a direct solver.
The corresponding files are:

```
exercise4d.msh
exercise4d.prb
```

4e Repeat 4d with an iterative solver.
The corresponding file is:

```
exercise4e.prb
```

4f We refine the mesh by taking 80 elements along the circle and 100 in z-direction. repeat exercises 4a and 4e.

Exercise 5 heat transfer and temperature in a Poisseuille flow

The goal of this exercise is to study the influence of outstream boundary conditions on the convection-diffusion equation as well as to study the influence of the streamline Petrov Galerkin upwinding.

Consider the flow in a pipe as sketched in Figure 2. We suppose that the flow

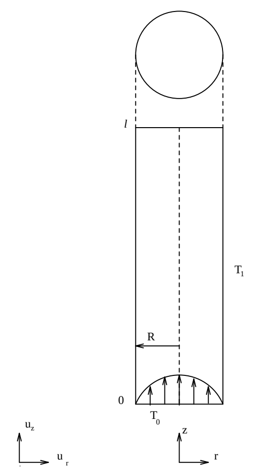

Figure 2: Pipe with Poisseuille flow

is a stationary Poisseuille flow. The heat dissipation is neglected and there are no other heat sources. The temperature equation is considered stationary. Changes of the velocity due to temperature differences are neglected.

We consider the flow in a straight, cylindrical tube with radius R. For z = 0 the walls have a temperature T_0 and for z > 0 a temperature T_1. We consider the region $0 \leq$ z $\leq l$ and assume that the liquid on z = l has a homogeneous temperature. Since flow and temperature are axi-symmetrical it is sufficient to solve the problem in a cross-section with axi-symmetrical coordinates.
Define the inlet length l_{in} as the minimal length for which temperature various at most 1% through the cross section. For z $\geq l_{in}$, it can be shown that

$$l_{in} = 0.6\frac{Pe}{R},$$

$$Pe = \frac{U_{max}R}{a},$$

with Pe the Peclet number , U_{max} the maximum velocity and a the so-called temperature adjustment coefficient.

The convection-diffusion equation for the temperature can be written in dimensionless form as:

$$-\frac{1}{Pe}\Delta T \; + \; \mathbf{u}\cdot\nabla T = \; 0. \tag{4}$$

In our example we use the next data:

$T_0 = 0$

$T_1 = 1$

$u_r = 0$, $u_z = 1$ - r^2

$R = 1$ $Pe = 700$

On the lower boundary we have $T = T_0$, on the side wall $T = T_1$. On the symmetry axis we have of course the symmetry boundary condition $\frac{\partial T}{\partial n} = 0$. From these data it follows that the inlet length is equal to 420.
We shall now compute the temperature for various choices of the outflow boundary and various boundary conditions on the outflow boundary.

Exercises:

5a Choose the outflow boundary on z = 500 and choose T = 1 as boundary condition on z = 500. Subdivide the region into 4 elements in r direction and 10 elements in z direction. Use the submesh generator quadrilateral and rectangular elements. This solution will be the reference for the other parts. Make a contour plot (or a colored contour plot) of the temperature .
The corresponding files are:

```
exercise5a.msh
exercise5a.prb
```

5b Choose the outflow boundary on z = 250 and use the same boundary conditions. Repeat exercise 5a. The temperature will show non-physical oscillations. Explain these oscillations.
The corresponding file is:

```
exercise5b.msh
```

Is it possible to remove the oscillations by refining the mesh?
Investigate this in practice.

5c In order to suppress the oscillations, we apply the SUPG method instead of the standard Galerkin method. Now repeat exercise 5b.
Compare the results.
The corresponding file is:

```
exercise5c.prb
```

Repeat the exercise for finer grids.

5d Repeat part 5b with SGA, using the outflow boundary condition $\frac{\partial T}{\partial n} = 0$.
Explain the results.
The corresponding file is:

```
exercise5d.prb
```

Exercise 6 The rotating cone problem

In this exercise we shall solve the rotating cone problem from the lecture notes, however, with a coarser grid.

Consider the region in Figure 3. We consider a rotating velocity field $\mathbf{u} = \begin{pmatrix} -y \\ x \end{pmatrix}$.

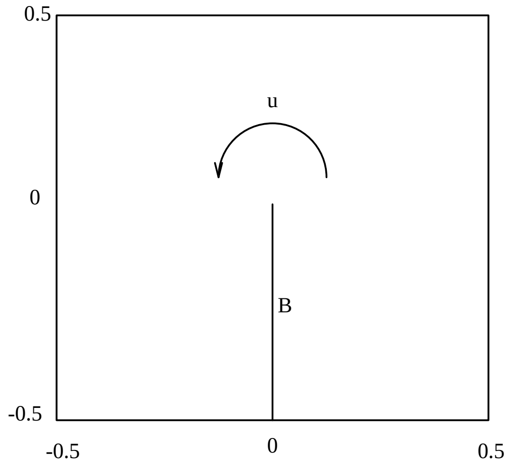

Figure 3: Definition of region for the rotating cone problem

The velocity field starts on the line B, rotates counter clockwise and ends on the line B. We convect a cosine shaped concentration by the velocity field. The concentration c satisfies the convection-diffusion equation

$$-\epsilon \Delta c \ + \ \mathbf{u} \cdot \nabla c = \ 0. \tag{5}$$

On the outer boundary we choose $c \ = \ 0$ as boundary condition. On the inflow boundary B we choose $c \ = \ cos(2\pi(y + \frac{1}{4}))$ as boundary condition. We suppose that there is a gap of thickness 0 between inflow boundary and outflow boundary. In other words we choose two different curves for inflow boundary and outflow boundary. On the outflow boundary we choose $\frac{\partial c}{\partial n} = 0$ as boundary condition. We shall solve this problem both with SGA and SUPG.

Exercises:

6a Make a mesh consisting of triangles by the submesh generator QUADRILATERAL. Use 20 elements along each outer side.
The corresponding file is:

```
exercise6a.msh
```

6b Choose $\epsilon = 10^{-6}$ and solve the convection-diffusion equation with SGA. Use a direct solver for the linear system. Make a contour plot and a 3D plot of the concentration.
The corresponding file is:

```
exercise6b.prb
```

6c Repeat 6b, however, now with SUPG.
The corresponding file is:

```
exercise6c.prb
```

6d Repeat 6b and 6c, but use a mesh consisting of quadrilaterals
The corresponding file is:

```
exercise6d.msh
```

Exercise 7 The use of iterative methods for the convection-diffusion equation

In this exercise we use the convection-diffusion equation as described in the SEPRAN Standard Problems manual, Section 3.1.
Consider the example of the curved pipe with given concentration on the inflow boundary as described in the Standard Problems manual, Section 3.1.2. Using this example we shall investigate the behavior of iterative methods.

First we start with the Poisson equation, in which the discretization matrix is symmetric and positive definite. This is done by using poisson as type in the problem input block.

7a Solve the system with a direct method

```
matrix_structure: symmetric     # matrix is symmetric
```

Record the memory usage and the computation time used. Plot the computed solution by sepview. Repeat this by adding the following input block before `end_of_sepran_input`

```
solve
   positive_definite
end
```

to the file.
The corresponding files are:

```
exercise7.msh
exercise7a.prb
```

7b Now we shall perform some experiments with the CG method. Use

```
matrix_structure: storage_scheme = compact, symmetric
```

and

```
solve
   iteration=method = cg, print_level = 1, preconditioner = ilu
end
```

Read the memory usage, the number of iterations and the computation time required. Choose the following preconditioners: diagonal, eisenstat, ilu and mod_eisenstat. If time is left vary the number of discretization points.
The corresponding file is:

```
exercise7b.prb
```

7c We now consider the effect of renumbering on the iterative method. For the CG method without preconditioning the number of iterations must be independent of the ordering. Check this by using the next orderings:

```
start
   renumber not
end
```

After that the Cuthill profile and Sloan profile renumbering:

```
start
   renumber cuthill profile
end
```

or

```
start
   renumber sloan profile
end
```

In case of a preconditioned CG method the ordering will have some influence.
In this case the mesh generator has already renumbered the nodes in an almost optimal way, so the effect of further renumbering is very small. Choose the mod_eisenstat preconditioning and use all previous reordering strategies.

7d Next we solve the original convection diffusion. Use central differences and compute the solution with a direct method. Plot the solution with sepview. Next solve the problem iteratively with the ilu preconditioning and Bi-CGSTAB, CGS, IDR, and GMRES. Choose for the `iteration_method` choose `bicgstab`, `cgs`, `idr`, and `gmres`.
The corresponding files are:

```
exercise7d.prb
```

In the case of a direct method we do not define the matrix storage, but use the default one, which implies a profile storage scheme for an asymmetric matrix.
In case of the iterative methods we must use a compact storage scheme:

```
matrix_structure: storage_scheme = compact
```

and add an extra block:

```
solve
   iteration_method = bicgstab, print_level = 1, preconditioner = ilu
end
```

7e Repeat these exercises with upwind discretization. Compare the solution and the computational time required with that of the previous part.
The corresponding file is:

```
exercise7e.prb
```

The difference between central and upwind scheme appears to be very small. However, make the diffusion parameter κ 10 times smaller (0.0005 instead of 0.005) and compare the results of both schemes. Also try a decrease with a factor 100.

7f From now on we use upwind discretization. Solve the system with the Bi-CGSTAB method without preconditioning. Record also the begin and end residual. Repeat this with eisenstat, ilu and mod_eisenstat preconditionings. Conclusion?

7g Use ilu as preconditioning and Bi-CGSTAB, CGS, IDR, and GMRES as iterative methods. Compare the results.

7h If time is left repeat the exercises with various diffusion parameters.

Exercise 8 Development of a flow in a straight pipe

The goal of this exercise is to give an introduction in the solution of the stationary Navier-Stokes equations by the finite element method.

Consider the flow between two flat plates. At the inflow we prescribe a constant velocity. After some time the flow will have been converted to a fully developed flow . In this exercise we shall simulate this numerically. Consider the configuration as sketched in Figure 4. Due to symmetry it is sufficient

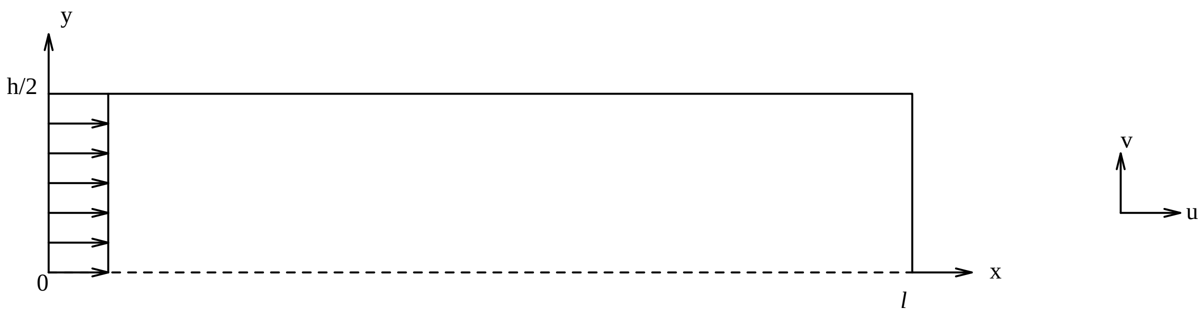

Figure 4: Region of definition for exercise 8

to consider a half channel only. The liquid satisfies the Navier-Stokes equations. At the inflow we prescribe a constant velocity field parallel to the flat plates. On the fixed walls we have a no-slip condition. At the outflow we assume that the flow is parallel to the plates. Furthermore we suppose that the pressure is equal to 0.

Exercises:

8a Make a mesh consisting of quadratic triangles by the submesh generator QUADRILATERAL. Use 8 elements in x direction and 3 in y direction. Choose the height h equal to 1 and the length l equal to 10.
The corresponding file is:

```
exercise8.msh
```

8b Solve the Stokes equations by the grid created in 8a. Use as inflow velocity u $= 1$, a density $\rho = 1$ and a viscosity $\mu = 0.01$. What is the corresponding Reynolds number based on the distance h? Choose $\sigma = 10^6$ as penalty parameter, i.e. $\epsilon = 10^{-6}$.
Make a vector plot of the velocity and a streamline plot.
The corresponding files are:

```
exercise8b.prb
```

8c Investigate the dependence σ by comparing the solution for a number of values of σ. Use values between 10 and 10^{12}.

8d Repeat exercise 8b, using the Navier-Stokes equations instead of Stokes. Note that in case we have to solve a non-linear problem. The iteration process is explicitly written in the structure block.
The corresponding file is:

```
exercise8d.prb
```

Why is the difference between both solutions so small?

Exercise 9 Flow over a backward facing step

In this exercise we shall compute the flow over a backward facing step, as described in the lecture notes. The effect of the outflow boundary conditions is studied. Consider the backward facing step as sketched in figure 5. Define

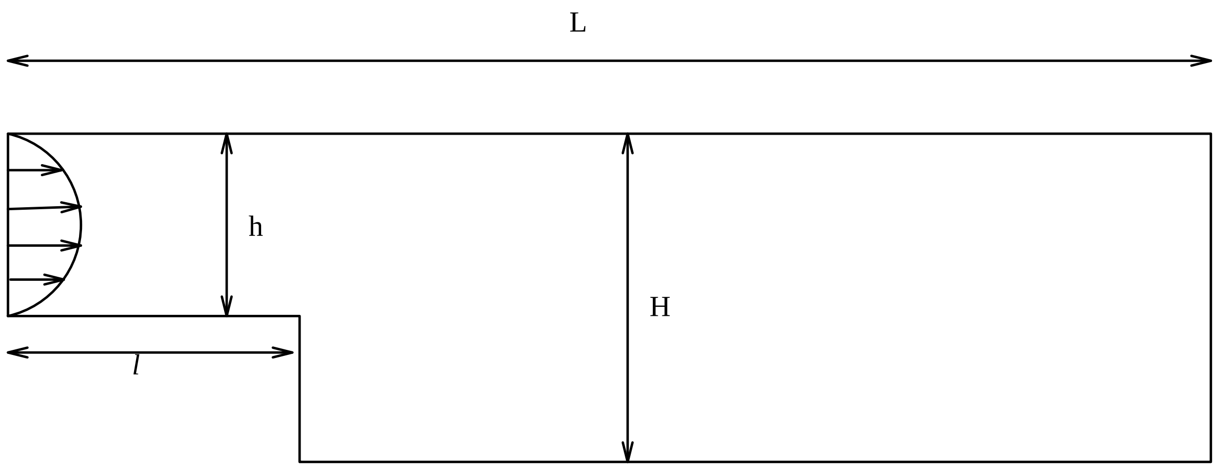

Figure 5: Region of definition backward facing step

the Reynolds number by $\text{Re} = U_{max}\frac{H-h}{\nu}$. At inflow we assume a fully developed velocity profile. At outflow you may choose between several types of outflow boundary conditions. At the walls there is a noslip condition.

Exercises:

9a Create a mesh by the submesh generator QUADRILATERAL consisting of quadratic triangles. Subdivide for that reason the region into two rectangles and apply QUADRILATERAL to each of the quadrilaterals. Increase the length of the elements in the direction of the outflow boundary. Do not use too many elements since otherwise the computational time increases too much. It is a good practice to start with a coarse grid and refine later if necessary.
Use the next data:

$\text{H} = 1$

h = 0.5

$l = 6$

L = 38

The corresponding file is:

```
exercise9.msh
```

9b Solve the Stokes equations for Re = 50. Check if the solution satisfies the boundary conditions.
The corresponding files are:

```
exercise9b.prb
```

9c Solve the Navier-Stokes equations for Re = 50. Make a stream line plot and a vector plot of the velocity. You can also make a contour plot of the pressure.
The corresponding file is:

```
exercise9c.prb
```

9d Solve the Navier-Stokes equations for Re = 150. If the iteration process does not converge try to construct a method to reach convergence.

9e Solve the Navier-Stokes equations for Re = 150. Choose L = 12 and try both the two different types of outflow boundary conditions mentioned in the lecture notes. What is your conclusion?
If time left, repeat these exercises with a finer mesh.

Exercise 10 Flow through a bend

In this exercise we consider the flow through a bend. We restrict ourselves to a two-dimensional Cartesian configuration. The region of definition is sketched in Figure 6. Compute and plot velocity and pressure in this bend

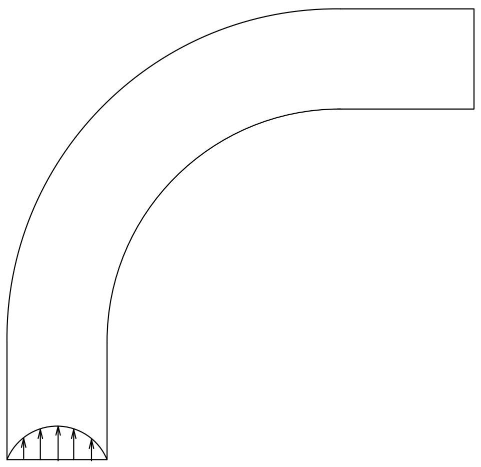

Figure 6: Region of definition bend problem

for various Reynolds numbers, where the Reynolds number is based on the width of the channel. For Re = 500 the flow must be laminar and stationary. The corresponding files are:

```
exercise10.msh
exercise10.prb
```

Repeat the exercise for smaller mesh sizes.

Exercise 11 Natural convection in a square cavity

In this exercise we consider the flow in a square cavity. The left and right walls are kept at a constant temperature. The left-hand wall gets the temperature T=1, the right-hand side wall: T=0. The lower wall and upper wall are assumed to be insulated. Compute and plot velocity, temperature and pressure in this cavity for various Rayleigh numbers in the range 10^3 to 10^6. Try both the coupled and the decoupled approach.
The corresponding files are:

```
exercise11.msh
exercise11a.prb

exercise11b.prb
```

Repeat the exercise for smaller mesh sizes.

Exercise 12* Heat exchanger

This exercise is meant for those that have time left.
In this exercise we consider the temperature distribution in a heat exchanger as sketched in Figure 7. It concerns two concentric pipes. Only the cross

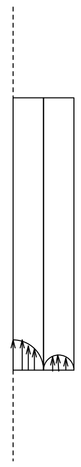

Figure 7: heat exchanger

section for $r > 0$ has been drawn. In the inner pipe the liquid is warm in the outer pipe it is cold. The outer temperature is assumed to be constant.

In first instance we assume that in both pipes we have a parabolic velocity profile and that the temperature distribution satisfies the convection-diffusion equation.

Formulate the differential equations and the boundary conditions. Devise a method to model the heat transfer between inner and outer pipe, where the temperature is different in both pipes. Use a number of values of the parameters.
Consider also the case that the liquid in the outer pipe has the opposite direction of that in the inner pipe.
Finally you can investigate if the velocity in the pipes is influenced by the temperature . For that purpose you have to solve the Boussinesq equations. Keep in mind that SEPRAN can only be used to solve laminar flows.
There are no example files available. Take an existing file and adapt it.

Exercise 13* Flow around a cylinder

This exercise is meant for those that have time left.
An application of a time-dependent flow, is the flow around a cylinder. it is known that for Re > 50 we can expect so-called von Karmann eddies. The Reynolds number is defined by the diameter of the cylinder. These von Karmann eddies can only arise if the flow is activated somewhere, for example by disturbing the symmetry in the velocity field.

Try to generate a von Karmann eddy by solving the time-dependent Navier-Stokes equations.
See the manual SEPRAN examples for some help.

Exercise 14
Experiments with the Gaussian elimination method

In this exercise we consider various properties of the Gaussian elimination method. The function `[a,f] = poisson(n1,n2,u1,u2,disc)` is used to construct a matrix, which originates from a discretization of the following equation:

$$-(\frac{\partial^2 c}{\partial x^2} + \frac{\partial^2 c}{\partial y^2}) + u_1 \frac{\partial c}{\partial x} + u_2 \frac{\partial c}{\partial y} = g \text{ where } x, y \in (0,1) \times (0,1),$$

$$c(x,y) = 0 \text{ for } x \in \{0,1\} \text{ or } y \in \{0,1\}.$$

The number of point in the x- and y-direction is denoted by n1, n2 respectively. The value `disc = 'central'` gives a central discretization, whereas `disc = 'upwind'` leads to an upwind discretization. The resulting matrix is a and the right-hand-side vector is f. To get more information of the Matlab programs one can use the `help`-function. Try for instance `help poisson`.

14a First the fill-in is investigated for small matrices. Therefore the command `[l,u,p] = gauss(a);` can be used. This function computes the LU-decomposition of a with pivoting such that $pa = lu$. Furthermore the number of nonzero elements and the bandwidth in l and u are given. Finally the nonzero structures of the matrices are shown.

Execute the following commands in Matlab:

```
[a,f] = poisson(3,10,0,0,'central');

[l,u,p] = gauss(a);
```

In the subroutine `gauss` the Matlab subroutine `lu` is called. Repeat this exercise with `[a,f] = poisson(10,3,0,0,'central');` and compare both examples.

Related theory: part II, Section 1.2, and 1.7
Exercise aims:

- The matrices L and U are less sparse than A.

- Numbering of the unknowns is important. Numbering such that n1 $\leq$ n2 is better than n1> n2.

14b In this exercise we take $u1 \neq 0$, or $u2 \neq 0$ and use central differences. It appears that for some choices partial pivoting is used.

Take the following choices:

```
[a,f] = poisson(5,5,0,0,'central');
[l,u,p] = gauss(a);

[a,f] = poisson(5,5,100,0,'central');
[l,u,p] = gauss(a);

[a,f] = poisson(5,5,0,100,'central');
[l,u,p] = gauss(a);
```

Try also some other values. Warning: for large matrices the CPU-time can increase considerably (`ctrl c` kills a job).

Related theory: part II, Section 1.2, 1.5, and 1.7
Exercise aims:

- Pivoting destroys the non-zero structure of the matrix.
- The amount of fill-in is comparable to that without pivoting.

14c Do the same exercises as before for upwind differences. Compare the results.

Related theory: part II, Section 1.2, 1.5, and 1.7
Exercise aims:

- The choice of discretization influences the Gaussian elimination algorithm.
- No pivoting is necessary if A^T is diagonal dominant.

14d In this exercise matrices are considered where the amount of fill-in can be very large. The matrices are not related to the Poisson equation. The matrices used here are known as arrowhead matrices. This type of matrix occurs in Domain Decomposition methods. Call `uparrow` and substitute 25 for the dimension. Execute thereafter `[l,u,p] = gauss(a);`. Repeat the exercise for the `downarrow` matrix and compare the results.

Related theory: part II, Section 1.2, and 1.8
Exercise aims:

– Pivoting (or renumbering) can be used to minimize fill-in.

14e In the following part we investigate the solution of the discretized Poisson equation. Execute `[a,f] = poisson(10,10,0,0,'central');`. The LU-decomposition of the matrix a can be obtained by:

```
[l,u,p] = lu(a);
```

Compute with this LU-decomposition the solution of the linear system $ax = f$. The solution of $ly = f$ can be done by:

```
y = l\f;
```

To check if you have computed the correct solution compute:

```
norm(a*x-f)
```

The resulting solution x can be visualized using the call:

```
plotsol(10,10,x)
```

Related theory: part II, Section 1.2, and 1.3
Exercise aims:

– Compute the solution of a linear system when the LU-decomposition is available.

– Visualization of the solution of the Poisson equation.

14f Repeat the exercise for

```
[a,f] = poisson(10,10,200,0,'central');
```

Compute again `norm(a*x-f)`. Its value should be less than 10^{-10}. Finally the same problem can be solved using upwind differences. Compare both solutions.

Related theory: part II, Section 1.2, and 1.5
Exercise aims:

- Compute the solution of a linear system when the LU-decomposition with pivoting is available.
- A comparison of the approximations using central and upwind differences.

14g Finally some experiments are done with iterative improvement. First the matrix and right-hand side are formed by `[a,f] = poisson(5,5,0,0,'central');` Compute the LU-decomposition with

```
[l,u,p] = lu(a);
```

It is not easy to work with single precision numbers in Matlab. To simulate iterative improvement we disturb the matrix l as follows:

```
l = random(l,10^(-5));
```

This function changes the lower triangular matrix l with random numbers of order less than 10^{-5}. Compute `norm(a-l*u)` to check that $a \neq lu$ (the matrix u can also be changed). Write a function `iterna(a,l,u,p,f)` which uses the disturbed matrix l (and possible the disturbed matrix u). First construct a termination criterion and make the function such that it stops after 100 iterations.

Related theory: part II, Section 1.2, and 1.5
Exercise aims:

- Get experience with the iterative improvement algorithm.

Exercise 15
Experiments with iterative methods

In this exercise we consider various iterative methods to solve linear systems. We start with systems where the coefficient matrix is symmetric and positive definite. These systems are solved by basic iterative methods and the Conjugate Gradient method. Finally we investigate if these methods can also be applied to non-symmetric systems.

15a Construct the matrix a and the right-hand-side vector f using the call `[a,f] = poisson(10,11,0,0,'central');` Compute the solution of the linear system with the Gauss-Jacobi method:

```
x = jacobi(a,f,10^-10);
```

Write down the number of iterations for comparison with other methods.

Related theory: part II, Section 2.2
Exercise aims:

- Gauss-Jacobi is a slowly converging method.
- A linear converging iteration method.

15b First try to modify the program `jacobi.m` to obtain the Gauss-Seidel method. To check your implementation you also use:

```
x = seidel(a,f,10^-10);
```

Compare the results with those obtained in exercise 15a.

Related theory: part II, Section 2.2
Exercise aims:

- Gauss-Seidel converges two times as fast as Gauss-Jacobi.

15c In this exercise we use the SOR method. The estimate of the optimal ω is not straightforward. Therefore we do some experiments with various choices of ω. Type

```
x = sor(a,f,10^-10,omega);
```

where `omega` is a real number between 0 and 2. Try to approximate the optimal value. Using the call

```
x = sor(a,f,10^-10,0);
```

an approximation of the optimal ω is calculated from theory. Compare the results.

Related theory: part II, Section 2.2
Exercise aims:

- SOR converges much faster than the previous methods.
- The convergence behavior can be non-linear.
- There are values of ω which leads to somewhat less iterations than the optimal ω calculated from theory.

15d In this part we consider the Conjugate Gradient method.

```
x = cg(a,f,10^-10);
```

Compare the results with the results obtained with the basic iterative methods.

Related theory: part II, Section 3.2 and 3.3
Exercise aims:

- CG converges very fast.
- It is not necessary to estimate an optimal parameter.
- The convergence behavior is super linear.

15e In the theory used to analyze CG three quantities are used: $\|x - x_i\|_2$, $\|x - x_i\|_A$, and $\|b - Ax_i\|_2$. These quantities are plotted by the call

```
x = cganal(a,f,10^-10);
```

Compute the condition of a by the Matlab command `cond(a)` and estimate the number of required CG iterations. Compare this with your experiments.

Related theory: part II, Section 3.2 and 3.3
Exercise aims:

- All quantities decrease.
- The number of required estimations obtained from the theory is an upper bound.

15f There are matrices where the theory for the CG method no longer holds, due to rounding errors. One of them is a discretization of the *bending beam equation*:

$$\frac{d^4c}{dx^4} = f.$$

The matrix can be constructed by `beam;` Take for the dimension 40 (thereafter also try 80 and 160). Solve the system by

```
x = cganal(a,f,10^-10);
```

and observe the results.

Related theory: part II, Section 3.3
Exercise aims:

- $\|x - x_i\|_A$ forms a monotone decreasing sequence.
- $\|b - Ax_i\|_2$ increases at some iterations.
- CG has not been converged when the number of iterations is equal to the dimension of a.

15g In the final exercises we investigate if the methods given in Chapter 2 and 3 can be used when the matrices are non-symmetric. Construct the matrix as follows: `[a,f] = poisson(5,5,0.1,0,'central');` and apply Gauss-Jacobi, Gauss-Seidel, SOR, and CG.

Exercise aims:

- All methods converge although the theory for CG is not valid.

15h Construct the matrix as follows: `[a,f] = poisson(5,5,1,0,'central');` and apply Gauss-Jacobi, Gauss-Seidel, SOR, and CG.

Exercise aims:

- The basic iterative methods converge but CG does no longer converge.

15i Construct the matrix as follows: `[a,f] = poisson(5,5,100,0,'central');` and apply Gauss-Jacobi, Gauss-Seidel, SOR, and CG.

Exercise aims:

- The iterative methods are divergent.

15j Construct the matrix as follows: `[a,f] = poisson(5,5,100,0,'upwind');` and apply Gauss-Jacobi, Gauss-Seidel, SOR, and CG.

Exercise aims:

- The basic iterative methods converge.
- The choice of the discretization influences the convergence behavior of an iterative method.

Exercise 16 Pressure in the earth's surface

In this exercise, we are investigating the convergence behavior of the ICCG process for problems with layers with large contrasts in the coefficients. For that reason we simplify the equation considerably and assume that we have to solve the stationary linearized 2D diffusion equation, in a layered region:

$$-\text{div}(\gamma \nabla p) = 0 \ , \tag{6}$$

with p the excess pressure and γ the permeability. At the earth's surface the excess pressure is prescribed. When the pressure field is required in some reservoir it is not practical to calculate the pressure in every position of the earth's crust. Therefore the domain of interest is restricted artificially. We assume that the lowest layer is bounded by an impermeable layer, so there is no flux through this boundary. The artificial vertical boundaries are taken at a sealing fault, or far away from the reservoir. Again a zero flux condition is a reasonable assumption at these boundaries.For our model problem we assume that γ in sandstone is equal to 1 and γ in shale is equal to 10^{-7}. Furthermore the Dirichlet boundary condition at the earth's surface is set equal to 1. The solution of equation (6) with these boundary conditions is of course $p = 1$, but if we start with $p = 0$ or a random vector, our linear solver will not notice the difference with a real problem. Numerical experiments show that the choice of one of these start vectors has only marginal effects on the convergence behavior. An advantage of this problem is that the exact error can easily be calculated. We consider this problem on a rectangular domain with 7 straight layers. In each layer 10 elements in the horizontal and 5 elements in the vertical direction are used.

Exercises:

Before starting this exercise copy the files exercise16* from the sepran directory to your working directory.

16a We start by solving this problem without preconditioning and stop the CG iterations if the residual has been reduced by a factor *accuracy* = 10^{-6}. Give the following commands:

```
sepmesh exercise16.msh
```

```
sepcomp exercise16a.prb
sepview sepplot.001
```

Note that the residual is small but the solution is not close to the exact solution $p = 1$. Repeat the computation with $accuracy = 10^{-7}$ and $accuracy = 10^{-8}$. Inspect the solution and measure the number of iterations.

16b We now include IC preconditioning. Therefor you can use the file

```
exercise16b.prb
```

Again inspect the solution and compare the number of iterations.

16c Make a table with the number of iterations where the k_shale in exercise16b.prb is chosen equal to 10^{-2}, 10^{-4}, and 10^{-7}.

16d Repeat exercise c with deflation. The deflation choices are specified in

```
exercise16d.prb
```

16e Finally, use the file exercise16d.prb and chose k_shale equal to 10^{-7} and vary *accuracy*: 10^{-2}, 10^{-4}, 10^{-6}, and 10^{-8}. Compare the number of iterations and the quality of the solution.

Exercise 17 One dimensional finite element code

On the interval $(0, 1)$, we consider a steady-state diffusion-reaction equation, with homogeneous Neumann boundary conditions:

$$
\begin{aligned}
&-D\frac{d^2u}{dx^2} + \lambda u = f(x), \\
&-D\frac{du}{dx}(0) = 0, \qquad D\frac{du}{dx}(1) = 0.
\end{aligned}
\tag{7}
$$

Here D, and λ are positive real constants. Further, $f(x)$ is a given function.

The interval $[0, 1]$ is divided into $n - 1$ elements (where n is a given positive integer), such that $e_i = [x_i, x_{i+1}]$, for $i \in \{1, \dots, n - 1\}$. So element e_i has end points (also called vertices) x_i and x_{i+1}, where we require $x_1 = 0$ and $x_n = 1$ and $h = 1/(n - 1)$. Hence there are n gridpoints. In this lab assignment, the participant develops a finite-element code for 1D in Matlab from scratch. The treatment is formal in terms of topology, element matrices and vectors such that the student gets the idea of how finite-element packages are constructed. Once the mesh and topology have been adapted to multi-dimensional problems, then it is relatively straightforward to adjust the code to higher dimensional problems.

Assignment 1 *Derive a weak form of the above problem (see equation (1)), where the order of the spatial derivative is minimised. Take care of the boundary conditions.* $\diamondsuit$

We are going to solve this differential equation by the use of Galerkin's Finite Element method.

Assignment 2 *Write the Galerkin formulation of the weak form as derived in the previous assignment for a general number of elements given by n (hence $x_n = 1$). Give the Galerkin equations, that is, the linear system in terms of*

$$S\underline{u} = \underline{f}, \tag{8}$$

all expressed in the basis-functions, $f(x)$, λ and D. $\diamondsuit$

Assignment 3 *Write a matlab routine, called* GenerateMesh.m *that generates an equidistant distribution of meshpoints over the interval* $[0, 1]$, *where $x_1 = 0$ and $x_n = 1$ and $h = \frac{1}{n-1}$. You may use x = linspace(0,1,n).* $\diamondsuit$

Further, we need to know which vertices belong to a certain element i.

Assignment 4 *Write a routine, called* GenerateTopology.m*, that generates a two dimensional array, called* elmat*, which contains the indices of the vertices of each element, that is*

$$\begin{array}{l} elmat(i,1) = i \\ elmat(i,2) = i+1 \end{array}, \text{ for } i \in \{1, \dots, n-1\}. \tag{9}$$

◊

Next we compute the element matrix S_{elem}. In this case, the matrix is the same for each element, that is, if we consider element e_i.

Assignment 5 *Compute the element matrix, S_{elem} over a generic line element e_i.* ◊

Assignment 6 *Write a matlab routine, called* GenerateElementMatrix.m*, in which S_{elem} (2×2-matrix) is generated.* ◊

Subsequently, we are going to sum the connections of the vertices in each element matrix, over all the elements. The result is an n-by-n matrix, called S.

Assignment 7 *Write a matlab routine, called* AssembleMatrix.m*, that performs this summation, such that S is first initialized as a zero n-by-n matrix and subsequently:*

$$S(elmat(i,j), elmat(i,k)) = S(elmat(i,j), elmat(i,k)) + S_{elem}(j,k), \tag{10}$$

for $j,k \in \{1,2\}$ over all elements $i \in \{1,\dots,n-1\}$. Note that GenerateElementMatrix.m needs to be called for each element. ◊

Now, you developed a routine for the assembly of the large matrix S from the element matrices S_{elem} for each element. This procedure is common for the construction of the large discretization matrices needed in Finite Element methods. The procedure, using the array *elmat* looks a bit overdone and complicated. However, this approach facilitates the application to multi dimensional problems. The next step is to generate a large right-hand side vector using the same procedure. First, we need the element vector.

Assignment 8 *Compute the element vector over a generic line-element.* $\Diamond$

For this purpose, we proceed as follows

Assignment 9 *Implementation of the right-hand vector:*

a *Write a matlab routine, called* GenerateElementVector.m*, that gives the vector* f_{elem} *(column vector of length 2). in which* $f_{elem}(1)$ *and* $f_{elem}(2)$ *respectively provide information about node* i *and node* $i+1$*, which are the vertices of element* e_i*. This is needed for all elements. Use* $f(x) = 1$ *here.*

b *Write a matlab routine, called* AssembleVector.m*, that performs the following summation after setting* $f = zeros(n, 1)$*:*

$$f(elmat(i,j)) = f(elmat(i,j)) + f_{elem}(j), \tag{11}$$

for $j \in \{1, 2\}$ *over all elements* $i \in \{1, \ldots, n-1\}$*.*

$\Diamond$

Assignment 10 *Run the assembly routines to get the matrix* S *and vector* f *for* $n = 100$*.* $\Diamond$

Assignment 11 *Write the main program that gives the finite-element solution. Call the main program* femsolve1d.m*. The program femsolve1d.m should consist of*

clear all
GenerateMesh
GenerateTopology
AssembleMatrix
AssembleVector
$u = S\backslash f$
plot(x,u) $\Diamond$

Now, you wrote the backbone of a simple Finite Element program for a one dimensional model problem. The discretisation matrix and right-hand side vector have been constructed.

Assignment 12 *Compute the Finite Element solution* u *for* $f(x) = 1$, $D = 1$, $\lambda = 1$ *and* $n = 100$ *by using* $u = S\backslash f$ *in matlab. Plot the solution. Is this what you would expect? Compare your solution to the exact solution for various stepsizes.* $\Diamond$

Assignment 13 *Choose* $f(x) = sin(20x)$, *do some experiments with several values of* n *(n = 10, 20, 40, 80 160). Plot the solutions for the various numbers of gridnodes in one plot. Explain what you see and compare the numerical approximation to the exact solution.* $\Diamond$

You just wrote a simple finite-element code in such a way that an extension to two- and three dimensional Finite Element programs is rather straightforward. All you need to know is, which mesh points are vertices of each element. The latter distribution is commonly called the topology of the elements.