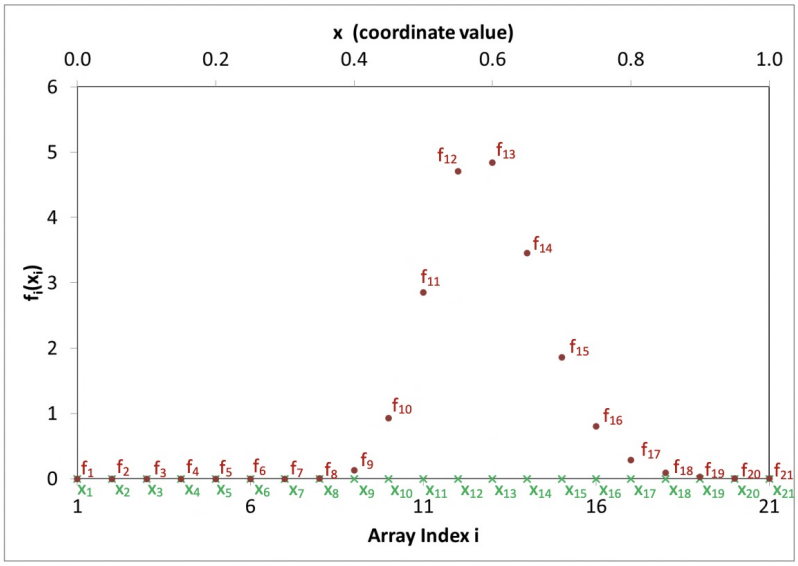
Numerical Integration and Differentiation Exercise

This exercise builds on the process of creating a coordinate axis in a function. In this exercise, simple methods for numerical integration and differentiation will be applied to our log-normal distribution and our coordinate axis function will be expanded to return an array of integration weights of length N, an NxN matrix of first derivative weights, and an NxN matrix of second derivative weights along with the coordinate axis points (vector of length N).

Task 1: Numerical integration and differentiation

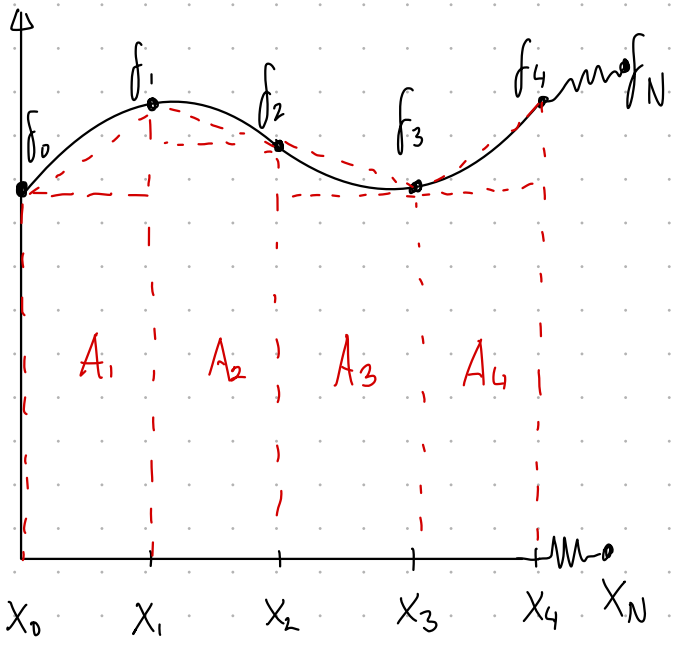
In the last exercise, we created function that took an integer N as input and returned an array of length N containing N evenly-spaced, consecutive numbers between 0 and 1 that we used to plot a log-normal distribution.



The plot of the distribution employs an array **x** [x_1, x_2, x_3, …, x_{N-2}, x_{N-1}, x_N] as an independent coordinate axis and an array **f** [f_1, f_2, f_3, …, f_{N-2}, f_{N-1}, f_N] as a dependent coordinate axis. Using notation the plot as a guide, write out by hand the formulas for the following:

Task 1a: Numerical integrals can be determined using rectangles or trapezoids. Use rectangles or trapezoids to show how you would calculate the integral of f(x) on the interval [0 1]. You need to show the first and last three terms at minimum.

Using trapezoids:



Assuming $f(x_i) = f_i$:

The trapezoid A_1 has area

$$A_1 = f_0(x_1 - x_0) + \frac{1}{2}(f_1 - f_0)(x_1 - x_0)$$

$$A_1 = \frac{1}{2}(f_0 + f_1)(x_1 - x_0)$$

Or in general:

$$A_i = \frac{1}{2}(f_{i-1} + f_i)(x_i - x_{i-1})$$

Approximating the integral using the trapezoids:

$$\int_0^1 f(x)dx \approx \frac{1}{2}\left[(f_0+f_1)(X_1-X_0)+(f_1+f_2)(X_2-X_1)+(f_2+f_3)(X_3-X_2)+\dots\right.$$

$$\left.+(f_{N-3}+f_{N-2})(X_{N-2}-X_{N-3})+(f_{N-2}+f_{N-1})(X_{N-1}-X_{N-2})+(f_{N-1}+f_N)(X_N-X_{N-1})\right]$$

Where $X_0 = 0$ and $X_N = 1$ for the integral from 0 to 1

The expression can also be written as:

$$\int_0^1 f(x)dx \approx \frac{1}{2}\sum_{i=1}^{N}(f_i+f_{i-1})(X_i-X_{i-1})$$

I just realized that the array in the problem begins at X_1, so my answer should be "shifted"

$$\int_0^1 f(x) \approx \frac{1}{2}\sum_{i=1}^{N-1}(f_i+f_{i+1})(X_{i+1}-X_i)$$

Task 1b: Numerical first derivatives can be calculated using secant lines. You can do either forward or backwards differentiation as follows:

$\left.\frac{df}{dx}\right|_{x_i} = \frac{f_{i+1}-f_i}{x_{i+1}-x_i}$ (forward) $\left.\frac{df}{dx}\right|_{x_i} = \frac{f_i-f_{i-1}}{x_i-x_{i-1}}$ (backward)

Use the formulas above to show how you would calculate the first derivate of f(x) at the first three points (x_1, x_2, x_3) and at the last 3 points (x_{N-2}, x_{N-1}, x_N).

Using the forward scheme, we get:

$$\left.\frac{df}{dx}\right|_{X_1} = \frac{f_2-f_1}{X_2-X_1}, \quad \left.\frac{df}{dx}\right|_{X_2} = \frac{f_3-f_2}{X_3-X_2}, \quad \left.\frac{df}{dx}\right|_{X_3} = \frac{f_4-f_3}{X_4-X_3},$$

$$\dots \left.\frac{df}{dx}\right|_{X_{N-2}} = \frac{f_{N-1}-f_{N-2}}{X_{N-1}-X_{N-2}}, \quad \left.\frac{df}{dx}\right|_{X_{N-1}} = \frac{f_N-f_{N-1}}{X_N-X_{N-1}}$$

For the final node, we can't use the forward scheme, as we do not have access to the N+1 node, therefore, we use the backwards scheme

$$\left.\frac{df}{dx}\right|_{X_N} = \frac{f_N-f_{N-1}}{X_N-X_{N-1}}$$

Task 1c: Numerical second derivatives can be calculated using secant lines as well. You can do either forward, central, or backwards differentiation as follows:

$\left.\frac{d^2f}{dx^2}\right|_{x_i} = \frac{f_{i+2}-2f_{i+1}+f_i}{(x_{i+1}-x_i)^2}$ (forward); $\left.\frac{d^2f}{dx^2}\right|_{x_i} = \frac{f_{i+1}-2f_i+f_{i-1}}{(x_{i+1}-x_i)^2}$ (central); $\left.\frac{d^2f}{dx^2}\right|_{x_i} = \frac{f_{i-2}-2f_{i-1}+f_i}{(x_i-x_{i-1})^2}$ (backward)

Use the formulas above to show how you would calculate the second derivate of f(x) at the first three points (x_1, x_2, x_3) and at the last 3 points (x_{N-2}, x_{N-1}, x_N).

For the first point, we use forward, and for the last, backward, everywhere else, we can use the central scheme:

$$\left.\frac{d^2f}{dx^2}\right|_{x_1} = \frac{f_3-2f_2+f_1}{(x_2-x_1)^2}$$ Forward

$$\left.\frac{d^2f}{dx^2}\right|_{x_2} = \frac{f_3-2f_2+f_1}{(x_3-x_2)^2}, \quad \left.\frac{d^2f}{dx^2}\right|_{x_3} = \frac{f_4-2f_3+f_2}{(x_4-x_3)^2}$$

$$\left.\frac{d^2f}{dx^2}\right|_{x_{N-2}} = \frac{f_{N-1}-2f_{N-2}+f_{N-3}}{(x_{N-1}-x_{N-2})^2}, \quad \left.\frac{d^2f}{dx^2}\right|_{x_{N-1}} = \frac{f_N-2f_{N-1}+f_{N-2}}{(x_N-x_{N-1})^2}$$ Central

$$\left.\frac{d^2f}{dx^2}\right|_{x_N} = \frac{f_{N-2}-2f_{N-1}+f_N}{(x_N-x_{N-1})^2}$$ Backwards

Task 2: Vector and matrix representations of numerical integration and differentiation.

Task 2.1: Using your pattern recognition skills, rearrange the numerical integral equation from task 1.1 to be a product of the array **f** [f_1, f_2, f_3, …, f_{N-2}, f_{N-1}, f_N] and an array of integration weights **w** [w_1, w_2, w_3, …, w_{N-2}, w_{N-1}, w_N] such that the following holds:

$$\int_0^1 f(x)dx \approx \boldsymbol{w}\cdot\boldsymbol{f}$$

In other words, write the elements of the array **w**:

$$\int_0^1 f(x)dx \approx [w_1 \quad w_2 \quad w_3,\ldots,w_{N-2} \quad w_{N-1} \quad w_N]\cdot[f_1 \quad f_2 \quad f_3,\ldots,f_{N-2} \quad f_{N-1} \quad f_N]$$

Writing out some of the terms of the result from 1.1, an assuming the nodes to be evenly distributed along the x-axis $\Rightarrow x_i - x_{i-1} = \Delta x$

$$\int_0^1 f(x)dx \approx \frac{1}{2}\left[(f_1+f_2)\Delta x + (f_2+f_3)\Delta x + (f_3+f_4)\Delta x + \cdots (f_{N-3}+f_{N-2})\Delta x + (f_{N-2}+f_{N-1})\Delta x + (f_{N-1}+f_N)\Delta x\right]$$

$$= \frac{\Delta x}{2}f_1 + \Delta x f_2 + \Delta x f_3 + \ldots + \Delta x f_{N-2} + \Delta x f_{N-1} + \frac{\Delta x}{2}f_N$$

$$= \left[\frac{\Delta x}{2}, \Delta x, \Delta x, \ldots, \Delta x, \Delta x, \frac{\Delta x}{2}\right]\cdot[f_1, f_2, f_3, \ldots, f_{N-2}, f_{N-1}, f_N]$$

$$\Rightarrow w = \left[\frac{\Delta x}{2}, \Delta x, \Delta x, \ldots, \Delta x, \Delta x, \frac{\Delta x}{2}\right]$$

Task 2.2: Using your pattern recognition skills, rearrange the numerical first derivative equations from task 1.2 to be a product of the N-dimensional array **f** and an NxN matrix of differentiation weights **A** such that the following holds:

$$\frac{d}{dx}\boldsymbol{f} \approx \boldsymbol{A} \cdot \boldsymbol{f}$$

In other words, write the elements of the matrix **A** (fill in the first three rows and last three rows at minimum).

Using the result from 1b),

$$\left.\frac{df}{dx}\right|_{x_1} = \frac{f_2 - f_1}{x_2 - x_1}, \quad \left.\frac{df}{dx}\right|_{x_2} = \frac{f_3 - f_2}{x_3 - x_2}, \ldots$$ and backwards in the last node:

$$\left.\frac{df}{dx}\right|_{x_N} = \frac{f_N - f_{N-1}}{x_N - x_{N-1}}$$ assuming evenly distributed nodes: $x_i - x_{i-1} = \Delta x$, then, we get:

Writing this system of equations on matrix for, we get:

$$\frac{d}{dx}f = \frac{1}{\Delta x}\begin{bmatrix} -1 & 1 & 0 & 0 & \cdots & \cdots & \cdots \\ 0 & -1 & 1 & 0 & \cdots & \cdots & \cdots \\ 0 & 0 & -1 & 1 & \cdots & \cdots & \cdots \\ \vdots & \vdots & \vdots & \ddots & & & \\ \cdots & \cdots & \cdots & 0 & -1 & 1 & 0 \\ \cdots & \cdots & \cdots & 0 & 0 & -1 & 1 \\ \cdots & \cdots & \cdots & 0 & 0 & -1 & 1 \end{bmatrix} \cdot \begin{bmatrix} f_1 \\ f_2 \\ f_3 \\ \vdots \\ f_{N-2} \\ f_{N-1} \\ f_N \end{bmatrix}$$

Then, we see that:

$$A = \frac{1}{\Delta x}\begin{bmatrix} -1 & 1 & 0 & 0 & \cdots & \cdots & \cdots \\ 0 & -1 & 1 & 0 & \cdots & \cdots & \cdots \\ 0 & 0 & -1 & 1 & \cdots & \cdots & \cdots \\ \vdots & \vdots & \vdots & \ddots & & & \\ \cdots & \cdots & \cdots & 0 & -1 & 1 & 0 \\ \cdots & \cdots & \cdots & 0 & 0 & -1 & 1 \\ \cdots & \cdots & \cdots & 0 & 0 & -1 & 1 \end{bmatrix}$$

Task 2.3: Using your pattern recognition skills, rearrange the numerical second derivative equations from task 1.3 to be a product of the N-dimensional array **f** and an NxN matrix of differentiation weights **B** such that the following holds:

$$\frac{d^2}{dx^2}\boldsymbol{f} \approx \boldsymbol{B} \cdot \boldsymbol{f}$$

In other words, write the elements of the matrix **B** (fill in the first three rows and last three rows at minimum).

Again, assuming that $x_i - x_{i-1} = \Delta x \ \forall \ i$:

$$\left.\frac{d^2f}{d^2x}\right|_{x_1} = \frac{f_3 - 2f_2 + f_1}{\Delta x^2}, \quad \left.\frac{d^2f}{d^2x}\right|_{x_2} = \frac{f_3 - 2f_2 + f_1}{\Delta x^2}, \quad \left.\frac{d^2f}{d^2x}\right|_{x_3} = \frac{f_4 - 2f_3 + f_2}{\Delta x^2}$$

$$\left.\frac{d^2f}{dx^2}\right|_{x_{N-2}} = \frac{f_{N-1} - 2f_{N-2} + f_{N-3}}{\Delta x^2}, \quad \left.\frac{d^2f}{dx}\right|_{x_{N-1}} = \frac{f_N - 2f_{N-1} + f_{N-2}}{\Delta x^2}, \quad \left.\frac{d^2f}{d^2x}\right|_{x_N} = \frac{f_{N-2} - 2f_{N-1} + f_N}{\Delta x^2}$$

Writing this system on matrix form:

$$\Rightarrow \frac{d^2}{d^2x}f \approx \frac{1}{\Delta x^2} \cdot \begin{bmatrix} 1 & -2 & 1 & 0 & 0 & \cdots & & & & \\ 1 & -2 & 1 & 0 & 0 & \cdots & & & & \\ 0 & 1 & -2 & 1 & 0 & \cdots & & & & \\ 0 & 0 & 1 & -2 & 1 & \cdots & & & & \\ \vdots & \vdots & \vdots & \vdots & \ddots & & & & & \\ & & & & & 1 & -2 & 1 & 0 & 0 \\ & & & & & 0 & 1 & -2 & 1 & 0 \\ & & & & & 0 & 0 & 1 & -2 & 1 \\ & & & & & 0 & 0 & 1 & -2 & 1 \end{bmatrix} \cdot \begin{bmatrix} f_1 \\ f_2 \\ f_3 \\ f_4 \\ \vdots \\ f_{N-3} \\ f_{N-2} \\ f_{N-1} \\ f_N \end{bmatrix}$$

$$\Rightarrow B = \frac{1}{\Delta x^2} \cdot \begin{bmatrix} 1 & -2 & 1 & 0 & 0 & \cdots & & & & \\ 1 & -2 & 1 & 0 & 0 & \cdots & & & & \\ 0 & 1 & -2 & 1 & 0 & \cdots & & & & \\ 0 & 0 & 1 & -2 & 1 & \cdots & & & & \\ \vdots & \vdots & \vdots & \vdots & \ddots & & & & & \\ & & & & & 1 & -2 & 1 & 0 & 0 \\ & & & & & 0 & 1 & -2 & 1 & 0 \\ & & & & & 0 & 0 & 1 & -2 & 1 \\ & & & & & 0 & 0 & 1 & -2 & 1 \end{bmatrix}$$

Task 3: Create grid generating Matlab function

Create a Matlab function called "FiniteDifferenceGrid" to take in an integer N as input and returns the (a) an N-dimensional vector **x** of equally spaced grid points between 0 and 1, (b) an N-dimensional vector **w** of integration weights (task 2.1), (c) an NxN-dimensional matrix **A** of first derivative weights (task 2.2), and (d) an NxN-dimensional matrix **B** of second derivative weights (task 2.3). The function call should look as follows:

[x, w, A, B] = FiniteDifferenceGrid (N)

a) This will be the same as in the previous exercise:

```
function [x, w, A, B] = FiniteDifferenceGrid (N)

% The distance between nodes will be used multiple times.
dx = 1/(N-1);

% a) Creating equally space vector
x = linspace(0,1,N);
```

b)

```
%b) Creating the weight array
w = dx * ones(1, N);
% Changing the first and last elements:
w(1) = dx/2;
w(end) = dx/2;
```

c)

```
% c) Creating the matrix for the 1st derivative
A = zeros(N,N);
A(1:N+1:end) = -1;
A(N+1:N+1:end) = 1;
% Fixing the end node:
A(end, end) = 1;
A(end, end-1) = -1;
% Multiplying by 1/dx:
A = 1/dx * A;
```

d)

```
% d) Creating the matrix for the 2nd derivative
B = zeros(N,N);
B(1:N+1:end) = -2;
B(2:N+1:end) = 1;
B(N+1:N+1:end) = 1;
% Fix the boundaries
B(1,1) = 1;
B(1,2) = -2;
B(1,3) = 1;
B(end, end) = 1;
B(end, end-1) = -2;
B(end, end-2) = 1;
% Multiply with 1/dx^2
B = 1/dx^2 * B;
```

The results for N=5:

$$X = \begin{matrix} 0 & 0.2500 & 0.5000 & 0.7500 & 1.0000 \end{matrix}$$

$$W = \begin{matrix} 0.1250 & 0.2500 & 0.2500 & 0.2500 & 0.1250 \end{matrix}$$

$$A = \begin{bmatrix} -4 & 4 & 0 & 0 & 0 \\ 0 & -4 & 4 & 0 & 0 \\ 0 & 0 & -4 & 4 & 0 \\ 0 & 0 & 0 & -4 & 4 \\ 0 & 0 & 0 & -4 & 4 \end{bmatrix}$$

$$B = \begin{bmatrix} 16 & -32 & 16 & 0 & 0 \\ 16 & -32 & 16 & 0 & 0 \\ 0 & 16 & -32 & 16 & 0 \\ 0 & 0 & 16 & -32 & 16 \\ 0 & 0 & 16 & -32 & 16 \end{bmatrix}$$

Task 4: Generate a log-normal distribution function and calculate its integral, first, and second derivatives.

Using the coordinate axis array **x** generated in the previous task, use a for-loop to create an array that stores the value of a log-normal distribution function defined as follows:

$$f_i = \frac{1}{x_i \sigma \sqrt{2\pi}} e^{-\frac{(\ln x_i - \mu)^2}{2\sigma^2}}$$

In the above equation, x_i is the ith element of the coordinate array generated in task 4, f_i is the ith element of the array containing the value of a log-normal distribution at the corresponding point x_i, and μ and σ are parameters of the distribution; use values of -0.530 and 0.136 for μ and σ, respectively.

Use your weight array and matrices (**w**, **A**, and **B**) to calculate the integral of f and its first and second derivatives. Plot the first and second derivatives of the function f that you calculated numerically. Use different colors for each curve and use markers on the curves to show where your grid points are located.

At $f(0)$, the function is undefined, so we must find a value for it:

$$\lim_{x \to 0} f(x) = \lim_{x \to 0} \left(e^{-\frac{(\ln x - \mu)^2}{2\sigma^2}} \cdot \frac{1}{x \cdot \sigma \cdot \sqrt{2\pi}} \right) = 0$$

$\Rightarrow$ We set $f(0) = 0$

For N = 75:

Using a for-loop to get the function values:

```
% Task 4

% Defining the given parameters
my = -0.530;
sigma = 0.136;

%Creating an empty vector
f = zeros(N,1);

% Filling the vector with the log-normal values
for i = 1:N
    f(i,1) = exp(-(log(x(i)) - my)^2/(2*sigma^2))/...
        (x(i)*sigma*sqrt(2*pi));
end
% Need to correct the first value, as it currently is NaN
f(1,1) = 0;
```

Calculating the integral by summing up the product of w and f:

```
% Calculating the integral
int_f = sum(w*f);

fprintf('The integral of f between 0 and 1 is: %1.4f \n', int_f)
```

```
The integral of f between 0 and 1 is: 1.0000
```

Calculating the derivatives using matrix multiplication:

```
% Calculating the derivatives
dfdx = A*f;
df2dx2 = B*f;
```

Plotting the result:

```
%Making the plot prettier
figProps = struct('Color', [1 1 1], 'OuterPosition', [170, 170, 1000, 700]);
fontProps = struct('FontName', 'Calibri', 'FontSize', 18, 'FontWeight', 'bold');

fig = figure(1);
ax = axes;
set(ax, 'FontSize', fontProps.FontSize);
hold on;
%Must plot in a subplot due to there being large variation in magnitude of
%the axes
subplot(3,1,1)
plot(x, f, 'Color', 'r', 'LineWidth', 2, 'Marker', 'o', 'MarkerEdgeColor', 'r', 'MarkerFaceColor', 'none', 'DisplayName', 'f(x)');
xlim([0 1]);
box("on");
grid("on");
set(legend(),'FontName' , fontProps.FontName)
set(title('Plot of the log-normal distribution with derivatives'), fontProps);
subplot(3,1,2)
plot(x, dfdx, 'Color', 'g', 'LineWidth', 2, 'Marker', 'o', 'MarkerEdgeColor', 'g', 'MarkerFaceColor', 'none', 'DisplayName', 'dfdx');
xlim([0 1]);
box("on");
grid("on");
set(legend(),'FontName' , fontProps.FontName)
subplot(3,1,3)
plot(x, df2dx2, 'Color', 'b', 'LineWidth', 2, 'Marker', 'o', 'MarkerEdgeColor', 'b', 'MarkerFaceColor', 'none', 'DisplayName', 'df2dx2');
set(xlabel('x-values'), fontProps);
xlim([0 1]);
box("on");
grid("on");
set(legend(),'FontName' , fontProps.FontName)
hold off;
set(fig, figProps);
saveas(fig, 'Ex5_plot', 'jpg')
```

The resulting plot is: