Week 9: Numerical differentiation and application to solving the heat equation

Finite differencing, pseudospectral methods and their application to the heat equation

Dr K Clough, Topics in Scientific computing, Autumn term 2023

Plan for today

1. Motivation - solution of the heat equation
2. Revision of last week’s interpolation content
3. Numerical differentiation - finite differencing
4. Numerical differentiation - pseudospectral methods
5. Application - solution of the heat equation

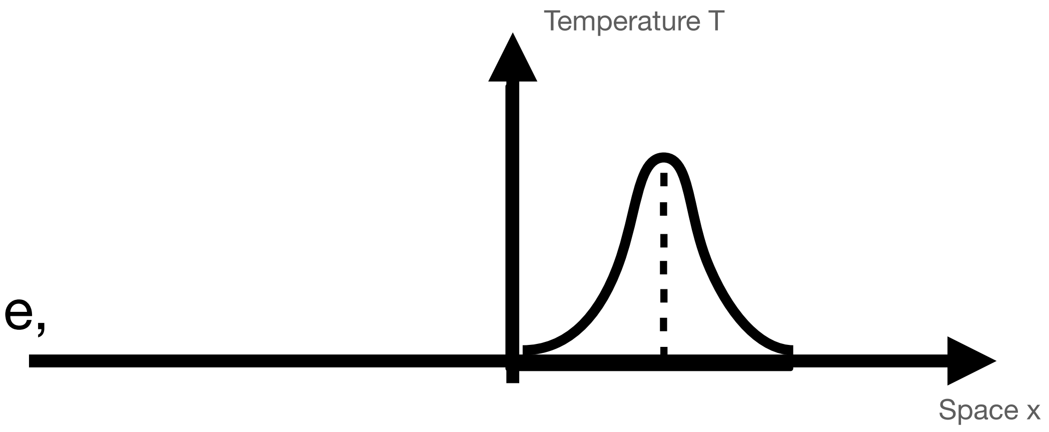
Motivation: solving the heat equation

- The heat equation

$$\frac{\partial T}{\partial t} = \alpha \frac{\partial^2 T}{\partial x^2}$$

- Tell me your initial temperature profile, and I can tell you how it changes over time

- Cannot usually just write down T = f(x,t) except in very simple cases



How does T change over time?

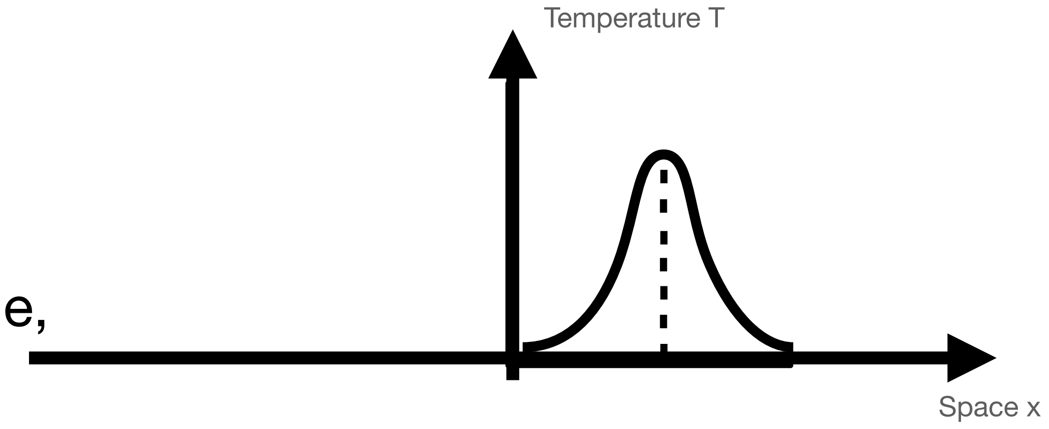
Motivation: solving the heat equation

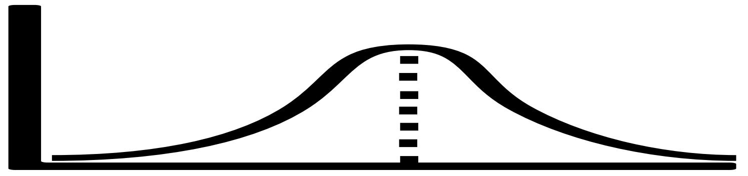
- The heat equation

$$\frac{\partial T}{\partial t} = \alpha \frac{\partial^2 T}{\partial x^2}$$

- Tell me your initial temperature profile, and I can tell you how it changes over time

- Cannot usually just write down T = f(x,t) except in very simple cases





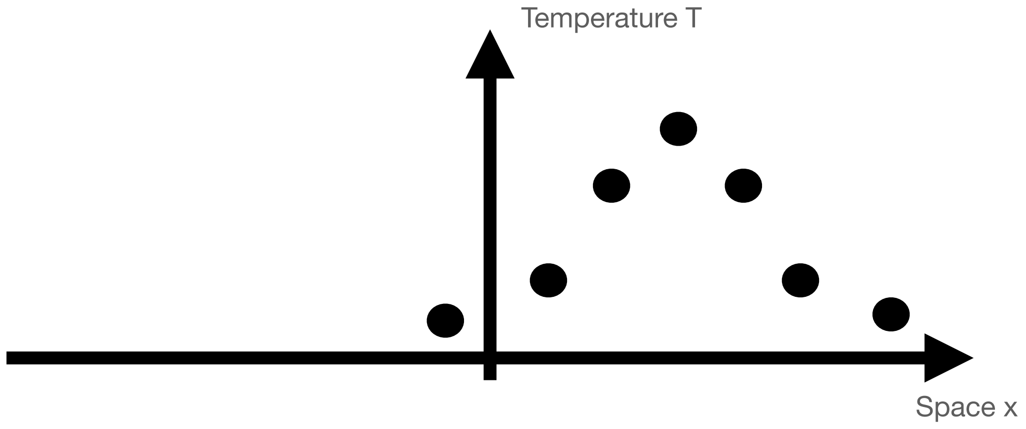
The temperature profile spreads out - the value decreases at a maximum where the second derivative is negative

Motivation: solving the heat equation

- The heat equation

$$\frac{\partial T}{\partial t} = \alpha \frac{\partial^2 T}{\partial x^2}$$

- How to find the spatial derivative of T if it is represented as a series of points?



Plan for today

1. ~~Motivation - solution of the heat equation~~
2. Revision of last week's interpolation content
3. Numerical differentiation - finite differencing
4. Numerical differentiation - pseudospectral methods
5. Application - solution of the heat equation

Lagrange polynomials use colocation at points

- Degree n Lagrange polynomials agree exactly with a function f(x) at n+1 distinct points, $(x_0, f(x_0)),\ (x_1, f(x_1)) \ldots (x_{n+1}, f(x_{n+1}))$
- First we construct the basis functions

$$L_k(x) = \prod_{i=0, i\neq k}^{n} \frac{(x - x_i)}{(x_k - x_i)}$$

What does the numerator and denominator achieve here?

- Then their weights are the values of the functions at each point, so that the Lagrange interpolant is:

$$P_n(x) = \sum_{k=0}^{n} L_k(x)\, f(x_k)$$

Lagrange polynomials use colocation at points

- Degree n Lagrange polynomials agree exactly with a function f(x) at n+1 distinct points, $(x_0, f(x_0)),\ (x_1, f(x_1)) \ldots (x_{n+1}, f(x_{n+1}))$
- First we construct the basis functions

$$L_k(x) = \prod_{i=0, i\neq k}^{n} \frac{(x - x_i)}{(x_k - x_i)}$$

Numerator - functions to be zero at all of the points other than x_k - denominator - function is normalised so it has value 1 at x_k

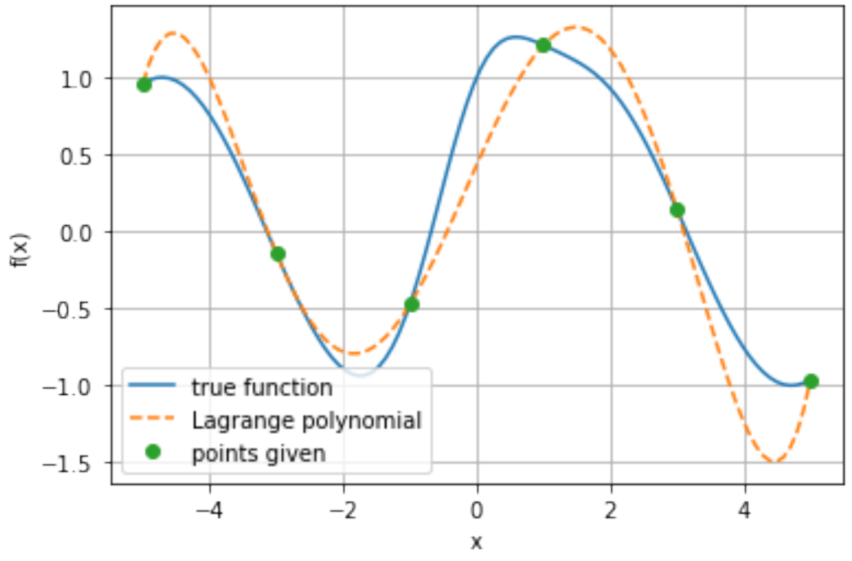
- Then their weights are the values of the functions at each point, so that the Lagrange interpolant is:

$$P_n(x) = \sum_{k=0}^{n} L_k(x)\, f(x_k)$$

Lagrange polynomials use colocation at points

- Can use a python function *scipy.interpolate.lagrange()* to construct using higher number of points

Is a higher number of points always better?

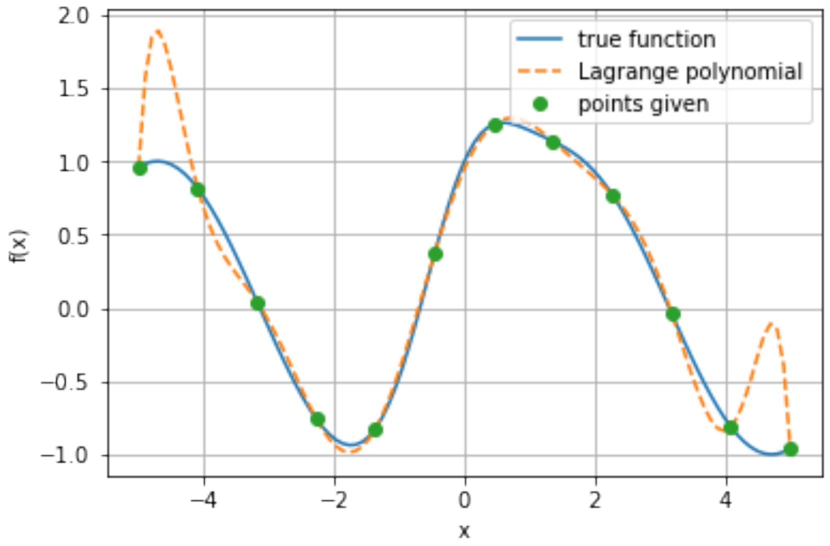


Lagrange polynomials use colocation at points

- More points improved the fit at the interior, but with regular intervals it tends to lead to spurious oscillations at the edges of the interval

  -> “Runge’s phenomenon”

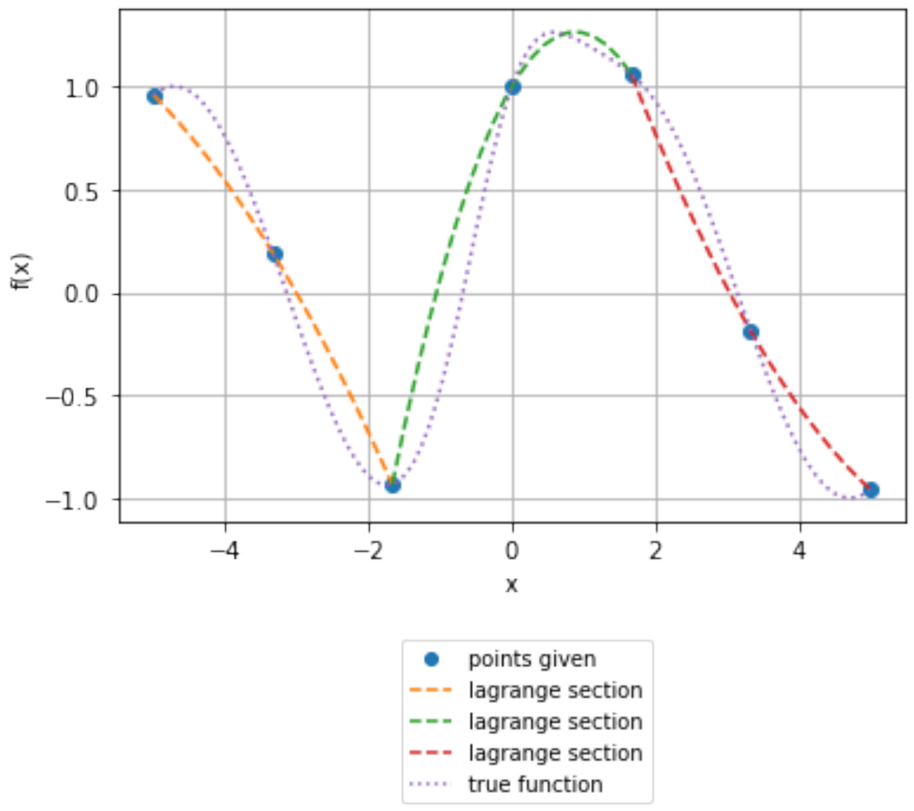
How can we do better?



Strategy 1: composite colocation

1. We could try to divide the interval up into smaller sections and fit lower order Lagrange polynomials to each part in turn - this is a **composite colocation** method.

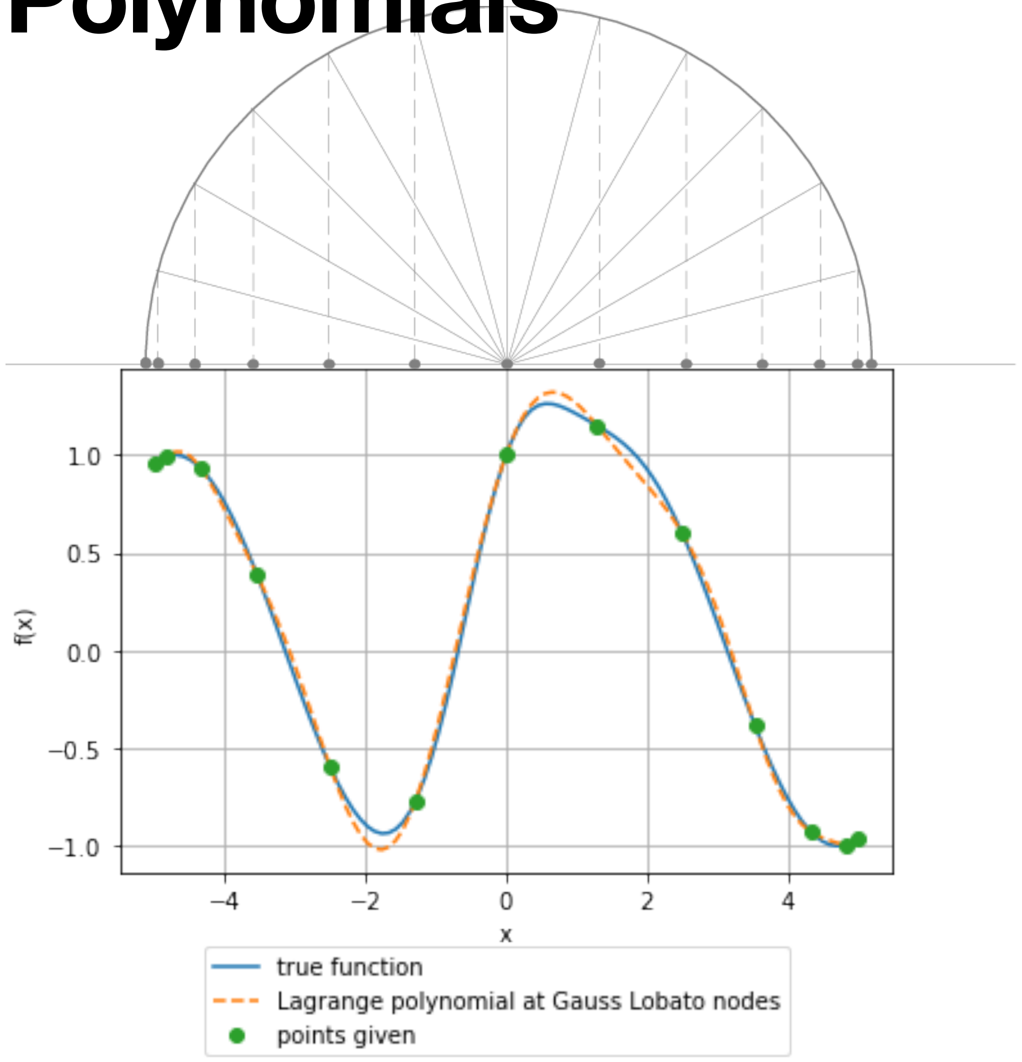
   This approach gives rise to the idea of ***finite differencing*** for finding derivatives.



Strategy 2 - Chebyshev Polynomials

2. If we locate the points not evenly, but at the zeros of the Chebyshev polynomials, we get an exponentially convergent fit and eliminate Runge's phenomenon

   This approach gives rise to the idea of ***pseudospectral*** methods for finding derivatives.

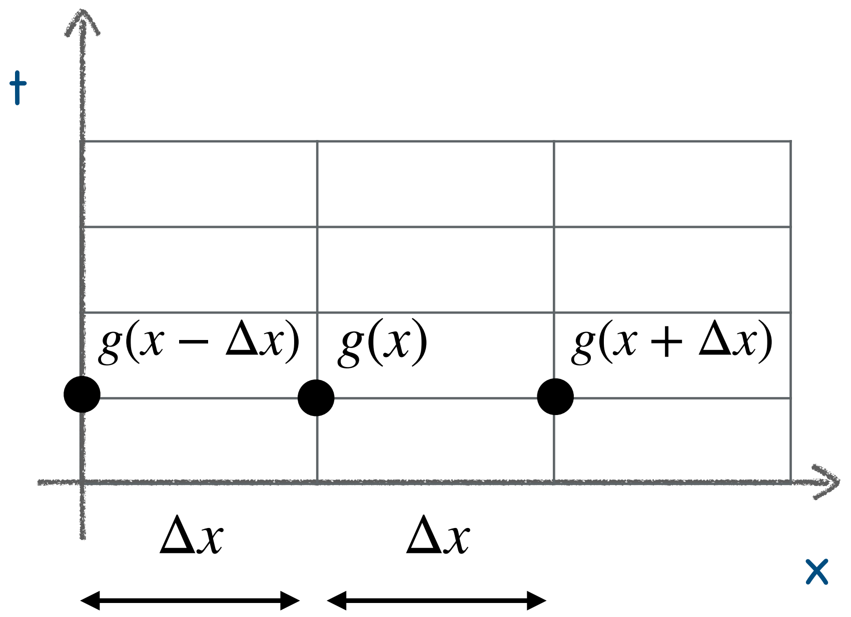


Plan for today

1. ~~Motivation - solution of the heat equation~~
2. ~~Revision of last week's interpolation content~~
3. Numerical differentiation - finite differencing
4. Numerical differentiation - pseudospectral methods
5. Application - solution of the heat equation

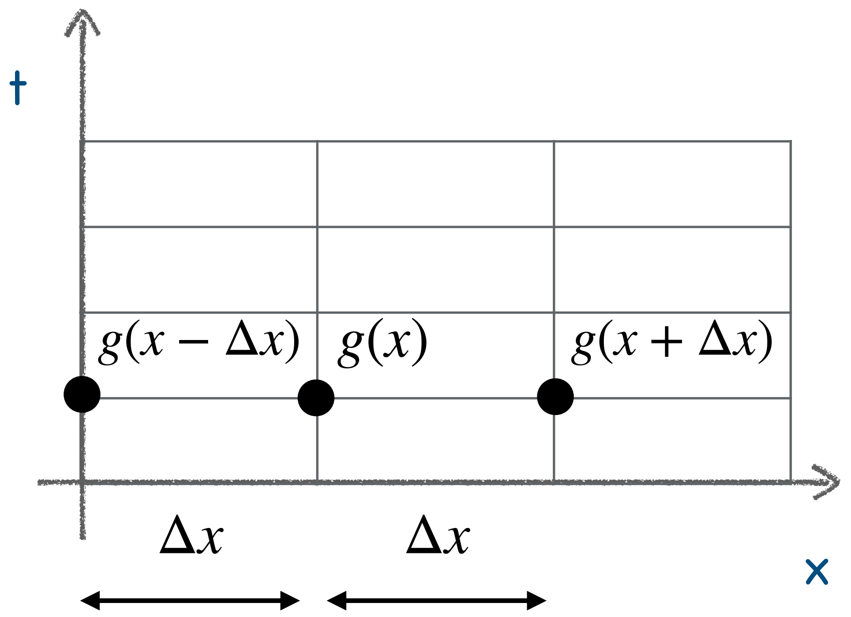
Numerical differentiation - finite differencing

How do I find the first derivative at the central point?



Numerical differentiation - finite differencing

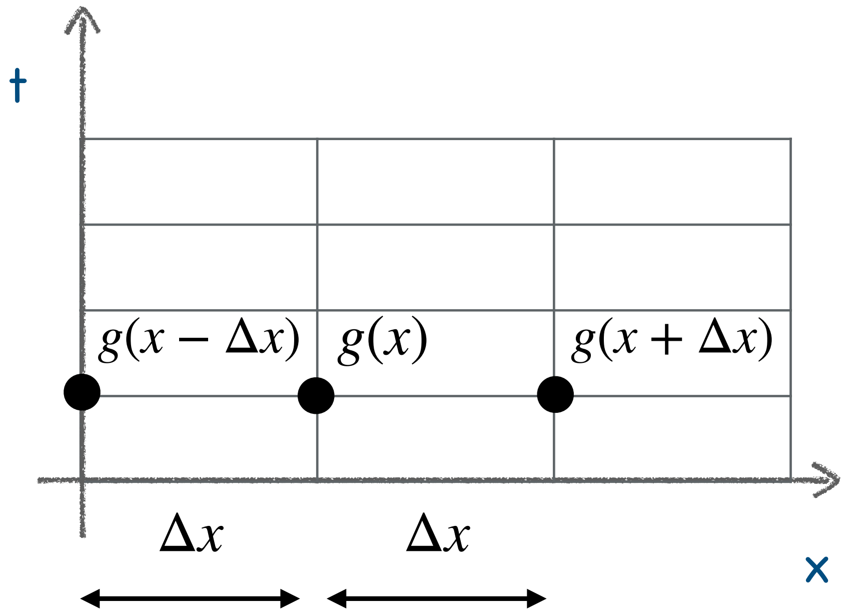
How do I find the first derivative at the central point?



$$\frac{\partial g}{\partial x} \approx \frac{g(x+\Delta x) - g(x-\Delta x)}{2\Delta x}$$

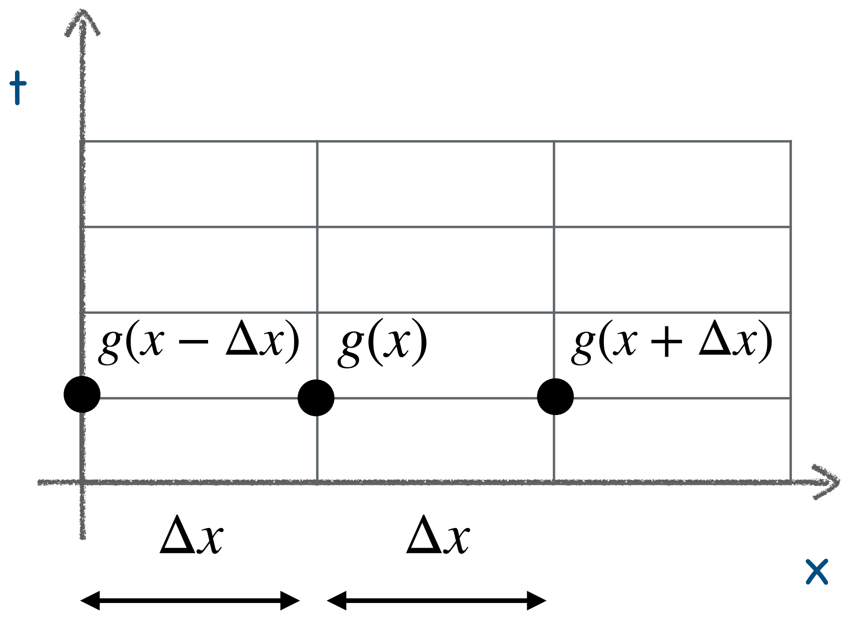
Numerical differentiation - finite differencing

How do I find the second derivative at the central point?



Numerical differentiation - finite differencing

How do I find the second derivative at the central point?



$$\frac{\partial^2 g}{\partial x^2} \approx \frac{\dfrac{g(x+\Delta x)-g(x)}{\Delta x} - \dfrac{g(x)-g(x-\Delta x)}{\Delta x}}{\Delta x}$$

$$\approx \frac{g(x+\Delta x) - 2g(x) + g(x-\Delta x)}{\Delta x^2}$$

Finite differencing - stencil representation

We can see ***finite differencing*** as the convolution of a stencil with the current state vector.

$\Delta x = 0.5$

Position x	0	0.5	1	1.5	2	2.5
Temperature T	0	1	3	2	1	0

$\uparrow$

-1	0	1

First derivative stencil

$$\frac{\partial g}{\partial x} \approx \frac{g(x+\Delta x) - g(x-\Delta x)}{2\Delta x}$$

dT/dx						

Finite differencing - stencil representation

We can see ***finite differencing*** as the convolution of a stencil with the current state vector.

Position x	0	0.5	1	1.5	2	2.5
Temperature T	0	1	3	2	1	0

$\uparrow$

-1	0	1

First derivative stencil

$$\frac{\partial g}{\partial x} \approx \frac{g(x+\Delta x) - g(x-\Delta x)}{2\Delta x}$$

dT/dx			**1**			

Finite differencing - stencil representation

We can see ***finite differencing*** as the convolution of a stencil with the current state vector.

Position x	0	0.5	1	1.5	2	2.5
Temperature T	0	1	3	2	1	0

	↑	
-1	0	1

First derivative stencil

$$\frac{\partial g}{\partial x} \approx \frac{g(x+\Delta x) - g(x-\Delta x)}{2\Delta x}$$

dT/dx			**1**			

Finite differencing - stencil representation

We can see ***finite differencing*** as the convolution of a stencil with the current state vector.

Position x	0	0.5	1	1.5	2	2.5
Temperature T	0	1	3	2	1	0

$\uparrow$

-1	0	1

First derivative stencil

$$\frac{\partial g}{\partial x} \approx \frac{g(x+\Delta x) - g(x-\Delta x)}{2\Delta x}$$

dT/dx		**3**	**1**			

Finite differencing - stencil representation

We can see ***finite differencing*** as the convolution of a stencil with the current state vector.

Position x	0	0.5	1	1.5	2	2.5
Temperature T	0	1	3	2	1	0

$\uparrow$

-1	0	1

dT/dx		**3**	**1**	**-2**		

Finite differencing - stencil representation

We can see ***finite differencing*** as the convolution of a stencil with the current state vector.

Position x	0	0.5	1	1.5	2	2.5
Temperature T	0	1	3	2	1	0

-1	0	1

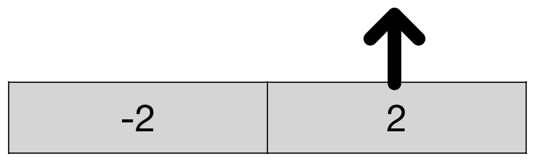
dT/dx		**3**	**1**	**-2**	**-2**	

What about the end points?

Finite differencing - stencil representation

We can see ***finite differencing*** as the convolution of a stencil with the current state vector.

Position x	0	0.5	1	1.5	2	2.5
Temperature T	0	1	3	2	1	0



Use one sided stencil - doesn't have to be centralised

dT/dx		**3**	**1**	**-2**	**-2**	**-2**

Finite differencing - stencil representation

We can see ***finite differencing*** as the convolution of a stencil with the current state vector.

Position x	0	0.5	1	1.5	2	2.5
Temperature T	0	1	3	2	1	0

*OR use a **boundary condition** -*
some knowledge about the function
- e.g. maybe its derivative goes to zero here

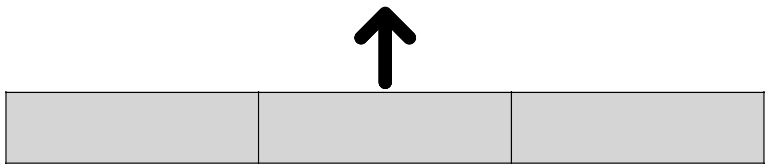
dT/dx		**3**	**1**	**-2**	**-2**	**0**

Finite differencing - stencil representation

We can see ***finite differencing*** as the convolution of a stencil with the current state vector.

$\Delta x = 0.5$

Position x	0	0.5	1	1.5	2	2.5
Temperature T	0	1	3	2	1	0



Second derivative stencil

$$\approx \frac{g(x+\Delta x) - 2g(x) + g(x-\Delta x)}{\Delta x^2}$$

d2T/dx2						

What is the second derivative stencil?

Finite differencing - stencil representation

We can see ***finite differencing*** as the convolution of a stencil with the current state vector.

$\Delta x = 0.5$

Position x	0	0.5	1	1.5	2	2.5
Temperature T	0	1	3	2	1	0

4	-8	4

Second derivative stencil

$$\approx \frac{g(x+\Delta x) - 2g(x) + g(x-\Delta x)}{\Delta x^2}$$

d2T/dx2			**-12**			

Finite differencing - matrix representation

We can also represent this convolution in matrix form:

Position x	0	0.5	1	1.5	2	2.5
Temperature T	0	1	3	2	1	0

DT/dx* = *Matrix D* • *T

$$
\begin{bmatrix} 2 \\ 3 \\ 1 \\ -2 \\ -2 \\ -2 \end{bmatrix}
=
\begin{bmatrix}
-2 & 2 & & & & \\
-1 & 0 & 1 & & & \\
 & -1 & 0 & 1 & & \\
 & & -1 & 0 & 1 & \\
 & & & -1 & 0 & 1 \\
 & & & & -2 & 2
\end{bmatrix}
\bullet
\begin{bmatrix} 0 \\ 1 \\ 3 \\ 2 \\ 1 \\ 0 \end{bmatrix}
$$

All blank entries zero

Finite differencing - relation to Lagrange polynomials

Using the Lagrange polynomials helps us to have general method for constructing the stencils.

This is important to answer the questions:

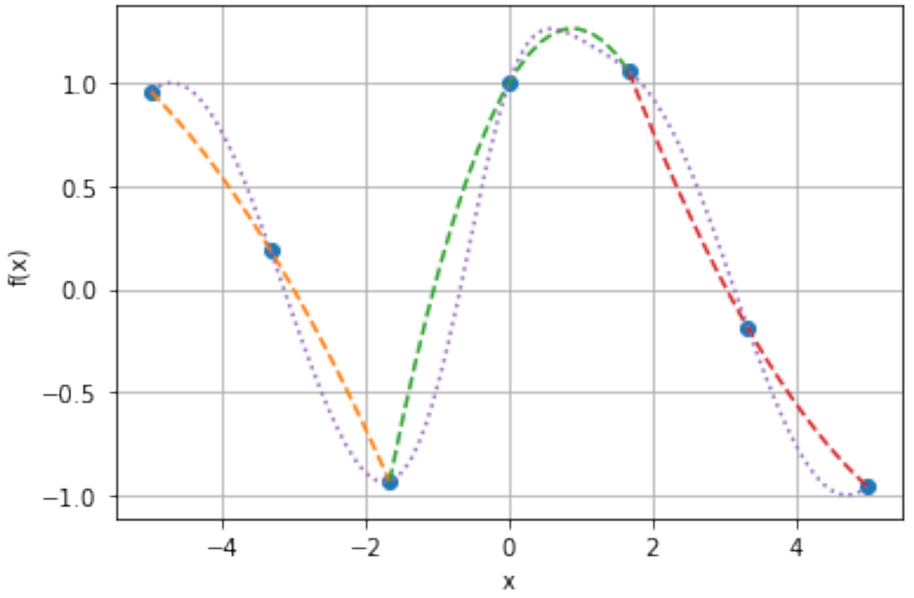
- What if I don’t have evenly spaced points?
- How do I know the order of accuracy of my method?
- How many points should I use?

Finite differencing - relation to Lagrange polynomials

We fit a polynomial of order N-1 to the N points we want to use in the stencil.

Recall that the order of the fit has an error related to the Nth derivative.

This function that fits all the points is just an order N polynomial, whose coefficients are combinations of the function values at each point and the points themselves e.g. for 3 points separated by a distance dx:



$$T = (-f_1(x - x_2)(dx - x + x_2) + 2f_2(dx - x + x_2)(dx + x - x_2) + f_3(x - x_2)(dx + x - x_2))/(2dx^2)$$

Finite differencing - relation to Lagrange polynomials

We can take the derivative of this polynomial

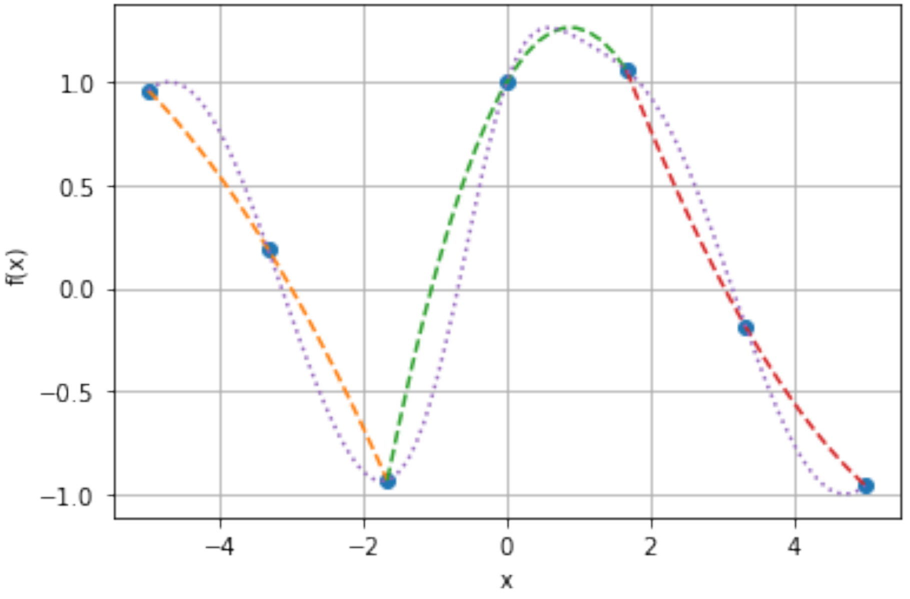
$$\frac{dT}{dx} = (-dx\, f_1/2 + dx\, f_3/2 + xf_1 - x_2 f_1 - 2xf_2 + 2f_2x_2 + xf_3 - x_2 f_3)/dx^2 + O(dx^2)$$

And evaluate it at x = x_2

$$\left.\frac{dT}{dx}\right|_{x=x_2} = \frac{(f_3 - f_1)}{2dx} + O(dx^2)$$

Order of error is N-1, coming from the derivative of the error:

$$E_{max} = \max\left[\left|\frac{f^{n+1}(\zeta)}{(n+1)!}\prod_{i=0}^{n}(x - x_i)\right|\right]$$

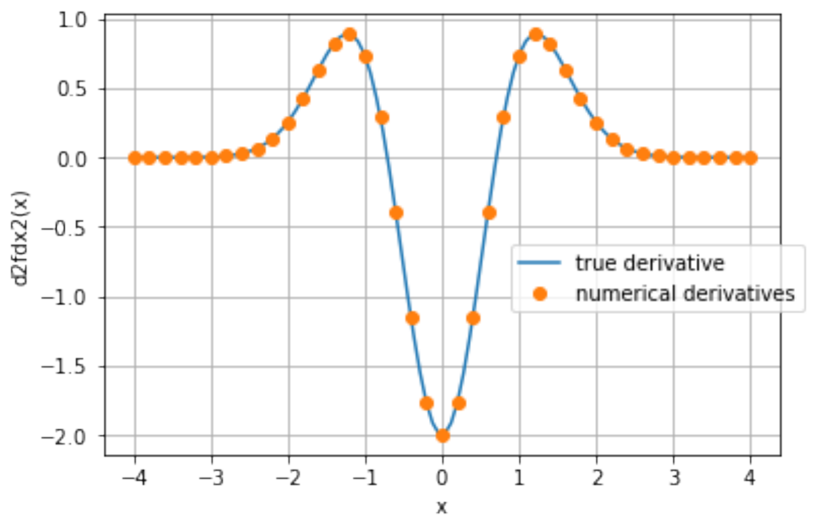


Finite differencing - relation to Lagrange polynomials

This will give us a stencil for the function values to apply at x2

$$\left.\frac{dT}{dx}\right|_{x=x_2} = \frac{(f_3 - f_1)}{2dx} + O(dx^2)$$

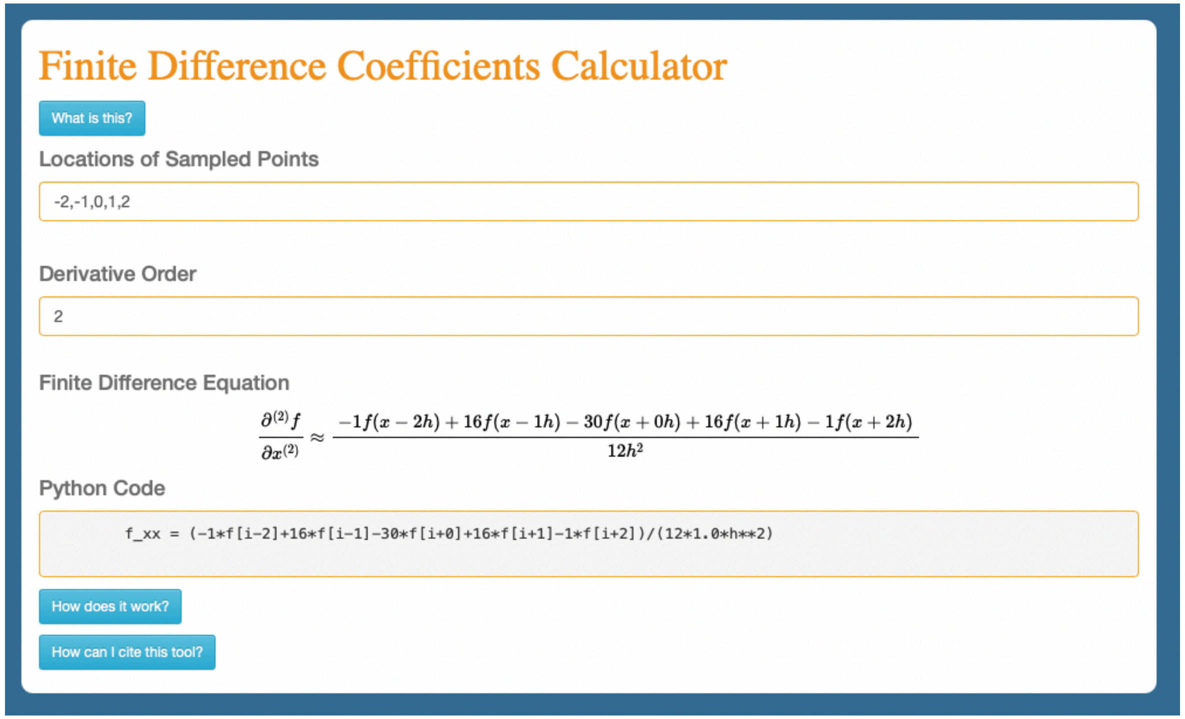
f_1	f_2	f_3
	$\uparrow$	
$-1/(2\ dx)$	0	$1/(2\ dx)$



Finite differencing - general number of points

The MIT finite difference calculator is a useful resource for working out the stencil for a general collection of **equally spaced** points, up to any derivative order.

For non equally spaced points thing get a lot more complicated, and may also depend on the position.



Finite Difference Coefficients Calculator

Locations of Sampled Points

-2,-1,0,1,2

Derivative Order

2

Finite Difference Equation

$$\frac{\partial^{(2)} f}{\partial x^{(2)}} \approx \frac{-1f(x-2h)+16f(x-1h)-30f(x+0h)+16f(x+1h)-1f(x+2h)}{12h^2}$$

Python Code

```
f_xx = (-1*f[i-2]+16*f[i-1]-30*f[i+0]+16*f[i+1]-1*f[i+2])/(12*1.0*h**2)
```

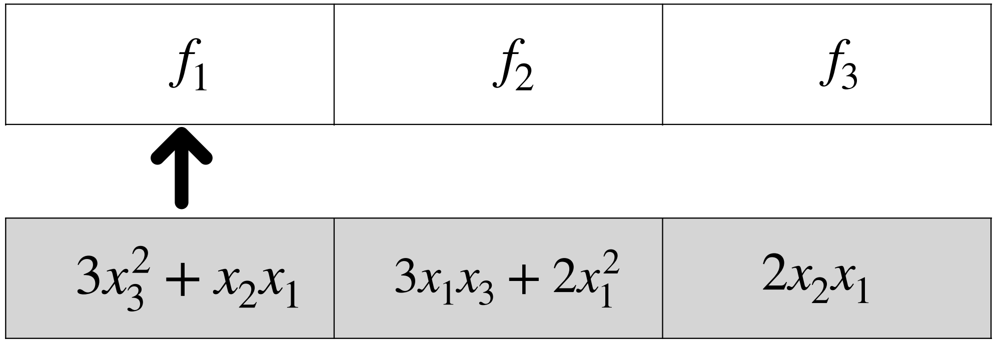
Plan for today

1. ~~Motivation - solution of the heat equation~~
2. ~~Revision of last week’s interpolation content~~
3. ~~Numerical differentiation - finite differencing~~
4. Numerical differentiation - pseudospectral methods
5. Application - solution of the heat equation

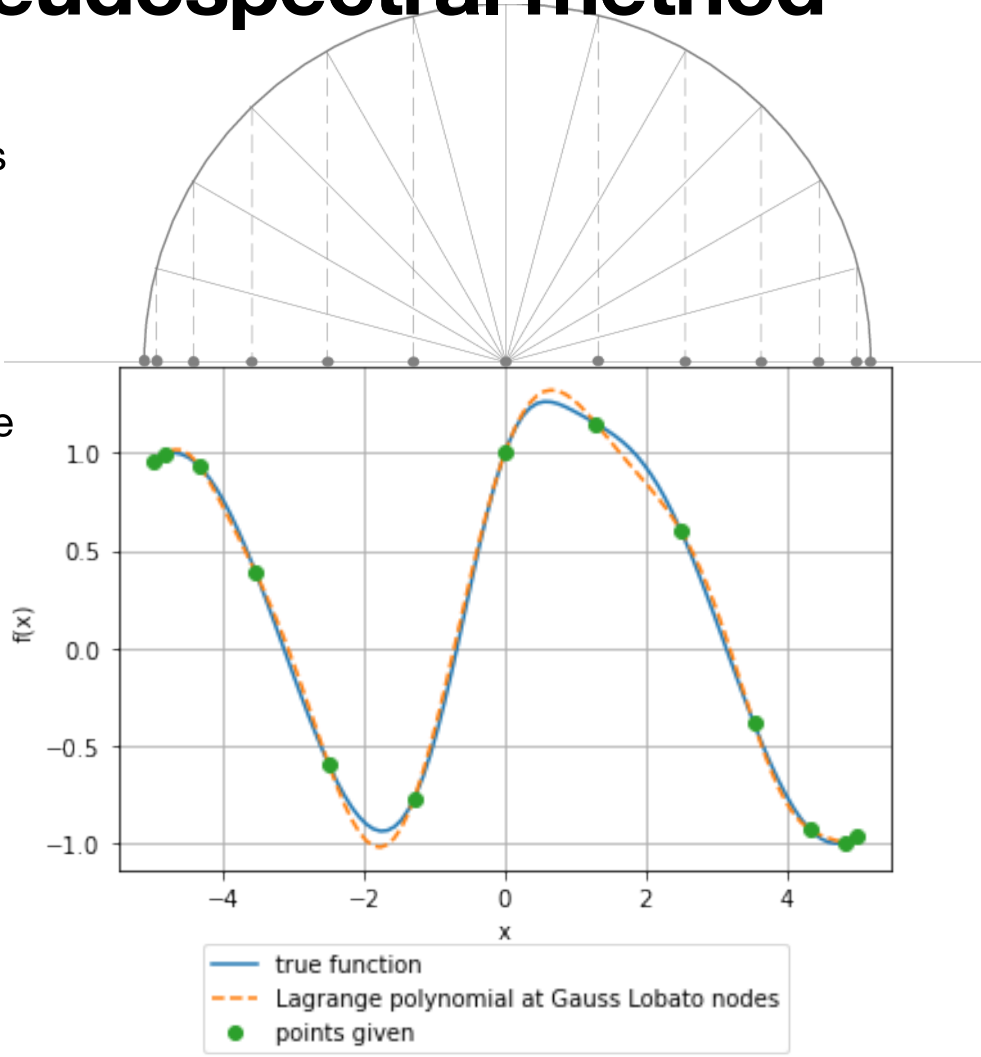
Numerical differentiation - pseudospectral method

We can apply the same Lagrange method to find the stencils for the Chebyshev polynomials used in the Pseudospectral method.

The complication is that the order of the polynomial is now N-1 where N is the number of points. Now all the values of the function at every point are used in the stencil, not just the neighbouring ones, and the grid spacing is not equal so the factors don't cancel nicely.



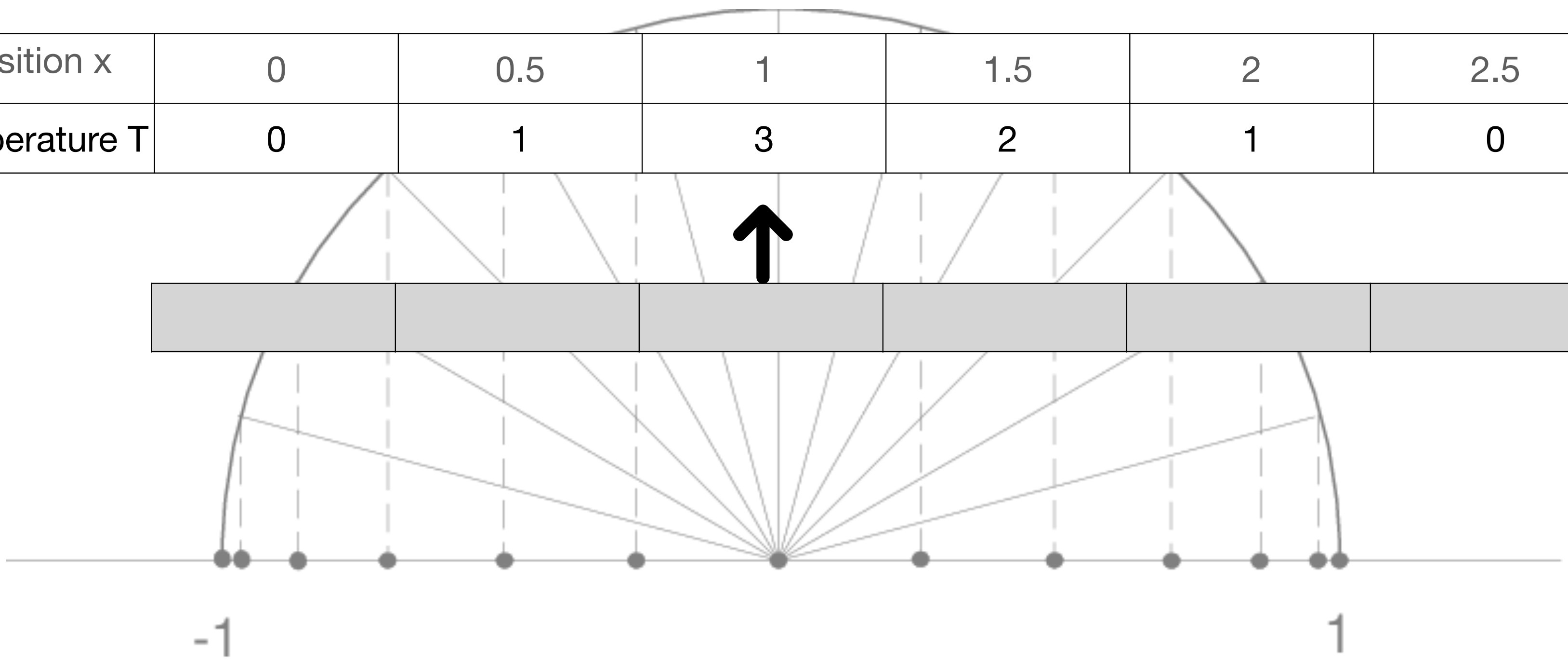
Made up complicated looking stencil



Pseudospectral method - stencil representation

Now the stencil includes all the points in the grid, and is a bit more complicated to work out, but the principle is the same!

Position x	0	0.5	1	1.5	2	2.5
Temperature T	0	1	3	2	1	0



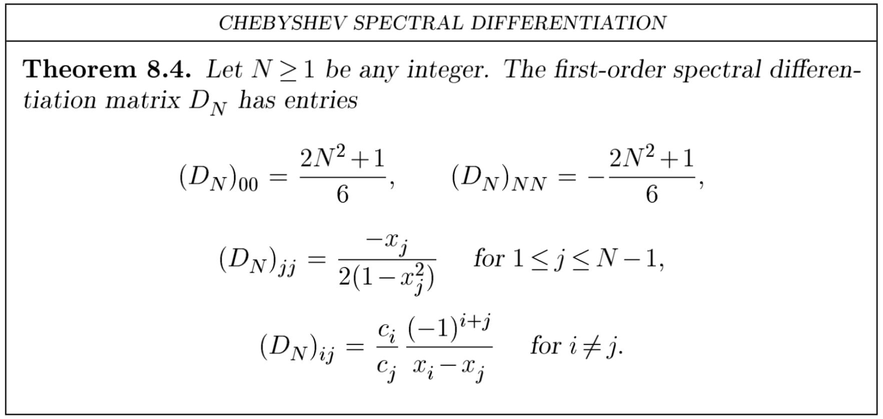
Pseudospectral methods - matrix representation

The simplest thing is to represent the derivatives in matrix form:

Position x	0	0.5	1	1.5	2	2.5
Temperature T	0	1	3	2	1	0

DT/dx* = *Matrix D* *T

$$\begin{pmatrix} 2 \\ 3 \\ 1 \\ -2 \\ -2 \\ -2 \end{pmatrix} = \text{Matrix } D \bullet \begin{pmatrix} 0 \\ 1 \\ 3 \\ 2 \\ 1 \\ 0 \end{pmatrix}$$



CHEBYSHEV SPECTRAL DIFFERENTIATION

Theorem 8.4. *Let $N \geq 1$ be any integer. The first-order spectral differentiation matrix D_N has entries*

$$(D_N)_{00} = \frac{2N^2+1}{6}, \qquad (D_N)_{NN} = -\frac{2N^2+1}{6},$$

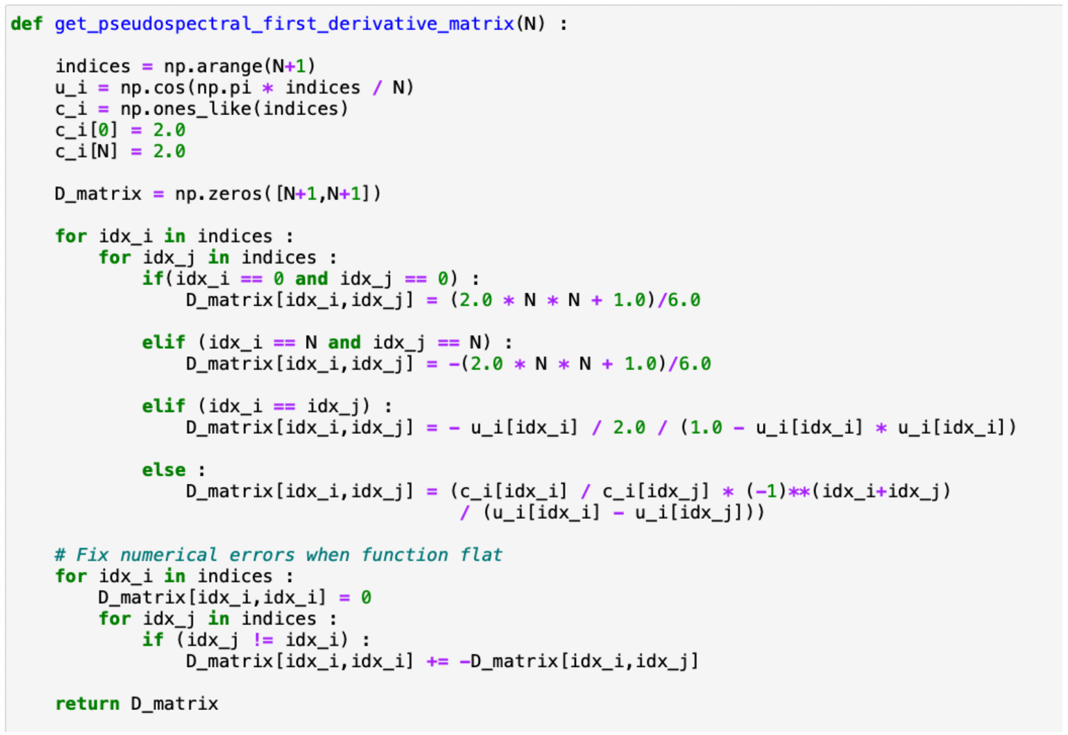
$$(D_N)_{jj} = \frac{-x_j}{2(1-x_j^2)} \qquad \text{for } 1 \leq j \leq N-1,$$

$$(D_N)_{ij} = \frac{c_i}{c_j}\frac{(-1)^{i+j}}{x_i - x_j} \qquad \text{for } i \neq j.$$

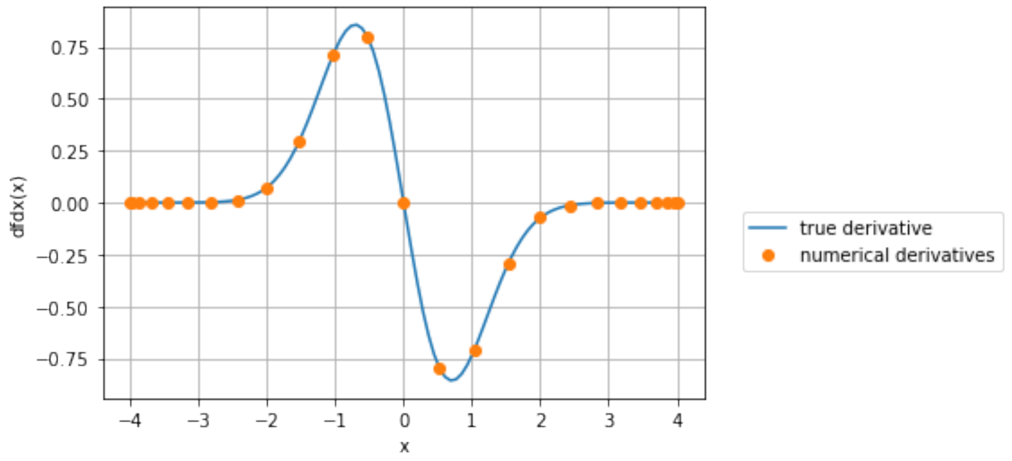
TREFETHEN 1994

Pseudospectral methods - matrix representation

In the tutorial (and the coursework) this matrix will be provided to you, but you need to know how to use it.

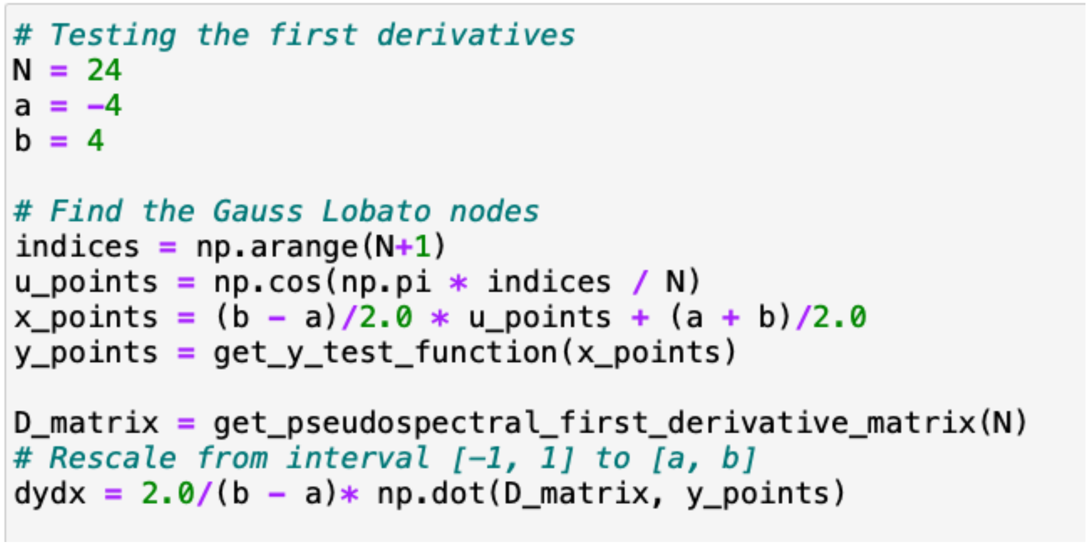


```python
def get_pseudospectral_first_derivative_matrix(N) :

    indices = np.arange(N+1)
    u_i = np.cos(np.pi * indices / N)
    c_i = np.ones_like(indices)
    c_i[0] = 2.0
    c_i[N] = 2.0

    D_matrix = np.zeros([N+1,N+1])

    for idx_i in indices :
        for idx_j in indices :
            if(idx_i == 0 and idx_j == 0) :
                D_matrix[idx_i,idx_j] = (2.0 * N * N + 1.0)/6.0

            elif (idx_i == N and idx_j == N) :
                D_matrix[idx_i,idx_j] = -(2.0 * N * N + 1.0)/6.0

            elif (idx_i == idx_j) :
                D_matrix[idx_i,idx_j] = - u_i[idx_i] / 2.0 / (1.0 - u_i[idx_i] * u_i[idx_i])

            else :
                D_matrix[idx_i,idx_j] = (c_i[idx_i] / c_i[idx_j] * (-1)**(idx_i+idx_j)
                                        / (u_i[idx_i] - u_i[idx_j]))

    # Fix numerical errors when function flat
    for idx_i in indices :
        D_matrix[idx_i,idx_i] = 0
        for idx_j in indices :
            if (idx_j != idx_i) :
                D_matrix[idx_i,idx_i] += -D_matrix[idx_i,idx_j]

    return D_matrix
```

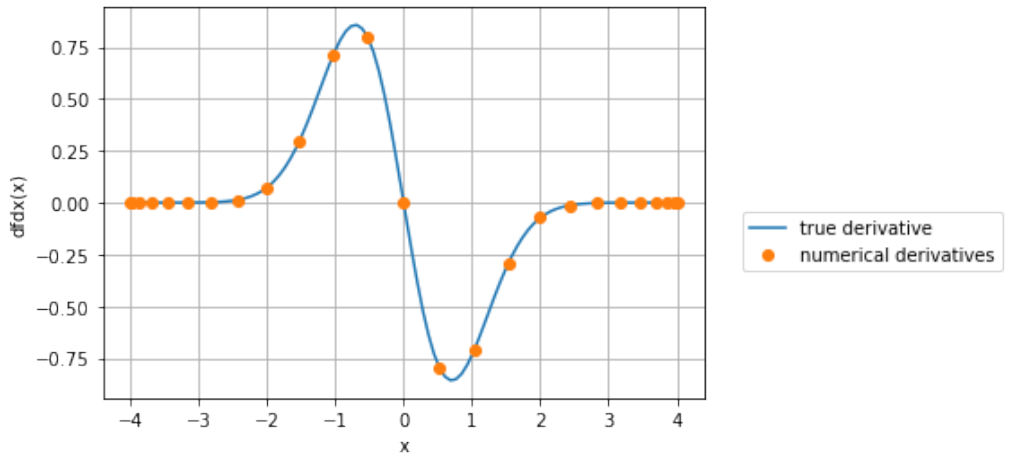


Pseudospectral methods - matrix representation

The main thing to remember is that the matrix is specific to the number of points used. It also needs to be **rescaled** if the interval is not [-1,1].



```
# Testing the first derivatives
N = 24
a = -4
b = 4

# Find the Gauss Lobato nodes
indices = np.arange(N+1)
u_points = np.cos(np.pi * indices / N)
x_points = (b - a)/2.0 * u_points + (a + b)/2.0
y_points = get_y_test_function(x_points)

D_matrix = get_pseudospectral_first_derivative_matrix(N)
# Rescale from interval [-1, 1] to [a, b]
dydx = 2.0/(b - a)* np.dot(D_matrix, y_points)
```



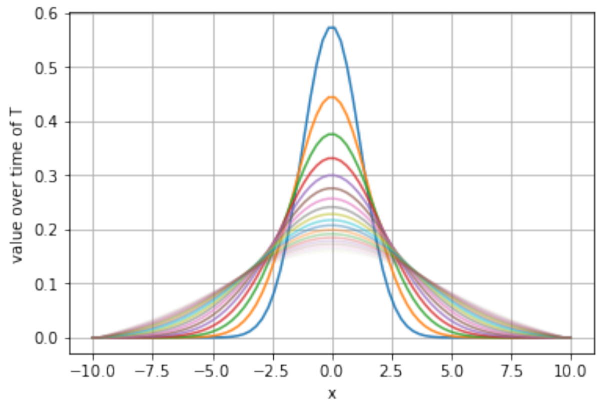
Plan for today

1. ~~Motivation - solution of the heat equation~~
2. ~~Revision of last week's interpolation content~~
3. ~~Numerical differentiation - finite differencing~~
4. ~~Numerical differentiation - pseudospectral methods~~
5. Application - solution of the heat equation

Application: solving the heat equation

- In the tutorial you will solve the heat equation using solve_ivp()

$$\frac{\partial T}{\partial t} = \alpha \frac{\partial^2 T}{\partial x^2}$$



```
def calculate_dydt(self, t, current_state) :

    # Just for readability
    dTdt = np.zeros_like(current_state)

    # Now actually work out the time derivatives
    dTdt[:] =  self.alpha * np.dot(self.D2_matrix, current_state)

    # Zero the derivatives at the end for stability
    # (especially important in the pseudospectral method)
    dTdt[0] = 0.0
    dTdt[1] = 0.0
    dTdt[self.N_grid-1] = 0.0
    dTdt[self.N_grid-2] = 0.0

    return dTdt
```

What is going on here?

Application: solving the heat equation

Here we are using the matrix representation to calculate the time derivative

Position x	0	0.5	1	1.5	2	2.5
Temperature T	0	1	3	2	1	0

D2Tdx2 **=** ***Matrix D^2*** ***T***

$$\begin{bmatrix}2\\3\\1\\-2\\-2\\-2\end{bmatrix} = \begin{bmatrix}X & X & & & & \\ X & X & X & & & \\ & X & X & X & & \\ & & X & X & X & \\ & & & X & X & X \\ & & & & X & X\end{bmatrix} \bullet \begin{bmatrix}0\\1\\3\\2\\1\\0\end{bmatrix}$$

Plan for today

1. Motivation - solution of the heat equation
2. Revision of last week’s interpolation content
3. Numerical differentiation - finite differencing
4. Numerical differentiation - pseudospectral methods
5. Application - solution of the heat equation