So far: **ODE problem**

$$\begin{cases} u'(t) = \mathcal{F}(t, u(t)) \\ u(0) = u_0 \end{cases}$$

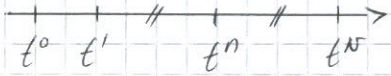
for linear ODEs

$$\mathcal{F}(t, u(t)) = A(t) \cdot u(t) + c(t)$$

Time discretisation

$$\Delta t > 0$$

$$t^n = n \Delta t \qquad \text{(for equally spaced intervals)}$$



where

$\{t^n\}_{n=0}^{N}$ are our time steps

and

t^n refers to the n^{th} time level

Finite Difference Methods

- Forward Euler

$$u'(t) \approx \frac{u(t^{n+1}) - u(t^n)}{\Delta t}$$

- Backward Euler

$$u'(t) \approx \frac{u(t^n) - u(t^{n-1})}{\Delta t}$$

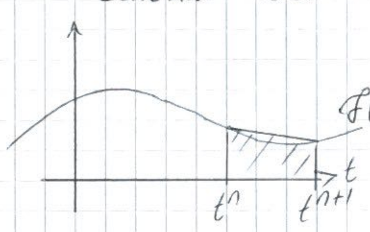
$\longrightarrow$ Implementation

FE: $U^{n+1} = U^n + \Delta t \cdot \mathcal{F}(t^n, U^n)$ explicit scheme 'easy'

BE: $U^{n+1} = U^n + \Delta t \cdot \mathcal{F}(t^{n+1}, U^{n+1})$ implicit scheme 'harder' $\rightarrow$ advantage: stability

Trapezoidal Rule

- General idea: $u(t^{n+1}) - u(t^n) = \int_{t^n}^{t^{n+1}} u'(s)\, ds$ (Fundamental Theorem of Calculus)



$$\overset{\text{by ODE}}{=} \underbrace{\int_{t^n}^{t^{n+1}} \mathcal{F}(s, u(s))\, ds}_{\text{for general } \mathcal{F} \,\to\, \text{approximate}}$$

Since we want to integrate: approximate with numerical quadrature

$$\int_a^b f(x)\, dx \approx (b-a) \frac{f(a)+f(b)}{2}$$

So between two consecutive timesteps:

$$u(t^{n+1}) - u(t^n) \approx \underbrace{(t^{n+1} - t^n)}_{\Delta t} \left(\frac{\mathcal{F}(t^n, u(t^n)) + \mathcal{F}(t^{n+1}, u(t^{n+1}))}{2} \right)$$

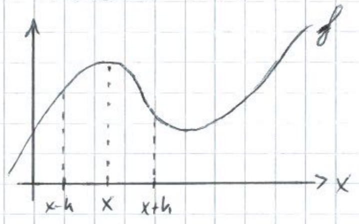
Approx. the solution $U^n \approx u(t^n) \quad \forall n \in \{0, 1, .., N-1\}$

yields: $U^{n+1} = U^n + \frac{\Delta t}{2}\left(\mathcal{F}(t^n, U^n) + \mathcal{F}(t^{n+1}, U^{n+1}) \right)$

$\to$ implicit method

Midpoint

- General idea:



Question: $f'(x) = ?$ (We know $f'(x)$ should be zero)

i.e. FE is derived in $x+h$ for x
BE is derived in $x-h$ for x

$\to$ approximate with both

Say: $f'(x) \approx \frac{1}{2h}\left(f(x+h) - f(x-h)\right)$ and use this for $u'(t)$

$$\left.\begin{aligned} \underbrace{u'(t^n)}_{= \mathcal{F}(t^n, u(t^n)) \approx \mathcal{F}(t^n, U^n)} &\approx \frac{1}{2\Delta t}\left(u(t^{n+1}) - u(t^{n-1})\right) \overset{\text{now } U}{\approx} \frac{1}{2\Delta t}\left(U^{n+1} - U^{n-1}\right) \end{aligned}\right\} n \in \{0, .., N-1\}$$

$$\Rightarrow \quad U^{n+1} - U^{n-1} = 2\Delta t\, \mathcal{F}(t^n, U^n) \quad \forall n \in \{1, \dots, N-1\}$$

$\to$ explicit, multistep method (i.e. to compute U^{n+1} we need U^n, U^{n-1} $\Rightarrow U^1$ has t.b. defined)

So let's formulate an algorithm for this:

Set $U^0 = u_0$

$U^1 = U^0 + \Delta t \cdot \mathcal{F}(0, U^0)$ (as an assumption)

For $n = 1, \dots, N-1$

$U^{n+1} = U^{n-1} + 2\Delta t\, \mathcal{F}(t^n, U^n)$

But how about implicit methods?

- Trapezoidal: $U^{n+1} = U^n + \frac{\Delta t}{2}\left(\mathcal{F}(t^n, U^n) + \mathcal{F}(t^{n+1}, U^{n+1})\right)$

  for each time step t^n we solve for U^{n+1}

  $$0 = U^{n+1} - U^n - \frac{\Delta t}{2}\left(\mathcal{F}(t^n, U^n) + \mathcal{F}(t^{n+1}, U^{n+1})\right) = G(U^{n+1})$$

  which we can solve with the Newton Method

Newton Method

For a system of eq'ns i.e.

$$G(Y) = 0 .$$

we approximate the derivation by the Jacobian Matrix J

$$J(Y) = \begin{pmatrix} \frac{\partial g_1}{\partial y_1} & \cdots & \frac{\partial g_1}{\partial y_N} \\ \frac{\partial g_N}{\partial y_1} & \cdots & \frac{\partial g_N}{\partial y_N} \end{pmatrix}$$

Now, we iterate for our Y.

In order to do so, we guess an initial solution $Y^{(k)}$

similar to initial values: $Y^{k=0} = U^n$

OR $Y^{k=0} = U^n + \Delta t\, \mathcal{F}(t^n, U^n)$

then:

$$Y^{(k+1)} = Y^{(k)} - J^{-1}(Y^{(k)}) \cdot G(Y^{(k)})$$

where we avoid inverting the Jacobian and instead solve

$$J(Y^{(k)})\, X^{(k)} = -G(Y^{(k)})$$

for $X^{(k)}$, and update

$$Y^{(k+1)} = Y^{(k)} + X^{(k)}$$

until we are in our set tolerance: Error estimate: $E_{k+1} = Y^{(k+1)} - Y^{(k)}$

Assuming we computed an approximation numerically i.e. U^n and we compare it to the true solution $u(t^n)$:

Global Error (at final time) $E_N = |u(t^N) - U^N|$

i.e. the 'sum' of all errors made at each time level.

One might ask: "How well does this scheme (FE, BE, Midpoint, ...) agree with the ODE?"

Truncation Error

1^{st} write the scheme in its consistent form i.e. $Q(U^{n+1}, U^n) = 0$

2^{nd} insert the exact solutions $u(t^{n+1}), u(t^n)$ into Q

Ex. Forward Euler $\quad U^{n+1} = U^n + \Delta t\, f(t^n, U^n)$

1^{st}: $\quad \frac{1}{\Delta t}(U^{n+1} - U^n) - f(t^n, U^n) = 0$

2^{nd}: $\quad \frac{1}{\Delta t}(u(t^{n+1}) - u(t^n)) - f(t^n, u(t^n)) = T_n$

taylor expand $u(t^{n+1}) \overset{\text{around } t^n}{=} u(t^n + \Delta t) = u(t^n) + u'(t^n)\underbrace{\Delta t}_{t^{n+1}-t^n} + \frac{u''(t^n)}{2}\Delta t^2 + O(\Delta t^3)$

Hence:

$$T_n = \frac{1}{\cancel{\Delta t}}\left(\cancel{u(t^n)} + u'(t^n)\cancel{\Delta t} + u''(t^n)\frac{\Delta t^{\cancel{2}}}{2} + O(\Delta t^3) - \cancel{u(t^n)}\right) - \underbrace{f(t^n, u(t^n))}_{\hat{=}\, u'(t^n)}$$

$$= \frac{u''(t^n)}{2}\Delta t + O(\Delta t^2)$$

$$\overset{\text{for small } \Delta t}{=} O(\Delta t)$$

The truncation error of the FE Method is 1^{st} order accurate.