JACOBI SOLVER (2D Stencil)

(discretized Laplace operator)

Finite difference method

Ordinary or partial differential equations underly many natural phenomena but can be difficult to solve.

One method to simplify the equations is called *finite difference*. This assumes differentials can be simplified such as:

$$\frac{\partial f(x)}{\partial h} = f(x)' \sim \frac{f(x+h) - f(x)}{h}$$

From the definition of f'(x) with error O(h)

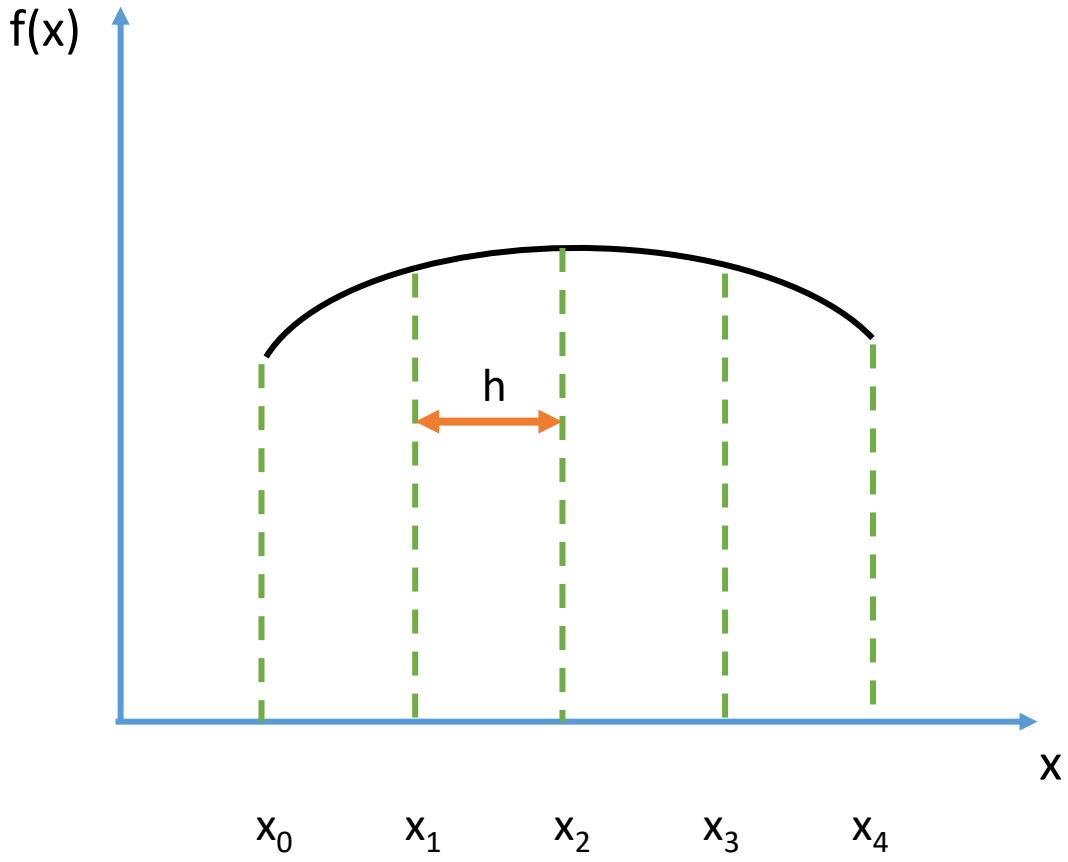
$$f'(x) \sim \frac{f(x+h) - f(x-h)}{2h}$$

Central difference from Taylor expansion, error $O(h^2)$

$$f''(x) \sim \frac{f(x-h) - 2f(x) + f(x+h)}{h^2}$$

Central difference second deriv. approximation

Finite difference method



By *discretising* the function, we wish to calculate, we can use finite differences to transform a differential equation into simpler algebraic equations.

So, if we let

$$y_i = f(x_i)$$

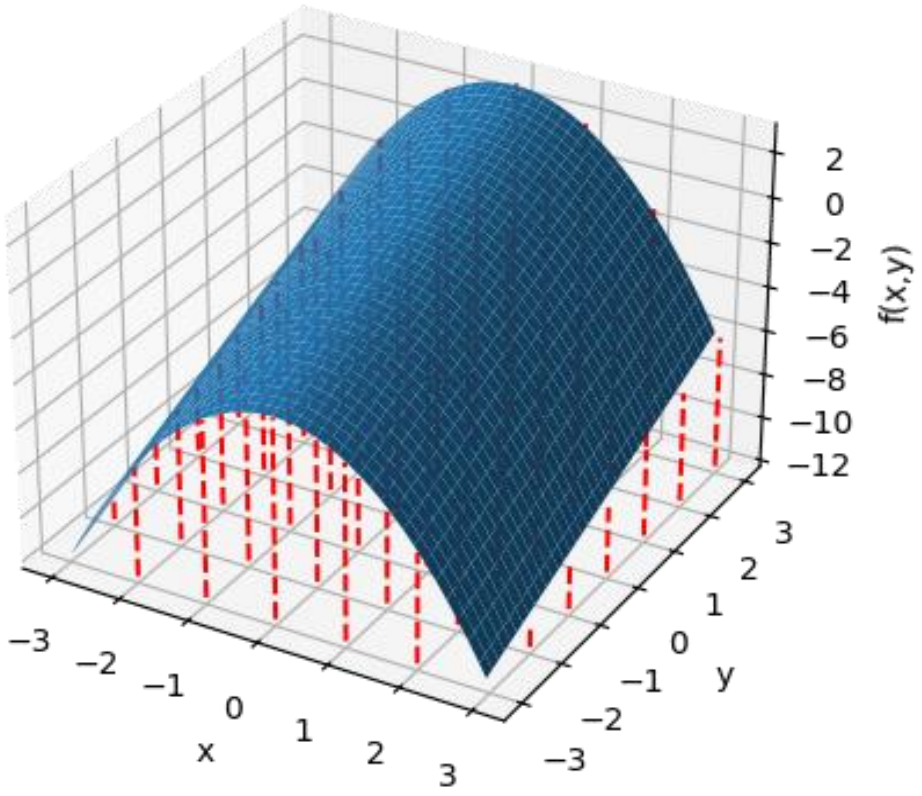
then

$$f'(x_i) = \frac{y_{i+1} - y_{i-1}}{2h}$$

i.e., our differential equations turn into simple subtractions and divisions.

We need at least two points to calculate our values so at the boundaries we need fixed values for which we dont calculate the derivatives and are unaffected by our operations. These are called *boundary conditions*, e.g. $f(x_0) = 0.0, f(x_4) = 10.0$.

Finite Differences in 2D



For functions of both x and y we can discretize the surface onto a 2D grid.

The same reasoning as for 1D applies – we fix the x and y spacings to be the same.

Boundary conditions now apply in both x and y directions.

2D Finite Difference Method – Poisson's equation

In the 2D version of Poisson's equation we have the following formulae:

$$-\Delta u = f$$ ← *in the domain*

$$u = 0$$ ← *outside the domain (boundary conditions)*

$$\Delta u = \partial_{xx} u + \partial_{yy} u$$ *the Laplace operator*

Using finite differences on a 2D grid, for second derivatives:

$$\partial_{xx} u(x_i, y_i) \approx \frac{u(x_i + h, y_i) - 2u(x_i, y_i) + u(x_i - h, y_i)}{h^2}$$

$$\partial_{yy} u(x_i, y_i) \approx \frac{u(x_i, y_i + h) - 2u(x_i, y_i) + u(x_i, y_i - h)}{h^2}$$

2D Finite Difference Method – Poisson's equation

Inserting into Poisson's equation and re-inserting, we have

$$1/h^2\left(u(x_i+h,y_i)-u(x_i-h,y_i)+4u(x_i,y_i)-u(x_i,y_i+h)-u(x_i,y-h)\right)=f_i$$

If we write $u(x_i,y_j)=u_{ij}$

the equation is more simply written as

$$1/h^2\left(u_{i+1,j}-u_{i-1,j}+4u_{i,j}-u_{i,j+1}-u_{i,j-1}\right)=f_i$$

Or in matrix form

$$1/h^2\begin{pmatrix}0 & -u_{i-1,j} & 0\\ -u_{i,j-1} & 4u_{ij} & -u_{i,j+1}\\ 0 & -u_{i+1,j} & 0\end{pmatrix}$$

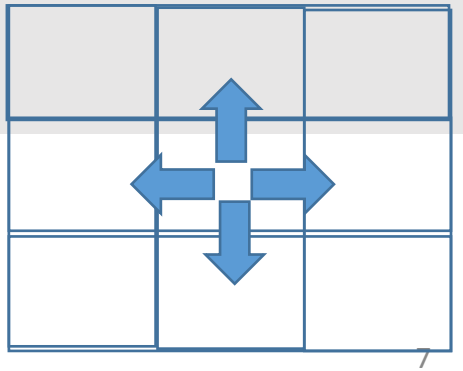
This is known as a *stencil* and can be applied all elements of the input grid.

Jacobi Stencil Solver (aka Laplace method)

- The Jacobi method is an iterative algorithm for solving a system of linear equations with finite differences.
- Solutions for the 2D poisson equations can be implemented very simply with the 4-point stencil
- The updated value for each cell is just the average of the neighbouring 4 cells
- Convergence: |old grid- updated grid| < tolerance

```
REAL A(0:n+1,0:n+1), B(1:n,1:n)
...
!Main Loop
DO WHILE(.NOT.converged)
   ! perform 4 point stencil
   DO j=1, n
      DO i=1, n
         B(i,j)=0.25*(A(i-1,j)+A(i+1,j)+A(i,j-1)+A(i,j+1))
      END DO
   END DO

   ! copy result back into array A
   DO j=1, n
      DO i=1, n
         A(i,j) = B(i,j)
      END DO
   END DO
...
! convergence test
END DO
```



4-point stencil

Jacobi Grid

- Can be used as a simple model for heatflow in a container
- The boundary conditions are then the temperatures of the surrounding walls