Truncation Error Analysis

Hans Petter Langtangen[1,2]

[1]Center for Biomedical Computing, Simula Research Laboratory
[2]Department of Informatics, University of Oslo

Sep 24, 2014

WARNING: Preliminary version (expect typos!)

Contents

Purpose.

Truncation error analysis provides a widely applicable framework for
lyzing the accuracy of finite difference schemes. This type of analysi
also be used for finite element and finite volume methods if the dis
equations are written in finite difference form. The result of the ana
is an asymptotic estimate of the error in the scheme on the form
where h is a discretization parameter (Δt, Δx, etc.), r is a number, k
as the convergence rate, and C is a constant, typically dependent o
derivatives of the exact solution.

Knowing r gives understanding of the accuracy of the scheme.
maybe even more important, a powerful verification method for com
codes is to check that the empirically observed convergence rates in e
ments coincide with the theoretical value of r found from truncation
analysis.

The analysis can be carried out by hand, by symbolic software, and
numerically. All three methods will be illustrated. From examinin
symbolic expressions of the truncation error we can add correction t
to the differential equations in order to increase the numerical accur

In general, the term truncation error refers to the discrepancy tha
from performing a finite number of steps to approximate a process with i
many steps. The term is used in a number of contexts, including tru
of infinite series, finite precision arithmetic, finite differences, and diff
equations. We shall be concerned with computing truncation errors a
finite difference formulas and in finite difference discretizations of diff
equations.

. Overview of truncation error analysis

.1 Abstract problem setting

'onsider an abstract differential equation

$$\mathcal{L}(u) = 0,$$

here $\mathcal{L}(u)$ is some formula involving the unknown u and its derivatives. One
xample is $\mathcal{L}(u) = u'(t) + a(t)u(t) - b(t)$, where a and b are contants or functions
f time. We can discretize the differential equation and obtain a corresponding
iscrete model, here written as

$$\mathcal{L}_\Delta(u) = 0\,.$$

he solution u of this equation is the *numerical solution*. To distinguish the
umerical solution from the exact solution of the differential equation problem,
e denote the latter by u_e and write the differential equation and its discrete
ounterpart as

$$\mathcal{L}(u_e) = 0,$$
$$\mathcal{L}_\Delta(u) = 0\,.$$

nitial and/or boundary conditions can usually be left out of the truncation error
nalysis and are omitted in the following.

The numerical solution u is in a finite difference method computed at a collec-
on of mesh points. The discrete equations represented by the abstract equation
$_\Delta(u) = 0$ are usually algebraic equations involving u at some neighboring mesh
oints.

.2 Error measures

. key issue is how accurate the numerical solution is. The ultimate way of
ddressing this issue would be to compute the error $u_e - u$ at the mesh points.
his is usually extremely demanding. In very simplified problem settings we
ay, however, manage to derive formulas for the numerical solution u, and
erefore closed form expressions for the error $u_e - u$. Such special cases can
rovide considerable insight regarding accuracy and stability, but the results are
stablished for special problems.

The error $u_e - u$ can be computed empirically in special cases where we know
e. Such cases can be constructed by the method of manufactured solutions,
here we choose some exact solution $u_e = v$ and fit a source term f in the
overning differential equation $\mathcal{L}(u_e) = f$ such that $u_e = v$ is a solution (i.e., $f =$
(v)). Assuming an error model of the form Ch^r, where h is the discretization
arameter, such as Δt or Δx, one can estimate the convergence rate r. This is a
idely applicable procedure, but the validity of the results is, strictly speaking,
ed to the chosen test problems.

Another error measure is to ask to what extent the exact solution u_e
discrete equations. Clearly, u_e is in general not a solution of $\mathcal{L}_\Delta(u) = 0$.
can define the residual

$$R = \mathcal{L}_\Delta(u_e),$$

and investigate how close R is to zero. A small R means intuitively t
discrete equations are close to the differential equation, and then we are t
to think that u^n must also be close to $u_e(t_n)$.

The residual R is known as the truncation error of the finite difference
$\mathcal{L}_\Delta(u) = 0$. It appears that the truncation error is relatively straight
to compute by hand or symbolic software *without specializing the diff
equation and the discrete model to a special case*. The resulting R i
as a power series in the discretization parameters. The leading-orde
in the series provide an asymptotic measure of the accuracy of the nu
solution method (as the discretization parameters tend to zero). An ad
of truncation error analysis compared empricial estimation of convergen
or detailed analysis of a special problem with a mathematical expressior
numerical solution, is that the truncation error analysis reveals the a
of the various building blocks in the numerical method and how each b
block impacts the overall accuracy. The analysis can therefore be used t
building blocks with lower accuracy than the others.

Knowing the truncation error or other error measures is important for
tion of programs by empirically establishing convergence rates. The fortl
text will provide many examples on how to compute truncation errors f
difference discretizations of ODEs and PDEs.

2 Truncation errors in finite difference forn

The accuracy of a finite difference formula is a fundamental issue when disc
differential equations. We shall first go through a particular example i
and thereafter list the truncation error in the most common finite di
approximation formulas.

2.1 Example: The backward difference for $u'(t)$

Consider a backward finite difference approximation of the first-order de
u':

$$[D_t^- u]^n = \frac{u^n - u^{n-1}}{\Delta t} \approx u'(t_n)\,.$$

Here, u^n means the value of some function $u(t)$ at a point t_n, and [L
the *discrete derivative* of $u(t)$ at $t = t_n$. The discrete derivative comput
finite difference is not exactly equal to the derivative $u'(t_n)$. The erro
approximation is

$$R^n = [D_t^- u]^n - u'(t_n)\,. \qquad (2)$$

The common way of calculating R^n is to

1. expand $u(t)$ in a Taylor series around the point where the derivative is evaluated, here t_n,
2. insert this Taylor series in (2), and
3. collect terms that cancel and simplify the expression.

he result is an expression for R^n in terms of a power series in Δt. The error R^n commonly referred to as the *truncation error* of the finite difference formula.

The Taylor series formula often found in calculus books takes the form

$$f(x+h) = \sum_{i=0}^{\infty} \frac{1}{i!}\frac{d^i f}{dx^i}(x)h^i\,.$$

ı our application, we expand the Taylor series around the point where the finite ifference formula approximates the derivative. The Taylor series of u^n at t_n simply $u(t_n)$, while the Taylor sereis of u^{n-1} at t_n must employ the general ɔrmula,

$$\begin{aligned} u(t_{n-1}) = u(t-\Delta t) &= \sum_{i=0}^{\infty} \frac{1}{i!}\frac{d^i u}{dt^i}(t_n)(-\Delta t)^i \\ &= u(t_n) - u'(t_n)\Delta t + \frac{1}{2}u''(t_n)\Delta t^2 + \mathcal{O}(\Delta t^3), \end{aligned}$$

here $\mathcal{O}(\Delta t^3)$ means a power-series in Δt where the lowest power is Δt^3. We ssume that Δt is small such that $\Delta t^p \gg \Delta t^q$ if p is smaller than q. The details f higher-order terms in Δt are therefore not of much interest. Inserting the aylor series above in the left-hand side of1 (2) gives rise to some algebra:

$$\begin{aligned} [D_t^- u]^n - u'(t_n) &= \frac{u(t_n) - u(t_{n-1})}{\Delta t} - u'(t_n) \\ &= \frac{u(t_n) - (u(t_n) - u'(t_n)\Delta t + \frac{1}{2}u''(t_n)\Delta t^2 + \mathcal{O}(\Delta t^3))}{\Delta t} - u'(t_n) \\ &= -\frac{1}{2}u''(t_n)\Delta t + \mathcal{O}(\Delta t^2)), \end{aligned}$$

hich is, according to (2), the truncation error:

$$R^n = -\frac{1}{2}u''(t_n)\Delta t + \mathcal{O}(\Delta t^2))\,. \qquad (3)$$

he dominating term for small Δt is $-\frac{1}{2}u''(t_n)\Delta t$, which is proportional to Δt, ıd we say that the truncation error is of *first order* in Δt.

2.2 Example: The forward difference for $u'(t)$

We can analyze the approximation error in the forward difference

$$u'(t_n) \approx [D_t^+ u]^n = \frac{u^{n+1} - u^n}{\Delta t},$$

by writing

$$R^n = [D_t^+ u]^n - u'(t_n),$$

and expanding u^{n+1} in a Taylor series around t_n,

$$u(t_{n+1}) = u(t_n) + u'(t_n)\Delta t + \frac{1}{2}u''(t_n)\Delta t^2 + \mathcal{O}(\Delta t^3)\,.$$

The result becomes

$$R = \frac{1}{2}u''(t_n)\Delta t + \mathcal{O}(\Delta t^2),$$

showing that also the forward difference is of first order.

2.3 Example: The central difference for $u'(t)$

For the central difference approximation,

$$u'(t_n) \approx [D_t u]^n, \quad [D_t u]^n = \frac{u^{n+\frac{1}{2}} - u^{n-\frac{1}{2}}}{\Delta t},$$

we write

$$R^n = [D_t u]^n - u'(t_n),$$

and expand $u(t_{n+\frac{1}{2}})$ and $u(t_{n-1/2})$ in Taylor series around the point t the derivative is evaluated. We have

$$\begin{aligned} u(t_{n+\frac{1}{2}}) =& u(t_n) + u'(t_n)\frac{1}{2}\Delta t + \frac{1}{2}u''(t_n)(\frac{1}{2}\Delta t)^2 + \\ & \frac{1}{6}u'''(t_n)(\frac{1}{2}\Delta t)^3 + \frac{1}{24}u''''(t_n)(\frac{1}{2}\Delta t)^4 + \\ & \frac{1}{120}u''''(t_n)(\frac{1}{2}\Delta t)^5 + \mathcal{O}(\Delta t^6), \\ u(t_{n-1/2}) =& u(t_n) - u'(t_n)\frac{1}{2}\Delta t + \frac{1}{2}u''(t_n)(\frac{1}{2}\Delta t)^2 - \\ & \frac{1}{6}u'''(t_n)(\frac{1}{2}\Delta t)^3 + \frac{1}{24}u''''(t_n)(\frac{1}{2}\Delta t)^4 - \\ & \frac{1}{120}u'''''(t_n)(\frac{1}{2}\Delta t)^5 + \mathcal{O}(\Delta t^6)\,. \end{aligned}$$

Now,

$$u(t_{n+\frac{1}{2}}) - u(t_{n-1/2}) = u'(t_n)\Delta t + \frac{1}{24}u'''(t_n)\Delta t^3 + \frac{1}{960}u'''''(t_n)\Delta t^5 + \mathcal{C}$$

y collecting terms in $[D_t u]^n - u(t_n)$ we find the truncation error to be

$$R^n = \frac{1}{24}u'''(t_n)\Delta t^2 + \mathcal{O}(\Delta t^4), \tag{4}$$

ith only even powers of Δt. Since $R \sim \Delta t^2$ we say the centered difference is of *econd order* in Δt.

.4 Overview of leading-order error terms in finite difference formulas

ere we list the leading-order terms of the truncation errors associated with everal common finite difference formulas for the first and second derivatives.

$$[D_t u]^n = \frac{u^{n+\frac{1}{2}} - u^{n-\frac{1}{2}}}{\Delta t} = u'(t_n) + R^n, \tag{5}$$

$$R^n = \frac{1}{24}u'''(t_n)\Delta t^2 + \mathcal{O}(\Delta t^4) \tag{6}$$

$$[D_{2t} u]^n = \frac{u^{n+1} - u^{n-1}}{2\Delta t} = u'(t_n) + R^n, \tag{7}$$

$$R^n = \frac{1}{6}u'''(t_n)\Delta t^2 + \mathcal{O}(\Delta t^4) \tag{8}$$

$$[D_t^- u]^n = \frac{u^n - u^{n-1}}{\Delta t} = u'(t_n) + R^n, \tag{9}$$

$$R^n = -\frac{1}{2}u''(t_n)\Delta t + \mathcal{O}(\Delta t^2) \tag{10}$$

$$[D_t^+ u]^n = \frac{u^{n+1} - u^n}{\Delta t} = u'(t_n) + R^n, \tag{11}$$

$$R^n = \frac{1}{2}u''(t_n)\Delta t + \mathcal{O}(\Delta t^2) \tag{12}$$

$$[\bar{D}_t u]^{n+\theta} = \frac{u^{n+1} - u^n}{\Delta t} = u'(t_{n+\theta}) + R^{n+\theta}, \tag{13}$$

$$R^{n+\theta} = \frac{1}{2}(1-2\theta)u''(t_{n+\theta})\Delta t - \frac{1}{6}((1-\theta)^3 - \theta^3)u'''(t_{n+\theta})\Delta t^2 + \mathcal{O}(\Delta t^3) \tag{14}$$

$$[D_t^{2-} u]^n = \frac{3u^n - 4u^{n-1} + u^{n-2}}{2\Delta t} = u'(t_n) + R^n, \tag{15}$$

$$R^n = -\frac{1}{3}u'''(t_n)\Delta t^2 + \mathcal{O}(\Delta t^3) \tag{16}$$

$$[D_t D_t u]^n = \frac{u^{n+1} - 2u^n + u^{n-1}}{\Delta t^2} = u''(t_n) + R^n, \tag{17}$$

$$R^n = \frac{1}{12}u''''(t_n)\Delta t^2 + \mathcal{O}(\Delta t^4) \tag{18}$$

It will also be convenient to have the truncation errors for various m averages. The weighted arithmetic mean leads to

$$[\overline{u}^{t,\theta}]^{n+\theta} = \theta u^{n+1} + (1-\theta)u^n = u(t_{n+\theta}) + R^{n+\theta},$$
$$R^{n+\theta} = \frac{1}{2}u''(t_{n+\theta})\Delta t^2\theta(1-\theta) + \mathcal{O}(\Delta t^3)\,.$$

The standard arithmetic mean follows from this formula when $\theta = 1/2$. Ex at point t_n we get

$$[\overline{u}^t]^n = \frac{1}{2}(u^{n-\frac{1}{2}} + u^{n+\frac{1}{2}}) = u(t_n) + R^n,$$
$$R^n = \frac{1}{8}u''(t_n)\Delta t^2 + \frac{1}{384}u''''(t_n)\Delta t^4 + \mathcal{O}(\Delta t^6)\,.$$

The geometric mean also has an error $\mathcal{O}(\Delta t^2)$:

$$[\overline{u^2}^{t,g}]^n = u^{n-\frac{1}{2}}u^{n+\frac{1}{2}} = (u^n)^2 + R^n,$$
$$R^n = -\frac{1}{4}u'(t_n)^2\Delta t^2 + \frac{1}{4}u(t_n)u''(t_n)\Delta t^2 + \mathcal{O}(\Delta t^4)\,.$$

The harmonic mean is also second-order accurate:

$$[\overline{u}^{t,h}]^n = u^n = \frac{2}{\frac{1}{u^{n-\frac{1}{2}}} + \frac{1}{u^{n+\frac{1}{2}}}} + R^{n+\frac{1}{2}},$$
$$R^n = -\frac{u'(t_n)^2}{4u(t_n)}\Delta t^2 + \frac{1}{8}u''(t_n)\Delta t^2\,.$$

2.5 Software for computing truncation errors

We can use `sympy` to aid calculations with Taylor series. The derivati be defined as symbols, say `D3f` for the 3rd derivative of some functic truncated Taylor series can then be written as `f + D1f*h + D2f*h**2` following class takes some symbol `f` for the function in question and mak of symbols for the derivatives. The `__call__` method computes the s form of the series truncated at `num_terms` terms.

```
import sympy as sp

class TaylorSeries:
    """Class for symbolic Taylor series."""
    def __init__(self, f, num_terms=4):
        self.f = f
        self.N = num_terms
        # Introduce symbols for the derivatives
        self.df = [f]
        for i in range(1, self.N+1):
```

```
            self.df.append(sp.Symbol('D%d%s' % (i, f.name)))

    def __call__(self, h):
        """Return the truncated Taylor series at x+h."""
        terms = self.f
        for i in range(1, self.N+1):
            terms += sp.Rational(1, sp.factorial(i))*self.df[i]*h**i
        return terms
```

We may, for example, use this class to compute the truncation error of the orward Euler finite difference formula:

```
>>> from truncation_errors import TaylorSeries
>>> from sympy import *
>>> u, dt = symbols('u dt')
>>> u_Taylor = TaylorSeries(u, 4)
>>> u_Taylor(dt)
D1u*dt + D2u*dt**2/2 + D3u*dt**3/6 + D4u*dt**4/24 + u
>>> FE = (u_Taylor(dt) - u)/dt
>>> FE
(D1u*dt + D2u*dt**2/2 + D3u*dt**3/6 + D4u*dt**4/24)/dt
>>> simplify(FE)
D1u + D2u*dt/2 + D3u*dt**2/6 + D4u*dt**3/24
```

he truncation error consists of the terms after the first one (u').

The module file `trunc/truncation_errors.py`[1] contains another class `DiffOp` ith symbolic expressions for most of the truncation errors listed in the previous ection. For example:

```
>>> from truncation_errors import DiffOp
>>> from sympy import *
>>> u = Symbol('u')
>>> diffop = DiffOp(u, independent_variable='t')
>>> diffop['geometric_mean']
-D1u**2*dt**2/4 - D1u*D3u*dt**4/48 + D2u**2*dt**4/64 + ...
>>> diffop['Dtm']
D1u + D2u*dt/2 + D3u*dt**2/6 + D4u*dt**3/24
>>> >>> diffop.operator_names()
['geometric_mean', 'harmonic_mean', 'Dtm', 'D2t', 'DtDt',
 'weighted_arithmetic_mean', 'Dtp', 'Dt']
```

he indexing of `diffop` applies names that correspond to the operators: `Dtp` or D_t^+, `Dtm` for D_t^-, `Dt` for D_t, `D2t` for D_{2t}, `DtDt` for D_tD_t.

3 Truncation errors in exponential decay ODE

We shall now compute the truncation error of a finite difference scheme for a ifferential equation. Our first problem involves the following the linear ODE odeling exponential decay,

$$u'(t) = -au(t)\,. \tag{27}$$

[1] http://tinyurl.com/jvzzcfn/trunc/truncation_errors.py

3.1 Truncation error of the Forward Euler scheme

We begin with the Forward Euler scheme for discretizing (27):

$$[D_t^+ u = -au]^n\,.$$

The idea behind the truncation error computation is to insert the exact $u_{\rm e}$ of the differential equation problem (27) in the discrete equations (find the residual that arises because $u_{\rm e}$ does not solve the discrete eq Instead, $u_{\rm e}$ solves the discrete equations with a residual R^n:

$$[D_t^+ u_{\rm e} + au_{\rm e} = R]^n\,.$$

From (11)-(12) it follows that

$$[D_t^+ u_{\rm e}]^n = u_{\rm e}'(t_n) + \frac{1}{2}u_{\rm e}''(t_n)\Delta t + \mathcal{O}(\Delta t^2),$$

which inserted in (29) results in

$$u_{\rm e}'(t_n) + \frac{1}{2}u_{\rm e}''(t_n)\Delta t + \mathcal{O}(\Delta t^2) + au_{\rm e}(t_n) = R^n\,.$$

Now, $u_{\rm e}'(t_n) + au_{\rm e}^n = 0$ since $u_{\rm e}$ solves the differential equation. The re terms constitute the residual:

$$R^n = \frac{1}{2}u_{\rm e}''(t_n)\Delta t + \mathcal{O}(\Delta t^2)\,.$$

This is the truncation error R^n of the Forward Euler scheme.

Because R^n is proportional to Δt, we say that the Forward Euler is of first order in Δt. However, the truncation error is just one error r and it is not equal to the true error $u_{\rm e}^n - u^n$. For this simple model p we can compute a range of different error measures for the Forwar scheme, including the true error $u_{\rm e}^n - u^n$, and all of them have dominatir proportional to Δt.

3.2 Truncation error of the Crank-Nicolson scheme

For the Crank-Nicolson scheme,

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

we compute the truncation error by inserting the exact solution of the O adding a residual R,

$$[D_t u_{\rm e} + a\overline{u_{\rm e}}^t = R]^{n+\frac{1}{2}}\,.$$

The term $[D_t u_{\rm e}]^{n+\frac{1}{2}}$ is easily computed from (5)-(6) by replacing n wit in the formula,

$$[D_t u_e]^{n+\frac{1}{2}} = u'(t_{n+\frac{1}{2}}) + \frac{1}{24}u_e'''(t_{n+\frac{1}{2}})\Delta t^2 + \mathcal{O}(\Delta t^4)\,.$$

he arithmetic mean is related to $u(t_{n+\frac{1}{2}})$ by (21)-(22) so

$$[a\overline{u_e}^{t}]^{n+\frac{1}{2}} = u(t_{n+\frac{1}{2}}) + \frac{1}{8}u''(t_n)\Delta t^2 + + \mathcal{O}(\Delta t^4)\,.$$

ıserting these expressions in (32) and observing that $u_e'(t_{n+\frac{1}{2}}) + au_e^{n+\frac{1}{2}} = 0$, ecause $u_e(t)$ solves the ODE $u'(t) = -au(t)$ at any point t, we find that

$$R^{n+\frac{1}{2}} = \left(\frac{1}{24}u_e'''(t_{n+\frac{1}{2}}) + \frac{1}{8}u''(t_n)\right)\Delta t^2 + \mathcal{O}(\Delta t^4) \qquad (33)$$

ere, the truncation error is of second order because the leading term in R is roportional to Δt^2.

At this point it is wise to redo some of the computations above to establish ıe truncation error of the Backward Euler scheme, see Exercise 4.

.3 Truncation error of the θ-rule

/e may also compute the truncation error of the θ-rule,

$$[\bar{D}_t u = -a\overline{u}^{t,\theta}]^{n+\theta}\,.$$

ur computational task is to find $R^{n+\theta}$ in

$$[\bar{D}_t u_e + a\overline{u_e}^{t,\theta} = R]^{n+\theta}\,.$$

rom (13)-(14) and (19)-(20) we get expressions for the terms with u_e. Using ıat $u_e'(t_{n+\theta}) + au_e(t_{n+\theta}) = 0$, we end up with

$$R^{n+\theta} = (\frac{1}{2}-\theta)u_e''(t_{n+\theta})\Delta t + \frac{1}{2}\theta(1-\theta)u_e''(t_{n+\theta})\Delta t^2 + \frac{1}{2}(\theta^2-\theta+3)u_e'''(t_{n+\theta})\Delta t^2 + \mathcal{O}(\Delta t^3) \qquad (34)$$

or $\theta = 1/2$ the first-order term vanishes and the scheme is of second order, hile for $\theta \neq 1/2$ we only have a first-order scheme.

.4 Using symbolic software

he previously mentioned `truncation_error` module can be used to automate ıe Taylor series expansions and the process of collecting terms. Here is an xample on possible use:


```
from truncation_error import DiffOp
from sympy import *

def decay():
    u, a = symbols('u a')
    diffop = DiffOp(u, independent_variable='t',
                    num_terms_Taylor_series=3)
    D1u = diffop.D(1)   # symbol for du/dt
    ODE = D1u + a*u     # define ODE

    # Define schemes
    FE = diffop['Dtp'] + a*u
    CN = diffop['Dt' ] + a*u
    BE = diffop['Dtm'] + a*u
    theta = diffop['barDt'] + a*diffop['weighted_arithmetic_mean'
    theta = sm.simplify(sm.expand(theta))
    # Residuals (truncation errors)
    R = {'FE': FE-ODE, 'BE': BE-ODE, 'CN': CN-ODE,
         'theta': theta-ODE}
    return R
```

The returned dictionary becomes

```
decay: {
 'BE': D2u*dt/2 + D3u*dt**2/6,
 'FE': -D2u*dt/2 + D3u*dt**2/6,
 'CN': D3u*dt**2/24,
 'theta': -D2u*a*dt**2*theta**2/2 + D2u*a*dt**2*theta/2 -
          D2u*dt*theta + D2u*dt/2 + D3u*a*dt**3*theta**3/3 -
          D3u*a*dt**3*theta**2/2 + D3u*a*dt**3*theta/6 +
          D3u*dt**2*theta**2/2 - D3u*dt**2*theta/2 + D3u*dt*
}
```

The results are in correspondence with our hand-derived expressions.

3.5 Empirical verification of the truncation error

The task of this section is to demonstrate how we can compute the tru error R numerically. For example, the truncation error of the Forwar scheme applied to the decay ODE $u' = -ua$ is

$$R^n = [D_t^+ u_e + au_e]^n\,.$$

If we happen to know the exact solution $u_e(t)$, we can easily evaluate I the above formula.

To estimate how R varies with the discretization parameter Δt, wl been our focus in the previous mathematical derivations, we first m assumption that $R = C\Delta t^r$ for appropriate constants C and r and small Δt. The rate r can be estimated from a series of experiments where Δt i Suppose we have m experiments $(\Delta t_i, R_i)$, $i = 0,\ldots,m-1$. For two con experiments $(\Delta t_{i-1}, R_{i-1})$ and $(\Delta t_i, R_i)$, a corresponding r_{i-1} can be es by

$$r_{i-1} = \frac{\ln(R_{i-1}/R_i)}{\ln(\Delta t_{i-1}/\Delta t_i)},$$

r $i = 1, \ldots, m-1$. Note that the truncation error R_i varies through the mesh,) (36) is to be applied pointwise. A complicating issue is that R_i and R_{i-1} refer) different meshes. Pointwise comparisons of the truncation error at a certain oint in all meshes therefore requires any computed R to be restricted to the *oarsest mesh* and that all finer meshes contain all the points in the coarsest esh. Suppose we have N_0 intervals in the coarsest mesh. Inserting a superscript in (36), where n counts mesh points in the coarsest mesh, $n = 0, \ldots, N_0$, leads) the formula

$$r_{i-1}^n = \frac{\ln(R_{i-1}^n/R_i^n)}{\ln(\Delta t_{i-1}/\Delta t_i)}\,. \tag{37}$$

xperiments are most conveniently defined by N_0 and a number of refinements . Suppose each mesh have twice as many cells N_i as the previous one:

$$N_i = 2^i N_0, \quad \Delta t_i = T N_i^{-1},$$

here $[0, T]$ is the total time interval for the computations. Suppose the computed $_i$ values on the mesh with N_i intervals are stored in an array `R[i]` (`R` being a st of arrays, one for each mesh). Restricting this R_i function to the coarsest esh means extracting every N_i/N_0 point and is done as follows:

```
stride = N[i]/N_0
R[i] = R[i][::stride]
```

he quantity `R[i][n]` now corresponds to R_i^n.

In addition to estimating r for the pointwise values of $R = C\Delta t^r$, we may lso consider an integrated quantity on mesh i,

$$R_{I,i} = \left(\Delta t_i \sum_{n=0}^{N_i} (R_i^n)^2\right)^{\frac{1}{2}} \approx \int_0^T R_i(t)dt\,. \tag{38}$$

he sequence $R_{I,i}$, $i = 0, \ldots, m-1$, is also expected to behave as $C\Delta t^r$, with e same r as for the pointwise quantity R, as $\Delta t \to 0$.

The function below computes the R_i and $R_{I,i}$ quantities, plots them and ompares with the theoretically derived truncation error (`R_a`) if available.

```
import numpy as np
import scitools.std as plt

def estimate(truncation_error, T, N_0, m, makeplot=True):
    """
    Compute the truncation error in a problem with one independent
    variable, using m meshes, and estimate the convergence
    rate of the truncation error.

    The user-supplied function truncation_error(dt, N) computes
    the truncation error on a uniform mesh with N intervals of
    length dt::
```



```
        R, t, R_a = truncation_error(dt, N)

    where R holds the truncation error at points in the array t,
    and R_a are the corresponding theoretical truncation error
    values (None if not available).

    The truncation_error function is run on a series of meshes
    with 2**i*N_0 intervals, i=0,1,...,m-1.
    The values of R and R_a are restricted to the coarsest mesh.
    and based on these data, the convergence rate of R (pointwise
    and time-integrated R can be estimated empirically.
    """
    N = [2**i*N_0 for i in range(m)]

    R_I = np.zeros(m) # time-integrated R values on various meshe
    R   = [None]*m    # time series of R restricted to coarsest m
    R_a = [None]*m    # time series of R_a restricted to coarsest
    dt = np.zeros(m)
    legends_R = [];  legends_R_a = []  # all legends of curves

    for i in range(m):
        dt[i] = T/float(N[i])
        R[i], t, R_a[i] = truncation_error(dt[i], N[i])

        R_I[i] = np.sqrt(dt[i]*np.sum(R[i]**2))

        if i == 0:
            t_coarse = t           # the coarsest mesh

        stride = N[i]/N_0
        R[i] = R[i][::stride]      # restrict to coarsest mesh
        R_a[i] = R_a[i][::stride]

        if makeplot:
            plt.figure(1)
            plt.plot(t_coarse, R[i], log='y')
            legends_R.append('N=%d' % N[i])
            plt.hold('on')

            plt.figure(2)
            plt.plot(t_coarse, R_a[i] - R[i], log='y')
            plt.hold('on')
            legends_R_a.append('N=%d' % N[i])

    if makeplot:
        plt.figure(1)
        plt.xlabel('time')
        plt.ylabel('pointwise truncation error')
        plt.legend(legends_R)
        plt.savefig('R_series.png')
        plt.savefig('R_series.pdf')
        plt.figure(2)
        plt.xlabel('time')
        plt.ylabel('pointwise error in estimated truncation error
        plt.legend(legends_R_a)
        plt.savefig('R_error.png')
        plt.savefig('R_error.pdf')

    # Convergence rates
    r_R_I = convergence_rates(dt, R_I)
```

```
    print 'R integrated in time; r:',
    print ' '.join(['%.1f' % r for r in r_R_I])
    R = np.array(R)  # two-dim. numpy array
    r_R = [convergence_rates(dt, R[:,n])[-1]
           for n in range(len(t_coarse))]
```

The first `makeplot` block demonstrates how to build up two figures in parallel, ısing `plt.figure(i)` to create and switch to figure number `i`. Figure numbers ;art at 1. A logarithmic scale is used on the y axis since we expect that R ; a function of time (or mesh points) is exponential. The reason is that the ıeoretical estimate (30) contains u_e'', which for the present model goes like e^{-at}. aking the logarithm makes a straight line.

The code follows closely the previously stated mathematical formulas, but the ;atements for computing the convergence rates might deserve an explanation. he generic help function `convergence_rate(h, E)` computes and returns r_{i-1}, $= 1, \ldots, m-1$ from (37), given Δt_i in `h` and R_i^n in `E`:

```
lef convergence_rates(h, E):
    from math import log
    r = [log(E[i]/E[i-1])/log(h[i]/h[i-1])
         for i in range(1, len(h))]
    return r
```

Calling `r_R_I = convergence_rates(dt, R_I)` computes the sequence of ıtes $r_0, r_1, \ldots, r_{m-2}$ for the model $R_I \sim \Delta t^r$, while the statements

```
} = np.array(R)  # two-dim. numpy array
:_R = [convergence_rates(dt, R[:,n])[-1]
       for n in range(len(t_coarse))]
```

ompute the final rate r_{m-2} for $R^n \sim \Delta t^r$ at each mesh point t_n in the coarsest ıesh. This latter computation deserves more explanation. Since `R[i][n]` holds ıe estimated truncation error R_i^n on mesh i, at point t_n in the coarsest mesh, `[:,n]` picks out the sequence R_i^n for $i = 0, \ldots, m-1$. The `convergence_rate` ınction computes the rates at t_n, and by indexing `[-1]` on the returned array om `convergence_rate`, we pick the rate r_{m-2}, which we believe is the best ;timation since it is based on the two finest meshes.

The `estimate` function is available in a module `trunc_empir.py`[2]. Let us pply this function to estimate the truncation error of the Forward Euler scheme. Ve need a function `decay_FE(dt, N)` that can compute (35) at the points in a ıesh with time step `dt` and `N` intervals:

```
import numpy as np
import trunc_empir

lef decay_FE(dt, N):
    dt = float(dt)
    t = np.linspace(0, N*dt, N+1)
    u_e = I*np.exp(-a*t)  # exact solution, I and a are global
```

[2]`http://tinyurl.com/jvzzcfn/trunc/trunc_empir.py`


```
    u = u_e  # naming convention when writing up the scheme
    R = np.zeros(N)

    for n in range(0, N):
        R[n] = (u[n+1] - u[n])/dt + a*u[n]

    # Theoretical expression for the trunction error
    R_a = 0.5*I*(-a)**2*np.exp(-a*t)*dt

    return R, t[:-1], R_a[:-1]

if __name__ == '__main__':
    I = 1; a = 2  # global variables needed in decay_FE
    trunc_empir.estimate(decay_FE, T=2.5, N_0=6, m=4, makeplot=Tr
```

The estimated rates for the integrated truncation error R_I become and 1.0 for this sequence of four meshes. All the rates for R^n, comp `r_R`, are also very close to 1 at all mesh points. The agreement betw theoretical formula (30) and the computed quantity (ref(35)) is very g illustrated in Figures 1 and 2. The program `trunc_decay_FE.py`[3] was perform the simulations and it can easily be modified to test other scher also Exericse 5).

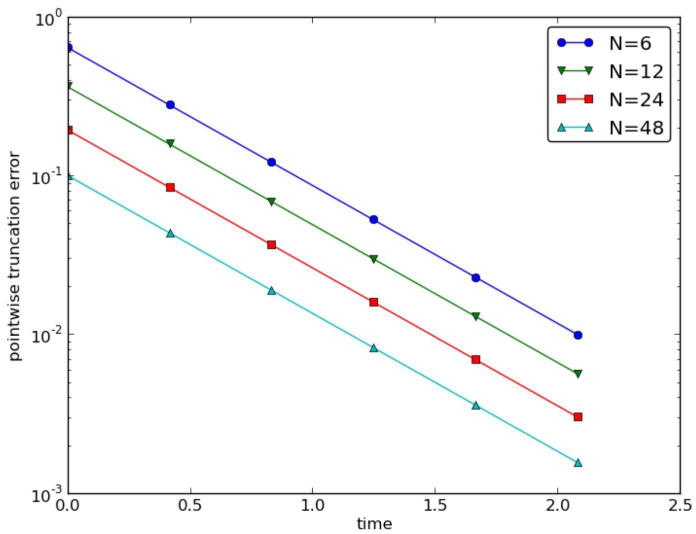
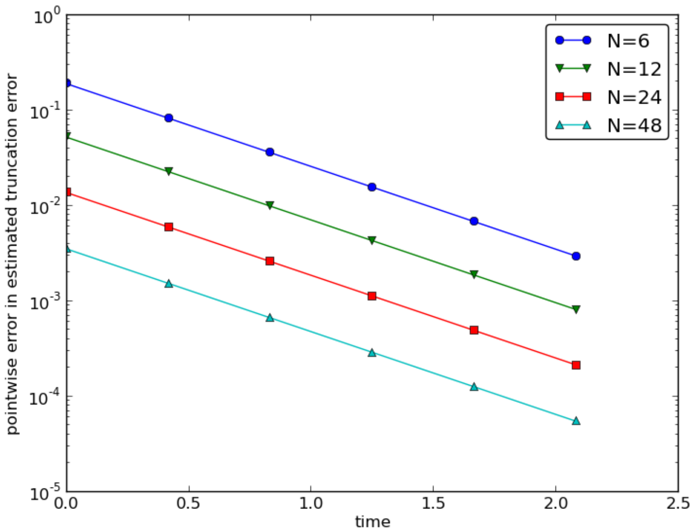


Figure 1: Estimated truncation error at mesh points for different me

[3]`http://tinyurl.com/jvzzcfn/trunc/trunc_decay_FE.py`



igure 2: Difference between theoretical and estimated truncation error at mesh oints for different meshes.

.6 Increasing the accuracy by adding correction terms

'ow we ask the question: can we add terms in the differential equation that an help increase the order of the truncation error? To be precise, let us revisit ıe Forward Euler scheme for $u' = -au$, insert the exact solution $u_{\rm e}$, include a esidual R, but also include new terms C:

$$[D_t^+ u_{\rm e} + au_{\rm e} = C + R]^n \,. \tag{39}$$

ıserting the Taylor expansions for $[D_t^+ u_{\rm e}]^n$ and keeping terms up to 3rd order ı Δt gives the equation

$$\frac{1}{2}u_{\rm e}''(t_n)\Delta t - \frac{1}{6}u_{\rm e}'''(t_n)\Delta t^2 + \frac{1}{24}u_{\rm e}''''(t_n)\Delta t^3 + \mathcal{O}(\Delta t^4) = C^n + R^n \,.$$

'an we find C^n such that R^n is $\mathcal{O}(\Delta t^2)$? Yes, by setting

$$C^n = \frac{1}{2}u_{\rm e}''(t_n)\Delta t,$$

e manage to cancel the first-order term and

$$R^n = \frac{1}{6}u_{\rm e}'''(t_n)\Delta t^2 + \mathcal{O}(\Delta t^3)\,.$$

The correction term C^n introduces $\frac{1}{2}\Delta t u''$ in the discrete equati we have to get rid of the derivative u''. One idea is to approximate second-order accurate finite difference formula, $u'' \approx (u^{n+1} - 2u^n + u^{n-}$ but this introduces an additional time level with u^{n-1}. Another approa rewrite u'' in terms of u' or u using the ODE:

$$u' = -au \quad \Rightarrow \quad u'' = -au' = -a(-au) = a^2 u\,.$$

This means that we can simply set $C^n = \frac{1}{2}a^2\Delta t u^n$. We can then eith the discrete equation

$$[D_t^+ u = -au + \frac{1}{2}a^2\Delta t u]^n,$$

or we can equivalently discretize the perturbed ODE

$$u' = -\hat{a}u, \quad \hat{a} = a(1 - \frac{1}{2}a\Delta t),$$

by a Forward Euler method. That is, we replace the original coefficient perturbed coefficient $\hat{a}$. Observe that $\hat{a} \rightarrow a$ as $\Delta t \rightarrow 0$.

The Forward Euler method applied to (41) results in

$$[D_t^+ u = -a(1 - \frac{1}{2}a\Delta t)u]^n\,.$$

We can control our computations and verify that the truncation erro scheme above is indeed $\mathcal{O}(\Delta t^2)$.

Another way of revealing the fact that the perturbed ODE leads to accurate solution is to look at the amplification factor. Our scheme written as

$$u^{n+1} = Au^n, \quad A = 1 - \hat{a}\Delta t = 1 - p + \frac{1}{2}p^2, \quad p = a\Delta t,$$

The amplification factor A as a function of $p = a\Delta t$ is seen to be the fir terms of the Taylor series for the exact amplification factor e^{-p}. The I Euler scheme for $u = -au$ gives only the first two terms $1 - p$ of the Tayl for e^{-p}. That is, using $\hat{a}$ increases the order of the accuracy in the ampl factor.

Instead of replacing u'' by $a^2 u$, we use the relation $u'' = -au'$ an term $-\frac{1}{2}a\Delta t u'$ in the ODE:

$$u' = -au - \frac{1}{2}a\Delta t u' \quad \Rightarrow \quad \left(1 + \frac{1}{2}a\Delta t\right)u' = -au\,.$$

Using a Forward Euler method results in

$$\left(1 + \frac{1}{2}a\Delta t\right)\frac{u^{n+1} - u^n}{\Delta t} = -au^n,$$

which after some algebra can be written as

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

his is the same formula as the one arising from a Crank-Nicolson scheme applied › $u' = -au$! It now recommended to do Exercise 6 and repeat the above steps › see what kind of correction term is needed in the Backward Euler scheme to ıake it second order.

The Crank-Nicolson scheme is a bit more challenging to analyze, but the leas and techniques are the same. The discrete equation reads

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

nd the truncation error is defined through

$$[D_t u_e + a\overline{u_e}^t = C + R]^{n+\frac{1}{2}},$$

here we have added a correction term. We need to Taylor expand both the iscrete derivative and the arithmetic mean with aid of (5)-(6) and (21)-(22), espectively. The result is

$$\frac{1}{24}u_e'''(t_{n+\frac{1}{2}})\Delta t^2 + \mathcal{O}(\Delta t^4) + \frac{a}{8}u_e''(t_{n+\frac{1}{2}})\Delta t^2 + \mathcal{O}(\Delta t^4) = C^{n+\frac{1}{2}} + R^{n+\frac{1}{2}}\,.$$

he goal now is to make $C^{n+\frac{1}{2}}$ cancel the Δt^2 terms:

$$C^{n+\frac{1}{2}} = \frac{1}{24}u_e'''(t_{n+\frac{1}{2}})\Delta t^2 + \frac{a}{8}u_e''(t_n)\Delta t^2\,.$$

'sing $u' = -au$, we have that $u'' = a^2 u$, and we find that $u''' = -a^3 u$. We can ıerefore solve the perturbed ODE problem

$$u' = -\hat{a}u, \quad \hat{a} = a(1 - \frac{1}{12}a^2\Delta t^2),$$

y the Crank-Nicolson scheme and obtain a method that is of fourth order ı Δt. Exercise 7 encourages you to implement these correction terms and alculate empirical convergence rates to verify that higher-order accuracy is ıdeed obtained in real computations.

.7 Extension to variable coefficients

et us address the decay ODE with variable coefficients,

$$u'(t) = -a(t)u(t) + b(t),$$

iscretized by the Forward Euler scheme,

$$[D_t^+ u = -au + b]^n\,. \tag{42}$$

The truncation error R is as always found by inserting the exact soluti in the discrete scheme:

$$[D_t^+ u_e + au_e - b = R]^n\,.$$

Using (11)-(12),

$$u_e'(t_n) - \frac{1}{2}u_e''(t_n)\Delta t + \mathcal{O}(\Delta t^2) + a(t_n)u_e(t_n) - b(t_n) = R^n\,.$$

Because of the ODE,

$$u_e'(t_n) + a(t_n)u_e(t_n) - b(t_n) = 0,$$

so we are left with the result

$$R^n = -\frac{1}{2}u_e''(t_n)\Delta t + \mathcal{O}(\Delta t^2)\,.$$

We see that the variable coefficients do not pose any additional difficultie case. Exercise 8 takes the analysis above one step further to the Crank-l scheme.

3.8 Exact solutions of the finite difference equation

Having a mathematical expression for the numerical solution is very val program verification since we then know the exact numbers that the p should produce. Looking at the various formulas for the truncation e (5)-(6) and (25)-(26) in Section 2.4, we see that all but two of the R exp contains a second or higher order derivative of u_e. The exceptions geometric and harmonic means where the truncation error involves u_e' a u_e in case of the harmonic mean. So, apart from these two means, choo to be a linear function of t, $u_e = ct + d$ for constants c and d, will m truncation error vanish since $u_e'' = 0$. Consqeuently, the truncation er finite difference scheme will be zero since the various approximations u all be exact. This means that the linear solution is an exact solutio discrete equations.

In a particular differential equation problem, the reasoning above used to determine if we expect a linear u_e to fulfill the discrete equati actually prove that this is true, we can either compute the truncation e see that it vanishes, or we can simply insert $u_e(t) = ct + d$ in the sche see that it fulfills the equations. The latter method is usually the sim will often be necessary to add some source term to the ODE in order to linear solution.

Many ODEs are discretized by centered differences. From Sectio see that all the centered difference formulas have truncation errors i u_e''' or higher-order derivatives. A quadratic solution, e.g., $u_e(t) = t^2$ - will then make the truncation errors vanish. This observation can to test if a quadratic solution will fulfill the discrete equations. Note uadratic solution will not obey the equations for a Crank-Nicolson scheme for $' = -au + b$ because the approximation applies an arithmetic mean, which ıvolves a truncation error with u_e''.

.9 Computing truncation errors in nonlinear problems

he general nonlinear ODE

$$u' = f(u, t), \qquad (45)$$

an be solved by a Crank-Nicolson scheme

$$[D_t u' = \overline{f}^t]^{n+\frac{1}{2}}\,. \qquad (46)$$

he truncation error is as always defined as the residual arising when inserting ıe exact solution u_e in the scheme:

$$[D_t u_e' - \overline{f}^t = R]^{n+\frac{1}{2}}\,. \qquad (47)$$

sing (21)-(22) for $\overline{f}^t$ results in

$$\begin{aligned}[\overline{f}^t]^{n+\frac{1}{2}} &= \frac{1}{2}(f(u_e^n, t_n) + f(u_e^{n+1}, t_{n+1})) \\ &= f(u_e^{n+\frac{1}{2}}, t_{n+\frac{1}{2}}) + \frac{1}{8}u_e''(t_{n+\frac{1}{2}})\Delta t^2 + \mathcal{O}(\Delta t^4)\,.\end{aligned}$$

/ith (5)-(6) the discrete equations (47) lead to

$${}_e'(t_{n+\frac{1}{2}}) + \frac{1}{24}u_e'''(t_{n+\frac{1}{2}})\Delta t^2 - f(u_e^{n+\frac{1}{2}}, t_{n+\frac{1}{2}}) - \frac{1}{8}u''(t_{n+\frac{1}{2}})\Delta t^2 + \mathcal{O}(\Delta t^4) = R^{n+\frac{1}{2}}\,.$$

ince $u_e'(t_{n+\frac{1}{2}}) - f(u_e^{n+\frac{1}{2}}, t_{n+\frac{1}{2}}) = 0$, the truncation error becomes

$$R^{n+\frac{1}{2}} = (\frac{1}{24}u_e'''(t_{n+\frac{1}{2}}) - \frac{1}{8}u''(t_{n+\frac{1}{2}}))\Delta t^2\,.$$

he computational techniques worked well even for this nonlinear ODE.

: Truncation errors in vibration ODEs

.1 Linear model without damping

he next example on computing the truncation error involves the following ODE ır vibration problems:

$$u''(t) + \omega^2 u(t) = 0\,. \qquad (48)$$

:ere, ω is a given constant.

The truncation error of a centered finite difference scheme. standard, second-ordered, central difference for the second-order derivati we have the scheme

$$[D_t D_t u + \omega^2 u = 0]^n\,.$$

Inserting the exact solution u_e in this equation and adding a residu that u_e can fulfill the equation results in

$$[D_t D_t u_e + \omega^2 u_e = R]^n\,.$$

To calculate the truncation error R^n, we use (17)-(18), i.e.,

$$[D_t D_t u_e]^n = u_e''(t_n) + \frac{1}{12}u_e''''(t_n)\Delta t^2,$$

and the fact that $u_e''(t) + \omega^2 u_e(t) = 0$. The result is

$$R^n = \frac{1}{12}u_e''''(t_n)\Delta t^2 + \mathcal{O}(\Delta t^4)\,.$$

The truncation error of approximating $u'(0)$. The initial condit (48) are $u(0) = I$ and $u'(0) = V$. The latter involves a finite difference . mation. The standard choice

$$[D_{2t} u = V]^0,$$

where u^{-1} is eliminated with the aid of the discretized ODE for $n = 0$, a centered difference with an $\mathcal{O}(\Delta t^2)$ truncation error given by (7)-(ɛ simpler choice

$$[D_t^+ u = V]^0,$$

is based on a forward difference with a truncation error $\mathcal{O}(\Delta t)$. A question is if this initial error will impact the order of the scheme througl simulation. Exercise 11 asks you to quickly perform an experiment to inv this question.

Truncation error of the equation for the first step. We have shc the truncation error of the difference used to approximate the initial cc $u'(0) = 0$ is $\mathcal{O}(\Delta t^2)$, but can also investigate the difference equation ı the first step. In a truncation error setting, the right way to view this є is not to use the initial condition $[D_{2t} u = V]^0$ to express $u^{-1} = u^1$ · in order to eliminate u^{-1} from the discretized differential equation, other way around: the fundamental equation is the discretized initial cc $[D_{2t} u = V]^0$ and we use the discretized ODE $[D_t D_t + \omega^2 u = 0]^0$ to el u^{-1} in the disretized initial condition. From $[D_t D_t + \omega^2 u = 0]^0$ we hav

$$u^{-1} = 2u^0 - u^1 - \Delta t^2 \omega^2 u^0,$$

hich inserted in $[D_{2t}u = V]^0$ gives

$$\frac{u^1 - u^0}{\Delta t} + \frac{1}{2}\omega^2 \Delta t u^0 = V \,. \qquad (52)$$

he first term can be recognized as a forward difference such that the equation an be written in operator notation as

$$[D_t^+ u + \frac{1}{2}\omega^2 \Delta t u = V]^0 \,.$$

he truncation error is defined as

$$[D_t^+ u_e + \frac{1}{2}\omega^2 \Delta t u_e - V = R]^0 \,.$$

sing (11)-(12) with one more term in the Taylor series, we get that

$$u_e'(0) + \frac{1}{2}u_e''(0)\Delta t + \frac{1}{6}u_e'''(0)\Delta t^2 + \mathcal{O}(\Delta t^3) + \frac{1}{2}\omega^2 \Delta t u_e(0) - V = R^n \,.$$

ow, $u_e'(0) = V$ and $u_e''(0) = -\omega^2 u_e(0)$ so we get

$$R^n = \frac{1}{6}u_e'''(0)\Delta t^2 + \mathcal{O}(\Delta t^3) \,.$$

There is another way of analyzing the discrete initial condition, because iminating u^{-1} via the discretized ODE can be expressed as

$$[D_{2t}u + \Delta t(D_t D_t u - \omega^2 u) = V]^0 \,. \qquad (53)$$

/riting out (53) shows that the equation is equivalent to (52). The truncation :ror is defined by

$$[D_{2t}u_e + \Delta t(D_t D_t u_e - \omega^2 u_e) = V + R]^0 \,.$$

eplacing the difference via (7)-(8) and (17)-(18), as well as using $u_e'(0) = V$ nd $u_e''(0) = -\omega^2 u_e(0)$, gives

$$R^n = \frac{1}{6}u'''(0)\Delta t^2 + \mathcal{O}(\Delta t^3) \,.$$

:omputing correction terms. The idea of using correction terms to increase ne order of R^n can be applied as described in Section 3.6. We look at

$$[D_t D_t u_e + \omega^2 u_e = C + R]^n,$$

nd observe that C^n must be chosen to cancel the Δt^2 term in R^n. That is,

$$C^n = \frac{1}{12}u_e''''(t_n)\Delta t^2 \,.$$

o get rid of the 4th-order derivative we can use the differential equation: $'' = -\omega^2 u$, which implies $u'''' = \omega^4 u$. Adding the correction term to the ODE esults in

$$u'' + \omega^2(1 - \frac{1}{12}\omega^2 \Delta t^2)u = 0 \,.$$

Solving this equation by the standard scheme

$$[D_t D_t u + \omega^2(1 - \frac{1}{12}\omega^2 \Delta t^2)u = 0]^n,$$

will result in a scheme with trunction error $\mathcal{O}(\Delta t^4)$.

We can use another set of arguments to justify that (54) leads to a high method. Mathematical analysis of the scheme (49) reveals that the nu frequency $\tilde{\omega}$ is (approximately as $\Delta t \to 0$)

$$\tilde{\omega} = \omega(1 + \frac{1}{24}\omega^2 \Delta t^2) \,.$$

One can therefore attempt to replace ω in the ODE by a slightly smaller the numerics will make it larger:

$$[u'' + (\omega(1 - \frac{1}{24}\omega^2 \Delta t^2))^2 u = 0 \,.$$

Expanding the squared term and omitting the higher-order term Δ exactly the ODE (54). Experiments show that u^n is computed to 4th Δt.

4.2 Model with damping and nonlinearity

The model (48) can be extended to include damping $\beta u'$, a nonlinear r (spring) force $s(u)$, and some known excitation force $F(t)$:

$$mu'' + \beta u' + s(u) = F(t) \,.$$

The coefficient m usually represents the mass of the system. This gc equation can by discretized by centered differences:

$$[mD_t D_t u + \beta D_{2t}u + s(u) = F]^n \,.$$

The exact solution u_e fulfills the discrete equations with a residual tern

$$[mD_t D_t u_e + \beta D_{2t}u_e + s(u_e) = F + R]^n \,.$$

Using (17)-(18) and (7)-(8) we get

$$[mD_t D_t u_e + \beta D_{2t}u_e]^n = mu_e''(t_n) + \beta u_e'(t_n) + \left(\frac{m}{12}u_e''''(t_n) + \frac{\beta}{6}u_e'''(t_n)\right)\Delta t^2 + \mathcal{O}(\Delta t^4$$

Combining this with the previous equation, we can collect the terms

$$mu_e''(t_n) + \beta u_e'(t_n) + \omega^2 u_e(t_n) + s(u_e(t_n)) - F^n,$$

ıd set this sum to zero because u_e solves the differential equation. We are left ith the truncation error

$$R^n = \left(\frac{m}{12}u_e''''(t_n) + \frac{\beta}{6}u_e'''(t_n)\right)\Delta t^2 + \mathcal{O}(\Delta t^4), \quad (58)$$

ɔ the scheme is of second order.

According to (58), we can add correction terms

$$C^n = \left(\frac{m}{12}u_e''''(t_n) + \frac{\beta}{6}u_e'''(t_n)\right)\Delta t^2,$$

ɔ the right-hand side of the ODE to obtain a fourth-order scheme. However, xpressing u'''' and u''' in terms of lower-order derivatives is now harder because ıe differential equation is more complicated:

$$\begin{aligned}
u''' &= \frac{1}{m}(F' - \beta u'' - s'(u)u'),\\
u'''' &= \frac{1}{m}(F'' - \beta u''' - s''(u)(u')^2 - s'(u)u''),\\
&= \frac{1}{m}(F'' - \beta\frac{1}{m}(F' - \beta u'' - s'(u)u') - s''(u)(u')^2 - s'(u)u'') .
\end{aligned}$$

; is not impossible to discretize the resulting modified ODE, but it is up to ebate whether correction terms are feasible and the way to go. Computing with smaller Δt is usually always possible in these problems to achieve the desired ccuracy.

.3 Extension to quadratic damping

ıstead of the linear damping term $\beta u'$ in (55) we now consider quadratic amping $\beta|u'|u'$:

$$mu'' + \beta|u'|u' + s(u) = F(t) . \quad (59)$$

centered difference for u' gives rise to a nonlinearity, which can be linearized sing a geometric mean: $[|u'|u']^n \approx |[u']^{n-\frac{1}{2}}|[u']^{n+\frac{1}{2}}$. The resulting scheme ecomes

$$[mD_tD_tu]^n + \beta|[D_tu]^{n-\frac{1}{2}}|[D_tu]^{n+\frac{1}{2}} + s(u^n) = F^n . \quad (60)$$

he truncation error is defined through

$$[mD_tD_tu_e]^n + \beta|[D_tu_e]^{n-\frac{1}{2}}|[D_tu_e]^{n+\frac{1}{2}} + s(u_e^n) - F^n = R^n . \quad (61)$$

We start with expressing the truncation error of the geometric mean. Acɔrding to (23)-(24),

$$D_tu_e]^{n-\frac{1}{2}}|[D_tu_e]^{n+\frac{1}{2}} = [|D_tu_e|D_tu_e]^n - \frac{1}{4}u'(t_n)^2\Delta t^2 + \frac{1}{4}u(t_n)u''(t_n)\Delta t^2 + \mathcal{O}(\Delta t^4) .$$

Using (5)-(6) for the D_tu_e factors results in

$$[|D_tu_e|D_tu_e]^n = |u_e' + \frac{1}{24}u_e'''(t_n)\Delta t^2 + \mathcal{O}(\Delta t^4)|(u_e' + \frac{1}{24}u_e'''(t_n)\Delta t^2 + \mathcal{C}$$

We can remove the absolute value since it essentially gives a factor 1 or Calculating the product, we have the leading-order terms

$$[D_tu_eD_tu_e]^n = (u_e'(t_n))^2 + \frac{1}{12}u_e(t_n)u_e'''(t_n)\Delta t^2 + \mathcal{O}(\Delta t^4) .$$

With

$$m[D_tD_tu_e]^n = mu_e''(t_n) + \frac{m}{12}u_e''''(t_n)\Delta t^2 + \mathcal{O}(\Delta t^4),$$

and using the differential equation on the form $mu'' + \beta(u')^2 + s(u) =$ end up with

$$R^n = (\frac{m}{12}u_e''''(t_n) + \frac{\beta}{12}u_e(t_n)u_e'''(t_n))\Delta t^2 + \mathcal{O}(\Delta t^4) .$$

This result demonstrates that we have second-order accuracy also with qu damping. The key elements that lead to the second-order accuracy is t difference approximations are $\mathcal{O}(\Delta t^2)$ *and* the geometric mean approxin also of $\mathcal{O}(\Delta t^2)$.

4.4 The general model formulated as first-order OD

The second-order model (59) can be formulated as a first-order system,

$$\begin{aligned}
u' &= v,\\
v' &= \frac{1}{m}\left(F(t) - \beta|v|v - s(u)\right) .
\end{aligned}$$

The system (62)-(62) can be solved either by a forward-backward sche centered scheme on a staggered mesh.

The forward-backward scheme. The discretization is based on the stepping (62) forward in time and then using a backward difference in (6 the recently computed (and therefore known) u:

$$\begin{aligned}
&[D_t^+u = v]^n,\\
&[D_t^-v = \frac{1}{m}(F(t) - \beta|v|v - s(u))]^{n+1} .
\end{aligned}$$

The term $|v|v$ gives rise to a nonlinearity $|v^{n+1}|v^{n+1}$, which can be linea $|v^n|v^{n+1}$:

$$[D_t^+ u = v]^n, \quad (66)$$

$$[D_t^- v]^{n+1} = \frac{1}{m}(F(t_{n+1}) - \beta|v^n|v^{n+1} - s(u^{n+1}))\,. \quad (67)$$

Each ODE will have a truncation error when inserting the exact solutions $u_{\rm e}$ nd $v_{\rm e}$ in (64)-(65):

$$[D_t^+ u_{\rm e} = v_{\rm e} + R_u]^n, \quad (68)$$

$$[D_t^- v_{\rm e}]^{n+1} = \frac{1}{m}(F(t_{n+1}) - \beta|v_{\rm e}(t_n)|v_{\rm e}(t_{n+1}) - s(u_{\rm e}(t_{n+1}))) + R_v^{n+1}\,. \quad (69)$$

pplication of (11)-(12) and (9)-(10) in (68) and (69), respectively, gives

$$u_{\rm e}'(t_n) + \frac{1}{2}u_{\rm e}''(t_n)\Delta t + \mathcal{O}(\Delta t^2) = v_{\rm e}(t_n) + R_u^n, \quad (70)$$

$$v_{\rm e}'(t_{n+1}) - \frac{1}{2}v_{\rm e}''(t_{n+1})\Delta t + \mathcal{O}(\Delta t^2) = \frac{1}{m}(F(t_{n+1}) - \beta|v_{\rm e}(t_n)|v_{\rm e}(t_{n+1}) + s(u_{\rm e}(t_{n+1})) + R_v^n\,. \quad (71)$$

ince $u_{\rm e}' = v_{\rm e}$, (70) gives

$$R_u^n = \frac{1}{2}u_{\rm e}''(t_n)\Delta t + \mathcal{O}(\Delta t^2)\,.$$

ı (71) we can collect the terms that constitute the ODE, but the damping term as the wrong form. Let us drop the absolute value in the damping term for mplicity. Adding a substracting the right form $v^{n+1}v^{n+1}$ helps:

$$v_{\rm e}'(t_{n+1}) - \frac{1}{m}(F(t_{n+1}) - \beta v_{\rm e}(t_{n+1})v_{\rm e}(t_{n+1}) + s(u_{\rm e}(t_{n+1})) + (\beta v_{\rm e}(t_n)v_{\rm e}(t_{n+1}) - \beta v_{\rm e}(t_{n+1})v_{\rm e}(t_{n+1}))),$$

hich reduces to

$$\frac{\beta}{m}v_{\rm e}(t_{n+1}(v_{\rm e}(t_n) - v_{\rm e}(t_{n+1})) = \frac{\beta}{m}v_{\rm e}(t_{n+1}[D_t^- v_{\rm e}]^{n+1}\Delta t = \frac{\beta}{m}v_{\rm e}(t_{n+1}(v_{\rm e}'(t_{n+1})\Delta t + -\frac{1}{2}v_{\rm e}'''(t_{n+1})\Delta t^+\mathcal{O}(\Delta t^3))\,.$$

/e end with R_u^n and R_v^{n+1} as $\mathcal{O}(\Delta t)$, simply because all the building blocks in ıe schemes (the forward and backward differences and the linearization trick) re only first-order accurate. However, this analysis is misleading: the building locks play together in a way that makes the scheme second-order accurate. This shown by considering an alternative, yet equivalent, formulation of the above cheme.

A centered scheme on a staggered mesh. We now introduce a st mesh where we seek u at mesh points t_n and v at points $t_{n+\frac{1}{2}}$ in betwe points. The staggered mesh makes it easy to formulate centered differ the system (62)-(62):

$$[D_t u = v]^{n-\frac{1}{2}},$$

$$[D_t v = \frac{1}{m}(F(t) - \beta|v|v - s(u))]^n\,.$$

The term $|v^n|v^n$ causes trouble since v^n is not computed, only $v^{n-\frac{1}{2}}$ ar Using geometric mean, we can express $|v^n|v^n$ in terms of known qu $|v^n|v^n \approx |v^{n-\frac{1}{2}}|v^{n+\frac{1}{2}}$. We then have

$$[D_t u]^{n-\frac{1}{2}} = v^{n-\frac{1}{2}},$$

$$[D_t v]^n = \frac{1}{m}(F(t_n) - \beta|v^{n-\frac{1}{2}}|v^{n+\frac{1}{2}} - s(u^n))\,.$$

The truncation error in each equation fulfills

$$[D_t u_{\rm e}]^{n-\frac{1}{2}} = v_{\rm e}(t_{n-\frac{1}{2}}) + R_u^{n-\frac{1}{2}},$$

$$[D_t v_{\rm e}]^n = \frac{1}{m}(F(t_n) - \beta|v_{\rm e}(t_{n-\frac{1}{2}})|v_{\rm e}(t_{n+\frac{1}{2}}) - s(u^n)) + R_v^n\,.$$

The truncation error of the centered differences is given by (5)-(6), geometric mean approximation analysis can be taken from (23)-(24). results lead to

$$u_{\rm e}'(t_{n-\frac{1}{2}}) + \frac{1}{24}u_{\rm e}'''(t_{n-\frac{1}{2}})\Delta t^2 + \mathcal{O}(\Delta t^4) = v_{\rm e}(t_{n-\frac{1}{2}}) + R_u^{n-\frac{1}{2}},$$

and

$$v_{\rm e}'(t_n) = \frac{1}{m}(F(t_n) - \beta|v_{\rm e}(t_n)|v_{\rm e}(t_n) + \mathcal{O}(\Delta t^2) - s(u^n)) + R_v^n\,.$$

The ODEs fulfilled by $u_{\rm e}$ and $v_{\rm e}$ are evident in these equations, and we second-order accuracy for the truncation error in both equations:

$$R_u^{n-\frac{1}{2}} = \mathcal{O}(\Delta t^2), \quad R_v^n = \mathcal{O}(\Delta t^2)\,.$$

Comparing (74)-(75) with (66)-(67), we can hopefully realize tha schemes are equivalent (which becomes clear when we implement bot obvious advantage with the staggered mesh approach is that we can all use second-order accurate building blocks and in this way concince o that the resulting scheme has an error of $\mathcal{O}(\Delta t^2)$.

Truncation errors in wave equations

.1 Linear wave equation in 1D

he standard, linear wave equation in 1D for a function $u(x,t)$ reads

$$\frac{\partial^2 u}{\partial t^2} = c^2\frac{\partial^2 u}{\partial x^2} + f(x,t), \quad x\in(0,L),\ t\in(0,T], \tag{76}$$

here c is the constant wave velocity of the physical medium $[0,L]$. The equation an also be more compactly written as

$$u_{tt} = c^2u_{xx} + f, \quad x\in(0,L),\ t\in(0,T], \tag{77}$$

entered, second-order finite differences are a natural choice for discretizing the erivatives, leading to

$$[D_tD_tu = c^2D_xD_xu + f]^n_i\,. \tag{78}$$

Inserting the exact solution $u_{\rm e}(x,t)$ in (78) makes this function fulfill the quation if we add the term R:

$$[D_tD_tu_{\rm e} = c^2D_xD_xu_{\rm e} + f + R]^n_i \tag{79}$$

Our purpose is to calculate the truncation error R. From (17)-(18) we have ıat

$$[D_tD_tu_{\rm e}]^n_i = u_{{\rm e},tt}(x_i,t_n) + \frac{1}{12}u_{{\rm e},tttt}(x_i,t_n)\Delta t^2 + \mathcal{O}(\Delta t^4),$$

hen we use a notation taking into account that $u_{\rm e}$ is a function of two variables nd that derivatives must be partial derivatives. The notation $u_{{\rm e},tt}$ means $^2u_{\rm e}/\partial t^2$.

The same formula may also be applied to the x-derivative term:

$$[D_xD_xu_{\rm e}]^n_i = u_{{\rm e},xx}(x_i,t_n) + \frac{1}{12}u_{{\rm e},xxxx}(x_i,t_n)\Delta x^2 + \mathcal{O}(\Delta x^4),$$

quation (81) now becomes

$$u_{{\rm e},tt} + \frac{1}{12}u_{{\rm e},tttt}(x_i,t_n)\Delta t^2 = c^2u_{{\rm e},xx} + c^2\frac{1}{12}u_{{\rm e},xxxx}(x_i,t_n)\Delta x^2 + f(x_i,t_n) + \mathcal{O}(\Delta t^4,\Delta x^4) + R^n_i\,.$$

ecause $u_{\rm e}$ fulfills the partial differential equation (PDE) (77), the first, third, nd fifth terms cancel out, and we are left with

$$R^n_i = \frac{1}{12}u_{{\rm e},tttt}(x_i,t_n)\Delta t^2 - c^2\frac{1}{12}u_{{\rm e},xxxx}(x_i,t_n)\Delta x^2 + \mathcal{O}(\Delta t^4,\Delta x^4), \tag{80}$$

ıowing that the scheme (78) is of second order in the time and space mesh ɔacing.

5.2 Finding correction terms

Can we add correction terms to the PDE and increase the order of R^n_i The starting point is

$$[D_tD_tu_{\rm e} = c^2D_xD_xu_{\rm e} + f + C + R]^n_i$$

From the previous analysis we simply get (80) again, but now with C:

$$R^n_i + C^n_i = \frac{1}{12}u_{{\rm e},tttt}(x_i,t_n)\Delta t^2 - c^2\frac{1}{12}u_{{\rm e},xxxx}(x_i,t_n)\Delta x^2 + \mathcal{O}(\Delta t^4,\Delta x$$

The idea is to let C^n_i cancel the Δt^2 and Δx^2 terms to make $R^n_i = \mathcal{O}(\Delta t$

$$C^n_i = \frac{1}{12}u_{{\rm e},tttt}(x_i,t_n)\Delta t^2 - c^2\frac{1}{12}u_{{\rm e},xxxx}(x_i,t_n)\Delta x^2\,.$$

Essentially, it means that we add a new term

$$C = \frac{1}{12}\left(u_{tttt}\Delta t^2 - c^2u_{xxxx}\Delta x^2\right),$$

to the right-hand side of the PDE. We must either discretize these 4t derivatives directly or rewrite them in terms of lower-order derivatives v aid of the PDE. The latter approach is more feasible. From the PDE that

$$\frac{\partial^2}{\partial t^2} = c^2\frac{\partial^2}{\partial x^2},$$

so

$$u_{tttt} = c^2u_{xxtt}, \quad u_{xxxx} = c^{-2}u_{ttxx}\,.$$

Assuming u is smooth enough that $u_{xxtt} = u_{ttxx}$, these relations lead t

$$C = \frac{1}{12}((c^2\Delta t^2 - \Delta x^2)u_{xx})_{tt}\,.$$

A natural discretization is

$$C^n_i = \frac{1}{12}((c^2\Delta t^2 - \Delta x^2)[D_xD_xD_tD_tu]^n_i\,.$$

Writing out $[D_xD_xD_tD_tu]^n_i$ as $[D_xD_x(D_tD_tu)]^n_i$ gives

$$\frac{1}{\Delta t^2}\Bigg(\frac{u^{n+1}_{i+1} - 2u^n_{i+1} + u^{n-1}_{i+1}}{\Delta x^2} - 2 \\ \frac{u^{n+1}_{i} - 2u^n_{i} + u^{n-1}_{i}}{\Delta x^2} + \frac{u^{n+1}_{i-1} - 2u^n_{i-1} + u^{n-1}_{i-1}}{\Delta x^2}\Bigg)$$

Now the unknown values u^{n+1}_{i+1}, u^{n+1}_i, and u^{n+1}_{i-1} are *coupled*, and we mu a tridiagonal system to find them. This is in principle straightforward results in an implicit finite difference schemes, while we had a convenient scheme without the correction terms.

.3 Extension to variable coefficients

ow we address the variable coefficient version of the linear 1D wave equation,

$$\frac{\partial^2 u}{\partial t^2} = \frac{\partial}{\partial x}\left(\lambda(x)\frac{\partial u}{\partial x}\right),$$

r written more compactly as

$$u_{tt} = (\lambda u_x)_x\,. \tag{83}$$

he discrete counterpart to this equation, using arithmetic mean for λ and entered differences, reads

$$[D_tD_t u = D_x\overline{\lambda}^x D_x u]^n_i\,. \tag{84}$$

he truncation error is the residual R in the equation

$$[D_tD_t u_{\rm e} = D_x\overline{\lambda}^x D_x u_{\rm e} + R]^n_i\,. \tag{85}$$

he difficulty in the present is how to compute the truncation error of the term $\mathrm{)}_x\overline{\lambda}^x D_x u_{\rm e}]^n_i$.

We start by writing out the outer operator:

$$[D_x\overline{\lambda}^x D_x u_{\rm e}]^n_i = \frac{1}{\Delta x}\left([\overline{\lambda}^x D_x u_{\rm e}]^n_{i+\frac{1}{2}} - [\overline{\lambda}^x D_x u_{\rm e}]^n_{i-\frac{1}{2}}\right). \tag{86}$$

/ith the aid of (5)-(6) and (21)-(22) we have

$$\begin{aligned}
[D_x u_{\rm e}]^n_{i+\frac{1}{2}} &= u_{{\rm e},x}(x_{i+\frac{1}{2}},t_n) + \frac{1}{24}u_{{\rm e},xxx}(x_{i+\frac{1}{2}},t_n)\Delta x^2 + \mathcal{O}(\Delta x^4),\\
[\overline{\lambda}^x]_{i+\frac{1}{2}} &= \lambda(x_{i+\frac{1}{2}}) + \frac{1}{8}\lambda''(x_{i+\frac{1}{2}})\Delta x^2 + \mathcal{O}(\Delta x^4),\\
[\overline{\lambda}^x D_x u_{\rm e}]^n_{i+\frac{1}{2}} &= (\lambda(x_{i+\frac{1}{2}}) + \frac{1}{8}\lambda''(x_{i+\frac{1}{2}})\Delta x^2 + \mathcal{O}(\Delta x^4))\times\\
&\quad (u_{{\rm e},x}(x_{i+\frac{1}{2}},t_n) + \frac{1}{24}u_{{\rm e},xxx}(x_{i+\frac{1}{2}},t_n)\Delta x^2 + \mathcal{O}(\Delta x^4))\\
&= \lambda(x_{i+\frac{1}{2}})u_{{\rm e},x}(x_{i+\frac{1}{2}},t_n) + \lambda(x_{i+\frac{1}{2}})\frac{1}{24}u_{{\rm e},xxx}(x_{i+\frac{1}{2}},t_n)\Delta x^2 +\\
&\quad u_{{\rm e},x}(x_{i+\frac{1}{2}})\frac{1}{8}\lambda''(x_{i+\frac{1}{2}})\Delta x^2 + \mathcal{O}(\Delta x^4)\\
&= [\lambda u_{{\rm e},x}]^n_{i+\frac{1}{2}} + G^n_{i+\frac{1}{2}}\Delta x^2 + \mathcal{O}(\Delta x^4),
\end{aligned}$$

here we have introduced the short form

$$G^n_{i+\frac{1}{2}} = (\frac{1}{24}u_{{\rm e},xxx}(x_{i+\frac{1}{2}},t_n)\lambda((x_{i+\frac{1}{2}}) + u_{{\rm e},x}(x_{i+\frac{1}{2}},t_n)\frac{1}{8}\lambda''(x_{i+\frac{1}{2}}))\Delta x^2\,.$$

imilarly, we find that

$$[\overline{\lambda}^x D_x u_{\rm e}]^n_{i-\frac{1}{2}} = [\lambda u_{{\rm e},x}]^n_{i-\frac{1}{2}} + G^n_{i-\frac{1}{2}}\Delta x^2 + \mathcal{O}(\Delta x^4)\,.$$

Inserting these expressions in the outer operator (86) results in

$$\begin{aligned}
[D_x\overline{\lambda}^x D_x u_{\rm e}]^n_i &= \frac{1}{\Delta x}([\overline{\lambda}^x D_x u_{\rm e}]^n_{i+\frac{1}{2}} - [\overline{\lambda}^x D_x u_{\rm e}]^n_{i-\frac{1}{2}})\\
&= \frac{1}{\Delta x}([\lambda u_{{\rm e},x}]^n_{i+\frac{1}{2}} + G^n_{i+\frac{1}{2}}\Delta x^2 - [\lambda u_{{\rm e},x}]^n_{i-\frac{1}{2}} - G^n_{i-\frac{1}{2}}\Delta x^2 +\\
&= [D_x\lambda u_{{\rm e},x}]^n_i + [D_x G]^n_i\Delta x^2 + \mathcal{O}(\Delta x^4)\,.
\end{aligned}$$

The reason for $\mathcal{O}(\Delta x^4)$ in the remainder is that there are coefficients of this term, say $H\Delta x^4$, and the subtraction and division by Δx re $[D_x H]^n_i\Delta x^4$.

We can now use (5)-(6) to express the D_x operator in $[D_x\lambda u_{{\rm e},x}$ derivative and a truncation error:

$$[D_x\lambda u_{{\rm e},x}]^n_i = \frac{\partial}{\partial x}\lambda(x_i)u_{{\rm e},x}(x_i,t_n) + \frac{1}{24}(\lambda u_{{\rm e},x})_{xxx}(x_i,t_n)\Delta x^2 + \mathcal{O}(\Delta$$

Expressions like $[D_x G]^n_i\Delta x^2$ can be treated in an identical way,

$$[D_x G]^n_i\Delta x^2 = G_x(x_i,t_n)\Delta x^2 + \frac{1}{24}G_{xxx}(x_i,t_n)\Delta x^4 + \mathcal{O}(\Delta x^4)\,.$$

There will be a number of terms with the Δx^2 factor. We lump th into $\mathcal{O}(\Delta x^2)$. The result of the truncation error analysis of the spatial de is therefore summarized as

$$[D_x\overline{\lambda}^x D_x u_{\rm e}]^n_i = \frac{\partial}{\partial x}\lambda(x_i)u_{{\rm e},x}(x_i,t_n) + \mathcal{O}(\Delta x^2)\,.$$

After having treated the $[D_tD_t u_{\rm e}]^n_i$ term as well, we achieve

$$R^n_i = \mathcal{O}(\Delta x^2) + \frac{1}{12}u_{{\rm e},tttt}(x_i,t_n)\Delta t^2\,.$$

The main conclusion is that the scheme is of second-order in time an also in this variable coefficient case. The key ingredients for second o the centered differences and the arithmetic mean for λ: all those buildin feature second-order accuracy.

5.4 1D wave equation on a staggered mesh

5.5 Linear wave equation in 2D/3D

The two-dimensional extension of (76) takes the form

$$\frac{\partial^2 u}{\partial t^2} = c^2\left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}\right) + f(x,y,t),\quad (x,y)\in(0,L)\times(0,H),\ t\in(0,$$

here now $c(x, y)$ is the constant wave velocity of the physical medium $[0, L] \times$), $H]$. In the compact notation, the PDE (87) can be written

$$u_{tt} = c^2(u_{xx} + u_{yy}) + f(x, y, t), \quad (x, y) \in (0, L) \times (0, H), \ t \in (0, T], \tag{88}$$

ı 2D, while the 3D version reads

$$u_{tt} = c^2(u_{xx} + u_{yy} + u_{zz}) + f(x, y, z, t), \tag{89}$$

or $(x, y, z) \in (0, L) \times (0, H) \times (0, B)$ and $t \in (0, T]$.

Approximating the second-order derivatives by the standard formulas (17)-.8) yields the scheme

$$[D_tD_tu = c^2(D_xD_xu + D_yD_yu) + f]^n_{i,j,k} . \tag{90}$$

he truncation error is found from

$$[D_tD_tu_e = c^2(D_xD_xu_e + D_yD_yu_e) + f + R]^n . \tag{91}$$

he calculations from the 1D case can be repeated to the terms in the y and z irections. Collecting terms that fulfill the PDE, we end up with

$$R^n_{i,j,k} = [\frac{1}{12}u_{e,tttt}\Delta t^2 - c^2\frac{1}{12}\left(u_{e,xxxx}\Delta x^2 + u_{e,yyyy}\Delta x^2 + u_{e,zzzz}\Delta z^2\right)]^n_{i,j,k} + \tag{92}$$

$$\mathcal{O}(\Delta t^4, \Delta x^4, \Delta y^4, \Delta z^4) .$$

; Truncation errors in diffusion equations

.1 Linear diffusion equation in 1D

he standard, linear, 1D diffusion equation takes the form

$$\frac{\partial u}{\partial t} = \alpha\frac{\partial^2 u}{\partial x^2} + f(x, t), \quad x \in (0, L), \ t \in (0, T], \tag{93}$$

here $\alpha > 0$ is the constant diffusion coefficient. A more compact form of the iffusion equation is $u_t = \alpha u_{xx} + f$.

The spatial derivative in the diffusion equation, αu_xx, is commonly discretized s $[D_xD_xu]^n_i$. The time-derivative, however, can be treated by a variety of ıethods.

'he Forward Euler scheme in time. Let us start with the simple Forward uler scheme:

$$[D_t^+u = \alpha D_xD_xu + f]^n .$$

The truncation error arises as the residual R when inserting the exact : u_e in the discrete equations:

$$[D_t^+u_e = \alpha D_xD_xu_e + f + R]^n_i .$$

Now, using (11)-(12) and (17)-(18), we can transform the difference oper derivatives:

$$u_{e,t}(x_i, t_n) + \frac{1}{2}u_{e,tt}(t_n)\Delta t + \mathcal{O}(\Delta t^2) = \alpha u_{e,xx}(x_i, t_n) + \frac{\alpha}{12}u_{e,xxxx}(x_i, t_n)\Delta x^2 + \mathcal{O}(\Delta x^4) + f(x_i, t_n) + R^n_i .$$

The terms $u_{e,t}(x_i, t_n) - \alpha u_{e,xx}(x_i, t_n) - f(x_i, t_n)$ vansih because u_e so PDE. The truncation error then becomes

$$R^n_i = \frac{1}{2}u_{e,tt}(t_n)\Delta t + \mathcal{O}(\Delta t^2) - \frac{\alpha}{12}u_{e,xxxx}(x_i, t_n)\Delta x^2 + \mathcal{O}(\Delta x^4)$$

The Crank-Nicolson scheme in time. The Crank-Nicolson method of using a centered difference for u_t and an arithmetic average of the $u_{\text{:}}$

$$[D_tu]^{n+\frac{1}{2}}_i = \alpha\frac{1}{2}([D_xD_xu]^n_i + [D_xD_xu]^{n+1}_i + f^{n+\frac{1}{2}}_i .$$

The equation for the truncation error is

$$[D_tu_e]^{n+\frac{1}{2}}_i = \alpha\frac{1}{2}([D_xD_xu_e]^n_i + [D_xD_xu_e]^{n+1}_i) + f^{n+\frac{1}{2}}_i + R^{n+\frac{1}{2}}_i .$$

To find the truncation error, we start by expressing the arithmetic av terms of values at time $t_{n+\frac{1}{2}}$. According to (21)-(22),

$$\frac{1}{2}([D_xD_xu_e]^n_i + [D_xD_xu_e]^{n+1}_i) = [D_xD_xu_e]^{n+\frac{1}{2}}_i + \frac{1}{8}[D_xD_xu_{e,tt}]^{n+\frac{1}{2}}_i\Delta t^2 + ($$

With (17)-(18) we can express the difference operator D_xD_xu in ter derivative:

$$[D_xD_xu_e]^{n+\frac{1}{2}}_i = u_{e,xx}(x_i, t_{n+\frac{1}{2}}) + \frac{1}{12}u_{e,xxxx}(x_i, t_{n+\frac{1}{2}})\Delta x^2 + \mathcal{O}(\Delta x$$

The error term from the arithmetic mean is similarly expanded,

$$\frac{1}{8}[D_xD_xu_{e,tt}]^{n+\frac{1}{2}}_i\Delta t^2 = \frac{1}{8}u_{e,ttxx}(x_i, t_{n+\frac{1}{2}})\Delta t^2 + \mathcal{O}(\Delta t^2\Delta x^2)$$

The time derivative is analyzed using (5)-(6):

$$[D_tu]^{n+\frac{1}{2}}_i = u_{e,t}(x_i, t_{n+\frac{1}{2}}) + \frac{1}{24}u_{e,ttt}(x_i, t_{n+\frac{1}{2}})\Delta t^2 + \mathcal{O}(\Delta t^4) .$$

Summing up all the contributions and notifying that

$$u_{e,t}(x_i, t_{n+\frac{1}{2}}) = \alpha u_{e,xx}(x_i, t_{n+\frac{1}{2}}) + f(x_i, t_{n+\frac{1}{2}}),$$

ne truncation error is given by

$$R_i^{n+\frac{1}{2}} = \frac{1}{8}u_{e,xx}(x_i, t_{n+\frac{1}{2}})\Delta t^2 + \frac{1}{12}u_{e,xxxx}(x_i, t_{n+\frac{1}{2}})\Delta x^2 + \frac{1}{24}u_{e,ttt}(x_i, t_{n+\frac{1}{2}})\Delta t^2 + +\mathcal{O}(\Delta x^4) + \mathcal{O}(\Delta t^4) + \mathcal{O}(\Delta t^2 \Delta x^2)$$

.2 Linear diffusion equation in 2D/3D

.3 A nonlinear diffusion equation in 2D

Exercises

Exercise 1: Truncation error of a weighted mean

Derive the truncation error of the weighted mean in (19)-(20).

Hint. Expand u_e^{n+1} and u_e^n around $t_{n+\theta}$.
Filename: `trunc_weighted_mean.pdf`.

Exercise 2: Simulate the error of a weighted mean

We consider the weighted mean

$$u_e(t_n) \approx \theta u_e^{n+1} + (1-\theta)u_e^n\,.$$

Choose some specific function for $u_e(t)$ and compute the error in this approximation for a sequence of decreasing $\Delta t = t_{n+1} - t_n$ and for $\theta = 0, 0.25, 0.5, 0.75, 1$. Assuming that the error equals $C\Delta t^r$, for some constants C and r, compute r for the two smallest Δt values for each choice of θ and compare with the truncation error (19)-(20). Filename: `trunc_theta_avg.py`.

Exercise 3: Verify a truncation error formula

Set up a numerical experiment as explained in Section 3.5 for verifying the formulas (15)-(16). Filename: `trunc_backward_2level.py`.

Exercise 4: Truncation error of the Backward Euler scheme

Derive the truncation error of the Backward Euler scheme for the decay ODE $u' = -au$ with constant a. Extend the analysis to cover the variable-coefficient case $u' = -a(t)u + b(t)$. Filename: `trunc_decay_BE.py`.

Exercise 5: Empirical estimation of truncation errors

Use the ideas and tools from Section 3.5 to estimate the rate of the tion error of the Backward Euler and Crank-Nicolson schemes applied exponential decay model $u' = -au$, $u(0) = I$.

Hint. In the Backward Euler scheme, the truncation error can be es at mesh points $n = 1, \ldots, N$, while the truncation error must be es at midpoints $t_{n+\frac{1}{2}}$, $n = 0, \ldots, N-1$ for the Crank-Nicolson schem `truncation_error(dt, N)` function to be supplied to the `estimate` f needs to carefully implement these details and return the right `t` array su `t[i]` is the time point corresponding to the quantities `R[i]` and `R_a[i` Filename: `trunc_decay_BNCN.py`.

Exercise 6: Correction term for a Backward Euler sc

Consider the model $u' = -au$, $u(0) = I$. Use the ideas of Section 3.6 t correction term to the ODE such that the Backward Euler scheme ap the perturbed ODE problem is of second order in Δt. Find the ampli factor. Filename: `trunc_decay_BE_corr.pdf`.

Exercise 7: Verify the effect of correction terms

The program `decay_convrate.py`[4] solves $u' = -au$, $u(0) = I$, by the θ- computes convergence rates. Copy this file and adjust a in the `solver` f such that it incorporates correction terms. Run the program to verify t error from the Forward and Backward Euler schemes with perturbed a is while the error arising from the Crank-Nicolson scheme with perturb $\mathcal{O}(\Delta t^4)$. Filename: `trunc_decay_corr_verify.py`.

Exercise 8: Truncation error of the Crank-Nicolson sc

The variable-coefficient ODE $u' = -a(t)u + b(t)$ can be discretized in two ways by the Crank-Nicolson scheme, depending on whether we use aver a and b or compute them at the midpoint $t_{n+\frac{1}{2}}$:

$$[D_t u = -a\overline{u}^t + b]^{n+\frac{1}{2}},$$

$$[D_t u = \overline{-au + b}^t]^{n+\frac{1}{2}}\,.$$

Compute the truncation error in both cases. Filename: `trunc_decay_CN_`

[4] `http://tinyurl.com/jvzzcfn/decay/decay_convrate.py`

:xercise 9: Truncation error of $u' = f(u,t)$

'onsider the general nonlinear first-order scalar ODE

$$u'(t) = f(u(t), t)\,.$$

how that the truncation error in the Forward Euler scheme,

$$[D_t^+ u = f(u,t)]^n,$$

nd in the Backward Euler scheme,

$$[D_t^- u = f(u,t)]^n,$$

oth are of first order, regardless of what f is.

Showing the order of the truncation error in the Crank-Nicolson scheme,

$$[D_t u = f(u,t)]^{n+\frac{1}{2}},$$

somewhat more involved: Taylor expand u_e^n, u_e^{n+1}, $f(u_e^n, t_n)$, and $f(u_e^{n+1}, t_{n+1})$ round $t_{n+\frac{1}{2}}$, and use that

$$\frac{df}{dt} = \frac{\partial f}{\partial u}u' + \frac{\partial f}{\partial t}\,.$$

'heck that the derived truncation error is consistent with previous results for ne case $f(u,t) = -au$. Filename: `trunc_nonlinear_ODE.pdf`.

:xercise 10: Truncation error of $[D_tD_tu]^n$

'erive the truncation error of the finite difference approximation (17)-(18) to ne second-order derivative. Filename: `trunc_d2u.pdf`.

:xercise 11: Investigate the impact of approximating $u'(0)$

ection 4.1 describes two ways of discretizing the initial conditon $u'(0) = V$ for vibration model $u'' + \omega^2 u = 0$: a centered difference $[D_{2t}u = V]^0$ or a forward ifference $[D_t^+ u = V]^0$. The program `vib_undamped.py`[5] solves $u'' + \omega^2 u = 0$ ith $[D_{2t}u = 0]^0$ and features a function `convergence_rates` for computing the rder of the error in the numerical solution. Modify this program such that it pplies the forward difference $[D_t^+ u = 0]^0$ and report how this simpler and more onvenient approximation impacts the overall convergence rate of the scheme. ilename: `trunc_vib_ic_fw.py`.

[5] `http://tinyurl.com/jvzzcfn/vib/vib_undamped.py`

Exercise 12: Investigate the accuracy of a simplified sc

Consider the ODE

$$mu'' + \beta|u'|u' + s(u) = F(t)\,.$$

The term $|u'|u'$ quickly gives rise to nonlinearities and complicates the Why not simply apply a backward difference to this term such that involves known values? That is, we propose to solve

$$[mD_tD_tu + \beta|D_t^- u|D_t^- u + s(u) = F]^n\,.$$

Drop the absolute value for simplicity and find the truncation error of the Perform numerical experiments with the scheme and compared with based on centered differences. Can you illustrate the accuracy loss visuall computations, or is the asymptotic analysis here mainly of theoretical i Filename: `trunc_vib_bw_damping.pdf`.

ndex