

# MTH 452-552 Winter 2018

## Class notes

Malgorzata Peszyńska

Department of Mathematics, Oregon State University

Winter 2018

# Introduction and Overview

These notes are intended to supplement rather than replace any textbook material or material covered in lectures.

Examples will be worked out in class.

*I will use material from several references*

---

- [RJL]. R.J. LeVeque, Finite Difference Methods for ODEs and PDEs, SIAM 2007
- [AP] U. Ascher, L. Petzold, SIAM 1998
- [Stro] S. Strogatz, Nonlinear Dynamics and Chaos, 1994
- [SH] Stuart and Humphries, Dynamical Systems and Numerical Analysis, Cambridge, 1996

## The purpose ... (by Richard Hamming)

- "The purpose of computing is insight, not numbers"
- "The purpose of analysis is to solve problems, not create pretty theorems"

# ODE types

$$u'(t) = f(t, u), \quad t \in I = [t_0, T], \quad u : I \rightarrow \mathbb{R}^n \quad (1)$$

- ❶ Higher order ODEs can be converted to (1)
  - IVP: supplement (1) with  $u(t_0) = u_0$ .
    - For higher order ODEs, need data  $u'(t_0) = \dots$
  - BVP for e.g.  $u'' = f(t, u)$ , require  $u(a) = u_a, u(b) = u_b$ .
- ❷ Does the solution exist? Is it unique? How does it depend on its data  $(f, u_0)$ ? Is the problem well-posed?
- ❸ What is the qualitative nature of solutions? (Stability, growth. For systems ( $n > 1$ ): equilibria. Behavior near equilibria. )
- ❹ Answers to 2-3 depend on the properties of  $f(\cdot, \cdot)$ : (non)linearity, smoothness, bounds on  $f, f_u$
- ❺ Numerical methods have to honor 3.

---

*Suggested review/reading: solving separable ODEs, bounding of functions, direction fields, phase plane, pendulum example*

# Well-posedness = E/U & continuous dependence

Consider a general IVP for (1),  $\mathcal{D} := \{(t, u) : t \in [0, T], \|u - u_0\| \leq \gamma\}$ .

- Well-posedness for general nonlinear  $f$ 
  - ➊ Require  $f$  continuous on  $\mathcal{D}$ , with  $M = \sup_{\mathcal{D}} \|f(t, u)\|$
  - ➋ Require  $f$  Lipschitz continuous in  $u$  on  $\mathcal{D}$   
 $\exists L \geq 0: \|f(t, u) - f(t, v)\| \leq L \|u - v\|, \quad \forall u, v, t \in \mathcal{D}.$ 
    - If  $f_u$  bounded, use  $L = \sup_{\mathcal{D}} \|f_u(t, u)\|$
    - If  $f$  defined piecewise, take maximum of  $f_u$  over the pieces
  - ➌ From 1-2 obtain (local) E/U for  $0 \leq t \leq \min(T, \gamma/M)$ .
  - ➍ Sometimes  $\mathcal{D} = [0, T] \times \mathbb{R}$ . Then E/U holds on  $[0, T]$ .
  - ➎ Sometimes  $f$  and  $f_u$  continuous in the interior of  $\mathcal{D}$ . Then local E/U holds.
  - ➏ For linear IVP  $u' = A(t)u + g(t)$  with continuous  $A(t), g(t)$ , we have global E/U, i.e., with  $T = \infty$ .
- For a given  $f$ , identify  $\mathcal{D}$  so that conditions 1-2 hold (or 4, or 5, or 6)

Ex. of Lipschitz continuity, local only existence, and non-uniqueness.

## Examples

- $y' = 3y$  and  $y' = -3y$
- $y' = \cos(t)y + t^2$  and  $u' = -tu$  and  $y' = \cos(y)$
- $u' = u^2$  and  $u' = \sqrt{u}$
- $y' = y(1 - y)$  (autonomous, two equilibria)
- $y' = -5ty^2 + \frac{5}{t} - \frac{1}{t^2}$ ,  $y(1) = 1$ .
- $y' = -100(y - \sin(t))$ ,  $y(0) = 1$  (stiff: large initial transient)
- pendulum  $\ddot{\theta} + \sin(\theta) = 0$
- oscillator  $\dot{x} = y$ ,  $\dot{y} = -x$  (eigenvalues?)
- van der Pol  $\dot{x} = y$ ,  $\dot{y} = -x + y(1 - x^2)$
- multi-body (or molecular dynamics)  $\ddot{r} = f(t)$ , with  $f = -\nabla V(r(t))$ , here  $r(t) \in \mathbb{R}^N \times \mathbb{R}^3$
- $u_t - u_{xx} = 0$  (with homogeneous DBC) discretized with FD in space gives  $U' + AU = 0$ , with a large spd matrix  $A$
- $u_t + u_x = 0$  discretized with FD/FV in space (or F-transformed) gives  $U' = AU$ ;  $A$  large with complex eigenvalues

# Continuous dependence & stability

Consider  $u' = f(t, u)$ ,  $u(0) = u_o$  and  $\hat{u}' = \hat{f}(t, \hat{u})$ ,  $\hat{u}(0) = \hat{u}_o$

## Theorem 1.1 (Continuous dependence)

*With notation from the E/U slide, we have a continuous dependence on the perturbation to initial data and perturbation  $\hat{f}$  to  $f$ :*

$$\|u(t) - \hat{u}(t)\| \leq e^{Lt} \|u(0) - \hat{u}(0)\| + \frac{\sup_{\mathcal{D}} \|f - \hat{f}\|}{L} (e^{LT} - 1) \quad (2)$$

## Stability for IVP

Solution  $u(t)$  to (1) is stable if

$\|u(0) - \hat{u}(0)\| \leq \delta$  implies  $\|u(t) - \hat{u}(t)\| \leq \varepsilon$ , for  $t > 0$ .

---

*Ex.: consider the linearized problem  $u' = \lambda u$  which is stable if, e.g.,  $\operatorname{Re}(\lambda) \leq 0$ .*

# Basic numerical analysis (FD for ODEs)

Goal: find  $u_h$  such that  $u_j = u_h(t_j) \approx u(t_j)$ . Analyze & control the error.

- Need an algorithm for  $u_h$ 
  - Approximate  $u'(x) \approx D_h u(x)$  and **analyse the error** *a-priori* depending on  $u$ ; e.g.  $D_h u(x) - f'(x) = O(h^\alpha)$ .

## Example 1.2

FE:  $u'(t) \approx \frac{u(t+h)-u(t)}{h}$ . Explicit  $u_j = u_{j-1} + hf(t_{j-1}, u_{j-1})$

BE:  $u'(t) \approx \frac{u(t)-u(t-h)}{h}$ . Implicit  $u_j = u_{j-1} + hf(t_j, u_j)$

- How large is the error  $u_h - u$ ?
  - In one step (FE, BE) we make  $O(h)$  local truncation error. (This is *a-priori* analysis)  
**How does it accumulate in  $u_h - u$ ?** (stability?)
  - How to **estimate the error** *a-posteriori* without knowing  $u$ ?

## FE example

```
function FEerr = FEerrexample(h)
%% set up function
f=@(t,y)(-t.*y);
uexact = @(t)(5*exp(-t.^2/2));
%% set up discretization
T=10;tsteps=(0:h:T)';n=length(tsteps);
%% the numerical solution will be in vector uh
uh = 0*tsteps;
%% initial condition
uh(1)=uexact(0);

%% time stepping loop
for j=2:n, uh(j)=uh(j-1)+h*f(tsteps(j-1),uh(j-1));end;

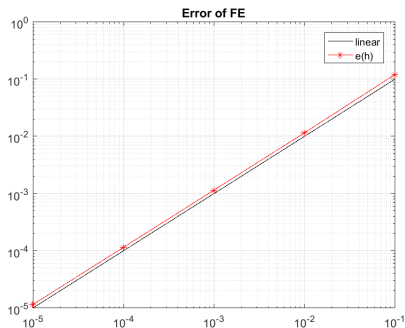
%% plot the numerical and exact solutions
plot(tsteps,uexact(tsteps),'-',tsteps,uh,'*');legend('exact','FE');
title(sprintf('Solution with dt=%g',h));
%% calculate error
FEerr = norm(uh-uexact(tsteps),inf);
```



# Results of FE convergence analysis

```
>> for j=1:5, h(j)=10^(-j);e(j)=FEerr(h(j));end
>> loglog(h,h,'k-',h,e,'r*-');grid on
>> title('Error of FE');legend('linear','e(h)');
>> print -dpng errorh.png
>> for j=2:5, alpha(j)=(log(e(j))-log(e(j-1)))/(log(h(j))-log(h(j-1)));end
>> format long; alpha
>> polyfit(h,e,2)
```

| $h$    | $e(h)$       | $\alpha$ |
|--------|--------------|----------|
| 0.1    | 0.1207867637 |          |
| 0.01   | 0.011560975  | 1.019    |
| 0.001  | 0.001150700  | 1.002    |
| 0.0001 | 0.000115016  | 1.0002   |



# More on the error in numerical solution to ODEs

## What about other sources of error?

- The main error contribution in numerical methods for ODEs is  
*discretization error = local truncation error + its accumulation*
  - We generally hope not to run into roundoff error. But ...
  - FE and other explicit methods require only function evaluations
  - BE and other implicit methods require nonlinear solvers, e.g.,  
*simple (Picard) iteration, Newton's method.*
- 
- If solved only to certain tolerance, further error incurs.*
- Systems of ODEs (may) require linear solvers.
- 
- This might contribute additional source of error.*

## Error $u - u_h$ is not all! Many flavors of numerical stability

- First, we constrain  $h$  to limit the LTE & its propagation.
- Second, we address the qualitative nature of  $u_h$  and  $u$ .

---

*We will frequently consider the test equation  $u' = \lambda u$  to test (linear) stability of a numerical method.*

The next few slides provide general remarks on testing algorithms, notation, and some new material on solving implicit schemes.

# Algorithms and code must be tested!. How?

**Remark 1** (Numerical methods are created because we cannot find (analytical) (true) solutions to (most) problems.)

*This is an issue for testing ☺.*

## Example 2.1 (First workaround: manufacture problems)

Assume you know  $u(t)$ , calculate  $u'(t)$ . Propose some  $f(t, u) = u'(t)$ .

## Example 2.2 (Second workaround: use refined grid solution)

Consider  $u_{h_{\text{very fine}}}$  calculated with your own scheme, and  $h_{\text{very fine}}$  at least one or two orders of magnitude smaller than  $h$  you test. Or, consider some other highly accurate well-known scheme as a proxy for  $u(t)$ .

**Remark 2** (When implementing a new (or a known) scheme, you must also test your implementation.)

*Test always. Test frequently.*

*Search literature to find analytical solutions for challenging examples.*

# Notation in FD (finite differences)

## Remark 3 (Notation in FD for IVP, and for one independent variable)

So far we have denoted the time steps  $t_j = hj$ ,  $j = 0, 1, \dots$  and  $u_j \approx u(t_j)$ . The collection of  $(u_j)_j$  forms the approximation  $u_h$  to  $u$ . ([RJL, Chap 1-2])

- Important: the grid function  $u_h$  is not defined at every  $t$
- When plotting numerical approximation, e.g., in MATLAB, we see, however, a line joining  $u_j$  and  $u_{j+1}$  etc. which leads some to believe that  $u_h$  is a piecewise linear interpolant of  $(u_j)_j$ , and a function of  $t$

---

This is not true: one could define many functions based on the same collection of values  $(u_j)_j$ . Other methods than FD such as Finite Elements make a distinction.

## Remark 4 (More than one independent variable)

It is customary [RJL, Chap 5-10] to denote the grid as follows

- in space by superscripts  $x_j, j = 1, \dots, M$  with  $U_j \approx u_j = u(x_j)$
- in time  $t_n, n = 0, 1, \dots, N$  with superscripts  $U^n \approx u^n = u(t_n)$

---

Clearly  $U^n$  has to be distinguished from the  $n$ 'th power of  $U$  !!!

- For PDEs, we will use  $U_j^n \approx u_j^n = u(x_j, t_n)$
- It is common to denote the spatial step by  $h$  and the time step by  $\tau, \Delta t$  or by  $k$  [RJL, Chap 5-10]. We will use  $h$  instead of  $k$  so we can use  $u_h$  not  $u_k$  as the grid function.

# Notation on error and on systems

## Definition 2.3

The error in FD for an IVP is  $E^n = U^n - u(t_n)$ ; we test  $\max_n \|E^n\|$ .

*Typically,  $\|E^n\|$  increases with  $n$ ; we frequently only consider  $\|E^n\|$  at  $t_n = T$ .*

## Remark 5 (Calculating error for a system of ODEs)

When  $u(t) \in \mathbb{R}^d$ , e.g., in the oscillator example, we write

$$u'(t) = \begin{bmatrix} x(t) \\ y(t) \end{bmatrix}' = \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix} \begin{bmatrix} x(t) \\ y(t) \end{bmatrix} = Au \quad (3)$$

with an orthogonal matrix  $A$ . The numerical solution

$U^n = [x^n, y^n]^T \approx u^n$  can be found, and the error is measured with a norm  $\|\cdot\|$  on  $\mathbb{R}^d$  of our choice. See [\[RJL, Appendix A\]](#).

The most common norms are the  $p$ -norms, and  $p = 2$  is the usual Euclidean norm

For  $d = 1$  the  $p$  norms reduce to  $|\cdot|$ .

Ex.: Calculate  $\|[-i, 5]^T\|_2$ . Show  $\|A\|_2 = 1$ .

# How to solve implicit schemes, e.g., BE

Recall FE scheme

$$u_j = u_{j-1} + hf(t_{j-1}, u_{j-1}). \quad (4)$$

For every  $j$  we can calculate  $u_j$  explicitly from the right hand side. However, BE scheme

$$u_j = u_{j-1} + hf(t_j, u_j), \quad (5)$$

unless  $f$  is linear in  $u$ , cannot be solved explicitly for  $u_j$ . The scheme is called implicit, and (5) is solved by iteration for  $u_j$ .

- There is no general direct way to solve nonlinear algebraic problems
- We can only use iterative methods such as simple (fixed-point, Picard) iteration or Newton's method, which are not guaranteed to converge
- Any iterative scheme converges only up to a certain tolerance. This produces an additional error in  $u_h$  at every time step  $t_j$ .

# Solving an implicit scheme by iteration

We consider two types of solvers: fixed point and Newton iteration.

- At every time step  $j$  start with an initial guess  $u_j^{(0)}$ .

---

*A good initial guess is provided by the previous time step value, so you can set  $u_j^{(0)} = u_{j-1}$ . But other possibilities, e.g., by extrapolation, may be better*

---

- Then iterate  $u_j^{(0)}, u_j^{(1)}, \dots, u_j^{(k)}, \dots$  until done.
- “Done” means that the tolerance criteria are met.
  - Sometimes iteration does not converge, or does not make much progress towards convergence. For ODE schemes, this means that a smaller time step  $h$  should be considered.
  - For scalar problems, fixed iteration should be set-up to converge in 10-20 iterations. Newton’s method should not take more than 5-10. Setting up the maximum number of iterations prevents the code from useless cycles which make no progress.
- Fixed point iteration requires function evaluations, and Newton’s method requires evaluations of a function (residual) and of its derivative (Jacobian).



# Solving implicit schemes by fixed point iteration

(Notation as from slide 7)

**Solving (5): find  $u_j$  as the fixed point of**  $v = g(v)$

**Here**  $g(v) := u_{j-1} + hf(t_j, v)$ .

Iteration proposes new guesses  $v^{(1)}, v^{(2)}, \dots$  from

$$v^{(k+1)} = g(v^{(k)}), \quad k = 0, 1, \dots \quad (6)$$

- 1 The iteration converges if  $g(\cdot)$  is a contraction (i.e., its Lipschitz constant  $L_g < 1$ ); this will always work of course, if  $f$  is Lipschitz, and  $h$  is small enough
- 2 The convergence is only linear, but implementation is easy.
- 3 “Done”: stop, e.g., if  $\|v^{(k)} - g(v^{(k)})\|$  is small enough.

**Example 2.4 (Find small enough time step for  $u' = \sin(5u)$  solved with BE and fixed point)**

The scheme reads  $u_j = u_{j-1} + h \sin(5u_j)$ , thus we iterate  $u_j^{(k+1)} = g(u_j^{(k)})$ , where  $g(v) = u_{j-1} + h \sin(5v)$ . We calculate Lipschitz constant  $L_g = \sup_v |g'(v)| = 5h$ . To ensure  $L_g < 1$ , we require  $h < \frac{1}{5}$ .

# Solving implicit schemes by Newton's method

**Solve (5): find  $u_j$  as the zero of  $G(v) = 0$ .**

**Here  $G(v) := v - g(v) = v - u_{j-1} - hf(t_j, v)$ .**

To solve  $G(v) = 0$ , at each iteration  $k = 0, 1, \dots$ , we linearize

$$G(v^{(k+1)}) \approx G(v^{(k)}) + (v^{(k+1)} - v^{(k)})DG|_{v^{(k)}}s^{(k)} = 0.$$

Rearranging, we solve for the correction (step)  $s^{(k)} = v^{(k)} - v^{(k+1)}$  the equation

$$DG|_{v^{(k)}}s^{(k)} = G(v^{(k)}) \tag{7}$$

and then update  $v^{(k+1)} = v^{(k)} - s^{(k)}$ .

The Jacobian  $DG|_{v^{(k)}}$  and the residual  $G(v^{(k)})$  are evaluated at every iteration; (7) is solvable as long as  $DG$  is nonsingular.

# Newton's method, cd.

- ❶ Newton's iteration converges if (i) the initial guess  $v^{(0)}$  is close to the true solution, and if (ii)  $G(\cdot)$  and  $DG$  are smooth enough, and if (iii)  $DG$  is never singular (for scalar problems, this means  $DG \neq 0$ ).
- ❷ Newton's iteration converges very fast when close to the root, but may diverge. Newton's method is more expensive than fixed point.
- ❸ Iteration may stop, e.g., if  $\|G(v^{(k)})\|$  is small enough.

---

*Some variants of Newton's method bypass the need to evaluate the Jacobian at every iteration*

---

**Example 2.5 (Find  $h$  so that the Newton's step for BE and  $u' = \sin(5u)$  is guaranteed to be solvable)**

The scheme reads  $u_j = u_{j-1} + h \sin(5u_j)$ , thus we iterate  $G(u_j^{(k+1)}) = 0$ , where  $G(v) = v - u_{j-1} - h \sin(5v)$ . To ensure  $G'(v) = 1 + 5h \cos(v) \neq 0$ , we require  $|5h \cos(v)| < 1$ , or  $h < \frac{1}{5}$ , same as in Ex. 2.4.

---

*This bound is sufficient for the solvability of the Newton step and may not be necessary. At the same time this bound does not guarantee that Newton iteration will converge for any initial guess!*

---

# Code for BE: fixed point or Newton

```
f = @(t,u)(u.^2); fprime = @(t,u)(2*u);
%% stopping criteria for nonlinear solver
atol = 1e-12;
if choice ==1, method= 'Picard'; maxit = 20; else method = 'Newton'; maxit=10; end
%%
for j=1:n
    %% initial guess
    guess = yi(j); iter = 0;
    %% iterate by Picard or Newton
    while 1
        if choice == 1 %% fixed point=Picard
            %% evaluate g(guess)=uprevious+hf(guess), check residual size, and continue
            gval = yi(j)+h*f(tsteps(j+1),guess);
            resnorm = abs(guess-gval);
            newguess = gval;
        else %% Newton
            %% evaluate G(guess) = uguess - g(guess), g as in Picard
            gval = yi(j)+h*f(tsteps(j+1),guess);
            res = guess - gval; jac = 1 - h*fprime(tsteps(j+1),guess);
            resnorm = abs(res);
            newguess = guess - jac \ res;
        end
        if resnorm<atol, guess = newguess;break;
        elseif iter>maxit, error(' no convergence for method= ',method);
        else iter = iter+1; guess = newguess;end
    end
    maxiter = max(maxiter,iter);
    aveiter = aveiter + iter;
    yi (j+1) = guess;
end
...
```



The next few pages summarize theory of consistency for FD schemes. In the first part of the class we study consistency of schemes, i.e., the order of LTE.

### Definition 3.1

A FD scheme is consistent if the LTE is at least  $O(h)$ .

Later we study **stability**, which measures how the error propagates and accumulates, from step to step.

# Truncation error

## Definition 3.2 (Local Truncation Error (LTE))

Informally, truncation error is what remains after you plug in the true solution to the FD scheme, apply Taylor's expansions, and the ODE.

*Make sure you calculate LTE using the form of the scheme resembling  $u' = f(t, u)$  rather than some other algebraically equivalent form.*

## Example 3.3 (Find $\tau$ =LTE for FE; see (4))

Scheme  $\frac{U^{n+1} - U^n}{h} = f(t_n, U^n)$ . Plug in  $\frac{u(t_{n+1}) - u(t_n)}{h} = f(t_n, u(t_n)) + \tau$ .

Apply Taylor expansions  $u(t_{n+1}) = u(t_n) + hu'(t_n) + \frac{h^2}{2}u''(t_n) + \dots$

Cancel terms with  $u'(t_n) = f(t_n, u(t_n))$ . Conclude  $\tau = O(h)$ .

## Remark 6 (LTE occurs at every step, and it will accumulate.)

*We will study the notion(s) of numerical stability of schemes which controls how the error accumulates, and prevents catastrophic and pathological accumulation.*

Repeat Ex. 3.3 for BE, trapezoidal, and midpoint methods. Choose the point for Taylor expansions carefully.

# Higher order approximations to $u'(t)$

## Example 3.4 (Symmetric second order approximation to $u'(t)$ )

Average  $D_h^+ u = \frac{u(t+h)-u(t)}{h}$  and  $D_h^- u = \frac{u(t)-u(t-h)}{h}$ ;  $D_h^0 u = \frac{u(t+h)-u(t-h)}{2h}$ .

Check accuracy: assume smoothness and Taylor-expand

$$u(t+h) = u(t) + hu'(t) + h^2/2u''(t) + h^3/6u^{(iii)}(t) + \dots \quad (8)$$

$$u(t-h) = u(t) - hu'(t) + h^2/2u''(t) - h^3/6u^{(iii)}(t) + \dots \quad (9)$$

Subtract (9) from (8) to check that  $u'(t) - D_h^0 u(t) = O(h^2)$ . (Midpoint or leapfrog).

**General idea: use Taylor expansions in an organized way [RJL,p7]**

To get a method of order  $p$ , we typically need  $p+1$  points in a stencil for the difference approximation. We Taylor expand each, and find how to mix and match these so that the terms of order up to  $O(h^p)$  to cancel. (Method of undetermined coefficients)

## Example 3.5 (One-sided forward second order approximation to $u'(t)$ )

Use  $u(t), u(t+h), u(t+2h)$ , each Taylor expanded about  $u(t)$ . Combine them in  $D_h^{\text{onesided},f} u(t) = \frac{au(t+2h)+bu(t+h)+cu(t)}{h}$ . Write down a system of equations, and solve for  $a, b, c$  so that  $D_h^{\text{onesided},f} u(t) - u'(t) = O(h^2)$ . (Find  $a = -1/2, b = 2, c = -3/2$ .)



# Another way to get higher order approximations

## General technique: Richardson extrapolation

Given an exact value  $I$ , its approximation  $I_h$  which is  $O(h^p)$  accurate, i.e.,  $E(h) = I - I_h = O(h^p) = Ch^p + O(h^{p+1})$ , produce a new scheme which is  $O(h^{p+1})$  accurate by step doubling  $I_h^{new} = 2^p I_h - I_{2h}$ , with  $E_h^{new} = O(h^{p+1})$ .

### Example 3.6 (One sided forward second order approximation to $u'(t)$ )

Use FE approximation to  $I = u'(t) \approx I_h = D_h^+ u = \frac{u(t+h) - u(t)}{h}$  with  $E(h) = Ch + O(h^2)$ , so  $p = 1$ .

Calculate  $D_h^{new} u = 2D_h^+ u - D_{2h}^+ u = \frac{-u(t+2h) + 4u(t+h) - 3u(t)}{2h}$ , and  $E_h^{new} = O(h^2)$ .

Compare with Ex. 3.5 to see  $D_h^{new} = D_h^{onesided, f}$

### Example 3.7 (One sided backward second order approximation to $u'(t)$ )

We can either follow Ex. 3.5 or Ex. 3.6, or simply replace  $h$  by  $-\tilde{h}$ . (Check the analysis with Taylor!). We obtain, after replacing  $h$  by  $\tilde{h}$  again

$$D_h^{onesided, b} u(t) = \frac{3u(t) - 4u(t-h) + u(t-2h)}{2h} \approx u'(t) + O(h^2) \quad (10)$$

Ex.: Construct fourth order accurate symmetric and one-sided approximation to  $u'(t)$

# Use higher order approximations to $u'(t)$ in FD multistep schemes for ODE $u'(t) = f(t, u(t))$

## Example 3.8 (Leapfrog $D_h^0 u$ , as in Ex. 3.4)

Recall  $D_h^0 u = \frac{u(t+h)-u(t-h)}{2h} \approx u'(t) + O(h^2) = f(t, u(t)) + O(h^2)$ .

The leapfrog scheme  $\frac{U^{n+1}-U^{n-1}}{2h} = f(t_n, U^n)$  is explicit.

## Example 3.9 (One-sided forward second order, Ex. 3.5 & 3.6)

Recall  $D_h^{\text{onesided},f} u(t) = \frac{-u(t+2h)+4u(t+h)-3u(t)}{2h} \approx u'(t) + O(h^2)$

The scheme  $\frac{-U^{n+2}+4U^{n+1}-3U^n}{2h} = f(t_n, U^n)$  is explicit.

## Example 3.10 (One-sided backward second order; Ex. 3.7)

Recall  $D_h^{\text{onesided},b} u(t) = \frac{3u(t)-4u(t-h)+u(t-2h)}{2h} \approx u'(t) + O(h^2)$

The scheme  $\frac{3U^{n+2}-4U^{n+1}+U^n}{2h} = f(t_{n+2}, U^{n+2})$  is implicit.

Equivalently,  $\frac{3U^{n+1}-4U^n+U^{n-1}}{2h} = f(t_{n+1}, U^{n+1})$  [R~~J~~L,Sec.5.3] (B~~D~~F)

## Another idea: sample $f$ in a better way

Integrate  $u'(t) = f(t, u(t))$  over  $(t_n, t_{n+1})$  to obtain

$$u(t_{n+1}) = u(t_n) + \int_{t_n}^{t_{n+1}} f(s, u(s)) ds. \quad (11)$$

Approximate (11) with numerical integration (quadrature)  $\implies$  FD schemes.

### Example 3.11 (FE and BE: left/right rectangular rule)

$$(FE) : U^{n+1} = U^n + hf(t_n, U^n) \quad (12)$$

$$(BE) : U^{n+1} = U^n + hf(t_{n+1}, U^{n+1}) \quad (13)$$

### Example 3.12 (Crank-Nic(h)olson/trapezoidal & midpoint)

$$U^{n+1} = U^n + \frac{h}{2} (f(t_n, U^n) + f(t_{n+1}, U^{n+1})) \quad (14)$$

$$U^{n+1} = U^n + hf(t_{n+1/2}, U^{n+1/2}) \quad (15)$$

The value  $U^{n+1/2}$  in (15) is cumbersome. More on this later.

## Another way to get higher order

**Break-up the evaluation of  $\int_{t_n}^{t_{n+1}} f(s, u(s))ds$  into stages over  $(t_n, t_{n+1})$  while keeping the approximation to  $u'(t)$  simple.**

This idea yields the family of Runge-Kutta methods.

**Example 3.13 (2-stage explicit RK [RJL, p125] (improved Euler))**

Obtained by calculating  $\hat{U}^{n+1/2}$  with FE in half-step, and applying midpoint scheme (15) over  $(t_n, t_{n+1})$ , with  $\hat{U}^{n+1/2}$  instead of  $U^{n+1/2}$ .

$$\hat{U}^{n+1/2} = U^n + \frac{h}{2}f(t_n, U^n); U^{n+1} = U^n + hf(t_{n+1/2}, \hat{U}^{n+1/2}) \quad (16)$$

**Example 3.14 (2-stage explicit RK [RJL, p125] (Heun method))**

Obtained by calculating  $\hat{U}^{n+1}$  with FE, and applying trapezoidal scheme (14) over  $(t_n, t_{n+1})$ , with  $\hat{U}^{n+1}$  instead of  $U^{n+1}$ .

$$\hat{U}^{n+1} = U^n + hf(t_n, U^n); U^{n+1} = U^n + \frac{h}{2} \left( f(t_n, U^n) + f(t_{n+1}, \hat{U}^{n+1}) \right) \quad (17)$$

Find formally the LTE of the improved Euler and Heun methods.

# Runge-Kutta methods and Butcher tableaus

- Each RK method of  $r$  stages can be associated a diagram called Butcher tableau [RJL, p127]), with numbers  $c_i, b_j, a_{ij}$  with  $i = 1, \dots, r, j = 1, \dots, r$ .

$$\text{stage } i : Y_i = U^n + h \sum_j a_{ij} f(t_n + c_j h, Y_j) \quad (18a)$$

$$\text{when done } U^{n+1} = U^n + h \sum_j b_j f(t_n + c_j h, Y_j) \quad (18b)$$

---

*The stages are implicit if  $a_{ij} \neq 0$  for some  $j \geq i$*

- From the tableau you can quickly check the consistency conditions
  - for LTE to be at least  $O(h)$ :  $\sum_j a_{ij} = c_j, \sum_j b_j = 1$
  - for LTE to be at least  $O(h^2)$ :  $\sum_j b_j c_j = 1/2$
  - ... further ... for higher order
- Further use of tableaus is to encode *embedded* methods which help to estimate and control the error

# Butcher tableaus for Runge-Kutta methods

## Definition 3.15

|         |          |          |                   |
|---------|----------|----------|-------------------|
| $c_1$   | $a_{11}$ | $a_{12}$ | $\dots$           |
| $c_2$   | $a_{21}$ | $a_{22}$ | $\dots$           |
| $\dots$ | $\dots$  | $\dots$  | $\dots$           |
| $c_r$   | $a_{r1}$ | $a_{r2}$ | $\dots$           |
|         | $b_1$    | $b_2$    | $\dots \quad b_r$ |

Tableau corresponds to

$$\begin{aligned}
 Y_i &= U^n + h \sum_j a_{ij} f(t_n + c_j h, Y_j) \\
 &\dots \\
 U^{n+1} &= U^n + h \sum_j b_j f(t_n + c_j h, Y_j)
 \end{aligned}$$

## Example 3.16

Produce tableaus for FE, BE, Heun (17), modified Euler (16).

Tableau for the 4-stage explicit RK is

|               |               |               |               |               |
|---------------|---------------|---------------|---------------|---------------|
| 0             |               |               |               |               |
| $\frac{1}{2}$ | $\frac{1}{2}$ |               |               |               |
| $\frac{1}{2}$ | 0             | $\frac{1}{2}$ |               |               |
| $\frac{1}{2}$ | 0             | 0             | 1             |               |
| 1             | 0             | 0             | 0             | 1             |
|               | $\frac{1}{6}$ | $\frac{1}{3}$ | $\frac{1}{3}$ | $\frac{1}{6}$ |

Ex.: Practice producing Butcher Tableaus and unscrambling them

## Yet another way to get higher order

**Evaluate  $\int_{t_n}^{t_{n+1}} f(s, u(s)) ds$  by approximating  $f(s, u(s))$  by a polynomial based on (old) time step values, and integrating the polynomial exactly.**

This yields the family of (explicit) Adams-Bashforth methods. While formally single-step, they use old time step values.

### Example 3.17 (2-step Adams-Bashforth [RJL, p132])

Obtained by calculating the exact integral of linear polynomial  $p_1(t)$  based on  $U^{n+1}, U^n$ . More precisely, this polynomial uses  $f(t_{n+1}, U^{n+1})$  and  $f(t_n, U^n)$  to approximate  $f(t, u(t))$  on  $(t_{n+1}, t_{n+2})$ .

$$U^{n+2} = U^{n+1} + \frac{h}{2} (-f(t_n, U^n) + 3f(t_{n+1}, U^{n+1})) \quad (19)$$

---

Ex.: derive (19). Derive higher order schemes with  $p_2(t)$ , etc.

## Yet another way to get higher order

**Evaluate  $\int_{t_n}^{t_{n+1}} f(s, u(s))ds$  by approximating  $f(s, u(s))$  by a polynomial based on old and new time step values.**

This yields the family of (implicit) Adams-Moulton methods.

### **Example 3.18 (1-step Moulton $\equiv$ trapezoidal scheme)**

Obtain (14) by calculating an exact integral of the linear polynomial  $p_1(t)$  drawn from  $f(t_n, U^n), f(t_{n+1}, U^{n+1})$ .

### **Example 3.19 (2-step Moulton)**

Use  $p_2(t)$  based on  $f(t_n, U^n), f(t_{n+1}, U^{n+1}), f(t_{n+2}, U^{n+2})$ .

$$U^{n+2} = U^{n+1} + \frac{h}{12} (-f(t_n, U^n) + 8f(t_{n+1}, U^{n+1}) + 5f(t_{n+2}, U^{n+2})) \quad (20)$$

---

Ex.: derive (20)



# Last but not least: predictor-corrector schemes

Powerful heuristic schemes “predict” with an explicit step, and “correct” by plugging in the “predicted” value in another scheme, originally implicit.

*This method avoids iteration needed in an implicit scheme, and maintains the same order of the error as the corrector scheme, as long as the guess provided is sufficiently accurate.*

**Example 3.20 (Predict with AB-1 (FE), correct with AM-1 (Trapz))**

The resulting scheme is the same as Heun method (17).

## Remark 7

*If other errors due to the discretization of PDE, round-off, or solver, are an issue, the corrector equation may be iterated.*

Suggest a predictor for the Nyström rule

$$U^{n+2} = U^n + \frac{h}{3} (f(U^n) + 4f(U^{n+1}) + f(U^{n+2}))$$

# Other use of predictor-corrector: error estimates

## Example 3.21 (Continue (3.20))

Since the Heun method is of second order, and its first stage (FE) is of first order, we can subtract  $\hat{U}^{n+1} - U^{n+1}$  to estimate the error.

# Recap

Above we studied consistency ... and quantified the LTE error make in each time step of a scheme.

Now we need to discuss how this error propagates, so that a scheme is convergent.

## Definition 4.1

A FD scheme for a well-posed IVP (1), with  $f, u_0$  satisfying conditions for well-posedness, is convergent if

$$\lim_{h \rightarrow 0, hN=T} U^N = u(t_N). \quad (21)$$

*We expect  $\lim_{h \rightarrow 0} \max_{n: hn \leq T} \|U^n - u(t_n)\| = 0$  as long as the pbm is well posed on  $(0, t_n)$ .*

*To prove convergence, we will require consistency and more.*

# Proof of convergence, FE

## Theorem 4.2 (FE is convergent, if $u''$ is bounded.)

**Proof.** Consider FE first as in Ex. 3.3. Subtract the expression for the LTE from the scheme, somewhat rearranged, and with Def. 2.3, to get

$$E^n = E^{n-1} + h(f(t_{n-1}, U^{n-1}) - f(t_{n-1}, u(t_{n-1})) - h\tau_n \quad (22)$$

Next take norms, exploiting Lipschitz property of  $f$ , and estimating the terms on the rhs from above, and grouping terms.

$$\|E^n\| \leq \|E^{n-1}\| + hL \|U^{n-1} - u(t_{n-1})\| + h\|\tau_n\| = (1 + hL)\|E^{n-1}\| + h\|\tau_n\|.$$

By recursion we get  $\|E^n\| \leq (1 + hL)^n \|E^0\| + h \sum_{j=0}^{n-1} (1 + hL)^{n-j} \|\tau_j\|$ . Now  $E^0 = 0$ , and assuming  $u''$  is bounded we have  $\max_n \|\tau_n\| = O(h)$ . We proceed in one of two ways.

(a) The Bernoulli inequality gives  $(1 + hL)^n \leq e^{Lhn} = e^{LT}$ . Since  $\|\tau_j\| = O(h)$ , we have

$$\|E^n\| \leq h \sum_{j=0}^{n-1} (1 + hL)^{n-j} \|\tau_j\| \leq h \max_j \|\tau_j\| (1 + hL)^n n \leq Te^{LT} O(h) \rightarrow 0. \frac{e^{LT} - 1}{L} O(h) \rightarrow 0.$$

(b) A slightly more careful estimate, by summing the geometric series gives

$$\begin{aligned} h \sum_{j=0}^{n-1} (1 + hL)^{n-j} \|\tau_j\| &\leq h \max_j \|\tau_j\| \sum_{j=0}^{n-1} (1 + hL)^{n-j} \\ &= h \max_j \|\tau_j\| \frac{1 - (1 + hL)^n}{1 - (1 + hL)} \leq h \max_j \|\tau_j\| \frac{e^{LT} - 1}{hL} = \frac{e^{LT} - 1}{L} O(h) \rightarrow 0. \end{aligned} \quad (23)$$

and the theorem is proved. ■

Ex.: carry out the proof for the simpler case when  $L = 0$

# Convergence for BE

For BE, we derive

$$\|E^n\| (1 - hL) \leq \|E^{n-1}\| + h \|\tau_n\| \quad (24)$$

and proceed as for FE.

## The catch is that ...

... instead of  $(1 + hL)$  we deal with  $\frac{1}{1-hL}$ . To make sure this is a positive number, we require  $hL < 1$ . Then of course  $\frac{1}{1-hL} > 1$ . Don't worry however, since as  $h \rightarrow 0$ , the number  $\frac{1}{1-hL} \rightarrow 1$ , thus it makes sense to try to assume some upper bound. For example, if we restrict  $h$  to be small enough so that  $hL < \frac{1}{2}$ , we can use  $\frac{1}{1-hL} \leq 2$ . You can proceed first with (a) as in proof for FE. Next, try (b).

---

Ex.: Complete the proof of convergence of BE.

Ex.: Carry out the proof for the trapezoidal method.

# How to prove convergence of a general scheme of higher order?

You can't, without making assumptions how the scheme  
“accumulates” the error!

The property of some schemes to *NOT accumulate and propagate error in an uncontrolled or pathological way* is called

## Stability

We will discuss various notions of stability

- For convergence, we require zero stability

*Use  $f = 0$ , don't allow any growth!*

- For convergent schemes, to ensure “nice behavior”, we require
  - absolute stability, A-stability, L-stability,...

*Use  $f = \lambda u$ , require qualitative agreement of the scheme with the ODE.  
If  $\lambda$  is large, do not restrict  $h$  unduly.*

## Zero-stability when $u' = f = 0$

Any useful scheme should be zero-stable, i.e., with a small perturbation to the initial data, should not produce large errors.

### Example 4.3 (Example 6.2 from [RJL])

Consider the difference scheme  $U^{n+2} - 3U^{n+1} + 2U^n = 0$ , and use  $U^0 = 0, U^1 = h$ . After  $N = 20$  steps with  $h = \frac{1}{N}$ , we have a disaster:  $U^N \approx 10^4!$

### Example 4.4 (Consider $D_h^{onesided,f}$ )

Construct an appropriate scheme, and test it similarly as in Ex. 4.3.

The examples above are pessimistic: we used FE and BE with success! These are, however, single-step schemes, which do not require any additional starting values which might cause a headache.

### Example 4.5 (Recall BDF from Ex.3.10)

Follow examples before with  $3U^{n+2} - 4U^{n+1} + U^n = 0$ . No problem!

## How to test zero-stability of LMM? ( $u' = 0$ )

LMM = “linear multistep methods”. Here  $\sum_{j=0}^r \alpha_j U^{n+j} = 0$ .

**Remark 8 (Solving homogeneous linear ODE with constant coefficients  $\sum_{j=0}^r \alpha_j u^{(j)} = 0$ .)**

Recall the Ansatz (guess)  $u(t) = e^{kt}$ . Substitute, and derive an equation for  $k$ :  $\sum_{j=0}^r \alpha_j (k)^j = 0$ . Find the roots  $k_1, \dots, k_r$ . If no repeated roots, a general solution is the linear combination of  $e^{k_j t}$ . If root  $k_l$  is repeated (once), augment the solution by  $te^{k_l t}$ ; etc...

**Remark 9 (Solving homogeneous linear difference equation with constant coefficients  $\sum_{j=0}^r \alpha_j U^{n+j} = 0$ .)**

Guess  $U^j = e^{khj} = (e^{kh})^j = (\xi)^j$ , substitute, and derive an equation for  $\xi$ :  $\sum_{j=0}^r \alpha_j (\xi)^{j+n} = 0$ . Find the roots  $\xi_1, \dots, \xi_r$ . If no repeated roots, a general solution  $U^n$  is the linear combination of  $(\xi_j)^n$ . If the root  $\xi_l$  is repeated, use also  $n(\xi_l)^n$ .

Note: we use parentheses in  $(\xi)^n$  to help distinguish it from  $U^n$ .



## Definition 4.6 (Root condition)

A polynomial in  $\xi$  satisfies root condition if all of its roots  $\xi_j$  satisfy  $|\xi_j| \leq 1$ , and if those with  $|\xi_j| = 1$  are not repeated.

**Lemma 4.7 (Zero-stability of LMM is guaranteed if the polynomial  $\rho(\xi) = \sum_j \alpha_j(\xi)^j$  satisfies the root condition.)**

### Example 4.8 (FE, BE, leapfrog, and Ex. 4.3–4.5)

For FE, and BE,  $\rho(\xi) = \xi - 1$ , with only one root  $\xi_1 = 1$ . For leapfrog,  $\rho(\xi) = \xi^2 - 1$ , with two non-repeated roots  $\xi_{1/2} = \pm 1$ . For the schemes in Ex. 4.3–4.4 we find some roots  $\xi$  with  $|\xi| > 1$ . In turn, in Ex. 4.5 we get that the root condition holds!

### Example 4.9 (Adams methods $U^{n+r} - U^{n+r-1} = 0$ )

Here  $\rho(\xi) = \xi - 1$ , with only one root  $\xi_1 = 1$ .

Practice verifying the root condition for various schemes including AB, AM, Nystrom, ...

# Dahlquist theorem

## Theorem 4.10 (Dahlquist, for LMM)

*Consistency & zero-stability  $\equiv$  convergence.*

**Proof.** Use Frobenius companion matrix related to the LMM, and its norm when the root condition holds. (Show stability is sufficient for convergence) ... Show stability is necessary for convergence ... ■

# Absolute stability: use test equation $u' = \lambda u$

Even with zero-stable schemes, we have to be careful with the choice of time step  $h$  so that the error is not excessively large. (Denote  $h\lambda = z$  from now on).

*We study regions of absolute stability studying the test equation  $u' = \lambda u$ . If  $z = h\lambda$  are inside of the stability region, the error behaves nicely.*

First, consider a single step scheme for the test equation  $\frac{U^{n+1} - U^n}{h} = \dots$ , where the right-hand side contains various terms depending on how  $f$  is sampled. After we substitute  $f(t, u) = \lambda u$ , and rearrange, we can rewrite the scheme as  $U^{n+1} = R(z)U^n$ .

**Definition 5.1 (Region of absolute stability, single step scheme)**

...is the set  $\mathcal{R}^{ABS}$  of  $z \in \mathbb{C}$  for which  $|R(z)| \leq 1$ .

**Example 5.2 (Absolute stability for FE and BE)**

We have  $U^{n+1} = (1 + z)U^n$  in FE. Thus  $R(z) = 1 + z$ . Similarly, for BE we derive  $R(z) = \frac{1}{1-z}$ . Next we solve the inequality  $|R(z)| \leq 1$ . (See below in Ex. 5.4..5.5)

# Absolute stability: use $u' = \lambda u$ and LMM

Write LMM for the test equation

$$\sum_{j=0}^r \alpha_j U^{n+j} = h \sum_{j=0}^r \beta_j \lambda U^{n+j} = z \sum_{j=0}^r \beta_j U^{n+j}. \quad (25)$$

We consider now the polynomial  $\pi(z; \xi) = \rho(\xi) - z\sigma(\xi) = \sum_{j=0}^r (\alpha_j - z\beta_j)(\xi)^j$  parametrized by  $z$ . For a given  $z$ , we check if it satisfies the root condition. If yes, no excessive growth should occur. The roots of course depend on  $z = h\lambda$ .

**Definition 5.3 (Region of absolute stability for LMM)**

... is the set  $\mathcal{R}^{ABS}$  of  $z \in \mathbb{C}$  for which the roots of the polynomial  $\pi(z; \xi)$  satisfy the root condition in Def. 4.6.

*Many simple single step schemes such as FE and BE are LMM. Many are not ...*

# Absolute stability regions

## Remark 10 (Notation on complex numbers)

$B_a(r) = \{z \in \mathbb{C} : |z - a| \leq r\}$  is the closed ball with center  $a$  and radius  $r$ . Numbers  $\mathbb{C} \ni z = x + iy = (x, y)$ ;  $\Re z = x$ ;  $\Im z = y$  are the real and imaginary parts of  $z$ . Modulus  $|z| = \sqrt{x^2 + y^2}$ .

For a given scheme with the corresponding polynomial  $\pi(z; \xi)$ , the goal is to find  $\mathcal{R}^{ABS}$ . See examples in [RJL, Ex.7.4-7.7]

### Example 5.4 (FE; $\rho(\xi) = \xi - 1$ , $\sigma(\xi) = 1$ )

Calculate  $\pi^{FE}(z; \xi) = \xi - (1 + z)$  with root  $\xi_1 = 1 + z$ .

We require  $|1 + z| \leq 1$ , thus  $\mathcal{R}^{ABS, FE} = B_{(-1, 0)}(1)$ .

*Note: this means that we must have  $\Re \lambda < 0$  and  $-2 \leq h\lambda \leq 0$ .*

### Example 5.5 (BE; $\rho(\xi) = \xi - 1$ , $\sigma(\xi) = \xi$ )

For BE  $\pi^{BE}(z; \xi) = (1 - z)\xi - 1$ , with root  $\xi_1 = \frac{1}{1 - z}$ .

To get  $|\frac{1}{1 - z}| \leq 1$  we must have  $|1 - z| \geq 1$ .  $\mathcal{R}^{ABS, BE}$  is the exterior of  $B_{(1, 0)}(1)$ , or the interior of  $\mathbb{C} \setminus B_{(1, 0)}(1)$  in  $\mathbb{C}$ .

*Note: this means that BE is absolutely stable for any  $h$  when  $\lambda \leq 0$ . Also, if  $\lambda > 0$ , the method is not absolutely stable as  $h \rightarrow 0$ .*

# Plotting stability regions; single-step methods

- 1 Write the scheme in the form  $U^{n+1} = R(z)U^n$ .

*$R(z)$  is called the growth factor or stability function*

- 2 The boundary of  $\mathcal{R}^{ABS}$  is the set of  $z$ :  $|R(z)| = 1$ .

- 3 Plot the contour of  $|R(z)|$  in MATLAB

```
%% Set-up grid for contour plotting in  $\mathbb{R}^2 = \mathbb{C}$ 
[xx,yy]=meshgrid(linspace(-2,2),linspace(-2,2));
%% Find the formula for the growth factor  $R(z)$ 
%% Example: FE  $U^{n+1} = U^n + zU^n$ ;  $R(z) = 1+z$ 
>> growth = @(z)((1+z));
>> contourf(xx,yy,abs(growth(xx+1i*yy)),[1,1]);axis([-2 2 -2 2]);grid on
```

# Finding the region $z \in \mathcal{R}^{ABS}$ for LMM

## Algorithm

- 1 Identify the polynomials  $\rho(\xi), \sigma(\xi), \pi(z; \xi)$  for the scheme.
- 2 Solve for  $\xi$  the equation  $\pi(z; \xi) = 0$ .
- 3 The roots  $\xi(z)$  depend on  $z$ .
- 4 For single roots, solve the inequality  $|\xi(z)| \leq 1$ : identify  $z$  for which this holds.
- 5 For repeated roots, make sure they satisfy  $|\xi(z)| < 1$ .

Mark on  $\mathbb{C}$  “all”  $z$  in  $\mathcal{R}^{ABS}$ .

In practice, you want to plot the boundary of  $\mathcal{R}^{ABS}$ . In MATLAB, this can be done in two ways described below.

## Remark 11

It is “safest” for  $z = \lambda h$  to be in the interior of  $\mathcal{R}^{ABS}$ .

# Plotting (boundary of) stability regions, LMM

- ➊ Identify the polynomials  $\rho(\xi), \sigma(\xi), \pi(z; \xi)$  for the scheme
- ➋ The boundary of  $\mathcal{R}^{ABS}$  is the set of roots  $\xi$  of  $\pi(z; \xi)$  for which  $|\xi| = 1$ . The points  $\xi = e^{i\theta}$  lie on unit circle.
- ➌ Solve for  $z$  the equation  $\pi(z; e^{i\theta}) = 0$ .
- ➍ Plot the curve of  $z$  (the boundary of  $\mathcal{R}^{ABS}$ ) in MATLAB

```
%% Set-up points on unit circle.
```

```
>> theta = linspace(0,2*pi); xi = exp(1i*theta);
```

```
%% (Learn how to plot in C two ways)
```

```
>> plot(xi); pause; plot(real(xi),imag(xi))
```

```
%% solve for z for which roots of pi(z;xi)=0 are on the unit circle
```

```
%% Example: FE rho(xi)=xi-1, sigma(xi)= 1, pi(z;xi)=xi-1-z=0.
```

```
%% Solve pi(z,xi)=0 for z in terms of xi. Obtain z=xi-1
```

```
>> mybd = @(xi)(xi-1); z = mybd(xi);
```

```
>> plot(z,'k');fill(real(z),imag(z),'blue');axis([ -3 3 -3 3]);grid on
```

```
%% add another contour to indicate properly the interior of  $\mathcal{R}^{ABS}$ 
```

```
>> z2=mybd(2*xi); hold on; plot(z2,'r');
```

*The algorithm fills the inside of the contour, not necessarily the interior of  $\mathcal{R}^{ABS}$ . z2 helps.*

Practice with BE, Trapezoidal method, AB-\*, AM-\*



# Order stars, single step scheme for $u' = \lambda u$ ,

Here we consider again a single step scheme  $U^{n+1} = R(z)U^n$ .  
The stability factor  $R(z)$  provides an approximation to  $e^z$ .

*Plotting contours of  $R(z)e^{-z}$  (order stars) gives an idea how well  $R(z) \approx e^z$ . We have  $R(z)e^{-z} = 1 + r(z)$ , where  $r(z)$  measures the discrepancy. One can show that for a method of order  $p$ ,  $r(z) \approx O(z^{p+1})$ .*

*Plotting the contour 1 of  $e^{-z}R(z)$  we expect to see it produce  $p+1$  wedges “inside” and  $p+1$  wedges “outside”. (This is the behavior of the function  $1 + Cz^{p+1}$  near the origin; its contours are made of  $2(p+1)$  wedges.)*

*The plot is known as “order-star” since it allows to identify the order of the method by counting the wedges.*

```
%% Recall plotting in R^2=C
[xx,yy]=meshgrid(linspace(-2,2),linspace(-2,2)); z=xx+1i*yy;
%% Use the growth factor R(z)=1+z+z^2/2 (order p=2 of accuracy)
>> growth = @(z)((1+z+z.^2/2));
>> contourf(xx,yy,abs(growth(z).*exp(-z)),[1,1]);axis([-2 2 -2 2]);grid on
%% Compare to the polynomial 1+Cz^{p+1}, C=-4 is rather arbitrary
>> contourf(xx,yy,abs(1-4*z.^3),[1,1]);axis([-2 2 -2 2]);grid on
%% You should have seen three wedges in both plots!
```

# How to choose a method, and how to choose $h$

- 1 Plotting absolute stability regions is an interesting activity, but it may be hard from the plot alone to check (for a given  $\lambda$ ), if  $z = \lambda h \in \mathcal{R}^{ABS}$ .
- 2 We do **not** have a “choice” of  $\lambda$ , since it comes from the problem we are solving. But we can (i) choose a method which **can** be absolutely stable for this  $\lambda$ , and (ii) select an appropriate  $h$  so that  $z = h\lambda \in \mathcal{R}^{ABS}$  for this method.
- 3 For **nonlinear** problems, we select the method as in (2) based on the sign of  $f_u$  and the magnitude of  $f_u$  which plays the same role as  $\lambda$ . (think of linearizing  $f(t, u) \approx f(t, u^*) + f_u(t, u^*)(u - u^*)$ , where  $u^*$  is some conveniently chosen point, typically, an equilibrium).
- 4 For linear **systems**  $u' = A(t)u + B(t)$ , we have to consider all the eigenvalues of  $A(t)$  in place of  $\lambda$ .
- 5 For nonlinear systems, we combine (3) and (4).

---

*Sometimes  $|\lambda|$  is called the time-scale. If  $|\lambda|$  is large, system has fast transients.*

*If  $|\lambda|$  is small, evolution is slow. A system can have multiple time scales.*

## Examples: linear and linearized problems

Recall slide 6:  $|u(t)|$  increases at worst exponentially with  $O(e^{L_f t})$ . To get more information for a nonlinear  $f$ , we try to linearize about an equilibrium  $u^*(t)$  where  $f(t, u^*) = 0$ ; we use  $\lambda = f_u(t, u^*)$ .

*Sometimes, however, this does not work.*

### Example 6.1 (Linear scalar pbm)

$u' = -30u$ ; As  $t \rightarrow \infty$   $|u(t)| \rightarrow 0$ ; asymptotically stable; fast decay

$u' = -0.01u$ ; Asymptotically stable; slow decay

$u' = -tu, t > 0$ ; Here  $\lambda < 0$ . Stable; decay slow near 0, decay accelerates as  $t \uparrow$ .

$u' = 3u$ ; clearly  $|u(t)|$  increases exponentially,  $\lambda = 3$ .

### Example 6.2 (Nonlinear scalar $f$ ; linearization useful or not)

$u' = -3u^2, u(0) > 0; u^* = 0$ ; Here  $\lambda = -6u^* = 0$ . For any  $u(0) > 0$ ,

$\lambda = -6u(0) < 0$ ; with  $f(u) \leq 0$ ; pbm is stable. However, with  $u(0) < 0$  pbm is unstable with blow-up.

$u' = u(1 - u)$ ; Two equilibria  $u_{1,2}^* = 0, 1$ ;  $\lambda_{1,2} = 1 - 2u^*$ . Stable with  $\lambda_2 < 0$  but unstable with  $\lambda_1 = 1 > 0$ .

$u' = -e^u, u(0) > 0$ ; No equilibria. Linearization about any point gives  $\lambda < 0$ ; solution decreases but  $|u(t)|$  grows logarithmically.

$u' = \frac{1}{1+u^2}$ ; No equilibria. Increasing solution even though linearization predicts  $\lambda < 0$  and  $|\lambda| \leq 1$  (Linearization underpredicts growth).

$u' = e^{-u}, u(0) > 0$ ; No equilibria. Logarithmic growth of  $|u(t)|$  even though  $\lambda < 0$ .

## Systems $u' = f(t, u)$ , $f(t, u) \in \mathbb{R}^d$

In this class we assume the matrix  $Df \in \mathbb{R}^{d \times d}$  system is always diagonalizable, i.e., that there is a basis for  $\mathbb{R}^d$  made of the eigenvectors of  $Df$ .

**Solving a linear constant coefficient system of ODEs  $u' = Au$  [RJL, Sec.7.4.2]**

- diagonalize  $A = R\Lambda R^{-1}$ ;
- $v = R^{-1}u$  is the new coordinate system;
- solve the diagonal system  $v' = \Lambda v$ ; (find the general solution)
- go back to  $u = Rv$ ;
- apply the I.C. to  $u(t)$ .

**Solving variable coefficient systems when  $A = A(t)$  or a nonlinear system “by hand” is usually not possible. For numerical schemes it is important to identify how the eigenvalues  $\lambda_j$  calculated near equilibria depend on  $u$ .**

# Examples of systems

## Example 6.3 (Diagonal system $u' = Au$ )

Let  $A = \begin{bmatrix} 1 & 0 \\ 0 & -10 \end{bmatrix}$ . We have  $\Lambda = A$ ;  $\lambda_1 = 1$ ;  $\lambda_2 = -10$ ; eigenvectors are the columns of  $R = I$ . Compute  $u_j(t) = v_j(t) = C_j e^{\lambda_j t}$ . Apply I.C. directly.

## Example 6.4 (Non-diagonal system $u' = Au$ )

Let  $A = \begin{bmatrix} 2 & 1 \\ 1 & 2 \end{bmatrix}$ . The eigenvalues are  $\lambda_1 = 1$ ;  $\lambda_2 = 3$ , and the eigenvectors are  $r_1 = [1, -1]^T$ , and  $r_2 = [1, 1]^T$ . (Normalize them to get an easy  $R$ ). Compute  $v_j(t) = C_j e^{\lambda_j t}$ ; calculate  $u(t) = Rv(t)$ . Apply I.C.

## Example 6.5 (Linearized pendulum system $u' = Au$ )

Let  $A = \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix}$ . The eigenvalues are  $\lambda_{1,2} = \pm i$ , and the eigenvectors are  $r_1 = [1, i]^T$ , and  $r_2 = [1, -i]^T$ . (Normalize them to get an easy  $R$ ). Compute  $v_j(t) = C_j e^{\lambda_j t}$ ; calculate  $u(t) = Rv(t)$ . Apply I.C. Verify the solution is the same as by solving  $x'' + x = 0$ .

# Examples of nonlinear systems

## Example 6.6 (Particle and its velocity in a double-well potential field)

Consider  $x' = y; y' = x - x^3$ . Compute the eigenvalues of the Jacobian at the equilibria  $(0, 0)$  and  $(\pm 1, 0)$  to find that the eigenvalues have different signs. As for behavior, the system is conservative, and  $E(x, y) = \frac{1}{2}y^2 - \frac{1}{2}x^2 + \frac{1}{4}x^4 = \text{const.}$

## Example 6.7 (Rabbits $x(t)$ vs sheep $y(t)$ : Lotka-Volterra competition)

We study  $x' = x(3 - x - 2y); y' = y(2 - x - y)$ . There are multiple equilibria, and many eigenvalues to consider.

## Example 6.8 (van der Pol system)

We consider  $\mu > 0$  and  $x'' + \mu(x^2 - 1)x' + x$  rewritten as a system. Here the system can be seen as a modification of the pendulum problem with a damping term  $\mu(x^2 - 1)x'$ .

---

*What numerical method would you choose for these problems? What  $h$ ?*

## Stiff problems; stiff systems

We call a problem “stiff” if  $|\lambda|$  is large. A system is also called “stiff” if the eigenvalues differ much in magnitude.

*A problem (scalar or a system) is stiff if there are large differences between the time scales in the problem.*

### Example 6.9 (Scalar)

Consider  $u' = \lambda(u - p(t)) + q(t)$  where  $p(\cdot), q(\cdot)$  are bounded functions, and when  $\lambda < 0$  with very large  $|\lambda|$ . This ODE is called “kinetic” (or a relaxation ODE) when  $q = 0$ , and has solutions that tend to the “equilibrium”  $p(t)$  very quickly, in  $T \approx O(\frac{1}{\lambda})$ .

For example, choose  $q(t) = 0, p(t) = \frac{5t}{1+t}$ .

See also [RJL, Ex.8.1] with  $p(t) = \cos(t), q(t) = \sin(t)$ .

*Since  $\lambda < 0$  and  $|\lambda|$  is large, we want to find a method for which  $R^{ABS}$  contains much of the left complex plane so that  $h$  is not too restricted. For example, for BE, no restriction on  $h$  is needed.*

### Example 6.10 (System $u' = Au$ )

Let  $A = \begin{bmatrix} 1 & 0 \\ 0 & -1000 \end{bmatrix}$ . The system ... is stiff. The eigenvalues are 1,  $-1000$ .

*When using FE, we would want to use  $h < \frac{2}{1000}$  due to stiff decay of the second component, while the first component is expected to mildly grow.*

Play with these examples: plot their numerical solutions.

# Choosing a method for dissipative problems.

## A-stability and L-stability

### Definition 6.11 (A-stability)

... holds if  $\mathcal{R}^{ABS}$  contains all of left plane.

### Remark 12

*We have seen that  $\mathcal{R}^{ABS, BE}$  is large, and contains all of left plane, i.e., the error in BE is easily controlled for any  $h$  when  $\Re \lambda < 0$ . BE is A-stable. Trapezoidal method shares this property.*

Recall the single step stability (growth) factor  $R(z)$ .

### Definition 6.12 (L-stability)

... holds if  $\lim_{z \rightarrow \infty} |R(z)| = 0$ .

### Example 6.13

Consider BDF methods such as Ex. 3.10, 4.5; they are L-stable.



## Continued examples

### Example 6.14 (Chemical reactions; see [RJL, Sec. 7.4.1] for intro)

The system below is known as the Robertson system

$$y_1' = -\alpha y_1 + \beta y_2 y_3; \quad y_2' = \alpha y_1 - \beta y_2 y_3 - \gamma y_2^2; \quad y_3' = \gamma y_2^2.$$

Values  $\alpha = 0.04, \beta = 10^4, \gamma = 3 \cdot 10^7$  correspond to slow, fast, and very fast reactions, and show this is a very stiff system. Start with  $y(0) = [1, 0, 0]^T$ .

### Example 6.15 ((Modified) Kepler problem, Hamiltonian system)

Consider Hamiltonian (total energy functional)  $H(q, p) = \frac{1}{2}(p_1^2 + p_2^2) - \frac{1}{r} - \frac{0.01}{r^3}$ , with  $r = \sqrt{q_1^2 + q_2^2}$ , and  $q' = H_p, p' = -H_q$ . Here  $p, q$  denote velocity and position components, respectively. One can show  $H|_{t=T} = H|_{t=0}$  for any  $T$ . Numerical solution is very noisy, even with higher order or A-stable methods.

### Example 6.16 (Lambert example)

$$\begin{aligned} u(t) &= e^{0.1t} \sin(8t) + e^{-50t}, \quad v(t) = e^{0.1t} \cos(8t) + e^{-50t}, \\ w(t) &= e^{0.1t} (\cos(8t) + \sin(8t)) + e^{-50t}. \end{aligned}$$

with a matrix with eigenvalues  $0.1 \pm 8i$ , and  $-50$ . Starting at  $T_0 = \pi/8$ , interesting dynamics unfolds. An A-stable numerical method (BE or Trapz) may damp the oscillations too much, if  $h$  is too large.

# Examples from spatially discretized PDEs.

## Damping frequently desirable; but not always.

**Example 6.17 (Diffusion system  $u' = Au$ , with Dirichlet b.c.)**

Let  $A = - \begin{bmatrix} 2 & -1 \\ -1 & 2 \end{bmatrix}$  with moderate eigenvalues  $-1, -3$ . System is not stiff, but can be generalized to  $A \in \mathbb{R}^{d \times d}$  (sparse, with 2 on the diagonal, and  $-1$  in sub- and super-diagonals). This corresponds to the spatially discretized diffusion equation, with ratio of the eigenvalues increasing as  $O(d^2)$ .  

---

*L-stable schemes are desirable. (BE is a good choice).*

**Example 6.18 (“Advection” system  $u' = Au$ , with periodic b.c.)**

Let  $A = - \begin{bmatrix} 1 & 0 & -1 \\ -1 & 1 & 0 \\ 0 & -1 & 1 \end{bmatrix}$ . Matrix  $A$  is singular, and the two complex eigenvalues have  $\Re \lambda < 0$ . If you generalize to  $d \times d$ , one always has  $\lambda_1 = 0$ ; most of the other  $d-1$  eigenvalues have nontrivial  $\Im$  part.  

---

*L-stable schemes are not necessarily desirable because they will damp the oscillatory behavior of the solutions to the problem. (FE is preferable over BE.)*

# How to find the “right” method & time step $h$

For a given problem  $u' = f(t, u)$  we know  $\|f\|_\infty$ , and  $L_f$ , and the “sign” of (the eigenvalues of)  $\lambda$ . For scalar problems the sign of  $f$  matters; for systems, it does not.

## Choose a method:

zero-stable, consistent, with a reasonably large region  $\mathcal{R}^{ABS}$  for typical  $\lambda$ .

## Choose time step $h$ :

- ❶  $h$  small enough for stability so that  $z \in \mathcal{R}^{ABS}$
- ❷  $h$  small enough for accuracy (the truncation error  $\rightarrow$  global error)  
*Control the truncation error to some tolerance*
  - use a-posteriori error control, since a-priori may be too pessimistic
- ❸  $h$  large enough to have an efficient scheme:  
*Very small  $h$  requires many time steps*
  - it takes too long to solve a problem
  - for complicated  $f$ , additional round-off error may accumulate

# A-posteriori error and time step control

We can try to estimate (without knowing the true solution  $u(t)$ )

- the local (one-step) truncation error ( $\tau$ =LTE) in each step,
  - with embedded RK, and/or predictor-corrector pairs
- the global error (accumulated LTE)
  - with Richardson extrapolation

**The local estimate for  $\tau$  helps to control the time step  $h$  by requiring, e.g.,  $\tau \leq tol$ .**

**If  $\tau$  is too large, we cut  $h$ .**

**If  $\tau$  is very small, we can increase  $h$ .**

## Local error estimates

### Remark 13 (Estimate $\tau$ with RK embedded (Fehlberg) methods)

*Calculate  $\hat{U}^{n+1}$  using  $r$  stages and  $U^{n+1}$  using  $r + 1$  stages, evaluating  $f$  at essentially the same stages  $Y_j$  for both methods. Use the higher order  $U^{n+1}$  to estimate  $\tau$  in  $\hat{U}^{n+1}$ .*

### Example 7.1 (Heun method (17) works as embedded method)

Calculate  $\hat{U}^{n+1} - U^{n+1} = \dots \approx U^n + \frac{h^2}{2} f(U^n) f'(U^n) \approx h\tau_n$ ;  $\tau_n$  is LTE for FE.  
The famous Dormand-Price  $r = 4$ , 4/5th order pair is implemented in `ode45`.

### Remark 14 (Estimate $\tau$ with predictor-corrector pairs)

*Calculate  $\hat{U}^{n+1}$  in a predictor step, and  $U^{n+1}$  in a corrector step. Estimate the error in predictor step from  $U^{n+1}$ ,  $\hat{U}^{n+1}$ .*

### Example 7.2 (Heun method (17) works as predictor-corrector)

Calculation of  $\hat{U}^{n+1}$  is a predictor step. The corrector step (higher order implicit method, trapezoidal), replaces the “implicit” computation by the predicted value.  
This is also the pair of AB-1 with AM-1; other Adams methods work similarly

## Code for local error estimate for Ex. 7.1

```
...
fprintf('DEMONSTRATING estimates of LTE\n');
prevsol = y0;
for n=1:5
    du = f (h*n, prevsol);
    nsolhat = prevsol + h * du;
    FEsol = nsolhat;
    %% calculate Heun estimate
    du2 = f(h*(n+1),nsolhat);
    HEUNsol = prevsol + h/2*(du+du2);
    %% LTE estimate, plus errors
    fprintf('Step %d time =%g h*LTE(true)=%g h*LTE estimate=%g\n',...
        n,h*n,d2exact(h*(n-1))*h^2/2,HEUNsol-FEsol);
    %% save FE solution
    prevsol = FEsol;
end
```

User must code the functions  $f(t, u)$  in function `f(t,u)`, and  $u''(t)$  in `d2exact`.

# Global error estimates, by Richardson extrapolation; recall slide 25

Let  $I$  be the true quantity, and  $I_h$  its approximation dependent on step  $h$ . For example,  $I = u(T)$ , and  $I_h = U^N$  when  $T = Nh$ .

Consider a first order method, with  $I - I_h \approx Ch + O(h^2)$ .

For this case,  $2I_h - I_{2h} = I + O(h^2)$ , so  $2I_h - I_{2h}$  provides a second order approximation to  $I$ .

An estimate for the error  $I - I_h \approx I_h - I_{2h} + O(h^2)$  is available.

## Example 7.3 (Global error estimate for FE with time step doubling)

Choose  $h$  so that  $T/2h = J$  is an integer.

Consider time steps  $0 = t_0, t_1, \dots, t_N = T$  with  $t_n = nh$ .

Define also the double-time grid  $0 = \bar{t}_0, \bar{t}_1, \dots, \bar{t}_J = T$ , with  $\bar{t}_j = j2h$ .

Note  $T = J2h = Nh$ ,  $N = 2J$ , and each  $t_{2j} = \bar{t}_j$ ; we expect  $U^{2j} \approx \bar{U}^j$ .

*Numbering may be different in your code*

Find  $U^n$ ,  $n = 1, \dots, N$ , on the single-time grid, with FE.

Find  $\bar{U}^j$ ,  $j = 1, \dots, J$ , on the double-time grid, with FE.

Compare the true error  $u(T) - U^N$  to its Richardson's estimate  $U^N - \bar{U}^J$ .

You can repeat at every time step  $n = 2j$ : compare  $u(\bar{t}_j) - U^{2j}$  to  $U^{2j} - \bar{U}^j$ .

## Code for global error estimate for Ex. 7.3

This code assumes that you have coded a function `euler_explicit`; in addition, you need `f`, and `exact`.

```
%%%%%%%% demonstrate estimates of global error
...
%% the numerical solution is in y1 and y2 (double step)
[t1,y1] = euler_explicit (y0,0,1,h);
[t2,y2] = euler_explicit (y0,0,1,2*h);
te = linspace(0,t1(end),size(t1,2)*10);ue = exact(te);
%% plot
plot(te,ue(1,:), 'k', t1,y1(:,1), 'r*-', t2,y2(:,1), 'bo-');
legend('exact', 'step h', 'step 2h');
%% show error and its estimate
for n=1:length(t2)
    fprintf('ex_error=%g err_estimate (Richardson)=%g \n', ...
        exact(t2(n))-y1(2*n-1), y1(2*n-1) - y2(n) );
end
```



# Recap: FD for IVPs

## Review concepts

- Analyze the problem itself  $u' = f(t, u)$ 
  - (\*) What is the behavior of solutions? What are the characteristic  $\lambda$ 's for your problem?
- (\*\*) What method would you choose based on (\*)? (single step, single-step multi-stage, multi-step, ...)
  - Is the method zero-stable?
  - How much do you care about preserving qualitative character?
  - How much do you care about the accuracy?
  - How expensive is the method ?
- (\*\*\*) What  $h$  would you choose for the method (\*\*) ?
  - Stability ? (stay in  $\mathcal{R}^{abs}$  if possible)
  - Accuracy ? (run the code with some simple methods for your problem to get a sense of LTE and global error)
  - How many time steps/stages do you need with this  $h$ ? How many evaluations of  $f$  ?

## BVP: boundary value problems

So far we have studied first order IVP  $u' = f(t, u), u(0) = u_0$ .

A second order IVP

$$u'' = f(t, u, u'), u(0) = u_0, u'(0) = u_1, \quad (26)$$

can be converted to (and analyzed as) an equivalent first order system of ODEs

$$u' = v, v' = f(t, u, v); \quad u(0) = u_0, v(0) = u_1.$$

**An important BVP class of second order ODEs requires boundary rather than initial conditions.**

$$u'' = f(t, u, u'), u(0) = u_0, u(1) = u_1. \quad (27)$$

The theory of well-posedness for BVPs is much more complicated (and more limited) than that for IVPs. We will only study some selected (classes of) problems.

# Boundary conditions

## Conditions in linear two-point constant coefficient BVP

$$u'' + mu' + nu = f(t), \quad t \in [a, b]; u(a) = u_a, u(b) = u_b \quad (28)$$

Condition  $u(a) = u_a$  is known as Dirichlet condition,  $u'(a) = u_b$  is known as Neumann condition,  $u'(a) + Cu(a) = u_c$  is known as the Robin condition. One can also impose periodic conditions, e.g.,  $u(a) = u(b), u'(a) = u'(b)$ . If the data equals 0, we call the BC homogeneous.

### Example 8.1 (Existence and uniqueness of solutions depending on BC)

Consider  $t \in [0, 1]$ . Find the general solution and determine if the solution exists and if it is unique.

$$u'' = 0, u(0) = 0, u(1) = 0$$

$$u'' = 1, u(0) = 0, u(1) = 0$$

$$u'' = 1, u'(0) = 0, u'(1) = 0$$

$$u'' = 1, u(0) = 0, u'(1) = 0$$

$$u'' = 1, u'(0) + u(0) = 0, u'(1) = 0$$

$$u'' = 1, u(0) = u(1), u'(0) = u'(1)$$

$$u'' = u, u(0) = 0, u(1) = 0$$

$$u'' = u, u(0) = 0, u'(1) = 0$$

$$u'' = -u, u(0) = 0, u(1) = 0$$

# Examples of two-point BVP

## Example 8.2 (Pendulum: when initial and final position are known)

Recall the pendulum problem  $\theta'' + \theta = 0$  which we set up as IVP when the initial position  $\theta(0)$  and initial velocity  $\theta'(0)$  are known.

The problem is a BVP if, e.g., initial and final positions are known  $\theta(0) = \theta_0$ , and  $\theta(T) = \theta_1$ . Instead, one could know  $\theta'(0)$  and  $\theta(T)$ .

## Example 8.3 (Equilibrium problem in elastic rod attached to walls: (generalizes a spring-mass system) attached to walls)

Here  $u$  denotes longitudinal displacement in an elastic rod from its equilibrium due to external force  $f$  (gravity), balanced by internal elasticity forces according to Hooke's law  $y = ce$ ,  $e = k \frac{du}{dx}$ , and by Newton's  $f + \frac{d}{dx}y = 0$ . Thus  $-\frac{d}{dx}(k \frac{du}{dx}) = f$ . BC reflect (i) no displacement at the ends of the rod  $u(0) = 0$ ,  $u(L) = 0$ , or (ii) right end unattached  $u'(L) = 0$ .

## Example 8.4 (Stationary heat conduction)

Here  $u$  denotes temperature in a rod. Heat flux  $q = -k \frac{du}{dx}$  is proportional to the gradient of temperature, and energy conservation requires  $\frac{dq}{dx} = f$ , with external heat sources given by  $f$ .

BC mean: (i) fixed temperature at two ends  $u(0) = u_0$ ,  $u(L) = u_L$ . (ii) If rod is insulated on left end, we have  $ku'(0) = 0$ . (iii) If flux on right end is proportional to the difference with ambient temperature  $u_*$ , we have  $ku'(L) + a(u(L) - u_*) = 0$ .

# Numerical solution of two-point BVP (28)

- ➊ Define a grid on  $[a, b]$ . Easiest is uniform grid  $a = t_0, t_1, \dots, t_M = b$ . Here  $h = \frac{b-a}{M}$ , and  $t_j = a + jh$ .
- ➋ Discretize. (Replace  $u''(t)$  by  $D_h^2 u(t) = \frac{u(t+h) - 2u(t) + u(t-h)}{h^2}$ , and other derivatives by difference quotients.)
- ➌ Write the discrete equation to be satisfied by  $U^{j-1}, U^j, U^{j+1}$  at every interior node  $t_j$
- ➍ Apply BC to  $U^0$  and  $U^M$
- ➎ Collect all unknowns in  $U = [U^0, U^1, \dots, U^{M-1}, U^M]^T$ , and right-hand side in  $F = [f_0, f_1, f_2, \dots, f_{M-1}, f_M]^T$ . The entries  $f_0, f_M$  will be given from boundary conditions.
- ➏ Identify the coefficients in steps 2,3 as components of a matrix  $A$
- ➐ Solve the linear system  $AU = F$

## Remark 15 (MATLAB numbering)

*The nodes and unknowns above are numbered from  $0, 1, \dots, M$ . MATLAB does not allow indexing from 0, so the unknowns in your code will likely be numbered  $U^1, \dots, U^{M+1}$ . The principles remain the same.*

## Example: $-u'' = f$ on $[a, b]$ with Dirichlet BC $u(a) = u_a, u(b) = u_b$

### Example 8.5 (Derive and solve the discrete system of equations)

Decide on  $M$  and  $h, h = \frac{b-a}{M}$ .

Equation for internal nodes and boundary conditions

$$\text{Interior nodes : } \quad \frac{1}{h^2}(2U^j - U^{j-1} - U^{j+1}) = f(t_j), \quad \forall j = 1, \dots, M-1 \quad (29a)$$

$$\text{Boundary nodes : } \quad U^0 = u_a, \quad U^M = u_b. \quad (29b)$$

Set-up system of equations  $AU = F$  with

$$A = \frac{1}{h^2} \begin{bmatrix} 1 & 0 & \dots & 0 \\ -1 & 2 & -1 & \dots \\ \dots & & & \\ \dots & & -1 & 2 & -1 \\ \dots & & & & 1 \end{bmatrix}, \quad F = \begin{bmatrix} u_a \\ f(t_1) \\ \dots \\ f(t_{M-1}) \\ u_b \end{bmatrix}. \quad (29c)$$

Solve linear system  $AU = F$ . In MATLAB, write  $U=A \backslash F$

---

*Note the first and last rows in (29c) correspond to BC, and could be trivially eliminated. The numbering above is not MATLAB-like.*

# Code for BVP from slide 70

## Numbering, again (recall Remark 15)

When coding, the interior nodes will be numbered  $2, \dots, M$ , and boundary nodes will be numbered 1 (left boundary) and  $M + 1$  (right boundary). In particular, when we apply the BC, we write  $U(1) = u_a; U(M + 1) = u_b$ , for the first and last entries of  $U$ . .

```
dx = (b-a)/(M); x = (a:dx:b)'; %% use uniform grid, x is numbered from 1...M+1
%% set up the matrix and the right hand side vector
f = rhsfun(x);
A = sparse(M+1,M+1);
for j=2:M
    %% set up j'th internal row of the matrix
    A(j,j) = 2; A(j,j-1) = -1; A(j,j+1) = -1;
end
A = A / (dx*dx);
%% apply the Dirichlet b.c. to the matrix and right hand side
A (1,1) = 1; f (1) = ua;
A (M+1,M+1) = 1; f (M+1) = ub;
%% solve the linear system;
U = A \ f;
```

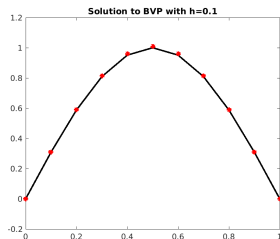
*This code can be vectorized for faster performance*

## Solution to BVP from slide 70

Start assuming an exact solution  $u(t) = \sin(\pi t)$ .

Calculate (manufacture)  $f(t) = \pi^2 \sin(\pi t)$ .

Implement the code and plot ...



| $h$  | $E(h, \infty)$ | $E(h, 2)$ |
|------|----------------|-----------|
| 0.1  | 0.008265       | 0.058     |
| 0.05 | 0.00205871     | 0.0015    |
| 0.01 | 0.0000822508   | 0.0000582 |

Test convergence in  $L^\infty$  and  $L^2$  to see that  $E(h, *) = O(h^2)$  for each norm; see table.

(See below for a precise definition of norms, and convergence analysis).



# Alternative to solving a system: shooting method

Notes from Prof. Faridani TBA.

# Analysis of convergence of FD for BVP

Define now the interior part  $A_h \in \mathbb{R}^{(M-1) \times (M-1)}$  of the matrix  $A$  from (29c).

## Remark 16 (Error analysis)

Writing the error equation we find  $A_h E = \tau$  where  $\tau$  is the vector of truncation errors  $\tau_j$  (truncation errors on the boundary are 0).

Proving convergence

$$\|E\|_* \rightarrow 0, \quad \text{as } h \rightarrow 0 \quad (30)$$

requires proving consistency and stability.

Here  $*$  stands for the functional space in which we want to measure the error.

- Consistency: with  $D_h^2$ , the truncation error  $\tau_j \approx O(h^2)$  if  $u \in C^4$
- Stability: to keep  $\|E\|_* = O(h^2)$ , we require  $\|A_h^{-1}\|_* \leq C$ .

## Choosing norm

We generally like to use  $*$  to be  $L^\infty$  space, but it is easier to get the bound  $\|A_h^{-1}\|_{L^2}$ .

# Vector, matrix, function & grid function norms.

## Remark 17 (Vector norms in $l_p$ spaces)

$$\mathbb{R}^d \ni x, \quad \|x\|_p := \left( \sum_j |x_j|^p \right)^{\frac{1}{p}}, \quad 1 \leq p < \infty. \quad \|x\|_\infty = \max_j |x_j|. \quad (31)$$

## Remark 18 (Norms for functions and grid functions)

For functions  $f : [a, b] \rightarrow \mathbb{R}$  we extend the  $l_p$  vector norms to the norms in Lebesgue spaces  $L^p$  of functions for which the integral below is well defined and finite

$$\|f\|_p = \left( \int_a^b |f(x)|^p dx \right)^{\frac{1}{p}}; \quad 1 \leq p < \infty. \quad \|f\|_\infty = \text{esssup}_{x \in [a, b]} |f(x)|. \quad (32)$$

For grid functions (sampled on a grid) with  $f(x_j), j = 1, \dots, M$ , and  $1 \leq p < \infty$ , we approximate these integrals with Riemann sums

$$\|f\|_p = \left( \sum_j h |f(x_j)|^p \right)^{\frac{1}{p}} = h^{\frac{1}{p}} \| [f(x_1), \dots, f(x_j), \dots, f(x_M)]^T \|_p \quad (33)$$

---

We will not distinguish between the symbols  $\|\cdot\|_p$  as they apply to vectors in  $\mathbb{R}^d$ , functions on  $\mathbb{R}$ , or grid functions. The meaning of a symbol should be clear from its context.

# Matrix norms; stability in the 2-norm

## Remark 19 (Some matrix norms)

$$\mathbb{R}^{d \times d} \ni A, \quad \|A\|_1 = \text{"max abs col sum"} = \max_j \sum_i |a_{ij}| \quad (34)$$

$$\mathbb{R}^{d \times d} \ni A, \quad \|A\|_\infty = \text{"max abs row sum"} = \max_i \sum_j |a_{ij}| \quad (35)$$

The norm  $\|A\|_2 = \sqrt{\rho(A^T A)}$ , where the spectral radius  $\rho(B)$  is the largest to magnitude eigenvalue of matrix  $B$ .

For a symmetric real matrix  $\|A\|_2 = \rho(A)$ .

## Example 8.6

Consider  $A = \begin{bmatrix} 1 & 0 \\ 0 & -10 \end{bmatrix}$ . We have  $\|A\|_1 = \|A\|_\infty = \|A\|_2 = 10$ .

Consider  $A = \begin{bmatrix} 1 & -1 \\ -1 & 2 \end{bmatrix}$ . We have  $\|A\|_1 = \|A\|_\infty = 3$ ;

$\|A\|_2 = \frac{3+\sqrt{5}}{2} \approx 2.62 = \rho(A)$ .

Let  $A = \begin{bmatrix} 1 & -1 \\ 2 & 2 \end{bmatrix}$ . We have  $\|A\|_1 = 3, \|A\|_\infty = 2, \|A\|_2 = \sqrt{3 + \sqrt{5}} \approx 2.28$ , while  $\rho(A) = 2$ .

# Stability in the 2-norm

## Information from the analysis of BVP

One can show that the eigenfunctions of the differential operator  $-\frac{d^2}{dx^2}$  with Dirichlet BC, on  $[0, 1]$ , are  $u_j(x) = \sin(\pi j x)$ , with the corresponding eigenvalues  $\Lambda_j = (j\pi)^2$ .

*Convince yourself that this is true, and check  $-u_j'' = \Lambda_j u_j$ .*

## Example 8.7 (Continue Ex. 8.5; $A = A^T$ ; estimate eigenvalues of $A^{-1}$ )

In the discrete setting, an approximation to  $-\frac{d^2}{dx^2}$  (with Dirichlet BC) is the interior part  $A_h \in \mathbb{R}^{(M-1) \times (M-1)}$  of the matrix  $A$  from (29c).

One can show that the eigenvectors  $U_1, U_2, \dots, U_{M-1}$  of  $A_h$  in (29c) have entries  $(U_j)_k = \sin(\pi j x_k)$ . Plugging in  $A_h U_j = \lambda_j U_j$  we get  $\lambda_j = \frac{2}{h^2}(1 - \cos(j\pi h)) \approx \Lambda_j$ . Now we want to find  $\lambda_j$  closest to 0 (which is for  $j = 1$ ), because this would give the largest to magnitude  $\lambda_j^{-1}$ , i.e.,  $\rho(A_h^{-1}) = \|A_h^{-1}\|_2$  (since  $A_h$  is symmetric).

A bit of (pre-)calculus shows  $1 - \cos(\pi h) = 2 \sin^2(\frac{\pi h}{2}) = O(h^2)$  for small  $h$ . Thus

$$\|A_h^{-1}\|_2 = \rho(A_h^{-1}) \leq C, \quad (36)$$

with a constant  $C$  independent of  $h$ , and we have stability in  $L^2$ .

## Stability in the $\infty$ -norm

### Further information from analysis of BVP ...

One can show that the solution  $u(t)$  to the BVP is given as

$$u(t) = \int_0^1 G(t, s) f(s) ds,$$

where  $G(t, s)$  is the Green's function for the problem. This function is piecewise linear in  $s$ , with a jump of derivative equal 1 at  $s = t$ .

We have  $G(x, y) = (1 - y)x$ , for  $x \in [0, y]$ , and  $y(1 - x)$  for  $x \in [y, 1]$ .

**Example 8.8 (Continue Ex. 8.5, get an estimate of  $\|A_h^{-1}\|_\infty$ )**

One can write the numerical solution

$$U^j = h \sum_i f(x_i) G(x_i, x_j) \quad (37)$$

In other words, the matrix  $(hG(x_i, x_j))_{ij} = A_h^{-1}$ .

To calculate  $\|A_h^{-1}\|_\infty$ , we need to estimate the (abs) row sum of  $A_h^{-1}$ , or estimate it from above by the integral  $\int_0^1 G(t, s) ds$ ,  $\forall t$ .

These can be shown to be bounded by a constant independent of  $h$ .

Thus we have stability in  $L^\infty$ .

# Wrap-up BVPs

## What we have covered

- Basic methods
  - shooting: exploits IVP methods, and iteration
  - direct BVP solvers: use BC, solve linear system
- Error analysis for direct BVP solver: LTE analysis & stability (bounds for  $\|A^{-1}\|_*$ )
- Error  $\|E\|_* = O(h^p)$ , same as of LTE

## What we have not covered (deferred to 4/553)

- BC other than Dirichlet
- higher order methods
- non-uniform grids
- How to extend to  $-\nabla \cdot (K \nabla u) = f$
- How to treat IBVP for  $u_t - u_{xx} = f$

# The End