

Matrix form

$$\begin{bmatrix} u_{1,j+1} \\ \vdots \\ u_{N,j+1} \end{bmatrix} = \begin{bmatrix} u_{1,j} \\ \vdots \\ u_{N,j} \end{bmatrix} + \lambda \begin{bmatrix} -2 & 1 & 0 & \cdots & 0 \\ 1 & -2 & 1 & \cdots & 0 \\ 0 & 1 & -2 & 1 & 0 & \cdots & 0 \\ \vdots & & & & & \ddots & \\ 0 & - & 0 & 1 & -2 & 1 & \cdots & 0 \\ 0 & - & - & 0 & 1 & -2 & \cdots & 1 \end{bmatrix} \begin{bmatrix} u_{1,j} \\ \vdots \\ u_{N,j} \end{bmatrix} + \begin{bmatrix} f_{1,j} \\ \vdots \\ f_{N,j} \end{bmatrix}$$

$\nearrow \vec{u}_{j+1}$

$\nearrow \vec{h}_{j+1}$

$\nearrow u(x_0, t_j)$

$\nearrow u_{0,j}$

with $u_{0,j} = 0$, $u_{N+1,j} = 0$ always

$|$
 $u_{N+1,j}$

Matrix form

$$\begin{bmatrix} u_{1,j+1} \\ \vdots \\ u_{N,j+1} \end{bmatrix} = \begin{bmatrix} u_{1,j} \\ \vdots \\ u_{N,j} \end{bmatrix} + \lambda \begin{bmatrix} -2 & 1 & 0 & \cdots & 0 \\ 1 & -2 & 1 & \cdots & 0 \\ 0 & 1 & -2 & 1 & 0 \\ \vdots & & & & \vdots \\ 0 & - & 0 & 1 & -2 \\ 0 & - & - & 0 & 1 \end{bmatrix} \begin{bmatrix} u_{1,j} \\ \vdots \\ u_{N,j} \end{bmatrix} + k \begin{bmatrix} f_{1,j} \\ \vdots \\ f_{N,j} \end{bmatrix}$$

with $u_{0,j} = 0$, $u_{N+1,j} = 0$ always

→ Why is this OK?

Matrix form

$$\begin{bmatrix} u_{1,j+1} \\ \vdots \\ u_{N,j+1} \end{bmatrix} = \begin{bmatrix} u_{1,j} \\ \vdots \\ u_{N,j} \end{bmatrix} + \lambda \begin{bmatrix} -2 & 1 & 0 & \cdots & 0 \\ 1 & -2 & 1 & \cdots & 0 \\ 0 & 1 & -2 & 1 & 0 \\ \vdots & & & & \vdots \\ 0 & -1 & 0 & 1 & -2 \\ 0 & -1 & 0 & 1 & -2 \end{bmatrix} \begin{bmatrix} u_{1,j} \\ \vdots \\ u_{N,j} \end{bmatrix} + k \begin{bmatrix} f_{1,j} \\ \vdots \\ f_{N,j} \end{bmatrix}$$

with $u_{0,j} = 0$, $u_{N+1,j} = 0$ always

$$\vec{u}_{j+1} = \underbrace{\begin{bmatrix} 1-2\lambda & 1 & 0 & \cdots & 0 \\ 1 & 1-2\lambda & 1 & 0 & \cdots & 0 \\ & & \ddots & & & \\ & & & 0 & 1 & 1-2\lambda \end{bmatrix}}_A \vec{u}_j + k \vec{f}_j$$

Direct Method:

$$\vec{u}_{j+1} = A \vec{u}_j + \vec{f}_j$$

4 q.g

doubles (64 b.t)

$N+N$

Full vs Tridiagonal

$n \times n$
 \downarrow
 $n \times 1$

$A x$

Full

$O(n^2)$

Tridiagonal

$O(n)$

$A x = b$

$O(n^3)$

$O(n)$

(LU factorization)

(Tridiagonal solver)

Full vs Tridiagonal

$n \times n$
 \downarrow
 $n \times 1$

$A x$

Full

$O(n^2)$

Tridiagonal

$O(n)$

$A x = b$

$O(n^3)$

$O(n)$

(LU factorization)

(Tridiagonal solver)

Relative labor if $n = 1000$:

Full vs Tridiagonal

$n \times n$
 \downarrow
 $n \times 1$

$A x$

Full

$O(n^2)$

Tridiagonal

$O(n)$

$\times 1000$

$A x = b$

$O(n^3)$

$O(n)$

(LU factorization)

(Tridiagonal solver)

Relative labor if $n = 1000$:

Full vs Tridiagonal

$n \times n$
 \downarrow
 $n \times 1$

$A x$

Full

$O(n^2)$

Tridiagonal

$O(n)$

$\times 1000$

$A x = b$

$O(n^3)$

$O(n)$

$\times 1000000$

(LU factorization)

(Tridiagonal solver)

Relative labor if $n = 1000$:

MATLAB

sparse, spzeros

$A \backslash b$ will notice the matrix is
tridiagonal. Easy!

Python

scipy.linalg.solve_banded

Python

scipy.linalg.solve_banded

$$(l, u)$$

hats below # above

The diagram shows a square matrix with indices. The main diagonal elements are labeled $a_{00}, a_{11}, \dots, a_{n-1,n}$. Above the main diagonal, there are two shaded regions: one from $(0,0)$ to $(l,0)$ and another from $(0,u)$ to (n,n) . Below the main diagonal, there is a shaded region from $(0,n)$ to (n,n) . The label (l, u) is positioned at the top left, with arrows pointing to the shaded regions. The label "# hats below" is to the left of the first arrow, and "# above" is to the right of the second arrow.

* is ignored

$$A = \begin{bmatrix} a_{00} & a_{01} & 0 & \cdots \\ a_{10} & a_{11} & a_{12} & \cdots \\ \vdots & \vdots & \ddots & \ddots \\ a_{n,n-1} & a_{n,n} & & \end{bmatrix}$$

Python

scipy.linalg.solve_banded

(l, u)
 ↑
 # hats
 below #
 above

$$\begin{bmatrix} * & a_{01} & & & \\ a_{00} & a_{11} & \cdots & a_{n-1,n} \\ a_{10} & a_{21} & & \\ & a_{n,n} & * & \\ & & a_{n,n} & * \end{bmatrix}$$

* is ignored

$$A = \begin{bmatrix} a_{00} & a_{01} & 0 & \cdots \\ a_{10} & a_{11} & a_{12} & \cdots \\ & \ddots & \ddots & \ddots \\ & & a_{n,n-1} & a_{nn} \end{bmatrix}$$

(l, l)

$$\begin{bmatrix} * & x & \cdots & x \\ -2x & 1-2x & \cdots & -2x \\ x & \cdots & \ddots & x \\ & & & * \end{bmatrix}$$

Ad

super easy to construct

Python

`scipy.linalg.solve_banded`

$$(l, u)$$

↑ ↑
 # hats below # above

\ast is ignored

$$A = \begin{bmatrix} a_{00} & a_{01} & 0 & \cdots \\ a_{10} & a_{11} & a_{12} & \cdots \\ \vdots & \vdots & \ddots & \ddots \\ a_{n-1, n-1} & a_{n-1, n} & a_{n, n-1} & a_{n, n} \end{bmatrix}$$

$$(l, 1)$$

\ast is ignored

super easy to construct

Ad

`scipy.sparse.spdiags (Ad, (l, 0, -1), N, N)`

A. dot (b)

Demo

Direct Method

$$\hat{u}_{j+1} = (I - \lambda D) \hat{u}_j + k \hat{f}_j$$

is exactly Euler's Method applied to

$$u' = \frac{1}{h^2} D u + f$$

$$D = \begin{bmatrix} -2 & 1 & & & \\ 1 & -2 & 1 & & 0 \\ & \ddots & \ddots & \ddots & \\ & 0 & \ddots & \ddots & -2 \end{bmatrix}$$

Now $\frac{1}{h^2} D$ is supposed to approximate ∂_x^2

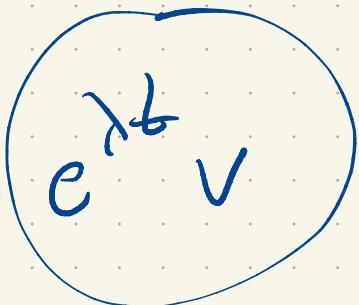
And $\partial_x^2 v_j = -j^2 \pi^2 v_j$ $v_j = \sin(j\pi x)$

So reasonable to expect $\frac{1}{h^2} D$ has eigenvalues

$$\sim -\pi^2, -2^2\pi^2, \dots, -N^2\pi^2$$

↳ large, negative.

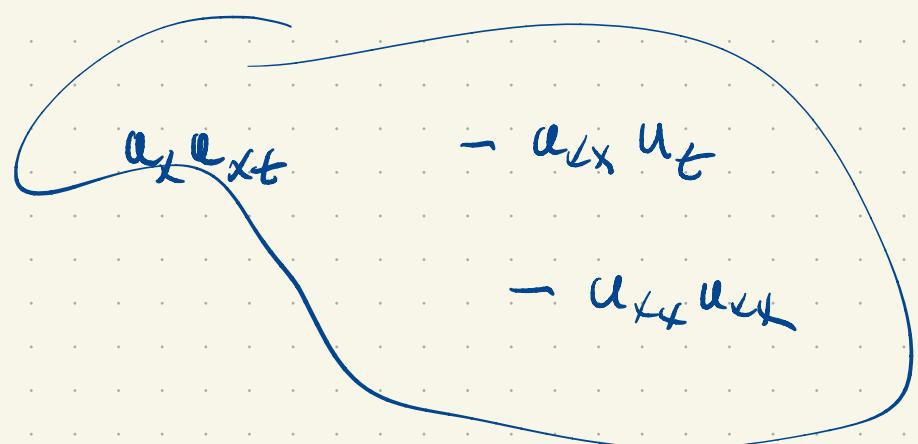
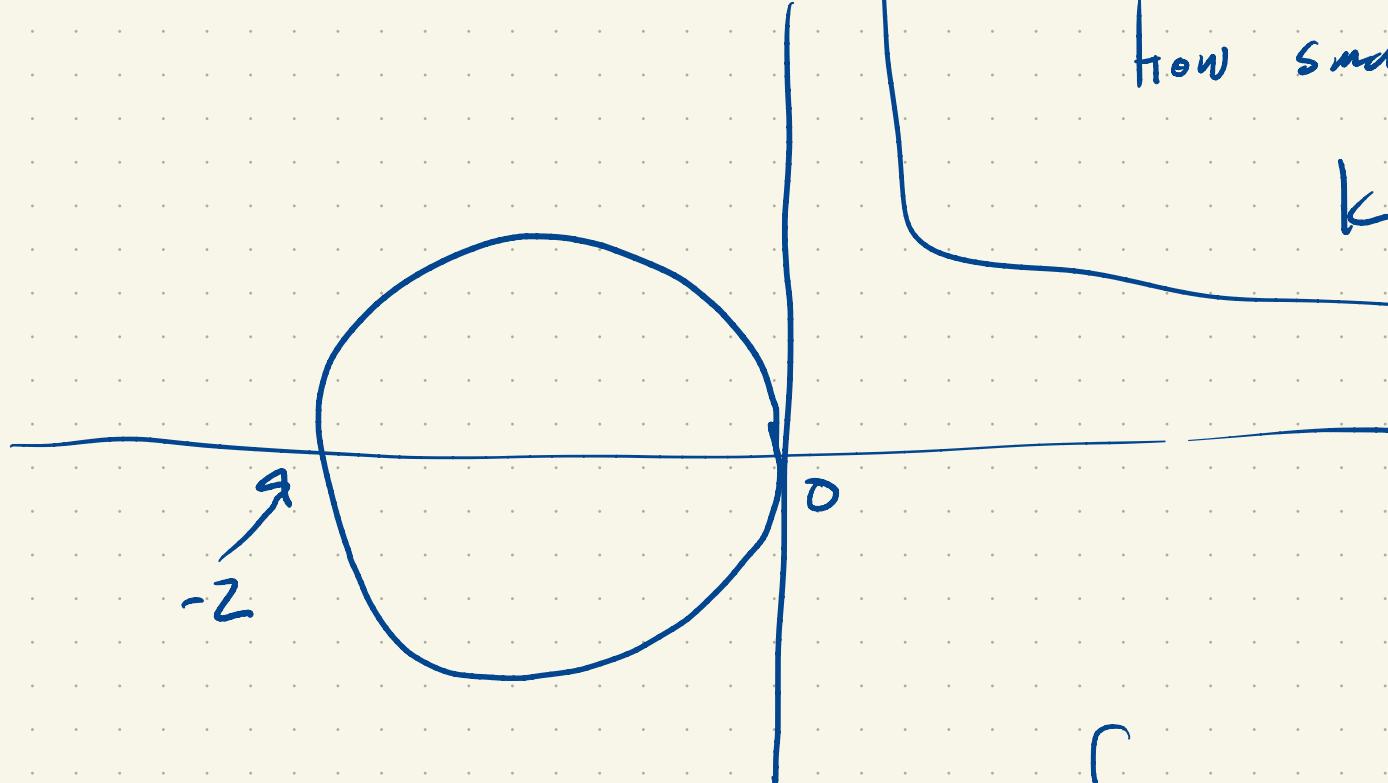
Transients!



Region of abs stability:

$$-\pi^2, -4\pi^2, \dots, -N^2\pi^2$$

How small should
 k be?



S_{xx}

$$S_{a_{xx} u_t} = -S_{a_{xxx} u_{xx}}$$

~~α_{xx}~~