Various error measures

Dream: the true error is the discrepancy that arises from approximating a process with infinitely many steps.

- Why? The truncation error is an error measure that is easy to compute.

Taylor series

General Taylor series expansion from calculus:

\[ f(x + h) = \sum_{n=0}^{\infty} \frac{d^n f}{dx^n}(c) h^n \]

Here: expand \( u^{(i)} \) around \( c \):

\[ u(t + \Delta t) = u(t) + u'(t)\Delta t + \sum_{i=2}^{\infty} \frac{1}{i!} u^{(i)}(t)\Delta t^i \]

\[ = u(t) - u'(t)\Delta t + \frac{1}{2} u''(t)\Delta t^2 + O(\Delta t^3) \]

- \( O(\Delta t^3) \): power-series in \( \Delta t \) where the lowest power is \( \Delta t^3 \)
- Small \( \Delta t \): \( \Delta t \gg \Delta t^3 \gg \Delta t^4 \)
Taylor series inserted in the backward difference approximation

\[
[D^- u]^n = \frac{u(t_n) - u(t_{n-1})}{\Delta t} - u(t_n)
\]

\[
\frac{u(t_n) - u(t_{n-1})}{\Delta t} - u(t_n) = \frac{1}{2}u''(t_n)(\Delta t)^2 + O(\Delta t^3)
\]

Result:

\[
R^n = \frac{1}{2}u''(t_n)(\Delta t)^2 + O(\Delta t^3)
\]

The difference approximation is of first order in \(\Delta t\). It is exact for linear \(u_e\).

The forward difference for \(u'(t)\) (1)

Now consider a forward difference:

\[
u'(t_n) = \frac{u(t_{n+1}) - u(t_n)}{\Delta t}
\]

Define the truncation error:

\[
R^n = [D^+ u]^n - u'(t_n)
\]

Expand \(u^{n+1}\) in a Taylor series around \(t_n\),

\[
u(t_{n+1}) = u(t_n) + u'(t_n)\Delta t + \frac{1}{2}u''(t_n)\Delta t^2 + O(\Delta t^3)
\]

We get

\[
R = \frac{1}{2}u''(t_n)\Delta t + O(\Delta t^2)
\]

Leading-order error terms in finite differences (1)

<table>
<thead>
<tr>
<th>Term</th>
<th>Coefficient</th>
</tr>
</thead>
<tbody>
<tr>
<td>([D^- u]^n)</td>
<td>(u^{n+1} - u^n) + (R^n)</td>
</tr>
<tr>
<td>([D^- u]^n)</td>
<td>(\frac{1}{2}u''(t_n)(\Delta t)^2 + O(\Delta t^3))</td>
</tr>
<tr>
<td>([D_2^- u]^n)</td>
<td>(u^{n+1} - u^n - u'(t_n) + R^n)</td>
</tr>
<tr>
<td>([D_2^- u]^n)</td>
<td>(\frac{1}{2}u''(t_n)(\Delta t)^2 + O(\Delta t^3))</td>
</tr>
<tr>
<td>([D_3^- u]^n)</td>
<td>(\frac{1}{6}u''''(t_n)(\Delta t)^3 + O(\Delta t^4))</td>
</tr>
<tr>
<td>([D_4^- u]^n)</td>
<td>(\frac{1}{2}u'''(t_n)(\Delta t)^2 + O(\Delta t^3))</td>
</tr>
<tr>
<td>([D_5^- u]^n)</td>
<td>(\frac{1}{2}u''''(t_n)(\Delta t)^2 + O(\Delta t^3))</td>
</tr>
</tbody>
</table>
Leading-order error terms in mean values (1)

Weighted arithmetic mean:
\[ [u_\text{w},\theta]_n = \theta u_{n+1} + (1-\theta)u_n = u(t_n + \theta \Delta t) + \cdots \]

Standard arithmetic mean:
\[ [u_\text{s}]_n = \frac{1}{2} (u_{n-1} + u_{n+1}) = u(t_n) + R_n \]

\[ R_n = \frac{1}{8} u''(t_n) \Delta t^2 + \frac{1}{384} u''''(t_n) \Delta t^4 + O(\Delta t^6) \]

Leading-order error terms in mean values (2)

Geometric mean:
\[ [u_\text{g}]_n = \frac{1}{2} (u_{n-1} + u_{n+1}) = u(t_n) + R_n \]

\[ R_n = -\frac{1}{2} u'(t_n) \Delta t^2 + \frac{1}{4} u(t_n)u'(t_n)\Delta t^2 + O(\Delta t^4) \]

Harmonic mean:
\[ [u_\text{h}]_n = \frac{2}{u'(t_n) + \frac{1}{\Delta t} u''(t_n) \Delta t} = u(t_n) + R_n \]

\[ R_n = -\frac{u(t_n)}{4u(t_n)\Delta t^2} + \frac{1}{8} u''(t_n) \Delta t^2 \]

Software for computing truncation errors

Can use Sympy to automate calculations with Taylor series.

A class DiffOp represents many common difference operators:
```python
>>> from truncation_errors import DiffOp
>>> u = symbols('u t')
>>> diffop = DiffOp(u, independent_variable='t')
>>> diffop['geometric_mean']
```

Names in `diffop`:
- `Dtp` for \( D^+ t \)
- `Dtm` for \( D^- t \)
- `Dt` for \( D_t \)
- `D2t` for \( D^2 t \)
- `DtDt` for \( D_t D_t \)

Tool: `truncation_errors` module from `truncation_errors.py`.

Symbolic computing with difference operators

A class DiffOp represents many common difference operators:
```python
>>> from truncation_errors import DiffOp
>>> u = symbols('u t')
>>> diffop = DiffOp(u, independent_variable='t')
>>> diffop['geometric_mean']
```

Names in `diffop`:
- `Dtp` for \( D^+ t \)
- `Dtm` for \( D^- t \)
- `Dt` for \( D_t \)
- `D2t` for \( D^2 t \)
- `DtDt` for \( D_t D_t \)

Truncation errors in exponential decay ODE

The Forward Euler scheme:
\[ u'(t) = -au(t) \]

Define the truncation error \( R_n \):
\[ u'(t) + au(t) = R_n \]

From (77)--(77):
\[ u'(t_n) + au(t_n) = R_n \]

Rewritten in (77):
\[ u'(t_n) + \frac{1}{2} u''(t_n) \Delta t + O(\Delta t^2) \]

Note: \( u''(t_n) + au(t_n) = 0 \) since \( u(t) \) solves the ODE.

Truncation error of the Forward Euler scheme

Define the truncation error \( R_n \):
\[ u'(t) + au(t) = R_n \]

From (77)--(77):
\[ u'(t_n) + \frac{1}{2} u''(t_n) \Delta t + O(\Delta t^2) + au(t_n) = R_n \]

Rewritten in (77):
\[ u'(t_n) + \frac{1}{2} u''(t_n) \Delta t + O(\Delta t^2) + au(t_n) = R_n \]

Note: \( u''(t_n) + au(t_n) = 0 \) since \( u(t) \) solves the ODE. Thus
\[ R_n = \frac{1}{2} u''(t_n) \Delta t + O(\Delta t^2) \]
Truncation error of the Crank-Nicolson scheme

Crank-Nicolson: \[ D_t u = -a u^{n+\frac{1}{2}} \]

Truncation error:
\[ [D_t u + au^{n+\frac{1}{2}} = O(\Delta t^{3/2}) \]

From (77)-(77) and (77)-(77):
\[ \frac{1}{2} \left( u_t(t_{n+\frac{1}{2}}) + u_t(t_n) \right) \Delta t^2 + O(\Delta t^3) \]
\[ \left[ \frac{1}{2} u_t(t_{n+\frac{1}{2}}) + \frac{1}{6} u_t(t_n) \right] \Delta t^2 + O(\Delta t^3) \]

Instead in the scheme we get:
\[ R^{n+\frac{1}{2}} = \left( \frac{1}{2} u_t(t_{n+\frac{1}{2}}) + \frac{1}{6} u_t(t_n) \right) \Delta t^2 + O(\Delta t^3) \]
\[ R^n = O(\Delta t^2) \] (second-order scheme)

Note: 2nd-order scheme if and only if \( \theta = \frac{1}{2} \).

Truncation error of the \( \theta \)-rule

The \( \theta \)-rule:
\[ \bar{D}_t u = -a u^\theta \]

Truncation error:
\[ [\bar{D}_t u + au^\theta = O(\Delta t^{3/2}) \]

Use (77)-(77) and (77)-(77) along with \( u_t(t_n) + au(t_n) = 0 \) to show:
\[ R^{n+\theta} = \left( \frac{1}{2} - \theta \right) u_t(t_{n+\theta}) \Delta t^2 + \frac{1}{2} \theta(1-\theta) u_t(t_n) \Delta t^2 + \frac{1}{2} (\theta^2 - \theta + 1) u_t(t_{n+\theta}) \Delta t^2 + O(\Delta t^3) \]
\[ \frac{1}{2} (\theta^2 - \theta + 3) u_t(t_n) \Delta t^2 + O(\Delta t^3) \] (1)

Note: 2nd-order scheme if and only if \( \theta = 1/2 \).

Empirical verification of the truncation error (1)

Meas:
- Compute \( R^n \)
- Put a sequence of meshes
- Estimate the convergence rate of \( R^n \)

For the Forward Euler scheme:
\[ R^n = [\bar{D}_t u + au^n] \]

Then correct \( u(t) = e^{-r t} \) (for use method of manufactured solution is more general case).

Test the understanding!

Analyze the the truncation error of the Backward Euler scheme and show that it is \( O(\Delta t) \) (first-order scheme).

Using symbolic software

Can use sympy and the tools in truncation_errors.py:

```python
def decay():
    n = 20
    return R

R = {'FE': FE-ODE, 'BE': BE-ODE, 'CN': CN-ODE}

print(R)
```

The returned dictionary becomes

```
{'FE': D1u + a*u, 'BE': D1u + a*u, 'CN': D1u + a*u}
```

\( \theta \)-rule: see truncation_errors.py (long expression, very

Empirical verification of the truncation error (2)

- Assume \( R^n = C \Delta t^r \)
- \( C \) and \( r \) will vary with \( n \) - must estimate \( r \) for each mesh point
- Use a sequence of meshes with \( N_j = 2^{-k} N_0 \) intervals,
  \( k = 1, 2, \ldots \)
- Transform \( R^n \) data to the coarsest mesh and estimate \( r \) for each coarse mesh point
- See text for more details and an implementation.
Empirical verification of the truncation error in the Forward Euler scheme

Figure: Estimated truncation error at mesh points for different meshes.

Increasing the accuracy by adding correction terms

Question
Can we add terms in the differential equation that can help increase the order of the truncation error?

To be precise for the Forward Euler scheme, can we find \( C \) so that

\[
[\Delta t] u_{n+1} + au_n = C + R_n^2. 
\]

Coeficienting

\[
\frac{1}{2} u''(t_n) \Delta t + \frac{1}{6} u'''(t_n) \Delta t^2 + O(\Delta t^3) = C + R_n^2. 
\]

Choosing

\[
C_n^2 - \frac{1}{2} u''(t_n) \Delta t 
\]

makes

\[
R_n^2 = \frac{1}{6} u'''(t_n) \Delta t^2 + O(\Delta t^3). 
\]

With a correction term Forward Euler becomes Crank-Nicolson

Use the order relation of \( u' = -au \):

\[
u' = -au - \frac{1}{2} a \Delta t u' 
\]

Apply Forward Euler:

\[
\left(1 + \frac{1}{2} a \Delta t\right) u^{n+1} - u^n = -a u^n 
\]

which, after some algebra can be written as

\[
\frac{u^{n+1} - \frac{1}{2} a \Delta t u^n}{1 - \frac{1}{2} a \Delta t} = -a u^n. 
\]

This is a Crank-Nicolson scheme (of second order)! 

Empirical verification of the truncation error in the Forward Euler scheme

Figure: Difference between theoretical and estimated truncation error at mesh points for different meshes.

Lowering the order of the derivative in the correction term

- \( C_n^n \) contains \( u'' \)
- Can decrease \( u'' \) (requires \( u''', u', \) and \( u''' \))
- Can also express \( u'' \) in terms of \( u' \) or \( u \)

\[
u' = -au, \quad u'' = -a u' = u_{n+1} - u_n. 
\]

Result for \( u'' = u_{n+1} - u_n \): apply Forward Euler to a perturbed ODE,

\[
u' = -au, \quad 3 = a(1 - \frac{1}{2} a \Delta t)
\]

so make a second-order scheme!

Correction terms in the Crank-Nicolson scheme [1]

\[
[D_t u = -au^n]^{n+\frac{1}{2}} 
\]

Definiton of the truncation error \( R \) and correction terms \( C \):

\[
[\Delta t] u_{n+1} + au_n = C + R_n^{n+\frac{1}{2}}. 
\]

Must Taylor expand

- the derivative
- the arithmetic mean

\[
C^{n+\frac{1}{2}} + R_{n+\frac{1}{2}} = \frac{1}{24} u''(t_{n+\frac{1}{2}}) \Delta t^2 + \frac{1}{2} u''(t_{n+\frac{1}{2}}) \Delta t^2 + O(\Delta t^3). 
\]

Let \( C^{n+\frac{1}{2}} \) cancel the \( \Delta t^2 \) terms:

\[
C^{n+\frac{1}{2}} = \frac{1}{24} u''(t_{n+\frac{1}{2}}) \Delta t^2 + \frac{1}{2} u''(t_{n+\frac{1}{2}}) \Delta t^2. 
\]
**Exact solutions of the finite difference equations**

- How does the truncation error depend on \( u_n \) in finite differences?
- O: one-sided difference: \( u\Delta t \) (lowest order)
- Centered difference: \( u^\prime \Delta t^2 \) (lowest order)
- Only harmonic and geometric mean involve \( u'_n \) or \( u_n \)

**Consequence:**
- \( u_n(t) = c/t + a \) will verify \( u(t) \) gives exact solution of the discrete equations \( (R = 0) \)
- Ideal for verification
- Centered schemes allow quadratic \( u_n \)

**Problem:** harmonic and geometric mean (error depends on \( u'_n \) and \( u_n \))

<table>
<thead>
<tr>
<th>( \Delta t )</th>
<th>( u'_n(t) )</th>
<th>( u_n(t) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>1</td>
</tr>
</tbody>
</table>

**Computing truncation errors in nonlinear problems (1)**

\[ u'(t) = f(u(t), t) \]

**Crank-Nicolson scheme:**

\[
\frac{u_n^{(n+1)}}{2} = f\left(u_n^{(n)}, t_n\right) + f\left(u_n^{(n+1)}, t_{n+1}\right) + \frac{1}{2} u_{n+1}'(t_{n+1}) \Delta t^2 + O(\Delta t^4).
\]

**Truncation error:**

\[ R_n = \frac{1}{2} u_{n+1}'(t_{n+1}) \Delta t^2 + O(\Delta t^4). \]

**Linear model without damping**

\[ u''(t) + \omega^2 u(t) = 0, \quad u(0) = I, \quad u'(0) = 0. \]

**Centered difference approximation:**

\[
\frac{u_n^{(n+2)}}{2} + \frac{1}{2} u_n^{(n+1)} \Delta t^2 - f(u_n^{(n)}, t_n) + \omega^2 u_n(t) = 0.
\]

**Truncation error:**

\[ R_n = -\frac{1}{2} u_n^{(n+1)} \Delta t^2 + O(\Delta t^4). \]
Truncation errors in the initial condition

- Initial conditions: $u(0) = I$, $u'(0) = V$
- Need discretization of $u'(0)$
- Standard, centered difference: $[D_t u - V]^2$, $R^0 = O(\Delta t^2)$
- Simple, forward difference: $[D_t u - V]^2$, $R^0 = O(\Delta t)$
- Does the lower order of the forward scheme impact the order of the whole simulation?
- Answer: run experiments!

Computing correction terms

- Can we add terms to the ODE such that the truncation error is improved?
  \[ D_t D_t u + \beta D_t u + \omega^2 u = C + R^n \]
- Idea: choose $C^n$ such that it absorbs the $\Delta t^2$ term in $R^n$,
  \[ C^n = \frac{1}{2} \omega^2 Q''(t_n) \Delta t^2. \]
- downstairs: go a $\omega^m$ term
- Remedy: use the ODE $u^n = -\omega u$ so that $u^m = \omega^m u$.
- Just apply the standard scheme to a modified ODE:
  \[ D_t D_t u + \omega^2 [1 - \frac{1}{12} \omega^2 \Delta t^2] u = 0^n. \]
- Accuracy is $O(\Delta t^2)$.

Model with damping and non-linearity

Linear damping $\beta u'$, co-torsional spring force $s(u)$, and excitation $F(t)$: $mu'' + \beta u' + s(u) = F(t)$

Central difference discretization:

\[ [mD_t D_t u + \beta D_t u + s(u) = F(t)]^n. \]

Truncation error is defined by

\[ [mD_t D_t u + \beta D_t u + s(u) = F + R_n]^n. \]

Carrying out the truncation error analysis

Using (??)-(??) and (??)-(??) we get

\[ [mD_t D_t u + \beta D_t u + s(u) = F + R_n]^n. \]

The terms
\[ mQ'(t_n) + \beta Q'(t_n) + \omega^2 Q(t_n) + s(Q(t_n)) - F^n \]
correspond to the ODE ($\omega = m\omega$).

Result: accuracy is $O(\Delta t^2)$ since

\[ R^n = \left( \frac{m}{12} \omega^2 Q(t_n) + \frac{1}{3} \omega Q'(t_n) \right) \Delta t^2 + O(\Delta t^4) \]

Correction terms complicated when the ODE has many terms...

Extension to quadratic damping

\[ mD_t D_t u'' + \beta D_t u' + s(u) = F(t) \]

Centered scheme: $u'/u'$ gives rise to a non-linearity.
Linearization trick: use a geometric mean,

\[ [u'/u']^{1/2} \approx [u''/u]^2, [u''/u]^2. \]

Scheme:

\[ [mD_t D_t u'' + \beta D_t u' + s(u') = F^n. \]

The truncation error for quadratic damping (1)

Define of $R^n$:

\[ [mD_t D_t u'' + \beta D_t u' + s(u') = F^n - R^n. \]

Truncation error of the geometric mean, see (??)-(??),

\[ \left[ D_t D_t u'' + \frac{1}{2} D_t D_t u' \right] = \left[ D_t D_t u'' + \frac{1}{2} D_t D_t u' \right]. \]

Using (??)-(??) for the $D_t D_t u'$ factor results in

\[ [D_t D_t u'' - \frac{1}{24} Q''(t_n) \Delta t^2 + O(\Delta t^4)] \times \]

\[ \left( Q' + \frac{1}{24} Q''(t_n) \Delta t^2 + O(\Delta t^4) \right). \]
The truncation error for quadratic damping (2)
For simplicity, remove the absolute value. The product becomes
\[ [D_t^2 u]_n = (u''(t_n))^2 + \frac{1}{2} m [u''(t_n)]^2 \Delta t^2 + O(\Delta t^4). \]

With
\[ m[D_t^2 u]_n = m[u''(t_n)]^2 + \frac{1}{2} m [u''(t_n)]^2 \Delta t^2 + O(\Delta t^4) \]
and using \( m[u''(t_n)]^2 + \beta (u'(t_n))^2 + s(u) = F \), we end up with
\[ \tilde{R} = \left( \frac{m}{2} u''(t_n) + \frac{1}{2} m [u''(t_n)]^2 \Delta t^2 + O(\Delta t^4) \right). \]

Second-order accuracy! Thanks to
- difference approximations with error \( O(\Delta t^2) \)
- geometric mean approximations with error \( O(\Delta t^2) \)

The forward-backward scheme
Forward step for \( u \), backward step for \( v \):

\[ [D_t^u]_n = v_n \]
\[ [D_t^v]_n = \frac{1}{m} [F(t_n) - \beta |v_{n-1}| v_{n+1} - s(u)] \]

- Note:
  - step \( a \) forward with known \( v \) in (77)
  - step \( b \) forward with known \( u \) in (77)
- Problem: \( |v| \) gives nonlinearity \( |v^{n1}| \) \( |v^{n+1}| \).
- Remedy: linearized as \( |v| v^{n1} \)

\[ [D_t^u]_n = v_n \]
\[ [D_t^v]^{n1+1} = \frac{1}{m} [F(t_{n1+1}) - \beta |v_n| v^{n+1} - s(u^{n+1})] \]

A centered scheme on a staggered mesh
Staggered mesh:
- \( u \) is computed at mesh points \( t_n \)
- \( v \) is computed at points \( t_{n+\frac{1}{2}} \)

Centered differences in (77), (77):
\[ [D_t u]_n = v_{n+\frac{1}{2}} \]
\[ [D_t v]_n = \frac{1}{m} [F(t_n) - \beta |v_{n+\frac{1}{2}}| v^{n+\frac{1}{2}} - s(u_{n+\frac{1}{2}})] \]

- Problem: \( |v||v| \) because \( v^v \) is not computed directly
- Remedy: Geometric mean,
  \[ |v||v| \approx |v^{n+\frac{1}{2}}| v^{n-\frac{1}{2}} \]

The general model formulated as first-order ODEs
\[ \mu u'' + \beta |u'| u' + s(u) = F(t) \]

Rewritten as first-order system:
\[ u' = v \]
\[ v' = \frac{1}{m} [F(t) - \beta |v| v - s(u)] \]

To solution methods:
- Forward-backward scheme
- Centered scheme on a staggered mesh

Truncation error analysis

- Aim (as always): turn difference operators into derivatives + truncation error terms
- One-sided forward/backward differences: error \( O(\Delta t) \)
- Linearization of \( |v^{n+1}| v^{n+1} \) to \( |v||v|^{n+1} \): error \( O(\Delta t^2) \)
- All errors are \( O(\Delta t) \)
- Fit second order? No!
- "Symmetric" use of the \( O(\Delta t^2) \) building blocks yields in fact a \( O(\Delta t^2) \) scheme! (1)
- Why? See next slide...
Truncation error analysis (2)

Resulting truncation error is $O(\Delta t^2)$:

$$R_{n-1}^2 = O(\Delta t^2), \quad R_n^v = O(\Delta t^2).$$

Observation

Comparing the schemes (??)-(??) and (??)-(??) are equivalent. Therefore, the forward/backward scheme with ad hoc linearisation is also $O(\Delta t^2)$!