Numerical Modelling in Geosciences

Lecture 5

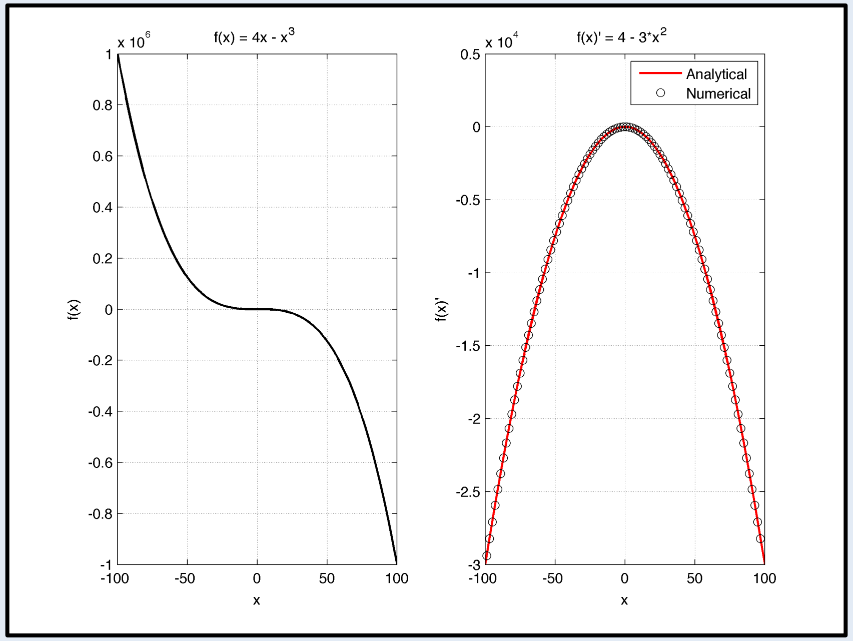
Numerical solution of PDEs

Finite difference method

First derivative in space of function f(x) (1D case):

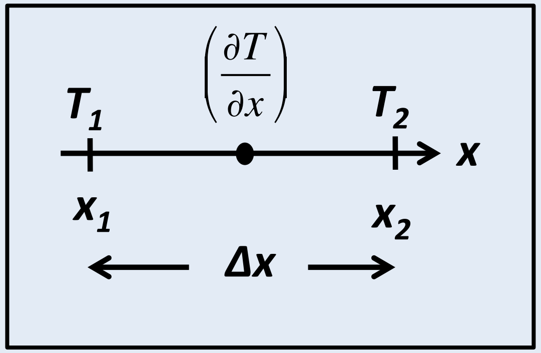
$$f'(x) = \frac{\partial f(x)}{\partial x} \approx \lim_{\Delta x \to 0} \frac{f(x+\Delta x) - f(x)}{(x+\Delta x) - (x)}$$

Analytical vs Numerical



Example:

$$\frac{\partial T}{\partial x} \approx \frac{\Delta T}{\Delta x} = \frac{T_2 - T_1}{x_2 - x_1}$$



The same is valid for first derivative in time:

$$\frac{\partial T}{\partial t} \approx \frac{\Delta T}{\Delta t} = \frac{T^{t+\Delta t} - T^{t}}{\Delta t}$$

How to apply finite differences

1) Define an Eulerian grid of nodal points

2) Assign input field variables to the nodes

3) Apply PDEs and boundary conditions to each nodal point

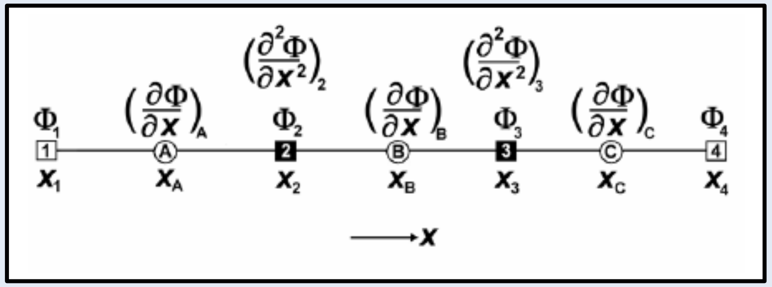
4) Solve the system of equations to get output field variables

1) Define an Eulerian grid of nodal point

The grid can be uniform (equal distance between nodal points) or non-uniform, but points need to be orthogonal. Lines connecting points are fictitious.

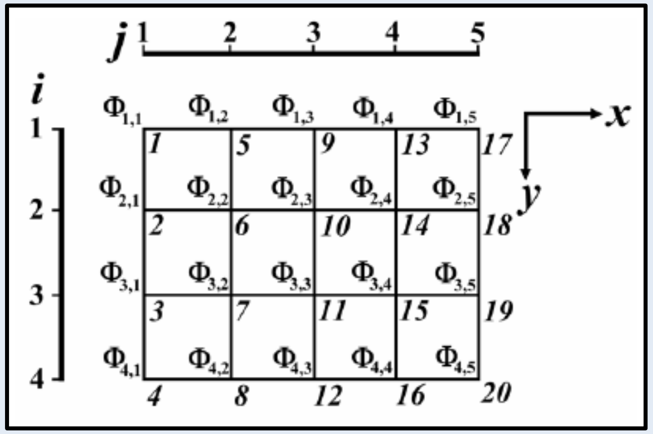
With this process we subdivide the continuous medium into discrete parts.

1D grid



Φ_1
$\left(\frac{\partial\Phi}{\partial x}\right)_A$
$\left(\frac{\partial^2\Phi}{\partial x^2}\right)_2$
Φ_2
$\left(\frac{\partial\Phi}{\partial x}\right)_B$
$\left(\frac{\partial^2\Phi}{\partial x^2}\right)_3$
Φ_3
$\left(\frac{\partial\Phi}{\partial x}\right)_C$
Φ_4
1 A 2 B 3 C 4
x_1 x_A x_2 x_B x_3 x_C x_4
x

2D grid



j 1 2 3 4 5
i 1 2 3 4
$\Phi_{1,1}$ $\Phi_{1,2}$ $\Phi_{1,3}$ $\Phi_{1,4}$ $\Phi_{1,5}$
$\Phi_{2,1}$ $\Phi_{2,2}$ $\Phi_{2,3}$ $\Phi_{2,4}$ $\Phi_{2,5}$
$\Phi_{3,1}$ $\Phi_{3,2}$ $\Phi_{3,3}$ $\Phi_{3,4}$ $\Phi_{3,5}$
$\Phi_{4,1}$ $\Phi_{4,2}$ $\Phi_{4,3}$ $\Phi_{4,4}$ $\Phi_{4,5}$
1 5 9 13 17
2 6 10 14 18
3 7 11 15 19
4 8 12 16 20
x
y

2D geometrical indexing:

Phi(3,4)

2D global indexing:
gi = i + N_i *(j-1)

Phi(15)

2) Assign input field variables to the nodes

In the case of Poisson's equation:

$$\Delta\Phi = \frac{\partial^2\Phi}{\partial x^2} + \frac{\partial^2\Phi}{\partial y^2} + \frac{\partial^2\Phi}{\partial z^2} = f$$

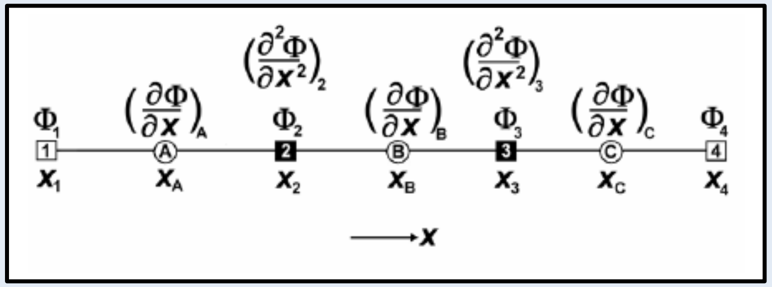
we need to assign function f to each node.

3a) Apply PDEs and boundary conditions to each nodal point.

In the case of 1D Poisson's equation:

$$\Delta\Phi = \frac{\partial^2\Phi}{\partial x^2} = f$$

1D grid



$\left(\frac{\partial^2\Phi}{\partial x^2}\right)_2$
$\left(\frac{\partial^2\Phi}{\partial x^2}\right)_3$
Φ_1
$\left(\frac{\partial\Phi}{\partial x}\right)_A$
Φ_2
$\left(\frac{\partial\Phi}{\partial x}\right)_B$
Φ_3
$\left(\frac{\partial\Phi}{\partial x}\right)_C$
Φ_4
1
A
2
B
3
C
4
x_1
x_A
x_2
x_B
x_3
x_C
x_4
x

Nodes **1** and **4**: basic boundary nodes where boundary conditions are assigned

Nodes **2** and **3**: basic internal nodes where Poisson equation is applied, and field variable Phi and its second derivative are defined.

Nodes **A**, **B** and **C**: additional nodes where first derivative of field variable Phi is defined

The number of indipendent linear equations must be equal to the number of unknowns:

# equations = # grid points * # of field variables = 4

3a) Apply PDEs and boundary conditions to each nodal point.

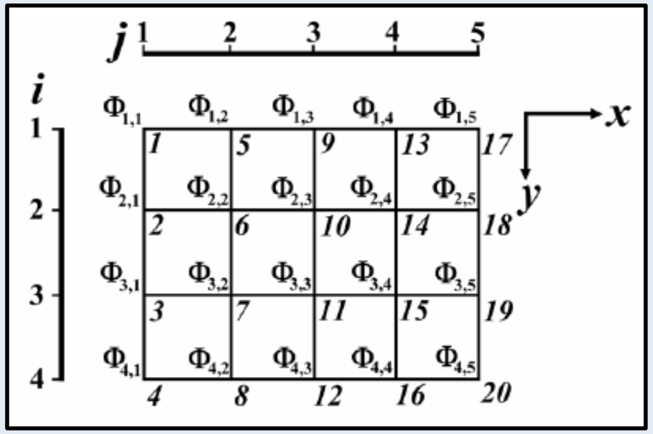
In the case of 2D Poisson's equation:

$$\Delta\Phi = \frac{\partial^2\Phi}{\partial x^2} + \frac{\partial^2\Phi}{\partial y^2} = f$$

Nodes (**1,j; 4,j; i,1; i,5**): basic boundary nodes where boundary conditions are assigned

All the others are basic internal nodes where Poisson equation is applied, and the field variable Phi and its second derivative are defined.

2D grid



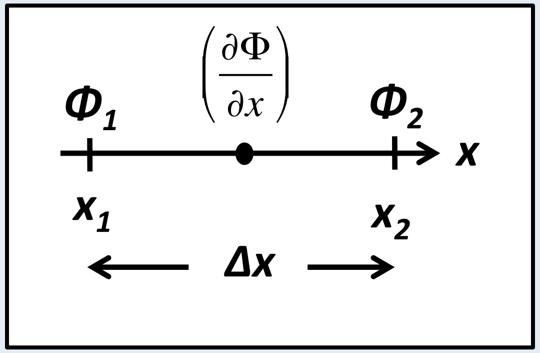
2D global indexing:
gi = i + N_i *(j-1)

# equations = # grid points * # of field variables = 20

Boundary conditions

Dirichlet BC: specify the value of the solution on the boundary nodes

Example: $\Phi_1 = 0$



Neumann BC: specify the value of the derivative of the solution on the boundary nodes

Example: $\frac{\partial \Phi}{\partial x} \approx \frac{\Delta \Phi}{\Delta x} = 4.2 \Rightarrow \Phi_1 = \Phi_2 - 4.2 \cdot \Delta x$

Very important in finite difference method:

For each output (unknown) field variable (e.g., P, T, v_x, v_y, etc.), we must assign a Dirichlet BC to at least 1 node. This is required in order to compute finite differences from an initial value.

3b) Apply PDEs and boundary conditions to each nodal point.

Build matrices and vectors

*A*x* = *b* (standard notation)
or in Gerya's book:
*L*S* = *R* (Left matrix*Solution = Right vector)

L is a n x n sparse matrix with coefficients
R is a n x 1 vector with the right-hand sides
S is a n x 1 vector with the unknowns

$$L_{1,1}S_1 + L_{1,2}S_2 + L_{1,3}S_3 + \ldots + L_{1,n-1}S_{n-1} + L_{1,n}S_n = R_1$$

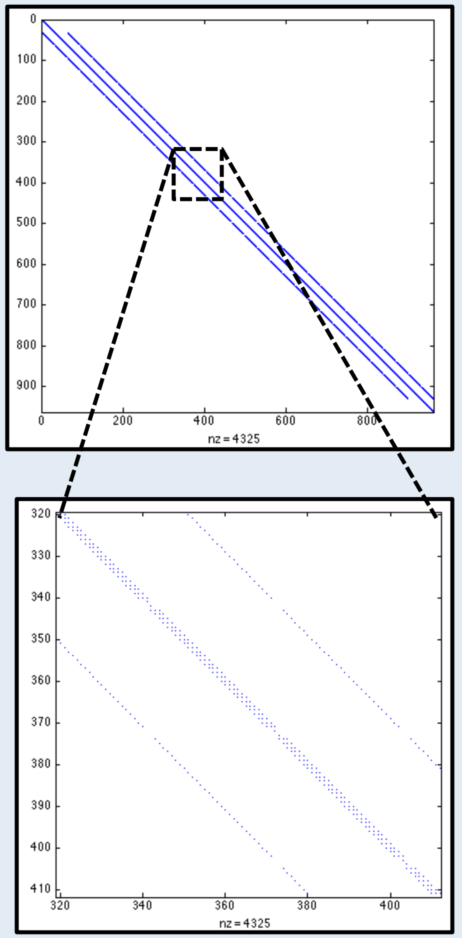
$$L_{2,1}S_1 + L_{2,2}S_2 + L_{2,3}S_3 + \ldots + L_{2,n-1}S_{n-1} + L_{2,n}S_n = R_2$$

$$\ldots$$

$$L_{n-1,1}S_1 + L_{n-1,2}S_2 + L_{n-1,3}S_3 + \ldots + L_{n-1,n-1}S_{n-1} + L_{n-1,n}S_n = R_{n-1}$$

$$L_{n,1}S_1 + L_{n,2}S_2 + L_{n,3}S_3 + \ldots + L_{n,n-1}S_{n-1} + L_{n,n}S_n = R_n$$

Matrix L for 2D grid: 31 x 31 nodes



MatLab command: spy(L)

4) Solve the system of linear equations

A*x = b

or in Gerya's book:

L*S = R

Two methods for solving a system of equations:

$$
\begin{aligned}
L_{1,1}S_1 + L_{1,2}S_2 + L_{1,3}S_3 + \ldots + L_{1,n-1}S_{n-1} + L_{1,n}S_n &= R_1\\
L_{2,1}S_1 + L_{2,2}S_2 + L_{2,3}S_3 + \ldots + L_{2,n-1}S_{n-1} + L_{2,n}S_n &= R_2\\
&\ldots\\
L_{n-1,1}S_1 + L_{n-1,2}S_2 + L_{n-1,3}S_3 + \ldots + L_{n-1,n-1}S_{n-1} + L_{n-1,n}S_n &= R_{n-1}\\
L_{n,1}S_1 + L_{n,2}S_2 + L_{n,3}S_3 + \ldots + L_{n,n-1}S_{n-1} + L_{n,n}S_n &= R_n
\end{aligned}
$$

1) iterative (Jacobi, Gauss-Siedel iteration):

define initial guess $S_i^{current}$ for each unknown,

then find residual ΔR_i.

Update solution for each unknown S_i^{new} by

means of relaxation parameter $0<\vartheta_i<1$.

In Jacobi →simultaneous update at the end of the cycle

In Gauss-Siedel→update during the cycle

$$
\begin{aligned}
\Delta R_1 &= R_1 - L_{1,1}S_1^{current} - L_{1,2}S_2^{current} - L_{1,3}S_3^{current} - \ldots - L_{1,n-1}S_{n-1}^{current} - L_{1,n}S_n^{current}\\
\Delta R_2 &= R_2 - L_{2,1}S_1^{current} - L_{2,2}S_2^{current} - L_{2,3}S_3^{current} - \ldots - L_{2,n-1}S_{n-1}^{current} - L_{2,n}S_n^{current}\\
&\ldots\\
\Delta R_{n-1} &= R_{n-1} - L_{n-1,1}S_1^{current} - L_{n-1,2}S_2^{current} - L_{n-1,3}S_3^{current} - \ldots - L_{n-1,n-1}S_{n-1}^{current} - L_{n-1,n}S_n^{current}\\
\Delta R_n &= R_n - L_{n,1}S_1^{current} - L_{n,2}S_2^{current} - L_{n,3}S_3^{current} - \ldots - L_{n,n-1}S_{n-1}^{current} - L_{n,n}S_n^{current}.
\end{aligned}
$$

Iterate while |ΔR|>req. accuracy

$$
\begin{aligned}
S_1^{new} &= S_1^{current} + \theta_1 \frac{\Delta R_1}{L_{1,1}}\\
S_2^{new} &= S_2^{current} + \theta_2 \frac{\Delta R_2}{L_{2,2}}\\
&\ldots\\
S_{n-1}^{new} &= S_{n-1}^{current} + \theta_{n-1} \frac{\Delta R_{n-1}}{L_{n-1,n-1}}\\
S_n^{new} &= S_n^{current} + \theta_n \frac{\Delta R_n}{L_{n,n}},
\end{aligned}
$$

4) Solve the system of linear equations

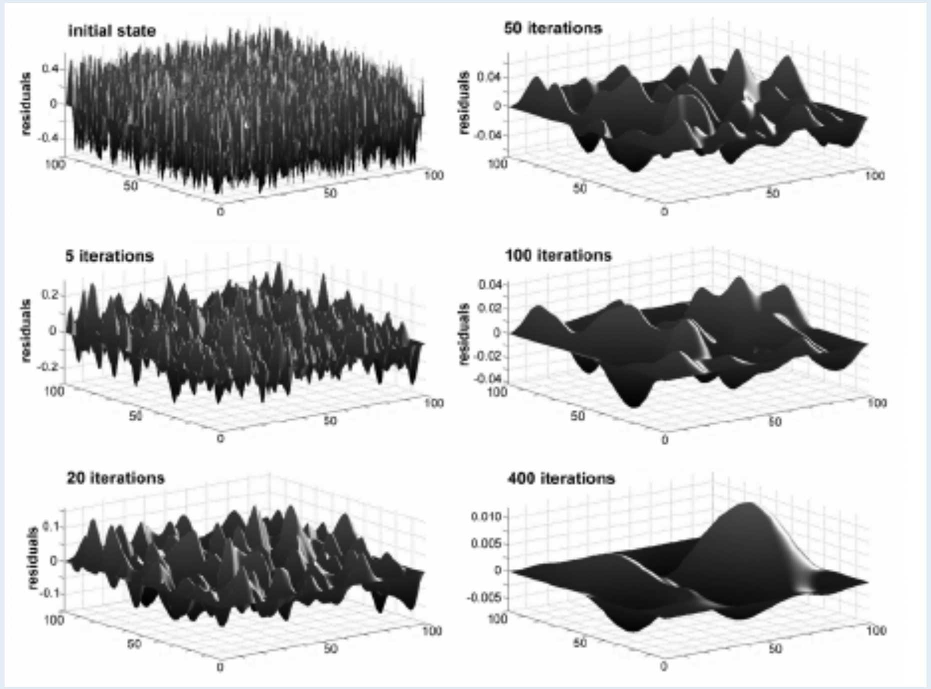
Two methods for solving a system of equations:

1) iterative (Jacobi, Gauss-Siedel iteration): because of the small memory consumption the relatively small amount of operations, iterative solvers are commonly used for solving large (3D) problems. However, the accuracy of the solution may be low and convergence may be slow because of large differences in matrix coefficients (high condition number) and residuals at wavelength longer than grid spacing.

Matrix preconditioning reduces the condition number and improve the convergence ($APP^{-1}b=x$).

Otherwise…

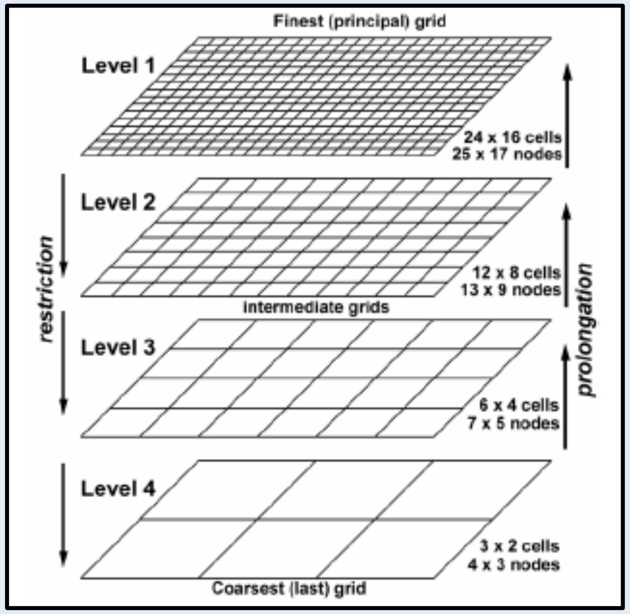
2D Poisson equation (100 x 100)



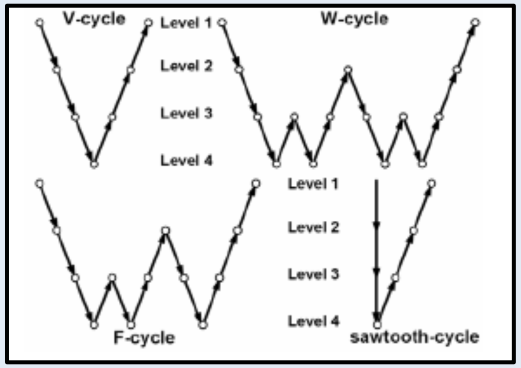
… MULTIGRID method

Higher accuracy and faster convergence is achieved by using the MULTIGRID method, where a series of coarser grids are built by interpolating residuals and field variables from finest grid (restriction). Iterations are then applied to these grids (smoothing), to find corrections at different wavelengths to the correct solution on the finest grid. The corrections are then interpolated back to finer grids (prolongation).

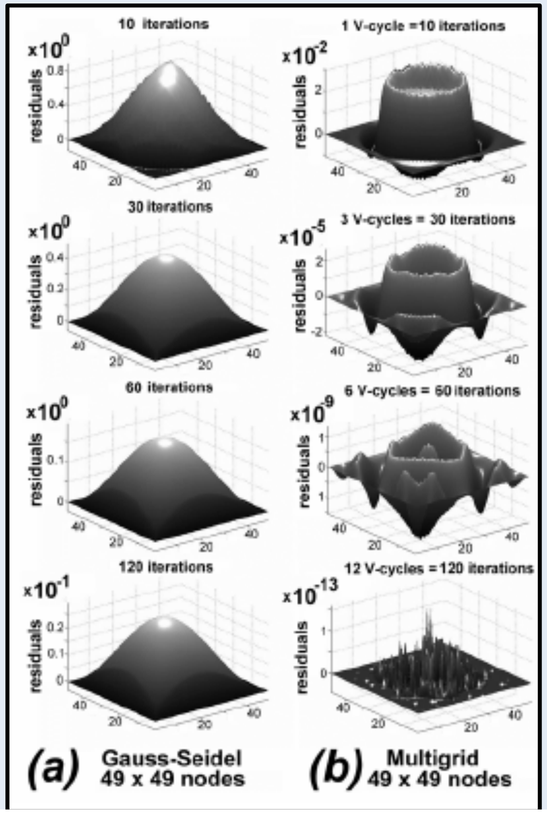
2D multigrid structure



Types of cycles



2D Poisson's equation



4) Solve the system of linear equations

Two methods for solving a system of equations:

2) Direct methods:

- Gaussian elimination
- Gauss-Jordan elimination (find matrix inverse: $A*x = b \rightarrow x = b*A^{-1}$)
- Faster methods based on matrix reordering, factorization (i.e., decomposition: LU decomposition, Cholesky decomposition), pivoting, etc.

Fastest direct solver nowadays:

- PARDISO (from Intel MKL libraries, http://www.pardiso-project.org/)
- MUMPS (open source, http://graal.ens-lyon.fr/MUMPS/)
- MILAMIN (open source, for FEM based codes, http://milamin.org/)

In MatLab, matrix inversion can be done by using:
S=inv(L)*R or better **S=L\R;**

Direct solvers do not require an initial guess and have accuracy to computer precision. However, they are computationally expensive (memory $\propto O(n^2)$, operations $\propto O(n^2\text{-}n^3)$, where n is the number of unknowns), and therefore can be used only for 1D and 2D (or small 3D) problems.

Numerical Modelling in Geosciences

Practice

Numerical solution of PDEs

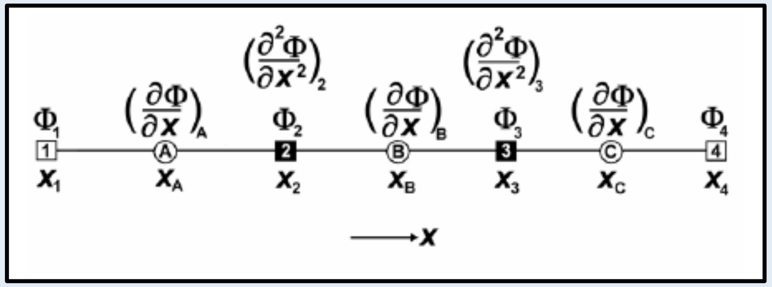
1) Define an Eulerian grid of nodal points…

… and save their space coordinates.

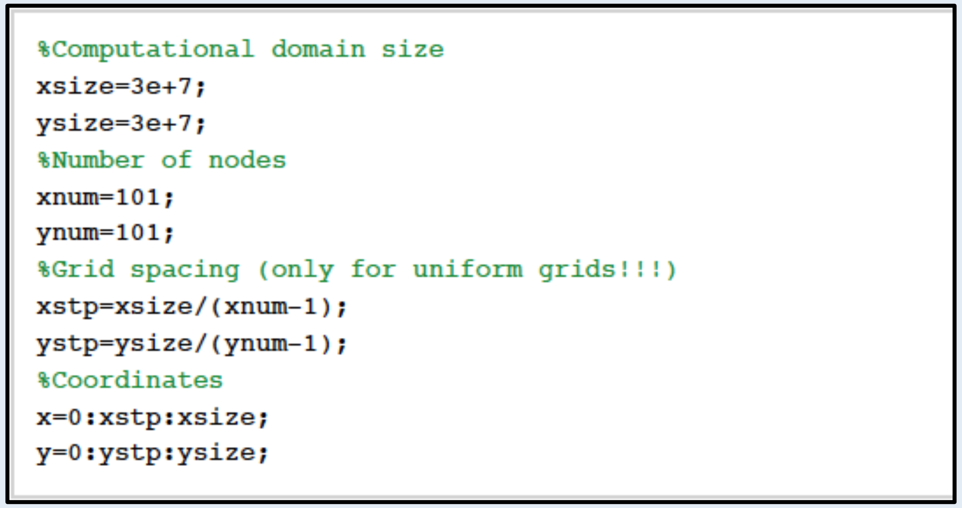
The grid can be uniform (equal distance between nodal points) or non-uniform, but points need to be orthogonal. Lines connecting points are fictitious.

With this process we subdivide the continuous medium into discrete parts.

1D grid

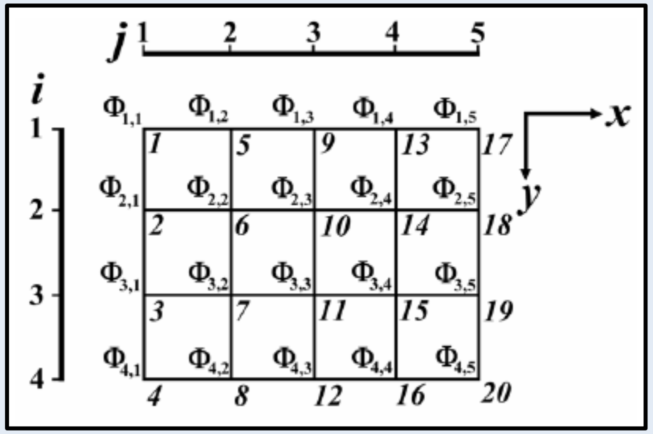


Φ_1
$\left(\frac{\partial\Phi}{\partial x}\right)_A$
$\left(\frac{\partial^2\Phi}{\partial x^2}\right)_2$
Φ_2
$\left(\frac{\partial\Phi}{\partial x}\right)_B$
$\left(\frac{\partial^2\Phi}{\partial x^2}\right)_3$
Φ_3
$\left(\frac{\partial\Phi}{\partial x}\right)_C$
Φ_4
1
A
2
B
3
C
4
x_1
x_A
x_2
x_B
x_3
x_C
x_4
x



```
%Computational domain size
xsize=3e+7;
ysize=3e+7;
%Number of nodes
xnum=101;
ynum=101;
%Grid spacing (only for uniform grids!!!)
xstp=xsize/(xnum-1);
ystp=ysize/(ynum-1);
%Coordinates
x=0:xstp:xsize;
y=0:ystp:ysize;
```

2D grid



2D geometrical indexing:

Phi(3,4)

2D global indexing:
gi = i + N_i *(j-1)

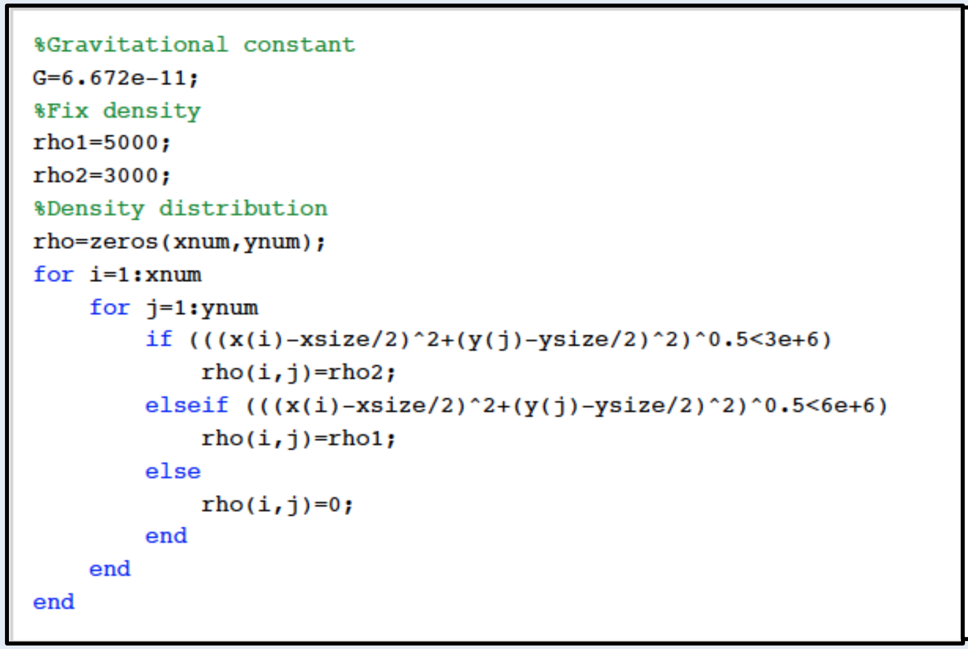
Phi(15)

2) Assign input field variables to the nodes

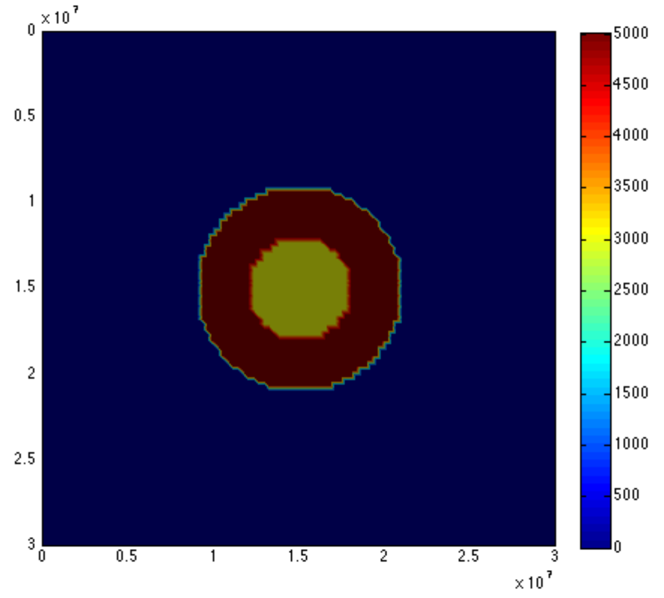
In the case of 2D Poisson's equation for gravity:

$$\Delta\Phi = 4\pi G\rho(x,y)$$

we need to assign density to each node (+ Gravitational constant).



```
%Gravitational constant
G=6.672e-11;
%Fix density
rho1=5000;
rho2=3000;
%Density distribution
rho=zeros(xnum,ynum);
for i=1:xnum
    for j=1:ynum
        if (((x(i)-xsize/2)^2+(y(j)-ysize/2)^2)^0.5<3e+6)
            rho(i,j)=rho2;
        elseif (((x(i)-xsize/2)^2+(y(j)-ysize/2)^2)^0.5<6e+6)
            rho(i,j)=rho1;
        else
            rho(i,j)=0;
        end
    end
end
```



3) Apply PDEs and boundary conditions to each nodal point.

Build matrices and vectors

*A*x* = *b* (standard notation)

or in Gerya's book:

*L*S* = *R* (Left matrix*Solution = Right vector)

L is a n x n sparse matrix with coefficients

R is a n x 1 vector with the right-hand sides

S is a n x 1 vector with the unknowns

$$L_{1,1}S_1 + L_{1,2}S_2 + L_{1,3}S_3 + \ldots + L_{1,n-1}S_{n-1} + L_{1,n}S_n = R_1$$

$$L_{2,1}S_1 + L_{2,2}S_2 + L_{2,3}S_3 + \ldots + L_{2,n-1}S_{n-1} + L_{2,n}S_n = R_2$$

$$\ldots$$

$$L_{n-1,1}S_1 + L_{n-1,2}S_2 + L_{n-1,3}S_3 + \ldots + L_{n-1,n-1}S_{n-1} + L_{n-1,n}S_n = R_{n-1}$$

$$L_{n,1}S_1 + L_{n,2}S_2 + L_{n,3}S_3 + \ldots + L_{n,n-1}S_{n-1} + L_{n,n}S_n = R_n$$

```
%Left sparse matrix
L=sparse(xnum*ynum,xnum*ynum);
%Right vector
R=zeros(xnum*ynum,1);
%Square distances
x2=xstp^2;
y2=ystp^2;
%Apply FD to each node of the 2D grid
for i=1:xnum
    for j=1:ynum
        %Global index
        gi=j+(i-1)*ynum;

        %Boundary nodes
        if(i==1 || i==xnum || j==1 || j==ynum)
            R(gi,1)=1e+8;
            L(gi,gi)=1;
        %Inner nodes
        else
            %Right part of equation
            R(gi,1)=4*pi*G*rho(i,j);
            %Left part of equation
            %Phi(i,j)
            L(gi,gi) = -2/x2-2/y2;
            %Phi(i,j+1)
            L(gi,gi+1) = 1/y2;
            %Phi(i,j-1)
            L(gi,gi-1) = 1/y2;
            %Phi(i,i+1)
            L(gi,gi+ynum) = 1/x2;
            %Phi(i,i-1)
            L(gi,gi-ynum) = 1/x2;
        end
    end
end
```

4) Solve the system of linear equations

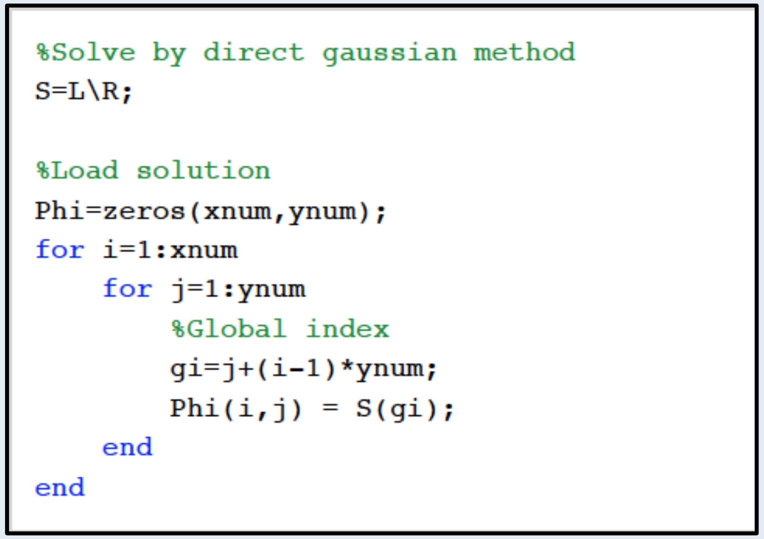
Remember: A*x = b $\rightarrow$ x = b*A^{-1}

or in Gerya's book: L*S = R (Left matrix*Solution = Right vector) $\rightarrow$ S = R*L^{-1}

This requires inversion of left matrix with all the coefficients.

In MatLab, matrix inversion can be done by using:

S=L\R;



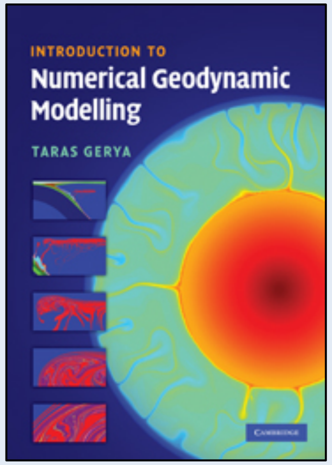
```
%Solve by direct gaussian method
S=L\R;

%Load solution
Phi=zeros(xnum,ynum);
for i=1:xnum
    for j=1:ynum
        %Global index
        gi=j+(i-1)*ynum;
        Phi(i,j) = S(gi);
    end
end
```

Homework

Read the chapter 3 and pp. 193-200 of textbook:

Gerya, T. *Introduction to numerical geodynamic modelling*. Cambridge University Press, 345 pp. (2010)



Exercise with Poisson's equation by changing density distribution and boundary conditions