Course Instructor
Dr. Raymond C. Rumpf
Office: A-337
Phone: (915) 747-6958
E-Mail: rcrumpf@utep.edu

Computational Methods in EE

Topic 6b – Numerical Differentiation

EE 4386/5301 Computational Methods in EE

Outline

Computational Methods in EE

- Introduction
- Deriving finite-difference approximations
- Numerical differentiation
- Boundary conditions

Computational Methods in EE

Introduction

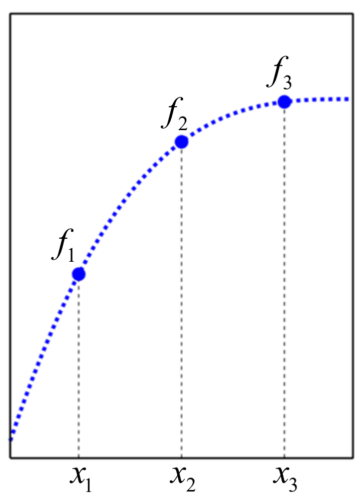
What are Finite-Difference Approximations? (1 of 3)

Computational Methods in EE

Very often in science and engineering we must calculate a derivative.

When we are processing data from measurements or simulations, there may not be an analytical equation to work with symbolically.

Typically, we only know the function at discrete points.



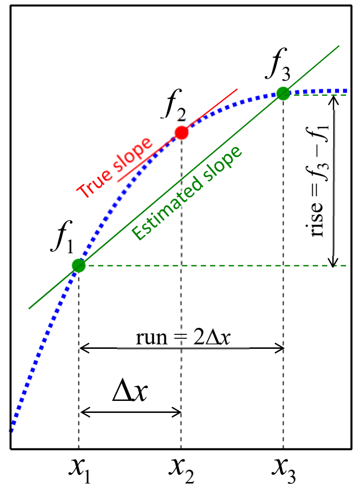
What are Finite-Difference Approximations? (2 of 3)

Computational Methods in EE

Suppose we wish to numerically calculate the first-order derivative at x_2.

The first-order derivative is slope. We can estimate the slope as rise÷run using information from surrounding points.

$$f'(x_2) \approx \frac{\text{rise}}{\text{run}} = \frac{f_3 - f_1}{2\Delta x}$$



What are Finite-Difference Approximations? (3 of 3)

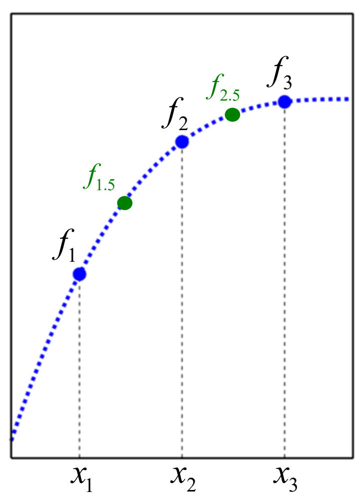
Computational Methods in EE

We can estimate the derivative at the midpoint between data points.

$$f'(x_{1.5}) = \frac{f_2 - f_1}{\Delta x} \qquad f'(x_{2.5}) = \frac{f_3 - f_2}{\Delta x}$$

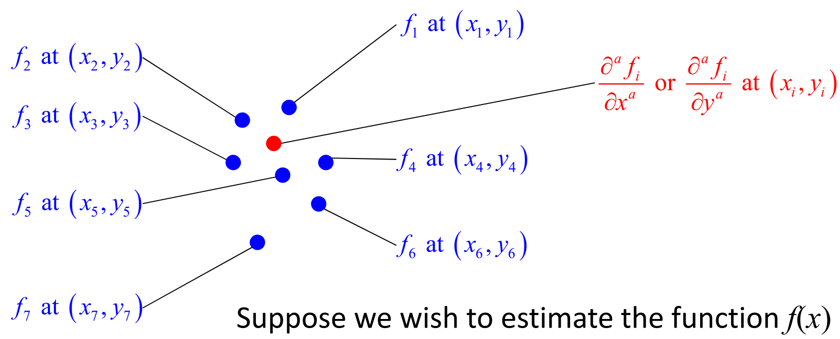
The second-order derivative is the slope of the slope.

$$f''(x_2) = \frac{f'(x_{2.5}) - f'(x_{1.5})}{\Delta x} = \frac{\dfrac{f_3 - f_2}{\Delta x} - \dfrac{f_2 - f_1}{\Delta x}}{\Delta x} = \frac{f_3 - 2f_2 + f_1}{\Delta x^2}$$



General Concept of Finite-Difference Approximations (1 of 2)

Computational Methods in EE



Suppose we wish to estimate the function $f(x)$ or one of its derivatives at location (x_i, y_i).

$$\frac{\partial^a f_i}{\partial x^a} \text{ or } \frac{\partial^a f_i}{\partial y^a} = a_1 f_1 + a_2 f_2 + a_3 f_3 + a_4 f_4 + a_5 f_5 + a_6 f_6 + a_7 f_7$$

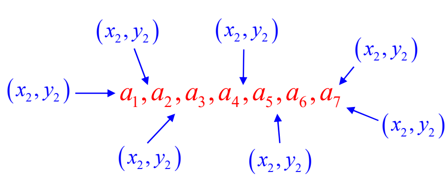
It is always possible to estimate this from a linear sum of the function values at surrounding points.

General Concept of Finite-Difference Approximations (2 of 2)

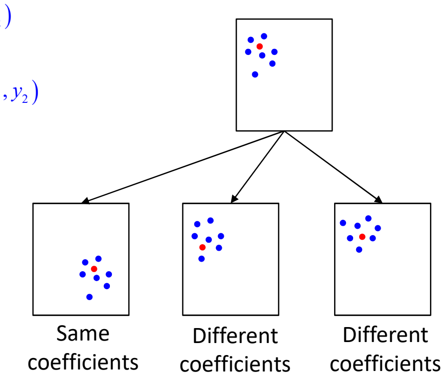
Computational Methods in EE

The trick is, how do we calculate the coefficients a_n?

These are a function of the positions of the points.



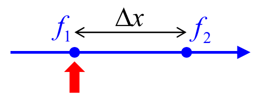
Finite-difference coefficients depend only on the relative position of the points. They do not depend on the absolute positions.



Types of Finite-Differences

Computational Methods in EE

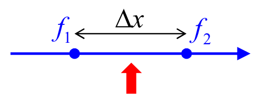
$$\frac{df_1}{dx} \approx \frac{f_2 - f_1}{\Delta x}$$



Forward Finite-Difference

Reaches ahead to use data in the forward direction.

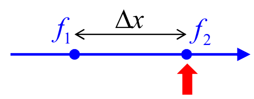
$$\frac{df_{1.5}}{dx} \approx \frac{f_2 - f_1}{\Delta x}$$



Central Finite-Difference

Reaches symmetrically to use data in both directions for highest accuracy.

$$\frac{df_2}{dx} \approx \frac{f_2 - f_1}{\Delta x}$$



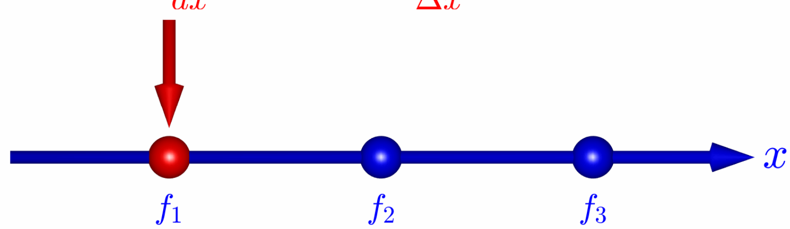
Backward Finite-Difference

Reaches behind to use data in the backward direction.

Continuum of Finite-Difference Approximations

Computational Methods in EE

$$\frac{df_{1.0}}{dx} \simeq \frac{-1.5f_1 + 2.0f_2 - 0.5f_3}{\Delta x}$$

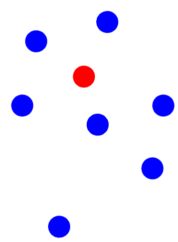


Two Key Considerations

Computational Methods in EE

1. The position of the points from which the finite-difference approximation is calculated.

2. The location of the point where the finite-difference is being evaluated. We typically want to be as centered as possible for best accuracy.



Computational Methods in EE

Deriving Finite-Difference Approximations

Concept of Using Polynomials

Computational Methods in EE

We can fit an Nth order polynomial given $N+1$ points.

$$f(x) = a_0 + a_1 x + a_2 x^2 + \cdots + a_N x^N$$

After we have done the curve fit, we can interpolate the function or any of its derivatives from this polynomial.

$$\begin{aligned}
f(x) &= a_0 + a_1 x + a_2 x^2 + a_3 x^3 + a_4 x^4 + \cdots + a_N x^N \\
f'(x) &= a_1 + 2a_2 x + 3a_3 x^2 + 4a_4 x^3 + \cdots + N a_N x^{N-1} \\
f''(x) &= 2a_2 + 6a_3 x + 12a_4 x^2 + \cdots + N(N-1) a_N x^{N-2} \\
f'''(x) &= 6a_3 + 24a_4 x + \cdots + N(N-1)(N-2) a_N x^{N-3} \\
&\vdots
\end{aligned}$$

Numerical Differentiation Slide 13

Fitting the Polynomial

Computational Methods in EE

We write our polynomial at each of our discrete points.

Since our Nth order polynomial contains $N+1$ coefficients a_n, we need at least $N+1$ discrete points to determine all of them.

This last point shows that more points in the approximation will give better accuracy in the finite-difference approximation.

$$\begin{aligned}
f(x_1) &= a_0 + a_1 x_1 + a_2 x_1^2 + \cdots + a_N x_1^N \\
f(x_2) &= a_0 + a_1 x_2 + a_2 x_2^2 + \cdots + a_N x_2^N \\
&\vdots \\
f(x_{N+1}) &= a_0 + a_1 x_{N+1} + a_2 x_{N+1}^2 + \cdots + a_N x_{N+1}^N
\end{aligned}$$

Numerical Differentiation Slide 14

Easiest Point for Evaluating $f(x)$

Computational Methods in EE

Recall the equations we will use to evaluate $f(x)$ or one of its derivatives:

$$f(x) = a_0 + a_1 x + a_2 x^2 + a_3 x^3 + a_4 x^4 + \cdots + a_N x^N$$
$$f'(x) = a_1 + 2a_2 x + 3a_3 x^2 + 4a_4 x^3 + \cdots + N a_N x^{N-1}$$
$$f''(x) = 2a_2 + 6a_3 x + 12a_4 x^2 + \cdots + N(N-1) a_N x^{N-2}$$

These are most easily evaluated at $x = 0$ because the above equations reduce to

$$f(0) = a_0$$
$$f'(0) = a_1$$
$$f''(0) = 2a_2$$
$$f'''(0) = 6a_3$$
$$\vdots$$

Numerical Differentiation Slide 15

How to Make Any Point Easy

Computational Methods in EE

Now suppose we wish to evaluate $f(x)$ or one of its derivatives at the general point $x = x_{\text{fd}}$.

To do this, we shift our x-axis by x_{fd} before fitting the polynomial.

Recall that the finite-difference coefficients depend only on the relative position of the points. An offset will not affect their values.

Now we write our polynomial at each shifted point.

$$f(\tilde{x}_1) = \tilde{a}_0 + \tilde{a}_1 \tilde{x}_1 + \tilde{a}_2 \tilde{x}_1^2 + \cdots + \tilde{a}_N \tilde{x}_1^N$$
$$f(\tilde{x}_2) = \tilde{a}_0 + \tilde{a}_1 \tilde{x}_2 + \tilde{a}_2 \tilde{x}_2^2 + \cdots + \tilde{a}_N \tilde{x}_2^N$$
$$\vdots$$
$$f(\tilde{x}_{N+1}) = \tilde{a}_0 + \tilde{a}_1 \tilde{x}_{N+1} + \tilde{a}_2 \tilde{x}_{N+1}^2 + \cdots + \tilde{a}_N \tilde{x}_{N+1}^N$$

$$\tilde{x}_n = x_n - x_{\text{fd}}$$

In our shifted coordinate system, our finite-difference is being evaluated at $\tilde{x} = 0$.

$$f(0) = \tilde{a}_0 \qquad f'(0) = \tilde{a}_1 \qquad f''(0) = 2\tilde{a}_2 \qquad f'''(0) = 6\tilde{a}_3 \qquad \cdots$$

Numerical Differentiation Slide 16

Matrix Form of Curve Fit

Computational Methods in EE

Given our shifted coordinate system, we write the polynomial at each discrete point and put this large set of equations in matrix form.

$$f(x) = \tilde{a}_0 + \tilde{a}_1\tilde{x} + \tilde{a}_2\tilde{x}^2 + \tilde{a}_3\tilde{x}^3 + \tilde{a}_4\tilde{x}^4 + \cdots + \tilde{a}_N\tilde{x}^N$$
$$f'(x) = \tilde{a}_1 + 2\tilde{a}_2\tilde{x} + 3\tilde{a}_3\tilde{x}^2 + 4\tilde{a}_4\tilde{x}^3 + \cdots + N\tilde{a}_N\tilde{x}^{N-1}$$
$$f''(x) = 2\tilde{a}_2 + 6\tilde{a}_3\tilde{x} + 12\tilde{a}_4\tilde{x}^2 + \cdots + N(N-1)\tilde{a}_N\tilde{x}^{N-2}$$
$$\vdots$$

$$\begin{bmatrix} f_1 \\ f_2 \\ \vdots \\ f_{N+1} \end{bmatrix} = \begin{bmatrix} 1 & \tilde{x}_1 & \cdots & \tilde{x}_1^N \\ 1 & \tilde{x}_2 & \cdots & \tilde{x}_2^N \\ \vdots & \vdots & \ddots & \\ 1 & \tilde{x}_{N+1} & & \tilde{x}_{N+1}^N \end{bmatrix} \begin{bmatrix} \tilde{a}_0 \\ \tilde{a}_1 \\ \vdots \\ \tilde{a}_N \end{bmatrix}$$
$$= \begin{bmatrix} 1 & (x_1 - x_0) & \cdots & (x_1 - x_0)^N \\ 1 & (x_2 - x_0) & \cdots & (x_2 - x_0)^N \\ \vdots & \vdots & \ddots & \\ 1 & (x_{N+1} - x_0) & & (x_{N+1} - x_0)^N \end{bmatrix} \begin{bmatrix} \tilde{a}_0 \\ \tilde{a}_1 \\ \vdots \\ \tilde{a}_N \end{bmatrix}$$

Step 1 – Choose x Coordinates

Computational Methods in EE

Identify the x-coordinates of the points from which you wish to approximate a derivate.

$$[x] = [x_1 \quad x_2 \quad \cdots \quad x_{N+1}]^T$$

Step 2 – Shift x-Axis

Computational Methods in EE

Shift the function across the x-axis until $x' = 0$ corresponds to the point where you wish to approximate the derivative.

$$\frac{d^a f(x = x_0)}{dx^a} = \frac{d^a f(\tilde{x} = 0)}{dx^a}$$

$$[\tilde{x}] = [x] - x_0$$
$$= \left[(x_1 - x_0) \quad (x_2 - x_0) \quad \cdots \quad (x_{N+1} - x_0) \right]^T$$

Step 3 – Build $[X]$ Matrix

Computational Methods in EE

Use the column vector $[\tilde{x}]$ to build $[X]$.

$$[\tilde{X}] = \begin{bmatrix} [\tilde{x}]^0 & [\tilde{x}]^1 & \cdots & [\tilde{x}]^N \end{bmatrix}$$

$$= \begin{bmatrix} 1 & \tilde{x}_1 & \cdots & \tilde{x}_1^N \\ 1 & \tilde{x}_2 & \cdots & \tilde{x}_2^N \\ \vdots & \vdots & \ddots & \\ 1 & \tilde{x}_{N+1} & & \tilde{x}_{N+1}^N \end{bmatrix} = \begin{bmatrix} 1 & x_1 - x_0 & \cdots & (x_1 - x_0)^N \\ 1 & x_2 - x_0 & \cdots & (x_2 - x_0)^N \\ \vdots & \vdots & \ddots & \vdots \\ 1 & x_{N+1} - x_0 & \cdots & (x_{N+1} - x_0)^N \end{bmatrix}$$

Insert 1's instead of $[\tilde{x}]^0$.

Step 4 – Invert $\left[\tilde{X}\right]$

Computational Methods in EE

Calculate the inverse of $\left[\tilde{X}\right]$.

$$\left[\tilde{Y}\right]=\left[\tilde{X}\right]^{-1}=\begin{bmatrix}\tilde{y}_{11} & \tilde{y}_{12} & \cdots & \tilde{y}_{1,N+1}\\ \tilde{y}_{21} & \tilde{y}_{22} & \cdots & \tilde{y}_{2,N+1}\\ \vdots & \vdots & \ddots & \vdots\\ \tilde{y}_{N+1,1} & \tilde{y}_{N+1,2} & \cdots & \tilde{y}_{N+1,N+1}\end{bmatrix}$$

Step 5 – Extract Polynomial Coefficients

Computational Methods in EE

$$\left[\tilde{a}\right]=\left[\tilde{X}\right]^{-1}\left[f\right]=\left[\tilde{Y}\right]\left[f\right]$$

$$\begin{bmatrix}\tilde{a}_0\\ \tilde{a}_1\\ \tilde{a}_2\\ \vdots\\ \tilde{a}_N\end{bmatrix}=\begin{bmatrix}\tilde{y}_{11} & \tilde{y}_{12} & \tilde{y}_{13} & \cdots & \tilde{y}_{1,N+1}\\ \tilde{y}_{21} & \tilde{y}_{22} & \tilde{y}_{23} & \cdots & \tilde{y}_{2,N+1}\\ \tilde{y}_{31} & \tilde{y}_{32} & \tilde{y}_{33} & \cdots & \tilde{y}_{3,N+1}\\ \vdots & \vdots & \vdots & \ddots & \vdots\\ \tilde{y}_{N+1,1} & \tilde{y}_{N+1,2} & \tilde{y}_{N+1,3} & \cdots & \tilde{y}_{N+1,N+1}\end{bmatrix}\begin{bmatrix}f_1\\ f_2\\ f_3\\ \vdots\\ f_{N+1}\end{bmatrix}$$

$$\begin{aligned}\tilde{a}_0 &= \tilde{y}_{11}f_1+\tilde{y}_{12}f_2+\tilde{y}_{13}f_3+\cdots+\tilde{y}_{1,N+1}f_{N+1}\\ \tilde{a}_1 &= \tilde{y}_{21}f_1+\tilde{y}_{22}f_2+\tilde{y}_{23}f_3+\cdots+\tilde{y}_{2,N+1}f_{N+1}\\ \tilde{a}_2 &= \tilde{y}_{31}f_1+\tilde{y}_{32}f_2+\tilde{y}_{33}f_3+\cdots+\tilde{y}_{3,N+1}f_{N+1}\\ &\vdots\\ \tilde{a}_N &= \tilde{y}_{N+1,1}f_1+\tilde{y}_{N+1,2}f_2+\tilde{y}_{N+1,3}f_3+\cdots+\tilde{y}_{1,N+1}f_{N+1}\end{aligned}$$

Step 6 – Write Finite-Difference Approximation

Computational Methods in EE

$$f(\tilde{x}=0)=\tilde{a}_0$$
$$\frac{d}{dx}f(\tilde{x}=0)=\tilde{a}_1$$
$$\frac{d^2}{dx^2}f(\tilde{x}=0)=2\tilde{a}_2$$
$$\vdots$$

$$\begin{aligned}
\tilde{a}_0 &= \tilde{y}_{11}f_1+\tilde{y}_{12}f_2+\tilde{y}_{13}f_3+\cdots+\tilde{y}_{1,N+1}f_{N+1}\\
\tilde{a}_1 &= \tilde{y}_{21}f_1+\tilde{y}_{22}f_2+\tilde{y}_{23}f_3+\cdots+\tilde{y}_{2,N+1}f_{N+1}\\
\tilde{a}_2 &= \tilde{y}_{31}f_1+\tilde{y}_{32}f_2+\tilde{y}_{33}f_3+\cdots+\tilde{y}_{3,N+1}f_{N+1}\\
&\vdots\\
\tilde{a}_N &= \tilde{y}_{N+1,1}f_1+\tilde{y}_{N+1,2}f_2+\tilde{y}_{N+1,3}f_3+\cdots+\tilde{y}_{1,N+1}f_{N+1}
\end{aligned}$$

Example #1

Computational Methods in EE

Derive first-order and second-order finite-difference approximations that span across three points. The approximations should be evaluated at the midpoint.

$$[\tilde{x}]=\begin{bmatrix}-h\\0\\h\end{bmatrix}\qquad [\tilde{X}]=\begin{bmatrix}[\tilde{x}]^0 & [\tilde{x}]^1 & [\tilde{x}]^2\end{bmatrix}=\begin{bmatrix}1 & -h & h^2\\1 & 0 & 0\\1 & h & h^2\end{bmatrix}\qquad [\tilde{Y}]=[\tilde{X}]^{-1}=\begin{bmatrix}0 & 1 & 0\\-1/2h & 0 & 1/2h\\1/2h^2 & -1/h^2 & 1/2h^2\end{bmatrix}$$

$$\begin{bmatrix}\tilde{a}_0\\\tilde{a}_1\\\tilde{a}_2\end{bmatrix}=\begin{bmatrix}0 & 1 & 0\\-1/2h & 0 & 1/2h\\1/2h^2 & -1/h^2 & 1/2h^2\end{bmatrix}\begin{bmatrix}f_1\\f_2\\f_3\end{bmatrix}\quad\Rightarrow\quad
\begin{aligned}
\tilde{a}_0 &= 0\cdot f_1+1\cdot f_2+0\cdot f_3=f_2\\
\tilde{a}_1 &= (-1/2h)\cdot f_1+0\cdot f_2+(1/2h)\cdot f_3=\frac{-f_1+f_3}{2h}\\
\tilde{a}_2 &= (1/2h^2)\cdot f_1+(-1/h^2)\cdot f_2+(1/2h^2)\cdot f_3=\frac{f_1-2f_2+f_3}{2h^2}
\end{aligned}$$

$$f(x_{\text{fd}})=\tilde{a}_0=f_2$$
$$\frac{df(x_{\text{fd}})}{dx}=\tilde{a}_1=\frac{f_3-f_1}{2h}$$
$$\frac{d^2f(x_{\text{fd}})}{dx^2}=2\tilde{a}_2=\frac{f_1-2f_2+f_3}{h^2}$$

Example #2

Computational Methods in EE

Derive first-order and second-order finite-difference approximations that span across three points. The approximations should be evaluated at the first point.

$$[\tilde{x}] = \begin{bmatrix} 0 \\ h \\ 2h \end{bmatrix} \qquad \left[\tilde{X}\right] = \left[[\tilde{x}]^0 \;\; [\tilde{x}]^1 \;\; [\tilde{x}]^2 \right] = \begin{bmatrix} 1 & 0 & 0 \\ 1 & h & h^2 \\ 1 & 2h & (2h)^2 \end{bmatrix} \qquad \left[\tilde{Y}\right] = \left[\tilde{X}\right]^{-1} = \begin{bmatrix} 1 & 0 & 0 \\ -3/(2h) & 2/h & -1/2h \\ 1/(2h^2) & -1/h^2 & 1/(2h^2) \end{bmatrix}$$

$$\begin{bmatrix} \tilde{a}_0 \\ \tilde{a}_1 \\ \tilde{a}_2 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ -3/(2h) & 2/h & -1/2h \\ 1/(2h^2) & -1/h^2 & 1/(2h^2) \end{bmatrix} \begin{bmatrix} f_1 \\ f_2 \\ f_3 \end{bmatrix} \Rightarrow \begin{aligned} \tilde{a}_0 &= 1 \cdot f_1 + 0 \cdot f_2 + 0 \cdot f_3 = f_1 \\ \tilde{a}_1 &= (-3/2h) \cdot f_1 + (2/h) \cdot f_2 + (-1/2h) \cdot f_3 = \frac{-1.5 f_1 + 2f - 0.5 f_3}{h} \\ \tilde{a}_2 &= (1/2h^2) \cdot f_1 + (-1/h^2) \cdot f_2 + (1/2h^2) \cdot f_3 = \frac{f_1 - 2f_2 + f_3}{2h^2} \end{aligned}$$

$$f(x_{\text{fd}}) = \tilde{a}_0 = f_1$$

$$\frac{df(x_{\text{fd}})}{dx} = \tilde{a}_1 = \frac{-1.5 f_1 + 2f - 0.5 f_3}{h}$$

$$\frac{d^2 f(x_{\text{fd}})}{dx^2} = 2\tilde{a}_2 = \frac{f_1 - 2f_2 + f_3}{h^2}$$

Example #3 – Higher Order Accuracy (1 of 2)

Computational Methods in EE

Let's evaluate some derivatives at the midpoint of four discrete points.

$$[\tilde{x}] = \begin{bmatrix} -3h/2 \\ -h/2 \\ h/2 \\ 3h/2 \end{bmatrix} \qquad \left[\tilde{X}\right] = \left[[\tilde{x}]^0 \;\; [\tilde{x}]^1 \;\; [\tilde{x}]^2 \;\; [\tilde{x}]^3 \right] = \begin{bmatrix} 1 & -\frac{3h}{2} & \frac{9h^2}{4} & -\frac{27h^3}{8} \\ 1 & -\frac{h}{2} & \frac{h^2}{4} & -\frac{h^3}{8} \\ 1 & \frac{h}{2} & \frac{h^2}{4} & \frac{h^3}{8} \\ 1 & \frac{3h}{2} & \frac{9h^2}{4} & \frac{27h^2}{8} \end{bmatrix}$$

$$\left[\tilde{Y}\right] = \left[\tilde{X}\right]^{-1} = \begin{bmatrix} -\frac{1}{16} & \frac{9}{16} & \frac{9}{16} & -\frac{1}{16} \\ \frac{1}{24h} & -\frac{9}{8h} & \frac{9}{8h} & -\frac{1}{24h} \\ \frac{1}{4h^2} & -\frac{1}{4h^2} & -\frac{1}{4h^2} & \frac{1}{4h^2} \\ -\frac{1}{6h^3} & \frac{1}{2h^3} & -\frac{1}{2h^3} & \frac{1}{6h^3} \end{bmatrix}$$

Example #3 – Higher Order Accuracy (2 of 2)

Computational Methods in EE

The coefficients are then

$$
\begin{bmatrix} \tilde{a}_0 \\ \tilde{a}_1 \\ \tilde{a}_2 \\ \tilde{a}_3 \end{bmatrix} = \underbrace{\begin{bmatrix} -\frac{1}{16} & \frac{9}{16} & \frac{9}{16} & -\frac{1}{16} \\ \frac{1}{24h} & -\frac{9}{8h} & \frac{9}{8h} & -\frac{1}{24h} \\ \frac{1}{4h^2} & -\frac{1}{4h^2} & -\frac{1}{4h^2} & \frac{1}{4h^2} \\ -\frac{1}{6h^3} & \frac{1}{2h^3} & -\frac{1}{2h^3} & \frac{1}{6h^3} \end{bmatrix}}_{[\tilde{Y}]} \begin{bmatrix} f_1 \\ f_2 \\ f_3 \\ f_4 \end{bmatrix}
$$

$$
\begin{aligned}
\tilde{a}_0 &= -\frac{1}{16} f_1 + \frac{9}{16} f_2 + \frac{9}{16} f_3 - \frac{1}{16} f_4 \\
\tilde{a}_1 &= \frac{1}{24h} f_1 - \frac{9}{8h} f_2 + \frac{9}{8h} f_3 - \frac{1}{24h} f_4 \\
\tilde{a}_2 &= \frac{1}{4h^2} f_1 - \frac{1}{4h^2} f_2 - \frac{1}{4h^2} f_3 + \frac{1}{4h^2} f_4 \\
\tilde{a}_3 &= -\frac{1}{6h^3} f_1 + \frac{1}{2h^3} f_2 - \frac{1}{2h^3} f_3 + \frac{1}{6h^3} f_4
\end{aligned}
$$

$$
\begin{aligned}
f(x_{2.5}) &= \tilde{a}_0 = \frac{-f_1 + 9f_2 + 9f_3 - f_4}{16} \\
\frac{df(x_{2.5})}{dx} &= \tilde{a}_1 = \frac{f_1 - 27f_2 + 27f_3 - f_4}{24\Delta x} \\
\frac{d^2 f(x_{2.5})}{dx^2} &= 2\tilde{a}_2 = \frac{f_1 - f_2 - f_3 + f_4}{2(\Delta x)^2}
\end{aligned}
$$

Determining Finite-Difference Approximations with MATLAB (1 of 6)

Computational Methods in EE

We have so far derived finite-difference approximations symbolically. What if we want 6th-order accurate finite-differences? This is unreasonable to do symbolically.

Recall our matrix equation representing the polynomial written at each discrete point. It always had the following form where the *w*'s are just constants.

$$
\begin{bmatrix} f_1 \\ f_2 \\ \vdots \\ f_{N+1} \end{bmatrix} = \begin{bmatrix} 1 & w_{12}h & w_{13}h^2 & \cdots & w_{1N}h^N \\ 1 & w_{22}h & w_{23}h^2 & & w_{2N}h^N \\ & \vdots & & \ddots & \\ 1 & w_{N+1,2}h & w_{N+1,3}h^2 & & w_{N+1,N}h^N \end{bmatrix} \begin{bmatrix} \tilde{a}_0 \\ \tilde{a}_1 \\ \vdots \\ \tilde{a}_N \end{bmatrix}
$$

Determining Finite-Difference Approximations with MATLAB (2 of 6)

Computational Methods in EE

Now, we are able to separate the w terms from the h terms.

$$\begin{bmatrix} f_1\\ f_2\\ \vdots\\ f_{N+1}\end{bmatrix} = \begin{bmatrix}1 & w_{12} & w_{13} & \cdots & w_{1N}\\ 1 & w_{22} & w_{23} & & w_{2N}\\ & \vdots & & \ddots & \\ 1 & w_{N+1,2} & w_{N+1,3} & & w_{N+1,N}\end{bmatrix}\begin{bmatrix}1 &&&&\\ & h &&&\\ && h^2 &&\\ &&&\ddots&\\ &&&& h^N\end{bmatrix}\begin{bmatrix}\tilde{a}_0\\ \tilde{a}_1\\ \vdots\\ \tilde{a}_N\end{bmatrix}$$

We were able to put numbers to all of these coefficients. This is a fully numerical matrix. It does not contain any symbolic variables.

These are our symbolic variables.

Hint: $[W]=[\tilde{X}]$ when $h=1$

So we build $[W]$ by building $[\tilde{X}]$ and pretending $h = 1$.

Determining Finite-Difference Approximations with MATLAB (3 of 6)

Computational Methods in EE

Solving our matrix equation for $[a]$ gives

$$\begin{bmatrix}\tilde{a}_0\\ \tilde{a}_1\\ \vdots\\ \tilde{a}_N\end{bmatrix} = \begin{bmatrix}1 &&&&\\ & h &&&\\ && h^2 &&\\ &&&\ddots&\\ &&&& h^N\end{bmatrix}^{-1}\begin{bmatrix}1 & w_{12} & w_{13} & \cdots & w_{1N}\\ 1 & w_{22} & w_{23} & & w_{2N}\\ & \vdots & & \ddots & \\ 1 & w_{N+1,2} & w_{N+1,3} & & w_{N+1,N}\end{bmatrix}^{-1}\begin{bmatrix} f_1\\ f_2\\ \vdots\\ f_{N+1}\end{bmatrix}$$

$$\downarrow$$

$$[V]=[W]^{-1}$$

$$\begin{bmatrix}\tilde{a}_0\\ \tilde{a}_1\\ \vdots\\ \tilde{a}_N\end{bmatrix} = \begin{bmatrix}1 &&&&\\ & \frac{1}{h} &&&\\ && \frac{1}{h^2} &&\\ &&&\ddots&\\ &&&& \frac{1}{h^N}\end{bmatrix}\begin{bmatrix}v_{11} & v_{12} & v_{13} & \cdots & v_{1N}\\ v_{21} & v_{22} & v_{23} & & v_{2N}\\ & \vdots & & \ddots & \\ v_{N+1,1} & v_{N+1,2} & v_{N+1,3} & & v_{N+1,N}\end{bmatrix}\begin{bmatrix} f_1\\ f_2\\ \vdots\\ f_{N+1}\end{bmatrix}$$

Determining Finite-Difference Approximations with MATLAB (4 of 6)

Computational Methods in EE

Next we multiply our matrices.

$$\begin{bmatrix} \tilde{a}_0 \\ \tilde{a}_1 \\ \vdots \\ \tilde{a}_N \end{bmatrix} = \begin{bmatrix} 1 & & & & \\ & \frac{1}{h} & & & \\ & & \frac{1}{h^2} & & \\ & & & \ddots & \\ & & & & \frac{1}{h^N} \end{bmatrix} \begin{bmatrix} v_{11} & v_{12} & v_{13} & \cdots & v_{1N} \\ v_{21} & v_{22} & v_{23} & & v_{2N} \\ & \vdots & & \ddots & \\ v_{N+1,1} & v_{N+1,2} & v_{N+1,3} & & v_{N+1,N} \end{bmatrix} \begin{bmatrix} f_1 \\ f_2 \\ \vdots \\ f_{N+1} \end{bmatrix}$$

$$\downarrow$$

$$\begin{bmatrix} \tilde{a}_0 \\ \tilde{a}_1 \\ \vdots \\ \tilde{a}_N \end{bmatrix} = \begin{bmatrix} 1 \cdot v_{11} & 1 \cdot v_{12} & 1 \cdot v_{13} & \cdots & 1 \cdot v_{1N} \\ \frac{1}{h} \cdot v_{21} & \frac{1}{h} \cdot v_{22} & \frac{1}{h} \cdot v_{23} & & \frac{1}{h} \cdot v_{2N} \\ & \vdots & & \ddots & \\ \frac{1}{h^N} \cdot v_{N+1,1} & \frac{1}{h^N} \cdot v_{N+1,2} & \frac{1}{h^N} \cdot v_{N+1,3} & & \frac{1}{h^N} \cdot v_{N+1,N} \end{bmatrix} \begin{bmatrix} f_1 \\ f_2 \\ \vdots \\ f_{N+1} \end{bmatrix}$$

Determining Finite-Difference Approximations with MATLAB (5 of 6)

Computational Methods in EE

Next, we read off the polynomial coefficients.

$$\begin{bmatrix} \tilde{a}_0 \\ \tilde{a}_1 \\ \vdots \\ \tilde{a}_N \end{bmatrix} = \begin{bmatrix} 1 \cdot v_{11} & 1 \cdot v_{12} & 1 \cdot v_{13} & \cdots & 1 \cdot v_{1N} \\ \frac{1}{h} \cdot v_{21} & \frac{1}{h} \cdot v_{22} & \frac{1}{h} \cdot v_{23} & & \frac{1}{h} \cdot v_{2N} \\ & \vdots & & \ddots & \\ \frac{1}{h^N} \cdot v_{N+1,1} & \frac{1}{h^N} \cdot v_{N+1,2} & \frac{1}{h^N} \cdot v_{N+1,3} & & \frac{1}{h^N} \cdot v_{N+1,N} \end{bmatrix} \begin{bmatrix} f_1 \\ f_2 \\ \vdots \\ f_{N+1} \end{bmatrix}$$

$$\downarrow$$

$$\tilde{a}_0 = v_{11} f_1 + v_{12} f_2 + \cdots + v_{1N} f_{N+1}$$

$$\tilde{a}_1 = \frac{v_{21} f_1 + v_{22} f_2 + \cdots + v_{2N} f_{N+1}}{h}$$

$$\vdots$$

$$\tilde{a}_N = \frac{v_{N+1,1} f_1 + v_{N+1,2} f_2 + \cdots + v_{N+1,N} f_{N+1}}{h^N}$$

Determining Finite-Difference Approximations with MATLAB (6 of 6)

Computational Methods in EE

Last, we write our finite-difference approximations from the polynomial coefficients.

$$f = \tilde{a}_0 = v_{11} f_1 + v_{12} f_2 + \cdots + v_{1N} f_{N+1}$$
$$\frac{df}{dx} = \tilde{a}_1 = \frac{v_{21} f_1 + v_{22} f_2 + \cdots + v_{2N} f_{N+1}}{h}$$
$$\frac{d^2 f}{dx^2} = 2\tilde{a}_2 = 2\frac{v_{31} f_1 + v_{32} f_2 + \cdots + v_{3N} f_{N+1}}{h^2}$$

Staring at these equations long enough, we realize that the v_{ij} coefficients can be determined completely numerically. We just have to remember to divide by h^α and perhaps multiply the finite-difference expression by a constant.

Example #4 – 6th Order Accurate Finite-Differences (1 of 2)

Computational Methods in EE

1. Here we need seven points to calculate seven polynomial coefficients

$$[\tilde{x}] = [-3h \quad -2h \quad -h \quad 0 \quad h \quad 2h \quad 3h]^T$$

2. To build the $[W]$ matrix, set $h = 1$ for now.

$$[\hat{\tilde{x}}] = [-3 \quad -2 \quad -1 \quad 0 \quad 1 \quad 2 \quad 3]^T$$

$$[W] = \left[[\hat{\tilde{x}}]^0 \quad [\hat{\tilde{x}}]^1 \quad [\hat{\tilde{x}}]^2 \quad [\hat{\tilde{x}}]^3 \quad [\hat{\tilde{x}}]^4 \quad [\hat{\tilde{x}}]^5 \quad [\hat{\tilde{x}}]^6 \right] = \begin{bmatrix} 1 & -3 & 9 & -27 & 81 & -243 & 729 \\ 1 & -2 & 4 & -8 & 16 & -32 & 64 \\ 1 & -1 & 1 & -1 & 1 & -1 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 2 & 4 & 8 & 16 & 32 & 64 \\ 1 & 3 & 9 & 27 & 81 & 243 & 729 \end{bmatrix}$$

Example #4 – 6th Order Accurate Finite-Differences (2 of 2)

Computational Methods in EE

3. Invert $[W]$.

$$[V]=[W]^{-1}=\begin{bmatrix} 0 & 0 & 0 & 1.0000 & 0 & 0 & 0 \\ -0.0167 & 0.1500 & -0.7500 & -0.0000 & 0.7500 & -0.1500 & 0.0167 \\ 0.0056 & -0.0750 & 0.7500 & -1.3611 & 0.7500 & -0.0750 & 0.0056 \\ 0.0208 & -0.1667 & 0.2708 & 0.0000 & -0.2708 & 0.1667 & -0.0208 \\ -0.0069 & 0.0833 & -0.2708 & 0.3889 & -0.2708 & 0.0833 & -0.0069 \\ -0.0042 & 0.0167 & -0.0208 & -0.0000 & 0.0208 & -0.0167 & 0.0042 \\ 0.0014 & -0.0083 & 0.0208 & -0.0278 & 0.0208 & -0.0083 & 0.0014 \end{bmatrix}$$

4. Write the finite-difference approximations, remembering to incorporate the symbolic h's back in.

$$f \approx a_0 = \frac{0\cdot f_1 + 0\cdot f_2 + 0\cdot f_3 + 1\cdot f_4 + 0\cdot f_5 - 0\cdot f_6 + 0\cdot f_7}{1}$$

$$\frac{\partial f}{\partial x} \approx a_1 = \frac{-0.0167 f_1 + 0.15 f_2 - 0.75 f_3 - 0\cdot f_4 + 0.75 f_5 - 0.15 f_6 + 0.0167 f_7}{h}$$

$$\frac{\partial^2 f}{\partial x^2} \approx 2a_2 = \frac{0.0111 f_1 - 0.15 f_2 + 1.5 f_3 - 2.7222 f_4 + 1.5 f_5 - 0.15 f_6 + 0.0111 f_7}{h^2}$$

Computational Methods in EE

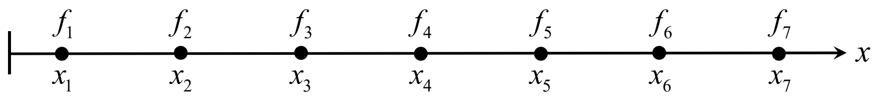
Numerical Differentiation

The Problem

Computational Methods in EE

We wish to calculate the second-order derivative of some function that is known at seven points.

$$\frac{d^2 f(x)}{dx^2} \cong ?$$

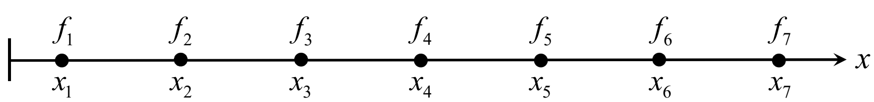


The Finite-Difference Approximation

Computational Methods in EE

We can estimate the second-order derivative with a 3-point finite-difference approximation.

$$\frac{d^2 f_i}{dx^2} \cong \frac{f_{i-1} - 2f_i + f_{i+1}}{h^2}$$

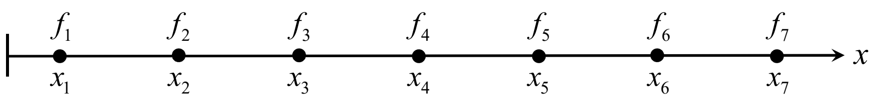


The Middle Points

Computational Methods in EE

We can calculate the derivatives at each intermediate point by applying our finite-difference approximation using the surrounding points.

$$\frac{d^2 f_2}{dx^2} \cong \frac{f_1 - 2f_2 + f_3}{h^2} \qquad \frac{d^2 f_5}{dx^2} \cong \frac{f_4 - 2f_5 + f_6}{h^2}$$



The Boundary Problem

Computational Methods in EE

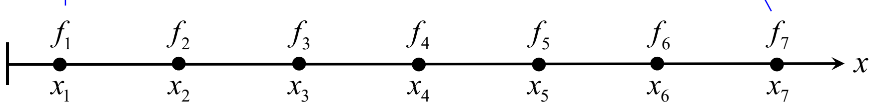
How do we evaluate these finite-differences at $i = 1$ and $i = 7$?

Does not exist

$$\frac{d^2 f_1}{dx^2} \cong \frac{f_0 - 2f_1 + f_2}{h^2}$$

Does not exist

$$\frac{d^2 f_7}{dx^2} \cong \frac{f_6 - 2f_7 + f_8}{h^2}$$



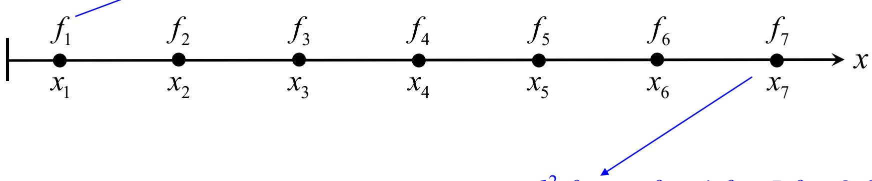
The finite-difference equations at the boundaries of the grid contain terms that do not exist because they are not stored in memory.

The Boundary Fix

Computational Methods in EE

We must derive new finite-difference approximations for each boundary point.

$$[x]=[0 \quad h \quad 2h \quad 3h]^T \qquad \frac{d^2 f_1}{dx^2} \cong \frac{2f_1 - 5f_2 + 4f_3 - f_4}{h^2}$$



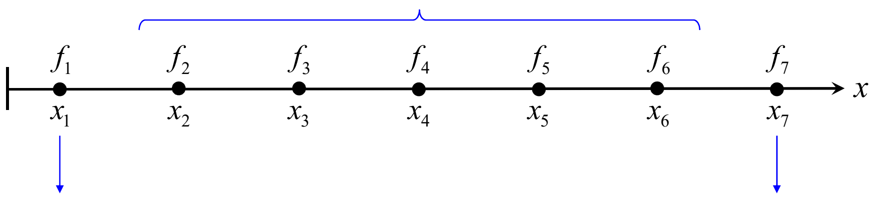
$$[x]=[-3h \quad -2h \quad -h \quad 0]^T \qquad \frac{d^2 f_7}{dx^2} \cong \frac{-f_4 + 4f_5 - 5f_6 + 2f_7}{h^2}$$

Summary of Finite-Difference Approximations

Computational Methods in EE

Below are all of the equations used across the entire grid.

$$\frac{d^2 f_i}{dx^2} \cong \frac{f_{i-1} - 2f_i + f_{i+1}}{h^2}$$

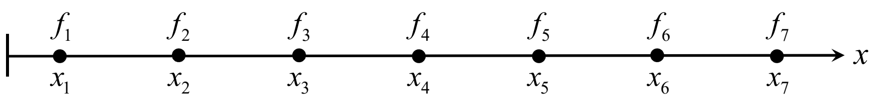


$$\frac{d^2 f_1}{dx^2} \cong \frac{2f_1 - 5f_2 + 4f_3 - f_4}{h^2} \qquad \frac{d^2 f_7}{dx^2} \cong \frac{-f_4 + 4f_5 - 5f_6 + 2f_7}{h^2}$$

MATLAB Code for Numerical Differentiation

Computational Methods in EE

```
fd(1) = (2*f(1) - 5*f(2) + 4*f(3) - f(4))/h^2;
for nx = 2 : Nx-1
   fd(nx) = (f(nx-1) - 2*f(nx) + f(nx+1))/h^2;
end
fd(Nx) = (-f(Nx-3) + 4*f(Nx-2) - 5*f(Nx-1) + 2*f(Nx))/h^2;
```

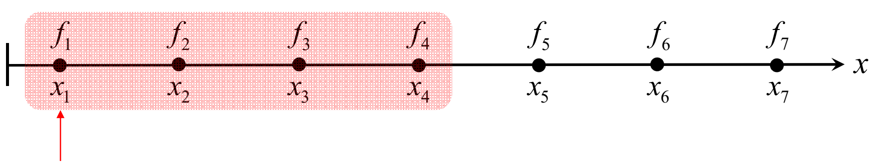


Calculation at Point 1

Computational Methods in EE

```
fd(1) = (2*f(1) - 5*f(2) + 4*f(3) - f(4))/h^2;
for nx = 2 : Nx-1
   fd(nx) = (f(nx-1) - 2*f(nx) + f(nx+1))/h^2;
end
fd(Nx) = (-f(Nx-3) + 4*f(Nx-2) - 5*f(Nx-1) + 2*f(Nx))/h^2;
```

```
fd(1) = (2*f(1) - 5*f(2) + 4*f(3) - f(4))/h^2;
```



$$[x] = [0 \quad h \quad 2h \quad 3h]^T$$

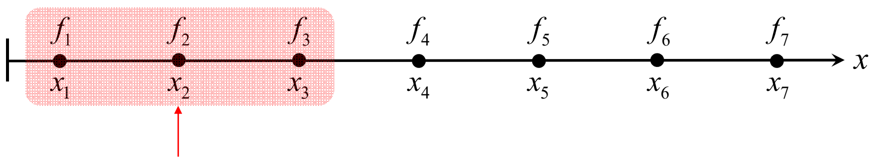
$$\frac{d^2 f_1}{dx^2} \cong \frac{2f_1 - 5f_2 + 4f_3 - f_4}{h^2}$$

Calculation at Point 2

Computational Methods in EE

```
fd(1) = (2*f(1) - 5*f(2) + 4*f(3) - f(4))/h^2;
for nx = 2 : Nx-1
  fd(nx) = (f(nx-1) - 2*f(nx) + f(nx+1))/h^2;          nx = 2
end
fd(Nx) = (-f(Nx-3) + 4*f(Nx-2) - 5*f(Nx-1) + 2*f(Nx))/h^2;
```

```
fd(2) = (f(1) - 2*f(2) + f(3))/h^2;
```



$$[x] = [-h \quad 0 \quad h]^T$$

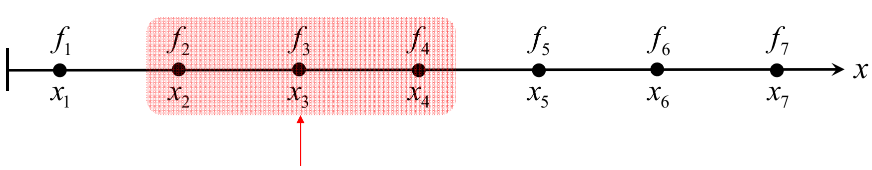
$$\frac{d^2 f_2}{dx^2} \cong \frac{f_1 - 2f_2 + f_3}{h^2}$$

Calculation at Point 3

Computational Methods in EE

```
fd(1) = (2*f(1) - 5*f(2) + 4*f(3) - f(4))/h^2;
for nx = 2 : Nx-1
  fd(nx) = (f(nx-1) - 2*f(nx) + f(nx+1))/h^2;          nx = 3
end
fd(Nx) = (-f(Nx-3) + 4*f(Nx-2) - 5*f(Nx-1) + 2*f(Nx))/h^2;
```

```
fd(3) = (f(2) - 2*f(3) + f(4))/h^2;
```



$$[x] = [-h \quad 0 \quad h]^T$$

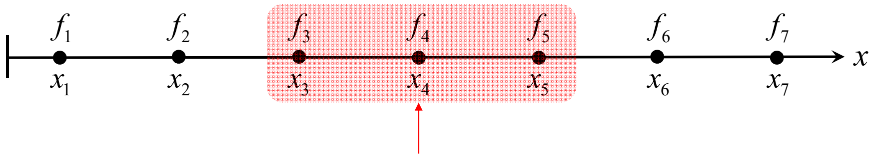
$$\frac{d^2 f_3}{dx^2} \cong \frac{f_2 - 2f_3 + f_4}{h^2}$$

Calculation at Point 4

Computational Methods in EE

```
fd(1) = (2*f(1) - 5*f(2) + 4*f(3) - f(4))/h^2;
for nx = 2 : Nx-1
  fd(nx) = (f(nx-1) - 2*f(nx) + f(nx+1))/h^2;              nx = 4
end
fd(Nx) = (-f(Nx-3) + 4*f(Nx-2) - 5*f(Nx-1) + 2*f(Nx))/h^2;
```

```
fd(4) = (f(3) - 2*f(4) + f(5))/h^2;
```



$$[x] = [-h \quad 0 \quad h]^T$$

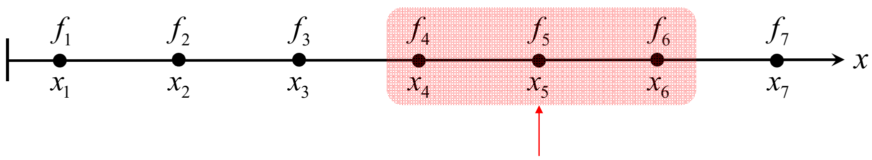
$$\frac{d^2 f_4}{dx^2} \cong \frac{f_3 - 2f_4 + f_5}{h^2}$$

Calculation at Point 5

Computational Methods in EE

```
fd(1) = (2*f(1) - 5*f(2) + 4*f(3) - f(4))/h^2;
for nx = 2 : Nx-1
  fd(nx) = (f(nx-1) - 2*f(nx) + f(nx+1))/h^2;              nx = 5
end
fd(Nx) = (-f(Nx-3) + 4*f(Nx-2) - 5*f(Nx-1) + 2*f(Nx))/h^2;
```

```
fd(5) = (f(4) - 2*f(5) + f(6))/h^2;
```



$$[x] = [-h \quad 0 \quad h]^T$$

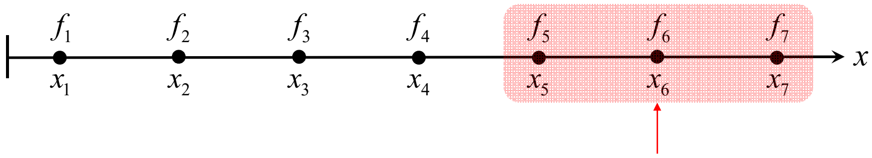
$$\frac{d^2 f_5}{dx^2} \cong \frac{f_4 - 2f_5 + f_6}{h^2}$$

Calculation at Point 6

Computational Methods in EE

```
fd(1) = (2*f(1) - 5*f(2) + 4*f(3) - f(4))/h^2;
for nx = 2 : Nx-1
   fd(nx) = (f(nx-1) - 2*f(nx) + f(nx+1))/h^2;            nx = 6
end
fd(Nx) = (-f(Nx-3) + 4*f(Nx-2) - 5*f(Nx-1) + 2*f(Nx))/h^2;
```

```
fd(6) = (f(5) - 2*f(6) + f(7))/h^2;
```



$$[x] = [-h \quad 0 \quad h]^T$$

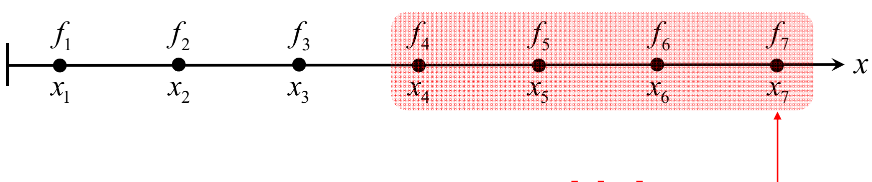
$$\frac{d^2 f_6}{dx^2} \cong \frac{f_5 - 2f_6 + f_7}{h^2}$$

Calculation at Point 7

Computational Methods in EE

```
fd(1) = (2*f(1) - 5*f(2) + 4*f(3) - f(4))/h^2;
for nx = 2 : Nx-1
   fd(nx) = (f(nx-1) - 2*f(nx) + f(nx+1))/h^2;
end
fd(Nx) = (-f(Nx-3) + 4*f(Nx-2) - 5*f(Nx-1) + 2*f(Nx))/h^2;
```

```
fd(7) = (-f(4) + 4*f(5) - 5*f(6) + 2*f(7))/h^2;
```



$$[x] = [-3h \quad -2h \quad -h \quad 0]^T$$

$$\frac{d^2 f_7}{dx^2} \cong \frac{-f_4 + 4f_5 - 5f_6 + 2f_7}{h^2}$$

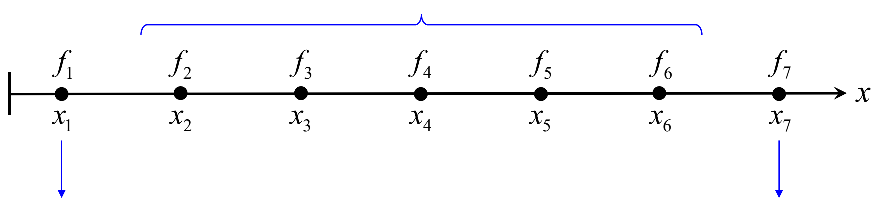
Boundary Conditions

High-Order Boundary Conditions

Computational Methods in EE

Here we estimate the derivative at the boundaries using special finite-difference equations derives specifically for these points.

$$\frac{d^2 f_i}{dx^2} \cong \frac{f_{i-1} - 2f_i + f_{i+1}}{h^2}$$



$$\frac{d^2 f_1}{dx^2} \cong \frac{2f_1 - 5f_2 + 4f_3 - f_4}{h^2}$$

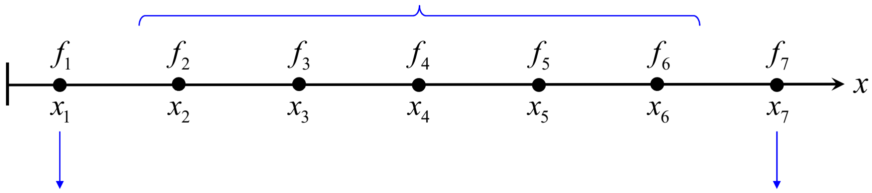
$$\frac{d^2 f_7}{dx^2} \cong \frac{-f_4 + 4f_5 - 5f_6 + 2f_7}{h^2}$$

Dirichlet Boundary Conditions

Computational Methods in EE

The simplest boundary condition is to assume all function values outside of the grid are zero.

$$\frac{d^2 f_i}{dx^2} \cong \frac{f_{i-1} - 2f_i + f_{i+1}}{h^2}$$



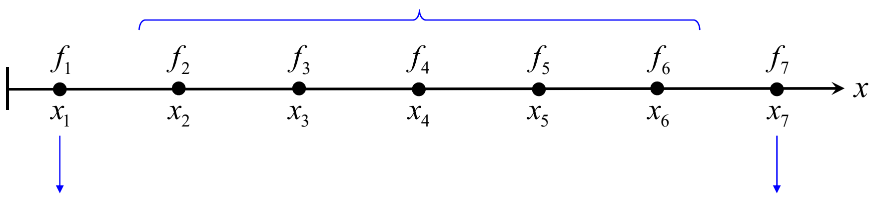
$$\frac{d^2 f_1}{dx^2} \cong \frac{0 - 2f_1 + f_2}{h^2} \qquad \frac{d^2 f_7}{dx^2} \cong \frac{f_6 - 2f_7 + 0}{h^2}$$

Periodic Boundary Conditions

Computational Methods in EE

If the problem is periodic (i.e. keeps repeating), then the value outside of the grid is the same as the value at the opposite side of the grid.

$$\frac{d^2 f_i}{dx^2} \cong \frac{f_{i-1} - 2f_i + f_{i+1}}{h^2}$$



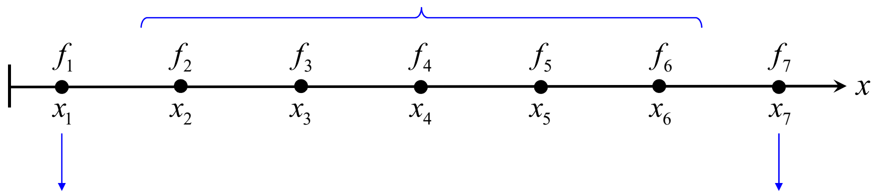
$$\frac{d^2 f_1}{dx^2} \cong \frac{f_7 - 2f_1 + f_2}{h^2} \qquad \frac{d^2 f_7}{dx^2} \cong \frac{f_6 - 2f_7 + f_1}{h^2}$$

Neuman Boundary Conditions

Computational Methods in EE

The Neuman boundary condition allows functions to continue linearly off of the grid as if to infinity.

$$\frac{df_i}{dx} \cong \frac{f_{i+1} - f_{i-1}}{2h} \qquad \frac{d^2 f_i}{dx^2} \cong \frac{f_{i-1} - 2f_i + f_{i+1}}{h^2}$$



$$\frac{df_1}{dx} \cong \frac{f_2 - f_1}{h} \qquad \frac{d^2 f_1}{dx^2} \cong 0 \qquad \qquad \frac{df_7}{dx} \cong \frac{f_7 - f_6}{h} \qquad \frac{d^2 f_7}{dx^2} \cong 0$$