Numerical Analysis III: differential equations

Victor Eijkhout, Karl Schulz

Numerical treatment of differential equations

Initial value problem: $u'(t) = f(t, u(t)), \qquad u(0) = u_0, \qquad t > 0$

Boundary value problem:
$u''(x) = f(x), \qquad x(0) = x_0, x(1) = x_1, \qquad x \in [0, 1]$

General assumption: f has higher derivatives.

IVP stability: solutions corresponding to different u_0 values converge as $t \to \infty$. Criterium:

$$\frac{\partial}{\partial u} f(t, u) = \begin{cases} > 0 & \textit{unstable} \\ = 0 & \textit{neutrallystable} \\ < 0 & \textit{stable} \end{cases}$$

Simple example: $f(t, u) = -\lambda u$, then $u(t) = u_0 e^{-\lambda t}$;
stable if $\lambda > 0$

Finite difference approximation

We turn the continuous problem into a discrete one, by looking at finite time/space steps.

Assume all functions are sufficiently smooth, and use Taylor series:

$$u(t+\Delta t) = u(t) + u'(t)\Delta t + u''(t)\frac{\Delta t^2}{2!} + u'''(t)\frac{\Delta t^3}{3!} + \cdots$$

This gives for u':

$$u'(t) = \frac{u(t+\Delta t) - u(t)}{\Delta t} + O(\Delta t^2)$$

So we approximate

$$u'(t) \approx \frac{u(t+\Delta t) - u(t)}{\Delta t}$$

and the "truncation erorr" is $O(\Delta t^2$").

Finite differences 2

How does this help? In $u' = f(t, u)$ substitute

$$u'(t) \to \frac{u(t + \Delta t) - u(t)}{\Delta t}$$

giving

$$\frac{u(t + \Delta t) - u(t)}{\Delta t} = f(t, u(t))$$

or

$$u(t + \Delta t) = u(t) + \Delta t\, f(t, u(t))$$

Let $t_0 = 0$, $t_{k+1} = t_k + \Delta t = \cdots = (k + 1)\Delta t$, $u(t_k) = u_k$:

$$u_{k+1} = u_k + \Delta t\, f(t_k, u_k)$$

'Explicit Euler' or 'Euler forward'.

Does this compute something close to the true solution?
'Discretization error'

Some error analysis

Local Truncation Error: assume computed solution is exact at step k, how wrong will we be at step $k+1$?

$$\begin{aligned} u(t_{k+1}) &= u(t_k) + u'(t_k)\Delta t + u''(t_k)\frac{\Delta t^2}{2!} + \cdots \\ &= u(t_k) + f(t_k, u(t_k))\Delta t + u''(t_k)\frac{\Delta t^2}{2!} + \cdots \\ u_{k+1} &= u_k + f(t_k u_k)\Delta t \end{aligned}$$

So

$$\begin{aligned} L_{k+1} &= u_{k+1} - u(t_{k+1}) = u_k - u(t_k) + f(t_k, u_k) - f(t_k, u(t_k)) - u''(t \\ &= -u''(t_k)\frac{\Delta t^2}{2!} + \cdots \end{aligned}$$

Global error: $E_k \approx \Sigma_k L_k = k\Delta t \frac{\Delta t^2}{2!} = O(\Delta t)$
First order method

An Euler forward example

Consider $f(t,u) = -\lambda u$, exact solution $u(t) = u_0 e^{-\lambda t}$;
stable if $\lambda > 0$

Explicit Euler scheme

$$u_{k+1} = u_k - \Delta t \lambda u_k = (1 - \lambda \Delta t) u_k = (1 - \lambda \Delta t)^k u_0$$

Then

$$\begin{aligned}
& u_k \to 0 \text{ as } k \to \infty \\
\Leftrightarrow \quad & |1 - \lambda \Delta t| < 1 \\
\Leftrightarrow \quad & -1 < 1 - \lambda \Delta t < 1 \\
\Leftrightarrow \quad & -2 < -\lambda \Delta t < 0 \\
\Leftrightarrow \quad & 0 < \lambda \Delta t < 2 \\
\Leftrightarrow \quad & \Delta t < 2/\lambda
\end{aligned}$$

Conditionally stable

Implicit Euler

Or 'Euler backward':

$$u(t-\Delta t) = u(t) - u'(t)\Delta t + u''(t)\frac{\Delta t^2}{2!} + \cdots$$

so

$$u'(t) = \frac{u(t) - u(t-\Delta t)}{\Delta t} + u''(t)\Delta t/2 + \cdots$$

Compute $u'(t) = f(t, u(t))$ as

$$\begin{aligned} & \frac{u(t) - u(t-\Delta t)}{\Delta t} = f(t, u(t)) \\ \Rightarrow \quad & u(t) = u(t-\Delta t) + \Delta t f(t, u(t)) \\ \Rightarrow \quad & u_{k+1} = u_k + \Delta t f(t_{k=1}, u_{k+1} \end{aligned}$$

Implicit equation for u_{k+1}!

Let $f(t,u) = -u^3$, then $u_{k+1} = u_k - \Delta t u_{k+1}^3$
needs nonlinear solver

Stability of Implicit Euler

Again the $f(t,u) = -\lambda u$ example:

$$u_{k+1} u_k - \lambda \Delta t u_{k+1}$$
$$(1 + \Delta t) u_{k+1} = u_k$$
$$u_{k+1} = \left(\frac{1}{1 + \lambda \Delta t} \right) u_k = \left(\frac{1}{1 + \lambda \Delta t} \right)^k u_0$$

If $\lambda > 0$ (stable equation), then $u_k \longrightarrow 0$ for all values of λ and Δt: unconditionally stable.

Pro: larger time steps possible, no worries
Con: implicit equation needs to be solved

Higher order methods

Runge-Kutta
Adams-Bashforth
Trapezoidal rule/Crank-Nicholson

Boundary value problems

Consider $u''(x) = f(x, u, u')$ for $x \in [a, b]$ where $u(a) = u_a$, $u(b) = u_b$

Taylor series (write h for δx):

$$u(x+h) = u(x) + u'(x)h + u''(x)\frac{h^2}{2!} + u'''(x)\frac{h^3}{3!} + u^{(4)}(x)\frac{h^4}{4!} + u^{(5)}(x)\frac{h^5}{5!} + \cdots$$

and

$$u(x-h) = u(x) - u'(x)h + u''(x)\frac{h^2}{2!} - u'''(x)\frac{h^3}{3!} + u^{(4)}(x)\frac{h^4}{4!} - u^{(5)}(x)\frac{h^5}{5!} + \cdots$$

Subtract:

$$u(x+h) + u(x-h) = 2u(x) + u''(x)h^2 + u^{(4)}(x)\frac{h^4}{12} + \cdots$$

so

$$u''(x) = \frac{u(x+h) - 2u(x) + u(x-h)}{h^2} - u^{(4)}(x)\frac{h^4}{12} + \cdots$$

Numerical scheme:

$$\frac{u(x+h) - 2u(x) + u(x-h)}{h^2} = f(x, u(x), u'(x))$$

(2nd order PDEs are very common!)

This leads to linear algebra

$$\frac{u(x+h) - 2u(x) + u(x-h)}{h^2} = f(x, u(x), u'(x))$$

Equally spaced points on $[0, 1]$: $x_k = kh$ where $h = 1/n$, then

$$-u_{k+1} + 2u_k - u_{k-1} = -1/h^2 \, f(x_k, u_k, u'_k) \quad \text{for } k = 1, \ldots, n-1$$

Written as matrix equation:

$$\begin{pmatrix} 2 & -1 & & \\ -1 & 2 & -1 & \\ & \ddots & \ddots & \ddots \end{pmatrix} \begin{pmatrix} u_1 \\ u_2 \\ \vdots \end{pmatrix} = \begin{pmatrix} f_1 + u_0 \\ f_2 \\ \vdots \end{pmatrix}$$

(f linear or not?)

Matrix properties

- Very sparse
- Symmetric (only because 2nd order problem)
- Sign pattern: positive diagonal, nonpositive off-diagonal (true for many second order methods)
- Positive definite (just like the continuous problem)

Initial Boundary value problem

Heat conduction in a rod $T(x,t)$ for $x \in [a,b]$, $t > 0$:

$$\frac{\partial}{\partial t} T(x,t) - \alpha \frac{\partial^2}{\partial x^2} T(x,t) = q(x,t)$$

- Initial condition: $T(x,0) = T_0(x)$
- Boundary conditions: $T(a,t) = T_a(t)$, $T(b,t) = T_b(t)$
- Material property: $\alpha > 0$ is thermal diffusivity
- Forcing function: $q(x,t)$ is externally applied heating.

The equation $u''(x) = f$ above is the steady state.

Discretization

Space discretization: $x_0 = a$, $x_n = b$, $x_{j+1} = x_j + \Delta x$
Time discretiation: $t_0 = 0$, $t_{k+1} = t_k + \Delta t$
Let T_j^k approximate $T(x_j, t_k)$

Space:

$$\frac{\partial}{\partial t} T(x_j, t) = \alpha \frac{T(x_{j-1}, t) - 2T(x_j, t) + T(x_{j+1}, t)}{\Delta x^2} = q(x_j, t)$$

Explicit time stepping:

$$\frac{T_j^{k+1} - T_j^k}{\Delta t} - \alpha \frac{T_{j-1}^k - 2T_j^k + T_{j+1}^k}{\Delta x^2} = q_j^k$$

Implicit time stepping:

$$\frac{T_j^{k+1} - T_j^k}{\Delta t} - \alpha \frac{T_{j-1}^{k+1} - 2T_j^{k+1} + T_{j+1}^{k+1}}{\Delta x^2} = q_j^{k+1}$$

Computational form: explicit

$$T_j^{k+1} = T_j^k + \frac{\alpha \Delta t}{\Delta x^2}(T_{j-1}^k - 2T_j^k + T_{j+1}^k) + \Delta t q_j^k$$

This has an explicit form:

$$\underset{\sim}{T}^{k+1} = \left(I + \frac{\alpha \Delta t}{\Delta x^2}\right) K \underset{\sim}{T}^k + \Delta t \underset{\sim}{q}^k$$

Computational form: implicit

$$T_j^{k+1} - \frac{\alpha \Delta t}{\Delta x^2}(T_{j-1}^k - 2T_j^k + T_{j+1}^k) = T_j^k + \Delta t q_j^k$$

This has an implicit form:

$$\left(I - \frac{\alpha \Delta t}{\Delta x^2} K\right) \underset{\sim}{T}^{k+1} = \underset{\sim}{T}^k + \Delta t \underset{\sim}{q}^k$$

Needs to solve a linear system in every time step

Stability of explicit scheme

Let $q \equiv 0$; assume $T_j^k = \beta_\ell^k e^{i\ell x_j}$:

$$
\begin{aligned}
T_j^{k+1} &= T_j^k + \frac{\alpha \Delta t}{\Delta x^2}(T_{j-1}^k - 2T_j^k + T_{j+1}^k) \\
\Rightarrow \beta_\ell^{k+1} e^{i\ell x_j} &= \beta_\ell^k e^{i\ell x_j} + \frac{\alpha \Delta t}{\Delta x^2}(\beta_\ell^k e^{i\ell x_{j-1}} - 2\beta_\ell^k e^{i\ell x_j} + \beta_\ell^k e^{i\ell x_{j+1}}) \\
&= \beta_\ell^k e^{i\ell x_j}\left[e^{-i\ell\Delta x} - 2 + e^{i\ell\Delta x}\right] \\
\Rightarrow \frac{\beta_\ell^{k+1}}{\beta_\ell^k} &= 1 + 2\frac{\alpha \Delta t}{\Delta x^2}[\frac{1}{2}(e^{i\ell\Delta x} + e^{-\ell\Delta x}) - 1] \\
&= 1 + 2\frac{\alpha \Delta t}{\Delta x^2}(\cos(\ell\Delta x) - 1)
\end{aligned}
$$

$$\frac{\beta_\ell^{k+1}}{\beta_\ell^k} = 1 + 2\frac{\alpha\Delta t}{\Delta x^2}(\cos(\ell\Delta x) - 1)$$

Assume that $|\beta_\ell^{k+1}/\beta_\ell^k| < 1$ for all ℓ

- $2\frac{\alpha\Delta t}{\Delta x^2}(\cos(\ell\Delta x) - 1) < 0$: automatic
- $2\frac{\alpha\Delta t}{\Delta x^2}(\cos(\ell\Delta x) - 1) > -2$: needs $2\frac{\alpha\Delta t}{\Delta x^2} < 1$, that is

$$\Delta t < \frac{\Delta x^2}{2\alpha}$$

big restriction on size of time steps

Stability of implicit scheme

$$T_j^{k+1} - \frac{\alpha\Delta t}{\Delta x^2}(T_{j_1}^{k+1} - 2T_j^{k+1} + T_{j+1}^{k+1}) = T_j^k$$

$$\Rightarrow \beta_\ell^{k+1} e^{i\ell\Delta x} - \frac{\alpha\Delta t}{\Delta x^2}(\beta_\ell^{k+1} e^{i\ell x_{j-1}} - 2\beta_\ell^{k+1} e^{i\ell x_j} + \beta_\ell^{k+1} e^{i\ell x_{j+1}}) = \beta_\ell^k e^{i\ell x_j}$$

$$\Rightarrow \frac{\beta_\ell^{k+1}}{\beta_\ell^k} = \frac{1}{1 - 2\frac{\alpha\Delta t}{\Delta x^2}(\cos(\ell\Delta x) - 1)}$$

$$= \frac{1}{1 + 2\frac{\alpha\Delta t}{\Delta x^2}(1 - \cos(\ell\Delta x))}$$

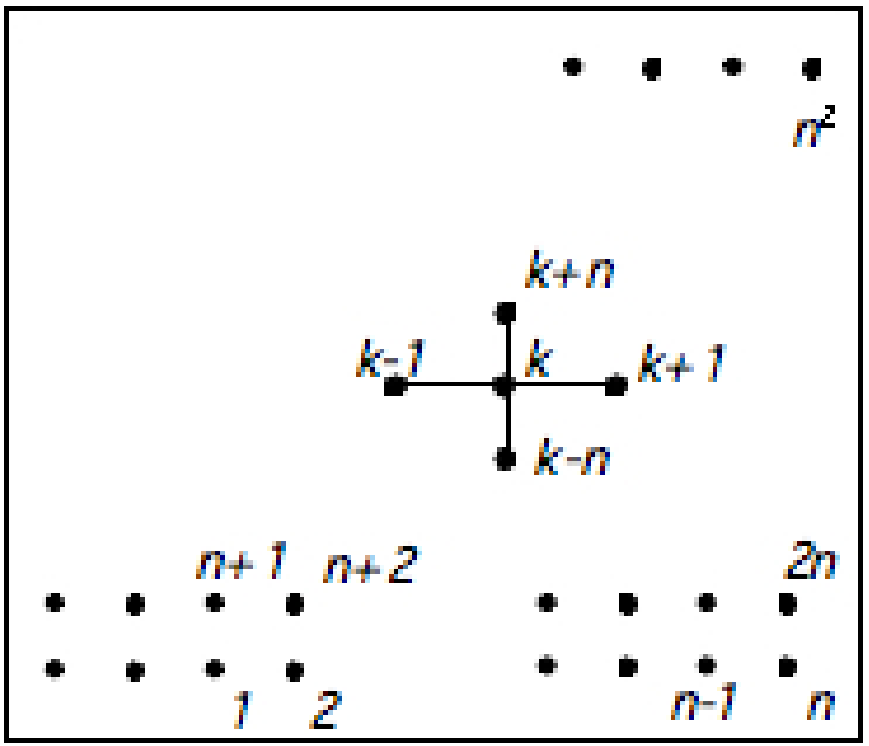
So condition $|\beta_\ell^{k+1}/\beta_\ell^k| < 1$ always satisfied: method always stable.

Sparse matrix in 2D case

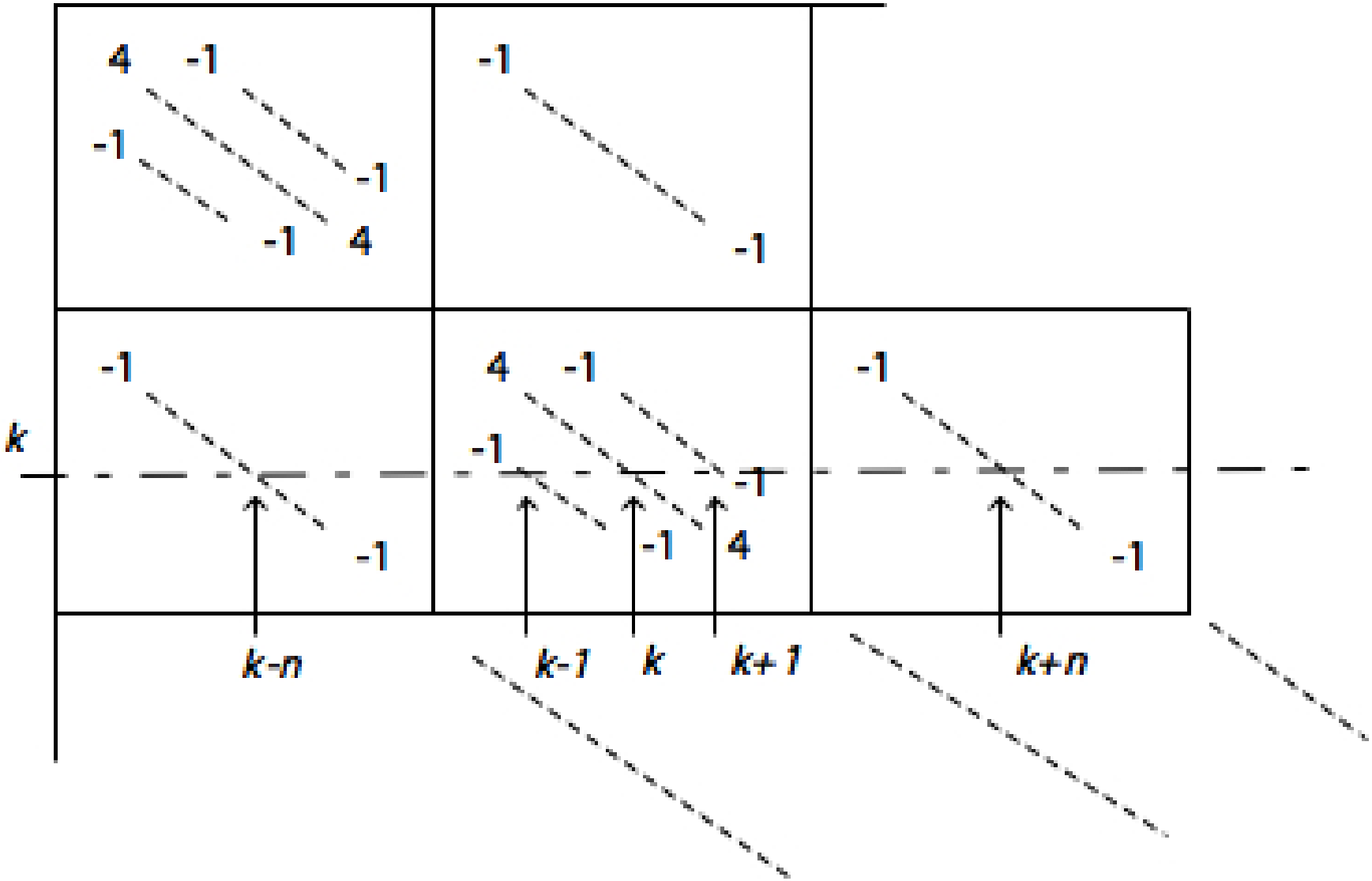
Sparse matrices so far were tridiagonal: only in 1D case.

Two-dimensional: $-u_{xx} - u_{yy} = f$ on unit square $[0,1]^2$

Difference equation:
$4u(x,y) - u(x+h,y) - u(x-h,y) - u(x,y+h) - u(x,y-h) = f(x,y)$



Sparse matrix from 2D equation



Sparse matrix storage

Matrix above has many zeros: n^2 elements but only $O(n)$ nonzeros. Big waste of space to store this as square array.

Matrix is called 'sparse' if there are enough zeros to make specialized storage feasible.

$$A = \begin{pmatrix} 10 & 0 & 0 & 0 & -2 & 0 \\ 3 & 9 & 0 & 0 & 0 & 3 \\ 0 & 7 & 8 & 7 & 0 & 0 \\ 3 & 0 & 8 & 7 & 5 & 0 \\ 0 & 8 & 0 & 9 & 9 & 13 \\ 0 & 4 & 0 & 0 & 2 & -1 \end{pmatrix} . \tag{1}$$

Compressed Row Storage (CRS): store all nonzeros by row, their column indices, pointers to where the columns start (1-based indexing):

`val`	10	-2	3	9	3	7	8	7	$3 \cdots 9$	13	4	2	-1
`col_ind`	1	5	1	2	6	2	3	4	$1 \cdots 5$	6	2	5	6

`row_ptr`	1	3	6	9	13	17	20

.

Sparse matrix operations

Most common operation: matrix-vector product

```
for (row=0; row<nrows; row++) {
   s = 0;
   for (icol=ptr[row]; icol<ptr[row+1]; icol++) {
      int col = ind[icol];
      s += a[aptr] * x[col];
      aptr++;
   }
   y[row] = s;
}
```

Operations with changes to the nonzero structure are much harder!

Indirect addressing of `x` gives low spatial and temporal locality.