**Midterm examination Computational Geophysics**
May 27, 2015, 13:15-15:00
Room BBL 115

Solve the following problems from the printout of the course notes:

- 2.8, 2.9, 2.10, 2.11, 2.12
- 3.8, 3.10
- 4.3
- 6.3

Give clear explanations with your derivations and argumentation in answering the questions.

Note that we have used a virtual gridpoint $p_{N+2}$ at a distant $\Delta z$ outside the domain. For the temperature in this virtual point we find from (2.19) (neglecting the second order term in $\Delta z$),

$$T_{N+2} = T_N + 2\frac{q_{N+1}}{k}\Delta z \tag{2.20}$$

Using (2.20) we can formulate the discrete equation in the boundary point $p_{N+1}$, resulting in,

$$(\rho c_p)_{N+1}\frac{dT_{N+1}}{dt} = \frac{2k}{\Delta z^2}\left[T_N - T_{N+1}\right] + \frac{2k}{\Delta z}\frac{q_{N+1}}{k} + W_{N+1} \tag{2.21}$$

We see that the inhomogeneous boundary condition results in a contribution to the right-hand side vector. Note that the coefficient matrix **A** is no longer symmetric. However symmetry can easily be obtained by dividing the equation for the boundary point by 2.

**problem 2.5.** *(2.20) shows that (2.19) is equivalent to a linear extrapolation of the temperature field. Show by Taylor expansion that the approximation of the boundary condition applied in (2.19) is indeed of second order accuracy in $\Delta z$ and show that a forward difference formula results in first order accuracy.*

## 2.2 A difference method with variable grid spacing

In the derivation of the finite difference method for the heat equation based on central difference approximation of the conduction term we used an equidistant grid of nodal points. We further assumed that the thermal conductivity coefficient was a constant.

To obtain sufficient accuracy in the numerical solution it may be necessary to use many gridpoints in a high resolution mesh, resulting in larger program requirements for memory and compute time. Such mesh refinement is applied on the whole domain in case of an equidistant grid whereas increased resolution may be necessary only on part of the domain where the solution shows strong variations (large gradient). It is clear that in such cases using equidistant grids is not efficient and methods allowing local grid refinement will be more efficient. Different methods exist allowing local refinement. In later chapters we focus on so called finite element methods which offer the most flexibility in local grid refinement of well known discretization methods.

As an example of a method allowing local grid refinement we treat here an other difference method which also includes a simple treatment of variable coefficients $k(z)$. We restrict ourselves again to the 1-D case. A 2-D generalization is introduced in Chapter 3. The method introduced here is known in the literature as a *finite volume method*. We first deal with the steady state problem and shall verify afterwards how this can be extended for time dependent problems.

### 2.2.1 Discretization of the equation

In the finite volume method the partial differential equation (PDE) is integrated over small grid cell's, the so called finite volumes. In our 1-D case the finite volumes are subintervals $I_i$, of the complete domain, the interval $I = [0, L]$.

These subintervals $I_i = [z_{m_{i-1}}, z_{m_i}]$, centered at nodalpoint $p_i$, are illustrated Fig.2.

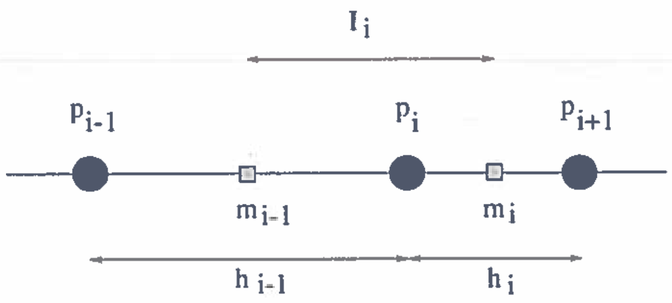

***Figure 2.2:*** *Detail of a 1-D grid with two grid segments and a integration interval of finite volume $I_i$.*

The mid-points of the grid cells, which span the finite volumes, are labeled $m_i$. Integration of the steady state heat equation over $I_i$ yields,

$$\int_{z_{m_{i-1}}}^{z_{m_i}} \left(\frac{d}{dz}k(z)\frac{d}{dz}T + \rho(z)H(z)\right) dz =$$
$$\left[k(z)\frac{dT}{dz}\right]_{z_{m_{i-1}}}^{z_{m_i}} + \int_{z_{m_{i-1}}}^{z_{m_i}} \rho(z)H(z)dz = 0 \tag{2.22}$$

The remaining derivative in (2.22) is evaluated in the mid-points $m_{i-1}$ and $m_i$. We approximate these derivatives by their central difference approximations in terms of the neighboring nodal point values and we define $k(z_{m_i}) = k_i$ and $z_{p_{i+1}} - z_{p_i} = h_i$.

$$\left(k\frac{dT}{dz}\right)_{z_{m_i}} \approx k_i\frac{T_{i+1}-T_i}{z_{p_{i+1}}-z_{p_i}} = k_i\frac{T_{i+1}-T_i}{h_i} \tag{2.23}$$

$$\left(k\frac{dT}{dz}\right)_{z_{m_{i-1}}} \approx k_{i-1}\frac{T_i-T_{i-1}}{z_{p_i}-z_{p_{i-1}}} = k_{i-1}\frac{T_i-T_{i-1}}{h_{i-1}} \tag{2.24}$$

The distribution of the heat productivity $H$ is assumed to be known and we define,

$$\int_{z_{m_{i-1}}}^{z_{m_i}} \rho(z)H(z)dz = F_i \tag{2.25}$$

Substitution of (2.23),(2.24) and (2.25) in (2.22) gives,

$$-\frac{k_{i-1}}{h_{i-1}}T_{i-1} + \left(\frac{k_{i-1}}{h_{i-1}} + \frac{k_i}{h_i}\right)T_i - \frac{k_i}{h_i}T_{i+1} = F_i \tag{2.26}$$

Equation (2.26) is a linear algebraic equation in the unknown nodal point values of the temperature $T_i$. By repeating the integration proces for all $N$ finite volumes $I_i$ we obtain a system of linear equations.

The resulting system of equations written in matrix form is,

$$\mathbf{AT} = \mathbf{F} \tag{2.27}$$

**problem 2.6.** *Verify that the system of equations obtained above is complete in case of prescribed boundary temperatures in $z = 0$ and $z = L$.*

**problem 2.7.** *Show that the system of equations build from (2.26) is identical to the equations obtained in the previous section for the special case of an equidistant grid and uniform coefficient $k$ and piecewise uniform internal heating $H$ (see problem 2.9).*

**problem 2.8.** *Show that the matrix* $\mathbf{A}$ *is a symmetric tri-diagonal matrix, i.e.* $A_{ij} = A_{ji}$, $A_{ij} = 0, |i-j| > 1$.

**problem 2.9.** *Suppose we whish to apply the finite volume method to a 1-D medium consisting of a stack of layers with uniform conductivity in each layer and that the conductivity is discontinuous in the layer interfaces. The conductivity model in this case is said to be piecewise uniform and consists of a list of discrete layer conductivity values* $k_i$. *Where would you put the nodal points in this model such that all the necessary entities in the derivation above are well defined?*

**problem 2.10.** *Suppose we define the heat productivity coefficient* $H(z)$ *by a piecewise constant model. Verify the following formula for the righthand side vector elements,*

$$F_i = \int_{z_{m_{i-1}}}^{z_{m_i}} \rho(z)H(z)dz = \frac{h_{i-1}}{2}\rho(z_{m_{i-1}})H(z_{m_{i-1}}) + \frac{h_i}{2}\rho(z_{m_i})H(z_{m_i}) \tag{2.28}$$

**problem 2.11.** *Assume that the heatproductivity is concentrated in a point,* $z = z_s$, $W(z) = \rho(z)H(z) = W_s\delta(z - z_s)$, *with* $z_{m_{k-1}} < z_s < z_{m_k}$, *i.e.* $z_s \in I_k$.

*Derive for the righthand side vector elements,* $F_i = W_s\delta_{ik}$, *where* $\delta_{ik}$ *is the Kronecker delta symbol.*

**problem 2.12.** *Investigate how the steady state equation (2.27) can be extended to a set of ODE's similar to (2.14) for the time dependent case. How would you treat a case with variable heat capacity* $\rho c_p$ *in this extension?*

### 2.2.2 Implementation of boundary conditions

Essential boundary conditions are implemented in the same way as in the discretization method of section 2.1. Essential boundary conditions result in a contribution to the right-hand side vector and a reduction of the number of degrees of freedom of the problem. Here we describe the implementation of natural or Neumann boundary conditions that are used in case the heatflow density is prescribed on the boundary.

We define $(kdT/dz)_{z=z_{N+1}} = q_{N+1}$. The implementation differs from the one described in section 2.1.1 for an equidistant grid. Here we derive the implementation for the boundary condition in nodal point $p_{N+1}$, illustrated in Fig.3.

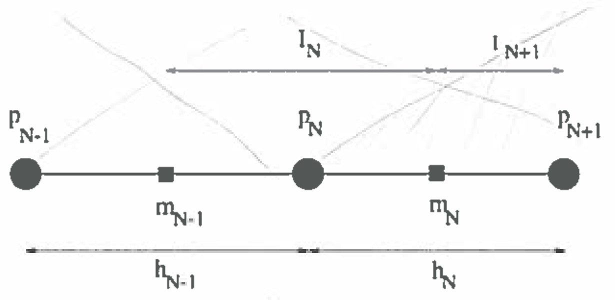

***Figure 2.3:*** *Detail of a 1-D grid with two grid segments including a boundary point and two finite volumes.*

In this case the temperature in the boundary point $p_{N+1}$ is also a degree of freedom of the problem. In order to obtain a complete set of equations we need to introduce an extra equation by integrating over a (half) finite volume from $m_N$ to $p_{N+1}$. We apply (2.22) for the nodal points $p_N$ and $p_{N+1}$.

nodal point values of the potential function. The matrix corresponding to these equations is defined **S**.

The bandwidth $w$ of this matrix **S** is defined by the location of non-zero diagonals in the upper and lower triangle matrix, excluding the main diagonal,

$$w = max|I - J|, \quad S_{IJ} \neq 0 \tag{3.10}$$

According to this definition a diagonal matrix has a zero bandwidth and tri-diagonal matrix has a bandwidth of one. Interpreting the expression (3.9) as a matrix row in a system of linear algebraic equations it follows that the bandwidth $w = n_{row}$. From this we find that the smallest bandwidth is obtained by defining the grid columns in the direction of the smallest dimension of the rectangular domain. Efficient algorithms are available for the solution of systems of linear algebraic equations that are based on a so called bandmatrix structure where only matrix elements within the bandwidth defined in (3.10) are stored in computer memory. Minimizing the bandwith of the matrix will result in minimizing the computer requirements (memory, compute time) when using such bandmatrix solvers.

**problem 3.3.** *The operator $D_h^2$ in (3.9), applied to the nodal point values of a uniform grid, produces a system of linear algebraic equations for a discrete approximation of (3.1). Consider the special case of a rectangular grid with two rows of internal gridpoints and apply a (grid) columnwise numbering of the degrees of freedom of the problem as in (3.8).*
*What is the structure of the matrix* **S** *of the resulting system?*

## 3.3 A difference method for variable grid spacing and variable coefficient

The difference formula for the Laplace operator (3.6) is derived for the special case of a uniform coefficient $c$ and does not apply to the more general case with differential operator $L = \nabla \cdot c(\mathbf{x})\nabla u$. Besides this the grid used in section 3.2 is equidistant.

Here we introduce the more general case with variable coefficient and apply a so called structured grid, defined by the *product* of two 1-D grids with variable nodal point spacing in both coordinate directions. This allows local mesh refinement. We consider a 2-D rectangular domain subdivided in rectangular grid cells spanned by the nodalpoints, illustrated in Fig. 3.1. Similar as in 2.2, but now for a 2-D domain, we integrate the PDE (3.1) over a small area, a finite volume, surrounding a single gridpoint $P_0$, illustrated in Fig. 3.1.

Relative coordinates of neighboring gridpoints of $P_0$, shown in Fig. 3.1, are parameterized as follows,

$$\begin{aligned} P_0 &= (0,0) \\ P_1 &= (s_1 h, 0) \\ P_2 &= (0, s_2 h) \\ P_3 &= (-s_3 h, 0) \\ P_4 &= (0, -s_4 h), \quad 0 < s_K \leq 1, \quad K = 1,\ldots,4 \end{aligned} \tag{3.11}$$

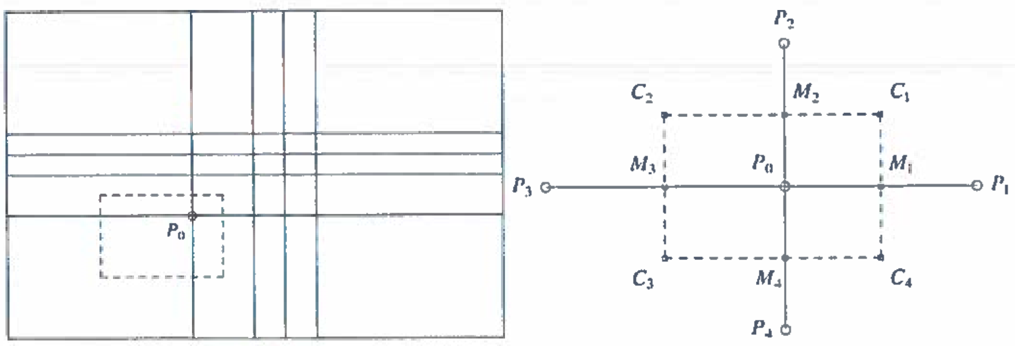

***Figure 3.1:*** *Left: rectangular computational domain showing local mesh refinement and a single (dashed) finite volume. Right: zoom-in finite volume with five grid points in a mesh with variable grid point spacing. $M_K, K = 1,\dots,4$ are midpoints positioned halfway two neighboring gridpoints $P_0, P_K$. A difference equation is derived by integration over the rectangle spanned by the corner points $C_1,\dots,C_4$.*

The PDE (3.1) is integrated over the rectangular area $V$ spanned by the corner points $C_1,\dots,C_4$ in Fig.3.1, that are the centre points of the neighboring grid cells.

$$I = \int_V \nabla \cdot c(\mathbf{x})\nabla u \; dV = \int_{\partial V} c(\mathbf{x})\partial_j u \; n_j \; dA = -\int_V f(\mathbf{x})dV \tag{3.12}$$

where $\partial V$ is the closed boundary curve $C_1, C_2, C_3, C_4, C_1$. The contribution from the vertical boundary segments in (3.12) is

$$I_1 - I_3 = \int_{-s_4h/2}^{s_2h/2} (c(\mathbf{x})\partial_x u)_{x=s_1h/2} \; dy - \int_{-s_4h/2}^{s_2h/2} (c(\mathbf{x})\partial_x u)_{x=-s_3h/2} \; dy \tag{3.13}$$

Both integrals in (3.13) are approximated using a 'mid-point rule' and the remaining partial derivative is replaced by a central difference approximation,

$$\partial_x u(M_1) \approx \frac{u(P_1) - u(P_0)}{s_1 h} \tag{3.14}$$

The first integral in (3.13) results in,

$$I_1 \approx c(M_1)\,(u(P_1) - u(P_0))\,\frac{s_2 + s_4}{2s_1} \tag{3.15}$$

In a similar way we find,

$$I_3 \approx c(M_3)\,(u(P_0) - u(P_3))\,\frac{s_2 + s_4}{2s_3} \tag{3.16}$$

The horizontal boundaries result in similar contributions,

$$I_2 = \int_{-s_3h/2}^{s_1h/2} (c(\mathbf{x})\partial_y u)_{y=s_2h/2} \; dx \approx c(M_2)\,(u(P_2) - u(P_0))\,\frac{s_1 + s_3}{2s_2} \tag{3.17}$$

$$I_4 = \int_{-s_3h/2}^{s_1h/2} (c(\mathbf{x})\partial_y u)_{y=-s_4h/2} \; dx \approx c(M_4)\,(u(P_0) - u(P_4))\,\frac{s_1 + s_3}{2s_4} \tag{3.18}$$

Putting the segment contributions together [2] in,

$$I = I_1 - I_3 + I_2 - I_4 \tag{3.19}$$

we obtain,

$$I \approx \sum_{K=1}^{4} \alpha_K u(P_K) - \alpha_0 u(P_0) \tag{3.20}$$

The coefficients in (3.20) are given in the Table.

| $K$ | $\alpha_K$ |
|---|---|
| 0 | $\sum_{K=1}^{4} \alpha_K$ |
| 1 | $c(M_1)(s_2+s_4)/2s_1$ |
| 2 | $c(M_2)(s_1+s_3)/2s_2$ |
| 3 | $c(M_3)(s_2+s_4)/2s_3$ |
| 4 | $c(M_4)(s_1+s_3)/2s_4$ |

**Table 3.1:** *Coefficients of the five-point finite difference 'molecule'.*

Using a 2-D mid-point rule, the right hand side term in (3.12) is approximated by,

$$\int_V f(\mathbf{x})\, dV \approx f(P_0)(s_1+s_3)(s_2+s_4)\frac{h^2}{4} = F \tag{3.21}$$

For the special case of an equidistant mesh the above results reduce to,

$$s_K = 1 \quad \rightarrow \quad \alpha_K = \begin{cases} c(M_K) & ,K = 1,\ldots,4 \\ \sum_{J=1}^{4} c(M_J) & ,K = 0 \end{cases} \quad , \; F = h^2 f(P_0) \tag{3.22}$$

Combination of (3.1),(3.20) and (3.21) results in the following difference equation for the nodalpoint $P_0$,

$$\alpha_0 u(P_0) - \sum_{K=1}^{4} \alpha_K u(P_K) = F \tag{3.23}$$

By evaluating the finite difference formula (3.23) for every nodal point we obtain a system of linear algebraic equations that can be solved numerically. Each nodal point corresponds to a single equation in this set or to a single row of the corresponding coefficient matrix.

**problem 3.4.** *Verify that the difference formula (3.23) corresponds to the five-point formula derived in the previous section in the special case of an equidistant mesh and a uniform coefficient* $(c(\mathbf{x}) = c)$,

$$4u(P_0) - \sum_{K=1}^{4} u(P_K) = \frac{h^2 f(P_0)}{c} \tag{3.24}$$

**problem 3.5.** *Extend the derivation of the finite difference formula (3.23) derived for the steady state heat conduction problem (3.1) for the time dependent problem described by,*

$$\rho c_p \frac{\partial T}{\partial t} = \nabla \cdot k \nabla T + H \tag{3.25}$$

*Hint: Consider a semi-discretization, leaving the continuous time variable in place to derive a system of first order ordinary differential equations similar to the 1-D case (2.14).*

[2]The minus sign for the contributions $I_3, I_4$ accounts for the direction of the outward pointing normal vector on the corresponding boundary segments. $\nabla u \cdot \mathbf{n} = -\partial_x u$ on the left hand vertical boundary segment and $\nabla u \cdot \mathbf{n} = -\partial_y u$ on the bottom boundary.

**problem 3.6.** *Derive an expression for the right hand side vector element $F$ for the case where the righthand side function represents a point source $f(\mathbf{x}) = a\delta(\mathbf{x}-\mathbf{x}_s)$, where $a$ is a constant.*

Figure 3.2 shows the structure diagram of an algorithm for filling the coefficient matrix that follows from evaluation of (3.24) in all the internal points of the mesh for the special case of an equidistant mesh and uniform coeffient $c(\mathbf{x}) = 1$. A 2-D rectangular domain $V$ is used and a grid consisting of $n_{col} + 2$ columns and $n_{row} + 2$ rows of nodal points. Furthermore the algorithm assumes that essential boundary conditions are given for all boundary points ($\partial V = \Gamma_g$, $\Gamma_h = \emptyset$). In that case the discretized problem has $n_{row} \times n_{col} = N$ degrees of freedom - one for each internal nodal point. The degrees of freedom are numbered columnwise in the grid of nodal points. This way an $N$-vector $\mathbf{U}$ is defined of unknown nodal point values.

$$\begin{aligned} \mathbf{U} = \ & (u(x_1,y_1), u(x_1,y_2), \dots, u(x_1,y_{n_{row}}), \\ & \dots, \\ & u(x_{n_{col}},y_1), u(x_{n_{col}},y_2), \dots, u(x_{n_{col}},y_{n_{row}}), )^T \end{aligned} \tag{3.26}$$

In this case with essential boundary conditions on the complete boundary, evaluating the difference equation (3.24) in every nodal point results in a complete system of equations.

**problem 3.7.** *Verify that the matrix of the above finite difference equations is symmetric and that the matrix rows outside the main diagonal and four other diagonals contain zero values. Show for the bandwitdh in (3.10): $w = n_{row}$.*

**problem 3.8.** *Extend the algorithm of Fig. 3.2 with the computation of a right hand side vector for the system of finite difference equations.*

**problem 3.9.** *Verify how the symmetry of the matrix can be applied to optimize the algorithm of Fig. 3.2.*

**problem 3.10.** *How could the algorithm of Fig. 3.2 be modified for the case of variable coefficient $c(\mathbf{x})$?*
*Hint: consider the equidistant case and apply (3.22).*

**problem 3.11.** *How could the algorithm of Fig. 3.2 be extended for the case of variable coefficient $c(\mathbf{x})$ and variable grid spacing?*

## 3.4 Implementation of boundary conditions

We distinguish between essential boundary conditions with prescribed values of the solution $u(\mathbf{x})$ and natural boundary conditions where the normal component of the gradient $c(\mathbf{x})\nabla u \cdot \mathbf{n}$ is prescribed in boundary points $\mathbf{x}$.

### 3.4.1 Essential boundary conditions

An implementation of essential boundary conditions follows directly from the difference equation,

$$\alpha_0 u(P_0) - \sum_{K=1}^{4} \alpha_K u(P_K) = F \tag{3.27}$$

Terms in (3.27) with prescribed values of $u$ in boundary points $\mathbf{x}_K \in \Gamma_g$ can be moved to the right hand side of the equation. Essential boundary conditions thus contribute to

```
ncol - number columns internal nodal points
nrow - number rows internal nodal points
idof - sequence number degree of freedom for nodal point (irow,jcol)

for a constant coefficient and equidistant mesh:
e0 =  4. - main diagonaal element difference formula
e1 = -1. - elements of second diagonal

|--------------------------------------------------------------------------|
| *loop over columns internal nodal points                                 |
|--------------------------------------------------------------------------|
| do jcol = 1, ncol                                                        |
|    |---------------------------------------------------------------------|
|    | *loop over rows internal nodal points                               |
|    |---------------------------------------------------------------------|
|    | do irow = 1, nrow                                                   |
|    |    |----------------------------------------------------------------|
|    |    | *seq. number d.o.f. central point                              |
|    |    | idof = (jcol-1)*nrow + irow                                    |
|    |    |----------------------------------------------------------------|
|    |    | *left                                                          |
|    |    |----------------------------------------------------------------|
|    |    |            T              jcol > 1             F               |
|    |    |-----------------------------------|----------------------------|
|    |    | jdof = idof - nrow                | * column 1 nod.point       |
|    |    | elmat = e1                        |   zero contrib. matrix     |
|    |    | call fillmat(elmat,idof,jdof,matrix)|                          |
|    |    |-----------------------------------|----------------------------|
|    |    | *right                                                         |
|    |    |----------------------------------------------------------------|
|    |    |            T              jcol < ncol          F               |
|    |    |-----------------------------------|----------------------------|
|    |    | jdof = idof + nrow                | * last column nod.point    |
|    |    | elmat = e1                        |   zero contrib. matrix     |
|    |    | call fillmat(elmat,idof,jdof,matrix)|                          |
|    |    |-----------------------------------|----------------------------|
|    |    | *check above                                                   |
|    |    |----------------------------------------------------------------|
|    |    |            T              irow < nrow          F               |
|    |    |-----------------------------------|----------------------------|
|    |    | jdof = idof + 1                   | * top row of nodal points  |
|    |    | elmat = e1                        |   zero contrib. matrix     |
|    |    | call fillmat(elmat,idof,jdof,matrix)|                          |
|    |    |-----------------------------------|----------------------------|
|    |    | *check below                                                   |
|    |    |----------------------------------------------------------------|
|    |    |            T              irow > 1             F               |
|    |    |-----------------------------------|----------------------------|
|    |    | jdof = idof - 1                   | * bottom row of nodal points |
|    |    | elmat = e1                        |   zero contrib. matrix     |
|    |    | call fillmat(elmat,idof,jdof,matrix)|                          |
|    |    |----------------------------------------------------------------|
|    |    | *central point (diagonal matrix element)                       |
|    |    | jdof = idof                                                    |
|    |    | elmat = e0; call fillmat(elmat,idof,jdof,matrix)               |
|----|----|----------------------------------------------------------------|
```

***Figure 3.2:*** *Structure diagram of an algorithm to fill the coefficient matrix of the finite difference equations. The subroutine* `fillmat` *is used for storing the matrix elements in an array* `matrix`. *This way the sparse structure of the matrix can be exploited in an easy way.*

the right hand side vector of the system of equations. This can be made more explicit by partitioning the vector of nodal point values $\mathbf{U} = (\mathbf{U}_f, \mathbf{U}_p)^T$, where $\mathbf{U}_p$ is the vector of prescribed (boundary) nodal point values, and $\mathbf{U}_f$ the vector of remaining (free) unknown nodal point values, the degrees of freedom. The matrix $\mathbf{S}$ and right hand side vector $\mathbf{F}$ partition correspondingly,

$$\begin{pmatrix} \mathbf{S}_{ff} & \mathbf{S}_{fp} \end{pmatrix} (\mathbf{U}_f, \mathbf{U}_p)^T = \mathbf{F}_f \tag{3.28}$$

By writing the multiplications of the partitioned matrix blocks in (3.28) explicitly we see that the vector part of unknown nodal point values $\mathbf{U}_f$ can be solved from the following reduced system of equations,

$$\mathbf{S}_{ff}\mathbf{U}_f = \mathbf{F}_f - \mathbf{S}_{fp}\mathbf{U}_p = \mathbf{R}_f \tag{3.29}$$

Note that $\mathbf{F}_p$ does not occur in (3.29).

**problem 3.12.** *How could the algorithm in Fig. 3.2 be extended to account for the contribution of inhomogeneous essential boundary conditions in the right hand side vector?*

### 3.4.2 Natural boundary conditions

Implementation of natural boundary conditions is less straight forward. It is clear that the number of degrees of freedom of the problem is now greater than in the previous case since the nodal point values corresponding to points $\mathbf{x}_K \in \Gamma_h$ are also degrees of freedom. In order to get a complete set of equations, difference equations must be formulated that include these degrees of freedom for $\mathbf{x}_K \in \Gamma_h$. This is done by integrating the differential equation over finite volumes associated with the boundary points $\mathbf{x}_K \in \Gamma_h$, as illustrated in Fig. 3.3. In the integration over the vertical boundaries of the cell, the integral $I_3$ over the segment $M_2M_4 \subset \Gamma_h$ can be expressed in the known boundary value $c(P_0)\partial_x u(P_0)$. This results in a contribution to the right hand side vector. Note that here the integration is over a reduced area compared to interior grid cells. The expressions for the resulting matrix coefficients differ from the ones for interior nodal points.

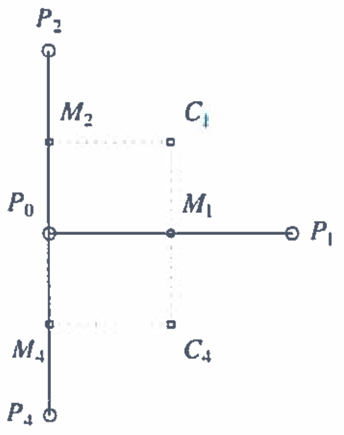

**Figure 3.3:** *Integration over a grid cell associated with a boundary point $\mathbf{x}_K \in \Gamma_h$. The interior of the computational domain is on the right hand side of the boundary segment $P_2, P_0, P_4$. The grid line $P_2P_0P_4$ is part of the bondary $\Gamma_h$ with natural boundary conditions.*

(4.4) of the solution $u$ is defined as an interpolation in terms of the nodal point values $u(x_J) = U_J$,

$$u(x) \approx u^h(x) = \sum_{J=1}^{N} U_J N_J(x) \tag{4.13}$$

This is illustrated for the one-dimensional linear element with two nodal points per element ($n_e = 2$),

$$l_1^1(x) = \frac{x - x_2}{x_1 - x_2} = \frac{1}{h}(x_2 - x) \tag{4.14}$$

$$l_2^1(x) = \frac{x - x_1}{x_2 - x_1} = \frac{1}{h}(x - x_1) \tag{4.15}$$

Both parts of this piecewise linear basis function are illustrated in Fig. 4.1.

**problem 4.3.** *Show that the so called trapezoidal rule for approximation of integrals* [2] *follows directly from an expansion as in (4.13), using equidistant evaluation points $x_J$. Also derive a corresponding trapezoidal rule for the general case with variable grid spacing.*

## 4.2 Discretization of the differential equation

The finite element method is introduced here as a special case of the method of Galerkin for solving differential equations. Galerkins method is defined in the following way: for a given differential equation $Lu = f$, the residual function $R = Lu - f$ is multiplied by a weighting function $w_I$ and integrated over the domain,

$$\int_V w_I\,(Lu - f)\;dV = 0, \quad I = 1, 2, \ldots \tag{4.17}$$

The linearly independent weighting functions $w_I(\mathbf{x})$ span a linear function space $S$. When the integral expression in (4.17) is interpreted as a special case of the general functional innerproduct of the functions $p,\ q \in S$,

$$(p \cdot q) = \int_V p(\mathbf{x}) q(\mathbf{x})\;dV \tag{4.18}$$

then the equation (4.17), $(w_I \cdot R) = 0$, specifies the condition for the solution $u$ that the residue of the differential equation is orthogonal to the vector space $S$ spanned by the weighting functions $w_I$, in the sense of the inner product (4.18).

We shall further use the special case where the weighting functions and the basis functions $N_J$ used in the expansion [3] of the solution $u$ are identical. This is known in the

[2]

$$I = \int_a^b f(x)dx \approx I^h = \sum_{J=1}^{N} w_j f(x_J), \; w_j = \begin{cases} \Delta x/2, & J = 1|N \\ \Delta x, & 1 < J < N \end{cases} \tag{4.16}$$

`http://en.wikipedia.org/wiki/Trapezoidal_rule`

[3]For the general $m$-dimensional case a generalization of (4.13) is used,

$$u(\mathbf{x}) \approx u^h(\mathbf{x}) = \sum_{J=1}^{N} U_J N_J(\mathbf{x}) \tag{4.19}$$

Where $u(\mathbf{x})$ and $N_J(\mathbf{x})$ are functions of the $m$ spatial coordinates $\mathbf{x} = (x_1, \ldots, x_m)$ of the $m$-dimensional domain.

*$T_m$ and $T_0$ respectively. We assume a horizontal symmetry condition on the vertical boundaries. Which formulations from (6.3,6.4) shall we choose to implement the boundary conditions on the four boundaries of the domain? How does this change if we prescribe the mantle heatflow $\mathbf{q}_m$ instead of the mantle temperature?*

An example of the mixed type boundary condition (6.5) is found in heat tranport problems in cases where the heatflow density through the boundary $\Gamma_i$ is assumed to be proportional to the temperature contrast across the boundary (a so called radiation condition). With the heatflow density defined by $\mathbf{q} = -c\nabla u$, we obtain from (6.5),

$$q_n = \alpha \left( u - \frac{r}{\alpha} \right) \tag{6.6}$$

we see that $r/\alpha$ acts as a reference temperature in defining the temperature contrast driving the heatflow across the boundary. $\alpha$ can be interpreted as the inverse of a *thermal resistance* coefficient.

**problem 6.2.** *We whish to model numerically a physical (lab) Raylcigh-Benard thermal convection experiment. In this experiment viscous fluid in a tank is heated from below and cooled from the top. The fluid layer of thickness h is bounded below and above by copper layers of thickness l. The temperature of the exterior surface of the copper layers is kept at constant values $T_0 + \Delta T$ and $T_0$ for the bottom and top respectively by separate circuits of heating/cooling liquid in contact with the bottom and top copper plates.*

*How can we apply boundary condition type (6.5) to this problem.*

*Hint: neglect horizontal heat transport in the copper plates and assume that the heatflow density $q_n(x)$ can be described in terms of the local temperature contrast across the copper plates.*

## 6.1 Discretization of the equation

We shall first describe a finite element solution for the elliptic problem based on general element types for 2-D or 3-D problems similar to the description in Chapter 4. In later sections more detailed examples will be given of such solutions for 2-D triangular elements combined with linear basis functions and quadrilateral elements with bi-linear basis functions.

In the Bubnov-Galerkin formulation the partial differential equation is transformed by integration over the domain, resulting in a system of linear algebraic equations.

$$\int_V N_I \{-\nabla \cdot c\nabla u - f\} \; dV = 0, \; I = 1, \ldots, N \tag{6.7}$$

Where $N$ is the number of degrees of freedom. Integrating by parts gives,

$$\int_V \{-\nabla \cdot (N_I c\nabla u) + \nabla N_I \cdot c\nabla u - N_I f\} \; dV = 0 \tag{6.8}$$

$$-\int_{\partial V} N_I c\nabla u \cdot \mathbf{n} \; dA + \int_V \nabla N_I \cdot c\nabla u \; dV = \int_V N_I f \; dV, \; I = 1, \ldots, N \tag{6.9}$$

Substitution of the expansion in interpolating Lagrangian basis functions,

$$u(\mathbf{x}) = \sum_J U_J N_J(\mathbf{x}), \; U_J = u(\mathbf{x}_J) \tag{6.10}$$

we get for the second term in (6.9)

$$\sum_J \left\{ \int_V \nabla N_I \cdot c\nabla N_J \; dV \right\} U_J = \sum_J S_{IJ} \; U_J \tag{6.11}$$

where the stiffness matrix $\mathbf{S}$ is defined by,

$$S_{IJ} = \int_V \nabla N_I \cdot c\nabla N_J \, dV \tag{6.12}$$

The expression for $\mathbf{S}$ is often rewritten in a different form that is also used in later chapters on vector problems dealing with elastic deformation and viscous flow. To this end we introduce a matrix $\mathbf{B}$ where the matrix columns are defined in terms of the gradient of the finite element basis functions. For a 3-D problem this gives,

$$\mathbf{B} = (\nabla N_1, \ldots, \nabla N_N) = \begin{pmatrix} \partial_1 N_1, \ldots, \partial_1 N_N \\ \partial_2 N_1, \ldots, \partial_2 N_N \\ \partial_3 N_1, \ldots, \partial_3 N_N \end{pmatrix} \tag{6.13}$$

or alternatively,

$$\mathbf{B} = \begin{pmatrix} \partial_1 \cdot \\ \partial_2 \cdot \\ \partial_3 \cdot \end{pmatrix} (N_1, \ldots, N_N) \tag{6.14}$$

The column vectors of the matrix $\mathbf{B}$ are,

$$\mathbf{B}_I = \nabla N_I = (\partial_1 N_I, \partial_2 N_I, \partial_3 N_I)^T \tag{6.15}$$

For the coefficients of the stiffness matrix we get,

$$S_{IJ} = \int_V c\nabla N_I \cdot \nabla N_J \, dV = \int_V \mathbf{B}_I \cdot \mathbf{D}\mathbf{B}_J \, dV = \int_V \mathbf{B}_I^T \mathbf{D}\mathbf{B}_J \, dV \tag{6.16}$$

where $\mathbf{B}_I$ and $\mathbf{B}_J$ are columnvectors and $D_{ij} = c\delta_{ij}$, $i, j = 1, 2, 3$. The global stiffness matrix can be written as a summation of matrices and assembled from the contributions of element matrices,

$$\mathbf{S} = \int_V \mathbf{B}^T \mathbf{D}\mathbf{B} \, dV = \sum_K \int_{e_K} \mathbf{B}^T \mathbf{D}\mathbf{B} \, dV = \sum_K \mathbf{S}^{(K)} \tag{6.17}$$

where $\mathbf{S}^{(K)}$ is the element matrix of element $e_K$. The summation over elements corresponds to the matrix assembly proces described for 1-D cases in Chapter 5.

**problem 6.3**. *Derive the following expressions for the element matrix $\mathbf{S}^{(K)}$ for a 1-D element with two degrees of freedom and a 2-D triangular element with three degrees of freedom. For the 1-D element,*

$$\mathbf{S}^{(K)} = \int_{e_K} c \begin{pmatrix} \frac{dN_1}{dz}\frac{dN_1}{dz} & \frac{dN_1}{dz}\frac{dN_2}{dz} \\ \frac{dN_2}{dz}\frac{dN_1}{dz} & \frac{dN_2}{dz}\frac{dN_2}{dz} \end{pmatrix} dz \tag{6.18}$$

*And for a 2-D traingular element, associated with 3 degrees of freedom and 3 basis functions introduced in section 6.2.1,*

$$\mathbf{S}^{(K)} = \int_{e_K} c \begin{pmatrix} (\partial_x N_1)^2 + (\partial_y N_1)^2 & \partial_x N_1 \partial_x N_2 + \partial_y N_1 \partial_y N_2 & \partial_x N_1 \partial_x N_3 + \partial_y N_1 \partial_y N_3 \\ \partial_x N_2 \partial_x N_1 + \partial_y N_2 \partial_y N_1 & (\partial_x N_2)^2 + (\partial_y N_2)^2 & \partial_x N_2 \partial_x N_3 + \partial_y N_2 \partial_y N_3 \\ \partial_x N_3 \partial_x N_1 + \partial_y N_3 \partial_y N_1 & \partial_x N_3 \partial_x N_2 + \partial_y N_3 \partial_y N_2 & (\partial_x N_3)^2 + (\partial_y N_3)^2 \end{pmatrix} dxdy$$

*where local numbering (per element) of the basis functions has been applied.*