3-D FEM Discretization of Poisson Equation on Tetrahedral and Hexahedral Meshes.

A.Smirnov

July 13, 2004

Contents

1 Governing Equations

Consider the boundary value problem:

$$\begin{aligned} \Delta P(\mathbf{x}) &= f(\mathbf{x}) \\ P(\mathbf{x}) \mid_{\mathbf{x}\in\partial_0 D} &= f^0(\mathbf{x}) \qquad (1) \\ \frac{\partial P(\mathbf{x})}{\partial \mathbf{n}}\mid_{\mathbf{x}\in\partial_1 D} \equiv (\nabla P\cdot\mathbf{n})_{\mathbf{x}\in\partial_1 D} &= f^1(\mathbf{x}) \qquad (2) \end{aligned}$$

where $(*\cdot *)$ denotes a scalar product operation on two vectors, $\mathbf{x}$ represents a vector of coordinates $\mathbf{x} = \{x_i\}, i = 1 : N_d$ in N_d dimensional space ($N_d = 3$), $\mathbf{n}$ is the normal vector to the boundary $\partial_i D$ of the domain D with $i = 0$ representing Dirichlet and $i = 1$ - Neuman boundary.

In what follows we consider 3D discretization schemes for this problem based on finite-element (FEM) and control volume (CVM) methods.

2 A 3-D FEM Discretization Scheme for the Poisson Equation

In this section we consider the Finite-Element discretization scheme for equation system (1)-(2) used to solve vertex-based scalar field variables on a tetrahedral mesh.

2.1 Poissson equation in a general form

Let's represent the functions used in the boundary value problem (1) as a super-position of the basis functions $\phi_i(\mathbf{x})$. This superposition will apply to both the solution variable $P(\mathbf{x})$ and source functions, $f(\mathbf{x}), f^k(\mathbf{x})_{k=0,1}$:

$$\begin{aligned} P(\mathbf{x}) &= \sum_{i=1}^{N} p_i\phi_i(\mathbf{x}) \\ f(\mathbf{x}) &= \sum_{i=1}^{N} f_i\phi_i(\mathbf{x}) \qquad (3) \end{aligned}$$

and the same for the boundary values $f^0(\mathbf{x})$ and fluxes $f^1(\mathbf{x})$

$$f^k(\mathbf{x}) = \sum_{i=1}^{N} f_i^k \phi_i(\mathbf{x}), \quad k = 0,1 \tag{4}$$

where N is the number of grid nodes. Each function $\phi_i(\mathbf{x})$ is a linear function inside the element containing node i and zero in all the elements not containing node i. Inside the elements containing node i $\phi_i(\mathbf{x})$ is unity at the node i and zero at the face lying opposite to the node. For example, in Fig.1 the $\phi_0 = 1$ at node 0 and zero at face $(1,2,3)$. The distribution of the function values inside the tetrahedron is linear, with $\phi_0(\mathbf{x})$ being constant on the planes parallel to $(1,2,3)$, like plane $(1',2',3')$, and equal to $\phi_0 = 1 - \xi$, where ξ is a continuous parameter between 0 and 1, with 0 corresponding to node 0 and 1 to the plane $(1,2,3)$. Thus, if h is a distance between node 0 and plane $(1,2,3)$ then ξh is a distance between the node 0 and plane $(1',2',3')$. Each function $\phi_i(\mathbf{x})$ is piece-wise continuous in the whole domain (first derivatives are discontinuous), and their total number is finite and equal to N. We shall denote the space of these functions as C_N^0.

Multiplying (1) by a test function $\psi = \psi(\mathbf{x})$, integrating over the whole domain D, and using integration-by-parts to get rid of Laplacian operator (Δ), we get[1]

$$\int_D \psi \Delta P \, d^3\mathbf{x} = \oint_{\partial D} \psi (\nabla P \cdot \mathbf{n}) \, d^2\mathbf{x} - \int_D (\nabla \psi \cdot \nabla P) \, d^3\mathbf{x} = \int_D \psi f \, d^3\mathbf{x} \tag{5}$$

where $\mathbf{n}$ is a unit vector normal to the domain boundary ∂D, and $d^2\mathbf{x}$ is a boundary surface area element. Introducing boundary conditions (2) into the boundary integral of (5), we have

$$\int_D (\nabla \psi \cdot \nabla P) \, d^3\mathbf{x} =$$
$$= \oint_{\partial_0 D} \psi (\nabla P \cdot \mathbf{n}) \, d^2\mathbf{x} + \oint_{\partial_1 D} \psi f^1 \, d^2\mathbf{x} - \int_D \psi f \, d^3\mathbf{x} \tag{6}$$

Substituting (3), (4) into (6) and getting constants p_i, f_i, f_i^k outside the integration sign we have

[1] in what follows we omit arguments $(\mathbf{x})$ for brevity

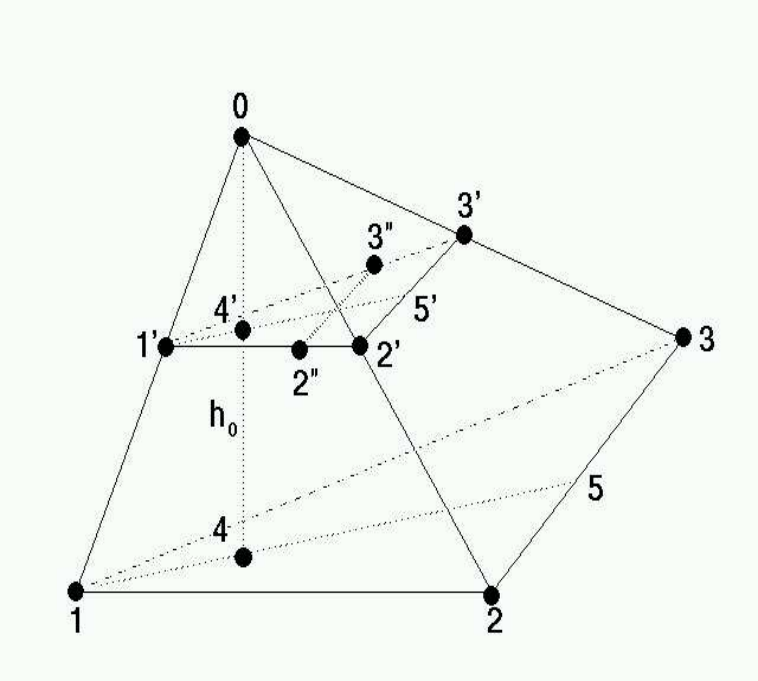

Figure 1: Integrating over a tetrahedral element.

$$\sum_{i=1}^{N} p_i \int_D (\nabla\psi \cdot \nabla\phi_i) d^3\mathbf{x} =$$

$$= \sum_{i=1}^{N} f_i^0 \oint_{\partial_0 D} \psi\,(\nabla\phi_i \cdot \mathbf{n})\, d^2\mathbf{x} + \sum_{i=1}^{N} f_i^1 \oint_{\partial_1 D} \psi\phi_i\, d^2\mathbf{x} - \sum_{i=1}^{N} f_i \int_D \psi\phi_i d^3\mathbf{x} \qquad (7)$$

A simple choice of test function is from the same function space as ϕ_i, namely $\psi \in C_N^0$. With this choice we have $\psi = \phi_j$ and (7) becomes

$$\sum_{j=1}^{N} c_{ij}^1 p_j = s_i \qquad (8)$$

$$s_i = \sum_{j=1}^{N} \left(b_{ij}^0 f_j^0 \mid_{\mathbf{x}\in\partial_0 D} + b_{ij}^1 f_j^1 \mid_{\mathbf{x}\in\partial_1 D} - c_{ij}^0 f_j \mid_{\mathbf{x}\in D} \right) \qquad (9)$$

where we introduced definitions

$$c^0_{ij} \equiv \int_D \phi_i \phi_j d^3\mathbf{x} \tag{10}$$

$$c^1_{ij} \equiv \int_D \nabla\phi_i \nabla\phi_j d^3\mathbf{x} \tag{11}$$

$$b^0_{ij} \equiv \int_{\partial_0 D} \phi_i (\nabla\phi_j \cdot \mathbf{n}) d^2\mathbf{x} \tag{12}$$

$$b^1_{ij} \equiv \int_{\partial_1 D} \phi_i \phi_j d^2\mathbf{x} \tag{13}$$

The coefficients c^n_{ij} above are also called the *stiffness matrix* coefficients in FEM literature. The influence of the boundaries is given by terms (12) and (13). Since the Dirichlet boundary conditions should keep the values at the boundary fixed, one should exclude from the set of test functions those which are non-zero at the Dirichlet boundary, i.e. $\phi_i \mid_{\partial_0} = 1$. This will make the first boundary integral (12) equal to zero. With regard to the second boundary integral (13), let's consider the case of zero-Neuman boundary conditions, i.e. $f^1 \equiv 0$. Then this integral can also be neglected[2], since its contribution will be zero by virtue of (9). After excluding the boundary integrals from relation (8), we have

$$\sum_{j=1}^{N} c^1_{ij} p_j = s_i \tag{14}$$

$$s_i = -\sum_{j=1}^{N} c^0_{ij} f_j$$

Since functions f, f^k are known, and all functions ϕ_i are well defined and linear, the integrals for the coefficients (10), (11) can be computed analytically, and then the problem (1) will be reduced to solving algebraic system of equations (14) for unknown p_i.

2.2 Deriving matrix coefficients

In this section we derive the analytical expressions for the matrix coefficients (10), (11). In contrast to a conventional approach [1, 2] we avoid

[2] for the case of non-zero Neuman boundary (2) see App.B

using Jacobian transformation matrices. Instead we rely on simple rules of geometrical differentiation, which results in rather compact expressions well suited for implementation in a computer code (see comments at the end of Sec.2.2.1).

By *grid* we understand a connected set of elements. By *element* we understand a connected set of vertexes. Each *vertex* of an element represents a *node* in the grid and is defined by it's coordinates $x_{i,\ i=1:N_d}$, where N_d is the dimensionality of the problem ($N_d = 3$ in our case).

Now consider the volume integrals (10), (11). The integration spans the whole domain, D. Since D is sub-divided into polyhedral elements we can re-write (10) and (11) as

$$c^0_{ij} = \sum_{k=1}^{N_e} \int_{E_k} \phi_i \phi_j d^3\mathbf{x} = \sum_{k=1}^{N_e} c^{0,k}_{\alpha_i \beta_j} \tag{15}$$

$$c^1_{ij} = \sum_{k=1}^{N_e} \int_{E_k} \nabla\phi_i \nabla\phi_j d^3\mathbf{x} = \sum_{k=1}^{N_e} c^{1,k}_{\alpha_i \beta_j} \tag{16}$$

where N_e is the number of grid-elements or cells, and the subindexes α, β are now indexes of element vertexes corresponding to grid nodes i, j inside each element. The coefficients $c^{*,k}_{\alpha\beta}$ are obtained by performing integration over each separate element.

$$c^{0,k}_{\alpha\beta} = \int_{E_k} \phi_\alpha \phi_\beta d^3\mathbf{x} \tag{17}$$

$$c^{1,k}_{\alpha\beta} = \int_{E_k} \nabla\phi_\alpha \nabla\phi_\beta d^3\mathbf{x} \tag{18}$$

2.2.1 Coefficients for tetrahedral elements

Let's perform the integration of coefficients (17), (18) for tetrahedral elements. Here we shall drop for convenience the super-index k, implying that the integration is performed within a single element. We shall also compute only coefficients c^0_{01}, c^1_{01} betweem element $\alpha = 0$ and $\beta = 1$. The result will be valid for any other coefficients α, β. For a tetrahedral element (Fig.1) the integration can be done as follows[3]

[3] We skip index k for brevity

$$c_{00}^0 = \int_0^{h_0} A_{1'2'3'}\phi_0^2(\xi)dh =$$

$$= h_0 \int_0^1 A_{1'2'3'}\phi_0^2(\xi)d\xi = h_0 A_{123}\int_0^1 \xi^2(1-\xi)^2 d\xi \tag{19}$$

where $h = h_0\xi$, $A_{1'2'3'} = \xi A_{123}$ and $h_0 = h_{04}$ (see Fig.1).

$$\begin{aligned} c_{01}^0 &= h_0 \int_0^1 \phi_0(\xi) h_{1'5'} \int_0^1 \phi_1(1')(1-\eta) l_{2''3''} d\eta d\xi \\ &= h_0 \int_0^1 \phi_0(\xi) h_{1'5'} l_{2'3'} \phi_1(1') \int_0^1 \eta(1-\eta) d\eta d\xi \\ &= h_0 h_{15} l_{23} \int_0^1 \xi^2 \phi_0(\xi)\phi_1(1') \int_0^1 \eta(1-\eta) d\eta d\xi \end{aligned}$$

Now, from the linearity of ϕ_i we have $\phi_0(\xi) = 1-\xi$, $\phi_1(1') = \xi$, and consequently

$$\begin{aligned} c_{01}^0 &= h_0 h_{15} l_{23} \int_0^1 \xi^3(1-\xi) \int_0^1 \eta(1-\eta) d\eta d\xi \\ &= 2h_0 A_{123} \int_0^1 \xi^3(1-\xi) d\xi \int_0^1 \eta(1-\eta) d\eta \end{aligned} \tag{20}$$

$$c_{02}^0 = c_{03}^0 = c_{01}^0 \tag{21}$$

Which for any two local indexes α, β will become

$$c_{\alpha\alpha}^0 = h_\alpha A_\alpha \int_0^1 \xi^2(1-\xi)^2 d\xi = \frac{1}{30} h_\alpha A_\alpha = \frac{1}{10} V \tag{22}$$

$$c_{\alpha\beta}^0 = 2h_\alpha A_\alpha \int_0^1 \xi^3(1-\xi) d\xi \int_0^1 \eta(1-\eta) d\eta = \frac{1}{60} h_\alpha A_\alpha = \frac{1}{20} V \tag{23}$$

where h_α is the height vector of the tetrahedron dropped from node α onto the face A_α facing this node. As can be seen, $c_{\alpha\beta}$ does not depend on β, that is, for each vertex α it is the same for all three vertexes on the face facing node α.

Now, let's consider the coefficients $c_{\alpha\beta}^1$. The derivatives of ϕ_i at node $i = 0$ can be computed, considering that $|\nabla\phi_0| = 1/h_0$, and is directed along

segment $\{40\}$. Generally, $|\nabla\phi_\alpha| = 1/h_\alpha$, and is directed along the vector normal to the plane opposite to node i. Thus, we have

$$c^1_{00} = h_0 \int_0^1 A_{1'2'3'} |\nabla\phi_0(\xi)|^2 d\xi =$$
$$h_0 A_{123} \int_0^1 \xi^2 \frac{1}{h_0^2} d\xi = \frac{A_{123}}{h_0} \int_0^1 \xi^2 d\xi = \frac{1}{3}\frac{A_{123}}{h_0} \tag{24}$$

$$\begin{aligned} c^1_{01} &= h_0 \int_0^1 h_{1'5'} \int_0^1 l_{2"3"} (\nabla\phi_0 \cdot \nabla\phi_1) d\eta d\xi \\ &= h_0 h_{15} \int_0^1 \xi l_{2'3'} \int_0^1 \eta (\nabla\phi_0 \cdot \nabla\phi_1) d\eta d\xi \\ &= h_0 h_{15} l_{23} \int_0^1 \xi^2 \int_0^1 \eta \frac{(\mathbf{n}_0 \cdot \mathbf{n}_1)}{h_0 h_1} d\eta d\xi \\ = \frac{2h_0 A_{123}}{h_0 h_1} (\mathbf{n}_0 \cdot \mathbf{n}_1) &\int_0^1 \xi^2 d\xi \int_0^1 \eta d\eta = \frac{1}{3}\frac{A_{123}}{h_1} (\mathbf{n}_0 \cdot \mathbf{n}_1) \end{aligned} \tag{25}$$

And generally,

$$c^1_{\alpha\beta} = \frac{1}{3}\frac{A_\alpha}{h_\beta} (\mathbf{n}_\alpha \cdot \mathbf{n}_\beta) = \frac{1}{9}\frac{A_\alpha A_\beta}{V} (\mathbf{n}_\alpha \cdot \mathbf{n}_\beta) = \frac{(\mathbf{a}_\alpha \cdot \mathbf{a}_\beta)}{9V} \tag{26}$$

Now let's reinstall the k-indexes that we dropped earlier for convenience, and use the results (22), (23) and (26) to assemble the stiffness matrix coefficients (15) and (16). As can be seen from (22), (23) and (26) the only information required for the computations are the volumes of the cells[4] and vector areas of the faces. This makes the expressions very suitable for implementation in a computer code, since the vector areas and volumes are usually computed within the solver, and the computation of the coefficients according to (22), (23) and (26) involves at worst just a simple scalar product operation (26).

2.2.2 Coefficients for hexahedral elements

For hexahedral elements each basis function $\phi(\mathbf{x})$ has 7 overlapping basis functions corresponding to the neighboring nodes (Fig.2). Since the

[4] see Appendix A.1 on how to compute the volumes

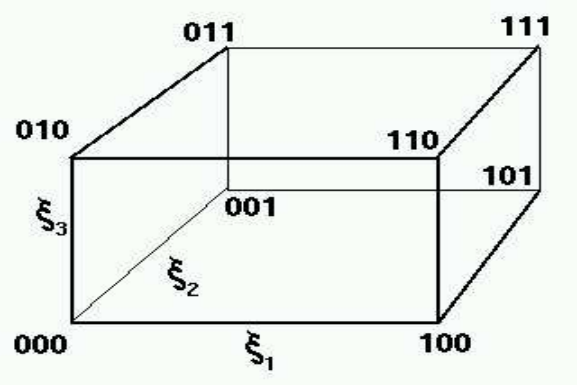

Figure 2: Integrating over a hexahedral element.

assembly operation is performed locally to each element, we shall confine the further discussion to one element, and denote all functions corresponding to the vertices of that element as $\phi_{\alpha_1,\alpha_2,\alpha_3}$, where $\alpha_i = 0,1$, thus giving eight functions $\phi_{000}, \phi_{001}, ..\phi_{111}$. Now we shall compute the integrals (17), (18), and as in the Sec.2.2.1 we shall drop index k, and without loss of generality we can consider the only matrix coefficients related to function ϕ_{000}: $c_{000;000}, c_{000;001}, ..., c_{000;111}$. Let's select a local coordinate system ξ_i with the origin at node 000.

As a first step a general hexahedral element is transformed into a perfect cube with edges of length 1, using the appropriate scaling operations and orthogonal rotations:

$$\eta_i = \frac{x_i - x_i^0}{\Delta x_i} \tag{27}$$

$$\xi_i = J_{ij}\eta_j \tag{28}$$

where $x_i = \{x_1, x_2, x_3\}$, and $\xi_i = \{\xi_1, \xi_2, \xi_3\}$. Using a compact tensor notation each basis function can be written as

$$\phi_{\alpha_1\alpha_2\alpha_3}(\{\xi_i\}) = \prod_i^3 (\alpha_i\xi_i + (1-\alpha_i)(1-\xi_i)) \tag{29}$$

and the non-dimensional matrix elements $\tilde{c}^{0,1}_{000;\alpha_1\alpha_2\alpha_3}$, (see (17), (18)), can be computed as follows:

$$\tilde{c}^0_{000;\alpha_1\alpha_2\alpha_3} = \int \phi_{000}(\vec{\xi})\phi_{\alpha_1\alpha_2\alpha_3}(\vec{\xi})d\xi_1\, d\xi_2\, d\xi_3 = \tag{30}$$

$$= \int \prod_i^3 (1-\xi_i) \prod_j^3 (\alpha_j\xi_j + (1-\alpha_j)(1-\xi_j))d^3\xi \tag{31}$$

$$\tilde{c}^1_{000;\alpha_1\alpha_2\alpha_3} = \int (\phi_{000}(\vec{\xi}))_{,k}(\phi_{\alpha_1\alpha_2\alpha_3}(\vec{\xi}))_{,k}d^3\xi = \tag{32}$$

$$= \int \sum_k^3 \left(\prod_i^3 (1-\xi_i)\right)_{,k} \left(\prod_j^3 (\alpha_j\xi_j + (1-\alpha_j)(1-\xi_j))\right)_{,k} d^3\xi \tag{33}$$

where α_i coefficients are either 0 or 1: $\{\alpha_i\} = 0,1$, and we use a tilde to signify that the coefficients are computed in a non-dimensional coordinate system. The eight coefficients are:

$$\tilde{c}^0_{000;000} = \frac{1}{27}, \tilde{c}^0_{000;111} = \frac{1}{216}$$
$$\tilde{c}^0_{000;001} = \tilde{c}^0_{000;010} = \tilde{c}^0_{000;100} = \frac{1}{54}$$
$$\tilde{c}^0_{000;011} = \tilde{c}^0_{000;110} = \tilde{c}^0_{000;101} = \frac{1}{108}$$
$$\tilde{c}^1_{000;000} = \frac{1}{3}$$
$$\tilde{c}^1_{000;001} = \tilde{c}^1_{000;010} = \tilde{c}^1_{000;100} = 0$$
$$\tilde{c}^1_{000;011} = \tilde{c}^1_{000;110} = \tilde{c}^1_{000;101} = \tilde{c}^1_{000;111} = \frac{-1}{12}$$

which corresponds to the eight corners of the cube shown in Fig.2. The procedure for computing the coefficients is given in App.A.4. For a uniform Cartesian grid with equal grid spacing in all three directions the dimensional matrix elements will be related to the non-dimensional ones by the following relations

$$c^0_{ij} = V\tilde{c}^0_{ij}$$
$$c^1_{ij} = V^{1/3}\tilde{c}^1_{ij} \tag{34}$$

where V is the volume of the cell: $V = dxdydz = (dx)^3$, and we returned back from cell-local indexes $\alpha_i, \beta_i = 0, 1$ to grid-global indexes: i, j. Relation (34) is valid for a general rectangular grid. For non-uniform and non-orthogonal grids a more complex matrix transformation will be necessary with the coefficients determined by the jacobian of transformation (28).

2.3 Iterative scheme

Since the basis functions ϕ_i are non-zero only for the elements that include node i as one of their vertexes, we can restrict the summation in (15) (16) to those elements E_k that include both nodes i, j, since all other elements will give a zero contribution to the integral. This can be done by looping through all elements and computing contributions to those c_{ij} that have both nodes (i, j) belonging to that element, which is called the *assembly* procedure in FEM. In addition to this one does not have to store large two dimensional arrays of size $N \times N$ for c_{ij} since they are very sparse. As an illustration consider the solution of the equation system (14) using a Jacobi iteration scheme.

$$p_i^{n+1} = -\frac{1}{c_{ii}^1}\left(\sum_{j=1}^{N} c_{ij}^0 f_j + \sum_{j\neq i}^{N} c_{ij}^1 p_j^n\right) = -\frac{A_i + C_i^n}{c_{ii}^1} \tag{35}$$

$$A_i = \sum_{j=1}^{N} c_{ij}^0 f_j \tag{36}$$

$$C_i^n = \sum_{j\neq i}^{N} c_{ij}^1 p_j^n \tag{37}$$

The iteration scheme (35) can be realized as follows. First compute coefficients $c_{ii}^{1,k}$ in (35) using (16) and (26) as in Figure 3[5]. Then assembly the A_i coefficients, representing source terms due to f in (1) in a loop over all cells (Fig.4).

Coefficients C_i^n can be repeatedly assembled together with upgrading variable values of p in (1) in two subsequent loops (Figs.5,6).

Loops (5) (6) can be included into another iteration loop (38) to guarantee convergence:

[5] We use the C-language convention to start indexes from 0

```
for (ic=0; ic<Nc; ic++)
{  //loop over the cells
   int   nv=Nv[ic];  //number of vertexes
   float vol=Vol[ic];//cell-volume
   for (iv=0; iv<nv; iv++)
   {  //loop over the vertices
      inode=NodeIndex(ic,iv);
      a=Area(ic,iv);
      C1ii[inode]+=a[inode]*a[inode]/(9.*vol);
   }
}
```

Figure 3: Assembly of c^1_{ii}.

$$
\begin{aligned}
&while(|P^{n+1} - P^n| > \varepsilon) \\
&\quad Assemble\, C_i\ (Fig.5) \\
&\quad Update\, P_i\ (Fig.6)
\end{aligned}
\tag{38}
$$

The difference in handling of Dirichlet vs zero-Neuman boundaries is in that there is no looping over the boundary nodes for the Dirichlet boundary in (6), whereas all the boundary nodes are included for the Neuman boundary.

2.4 Poissson equation in a gradient-source form

Consider a modified boundary value problem (1):

$$\Delta p(\mathbf{x}) = \nabla \cdot \mathbf{g}(\mathbf{x})$$

$$p(\mathbf{x})\,|_{\mathbf{x}\in\partial_0 D} = f^0(\mathbf{x}) \tag{39}$$

$$\frac{\partial p(\mathbf{x})}{\partial \mathbf{n}}|_{\mathbf{x}\in\partial_1 D} \equiv (\nabla p \cdot \mathbf{n})_{\mathbf{x}\in\partial_1 D} = f^1(\mathbf{x}) \tag{40}$$

```
A[*]=0;
for (ic=0; ic<Nc; ic++)
{  //loop over the cells
   int
      nv=Nv[ic]; //number of vertexes for cell ic
   float
      c0ii=0.10*Vol[ic],//compute c-coeffs
      c0ij=0.05*Vol[ic];
   for (iv=0; iv<nv; iv++)
   {  //loop over all the cell-vertexes
      inode=NodeIndex[ic,iv]; //global index
      A[inode]+=c0ii*f(inode);
      for (jv=0; jv<nv-1; jv++)
      {  //cell vertexes connected to vertex iv
         jnode=NodeIndex[ic,(iv+jv+1)%nv];
         A[jnode]+=c0ij*f(jnode);
      }
   }
}
```

Figure 4: Assembly of A_i coefficients.

where the source term has a gradient form $\nabla \cdot \mathbf{g}$ with $\mathbf{g}$ a given vector field. This form is encountered in many applications, like in the equation for electric potential field or for a pressure field in in Navier Stokes equation. As far as the previous derivation is conserned this will affect only the last term in (6), which now becomes

$$\int_D (\nabla\psi \cdot \nabla p)\, d^3\mathbf{x} = -\int_D \psi \nabla \cdot \mathbf{g}\, d^3\mathbf{x} \tag{41}$$

Here we neglected the boundary integrals, on the grounds given in Sec. 2.1. Discretizng it further, analogously to (7), we have

$$\sum_{i=1}^{N} p_i \int_D (\nabla\psi \cdot \nabla\phi_i) d^3\mathbf{x} = -\sum_{i=1}^{N} (\mathbf{g}_i \cdot \int_D \psi \nabla\phi_i d^3\mathbf{x}) \tag{42}$$

Renaming indexes and choosing $\psi = \phi_j$ we can reduce the above to the form of (14), namely

```
C[*]=0;
for (ie=0; ie<Ne; ie++)
{  //loop over the cells
   int   nv=Nv[ie];  //number of vertexes
   float vol=Vol[ie];//cell-volume
   for (i=0; i<nv; i++)
   {  //loop over all the cell-vertexes
      inode=NodeIndex[ie,i]; //global index
      p=P[inode];//variable value at vertex i
      for (j=0; j<nv-1; j++)
      {  //cell vertexes connected to vertex i
        jnode=NodeIndex[ie,(i+j+1)%nv];
        c1ij=SCLP(a[i],a[j])/(9*vol);
        C[jnode]+=c1ij*p;
      }
   }
}
```

Figure 5: Assembly of C_i coefficients.

```
for (i=0; i<Nn; i++)
   P[i]=(A[i]-B[i])/C1ii[i];
```

Figure 6: Updating all the node values for p.

$$\sum_{j=1}^{N} c_{ij}^{1} p_j = s_i \tag{43}$$

$$s_i = -\sum_{j=1}^{N} \mathbf{c}_{ij} \cdot \mathbf{g}_j$$

where the source term can be explicitly written as

$$s_i = -\sum_{j=1}^{N} c_{\alpha ij} g_{\alpha j} \tag{44}$$

$$c_{\alpha ij} = \int_D \phi_{i,\alpha}\phi_j d^3\mathbf{x} \tag{45}$$

where $\alpha = 1,2,3$ designates the number of a Cartesian vector component.

2.4.1 Deriving coefficients $c_{\alpha ij}$

This time instead of coefficients c^0_{ij} defined in (15) we have to use

$$c_{\alpha ij} = \sum_{k=1}^{N_e} \int_{E_k} \phi_{i,\alpha}\phi_j d^3\mathbf{x} = \sum_{k=1}^{N_e} c^{0,k}_{\alpha ij} \tag{46}$$

which should be integrated on each tetrahedral element (Fig.1). Since ϕ_i is a linear function, the derivative $\phi_{i,\alpha}$ is a constant vector directed from node i to the opposite face and equal to

$$\phi_{i,\alpha} = \frac{h^{(i)}_\alpha}{h^{(i)}_\beta h^{(i)}_\beta} = \frac{h^{(i)}_\alpha}{h^2_{(i)}} \tag{47}$$

where $h^{(i)}_\alpha$ is a face-normal vector, connecting node i and the opposite face[6], and $h_{(i)} = h^{(i)}_\beta h^{(i)}_\beta$. Now (46), considering that ϕ can be parametrized as $\phi_i(\xi) = \xi$, where ξ is 1 at the node i and 0 at the opposite face, and $\phi_i(\xi)$ changes linerarly between the two, then (46) can be easily integrated

$$c_{\alpha ij} = \frac{h^{(i)}_\alpha}{h^2_{(i)}} h_{(j)} A_{(j)} \int_0^1 (1-\xi)\phi_j(\xi) d\xi =$$

$$\frac{h^{(i)}_\alpha}{h^2_{(i)}} h_{(j)} A_{(}j) \int_0^1 (1-\xi)\xi d\xi = \frac{h^{(i)}_\alpha}{h^2_{(i)}} h_{(j)} A_{(j)} \left[\frac{\xi^2}{2} - \frac{\xi^3}{3}\right]_0^1 \tag{48}$$

$$= \frac{h^{(i)}_\alpha}{h^2_{(i)}} \frac{1}{6} 3V_{(j)} = \frac{1}{2}\frac{h^{(i)}_\alpha}{h^2_{(i)}} V_{(j)} \tag{49}$$

Where $V_{(j)}$ is the volume of a cell, which is the same for all $j's$: $V_{(j)} = V$.

[6] there is no summation over indexes in parenthesis

3 A 3-D Cell-Centered Discretization Scheme for the Poisson Equation

In this section we consider a finite-volume discretization scheme used to solve cell-centered scalar field variables on a tetrahedral grid.

3.1 Method

Consider a boundary value problem (1):

Let's consider a cell-centered location of grid variables for p on a general non-structured grid in n-dimensions. The Laplace operator, Δ, can be represented as[7]

$$\begin{aligned} \Delta p = \nabla \cdot \nabla p = \frac{\partial^2 p}{\partial x_i^2} \\ = \frac{\partial}{\partial x_i}\left(\frac{\partial p}{\partial x_i}\right) = (p_{,i})_{,i} \end{aligned} \tag{50}$$

or

$$\Delta p = p_{,ii} \tag{51}$$

For a smooth function $\phi \in C_2$ the Gauss-theorem states

$$\int_C \phi_{,i} dV = \int_{\partial C} \phi \, da_i \tag{52}$$

where dV is the volume element and da_i is the i-th Cartesian component of the outward normal surface area-element vector $d\vec{a}$ of the boundary surface ∂C of the cell. Using (52), we can approximate the gradient $\phi_{,i}$ as

$$\phi_{,i} \approx \frac{1}{V} \int_{\partial C} \phi \, da_i$$

where $V = \int_C dV$ is the volume of the grid cell. For a simple polygonal cell the latter reduces to

[7] We use the definition $f_{,i} \equiv \frac{\partial f}{\partial x_i}$ and repeated indexes imply summation.

$$\phi_{,i} \approx \frac{1}{V}\phi^k a_i^k \tag{53}$$

where index $k = 1 : N_f$ spans all the faces of the element and we use our summation rule. The value of ϕ^k should be evaluated at the k-th face between the current cell and its k-th neighbor.

Using $p_{,i}$ instead of ϕ in (50) and using (53) we have

$$\Delta p \equiv p_{,ii} = \frac{1}{V} p_{,i}^k a_i^k \tag{54}$$

In this expression $p_{,i}^k$ represents the $i-th$ component of gradient of p evaluated at k-th face. A natural approximation for (54) in a cell-centered scheme is

$$p_{,i}^k a_i^k \approx \frac{p^k - p}{d^k} n_i^k a_i^k \tag{55}$$

where p is the value of the variable at the current cell-center, p^k is it's value at the center of the neighbor cell k, d^k is the distance between the current and the neighbor cell centers, n_i^k is the face outward normal unit vector:

$$d^k = [\sum_i (x_i^k - x_i)^2]^{1/2} \tag{56}$$

$$n_i^k = \frac{a_i^k}{a^k} \tag{57}$$

Substituting (57) into (55) and using the relation for the face area: a^k of the k-th face:

$$a^k = (\sum_i a_i^k a_i^k)^{1/2} \tag{58}$$

we have

$$p_{,i}^k a_i^k \approx \frac{p^k - p}{d^k} a^k \tag{59}$$

Substituting (59) and (54) into (1) we have

$$\frac{a^k}{V} \cdot \frac{p^k - p}{d^k} = f \tag{60}$$

```
for (c=1; c<Nc; c++)
{ //Looping over all Nc cells
  A=0; B=0; V=Volume(c);
  for (b=1; b<Nb(c); b++)
  { //Looping over neighbor-cells
    d=Distance(b,c);
    a=FaceArea(Face(b,c))/d;
    A+=a*P[b];
    B+=a;
  }
  P[c]=(A-V*f(c))/B;
}
```

After rearranging the terms we have

$$A - Bp = f \tag{61}$$

$$A = \frac{1}{V}\sum_k \frac{a^k p^k}{d^k} \tag{62}$$

$$B = \frac{1}{V}\sum_k \frac{a^k}{d^k} \tag{63}$$

Equation (61) is applied to a single cell C. It can be used to compute (1) in a simple iterative scheme (Jacobi/Gauss-Seidel):

$$p_{n+1} = \frac{A(p_n) - f}{B(p_n)} \tag{64}$$

which can be realized in repeated looping over cells like this

References

[1] P.M. Gresho and R.L. Sani. *Incompressible Flow and the Finite Element Method: Isothermal Laminar Flow*, volume 2. John Wiley & Sons, Ltd., New York, 2000.

[2] J.N. Reddy and D.K. Gartling. *The Finite Element Method in Heat Transfer and Fluid Dynamcs*. CRC Press, 1996.

[3] Stephen Wolfram. *The Mathematica Book. Third Edition*. Cambridge University Press, 1996.

A Some relations

A.1 Volume of a polyhedron

Volume V of a solid body C can be computed as

$$V = \int_C dV \tag{65}$$

Using relation

$$x_{j,j} = 3 \tag{66}$$

we have

$$V = \frac{1}{3}\int_C x_{j,j} dV \tag{67}$$

And using the Gauss theorem we get

$$V = \frac{1}{3}\int_{\partial C} x_j \, da_j \tag{68}$$

For a polyhedron with N faces we have

$$V = \frac{1}{3}\sum_{k=1}^{N} x_j^k a_j^k = \frac{1}{3}\sum_{k=1}^{N} x_j^k n_j^k a^k \tag{69}$$

where x_j^k are the center-coordinates of face k, and n_j^k, a^k are the coordinates of the outward normal unit vector and the area of face k respectively.

A.2 Center of mass of a polyhedron: Method 1

Center of mass r_i of a solid body C can be computed as

$$r_i = \frac{1}{V}\int_C x_i\, dV$$

Using relation 66, we have

$$r_i = \frac{1}{3V}\int_C x_i x_{j,j} dV$$

Using

$$(x_i x_j)_{,j} = x_{i,j}x_j + x_i x_{j,j} = \delta_{ij}x_j + x_i x_{j,j}$$

we have

$$r_i = \frac{1}{3V}\left[\int_C (x_i x_j)_{,j} dV - \int_C x_i\, dV\right]$$

$$r_i = \frac{1}{3V}\int_C (x_i x_j)_{,j} dV - \frac{1}{3}r_i$$

Therefore,

$$r_i = \frac{1}{4V}\int_C (x_i x_j)_{,j} dV$$

And using Gauss theorem, we have

$$r_i = \frac{1}{4V}\int_{\partial C} x_i x_j\, da_j$$

For an polyhedron with N faces this becomes

$$r_i = \frac{1}{4V}\sum_k^N x_i^k x_j^k a_j^k$$

A.3 Center of mass of a polyhedron: Method 2

Center of mass r_i of a solid body C can be computed as

$$r_i = \frac{1}{V}\int_C x_i dV$$

Using relation 66 we have

$$(x_j x_j)_{,i} = 2x_{j,i}x_j = 2x_i$$

and for r_i:

$$r_i = \frac{1}{2V}\int_C (x_j x_j)_{,i} dV$$

By Gauss theorem:

$$r_i = \frac{1}{2V}\int_C (x_j x_j)_{,i} dV$$

$$= \frac{1}{2V}\int_{\partial C} x_j x_j da_i$$

For an polyhedron with N faces this becomes

$$r_i = \frac{1}{2V}\sum_k^N x_j^k x_j^k a_i^k$$

A.4 Coefficients for hexahedral elements.

The numerical values of the coefficients for hexahedral grid (34) were computed using the following Mathematica procedure [3]

```
Do[Print[i,j,k];
P0=(1-x)*(1-y)*(1-z);
P1=(i*x+(1-i)*(1-x))*(j*y+(1-j)*(1-y))*(k*z+(1-k)*(1-z));
C0=Integrate[P0*P1,{x,0,1},{y,0,1},{z,0,1}];
Print[C0];
DP0[1]=D[P0,x];
DP0[2]=D[P0,y];
```

```
DP0[3]=D[P0,z];
DP1[1]=D[P1,x];
DP1[2]=D[P1,y];
DP1[3]=D[P1,z];
DP=Sum[DP0[n]*DP1[n],{n,1,3}];
C1=Integrate[DP,{x,0,1},{y,0,1},{z,0,1}];
Print[C1]
,{i,0,1},{j,0,1},{k,0,1}
];
```

B General Neuman boundary conditions

In the case of non-zero gradient Neuman boundary conditions the second boundary integral (13) can no longer be neglected. Integration of (13) can be reduced to the sum of integrals over triangular cells (Fig.7). Although the integration spans the whole Neuman boundary $\partial_1 D$, only the nodes sharing a common element (triangle) will give contribution to the integral. Therefore the integral can be split into the sum over the elements as follows.

$$b_{ij} = \sum_{k \in \partial_1 D} \int_{B_k} \phi_i \phi_j d^2\mathbf{x} = \sum_{k \in \partial_1 D} b_{ij}^k \tag{70}$$

where B_k is a two-dimensional boundary element, which is a face of 3D element E_k lying at the boundary $\partial_1 D$. Integration of (70) produces (Fig.7) the following result

$$b_{ii}^k = \int_{B_k} \phi_i \phi_j d^2\mathbf{x} = \int_0^{h_{00'}} \phi_i^2(h) l_{1'2'} dh$$
$$= h_{00'} \int_0^1 \phi_i^2(\alpha) \alpha l_{12} d\alpha = h_{00'} l_{12} \int_0^1 \phi_i^2(\alpha) \alpha d\alpha = 2A \int_0^1 \phi_i^2(\alpha) \alpha d\alpha \tag{71}$$

where A is the area of the triangle (012). Using the linear profile of a shape function in the given element $\phi_i(\alpha) = 1 - \alpha$, we have

$$b_{ii}^k = 2A \int_0^1 (1-\alpha)^2 \alpha d\alpha = \frac{A}{6} \tag{72}$$

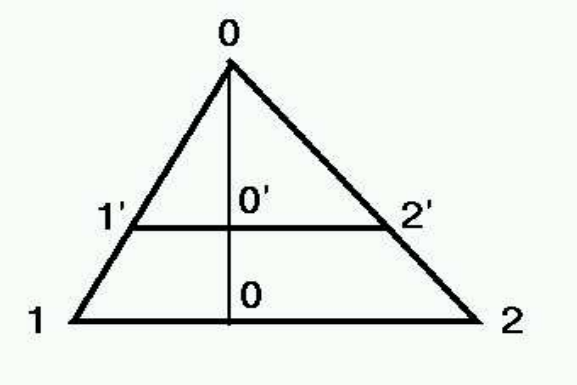

Figure 7: Integrating over a triangular element.

Analogously,

$$b_{ij}^k = 2A\int_0^1 \phi_i(\alpha)\int_0^{1-\alpha}\phi_j(\beta)d\alpha d\beta$$
$$= 2A\int_0^1 (1-\alpha)\int_0^{1-\alpha}\beta d\beta d\alpha = \frac{A}{4} \tag{73}$$

Now we can use these coefficients to solve (8) using Jacobi iteration scheme.

$$p_i^{n+1} = \frac{1}{c_{ii}^1}\left(\sum_{j\in\partial_1 D} b_{ij}f_j^1 - \sum_{j=1}^{N_n} c_{ij}^0 f_j - \sum_{j\neq i}^{N_n} c_{ij}^1 p_j^n\right)$$
$$= \frac{1}{c_{ii}^1}(-A_i + B_i - C_i^n) \tag{74}$$
$$A_i = \sum_{j=1}^{N_n} c_{ij}^0 f_j \tag{75}$$
$$B_i = \sum_{j\in\partial_1 D} b_{ij}f_j^1 \tag{76}$$
$$C_i^n = \sum_{j\neq i}^{N_n} c_{ij}^1 p_j^n \tag{77}$$

where the boundary sum B_i in (76) is non-zero only when element i has at least one face belonging to the boundary.

The iterative scheme will now include an additional loop over boundary cells to assemble B_i coefficients, representing boundary source-terms due to f^1 (Fig.8). The assembly of B coefficients in Fig.8 can be done more efficiently if the elements are sorted so that the boundary elements occupy first N_b places in the global elements array.

```
B[*]=0;
for (iface=0; iface<Nbf; iface++)
{  //loop over the boundary faces
   int
      nv=Nfv[iface]; //number of vertexes for iface
   for (iv=0; iv<nv; iv++)
   {  //loop over all the face-vertexes
      float
        bii=area[iface,iv]/6,//compute c-coeffs
        bij=area[iface,iv]/4;
      inode=NodeIndex[iface,iv]; //global index
      B[inode]+=bii*f1(inode);
      for (jv=0; jv<nv-1; jv++)
      {  //cell vertexes connected to vertex iv
         jnode=NodeIndex[ic,(iv+jv+1)%nv];
         B[jnode]+=bij*f1(jnode);
      }
   }
}
```

Figure 8: Assembly of B_i coefficients.