Explicit Method for heat equation

$$u_t = u_{xx} + f(x,t)$$

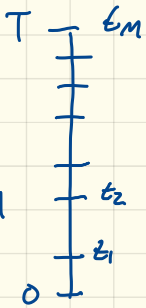
$$u(0,t) = 0$$
$$u(l,t) = 0$$

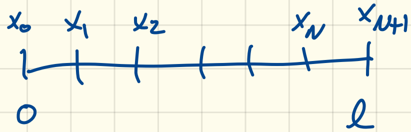
$x \longleftrightarrow t$ swap
l instead of 1.

$$u(x,0) = g(x) \qquad 0 \le x \le l.$$

Introduce a grid of sample points on our domain

M intervals
$k = T/M$





M+1 sample times,
M unknown, one initial

N+1 intervals
N+2 sample points
N unknowns at $x_1, \dots, x_N$

Introduce approximations for the derivatives.

We've spent a lot of time thinking about discritizing time derivatives. Let's hold off on those. Instead, how about the space derivatives?

$$u_x(x_i) = \frac{u(x_i+h) - u(x_i)}{h} + O(h)$$

$$u_x(x_i) = \frac{u(x_i) - u(x_i-h)}{h} + O(h)$$

$$u_{xx}(x_i) = \frac{u(x_i+h) - 2u(x_i) + u(x_i-h)}{h^2} + ?$$

$$u(x_i+h) = u(x_i) + u_x(x_i)h + \frac{1}{2}u_{xx}(x_i)h^2 + \frac{1}{6}u_{xxx}(x_i)h^3 +$$

$$u(x_i-h) = u(x_i) - u_x(x_i)h + \frac{1}{2}u_{xx}(x_i)h^2 - \frac{1}{6}u_{xxx}(x_i)h^3 +$$

$O(h^4)$

$$u(x_i+h) - 2u(x_i) + u(x_i-h) = u_{xx}(x_i)h^2 + O(h^4)$$

$$u_{xx}(x_i) = \frac{(\quad)}{h^2} + O(h^2)$$

vanish as $h \to 0$

$$u_t(x_i, t_j) = \frac{u(x_i, t_j + k) - u(x_i, t_j)}{k} + O(k)$$

$$u_{i,j} \approx u(x_i, t_j)$$

$$\frac{u_{i,j+1} - u_{i,j}}{k} = \frac{u_{i+1,j} - 2u_{i,j} + u_{i-1,j}}{h^2} + f(x_i, t_j)$$

Local truncation error: subsitute true solution

$$O(k) + O(h^2)$$

$$f(x_i, t_j) = u_t - u_{xx} \quad \text{at } (x_i, t_j).$$

$$= \frac{\quad}{k} + O(k) \qquad = \frac{\quad}{h^2} + O(h^2)$$

It will be helpful to write this as

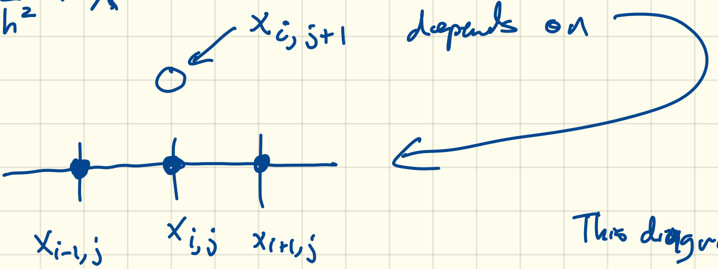
$$u_{i,j+1} = u_{i,j} + \left(\frac{k}{h^2}\right)\left[u_{i+1,j} - 2u_{i,j} + u_{i,j}\right] + k\,\underbrace{f(x_i, t_j)}_{f_{i,j}}$$

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

so above holds $1 \le i \le N$

$0 \le j \le M-1$

$\frac{k}{h^2} : \lambda$



This diagram is known as a stencil.

might be 0
$i=1$

might be 0
$i=N$

$$u_{i,j+1} = \lambda u_{i-1,j} + (1-2\lambda) u_{i,j} + \lambda u_{i+1,j} + f_{ij}$$

$$\underbrace{\begin{bmatrix} u_{1,j+1} \\ \vdots \\ \vdots \\ u_{N,j+1} \end{bmatrix}}_{\vec{u}_{j+1}} = \underbrace{\begin{bmatrix} (1-2\lambda) & \lambda & & & \\ \lambda & 1-2\lambda & \lambda & & \\ & \lambda & 1-2\lambda & \lambda & \\ & & \ddots & \ddots & \lambda \\ & & & \lambda & (1-2\lambda) \end{bmatrix}}_{A} \underbrace{\begin{bmatrix} u_{1,j} \\ \vdots \\ \vdots \\ u_{N,j} \end{bmatrix}}_{\vec{u}_j} + \underbrace{\begin{bmatrix} f_{1,j} \\ \vdots \\ \vdots \\ f_{N,j} \end{bmatrix}}_{\vec{f}_j}$$

$$\vec{u}_{j+1} = A\vec{u}_j + f_j \qquad \vec{u}_0 = \vec{g} \qquad g_i = u_0(x_i)$$

So this gives us a compact way to express the operations of solving this equation.

You wouldn't want to build A as a full matrix for a big problem though: it's mostly 0's

Full

Tridiagonal

Ax : $O(n^2)$

Ax : $O(n)$

$Ax = b$, solve

LU $O(n^3)$

$O(n)$

1000×1000 : multiplication takes 1000 · more work

solve takes 1,000,000 times more work.

These are losing odds. For small problems you won't notice, but your code won't scale.

Matlab: use sparse matrices

sparse(m,n) ~ zeros(m,n)

b\A will detect A is banded.

Python: need to hold its hand

scipy.linalg.solve_banded

(l, u)

bands below

above

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

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

$*$ is ignored

$(1, 1)$

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

super easy to construct.

Ad

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

A.multiply(b)

		$k=0.02$	$k=0.01$	$k=0.005$
		M	M	M
Timesteps:	$t=0.02$	1	2	4
	0.04	2	4	8
	0.1	5	10	20

$t=0.2 \quad M=20 \quad (h=0.01)$

$t=0.4 \quad M=20 \quad (h=0.02)$

Visualize the instability

None are great.

$$\frac{k}{h^2} = \frac{\frac{2}{100}}{\left(\frac{1}{20}\right)^2} = \frac{\frac{2}{100}}{\frac{1}{400}} = 8$$

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

$$u_j = e^{r x_j \mathcal{J}} \qquad \mathcal{J}^2 = -1$$

$$\begin{aligned} u_{j-1} - 2u_j + u_{j+1} &= \left[u_{j-1} - 2u_j + u_{j+1} \right] \\ &= e^{r x_j \mathcal{J}} \left[e^{-hr\mathcal{J}} - 2 + e^{hr\mathcal{J}} \right] \\ &= -2\, e^{r x_j \mathcal{J}} \left[1 - \left(\frac{e^{hr\mathcal{J}} + e^{-hr\mathcal{J}}}{2} \right) \right] \\ &= -2\, e^{r x_j \mathcal{J}} \left[1 - \cos(rh) \right] \end{aligned}$$

Subtlety: this analysis doesn't apply to the boundary points.

But: $D\,\mathrm{Re}(\vec{u}_r) = -2(1-\cos(rh))\,\mathrm{Re}(\vec{u}_r)$

$D\,\mathrm{Im}(\vec{u}_r) = \qquad\qquad \mathrm{Im}(\vec{u}_r)$ as well

$\mathrm{Im}\,\vec{u}_{r,j} = \sin(r x_j)$

$\sin(r x_0) = \sin(0) = 0$

$r = \pi m$

$\sin(\pi m\, x_{N+1}) = \sin(\pi m) = 0$

If $r = \pi k$, $\vec{v}_r = \mathrm{Im}\,\vec{u}_r$ is an eigenvector of D

with eigenvalue $\quad -2(1-\cos(rk))$

$$\frac{1}{h^2} D \vec{v}_r = \frac{-2\left(1 - \cos(rh)\right)}{h^2} \vec{u}_r$$

$$\cos(rh) = 1 - \frac{(rh)^2}{2} + O((rh)^4)$$

$$= \left[-r^2 + O(h^2) \right] \vec{u}_r$$

$r = \pi m$ $\qquad -(\pi m)^2$

It's trying to be an eigenvector of the laplacian.

Now apply to the system

If $\vec{u}_j = c\vec{v}_r$

$$\vec{u}_{j+1} = (1 + \lambda D) c \vec{v}_r$$

$$= c \left(1 - 2\lambda(1-\cos(rh))\right) \vec{v}_r$$

$$= \underbrace{\left[1 - 2\lambda(1-\cos(rh)\right]}_{\text{amplification factor}} \underbrace{c\vec{v}_r}_{\vec{u}_j}$$

$$\frac{1-\cos(\theta)}{2} = \sin^2(\theta/2)$$

$$\left[1 - 4\lambda \sin^2(rh/2)\right]$$

We obtain a growing mode if

$$\Big| \underbrace{1 - 4\lambda \sin^2(\nu h/2)}_{\le 1 \text{ always.}} \Big| > 1$$

But ≤ 1 always.

$$1 - 4\lambda \sin^2(\nu h/2) < -1$$

$$\lambda \sin^2(\nu h/2) > +\frac{1}{2}$$

no good control on this. could be ~ 1.

$\lambda > \frac{1}{2}$ leaves possibility of a growing mode

We say the system is <u>stable</u> if $\lambda \le \frac{1}{2}$

$\lambda = \frac{k}{h^2}$ so $k \le \frac{h^2}{2}$