

Finite Difference Methods

Finite differences in 1D



Sparse matrix formulation



Boundary conditions



Solving two-point boundary value problems



Setting up higher dimensional problems

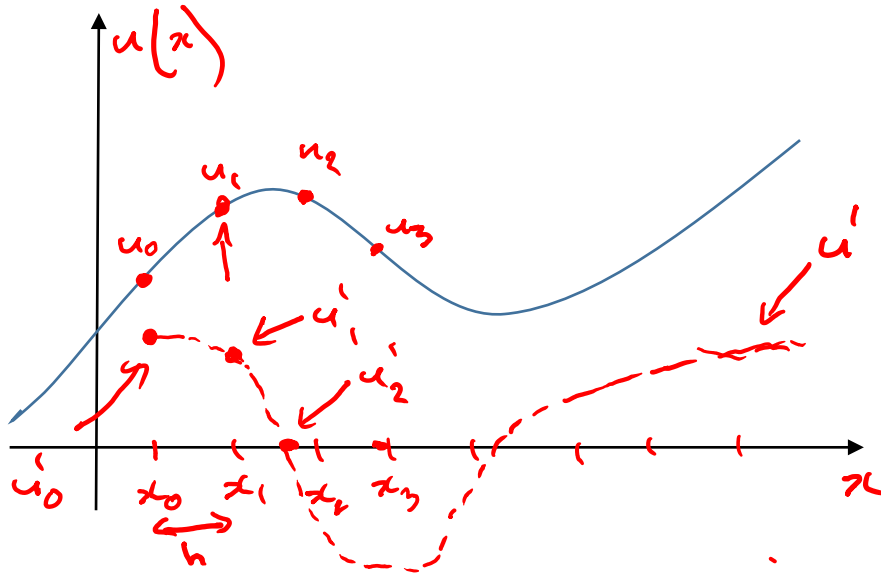
Finite differences in higher dimensions



Boundary conditions

Other approaches

Imagine we have a function $u(x)$ with known values at a fixed number of equally-spaced nodes, with spacing h :



$$u'_1 = \frac{u_2 - u_0}{2h}$$

$$\frac{du}{dx} = f(x, u)$$

We saw in Week 2 that an approximation for the first derivative of u at the j^{th} node is

$$u'_j = \frac{1}{2h} (u_{j+1} - u_{j-1})$$

If we represent u as a *vector*, then the numerical derivative can be represented as a matrix:

$$\begin{array}{c} u' \\ \downarrow \end{array} \begin{pmatrix} u_0 \\ u_1 \\ u_2 \\ \vdots \\ u_j \\ \vdots \end{pmatrix} = \begin{pmatrix} \frac{1}{2h} & 0 & \frac{1}{2h} & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & -\frac{1}{2h} & 0 & \frac{1}{2h} & 0 & 0 & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \end{pmatrix} \begin{pmatrix} u_0 \\ u_1 \\ u_2 \\ \vdots \\ u_j \\ \vdots \end{pmatrix}$$

Handwritten annotations in red:

- Arrows pointing to the u_0 and u_1 terms in the first row of the matrix, with a label u_0 and u_1 respectively.
- Arrows pointing to the u_2 and u_1 terms in the second row of the matrix, with a label u_2 and u_1 respectively.
- Arrows pointing to the u_j and u_{j-1} terms in the j -th row of the matrix, with a label u_j and u_{j-1} respectively.

$$u'_j = \frac{1}{2h} (u_{j+1} - u_{j-1})$$

$$u'_2 = \frac{1}{2h} (u_3 - u_1)$$

This is a *tridiagonal, sparse matrix*.

The first-order central difference derivative is

$$u'_j = \frac{1}{2h}(-u_{j-1} + u_{j+1})$$

Which leads to the matrix equation

$$\begin{bmatrix} u'_0 \\ \vdots \\ u'_{j-1} \\ u'_j \\ u'_{j+1} \\ \vdots \end{bmatrix} = \frac{1}{2h} \begin{bmatrix} -2 & 2 & & & \\ -1 & 0 & 1 & & \\ & \ddots & & & \\ & & -1 & 0 & 1 \\ & & & -1 & 0 & 1 \\ & & & & -1 & 0 & 1 \\ & & & & & \ddots & \\ & & & & & & -1 & 0 & 1 \\ & & & & & & & -2 & 2 \end{bmatrix} \begin{bmatrix} u_0 \\ \vdots \\ u_{j-1} \\ u_j \\ u_{j+1} \\ \vdots \\ u_{N-1} \end{bmatrix}$$

Handwritten annotations on the matrix equation:

- A red arrow points to the first row of the matrix, which is highlighted in orange. The elements -2 and 2 are circled in red.
- A red arrow points to the element 1 in the j -th row of the matrix, which is circled in red.
- A red arrow points to the element -1 in the $(j+1)$ -th row of the matrix, which is circled in red.
- A red arrow points to the last row of the matrix, which is highlighted in orange. The elements -2 and 2 are circled in red.

forward difference

$$u'_0 = \frac{1}{h}(u_1 - u_0)$$

$$= -\frac{1}{2h}u_0 + \frac{1}{2h}u_1$$

The second-order derivative (using central differences) is

$$u_j'' = \frac{1}{h^2} (u_{j-1} - 2u_j + u_{j+1})$$

Which leads to the matrix equation

$$\begin{pmatrix} \vdots \\ \vdots \\ u_{j-1}'' \\ u_j'' \\ u_{j+1}'' \\ \vdots \end{pmatrix} = \frac{1}{h^2} \begin{pmatrix} 1 & -2 & 1 & & & \\ & 1 & -2 & 1 & & \\ & & \ddots & \ddots & \ddots & \\ & & & 1 & -2 & 1 \\ & & & & 1 & -2 & 1 \\ & & & & & 1 & -2 & 1 \\ & & & & & & 1 & -2 & 1 \end{pmatrix} \begin{pmatrix} u_0 \\ \vdots \\ u_{j-1} \\ u_j \\ u_{j+1} \\ \vdots \\ u_{N-1} \end{pmatrix}$$

Handwritten annotations: A red arrow points from D_2 to the matrix. A red arrow points from u to the vector $\begin{pmatrix} u_0 \\ \vdots \\ u_{j-1} \\ u_j \\ u_{j+1} \\ \vdots \\ u_{N-1} \end{pmatrix}$. A red arrow points from u'' to the vector $\begin{pmatrix} \vdots \\ \vdots \\ u_{j-1}'' \\ u_j'' \\ u_{j+1}'' \\ \vdots \end{pmatrix}$. The matrix is annotated with red diagonal lines and circles, indicating its banded structure.

We say that D_2 is a finite difference representation of the differential operator

$$L = \frac{d^2}{dx^2}$$

$$u'' = D_2 u$$

Finite differences for boundary value problems

We can solve differential equations numerically by substituting the matrix representations of the operator and then solving as if it were a matrix equation.

E.g. to solve

$$-\frac{d^2 u}{dx^2} = f(x)$$

We would set up the matrix equation

where

$$D_2 u = f$$
$$D_2 = \frac{1}{h^2} \begin{bmatrix} 1 & -2 & 1 & & & \\ & 1 & -2 & 1 & & \\ & & \ddots & \ddots & \ddots & \\ & & & 1 & -2 & 1 \\ & & & & 1 & -2 & 1 \\ & & & & & 1 & -2 & 1 \\ & & & & & & \ddots & \ddots \\ & & & & & & & 1 & -2 & 1 \\ & & & & & & & & 1 & -2 & 1 \end{bmatrix}$$

$$u = \begin{bmatrix} u_0 \\ \vdots \\ u_{j-1} \\ u_j \\ u_{j+1} \\ \vdots \\ u_{N-1} \end{bmatrix}$$

$$f = \begin{bmatrix} f(x_0) \\ \vdots \\ f(x_{j-1}) \\ f(x_j) \\ f(x_{j+1}) \\ \vdots \\ f(x_{N-1}) \end{bmatrix}$$

To solve this system (i.e. to find the unknowns u_j) we *invert* this equation:

$$u = D_2^{-1} f$$

However we still have to apply Boundary conditions

Boundary Conditions usually take the form

$$\underline{u(x_0) = A}, \quad \underline{u(x_{N-1}) = B} \quad (\text{two-point boundary value problems})$$



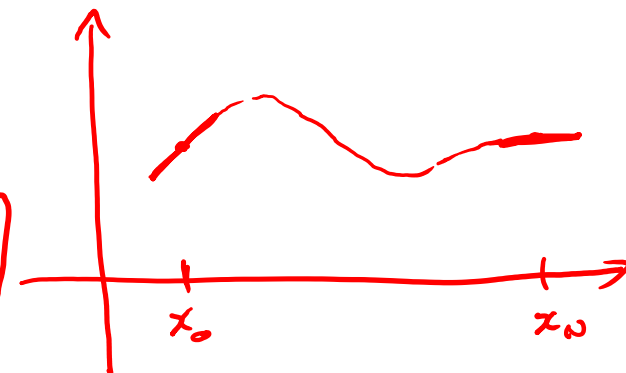
To enforce a two-point BVP we modify the first and last lines of the matrix equation:

$$\frac{1}{h^2} \begin{pmatrix} \boxed{1} & \boxed{-2} & \boxed{1} & & & \\ 1 & -2 & 1 & & & \\ & & \ddots & & & \\ & & & 1 & -2 & 1 \\ & & & & 1 & -2 & 1 \\ & & & & & 1 & -2 & 1 \\ & & & & & & \ddots & \\ & & & & & & & 1 & -2 & 1 \\ \boxed{1} & \boxed{-2} & \boxed{1} & & & \end{pmatrix} \begin{pmatrix} u_0 \\ \vdots \\ u_{j-1} \\ u_j \\ u_{j+1} \\ \vdots \\ u_{N-1} \end{pmatrix} = \begin{pmatrix} \boxed{f(x_0)} \\ \vdots \\ f(x_{j-1}) \\ f(x_j) \\ f(x_{j+1}) \\ \vdots \\ \boxed{f(x_{N-1})} \end{pmatrix}$$

To create a boundary condition with a derivative, e.g.

$$u'(x_0) = C \rightarrow u'(x_0) = \frac{1}{h}(u_1 - u_0) = \left[\frac{1}{h^2}(u_1) - \frac{1}{h^2}(u_0) \right] = C$$

You modify the first (or last) line to approximate the derivative using forward (or backward) differences:



$$\frac{1}{h^2} \begin{pmatrix} -1 & 1 & 0 & 0 & 0 & 0 \\ 1 & -2 & 1 & & & \\ & & \ddots & & & \\ & & & 1 & -2 & 1 \\ & & & & 1 & -2 & 1 \\ & & & & & 1 & -2 & 1 \\ & & & & & & \ddots & \\ & & & & & & & 1 & -2 & 1 \end{pmatrix} \begin{pmatrix} u_0 \\ u_1 \\ \vdots \\ u_{j-1} \\ u_j \\ u_{j+1} \\ \vdots \\ u_{N-1} \end{pmatrix} = \begin{pmatrix} C \\ \vdots \\ f(x_{j-1}) \\ f(x_j) \\ f(x_{j+1}) \\ \vdots \end{pmatrix}$$

$\uparrow D_2^B$
 $\uparrow u$
 $\uparrow f^B$

$$D_2^B u = f^B$$

$$\rightarrow u = (D_2^B)^{-1} f^B$$

Finite differences can be used to solve almost anything in 1D.

Advantages:

- Simple to implement
- Fast



Disadvantages:

- Requires knowledge of sparse systems
- A bit fiddly with boundary conditions
- Not so good for “stiff” problems (i.e. ones for which the function varies rapidly).

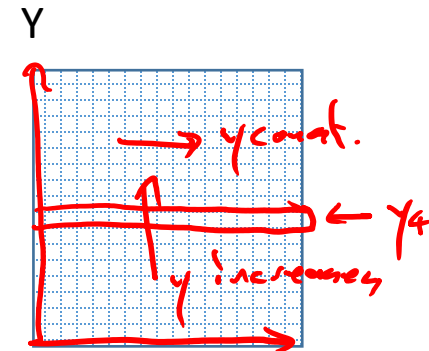
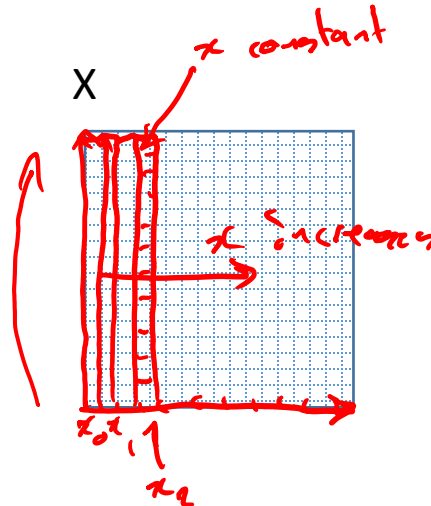
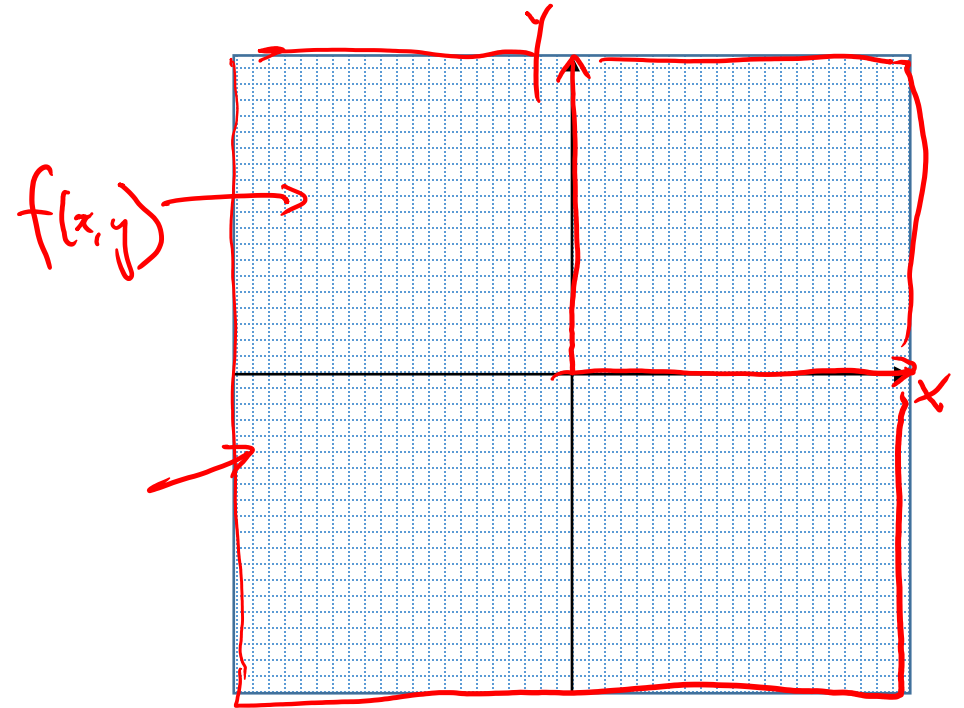
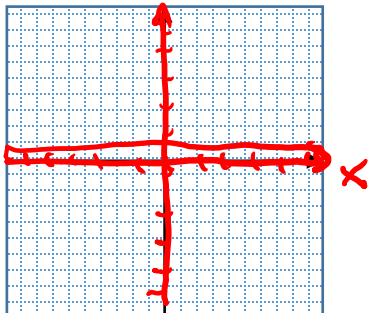
Finite Differences in higher dimensions

A big strength of the FD method is its ability to solve Partial Differential Equations (PDEs) in 2D and higher dimension.

To formulate a FD method in 2D we need to convert a 2D grid into a vector, and back again.

A 2D numpy array can be created using a combination of linspace and meshgrid (See Lab 4):

```
22 x = np.linspace(-5,5,10)
23 y = np.linspace(-5,5,10)
24
25 X,Y = np.meshgrid(x,y)
26
```

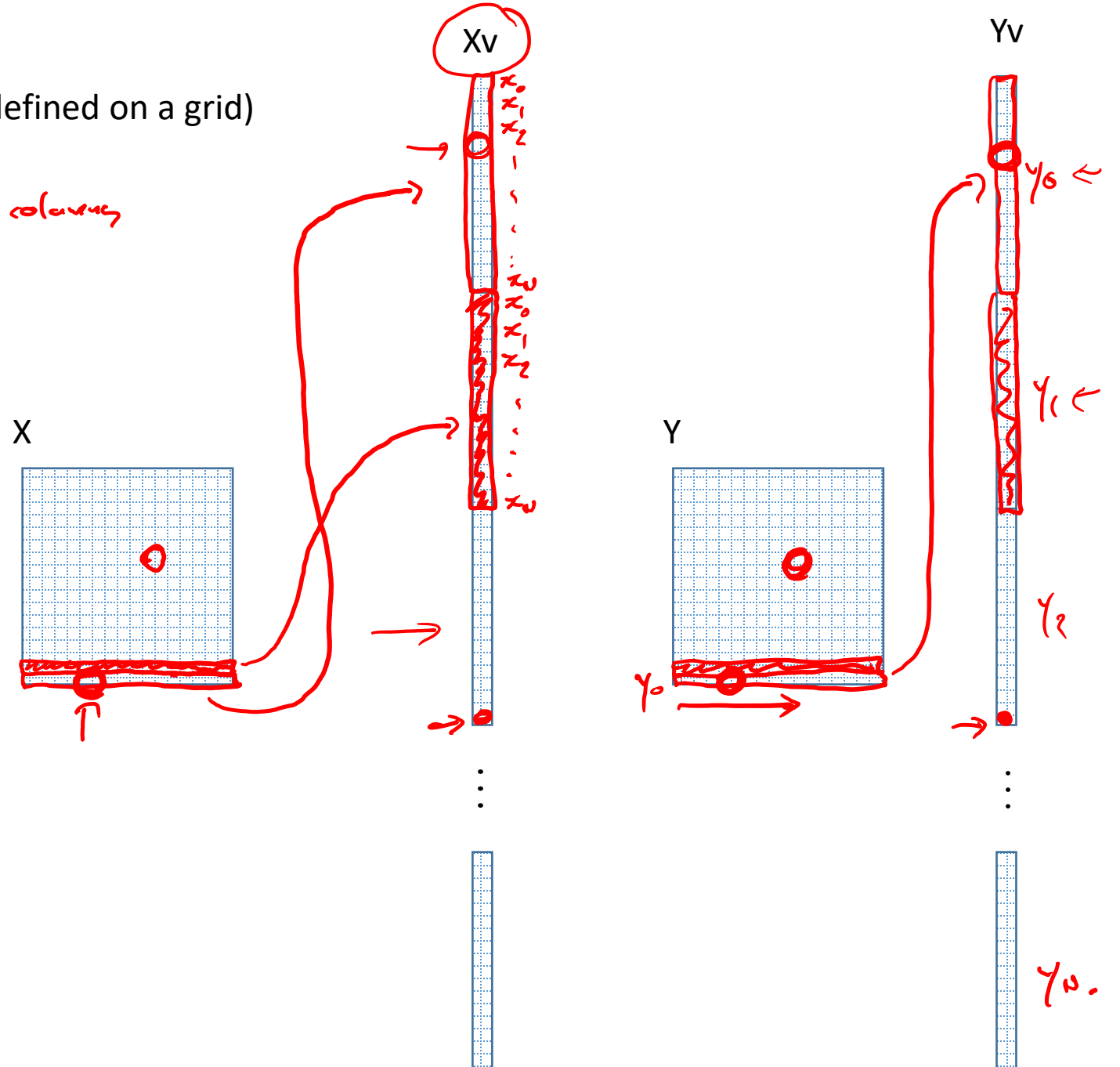


We can then unfold the grid (and any functions defined on a grid) using the numpy *reshape* function:

```
39 Xv = np.reshape(X, (10*10, 1))
40 Yv = np.reshape(Y, (10*10, 1))
```

Create a 100 x 1 column vector

(Whenever you do this, it helps (a lot) to draw this out on paper)



If we have a function defined on a grid this can also be disassembled:

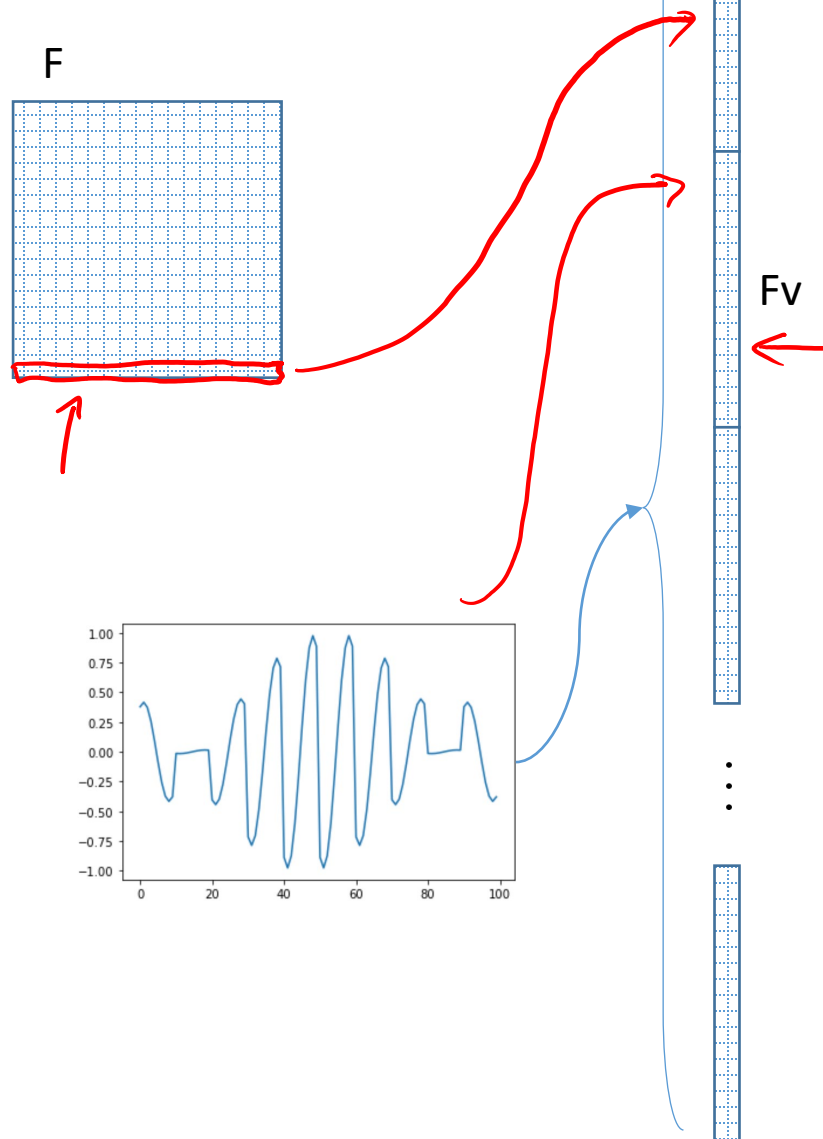
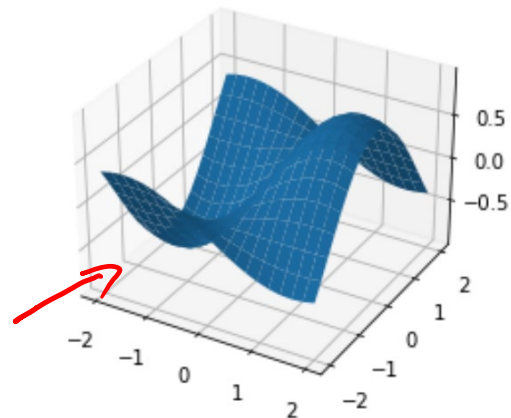
```
16 def f(x,y):  
17     f = np.sin(x)*np.cos(y)  
18     return f
```

⋮

```
26  
27 F = f(X,Y)  
28
```

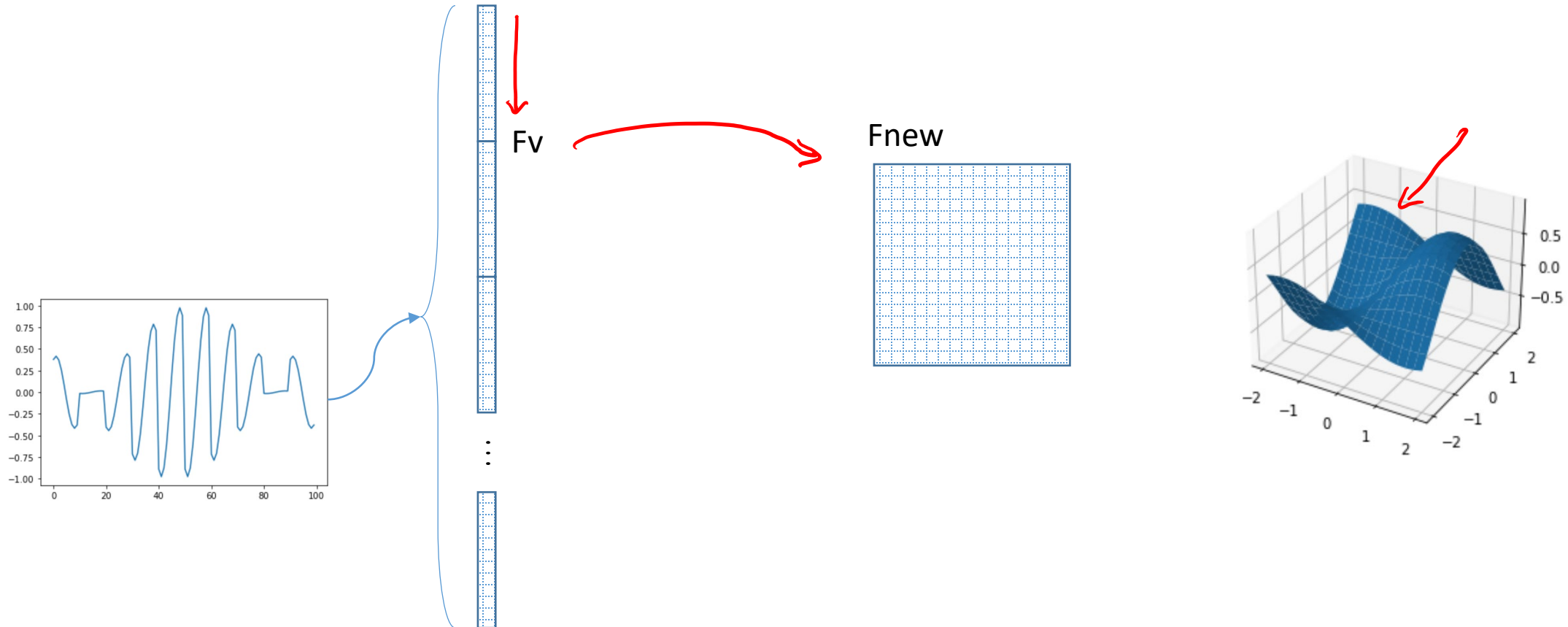
⋮

```
41  
42 Fv = np.reshape(F, (N*N,1))  
43
```



We can then re-assemble the functions into the original grid using reshape:

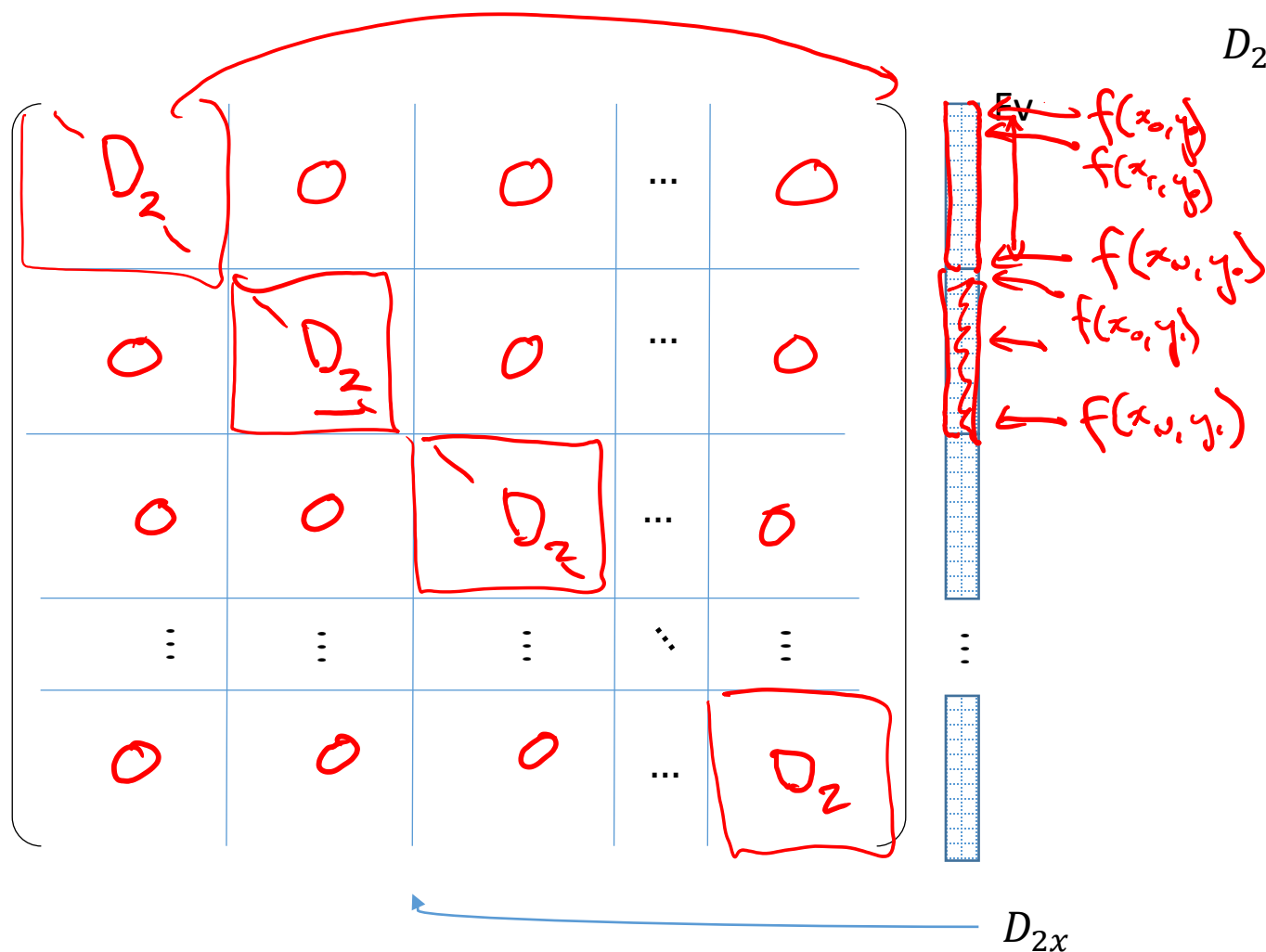
```
49  
50 Fnew = np.reshape(Fv, (N,N))  
51
```



The partial derivative with respect to x

Recall that D_2 gives the 2nd partial derivative with respect to x of a single vector $\underline{u}(x_j)$

To get the partial derivative, we build a block matrix consisting of the D_2 matrices:



$$D_2 = \frac{1}{h^2}$$

$$\begin{bmatrix} 1 & -2 & 1 & & & \\ 1 & -2 & 1 & & & \\ & & \ddots & & & \\ & & & 1 & -2 & 1 \\ & & & 1 & -2 & 1 \\ & & & & \ddots & \\ & & & & & 1 & -2 & 1 \\ & & & & & & 1 & -2 & 1 \end{bmatrix}$$

The partial derivative with respect to y

To get the partial derivative with respect to y, we need to interleave the D2 matrix so that it hits the right y values:

$$\frac{1}{h^2} \begin{pmatrix} \begin{matrix} -2 & 0 & 0 \\ 0 & -2 & 0 \\ 0 & 0 & -2 \end{matrix} & \begin{matrix} 1 \\ 0 \\ 0 \end{matrix} & \dots & \dots \\ \begin{matrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ \vdots & \vdots & \vdots \end{matrix} & \begin{matrix} -2 & 0 & 0 \\ 0 & -2 & 0 \\ 0 & 0 & -2 \end{matrix} & \begin{matrix} 1 \\ 0 \\ 0 \end{matrix} & \dots \\ \vdots & \vdots & \vdots & \ddots & \vdots \end{pmatrix} \begin{pmatrix} f(y_0) \\ f(y_1) \\ f(y_2) \\ \vdots \end{pmatrix} = D_{2y}$$

Diagram illustrating the construction of the matrix D_{2y} for the partial derivative with respect to y. The matrix is shown as a block matrix where the original 2D Laplacian matrix (with -2 on the diagonal and 1 on the off-diagonals) is interleaved with a vector of function values $f(y_i)$. The resulting matrix D_{2y} is used to compute the partial derivative.

$$D_2 = \frac{1}{h^2} \begin{pmatrix} 1 & -2 & 1 & & & \\ 1 & -2 & 1 & & & \\ & & \ddots & & & \\ & & & 1 & -2 & 1 \\ & & & 1 & -2 & 1 \\ & & & & \ddots & \\ & & & & & 1 & -2 & 1 \\ & & & & & 1 & -2 & 1 \end{pmatrix}$$

$$\frac{\partial^2 f}{\partial y^2} \bigg|_{y_1} \approx \frac{1}{h^2} (f(y_0) - 2f(y_1) + f(y_2))$$

Forming the Laplacian (and more complicated operators)

General PDEs like the Laplacian can be constructed just by adding matrices:

$$\nabla^2 = \frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2} \approx D_{2x} + D_{2y}$$

Handwritten notes: ∇^2 has a red arrow pointing to it. The sum $\frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2}$ is underlined in red. D_{2x} and D_{2y} are labeled as $M \times M$ matrices with a red arrow pointing to them. To the right, two vertical red lines represent matrices, with an upward arrow on the left and a downward arrow on the right, both labeled with a red M .

More complicated derivatives can also be constructed. E.g.

$$\frac{\partial^3}{\partial x^3} + \frac{\partial^4}{\partial y^4} = D_x D_x D_x + D_y D_y D_y D_y$$
$$2x \frac{\partial}{\partial x} + \frac{\partial^2}{\partial x \partial y} =$$

Handwritten notes: The expression $2x \frac{\partial}{\partial x} + \frac{\partial^2}{\partial x \partial y}$ is grouped by a blue bracket and labeled L in black. Below L is a red underline and a red arrow pointing up to it.

We can then create and solve PDEs by forming the matrix equation

$$\underline{L u = f}$$

$$u = L^{-1} f.$$

$\underbrace{u(x,0) = 0}_{\text{BC}}$

$\underbrace{u(x,0) = 0}_{\text{BC}}$

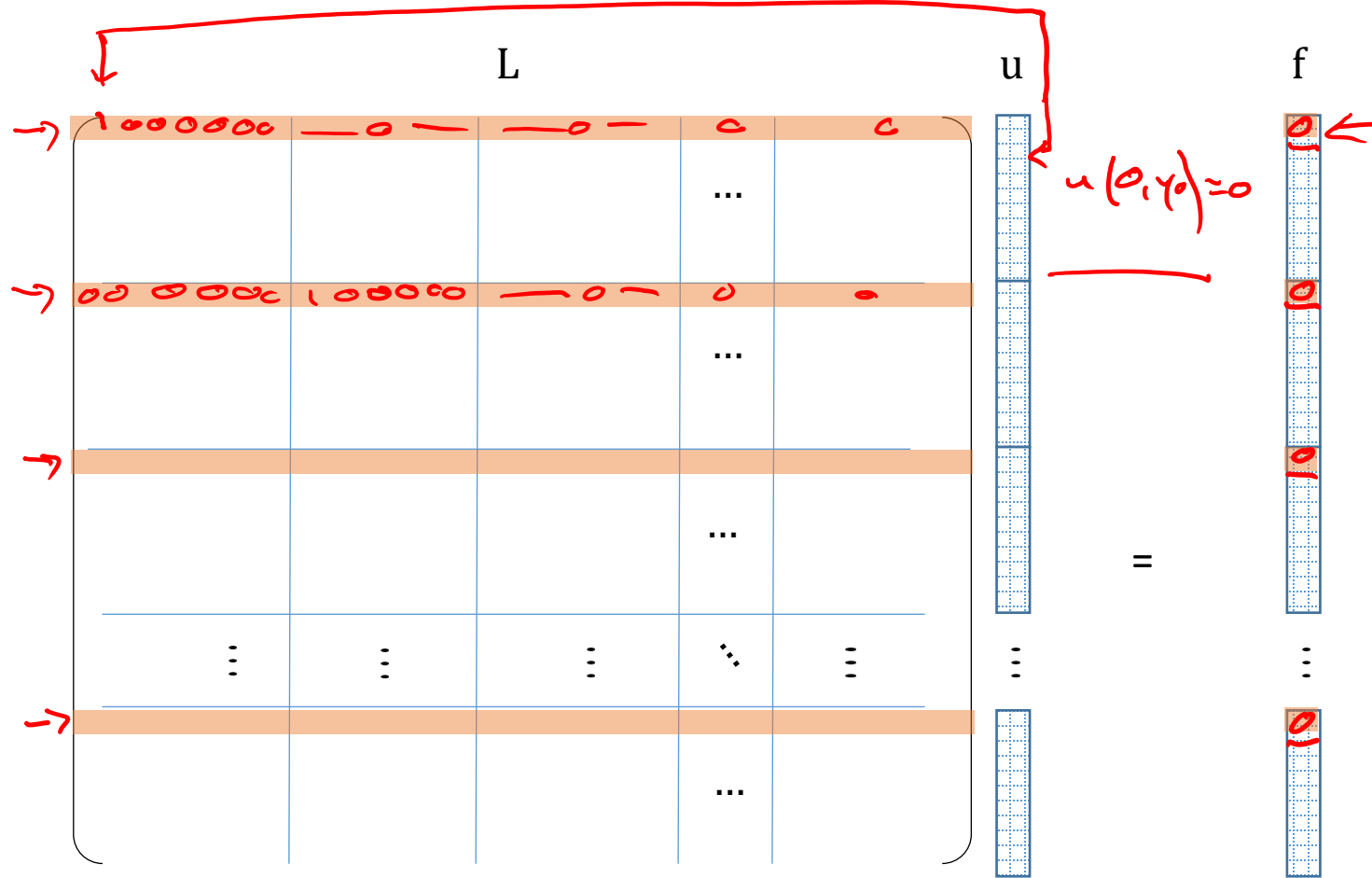
Diagram illustrating the matrix multiplication $L \cdot u = f$.

- L is a matrix with a green header row containing 0s and a blue dotted body.
- u is a blue dotted column vector.
- f is a blue dotted column vector with a green header element.
- A red bracket highlights the first row of L , and a red arrow points to the first element of u , labeled $u(x_i; 0)$.

Boundary conditions can be applied by modifying the resulting matrix
To create equations that are like, for example:

$$u(0, y) = 0:$$

$t=0$



Alternative numerical methods for PDEs:

Finite Element Method

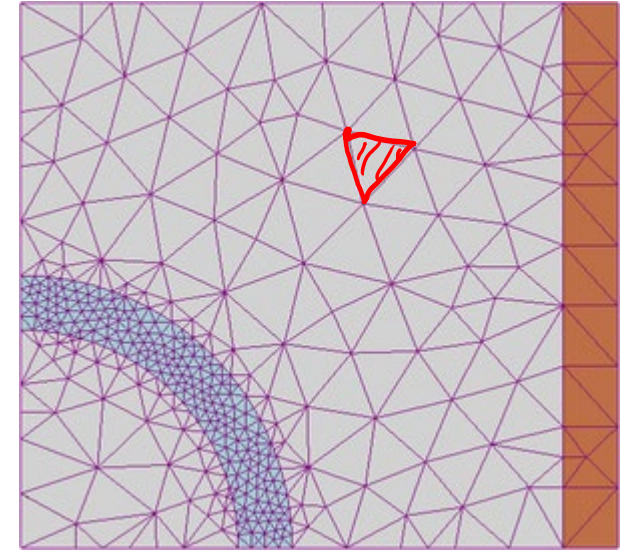
Expand in a mesh of simplexes (i.e. triangles), on which the solution is interpolated using polynomials

The Boundary Element Method

Uses Green's functions on the boundary to construct the solution in an interior domain

Semi-analytical or combined methods

e.g. Crank-Nicholson, Modal methods



The End