Introduction

- In this chapter we investigate methods for solving systems of nonlinear equations of the form:

$$\mathbf{F}(\mathbf{x}) = \mathbf{0}, \tag{1}$$

where $\mathbf{F} : (D \subset \mathbb{R}^N) \to \mathbb{R}^N$.

- As a motivating example, consider the following nonlinear conservation equation on the interval $[0, L]$:

$$\frac{\partial \Psi(\varphi)}{\partial t} - \frac{\partial}{\partial x}\left(D(\varphi)\frac{\partial \varphi}{\partial x}\right) = S(\varphi), \quad 0 \le x \le L, \quad t > 0, \tag{2}$$

subject to boundary conditions

$$\varphi(0, t) = C_0, \quad \varphi(L, t) = C_L, \quad t > 0, \tag{3}$$

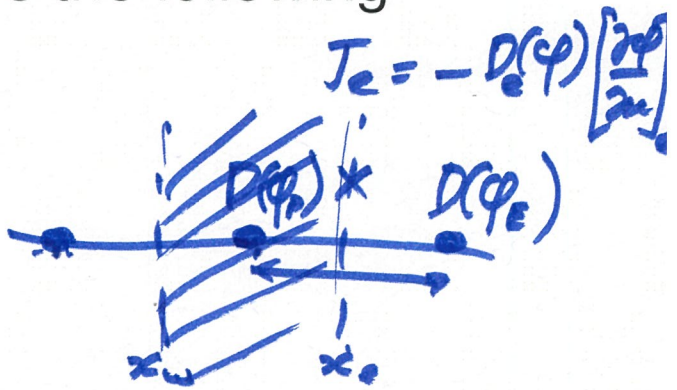
and the initial condition

$$\varphi(x, 0) = \varphi_0(x), \quad 0 \le x \le L. \tag{4}$$

Introduction

- Using the backward Euler method and arithmetic averaging of the diffusivities at the control volume faces produces the following finite volume discretisation of (2)–(4):
  - Left boundary node ($x = 0$)

$$\varphi_P = C_0$$



  - Internal nodes ($0 < x < L$)

$$\Psi(\varphi_P^{n+1}) - \Psi(\varphi_P^n) - \frac{\delta t}{\Delta x_P}\left[\frac{D(\varphi_P^{n+1}) + D(\varphi_E^{n+1})}{2}\left(\frac{\varphi_E^{n+1} - \varphi_P^{n+1}}{\delta x_e}\right) - \frac{D(\varphi_P^{n+1}) + D(\varphi_W^{n+1})}{2}\left(\frac{\varphi_P^{n+1} - \varphi_W^{n+1}}{\delta x_w}\right)\right] = S(\varphi_P^{n+1}).$$

  - Right boundary node ($x = L$)

$$\varphi_P = C_L$$

where $\varphi_P^0 = \varphi_0(x_P)$.

Introduction

$$\underline{F}(\underline{u}^{(n+1)}) = \underline{0} \leftarrow \text{Solve at every time step.}$$

- Introducing the index notation $\varphi_P = \varphi_i$, $\varphi_E = \varphi_{i+1}$ and $\varphi_W = \varphi_{i-1}$, and writing the numerical solution as the vector $\mathbf{u}^{(n+1)} = (\varphi_1^{n+1}, \ldots, \varphi_N^{n+1})^T$ we can express the above discretised equations in terms of the following functions:

$$\begin{cases} f_1(\mathbf{u}^{(n+1)}) = \varphi_1^{n+1} - C_0, & f_1 : D_1 \subset \mathbb{R}^N \longrightarrow \mathbb{R} \quad \varphi_{i-1}, \varphi_i, \varphi_{i+1} \\ f_i(\mathbf{u}^{(n+1)}) = \Psi(\varphi_i^{n+1}) - \Psi(\varphi_i^n) - \dfrac{\delta t}{\Delta x_i}\left[\dfrac{D(\varphi_i^{n+1}) + D(\varphi_{i+1}^{n+1})}{2}\left(\dfrac{\varphi_{i+1}^{n+1} - \varphi_i^{n+1}}{\delta x_e}\right) - \right. \\ \qquad \left. \dfrac{D(\varphi_i^{n+1}) + D(\varphi_{i-1}^{n+1})}{2}\left(\dfrac{\varphi_i^{n+1} - \varphi_{i-1}^{n+1}}{\delta x_w}\right)\right] - S(\varphi_P^{n+1}), i = 2, \ldots, N-1, \\ f_N(\mathbf{u}^{(n+1)}) = \varphi_N^{n+1} - C_L, \end{cases}$$

- Each function f_i maps the vector $\mathbf{u}^{(n+1)}$ in $\mathbb{R}^N$ to a scalar in $\mathbb{R}$.

$$\underline{F} : D \subset \mathbb{R}^N \longrightarrow \mathbb{R}^N.$$

Introduction

- To advance the numerical solution from $t = t_n$ to $t = t_{n+1} = t_n + \delta t$ we need to solve a system of nonlinear equations (1):

$$\mathbf{F}(\mathbf{u}^{(n+1)}) = \mathbf{0}\,,$$

where the *vector-valued* function

$$\mathbf{F}(\mathbf{u}^{(n+1)}) = \left(f_1(\mathbf{u}^{(n+1)}), \ldots, f_N(\mathbf{u}^{(n+1)})\right)^T.$$

- Observe that the function $\mathbf{F}$ maps the vector $\mathbf{u}^{(n+1)}$ in $\mathbb{R}^N$ to another vector in $\mathbb{R}^N$.

Newton's Method

- Recall Newton's method for solving a nonlinear equation of a single variable

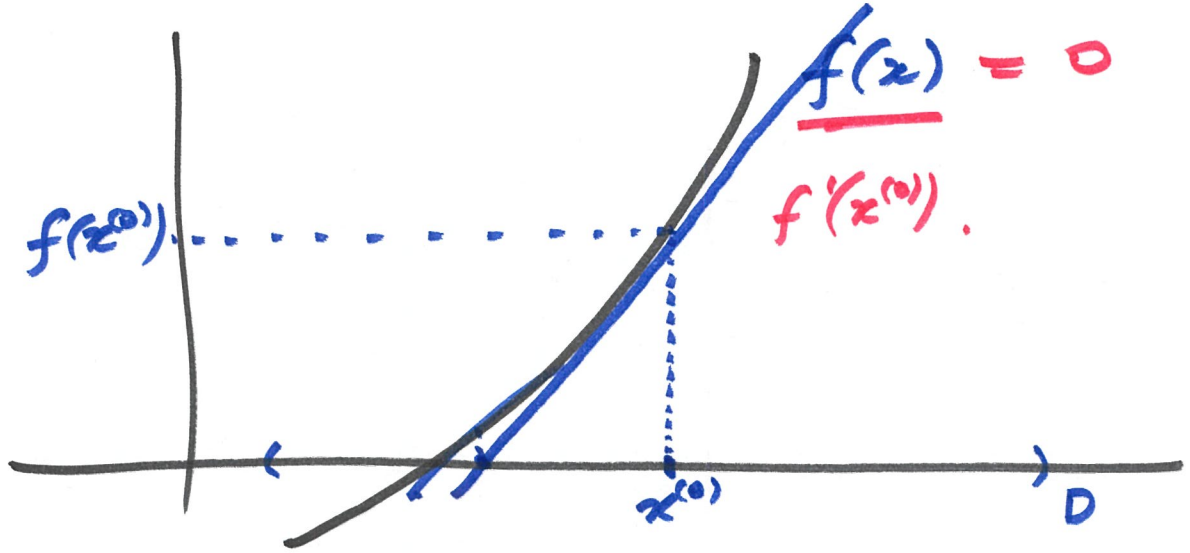
$$f(x) = 0\,,$$

where $f : (D \subset \mathbb{R}) \to \mathbb{R}$.

- Given an initial iterate $x^{(0)}$, the Newton iteration takes the form:

$$x^{(k+1)} = x^{(k)} - \frac{f(x^{(k)})}{f'(x^{(k)})}, k = 0, 1, \ldots k_{max} \tag{5}$$

$x^{(k)} \longrightarrow \xi$



x_0
$f(x)$
$f'(x)$

$M_1(x) = \underline{A} + \underline{B}\,x$

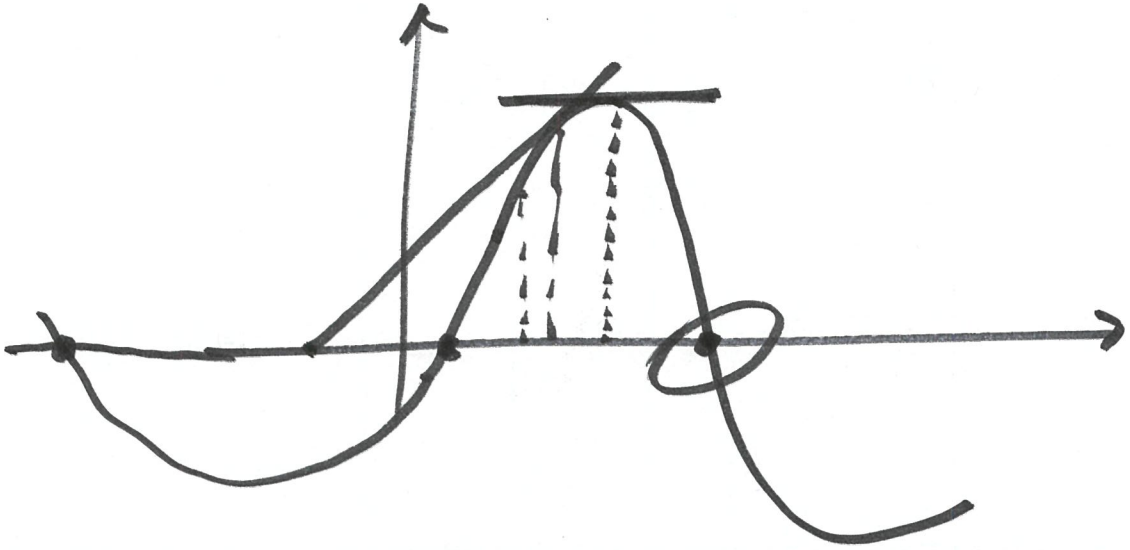
$M_1(x_0^{(0)}) = A + B x_0^{(0)} = f_0^{(0)}$

$M_1'(x) = B = f'(x^{(0)})$.

$$M_1(x) = f(x^{(0)}) + f'(x^{(0)})(x - x^{(0)}) = 0$$

$$x^{(1)} = x^{(0)} - f(x^{(0)}) / f'(x^{(0)})$$

Solve



$x^{(0)}, x^{(1)}, x^{(2)}, \ldots\ldots$

$$\{x^{(k)}\}_{k=1}^{\infty}$$

$|f(x^{(k)})| < \tau_1$,

$|x^{(k+1)} - x^{(k)}| < \tau_2$

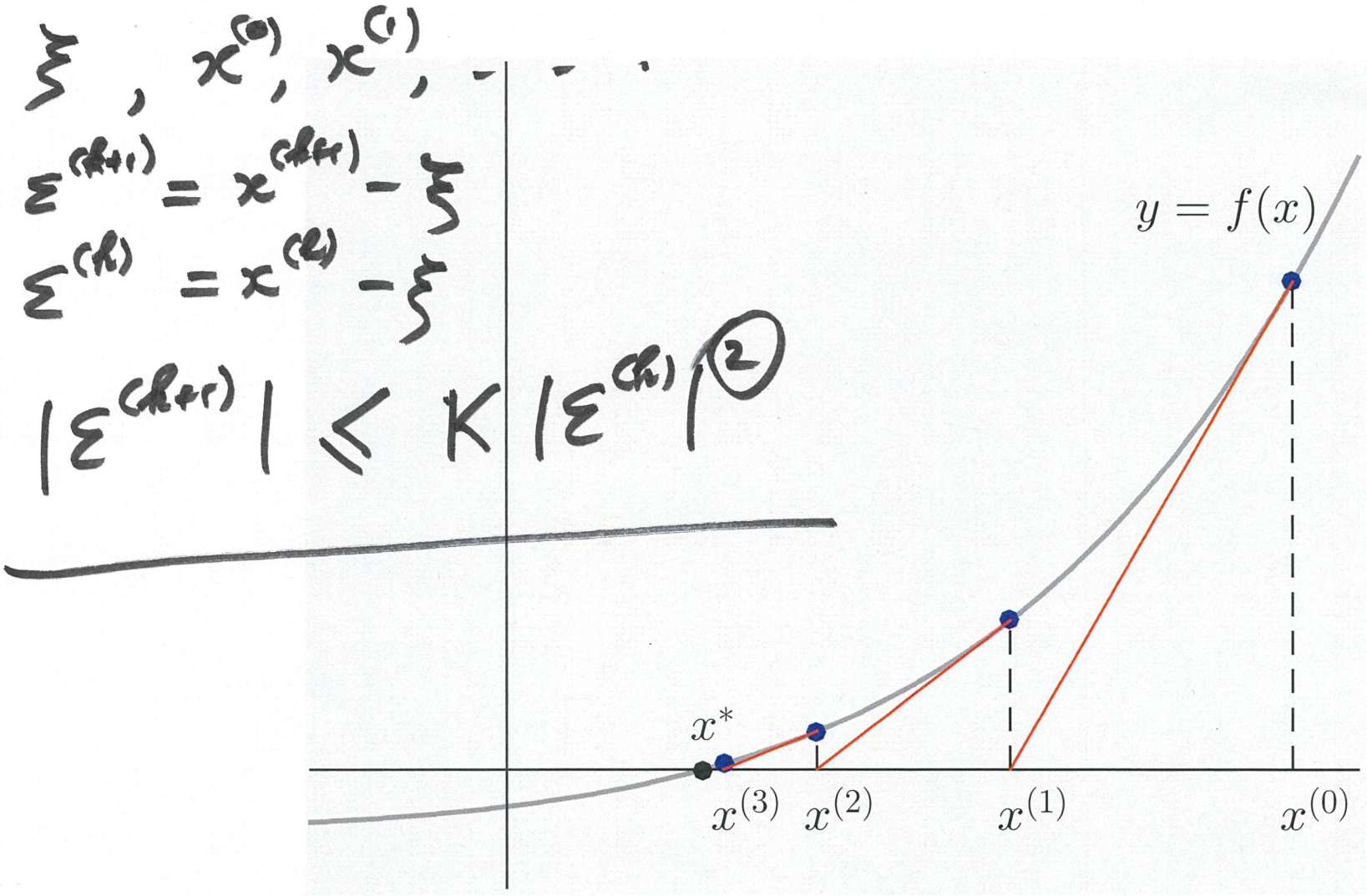


Figure: Schematic of Newton's method for $f(x) = 0$ showing convergence to the root x^*.

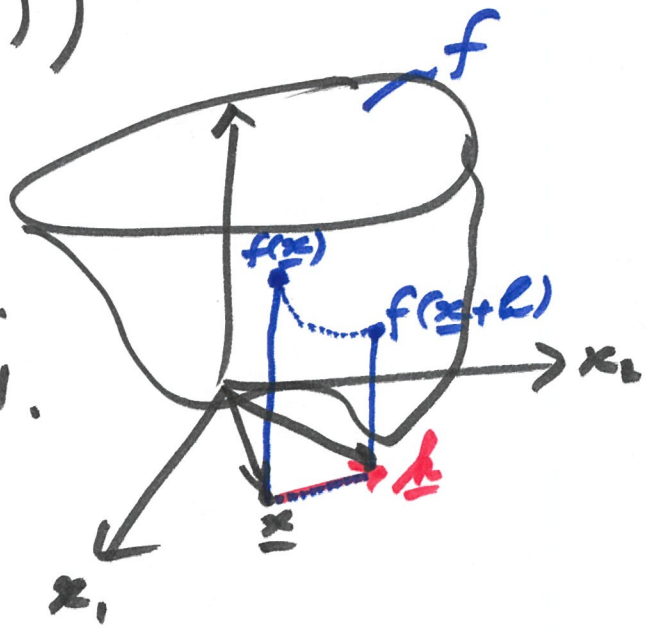
Newton's Method offers QUADRATIC (2nd order) convergence

How do we generalise this

To $\underline{F}(\underline{x}) = \underline{0}$, $\underline{F} : D \subset \mathbb{R}^N \to \mathbb{R}^N$.

$\underline{F}(\underline{x}) = \left(f_1(\underline{x}), f_2(\underline{x}), \ldots, f_N(\underline{x}) \right)^T$

Coordinate $f_i : D_i \subset \mathbb{R}^N \to \mathbb{R}$.



First consider, $g(t) = f(\underbrace{\underline{x} + t\underline{h}}_{\underline{r}(t)})$.

$g(0) = f(\underline{x})$, $g(1) = f(\underline{x} + \underline{h})$, $0 \le t \le 1$.

Newton's Th^m:

$$g(1) = g(0) + \int_0^1 g'(t)\, dt .$$

Recall $g'(t) = \nabla f(\underline{x} + t\underline{h}) \cdot \underline{h}$ chain rule.

$$f(\underline{x} + \underline{h}) = f(\underline{x}) + \int_0^1 \nabla f(\underline{x} + t\underline{h})^T \underline{h}\, dt .$$

$$\underline{F}(\underline{x}+\underline{h}) = \begin{pmatrix} f_1(\underline{x}+\underline{h}) \\ f_2(\underline{x}+\underline{h}) \\ \vdots \\ f_i(\underline{x}+\underline{h}) \\ \vdots \\ f_N(\underline{x}+\underline{h}) \end{pmatrix} = \begin{pmatrix} f_1(\underline{x}) + \int_0^1 \nabla f_1(\underline{x}+t\underline{h})^T \underline{h}\, dt \\ f_2(\underline{x}) + \int_0^1 \nabla f_2(\underline{x}+t\underline{h})^T \underline{h}\, dt \\ \vdots \\ f_N(\underline{x}) + \int_0^1 \nabla f_N(\underline{x}+t\underline{h})^T \underline{h}\, dt \end{pmatrix}$$

$$= \underbrace{\begin{pmatrix} f_1(\underline{x}) \\ f_2(\underline{x}) \\ \vdots \\ f_N(\underline{x}) \end{pmatrix}}_{\underline{F}(\underline{x})} + \begin{pmatrix} \int_0^1 \nabla f_1(\underline{x}+t\underline{h})^T \underline{h}\, dt \\ \vdots \\ \int_0^1 \nabla f_N(\underline{x}+t\underline{h})^T \underline{h}\, dt \end{pmatrix}$$

$$= \underline{F}(\underline{x}) + \int_0^1 \begin{pmatrix} \nabla f_1(\underline{x}+t\underline{h})^T \underline{h} \\ \vdots \\ \nabla f_N(\underline{x}+t\underline{h})^T \underline{h} \end{pmatrix} dt \quad \left(\begin{matrix} \text{Matrix} \\ \text{calculus} \end{matrix}\right)$$

Recall, $\underbrace{\nabla f_1(\underline{x}+t\underline{h})^T \underline{h}}$

$$\left(\frac{\partial f_1}{\partial x_1}(\underline{x}+t\underline{h}) \quad \frac{\partial f_1}{\partial x_2}(\underline{x}+t\underline{h}) \quad \cdots \quad \frac{\partial f_1}{\partial x_N}(\underline{x}+t\underline{h}) \right) \begin{pmatrix} h_1 \\ h_2 \\ \vdots \\ h_N \end{pmatrix} \underline{h}$$

This is the first row of the Jacobian matrix $\times \underline{h}$

$\underline{x}^{(k+1)} = \underline{x}^{(k)} + \underline{h}$,

i.e. $\underline{F}(\underline{x}+\underline{h}) \approx 0$

$$\underline{F}(\underline{x}+\underline{h}) = \underline{F}(\underline{x}) + \int_0^{\cdot} J_F(\underline{x}+t\underline{h})\,\underline{h}\,dt \approx 0$$

$\approx J_F(\underline{x})\underline{h}$

$$J_F = \begin{pmatrix} \frac{\partial f_1}{\partial x_1} & \frac{\partial f_1}{\partial x_2} & \cdots & \frac{\partial f_1}{\partial x_N} \\ \vdots & & & \vdots \\ \frac{\partial f_N}{\partial x_1} & - & - & \frac{\partial f_N}{\partial x_N} \end{pmatrix}$$

$\underline{F}(\underline{x}^{(k)}) + J_F(\underline{x}^{(k)})\underline{h} \approx 0$

Solve

$\boxed{J_F(\underline{x}^{(k)})\underline{h} = -\underline{F}(\underline{x}^{(k)})}$

$\underline{h} = -J_F^{-1}(\underline{x}^{(k)})\,\underline{F}(\underline{x}^{(k)})$

(Full) Newton Method

- The computational framework presented above can be extended to higher dimensions. Given an initial iterate $\mathbf{x}^{(0)}$, Newton's method for solving (1) takes the form:

$$\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} - \mathbf{J}\left(\mathbf{x}^{(k)}\right)^{-1} \mathbf{F}\left(\mathbf{x}^{(k)}\right), \tag{6}$$

where $\mathbf{J} \in \mathbb{R}^{N \times N}$ is the Jacobian matrix of $\mathbf{F}$:

$$\mathbf{J}(\mathbf{x}) = \begin{bmatrix} \nabla f_1(\mathbf{x})^T \\ \nabla f_2(\mathbf{x})^T \\ \vdots \\ \nabla f_N(\mathbf{x})^T \end{bmatrix} = \begin{bmatrix} \frac{\partial f_1(\mathbf{x})}{\partial x_1} & \frac{\partial f_1(\mathbf{x})}{\partial x_2} & \cdots & \frac{\partial f_1(\mathbf{x})}{\partial x_N} \\ \frac{\partial f_2(\mathbf{x})}{\partial x_1} & \frac{\partial f_2(\mathbf{x})}{\partial x_2} & \cdots & \frac{\partial f_2(\mathbf{x})}{\partial x_N} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial f_N(\mathbf{x})}{\partial x_1} & \frac{\partial f_N(\mathbf{x})}{\partial x_2} & \cdots & \frac{\partial f_N(\mathbf{x})}{\partial x_N} \end{bmatrix}.$$

(Full) Newton Method

An algorithm for Newton's method is summarised as follows.

Newton's method

Choose $\mathbf{x}^{(0)}$

While not converged for $k = 0, 1, \ldots$

Solve $\mathbf{J}(\mathbf{x}^{(k)})\delta\mathbf{x}^{(k)} = -\mathbf{F}(\mathbf{x}^{(k)})$ for $\delta\mathbf{x}^{(k)}$

$\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + \delta\mathbf{x}^{(k)}$

End

$\|E(\underline{x}^{(k+1)})\|_{\infty} < \varepsilon$

$$\|\underline{x}\|_p = \left[\sum_{i=1}^{N} |x_i|^p\right]^{1/p},$$

$p = 1$: $\|\underline{x}\|_1 = \sum_{i=1}^{N} |x_i|$

$p = 2$: $\|x\|_2 = \sqrt{\sum_{i=1}^{N} |x_i|^2}$

$p = \infty$: $\|\underline{x}\|_\infty = \max_{1 \le i \le N} |x_i|.$

Terminating Newton's Method

- One common criterion used to terminate Newton's method is based on the norm of the nonlinear residual $\|\mathbf{F}(\mathbf{x}^{(k+1)})\|$.
- Typically, a combination of both a **relative error tolerance** τ_r and **absolute error tolerance** τ_a is used to halt the iteration:

$$\left\|\mathbf{F}\left(\mathbf{x}^{(k+1)}\right)\right\| \leq \tau_r \left\|\mathbf{F}\left(\mathbf{x}^{(0)}\right)\right\| + \tau_a. \tag{7}$$

Chord Method

- The requirement to compute the Jacobian matrix and solve a linear system at each iteration of Newton's method makes it potentially very expensive.
- If using a direct solver, this would require forming and factorising the Jacobian matrix once for each Newton step.
- The **Chord method** is an alternative method that forms and factorises the Jacobian only once, at the beginning of the iteration. In this case, the iteration is given by:

$$\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + \delta\mathbf{x}^{(k)}, \quad \delta\mathbf{x}^{(k)} = -\mathbf{J}\left(\mathbf{x}^{(0)}\right)^{-1} \mathbf{F}\left(\mathbf{x}^{(k)}\right). \qquad (8)$$

Chord Method

- In practice, a very good initial iterate is usually required for the Chord method to converge.
- If the initial iterate $\mathbf{x}^{(0)}$ is nothing like the solution, then the Chord steps $\delta\mathbf{x}^{(k)} = -\mathbf{J}\left(\mathbf{x}^{(0)}\right)^{-1}\mathbf{F}\left(\mathbf{x}^{(k)}\right)$ will quickly degrade, since the Jacobian matrix is not being updated at each iteration.

Chord Method

An algorithm for the Chord method is summarised as follows.

Chord method

Choose $\mathbf{x}^{(0)}$ and set $\overline{\mathbf{J}} = \mathbf{J}(\mathbf{x}^{(0)})$

While not converged for $k = 0, 1, \ldots$

Solve $\overline{\mathbf{J}}\delta\mathbf{x}^{(k)} = -\mathbf{F}(\mathbf{x}^{(k)})$ for $\delta\mathbf{x}^{(k)}$

$\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + \delta\mathbf{x}^{(k)}$

End

Shamanskii Method

Alternatively, the Jacobian matrix may be *periodically* updated. For example, alternating a full Newton step with a sequence of Chord steps leads to a method often attributed to Shamanskii.

Shamanskii method

Choose $\mathbf{x}^{(0)}$

While not converged for $k = 0, 1, \ldots$

    If $k \equiv 0 \ (mod\ m)$

        Compute/Update $\overline{\mathbf{J}} = \mathbf{J}(\mathbf{x}^{(k)})$

    End

    Solve $\overline{\mathbf{J}}\delta\mathbf{x}^{(k)} = -\mathbf{F}(\mathbf{x}^{(k)})$ for $\delta\mathbf{x}^{(k)}$

    $\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + \delta\mathbf{x}^{(k)}$

End

Shamanskii Method

- **Note:** The choice $m = 1$ recovers Newton's method, and as $m \to \infty$ the Chord method is recovered.
- In more sophisticated algorithms the test $k \equiv 0 \pmod{m}$ is replaced with a test based on how slowly or rapidly the method is converging.
- If the method appears to be converging too slowly (based on the rate at which the nonlinear residual is going to zero), the algorithm triggers the Jacobian matrix to be updated.
- If the speed of convergence is satisfactory, the Jacobian matrix can be kept "out of date" for at least another iteration.

Example 1

Develop a code in MATLAB that performs the different variants of Newton's method (Full, Chord, Shamanskii) to solve the following nonlinear boundary-value problem:

B.V.P.

$$y'' + \frac{1}{8}yy' = (4 + \frac{1}{4}x^3), \ 1 \le x \le 3$$

$$y(1) = 17, \quad y(3) = \frac{43}{3}.$$

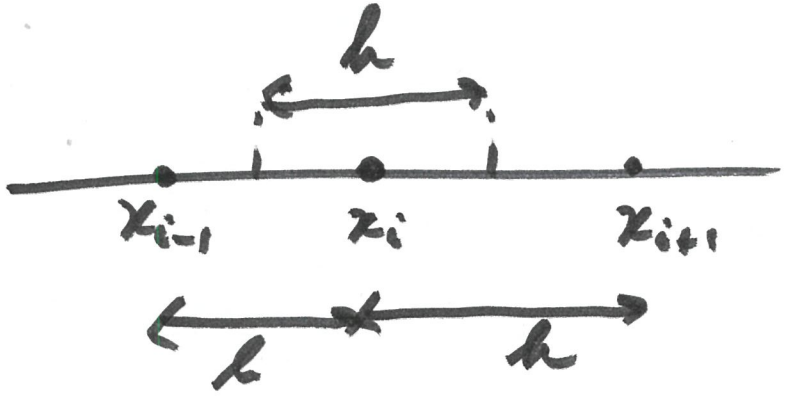
$\frac{\partial T}{\partial x} = S(x)$

The exact solution of this problem is $y(x) = x^2 + 16/x$.

$$\frac{d}{dx}\underbrace{\left[y' + \frac{y^2}{16}\right]}_{J} = S(x), \quad S(x) = 4 + \frac{x^3}{4}.$$

Finite Volume Method

$\underline{y} = (y_1, y_2, \ldots, y_N)^T$.



uniform grid.

$$f_1(\underline{y}) := y_1 - 17$$

$$f_i(\underline{y}) := \underbrace{\frac{y_{i+1} - y_i}{h} + \frac{h}{16}\left(\frac{y_{i+1} + y_i}{2}\right)^2}_{J_e} - \underbrace{\left[\frac{y_i - y_{i-1}}{h} + \frac{h}{16}\left(\frac{y_i + y_{i-1}}{2}\right)^2\right]}_{J_w} - h^2 S(x_i)$$

$$= y_{i+1} - 2y_i + y_{i-1} + \frac{h}{64}\left[(y_{i+1} + y_i)^2 - (y_i + y_{i-1})^2\right] - h^2 S(x_i) \qquad i = 2, \ldots, N-1.$$

$$f_N(\underline{y}) = y_N - 43/3$$

$$S(x_i) = 4 + \frac{x_i^3}{4}.$$

$$\underline{F}(\underline{x}) = \left(f_1(\underline{x}), f_2(\underline{x}), \dots, f_N(\underline{x})\right)^T$$

$$J_F(\underline{x}) = \begin{pmatrix} \frac{\partial f_1}{\partial y_1} & \frac{\partial f_1}{\partial y_2} & \cdots & \frac{\partial f_1}{\partial y_N} \\ \frac{\partial f_2}{\partial y_1} & \frac{\partial f_2}{\partial y_2} & \cdots & \frac{\partial f_2}{\partial y_N} \\ \vdots & & & \\ \frac{\partial f_N}{\partial y_1} & \frac{\partial f_N}{\partial y_2} & \cdots & \frac{\partial f_N}{\partial y_N} \end{pmatrix} \in \mathbb{R}^{N \times N}$$

$$= \begin{pmatrix} 1 & 0 & \cdots & 0 \\ \frac{\partial f_i}{\partial y_{i-1}} & \frac{\partial f_i}{\partial y_i} & \frac{\partial f_i}{\partial y_{i+1}} & \text{, } i = 2, \dots N-1. \\ 0 & \cdots & & 1 \end{pmatrix}$$

$$\frac{\partial f_i}{\partial y_{i-1}} = 1 * - \frac{h}{32}(y_i + y_{i-1})$$

$$\frac{\partial f_i}{\partial y_i} = -2 + \frac{h}{32}\left[(y_{i+1} + y_i) - (y_i + y_{i-1})\right]$$

$y_{i+1} - y_{i-1}$

$$\frac{\partial f_i}{\partial y_{i+1}} = 1 + \frac{h}{32}(y_{i+1} + y_i)$$