MTMW14

The University of Reading

Modelling the Atmosphere and Oceans

Lecture 2

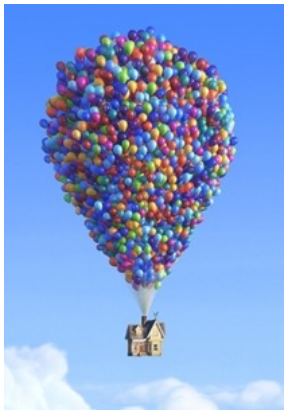
From 1-D advection to time schemes

PL Vidale

Largely based on Dave Randall's AT 604

Spring term 2023

Contents of MTMW14



What you should learn today

- Time integration of advection and diffusion equations:
  - even **simple equations can be integrated in a vast number of ways**. Which is the best numerical scheme?
  - The answer is: "None, it depends on the problem to be solved"

- This morning: some simple and advanced **time** schemes
- This afternoon:
  - A discussion on stability analysis for Practical 1: how to perform stability analysis of coupled equations

Your homework from this lesson:

- Revise simple stability analysis, either Hilary's notes, or else the complementary notes (MTMW14_notes_02.pdf):
  - Von Neumann
  - Energy method

Homework for the exam: learn how to show that Leapfrog is 2nd order accurate.

Reasons for developing time schemes

1. No matter the application, time is a crucial dimension, e.g. in:
   - Weather prediction: always going to the future, one step at a time
   - Climate simulation: simplistically, just an extension of weather prediction
   - Re-analysis: similar to a climate prediction, but we assimilate observations

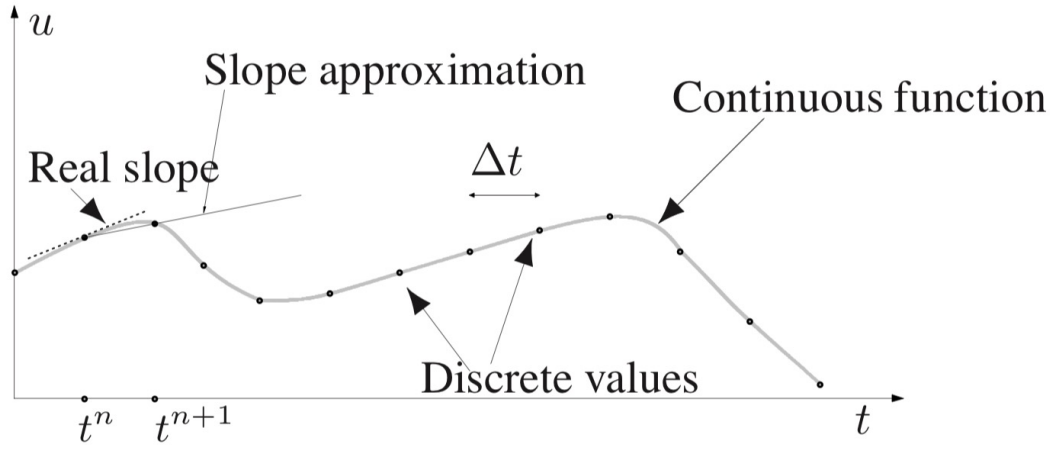


Figure 1-12 Representation of a function by a finite number of sampled values and approximation of a first derivative by a finite difference over Δt.

2. We want to be efficient, so we want to use as long a time step as possible, but there are limitations:
   - Accuracy
   - CFL (stability) criterion says that we cannot use arbitrarily long time steps: it depends on the fluid velocity and on the spatial resolution. Remember that **doubling spatial resolution means roughly a factor 8 in computational costs** (often even a factor of 10), and part of this is the time step.

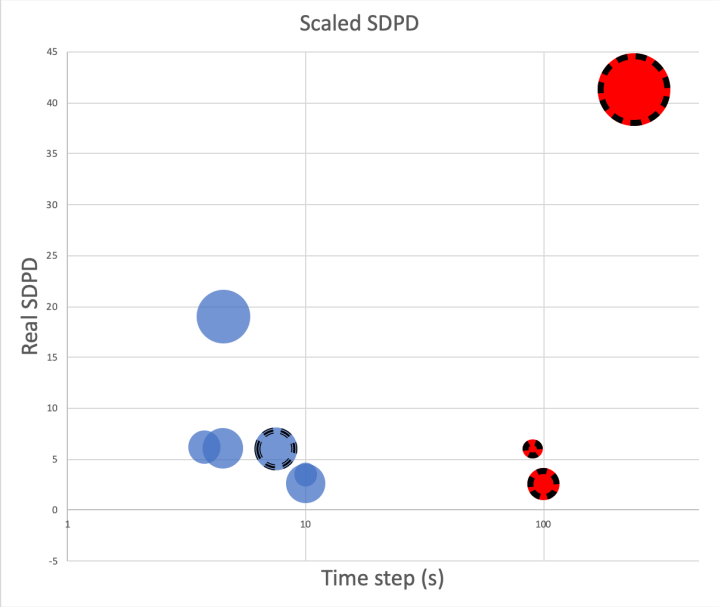
Two very important labels: 1) Semi-Lagrangian 2) Semi-Implicit

Comparison of DYAMOND GCMs

Table 5 Computational Aspects. To give a rough idea of the performance of the different models their throughput is given, and contextualized by some basic information about architectures on which they were run. Columns indicate the number of nodes and cores per node on which the simulations were run, simulated days per day, shortest (usually acoustic) timestep (Δt) and the type of processor.

	Model	Nodes	Cores	SDPD	Δt	Processor (date of launch into market)
SISL	ARPEGE-NH	300	7200	2.6	100 s	Intel Xeon "Ivy Bridge" (2013)
	FV3	384	13824	19	4.5 s	Intel Xeon "Broadwell" (2016)
	GEOS	512	20480	6.2	3.75 s	Intel Xeon "Skylake" (2017)
	ICON	540	12960	6.1	4.5 s	Intel Xeon "Haswell" (2014))
SISL	IFS	360	12960	124	240 s	Intel Xeon "Broadwell" (2016)
	MPAS	256	9216	3.5	10 s	Intel Xeon "Broadwell" (2016)
	NICAM	640	2560	2.6	10 s	NEX SX-ACE vector processor (2015)
	SAM	128	4608	6.0	7.5 s	Intel Xeon "Broadwell" (2016)
SISL	UM	340	12240	6.0	90.0 s	Intel Xeon "Broadwell" (2016)

Table 5 in Stevens et al., 2019



The three SISL models in DYAMOND use very long time steps, and we expect this to show in the SDPD statistics, once rescaled for problem size and number of cores.

There is a simple reason why SISL models can use long time steps, and the clue here is in the letter I, which stands for **implicit**. We can often achieve unconditional stability with implicit methods, but at the cost of damped solutions.

In the spatial domain, it is clear that, with one notable exception, the more homogeneous grids do provide a strong advantage.

So, for lat/lon models, there must be potential for some form of grid adaptation in value-poor regions: the Poles

What are we trying to do when we integrate in time?

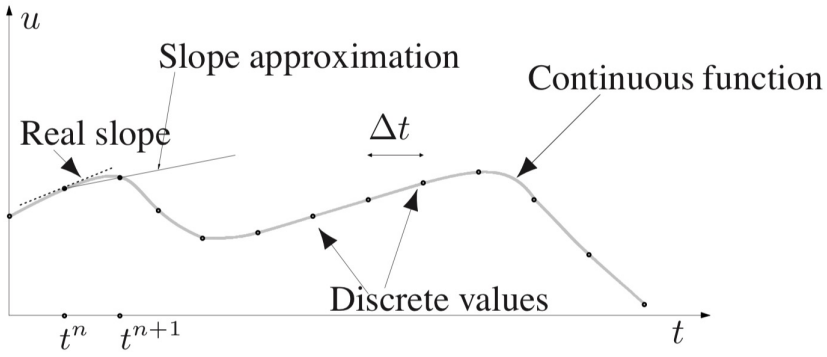


Figure 1-12 Representation of a function by a finite number of sampled values and approximation of a first derivative by a finite difference over Δt.

First, it is necessary to discretize the independent variable time t, since the first dynamical equations that we shall solve numerically are time-evolving equations. For simplicity, we shall suppose that the discrete time moments t^n, at which the function values are to be known, are uniformly distributed with a constant *time step* Δt

$$t^n = t^0 + n\,\Delta t, \quad n = 1, 2, ... \tag{1.9}$$

where the superscript index (not an exponent) n identifies the discrete time. Then, we note by u^n the value of u at time t^n, *i.e.*, $u^n = u(t^n)$.

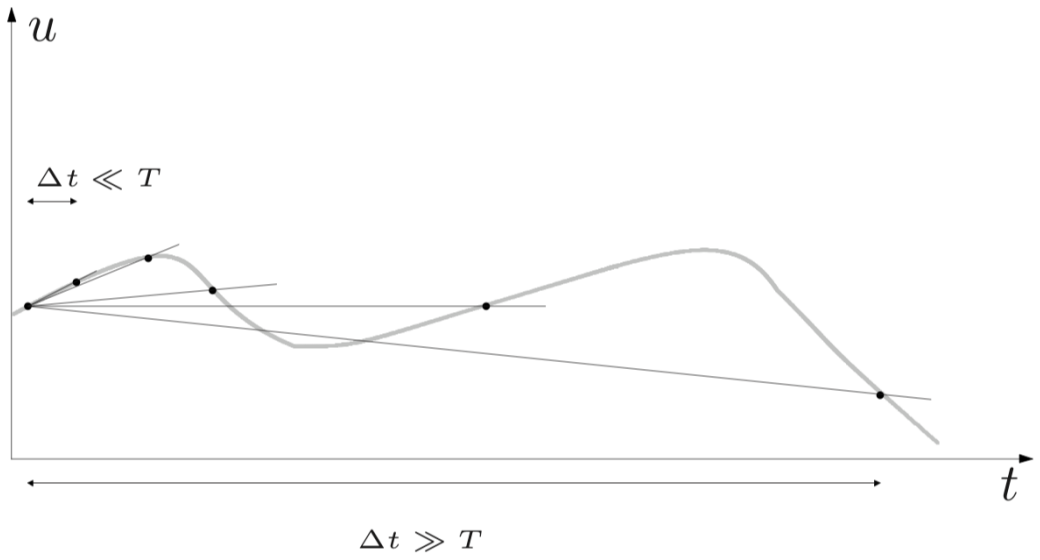


Figure 1-13 Finite differencing with various Δt values. Only when the time step is sufficiently short compared to the time scale, $\Delta t \ll T$, is the finite-difference slope close to the derivative, *i.e.*, the true slope.

Approximating a time derivative numerically

From the definition of a derivative

$$\frac{du}{dt} = \lim_{\Delta t \to 0} \frac{u(t+\Delta t) - u(t)}{\Delta t}, \tag{1.10}$$

we could directly deduce an approximation by allowing Δt to remain the finite time step

$$\frac{du}{dt} \simeq \frac{u(t+\Delta t) - u(t)}{\Delta t} \to \left.\frac{du}{dt}\right|_{t^n} \simeq \frac{u^{n+1} - u^n}{\Delta t}. \tag{1.11}$$

The accuracy of this approximation can be determined with the help of a Taylor series:

$$u(t+\Delta t) = u(t) + \Delta t \left.\frac{du}{dt}\right|_t + \underbrace{\frac{\Delta t^2}{2} \left.\frac{d^2u}{dt^2}\right|_t}_{\Delta t^2 \frac{U}{T^2}} + \underbrace{\frac{\Delta t^3}{6} \left.\frac{d^3u}{dt^3}\right|_t}_{\Delta t^3 \frac{U}{T^3}} + \underbrace{\mathcal{O}(\Delta t^4)}_{\Delta t^4 \frac{U}{T^4}}. \tag{1.12}$$

To the leading order for small Δt, we obtain the following estimate

$$\frac{du}{dt} = \frac{u(t+\Delta t) - u(t)}{\Delta t} + \mathcal{O}\left(\frac{\Delta t}{T}\frac{U}{T}\right). \tag{1.13}$$

The *relative* error on the derivative (the difference between the finite-difference approximation and the actual derivative, divided by the scale U/T) is therefore of the order $\Delta t/T$.

In-class discussion on explicit and implicit time integration

Time goes by ... so slowly

A small sampler of simple and advanced time schemes

Consider an arbitrary first-order ordinary differential equation of the form:

$$\frac{dq}{dt} = f\left[q(t),\ t\right]. \qquad (1)$$

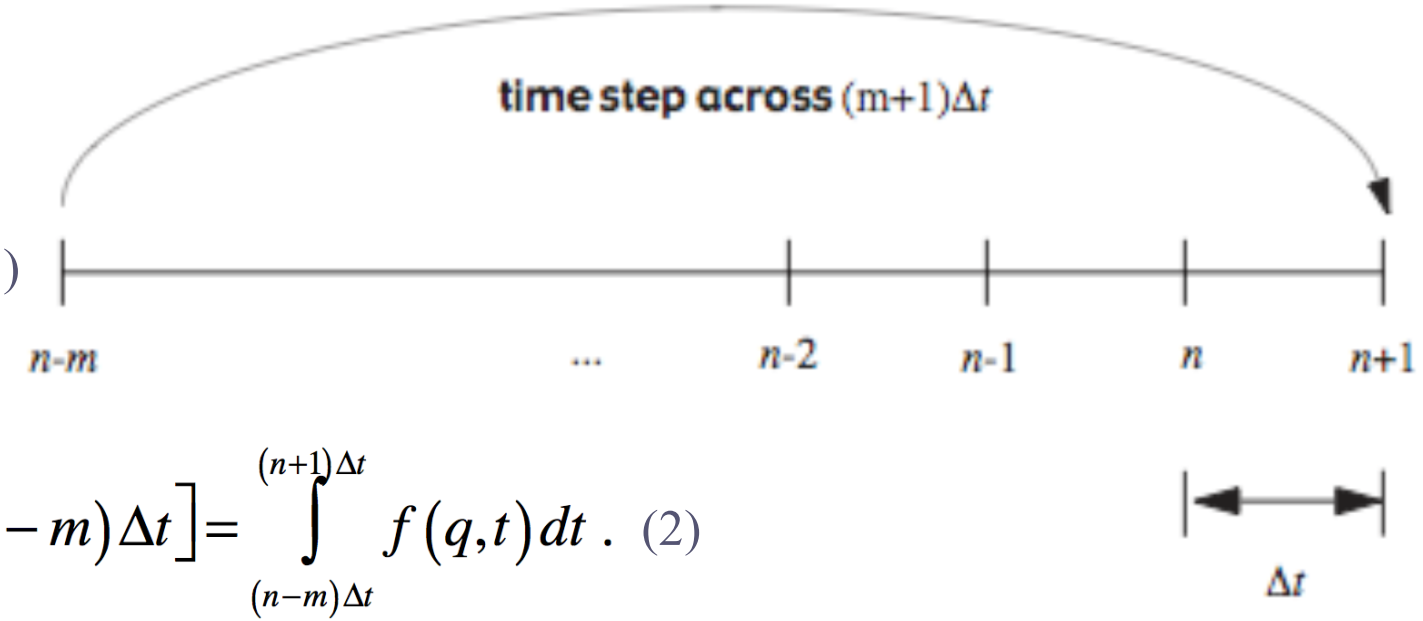
4.2 *Non-iterative schemes.*

Suppose that we integrate (1) with respect to time, from $(n-m)\Delta t$ to $(n+1)\Delta t$. Here we assume that m is either zero or a positive integer. We also assume that $n \geq m$, which may not be true close to the initial condition; this point is considered later. Then we have

$$q\left[(n+1)\Delta t\right] - q\left[(n-m)\Delta t\right] = \int_{(n-m)\Delta t}^{(n+1)\Delta t} f(q,t)\,dt\ .$$

(2)

$$\frac{dq}{dt} = f\left[q(t),\ t\right]. \quad (1)$$



$$q\left[(n+1)\Delta t\right] - q\left[(n-m)\Delta t\right] = \int_{(n-m)\Delta t}^{(n+1)\Delta t} f\left(q,t\right)dt\ . \quad (2)$$

Figure 4.1: In Eq. (4.3), we use a weighted combination of f^{n+1}, f^{n}, f^{n-1}, ... f^{n-l} to compute an average value of $f \equiv \frac{dq}{dt}$ over the time interval $(1+m)\Delta t$.

A family of schemes

β=0 explicit
β>0 implicit

Suppose that we approxim[ate] the right-hand side of (2) using the values of f at the discrete time levels. W[e] use the value of $q\left[(n+1)\Delta t\right]$, f^{n+1} in place of $f\left\{q\left[(n+1)\Delta t,\ (n+1)\Delta t\right\}\right.$, etc. Eq[uation] (2), can be approximated by

$$\frac{q^{n+1} - q^{n-m}}{(1+m)\Delta t} \cong \beta f^{n+1} + \alpha_n f^{n} + \alpha_{n-1} f^{n-1} + \alpha_{n-2} f^{n-2} + \ldots a_{n-l} f^{n-l}\ ,$$

(3)

Suppose that we approximate the integral on the right-hand side of (2) using the values of f at the discrete time levels. We use symbol q^{n+1} in place of $q\left[(n+1)\Delta t\right]$, f^{n+1} in place of $f\left\{q\left[(n+1)\Delta t,\ (n+1)\Delta t\right]\right\}$, etc. Eq. (2), divided by $(1+m)\Delta t$, can be approximated by

$$\frac{q^{n+1}-q^{n-m}}{(1+m)\Delta t} \cong \beta f^{n+1}+\alpha_n f^n+\alpha_{n-1}f^{n-1}+\alpha_{n-2}f^{n-2}+\ldots a_{n-l}f^{n-l},$$

Modification of an equation by Baer and Simons (1970) (3)

The LHS is a "time step" across a time interval of (1+ *m*)Δ*t*

The RHS consists of a weighted sum of instances of the function *f*, evaluated at various time levels, going back to time n-l, so we have:

- *m+1*= "time step" across a time interval of (1+*m*)Δ*t*
- *n+1* = the future
- *n* = the present
- *n-l* = the (remotest) past
- *l* = Time level *n* – *l* is furthest back in the past

- All combinations possible: l>m; l<m; l=m

Now substitute the *true solution,* $q(t)$, and the corresponding $f\left[q(t),t\right]$, into (3), and expand into a Taylor series around $t = n\Delta t$. We get

$$
\begin{aligned}
&\frac{1}{(1+m)\Delta t}\left\{\left[q+(\Delta t)q'+\frac{(\Delta t)^2}{2!}q''+\frac{(\Delta t)^3}{3!}q'''+\frac{(\Delta t)^4}{4!}q''''+\ldots\right]\right.\\
&\qquad\left.-\left[q-(m\Delta t)q'+\frac{(m\Delta t)^2}{2!}q''-\frac{(m\Delta t)^3}{3!}q'''+\frac{(m\Delta t)^4}{4!}q''''\ldots\right]\right\}\\
&\qquad=\beta\left[f+(\Delta t)f'+\frac{(\Delta t)^2}{2!}f''+\frac{(\Delta t)^3}{3!}f'''+\ldots\right]\\
&\qquad+\alpha_n f\\
&\qquad+\alpha_{n-1}\left[f-(\Delta t)f'+\frac{(\Delta t)^2}{2!}f''-\frac{(\Delta t)^3}{3!}f'''+\ldots\right]\\
&\qquad+\alpha_{n-2}\left[f-2(\Delta t)f'+\frac{(2\Delta t)^2}{2!}f''-\frac{(2\Delta t)^3}{3!}f'''+\ldots\right]\\
&\qquad+\alpha_{n-3}\left[f-3(\Delta t)f'+\frac{(3\Delta t)^2}{2!}f''-\frac{(3\Delta t)^3}{3!}f'''+\ldots\right]\\
&\qquad+\ldots\\
&\qquad+\alpha_{n-l}\left[f-l(\Delta t)f'+\frac{(l\Delta t)^2}{2!}f''-\frac{(l\Delta t)^3}{3!}f'''+\ldots\right]\\
&\qquad+\varepsilon,
\end{aligned}
\tag{4}
$$

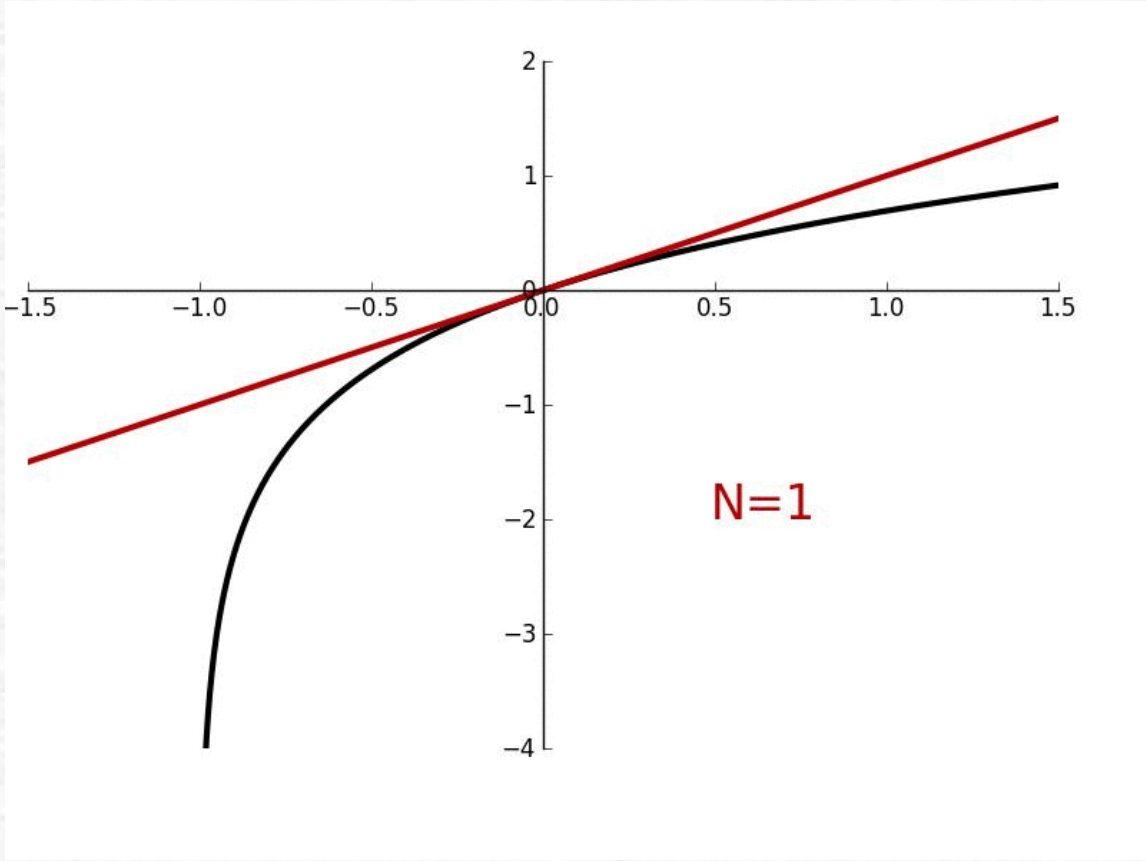
Where ε is the truncation error

Graphical depiction of Taylor series

Say that we are trying to predict the evolution of surface temperature from early morning to sunrise.

We could extrapolate linearly, or we could try to fit more and more terms in order to be more accurate.

However, as we increase the order of the polynomial, we need more and more points to compute the derivatives…



Multiply by (1+m)Δt, collect powers of Δt, use q′=f; q″=f′ …

$$
\begin{aligned}
& q'\left[1-\left(\beta+\alpha_n+\alpha_{n-1}+\alpha_{n-2}+\alpha_{n-3}+\ldots+\alpha_{n-l}\right)\right] \\
& +\Delta t q''\left\{\frac{1}{2}\left(\frac{1-m^2}{1+m}\right)-\beta+\alpha_{n-1}+2\alpha_{n-2}+2\alpha_{n-3}+\ldots+l\alpha_{n-l}\right\} \\
& +\frac{(\Delta t)^2}{2!}q'''\left\{\frac{1}{3}\left(\frac{1+m^3}{1+m}\right)-\beta-\alpha_{n-1}-4\alpha_{n-2}-9\alpha_{n-3}-\ldots-l^2\alpha_{n-l}\right\} \\
& +\frac{(\Delta t)^3}{3!}q''''\left\{\frac{1}{4}\left(\frac{1-m^4}{1+m}\right)-\beta+\alpha_{n-1}+8\alpha_{n-2}+27\alpha_{n-3}+\ldots+l^3\alpha_{n-1}\right\} \\
& +\ldots \\
& =\varepsilon \quad .
\end{aligned}
\tag{5}
$$

Each line on the left-hand side of (5) goes to zero "automatically" as $\Delta t \to 0$, except for the first line, which does not involve Δt at all. We have to force the first line to be zero, because otherwise the error, ε, will not go to zero as $\Delta t \to 0$. In order to force the first line to be zero, we have to choose β and the various α's in such a way that

$$
1=\beta+\alpha_n+\alpha_{n-1}+\alpha_{n-2}+\alpha_{n-3}+\ldots+\alpha_{n-l}\,. \tag{6}
$$

Eq. (6) simply means that the sum of the coefficients on the right-hand side of (3) is equal to one, so that the right-hand side is a kind of "average f." We refer to (6) as the "*consistency condition*." When (6) is satisfied, the expression for the truncation error reduces to

$$
\varepsilon=\Delta t q''\left\{\frac{1}{2}\left(\frac{1-m^2}{1+m}\right)-\beta+\alpha_{n-1}+2\alpha_{n-2}+3\alpha_{n-3}+\ldots+l\alpha_{n-l}\right\}+O\left[(\Delta t)^2\right]. \tag{7}
$$

Properties of our family of time schemes

1. If consistency condition satisfied, the scheme will be at least first order accurate (FOA)
2. We are free to choose:
   a. $l+1$ coefficients
   b. the value of l
3. If $l \geq 0$, we can force the scheme to be second order accurate, but we can go much further

In summary, order of accuracy can be at least $l+2$.
But remember that one of our coefficients is β!

-> An explicit scheme can have accuracy of at least $l+1$

Examples of common time schemes

Explicit (β=0): m=0, l=0, Forward (Euler)

For this simple case we have $\alpha_n \neq 0$, and all of the other α's are zero. The consistency condition, (6), immediately forces us to choose $\alpha_n = 1$. The scheme (3) then reduces to

$$\frac{q^{n+1} - q^n}{\Delta t} = f^n . \tag{8}$$

The truncation error is $\frac{\Delta t}{2} q'' + O\left(\Delta t^2\right) = O\left(\Delta t\right)$. Therefore, the scheme has first-order accuracy.

Explicit (β=0): m=0, l>0, Adams-Bashforth

Better accuracy can be obtained by proper choice of the α's, if we use $l > 0$. For example, consider the case $l = 1$. The scheme reduces to

$$\frac{q^{n+1} - q^n}{\Delta t} = \alpha_n f^n + \alpha_{n-1} f^{n-1}, \tag{9}$$

the consistency condition, (6), reduces to

$$\alpha_n + \alpha_{n-1} = 1, \tag{10}$$

and the truncation error is

$$\varepsilon = \Delta t q'' \left(\alpha_{n-1} + \frac{1}{2} \right) + O\left(\Delta t^2\right).$$

Explicit (β=0): m=0, l>0, Adams-Bashforth

- But what happens if we re-tune our *a*?
- Instead of: $a_n + a_{n-1} = 1$
- Use: $a_{n-1} = -1/2$
- Which of course means that a_n = ???

- Show that this results in 2nd order accurate

In a similar way, we can obtain Adams-Bashforth schemes with higher accuracy by using

l	α_n	α_{n-1}	α_{n-2}	α_{n-3}	truncation error
1	3 / 2	−1 / 2			$O(\Delta t^2)$
2	23 / 12	−4 / 3	5 / 12		$O(\Delta t^3)$
3	55 / 24	−59 / 24	37 / 24	−9 / 24	$O(\Delta t^4)$

Table 4.1. Adams-Bashforth Schemes $(\beta = m = 0,\ l > 0)$

More "all time favourites"

- Explicit (β=0): m=1, l=0, "Leapfrog"

The famous "leap-frog" scheme is given by

$$\frac{1}{2\Delta t}\left(q^{n+1}-q^{n-1}\right)=f^{n}. \tag{12}$$

From (5) we can immediately see that the truncation error is $\frac{\Delta t^{2}}{6}q'''+O\left(\Delta t^{4}\right)$.

So, the accuracy is larger than l+1=1 !!!
But there is a hidden cost…

- Explicit (β=0): m=1, ??l=1??: no gain!
- Explicit (β=0): m=1, l>1, Nystrom schemes

How do we prove that the Leapfrog scheme is higher order?

1.11 Higher-order methods

Rather than to increase resolution to better represent structures, we may wonder whether using other approximations for derivatives than our simple finite difference (1.11) would allow larger time-steps or higher quality approximations and improved model results. Based on a Taylor series

$$u^{n+1} = u^n + \Delta t \left.\frac{du}{dt}\right|_{t^n} + \frac{\Delta t^2}{2}\left.\frac{d^2u}{dt^2}\right|_{t^n} + \frac{\Delta t^3}{6}\left.\frac{d^3u}{dt^3}\right|_{t^n} + \mathcal{O}(\Delta t^4) \tag{1.19}$$

$$u^{n-1} = u^n - \Delta t \left.\frac{du}{dt}\right|_{t^n} + \frac{\Delta t^2}{2}\left.\frac{d^2u}{dt^2}\right|_{t^n} - \frac{\Delta t^3}{6}\left.\frac{d^3u}{dt^3}\right|_{t^n} + \mathcal{O}(\Delta t^4), \tag{1.20}$$

we can imagine that instead of using a *forward difference* approximation of the time derivative (1.11) we try a backward Taylor series (1.20) to design a *backward difference* approximation. This approximation is obviously still of first order because of its truncation error:

$$\left.\frac{du}{dt}\right|_{t^n} = \frac{u^n - u^{n-1}}{\Delta t} + \mathcal{O}(\Delta t). \tag{1.21}$$

Comparing (1.19) with (1.20), we observe that the truncation errors of the first-order forward and backward finite differences are the same but have opposite signs, so that by averaging both, we obtain a second-order truncation error (you can verify this statement by taking the difference between (1.19) and (1.20)):

$$\left.\frac{du}{dt}\right|_{t^n} = \frac{u^{n+1} - u^{n-1}}{2\Delta t} + \mathcal{O}(\Delta t^2). \tag{1.22}$$

More "all time favourites"

- **Implicit** (β≠ 0): m=0, l=0:
  - *backward (FOA)* and *trapezoidal (SOA)*

4.2.2 Implicit schemes

For implicit schemes, with $\beta \neq 0$, we can achieve accuracy at least as high as $l+2$. We consider several special cases:

$m=0,\ l=0$

Eq. (3) reduces to

$$\frac{q^{n+1}-q^n}{\Delta t} = \beta f^{n+1} + \alpha_n f^n . \tag{13}$$

In this case, the consistency condition reduces to $\alpha_n + \beta = 1$. The truncation error is $\Delta t q''\left(\frac{1}{2}-\beta\right)+O\left(\Delta t^2\right)$. For the special case $\beta = 1,\ \alpha_n = 0$, the scheme is called the backward (implicit) scheme. It has first-order accuracy. It can be said to correspond to $l=-1$. Higher accuracy is obtained for $\beta = \alpha = \frac{1}{2}$, which gives the "trapezoidal" (implicit) scheme. It has second-order accuracy, as we expect from the general rule for implicit schemes.

More "all time favourites"

- Implicit (β≠ 0): m=0, l>1: Adams-Moulton

$m=0,\ l>0$ *(Adams-Moulton schemes)*

These are analogous to the Adams-Bashforth schemes, except that $\beta \neq 0$. Table 4.2

l	β	α_n	α_{n-1}	α_{n-2}	α_{n-3}	truncation error
1	5 / 12	8 / 12	−1 / 12			$O(\Delta t^3)$
2	9 / 24	19 / 24	−5 / 24	1 / 24		$O(\Delta t^4)$
3	251 / 720	646 / 720	−264 / 720	106 / 720	−19 / 720	$O(\Delta t^5)$

Note that if l=0 (not in table), the maximum accuracy (SOA) is obtained for β=0, which is the ?????? scheme.

More "all time favourites"

- Implicit ($\beta \neq 0$): m=1, l=1: Milne corrector

Eq. (3) reduces to

$$\frac{q^{n+1} - q^{n-1}}{2\Delta t} = \beta f^{n+1} + \alpha_n f^n + \alpha_{n-1} f^{n-1}, \tag{14}$$

where

$$\beta + \alpha_n + \alpha_{n-1} = 1\,. \tag{15}$$

The truncation error is

$$\varepsilon = \Delta t q''\left(-\beta + \alpha_{n-1}\right) + \frac{\Delta t^2}{2!} q'''\left(\frac{1}{3} - \beta - \alpha_{n-1}\right) + \frac{\Delta t^3}{3!} q''''\left(-\beta + \alpha_{n-1}\right) + O\left(\Delta t^4\right).$$

The choices $\beta = \frac{1}{6},\ \alpha_n = \frac{4}{6},\ \alpha_{n-1} = \frac{1}{6},$ give fourth-order accuracy. This is again more than would be expected from the general rule.

[1] If there is a "Milne corrector," then there must be "Milne predictor." (See subsection 4.3 for an explanation of this terminology.) In fact, the Milne predictor is an explicit scheme with

$$m = 3,\ l = 3,\ \alpha_n = \frac{2}{3},\ \alpha_{n-1} = -\frac{1}{3},\ \alpha_{n-2} = \frac{2}{3},\ \alpha_{n-3} = 0\,.$$

So, there are many possible compromises so far

- Use short time steps at low order: high CPU cost, low memory cost, but low accuracy
- Use longer time steps at high order: save some CPU on the number of steps, but increase CPU costs due to a larger number of points in time, but also use more memory, which is a very significant bottleneck
- Use very long time steps with implicit methods, often at lower order, and achieve higher comparative accuracy/stability, but this ends up causing damping of the solutions
- Can we find other compromises, maybe suitable for specific problems and/or architectures?

Iterative time schemes, or "predictor-corrector" schemes

- Unlike the previous schemes, we arrive at a solution for q^{n+1} through an iterative, multi-step procedure. For each iteration, we must evaluate the function f.

$$\frac{q^{n+1}-q^n}{\Delta t}=\beta f^{n+1}+\alpha_n f^n . \tag{13}$$

Consider (13) as an example. Replace $f^{n+1}\equiv f\left[q^{n+1},(n+1)\Delta t\right]$ by $f^{n+1*}\equiv f\left[q^{n+1*},(n+1)\Delta t\right]$, where q^{n+1*} is obtained by the Euler scheme,

$$\frac{q^{n+1*}-q^n}{\Delta t}=f^n . \tag{16}$$

Then

$$\frac{q^{n+1}-q^n}{\Delta t}=\beta^* f^{n+1*}+\alpha f^n . \tag{17}$$

When $\beta^*=1,\ \alpha=0$, Eq. (17) is an imitation of the backward (implicit) difference scheme, and is called the Euler-backward scheme or the Matsuno scheme (Matsuno, 1966). When

Iterative time schemes, or "predictor-corrector" schemes

- The Heun scheme (also called modified Euler) is interesting.
- Pick $\beta^*=1/2$; $\alpha=1/2$
  - Imitation of the (implicit) trapezoidal scheme
- Matsuno is FOA; Heun is SOA
- Heun does not fit in the general framework of Eqn. (3), because it features f^{n+1*}, a term that cannot be written as a linear combination f's that we possess
- Heun is an explicit scheme and does not require past history ($l>0$ not needed)
- Despite all this it is SOA, because of iteration
- **ITERATION CAN INCREASE THE ORDER OF ACCURACY**

$$\frac{q^{n+1}-q^{n-m}}{(1+m)\Delta t} \cong \beta f^{n+1} + \alpha_n f^n + \alpha_{n-1} f^{n-1} + \alpha_{n-2} f^{n-2} + \ldots a_{n-l} f^{n-l}\,, \qquad (3)$$

Iterative time schemes, or "predictor-corrector" schemes
The Fourth-Order-Accurate Runge-Kutta

A famous example of an iterative scheme is the fourth-order Runge-Kutta scheme, which is given by:

$$q^{n+1} = q^n + \Delta t \frac{1}{6}\left(k_1 + 2k_2 + 2k_3 + k_4\right),$$

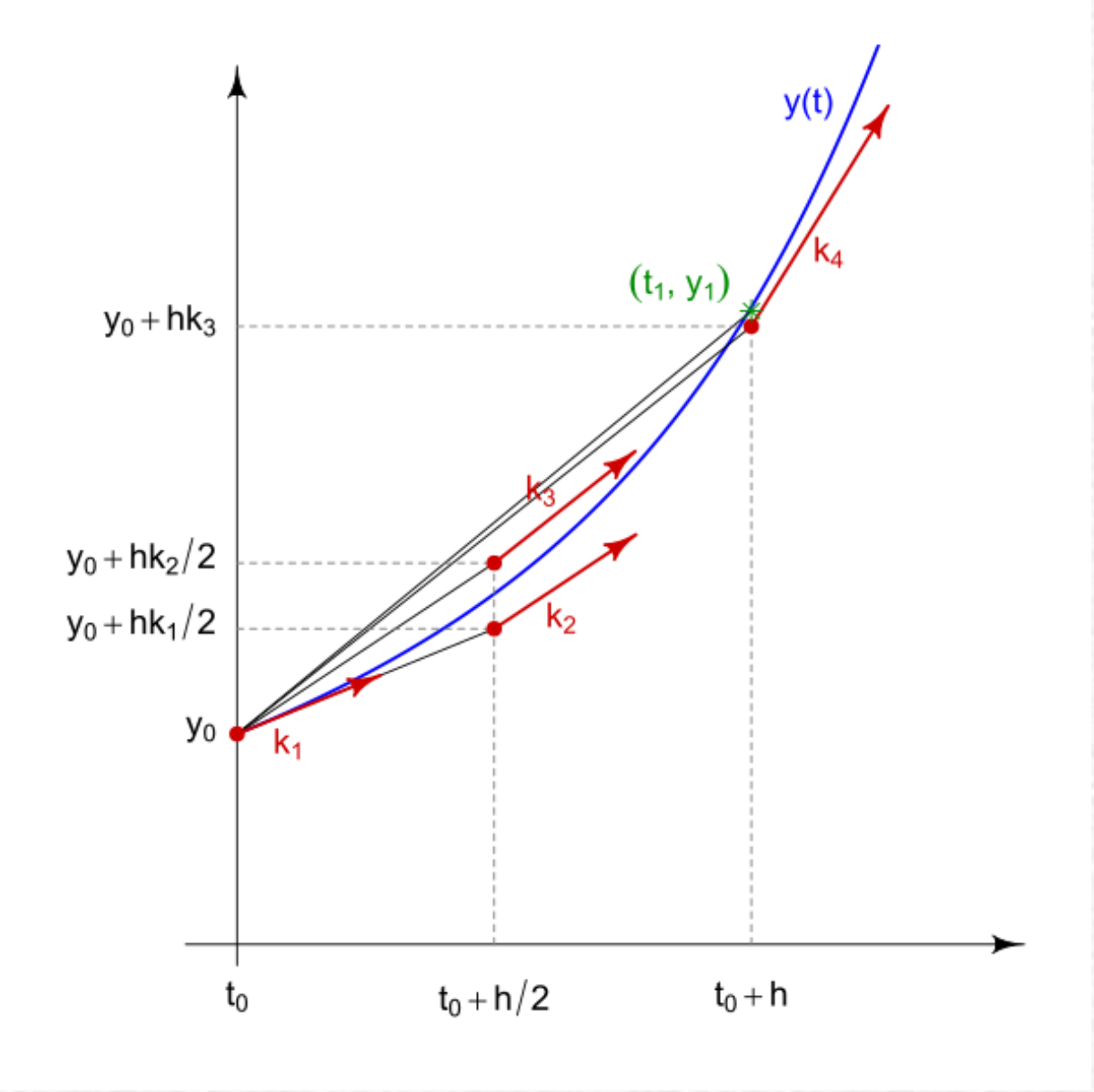
$$k_1 = f\left(q^n, n\Delta t\right), \quad k_2 = f\left[q^n + \frac{k_1 \Delta t}{2}, \left(n + \frac{1}{2}\right)\Delta t\right],$$

$$k_3 = f\left[q^n + \frac{k_2 \Delta t}{2}, \left(n + \frac{1}{2}\right)\Delta t\right], \quad k_4 = f\left[q^n + k_3 \Delta t, (n+1)\Delta t\right]$$

(18)

Each of the k 's can be interpreted as an approximation to f . The k 's have to be evaluated successively, which means that the function f has to be evaluated four times to take one time step. None of these f 's can be "re-used" on the next time step. For this reason, the scheme is not very practical unless a long time step can be used. Fortunately, long time steps are often possible.

The Runge-Kutta 4th order iterative time scheme



Finite difference schemes applied to the "oscillation equation"

$\frac{dq}{dt} = i\omega q$, where q is complex and ω is real.

This equation has the exact solution: $q(t) = \hat{q}e^{i\omega t}$

Remember that: $e^{i\omega t} = \cos(\omega t) + i\sin(\omega t)$

$\hat{q}$ is the initial value of q at $t = 0$

The amplitude $|\hat{q}|$ is an invariant of the system: $\frac{d}{dt}|\hat{q}| = 0$

Two notable examples:

Advection

$$\frac{\partial u}{\partial t} + c\frac{\partial u}{\partial x} = 0$$

If $u(t) = \hat{u}(t)e^{ikx}$, then $\frac{d\hat{u}}{dt} = -ick\hat{u} = i\omega\hat{u}$, where $\omega = -kc$

An observer at a point will see a single Fourier mode advect by, that is, an oscillation.

and

Rotation

$$\frac{du}{dt} - fv = 0$$

$$\frac{dv}{dt} + fu = 0$$

Multiply the second by i and add:

$$\frac{d}{dt}(u + iv) + f(-v + iu) = 0$$

If $q \equiv u + iv \quad \rightarrow \quad \frac{dq}{dt} + ifq = 0$

Numerical properties of our schemes as applied to those two problems

$\frac{dq}{dt} = i\omega q$, where q is complex and ω is real.

$|\hat{q}|$ is invariant: $\frac{d}{dt}|\hat{q}| = 0$

The amplitude: $|\lambda| = 1$ invariant

Consider:

$\lambda = \lambda_r + i\lambda_i$ real and imaginary

or $\lambda = |\lambda| e^{i\theta}$ where $\theta = \tan^{-1}\left(\frac{\lambda_i}{\lambda_r}\right)$

so that $\lambda_r = |\lambda|\cos\theta,\ \lambda_i = |\lambda|\sin\theta$

θ is the phase change in a time step.

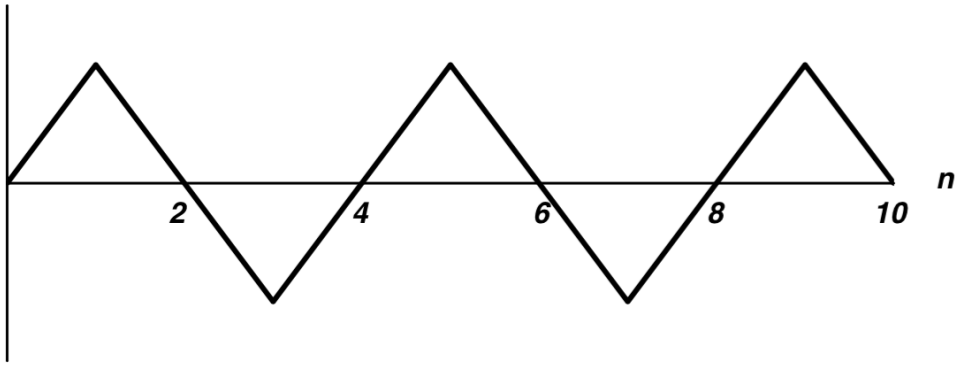


Figure 4.3: Schematic illustration of the solution of the oscillation equation for the case in which λ pure imaginary and the phase changes by $\theta = \frac{\pi}{2}$ on each time step.

If θ changes by π/2 each time step, then the solution will oscillate in time as shown in fig 4.3, taking four time steps to complete the cycle

Numerical properties of our schemes as applied to those two problems

- Two errors will be considered:
  - 1 an amplitude error
    - By definition the true solution has constant amplitude
  - 2 a phase error

For the true solution:

$\Omega \equiv \omega \Delta t$ $\quad q \sim e^{i\omega t}$

Amplification factor

$$q\left[\left(n+1\right)\Delta t\right] = e^{i\omega \Delta t} q\left(n\Delta t\right) = e^{i\Omega} q\left(n\Delta t\right)$$

Because amplitude should not change:

True lambda: $\lambda_T = e^{i\Omega}$, but $\left|e^{i\Omega}\right| = 1$

So, for a perfect numerical scheme:

$\left|\lambda\right| = 1$ and $\theta = \Omega$

What happens with 3 common schemes

$$\frac{dq}{dt} = i\omega q \Rightarrow q^{n+1} - q^n = i\omega\Delta t\left(\alpha q^n + \beta q^{n+1}\right) \Rightarrow$$

$$(1 - i\Omega\beta)q^{n+1} = (1 + i\Omega\alpha)q^n ,$$

$$\boxed{q^{n+1} = \left(\frac{1 + i\Omega\alpha}{1 - i\Omega\beta}\right)q^n}$$

$$\equiv \lambda q^n .$$

$$\lambda = \frac{1}{1 - i\Omega} = \frac{1 + i\Omega}{1 + \Omega^2},$$

Euler: α=1,β=0

$$\lambda = 1 + i\Omega ,$$

$$|\lambda| = \sqrt{1 + \Omega^2} > 1 .$$

Unconditionally unstable

Backward: α=0,β=1

$$|\lambda| = \frac{\sqrt{1 + \Omega^2}}{1 + \Omega^2} = \frac{1}{\sqrt{1 + \Omega^2}} < 1 .$$

Unconditionally stable
But decaying...

Trapezoidal: α=1/2,β=1/2

$$|\lambda| = \frac{\left|1 + \frac{i\Omega}{2}\right|}{\left|1 - \frac{i\Omega}{2}\right|} = 1 .$$

Unconditionally stable
Perfect!

And now with iterative schemes

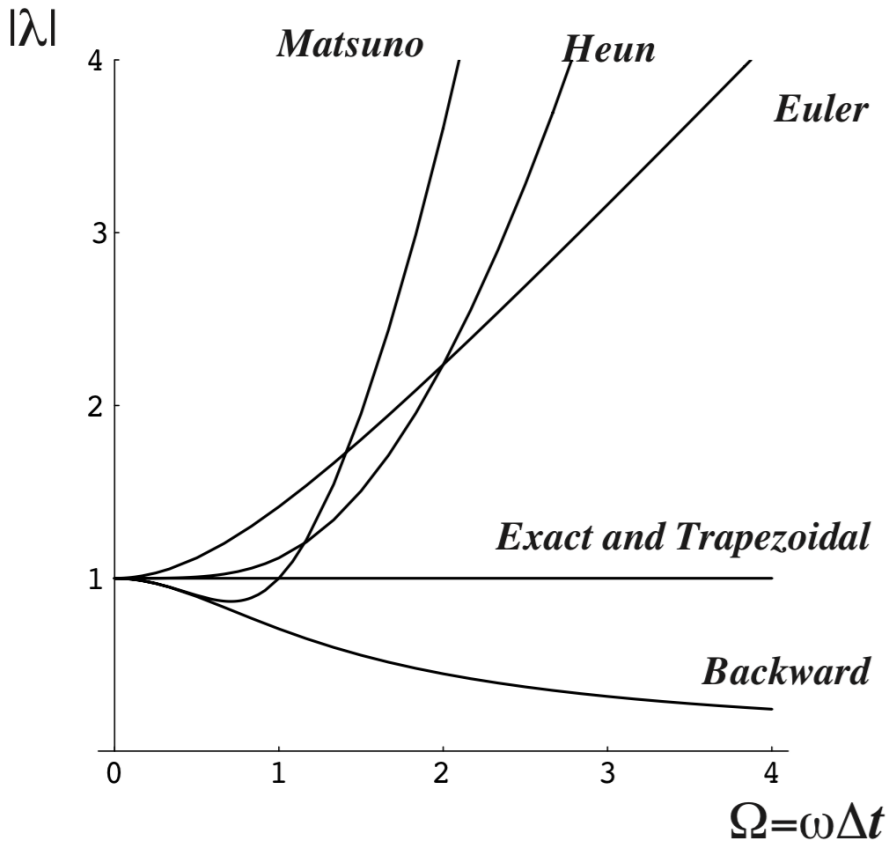


Figure 4.4: The magnitude of the amplification factor λ as a function of $\Omega \equiv \omega \Delta t$, for various difference schemes. The Euler, backward, trapezoidal, Matsuno, and Heun schemes are shown. The magnitude of the amplification factor for the trapezoidal scheme coincides with that of the true solution for all values of Ω. Caution: This does not mean that the trapezoidal scheme gives the exact solution!

Reminder: aliasing

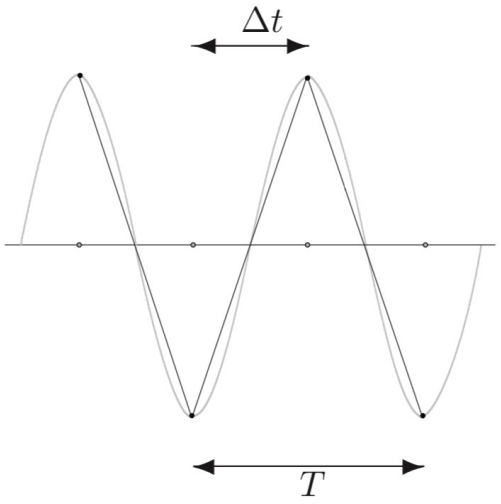


Figure 1-16 Shortest wave (at *cut-off frequency* $\pi/\Delta t$ or period $2\Delta t$) resolved by uniform grid in time.

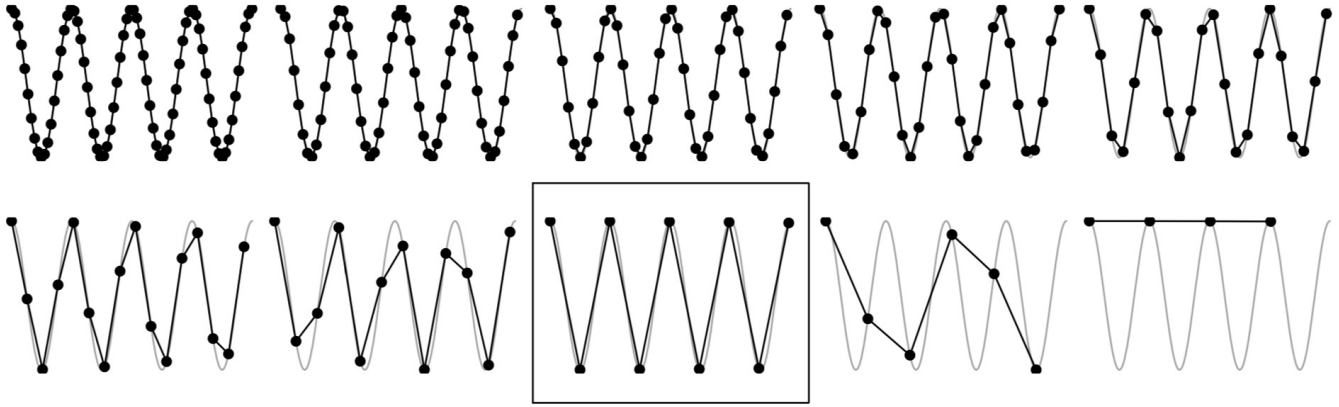


Figure 1-17 Aliasing illustrated by sampling a given signal (gray sinusoidal curve) with an increasing time interval. A high sampling rate (top row of images) resolves the signal properly. The boxed image on the bottom row corresponds to the cut-off frequency, and the sampled signal appears as a seesaw. The last two images correspond to excessively long time intervals that alias the signal, making it appear as if it had a longer period than it actually has.

Remember how great Leapfrog (SOA) seemed to be?

$$q^{n+1} - q^{n-1} = 2i\Omega q^n .$$

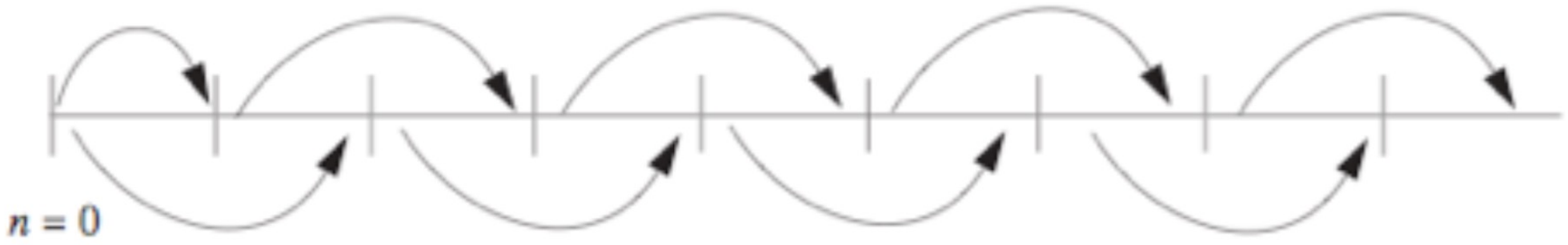


Figure 4.6: The leapfrog scheme.

We do not know the value of q^1 at time *0*
We must find some way to start up the system.
This means that we need TWO initial conditions to solve the finite-difference problem, when in fact the real problem only requires ONE initial condition.

Most schemes with more than two time levels present with this problem.

A computational mode in time

Say that:

1) $\omega=0 \rightarrow q^{n+1} - q^{n-1} = 0$, so a constant q is the true solution

2) we pick two Initial Conditions (ICs) that are not the same

This is what happens: $q^2=q^0$; $q^3=q^1$; $q^4=q^2=q^0$

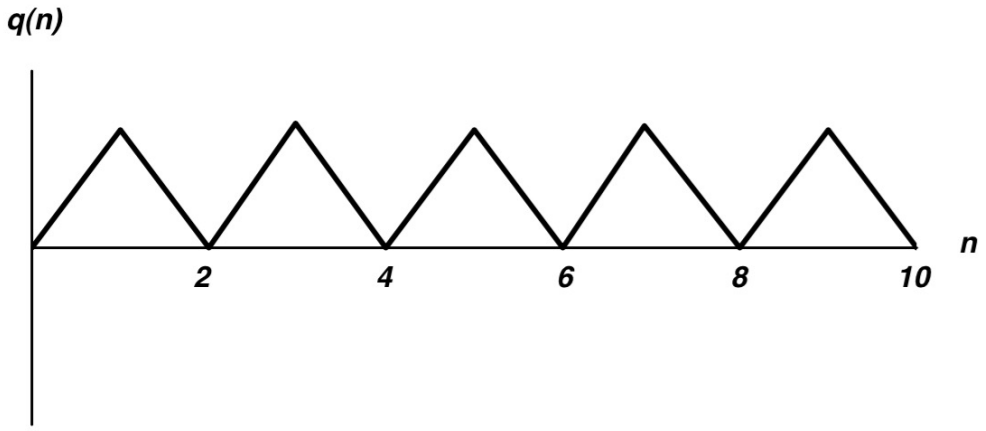


Figure 4.7: An oscillatory solution that arises with the leapfrog scheme for $\frac{dq}{dt} = 0$, in case the two initial values of q are not the same.

We must be very careful in how we pick ICs with 2-time level schemes

What about the Leapfrog scheme itself?

$$q^{n+1} - q^{n-1} = 2i\Omega q^n . \quad \Longrightarrow \quad q^{n+1} - 2i\Omega q^n - q^{n-1} = 0 ,$$

We seek a solution of the type $q^{n+1}=\lambda q^n$

$$\lambda^2 - 2i\Omega\lambda - 1 = 0 .$$

The solutions of (43) are

$$\lambda_1 = i\Omega + \sqrt{1-\Omega^2}, \quad \lambda_2 = i\Omega - \sqrt{1-\Omega^2}$$

i.e., there are two "modes", satisfying

$$q_1^{n+1} = \lambda_1 q_1^n , \; q_2^{n+1} = \lambda_2 q_2^n .$$

Very many bad things are happening:

- The analytical equations only have a single solution; we have two solutions, one of which is purely computational
- As $\Delta t \to 0$ or $\Omega \to 0$, $\lambda_1 \to 1$ but $\lambda_2 \to -1$
- As $\Delta t \to 0$ q_2^{n+1} will never approach q_1^n

What can we do?

1. Very carefully choose the ICs for the computational mode, e.g. initialise it so that it will remain = 0
   - Can this work? Why?
2. Use a time filter to control the computational mode

A second approach is to use a time filter, as suggested by Robert (1966) and Asselin (1972). We write

$$\overline{q^n} = q^n + \alpha\left(\overline{q^{n-1}} - 2q^n + q^{n+1}\right) \tag{60}$$

Here the overbar denotes a filtered quantity, and α is a parameter that controls the strength of the filter. For $\alpha = 0$, the filter is "turned off." Models that employ the Asselin filter often use $\alpha = 0.5$.

3. Throw more resource at the problem: the second-order Adams Bashforth Scheme (m = 0 , l = 1) and its third-order cousin (Durran) have nice properties, e.g. they damp the computational mode.

What about other applications? The decay equation

$$\frac{dq}{dt} = -\kappa q, \ \kappa > 0 , \tag{73}$$

which is relevant to many physical parameterizations, including those of the boundary layer, radiative transfer, cloud physics, and convection. The exact solution is

$$q(t) = q(0)e^{-\kappa t} .$$

For the Euler (forward) scheme, the finite-difference analogue of (73) is

$$q^{n+1} - q^n = -Kq^n , \tag{75}$$

where $K \equiv \kappa \Delta t$. The solution is

$$q^{n+1} = (1-K)q^n$$

Backward=unconditionally unstable;
Trapezoidal=unconditionally stable
Matsuno= conditionally stable
Heun=conditionally stable
Adams-Bashforth=conditionally stable

Finally, the leapfrog scheme for the decay equation is

$$q^{n+1} - q^{n-1} = -2Kq^n ,$$

$$\lambda^2 + 2K\lambda - 1 = 0 .$$

Leapfrog=unstable+computational mode! ⇦ $\lambda_1 = -K + \sqrt{K^2+1}, \ \lambda_2 = -K - \sqrt{K^2+1} .$

Why you should never use Leapfrog for any damping component of a model
(radiation, PBL, cloud physics, convection)

$$q^0 = 0$$

$$q^1 > 0$$

$$q^2 = q^0 - 2Kq^1 = 0 - 2Kq^1 < 0$$

$$q^3 = q^1 - 2Kq^2 = q^1 - 2K\left(-2Kq^1\right) = q^1\left(1+4K^2\right) > q^1$$

$$q^4 = q^2 - 2Kq^3 = -2Kq^1\left[1+\left(1+4K^2\right)\right] = q^2\left[1+\left(1+4K^2\right)\right] < q^2$$

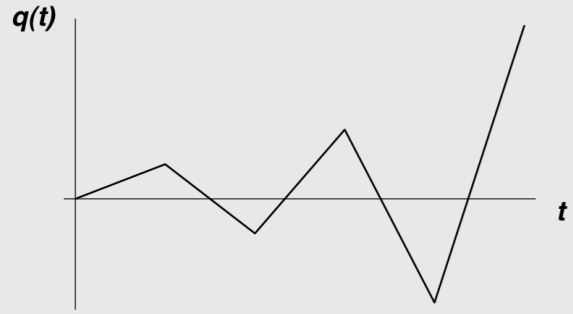


Figure 4.14: An illustration of how the leapfrog scheme leads to instability with the decay equation. The solution plotted represents the computational mode only and would be superimposed on the physical mode.

What about other applications? Damped oscillations

What should we do if we have an equation of the form

$$\frac{dq}{dt} = (i\omega - \kappa)q \ ? \tag{80}$$

One possibility is to mix the leapfrog and forward or backward schemes in the following manner. We write the finite difference analogue of as

$$q^{n+1} - q^{n-1} = 2i\Omega q^n - 2Kq^{n-1}, \tag{81}$$

(decay term forward differenced), or as

$$q^{n+1} - q^{n-1} = 2i\Omega q^n - 2Kq^{n+1}, \tag{82}$$

(decay term backward differenced). The oscillation terms on the right-hand sides of (81) and (82) are in "centered" form, whereas the damping terms have an uncentered form. These schemes are conditionally stable.

But often K will depend on q and (80) will become non-linear…

Summary

- Compromises must be made between simplicity, accuracy, cost, stability
- Once a good scheme is found for a given equation, **one should not be tempted to believe that it will be good for another equation**: analysis still needed.
  - Example: the leapfrog is cheap and second order accurate, but a really poor choice for solving the decay equation numerically
- Any scheme with more than two time levels will have a computational mode (we will see similarities in space in future lectures)