The Abdus Salam
International Centre for Theoretical Physics

United Nations
Educational, Scientific and
Cultural Organization

IAEA
International Atomic Energy Agency

2042-2

Co-sponsored School/Workshop on Integrable Systems and Scientific Computing

15 - 20 June 2009

Exponential integrators for stiff systems

Paul Matthews
University of Nottingham
U.K.

e-mail: Paul.Matthews@nottingham.ac.uk

Exponential integrators for stiff systems

Paul Matthews
School of Mathematical Sciences,
University of Nottingham, UK

Aim: Develop efficient time-stepping methods for PDEs.

- Lecture 1: Stiffness and numerical instability in ODEs and PDEs
- Lecture 2: Introduction to exponential integrators
- Lecture 3: Application to PDEs and more details

Lecture 1: Stiffness and numerical instability in ODEs and PDEs

There are many interesting nonlinear partial differential equations (PDEs).

These include dissipative equations such as the Kuramoto–Sivashinsky equation

$$u_t + uu_x = -u_{xx} - u_{xxxx},$$

the Swift–Hohenberg equation

$$u_t = ru - (1 + \nabla^2)^2 u - u^3,$$

and integrable PDEs such as the KdV equation

$$u_t + uu_x = u_{xxx}$$

and the nonlinear Schrödinger equation

$$u_t = iu_{xx} + i|u|^2 u.$$

Partial differential equations

In most cases, we cannot solve these PDEs analytically, so it is useful to have accurate and efficient numerical methods.

We will concentrate on the case of 1 space variable, although the methods generalise to 2 or 3 dimensions.

Similarly, we can easily generalise to coupled equations, for example reaction–diffusion equations, Navier–Stokes equations.

Definition:

A PDE is $semilinear$ if the term with the highest spatial derivative is linear.

Most of the nonlinear PDEs of interest are semilinear.

Spatial discretization of PDEs

To solve a PDE numerically we first discretize in space using finite differences (FD), finite elements or a spectral method.

This leads to a coupled system of ODEs in time for the value of the function at grid points (FD) or the spectral coefficients.
This is known as *semi-discretization*.

Spectral methods have a number of advantages:

- Spectral accuracy (exponential)
- Easy to code in Matlab
- Fast Fourier transform – almost as fast as FD methods
- Linear terms uncouple for Fourier spectral methods

Spatial discretization of PDEs

Fourier spectral methods can be used for PDEs with

- Periodic boundaries (full Fourier series)
- Dirichlet boundary conditions (Sine series)
- Neumann boundary conditions (Cosine series)

In the last two cases, we get full spectral accuracy for some PDEs, but only algebraic decay of coefficients in some cases.

Stiff systems

Suppose we semi-discretize the diffusion equation

$$u_t = u_{xx}, \qquad 0 < x < L, \qquad u(x+L,t) = u(x,t),$$

with a Fourier spectral method,

$$u(x,t) = \sum_{j=-N/2+1}^{N/2} a_j(t) \exp(ij2\pi x/L).$$

Then we get the uncoupled system of ODES,

$$\frac{da_j}{dt} = -\frac{4\pi^2 j^2}{L^2} a_j.$$

This set of ODEs is *stiff*.

Stiff systems

Definition:
A system of ODEs is said to be *stiff* if it involves a wide range of timescales.

Obviously not a precise mathematical definition!

Examples include systems of chemical reactions, where some reactions occur much faster than others.

A very simple example is

$$\begin{aligned} \frac{dy}{dt} &= -y \\ \frac{dz}{dt} &= -100z. \end{aligned}$$

Stiff systems

Here y evolves on a timescale of 'order' 1, z on a timescale of 'order' $1/100$. The stiffness can be measured by the $stiffness\ ratio$ of the timescales, 100 in this example.

Stiff systems are difficult to solve numerically:
we must resolve the short (fast) timescale,
but we are interested in the solution over the long timescale.
Usually this means that a large number of time-steps must be taken.

Stiffness in PDEs

For the system of ODEs arising from the diffusion equation,

$$\frac{da_j}{dt} = -\frac{4\pi^2 j^2}{L^2} a_j,$$

the stiffness ratio is of order N^2, where N is the number of Fourier modes.

This gets larger as the number of spatial derivatives increases, so the stiffness ratio for the KdV equation is $O(N^3)$, and $O(N^k)$ for a PDE with a kth x derivative.

Important point:
The stiffness is not caused by the spectral method.
It is an intrinsic property of the PDE.
A finite difference method or finite element method leads to a system of ODEs with a stiffness ratio of the same order.

Numerical methods for ODEs

The simplest numerical method for the ODE

$$\frac{dy}{dt} = f(y)$$

is Euler's method,

$$y_{n+1} = y_n + hf(y_n),$$

where h is the time-step and $y_n = y(nh)$.

Consider the simple linear ODE, $f(y) = cy$, where c is a negative constant, so the exact solution is exponential decay. With Euler's method, we have

$$y_{n+1} = y_n + hcy_n = (1 + hc)y_n.$$

Now if $|1 + hc| < 1$, then $y_n \to 0$ as $n \to \infty$, so the method gives decaying solutions (correct).

Numerical instability in Euler's method

But if $|1 + hc| > 1$ the numerical method gives growing solutions (incorrect). This is known as a $numerical\ instability$. To avoid numerical instability we need

$$-1 < 1 + hc < 1.$$

Since $c < 0$, the conditon to avoid numerical instability is

$$h < 2/(-c).$$

If $|c|$ is large, this means we must use a small time-step h.

(If $1/(-c) < h < 2/(-c)$ there are incorrect oscillations in y_n, but y_n still decays to 0.)

Explicit and implicit numerical methods

Euler's method is an $explicit$ method: y_{n+1} is a function of y_n.
More generally, a method is explicit if y_{n+1} is a function of y_m, $m \leq n$, so y_{n+1} depends on values of y that are already known.

In an $implicit$ method, y_{n+1} can be defined implicitly in terms of itself, for example the "Backward Euler method",

$$y_{n+1} = y_n + hf(y_{n+1}),$$

Applying this method to the case $f(y) = cy$, $c < 0$,

$$y_{n+1} = y_n + hcy_{n+1} = y_n/(1 - hc).$$

Now $1 - hc > 1$, $y_{n+1} < y_n$, and numerical instability does not occur.

General rule:
Implicit methods are less susceptible to numerical instability than explicit methods.

Higher order methods for ODEs

Euler's method and the backward Euler method are *first order*. The error **per step** is $O(h^2)$, so the error over a finite interval is $O(h)$.

Higher order methods can be constructed of *multi-step* type, using data from earlier steps, such as the 2nd order Adams–Bashforth method,

$$y_{n+1} = y_n + (3h/2)f(y_n) - (h/2)f(y_{n-1}).$$

Alternatively we can use *Runge–Kutta* methods using intermediate steps, for example the 2nd order method

$$y_m = y_n + (h/2)f(y_n), \qquad y_{n+1} = y_n + hf(y_m).$$

These are both explicit methods.

Numerical instability for higher order methods

Recall that Euler's method is unstable for $dy/dt = cy$, $c < 0$, unless $h < 2/(-c)$.

This restriction on h cannot be avoided by using a higher order method. All explicit methods have a stability condition $h < A/(-c)$ for some constant A.

Stability and accuracy are very different things.
In fact **higher** accuracy often leads to **worse** stability.

This is made more precise by the following theorem.

Dahlquist second stability barrier

Definition:
A method is $A\text{-}stable$ if the numerical solution y_n to $dy/dt = cy$, $Re(c) < 0$, tends to 0 as $n \to \infty$, for any step size h.

For example, the backward Euler method is A-stable.

Theorem:
• An explicit multi-step method cannot be A-stable.
• An implicit, A-stable multi-step method cannot be more than 2nd order accurate.

This suggests that implicit methods are better for stiff systems.

But implicit methods are more difficult to use, since we have to solve for y_{n+1}.

Summary of Lecture 1

- Space discetization of a PDE leads to a stiff system of ODEs.
- The different Fourier modes evolve on very different timescales.
- An explicit method requires a time step of order N^{-k}, where N is the number of Fourier modes/grid points, k is the number of x derivatives.
- Implicit methods are more stable, but require the solution of a large nonlinear system at each time step.

Lecture 2: Introduction to exponential integrators

Suppose we semi-discretize a nonlinear PDE with period boundary conditions on $[0, L]$ using a Fourier spectral method.
We get a large stiff system of ODEs for the Fourier coefficients:

$$\frac{da_j}{dt} = c_j a_j + F_j,$$

where F_j represents the nonlinear terms. The linear terms are uncoupled.
The constant c_j is related to the wavenumber

$$k_j = 2\pi j/L, \qquad j = -N/2 + 1 \ldots N/2.$$

For the (dissipative) KS equation, $c_j = k_j^2 - k_j^4$ so for large N, c_j is large, real and negative.
For the (dispersive) NLS equation, $c_j = -ik_j^2$, so c_j is large and imaginary.

Lecture 2: Introduction to exponential integrators

These PDEs are semi-linear, so the stiffness arises from the linear term. The nonlinear term is usually not stiff.

If we solve with a standard explicit method, we must use a very small timestep, $h = O(1/|c_{max}|)$, limited by the linear term.

But we can solve the linear term exactly, giving exponential decay or oscillations. This is the idea behind $exponential\ integrators$.

Definition (mine):
Consider the system

$$\frac{du}{dt} = c\,u + F(u,t).$$

An $exponential\ integrator$ is a numerical method that is exact when $F = 0$.

Derivation of exponential integrators

Consider the ODE

$$\frac{du}{dt} = c\,u + F(u,t). \tag{1}$$

Think of this as one of the ODEs arising from the semi-discretized PDE, with $F(u,t)$ being the nonlinear terms. Think of $|c| \gg 1$.

Multiply (1) through by integrating factor e^{-ct}:

$$d(e^{-ct}u)/dt = e^{-ct}F(u,t). \tag{2}$$

Now integrate (2) over one time step from $t = t_n$ to $t = t_{n+1} = t_n + h$:

$$u(t_{n+1}) = e^{ch}u(t_n) + e^{ch}\int_0^h e^{-c\tau}F(u(t_n+\tau), t_n+\tau)\,d\tau. \tag{3}$$

This is **exact**. Now approximate the integral in (3) in different ways to get different exponential integrators. There are two main types.

Derivation of exponential time differencing method

The simplest approximation is $F = \text{constant} = F(u(t_n), t_n) = F_n$ in (3). With this assumption we can do the integral and get

$$u_{n+1} = e^{ch} u_n + \frac{e^{ch} - 1}{c} F_n. \qquad \text{(ETD1)}$$

This method is called "exponential time differencing" (ETD).

Properties:

- First order, explicit.
- Exact if $F = \text{constant}$.
- Can also be written

$$u_{n+1} = u_n + \frac{e^{ch} - 1}{c}(cu_n + F_n).$$

  Hence the method gets fixed points of the ODE exactly correct.
- Error per step is $h^2 \dot{F}/2$, so accurate if F varies slowly.

Derivation of integrating factor method

Apply Euler's method directly to (2).

$$u_{n+1} = e^{ch}(u_n + hF_n). \tag{IF1}$$

Alternatively, approximate $e^{-c\tau}F$ in the integral (3) by its value at the beginning of the interval, F_n.

Properties:

- First order, explicit.
- Fast exponential part is handled exactly.
- Fixed points wrong, with relative error $O(ch)$.
- Error per step is $h^2(-cF + \dot{F})/2$, so inaccurate if F is slowly varying (would be exact if $F \propto \exp(ct)$).

Two types of exponential integrator

- Both get the linear part exactly right (exponential).
- Both can be extended to any order of accuracy.
- Both avoid the stability restriction $h < O(1/|c|)$.
- Can construct implicit versions – but usually no point.

Exponential time differencing (ETD)

- Exact if nonlinear term F is constant.
- Accurate if F is slowly varying.
- Derive by approximation F in (3) as a polynomial in t.

Integrating factor method (IF)

- Derive by applying any standard ODE solver to (2).
- Less accurate if F is slowly varying.

Higher order exponential integrators (multi-step)

If we approximate F as a linear function of τ,

$$F = F_n + \tau(F_n - F_{n-1})/h + O(h^2),$$

in (3), then do the integral, we get a second-order ETD method,

$$u_{n+1} = e^{ch} u_n + \frac{e^{ch} - 1}{c} F_n + \frac{e^{ch} - 1 - ch}{c^2 h}(F_n - F_{n-1}). \qquad \text{(ETD2)}$$

If we apply the second-order Adams–Bashforth method to (2), we get a second-order multi-step integrating factor method,

$$u_{n+1} = e^{ch} u_n + \frac{3h}{2} e^{ch} F_n - \frac{h}{2} e^{2ch} F_{n-1}. \qquad \text{(IF2)}$$

(*Exercise: derive these two formulas*).

Higher order exponential integrators (Runge–Kutta)

We can also derive Runge-Kutta-like exponential integrators.
For example analogous to the RK2 method in Lecture 1,

$$u_m = e^{ch/2} u_n + \frac{e^{ch/2} - 1}{c} F_n,$$

$$u_{n+1} = e^{ch} u_n + \frac{e^{ch} - 1}{c} F_n + 2\frac{e^{ch} - 1 - ch}{c^2 h}(F(u_m) - F_n). \quad \text{(ETDRK2)}$$

Higher order ETDRK methods are quite difficult to derive – need to satisfy a large number of "order conditions".

The ETDRK methods are generally better (more accurate and more stable) than the ETD multistep methods.

RK integrating factor methods are easy to derive, just by applying a standard RK method to (2).

Aside: linearly implicit method

Also known as 'IMEX' (Implicit-Explicit) or 'semi-implicit'.
See Ascher, Ruuth & Wetton (1995). Well known and widely used.

Use implicit method (e.g. backwards Euler) for linear term, explicit method (e.g. Euler) for nonlinear term:

$$u_{n+1} = \frac{(u_n + hF_n)}{(1 - ch)}. \qquad \text{(LI1)}$$

The implicit part is stable for $any\ h$, (method is A-stable).

Disadvantages:

- Cannot extend beyond second order without losing A-stability (Dahlquist second stability barrier).
- Fails to capture rapid exponential decay / rapid oscillations if $|c| \gg 1$.

But why solve the linear part implicitly when we can solve it exactly?

Exponential integrators – Literature and History

Textbooks

There is very little on exponential integrators in numerical analysis textbooks.

In particular the ETD methods are only mentioned briefly in
Numerical Methods for Stiff Equations and Singular Perturbation Problems, W L Miranker (1981)
and the new book
Numerical Methods for Evolutionary Differential Equations,
U M Ascher (2008).

The IF methods are described in two books on spectral methods:
Spectral Methods in Matlab, L N Trefethen (2000),
Chebyshev and Fourier Spectral Methods, J P Boyd (2001).
Both books point out the weakness of IF methods (inaccurate for slowly varying solutions) but do not mention ETD methods.

Exponential integrators – Literature and History

Review articles

There are no published review articles on exponential integrators.

There is one unpublished article:

A review of exponential integrators for first order semi-linear problems.
Borislav V. Minchev and Will M. Wright (2005)

Available from website of Trondheim research group,
http://www.math.ntnu.no/num/expint

From the same group, there is
EXPINT—A MATLAB package for exponential integrators.
H Berland, B Skaflestad and W Wright.
ACM Transactions on Mathematical Software (TOMS) (2007)

Exponential integrators – Literature and History

History

ETD has been re-invented very many times over the years. Each inventor gives the method a different name (ELP, EPI, QSS, PSS, slave-frog, method of patches, . . .)

ETD1 is used in computational electromagnetism
(papers by Holland, Schuster, book by Taflove, 1995).
The term 'Exponential time differencing' comes from this field.

• Certaine (1960) "The solution of ordinary differential equations with large time constants". Key ideas. How to develop multistep ETD methods of any order. Formulas for implicit forms of ETD2 and ETD3.

• Nørsett (1969). Formulas for multistep ETD methods of any order using generating function.

• Friedli (1978). (Auf Deutsch). Consistency conditions for ETDRK methods.

• Hochbruck Lubich & Selhofer (1998). Re-linearising finding Jacobian at each step. Krylov subspace method.

• Beylkin et al. (1998). Formulation, stability regions.

• Cox & Matthews (2002). Formulas for ETDRK methods of order up to 4. Explanation of superiority of ETD over LI and IF, ODE and PDE tests.

• Kassam & Trefethen (2005). Method for evaluating coefficients. Tests 5 types of 4th order method on 4 PDEs – ETDRK4 is best. Matlab codes.

• Hochbruck & Ostermann (2005) Convergence proofs, order conditions for ETDRK methods.

The first paper on the IF method is
• Lawson (1967) "Generalized Runge-Kutta processes for stable systems with large Lipschitz constants".

So we could refer to ETD methods as 'Certaine methods' and IF methods as 'Lawson methods'.

Simple ODE Example

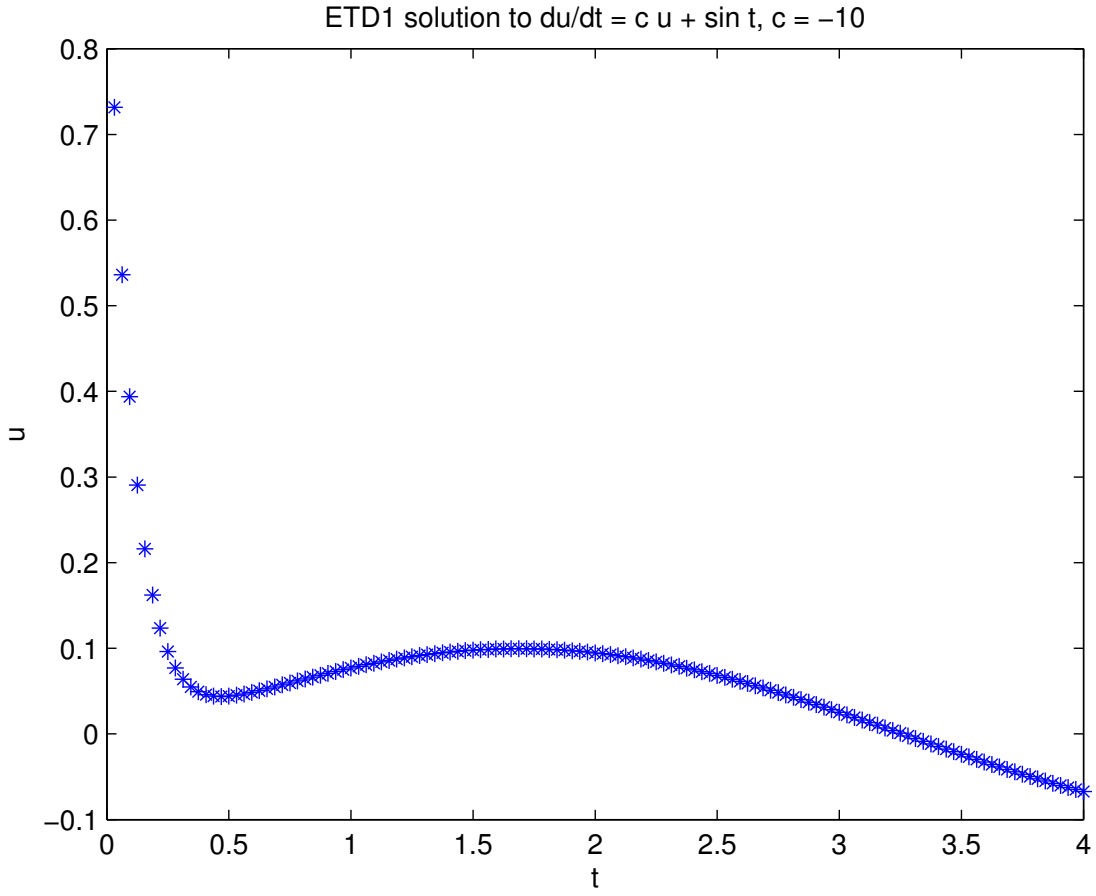
Consider the ODE

$$\frac{du}{dt} = cu + \sin t, \qquad u(0) = 1.$$

Simple Matlab code:

```
% solve du/dt = c u + sin t  with ETD1 method
clear
c=-10;nsteps=128;tmax=4;h=tmax/nsteps;
u=1;t=0;
for k=1:nsteps
   u = u*exp(c*h) + (exp(c*h)-1)*sin(t)/c;
   t = k*h;
   plot(t,u,'*');hold on
end;hold off
xlabel('t');ylabel('u')
title('ETD1 solution to du/dt = c u + sin t, c = -10')
```

Simple ODE Example



Matlab Exercises

1. Modify this code to plot the exact solution for comparison.

2. Include the IF1 and LI1 methods in the same code and compare.

3. Investigate effect of changing stiffness c and step size h.

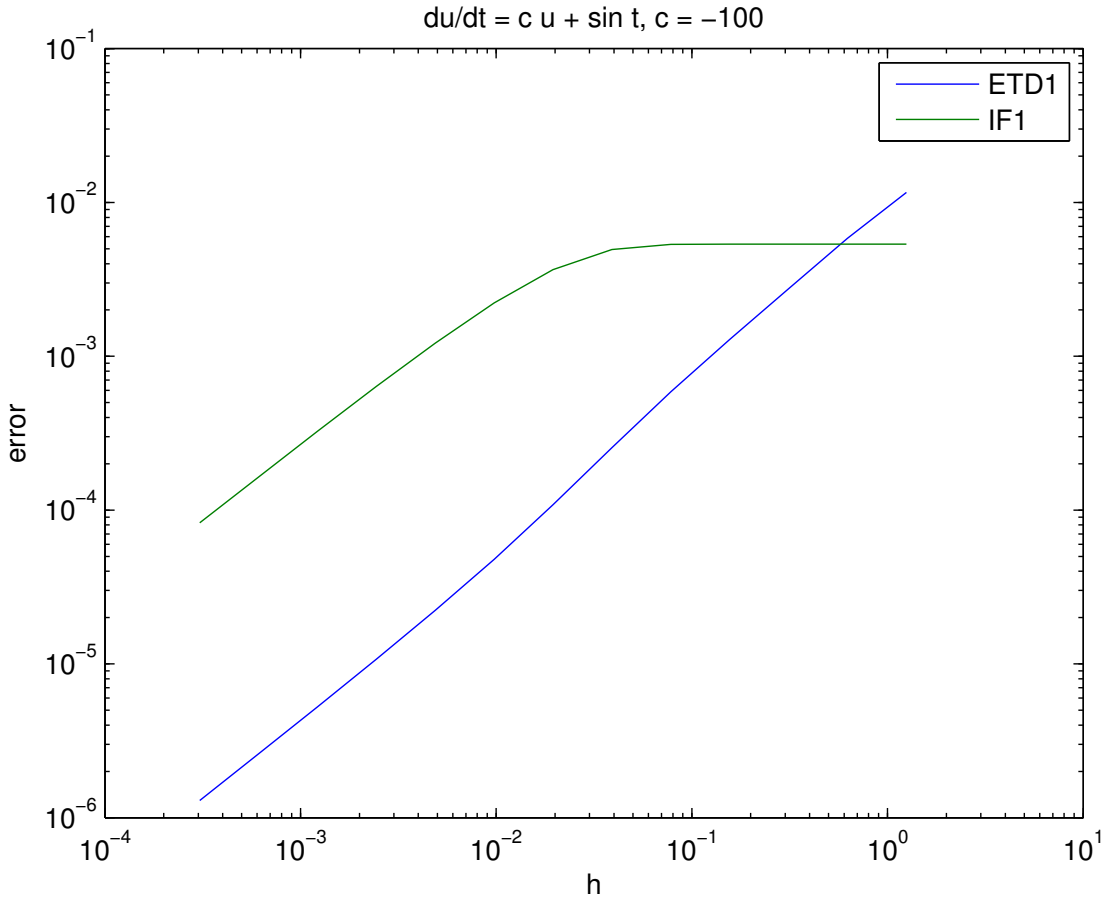
4. Plot the error as a function of h for each method.

5. Investigate what happens if c is imaginary.

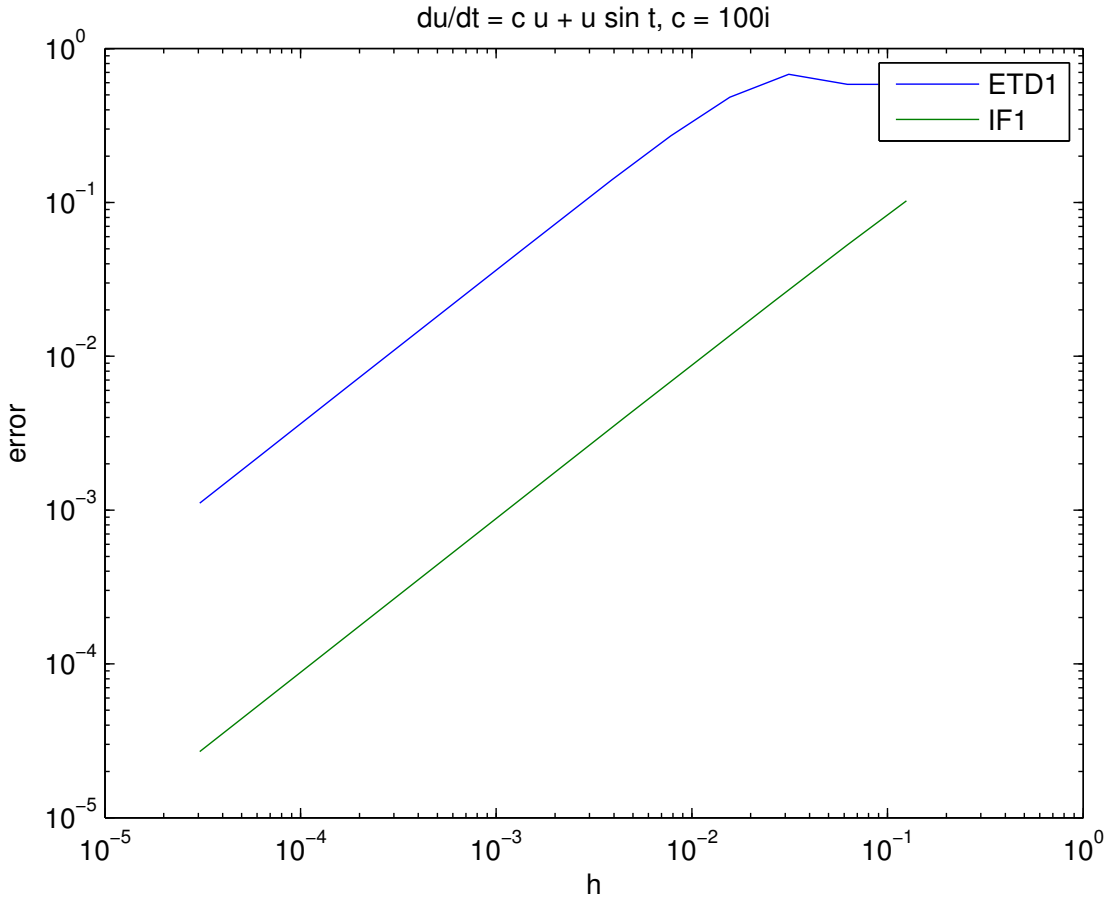
6. Change the ODE to $du/dt = cu + u \sin t$.

7. Extend the code to use a second-order method.

Comparison of ETD and IF



Comparison of ETD and IF



Summary of Lecture 2

- Exponential integrators are the best method for systems of ODEs with stiff linear terms.
- Linear terms are solved exactly. Step size not limited by stiffness.
- Two types: ETD (Certaine), IF (Lawson).
- ETD type better if nonlinear term does not vary rapidly.
- Higher order methods of multi-step or Runge–Kutta type.
- Not well represented in textbooks and literature.

Lecture 3: Application to PDEs and more details

It is very easy to write a code to solve a PDE with periodic boundary conditions, using a Fourier spectral method and an exponential integrator, in Matlab.

The time-stepping is done in Fourier space so we have a vector of ODEs for each Fourier coefficient. The stiff linear parts decouple.

The nonlinear term is evaluated in physical space using FFT.
This is "fast" (CPU time $= O(N \log N)$) so code is almost as fast as a finite difference method, and much more accurate.
This is sometimes called a "pseudo-spectral" method.

Example: nonlinear Schrödinger equation

The nonlinear Schrödinger equation

$$u_t = iu_{xx} + i|u|^2 u$$

is often used as an example (papers by Berland et al, C. Klein).

Features:
- Applications to nonlinear waves in optics, water waves.
- Integrable – infinitely many conserved quantities.
- Dispersive – speed of linear waves depends on wavelength.
- Several exact solutions known, e.g. solitons (good for testing).
- Many different regimes / types of behaviour.

Soliton solution is $u(x,t) = a\,\mathrm{sech}(b(x-vt))\exp(i(cx+\omega t))$
with $b = a/\sqrt{2}$, $c = -v/2$, $\omega = c^2 - b^2$, free parameters a and v.

Matlab code for NLS equation

```
% NLS equation,  udot = i(u'' + |u|^2 u)
% solve in Matlab with spectral method and ETD1 timestepping
N = 256; L=6*pi; a=2;                          % set no. of pts, length
x = (L/N)*(0:N-1);                             % set x at grid points
u=(a*exp(i*x)).*sech(a/sqrt(2)*(x-L/2)); % set initial condition
v = fft(u);                                    % v is Fourier transform of u
k = [0:N/2 -N/2+1:-1]*2*pi/L;                  % set wavenumbers
c = -i*k.^2;                                   % linear part
nmax=4000; tmax=10; h=tmax/nmax;               % Choose timestep etc
E = exp(h*c); etd =(E-1)./c; etd(1)=h;    % set coeffs for ETD1 method
for n = 1:nmax                                 % start the main loop
  u = ifft(v);                                 % go back to physical space
  nl = fft(i*u.*u.*conj(u));                   % and find nonlinear term
  v = E.*v + nl.*etd;                          % update with ETD1
  if mod(n,20) == 0;                           % plot it every few steps
     plot(x,real(u));axis([0,L,-2,2]);drawnow;
  end
end
```

Matlab Exercises

1. Modify this code to print the error (exact soliton known).
2. Modify this code to monitor conserved quantities e.g. mean($|u|^2$).
3. Include the IF1 method in the same code and compare.
4. Plot the error as a function of h for each method.
5. Investigate how error depends on soliton parameters.
6. Extend the code to second order and check.
7. Change the PDE to KdV/KS/Your favourite PDE.

Accuracy of methods for NLS equation

Hala Ashi, PhD thesis, Nottingham 2008 (available on-line).

For stationary or slowly moving solitons, ETD is more accurate than IF.
As speed v of soliton increases, ETD becomes less accurate, but IF methods are not affected, so IF methods are better for fast solitons.

This behaviour can be understood by looking at "plane wave" solutions $u(x,t) = a\exp(i\omega t + ikx)$, with $\omega = k^2 - a^2$.
Can show that error term for ETD method increases as k (hence ω) increases, but error for IF does not.

Important point: we cannot say
"The . . . method is the best method for the . . . equation".
Most PDEs have different parameter regimes giving different types of solution, each being best treated by a different numerical method.

ETD methods are better for slowly varying ("slow manifold") solutions.
IF methods are better for rapidly oscillating small amplitude solutions.

Example: KdV equation

KdV equation

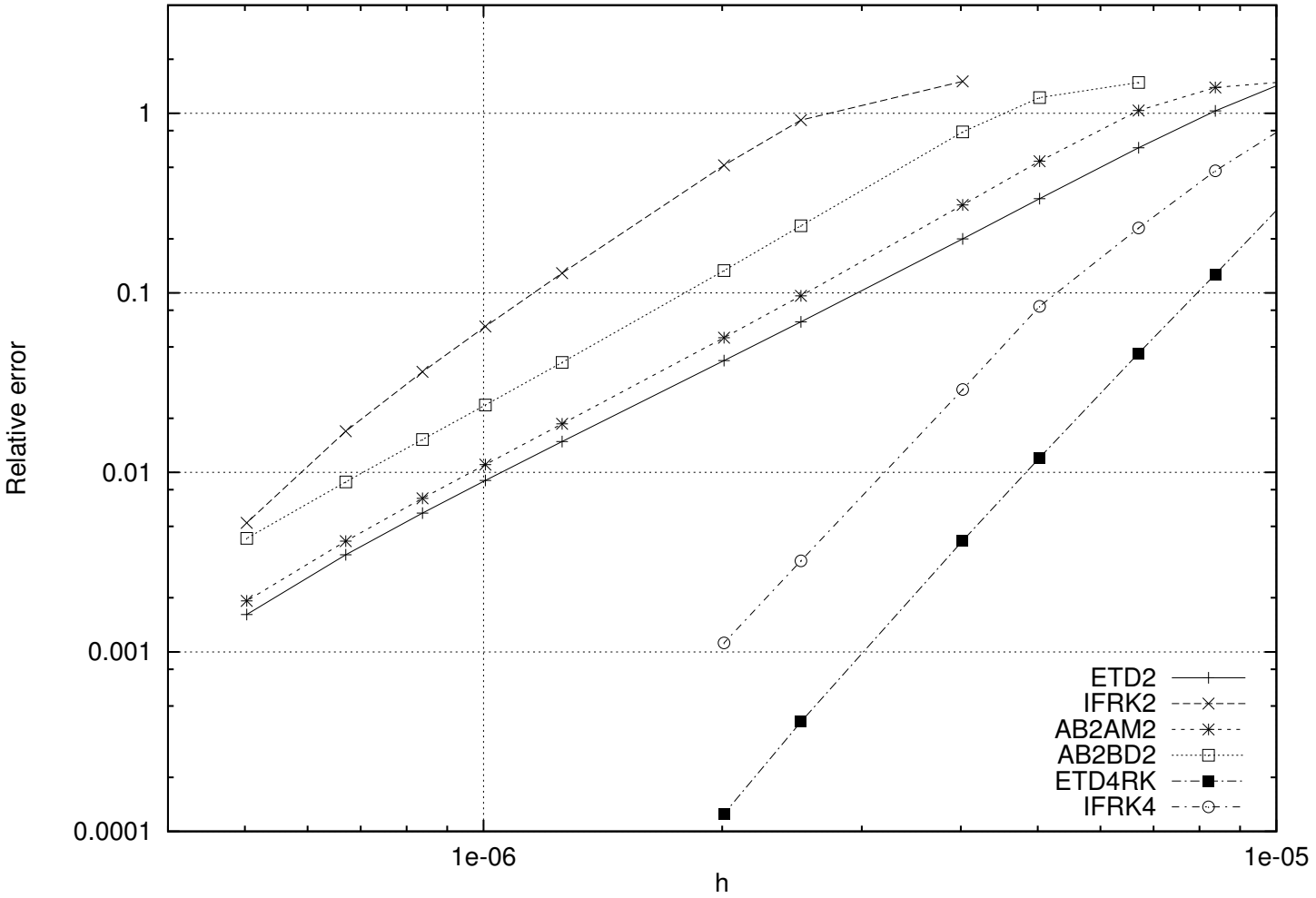
$$\frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} + \frac{\partial^3 u}{\partial x^3} = 0.$$

Soliton solution $u = f(x - ct)$, where $f(x) = 3c\,\mathrm{sech}^2(c^{1/2}x/2)$, $c = 625$.
Follow one period, i.e. up to $t = 2\pi/c$.

Compare different methods.

2nd order methods: ETD better than LI better than IF.

4th order methods: ETDRK better than IFRK (LI methods unstable).



Errors for KdV Soliton. From Cox & Matthews (2002).

Non-diagonal case

For Fourier methods the linear parts decouple.
But for finite differences or finite elements the linear terms are coupled,

$$\frac{du}{dt} = Mu + F$$

where M is a (banded) matrix.
We can still use exponential methods, for example ETD1 becomes

$$u_{n+1} = e^{Mh}u_n + M^{-1}(e^{Mh} - I)F_n.$$

This requires the matrix exponential (`expm` in Matlab) and the inverse of M. This is computationally expensive if M is large but only needs to be done once at the start of the computation.

Exponential time differencing coefficients

ETD methods of order n involve the 'ϕ functions' $\phi_0 \ldots \phi_n$.

$$\begin{aligned}
\phi_0(z) &= e^z, \\
\phi_1(z) &= (e^z - 1)/z, \\
\phi_2(z) &= (e^z - 1 - z)/z^2, \\
\phi_n(z) &= \frac{1}{z^n}\left(e^z - \sum_{k=0}^{n-1} \frac{z^k}{k!}\right) = \sum_{k=0}^{\infty} \frac{z^k}{(k+n)!}
\end{aligned}$$

These functions become increasingly difficult to evaluate as n increases, for small z, due to subtraction errors.

In the scalar case we can simply use the explicit formula for large z and truncated Taylor series for small z.

But in the matrix case, we cannot do this, since typically z has large and small (or zero) eigenvalues.

Remarks:

- To solve $\dot{u} = M\,u + F(u,t)$ with fixed time step h and fixed linearisation, we only need to find the functions $\phi_n(Mh)$ once at the start.
- For Fourier spectral methods M is diagonal so the problem reduces to the scalar case.
- For finite difference or finite element methods, M is sparse. In this case, e^{Mh} and $\phi_n(Mh)$ are 'fairly sparse' (Iserles 2000).
- If we are re-linearising each step, there is no need to find the ϕ_n functions. But need a matrix exponential each step (Saad 1992) or use a Krylov subspace approximation (Friesner et al 1989, Hochbruck et al 1998).

Tests:
Compare methods to find the functions $\phi_n(Mh)$ where M is the 40×40 finite difference matrix for the second derivative,
$M(j,j) = -2$, $M(j, j \pm 1) = 1$, for different values of h.
(Similar results for FD first derivative and Chebyshev matrix).

(i) Cauchy integral method (Kassam & Trefethen)

Use Cauchy Integral Formula (matrix form):

$$f(z) = \frac{1}{2\pi i} \int_\Gamma f(t)(tI - z)^{-1}\, dt$$

where the contour Γ encloses all the eigenvalues of z. Approximate using periodic trapezium rule, using a circle of N points $t_j = z_0 + re^{i\theta_j}$:

$$f(z) \approx \frac{1}{N} \sum_{j=1}^{N} re^{i\theta_j} f(t_j)(t_j I - z)^{-1}.$$

- Fairly slow – need N matrix inverses
- Need to know where eigenvalues are
- Inaccurate if contour too large: error $\sim r^N/N!$

(Recent improved method, Schmelzer & Trefethen 2007)

(ii) Scaling and squaring

Choose m so that $2^{-m}||z|| < \delta$, some threshold value ~ 1.
Use Taylor series or Pade to find $\phi_n(2^{-m}z)$.
Then apply squaring rules m times to find $\phi_n(z)$.

$$\begin{aligned}
\phi_0(2z) &= (\phi_0(z))^2, \\
\phi_1(2z) &= (\phi_0(z)\phi_1(z) + \phi_1(z))/2, \\
\phi_2(2z) &= (\phi_1(z)^2 + 2\phi_2(z))/4.
\end{aligned}$$

- Widely used for the matrix exponential (Matlab, Higham)
- Speed depends on number of squarings m required
- Accuracy decreases as m increases – relative error $\sim 2^m$
- Different forms of scaling rules give very similar accuracy

(iii) 'Compound matrix' method

Consider the matrix

$$M_4 = \begin{pmatrix} z & I & 0 & 0 \\ 0 & 0 & I & 0 \\ 0 & 0 & 0 & I \\ 0 & 0 & 0 & 0 \end{pmatrix}.$$

Then it can easily be shown that

$$e^{M_4} = \begin{pmatrix} \phi_0(z) & \phi_1(z) & \phi_2(z) & \phi_3(z) \\ \cdots & \cdots & \cdots & \cdots \end{pmatrix}$$

(minor variation of results of Saad, Sidje). So, if we have a routine for the matrix exponential (`expm`) we can find all the ϕ_n functions with one call.

- Very easy to code
- As accurate as the underlying matrix exponential subroutine
- Inefficient – slow if z is large (1.5 sec if z is 128x128)

(iv) Diagonalisation method

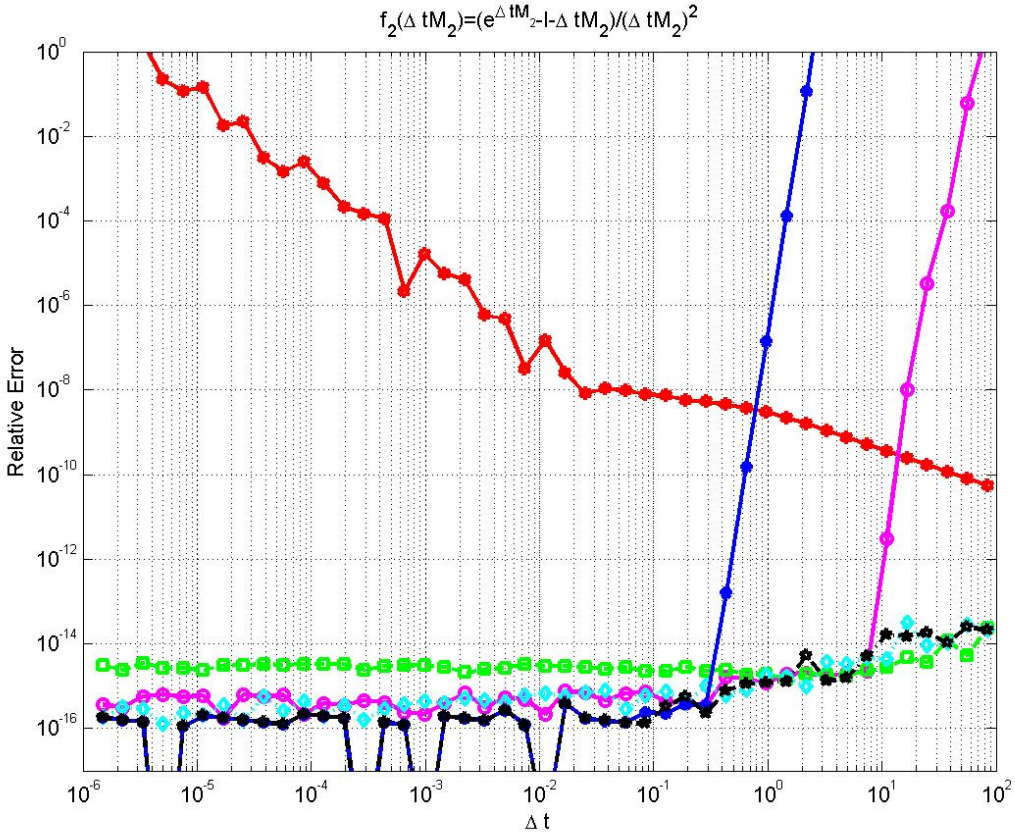
If V is the matrix of eigenvectors of z then $z = VDV^{-1}$
where D is the diagonal matrix of eigenvalues λ, so

$$\phi_n(z) = V\phi_n(D)V^{-1}.$$

Find $\phi_n(D)$ as in scalar case using explicit formula for large λ, Taylor series for small λ.

- Fast and accurate for 'nice' matrices (eg 2nd derivative FD)
- Fails if z is not diagonalisable

Results for ϕ_2



Explicit formula, Taylor series, Cauchy (128 points), Scaling and squaring, Diagonalisation, Composite matrix

Summary of Lecture 3

- Fourier in space, exponential in time methods are very accurate.
- Coding is easy in Matlab.
- ETD better than IF in most applications.
- It is worth using 4th order methods.
- Need to find ϕ_n functions at start of computation.
  Easy for Fourier spectral methods.
  For other space discretisations there are several suitable methods, including diagonalisation, scaling and squaring.
- More work needed on application to serious large-scale problems.