MATH 243A LECTURE NOTES

PLAMEN KOEV

1. Determining the complexity of an algorithm

"Programs that take too long to run never get run." William Kahan

The complexity $f(n)$ of an algorithm is the number of arithmetic operations that it performs. It is a function of the problem size n. The goal is expressing $f(n)$ as a function of n, most often in the form

$$f(n) = an^k + \mathcal{O}(n^{k-1}).$$

Note that:

- The problem size n is usually the size of a matrix, the number of discretization points, size of the mesh or other appropriate quantity that measures the "size" of the problem.
- The complexity behavior matters when n gets larger, which is exactly when the lower order terms ($\mathcal{O}(n^{k-1})$) no longer contribute anything significant compared with an^k. This is why these lower order terms are most often ignored in the complexity analysis and an algorithm is said to have complexity an^k or simply n^k.
- It is common practice in numerical analysis to count each arithmetic operation ($+, -, \times, /$, square root) as 1 and compare algorithms that way. Computer technology changes rapidly so who knows how many clock cycles each operation takes on a computer this month. At the time of this writing, multiplication, addition, or subtraction can be done in 1 clock cycle, with division taking 6 to 8. Such performance is only possible, however, through sophisticated cache optimization and programming techniques, which we will not address.

The complexity of an algorithm can be determined by counting the number of arithmetic operations which can be done either analytically or experimentally.

The goal is to figure out what $f(n)$ is as a function of n, i.e., to find a and k.

Analytically, one uses formulas such as:

$$\sum_{i=1}^{n} i = \frac{n(n+1)}{2} \quad \text{and} \quad \sum_{i=1}^{n} i^2 = \frac{n(n+1)(2n+1)}{6}.$$

Experimentally, one adds a variable, e.g., `flopcount`, to the code and adds 1 to it every time an arithmetic operation is performed. Then the code is run for different (reasonably large) values of n, say n_1 and n_2.

The variable `flopcount` will return $f(n_1)$ and $f(n_2)$. Since

$$f(n_1) = an_1^k + \mathcal{O}(n_1^{k-1}) \quad \text{and} \quad f(n_2) = an_2^k + \mathcal{O}(n_2^{k-1})$$

we have for large n_1, n_2

$$\log(f(n_1)) \approx \log(an_1^k) = \log(a) + k\log(n_1)$$
$$\log(f(n_2)) \approx \log(an_1^k) = \log(a) + k\log(n_2)$$

Date: December 4, 2012.

so

$$\log(f(n_1)) - \log(f(n_2)) = k(\log(n_1) - \log(n_2))$$

and

$$k = \frac{\log(f(n_1)) - \log(f(n_2))}{\log(n_1) - \log(n_2)}.$$

Then

$$a \approx \frac{f(n_1)}{n_1^k}.$$

This relationship should improve as n_1, n_2 increase.

Example 1.1. Consider the following algorithm that back solves for the solution x of a lower triangular linear system $Ax = b$. An appropriate measure of the size of this problem is n, the size of the matrix. The solution is

$$\begin{aligned} x_1 &= b_1/a_{11} \\ x_2 &= (b_2 - a_{21}x_1)/a_{22} \\ &\vdots \\ x_n &= (b_n - a_{n1}x_1 - \ldots a_{n,n-1}x_{n-1})/a_{nn}. \end{aligned}$$

So, analytically, this takes

$$1 + 3 + 5 + \ldots + (2n-1) = \sum_{i=1}^{n}(2i-1) = 2\sum_{i=1}^{n} i - n = 2\frac{n(n+1)}{2} - n = n^2$$

operations.

On the other side, experimentally, the code that solves $Ax = b$ is

```
for i=1:n
    x(i)=b(i);
    for j=1:i-1
       x(i)=x(i)-A(i,j)*x(j);
    end
    x(i)=x(i)/A(i,i);
end
```

Adding to the code the counter `flopcounter`, we get

```
for i=1:n
    x(i)=b(i);
    for j=1:i-1
       x(i)=x(i)-A(i,j)*x(j);
       flopcount=flopcount+2;
    end
    x(i)=x(i)/A(i,i);
    flopcount=flopcount+1;
end
```

Running the augmented code for some reasonable values of n, say 500 and 505, we get for `flopcount` 250000 and 255025, respectively. Now we have

$$k = \frac{\log(255025) - \log(250000)}{\log(505) - \log(500)} = 2$$

and

$$a = \frac{255025}{505^2} = 1$$

as expected.

Thus the complexity of the algorithm is n^2.

The advantage of the experimental method is that one does not need to understand how an algorithm works or what it does in order to analyze it.

2. Crank–Nicolson

This is the scheme for

$$\frac{\partial u}{\partial t} = \frac{\partial^2 u}{\partial x^2}$$

discretized as

$$\frac{u_{i,j+1} - u_{ij}}{k} = \frac{1}{2}\left(\frac{u_{i+1,j+1} - 2u_{i,j+1} + u_{i-1,j+1}}{h^2} + \frac{u_{i+1,j} - 2u_{ij} + u_{i-1,j}}{h^2}\right).$$

Equivalently,

$$u_{i,j+1} - u_{ij} = \frac{\mu}{2}\left(u_{i+1,j+1} - 2u_{i,j+1} + u_{i-1,j+1} + u_{i+1,j} - 2u_{ij} + u_{i-1,j}\right),$$

where $\mu = \frac{k}{h^2}$. This means

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

Assuming we're solving on $x \in [0,1]$ with n discretization points, $h = 1/n$, $i = 1, 2, \ldots, n$. WIth initial conditions for $j = 1$ and boundary conditions for $i = 1$ and $i = n+1$, this implies a tridiagonal linear system for the unknown vector $(u_{2,j+1}, \ldots, u_{n,j+1})$.

3. Consistency, Stability, Well-posedness, and Convergence

Definition 3.1 (Consistency)**.** We say that a finite difference scheme $P_{k,h}v = f$ is *consistent* with the PDE $Pu = f$ *of order* (r, s) if for any smooth function ϕ

$$P\phi - P_{k,h}\phi = O(k^r, h^s) \tag{3.1}$$

To verify consistency expand ϕ in Taylor series and make sure (3.1) holds.

Definition 3.2 (L^2 norm)**.** For a function $w = (\ldots, w_{-2}, w_{-1}, w_0, w_1, w_2, \ldots)$ on a grid with step size h:

$$\|w\| = \left(h \sum_{m=-\infty}^{\infty} |w_m|^2 \right)^{1/2}$$

For a function f on the real line:

$$\|f\| = \left(\int_{-\infty}^{\infty} |f(x)|^2 dx \right)^{1/2}$$

Definition 3.3 (Stability)**.** A finite one-step difference scheme $P_{k,h}v_m^n = 0$ for a first-order PDE is *stable* if there exist numbers $k_0 > 0$ and $h_0 > 0$ such that for any $T > 0$ there exists a constant C_T such that

$$\|v^n\| \le C_T \|v^0\|$$

for $0 \le nk \le T, 0 < h \le h_0, 0 < k \le k_0$.

Definition 3.4 (Well-posedness)**.** The initial value problem for the first-order PDE $Pu = 0$ is *well-posed* if for any time $T \ge 0$, there is a constant C_T, such that any solution $u(t, x)$ satisfies

$$\|u(t, x)\| \le C_T \|u(0, x)\|, \qquad \text{for } 0 \le t \le T.$$

Definition 3.5 (Convergence)**.** A one-step finite difference scheme approximating a PDE is *convergent* if for any solution to the PDE, $u(t, x)$, and solution to the finite difference scheme v_m^n, such that $v_m^0 \to u(0, x)$ as $mh \to x$, we have $v_m^n \to u(t, x)$ as $(nk, mh) \to (t, x)$ (as $h, k \to 0$).

Theorem 3.6 (Lax)**.** *A consistent finite difference scheme for a PDE for which the initial value problem is well-posed is convergent if and only if it is stable.*

4. Eigenvalues of certain tridiagonal matrices

In this section, we will study the eigenvalues of the $n \times n$ tridiagonal matrix

$$A = \begin{bmatrix} a & b & & & \\ c & a & b & & \\ & c & a & \ddots & \\ & & \ddots & \ddots & b \\ & & & c & a \end{bmatrix}$$

Since $A = aI + B$, where

$$B = \begin{bmatrix} 0 & b & & & \\ c & 0 & \ddots & & \\ & \ddots & \ddots & b & \\ & & c & 0 & \end{bmatrix}$$

it suffices to find the eigenvalues of B (the eigenvalues of A would then be $a+$ the eigenvalues of B).

First, we observe that B is similar to the matrix

$$C = \begin{bmatrix} 0 & d & & & \\ d & 0 & \ddots & & \\ & \ddots & \ddots & d & \\ & & d & 0 & \end{bmatrix}$$

where $d = \sqrt{bc}$. The similarity is $C = DCD^{-1}$, where

$$D = \operatorname{diag}\left(1, (c/b)^{1/2}, (c/b)^{2/2}, (c/b)^{3/2}, \ldots, (c/b)^{(n-1)/2}\right).$$

The verification is trivial. The matrices B and C thus have the same eigenvalues.

Now, $C = d \cdot E$, where

$$E = \begin{bmatrix} 0 & 1 & & & \\ 1 & 0 & \ddots & & \\ & \ddots & \ddots & 1 & \\ & & 1 & 0 & \end{bmatrix}.$$

If $\mu_1, \mu_2, \ldots, \mu_n$ are the eigenvalues of E, the eigenvalues of A are then $\lambda_i = a + \sqrt{bc} \cdot \mu_i$, $i = 1, 2, \ldots, n$.

The eigenvectors[1] of E are then z_j, $j = 1, 2 \ldots, n$, where $z_j(k) = \sin \frac{jk\pi}{n+1}$.

Using the trig relationship $\sin\alpha + \sin\beta = 2\sin\frac{\alpha+\beta}{2}\cos\frac{\alpha-\beta}{2}$ we get

$$\sin\frac{j(k-1)\pi}{n+1} + \sin\frac{j(k+1)\pi}{n+1} = 2\sin\frac{jk\pi}{n+1}\cos\frac{j\pi}{N+1}.$$

Thus we confirm that the z_j are the eigenvectors of E and note that the eigenvalues are

$$\mu_j = 2\cos\frac{j\pi}{N+1},\ j = 1, 2, \ldots, n.$$

The eigenvalues of A are thus

$$\lambda_j = a + 2\sqrt{bc}\cos\frac{j\pi}{N+1},\ j = 1, 2, \ldots, n.$$

[1]The eigenvector z_j would need to be multiplied by $\sqrt{\frac{2}{N+1}}$ to be of unit length, but this is not necessary here.

5. Norms of vectors and matrices

A vector norm is a function $\|\cdot\| : \mathbb{R}^n \to \mathbb{R}$ such that for any vector x

(1) $\|x\| \geq 0$, with $\|x\| = 0$ only when $x = 0$;
(2) $\|cx\| = |c| \cdot \|x\|$ for all scalars $c \in \mathbb{R}$;
(3) $\|x + y\| \leq \|x\| + \|y\|$.

Examples of vector norms are

$$\begin{aligned} \|x\|_1 &= |x_1| + |x_2| + \cdots + |x_n| \\ \|x\|_\infty &= \max_j |x_j| \\ \|x\|_2 &= \sqrt{x_1^2 + x_2^+ \cdots + x_n^2} \end{aligned}$$

For the 2-norm, we have, very importantly

$$\|x\|_2 = \sqrt{x^T x}.$$

Also, if Q is an orthogonal matrix (i.e., $Q^T Q = I$), then $\|Qx\|_2 = \sqrt{(Qx)^T Qx} = \sqrt{x^T Q^T Q x} = \sqrt{x^T x} = \|x\|_2$. Namely, orthogonal matrices do not change the 2-norm.

The for matrices A and B, a norm must satisfy all 3 properties above and one additional one.

(1) $\|A\| \geq 0$, with $\|A\| = 0$ only when $A = 0$;
(2) $\|cA\| = |c| \cdot \|A\|$ for all scalars $c \in \mathbb{R}$;
(3) $\|A + B\| \leq \|A\| + \|B\|$;
(4) $\|AB\| \leq \|A\| \cdot \|B\|$ (*submultiplicativity*).

Every vector norm induces what is called an operator norm via the definition

$$\|A\| = \max_{x \neq 0} \frac{\|Ax\|}{\|x\|}.$$

Equivalently,

$$\|A\| = \max_{\|x\|=1} \|Ax\|.$$

Trivially, for any operator norm

$$\|Ax\| \leq \|A\| \|x\|.$$

Any operator norm automatically satisfies conditions (1) through (3). For the submultiplicativity, if x is the vector of norm 1 such that $\|AB\| = \max_{\|x\|=1} \|ABx\|$, then

$$\|AB\| = \|ABx\| = \|A(Bx)\| \leq \|A\| \cdot \|Bx\| \leq \|A\| \cdot \|B\| \cdot \|x\| = \|A\| \cdot \|B\|.$$

The following are the induced $1, 2$, and ∞ norms for matrices.

$$\begin{aligned} \|A\|_1 &= \max_j \sum_i |a_{ij}| \\ \|A\|_\infty &= \max_i \sum_j |a_{ij}| \\ \|A\|_2 &= \max_i |\lambda_i(A^T A)| \end{aligned}$$

For symmetric matrices, clearly, $\|A\|_2 = \max_i |\lambda_i(A)|$.

The *Frobenius* norm is a matrix norm, which is not an operator norm.

$$\|A\|_F = \sqrt{\sum_{i,j=1}^n a_{ij}^2}.$$

6. Analysis of the Crank–Nicolson method for the wave equation $u_t + au_x = 0$

The Crank-Nicolson method is

$$\frac{u_{i,j+1} - u_{ij}}{k} + a\frac{1}{2}\left(\frac{u_{i+1,j} - u_{i-1,j}}{2h} + \frac{u_{i+1,j+1} - u_{i-1,j+1}}{2h}\right) = 0$$

or equivalently, for $\mu = a\frac{k}{4h}$,

$$-\mu u_{i-1,j+1} + u_{i,j+1} + \mu u_{i+1,j+1} = \mu u_{i-1,j} + u_{i,j} - \mu u_{i+1,j}.$$

For the vector $u_j = (u_{1,j}, \ldots, u_{N,j})^T$ with boundary conditions $u(0, j+1)$ and $u(N+1, j+1)$, we get

$$Bu_{j+1} = Cu_j,$$

where

$$B = \begin{bmatrix} 1 & \mu & & \\ -\mu & 1 & \ddots & \\ & \ddots & \ddots & \mu \\ & & -\mu & 1 \end{bmatrix} \quad \text{and} \quad C = \begin{bmatrix} 1 & -\mu & & \\ \mu & 1 & \ddots & \\ & \ddots & \ddots & -\mu \\ & & \mu & 1 \end{bmatrix}.$$

We have $B = I - M$, $C = I + M$, where

$$M = \begin{bmatrix} 0 & -\mu & & \\ \mu & 0 & \ddots & \\ & \ddots & \ddots & -\mu \\ & & \mu & 0 \end{bmatrix}.$$

The eigenvalues of M are $\lambda_j = 2i\mu\cos\frac{j\pi}{n+1}$. Let V be the (orthogonal) eigenvector matrix of E and let $\Lambda = \operatorname{diag}(\lambda_1, \lambda_2, \ldots, \lambda_n)$.

Then $u_{j+1} = Au_j$, where

$$\begin{aligned} A &= B^{-1}C \\ &= (I - M)^{-1}(I + M) \\ &= (I - V\Lambda V^{-1})^{-1}(I + V\Lambda V^{-1}) \\ &= (V(I - \Lambda)V^{-1})^{-1}V(I + \Lambda)V^{-1} \\ &= V(I - \Lambda)^{-1}V^{-1}V(I + \Lambda)V^{-1} \\ &= V(I - \Lambda)^{-1}(I + \Lambda)V^{-1} \end{aligned}$$

Now since for a diagonal matrix D, $\|D\|_2 = \sqrt{\lambda_{\max}(D^T D)} = |\lambda_{\max}(D)|$

$$\begin{aligned} \|A\|_2 &= \|V(I - \Lambda)^{-1}(I + \Lambda)V^{-1}\|_2 \\ &= \|(I - \Lambda)^{-1}(I + \Lambda)\|_2, \quad \text{since V is orthogonal} \\ &= \max_j \left|\frac{1 - i2\mu\cos\frac{j\pi}{n+1}}{1 + i2\mu\cos\frac{j\pi}{n+1}}\right| \\ &= \max_j \left|\frac{1 + (2\mu\cos\frac{j\pi}{n+1})^2}{1 + (2\mu\cos\frac{j\pi}{n+1})^2}\right| \\ &= 1. \end{aligned}$$

Thus, Crank–Nicolson is always stable.

7. Dissipation

We would expect the wave equation to propagate the initial condition with a constant speed a, including all frequencies that make up that initial condition.

Unfortunately the discrete nature of our data means that instead of the initial condition $u_0(x)$ we have a discrete version of it—v_0^n.

The initial condition $u_0(x)$ is a superposition (in theory) of an infinite number of frequencies (think Fourier expansion), whereas v_0^n only inherits the frequencies $\xi \in [-\pi/h, \pi/h]$. All higher frequencies are ignored by our discrete initial condition. Recall that the Fourier transform $\hat{v}^n(\xi)$ of v^n is only defined for $\xi \in [-\pi/h, \pi/h]$.

Obviously different frequencies are treated differently and we would like to get a better understanding of that treatment. Example is the best way to go here. Consider Lax–Friedrichs:

$$\frac{v_m^{n+1} - \frac{1}{2}(v_{m+1}^n + v_{m-1}^n)}{k} + a\frac{v_{m+1}^n - v_{m-1}^n}{2h} = 0,$$

equivalently,

$$v_m^{n+1} = \tfrac{1-a\lambda}{2} v_{m+1}^n + \tfrac{1+a\lambda}{2} v_{m-1}^n.$$

Von Neumann analysis implies

$$\begin{aligned} g(h\xi) &= \cos(h\xi) - ia\lambda \sin(h\xi), \\ |g(h\xi)|^2 &= \cos^2(h\xi) + (a\lambda)^2 \sin^2(h\xi). \end{aligned}$$

Let $\theta = h\xi$ as usual.

We see that $\theta = 0$ and $\theta = \pi$ are not dampened, but all other θ are. Let's observe closely. Pick $a\lambda$ to be (say) $1/2$. Then

$$v_m^{n+1} = \frac{1}{4} v_{m+1}^n + \frac{3}{4} v_{m-1}^n$$

- $\theta = \pi/2$. Then $e^{imh\xi} = e^{im\theta} = e^{im\pi/2} = \{\ldots, 1, 0, -1, 0, 1, 0, -1, 0, 1, \ldots\}$

$n=4$					$1/16$				
$n=3$				$1/8$	0	$-1/8$			
$n=2$			$1/4$	0	$-1/4$	0	$1/4$		
$n=1$		$1/2$	0	$-1/2$	0	$1/2$	0	$-1/2$	
$n=0$	1	0	-1	0	1	0	-1	0	1

- $\theta = \pi$, we have $e^{imh\xi} = e^{im\theta} = e^{im\pi} = \{\ldots, 1, -1, 1, -1, 1, \ldots\} = (-1)^m$.
  We can verify that $v_m^n = (-1)^{m+n}$ is a solution to Lax–Friedrichs, so $\theta = \pi$ is not dampened at all.

We don't really expect good results for wildly oscillating solutions, so we can expect that the higher frequencies will not be well-represented in our calculation. However it is unacceptable for higher frequencies to be less dampened than the middle-range ones.

Another example. Look at Lax–Wendroff: $|g|^2 = 1 - 4a^2\lambda^2(1 - a^2\lambda^2)\sin^4\frac{\theta}{2} \le 1 - \text{const} \cdot \sin^4\frac{\theta}{2}$.

This is very important—says that all frequencies, except $\xi = 0$ (then $\theta = 0$) are decreasing and the highest frequencies are suppressed the most. This is exactly what we want and will call schemes that have this property dissipative.

Definition 7.1 (Dissipative Scheme)**.** A scheme is dissipative of order $2r$ if

$$|g(\theta)| \le 1 - c \cdot \sin^{2r}\frac{\theta}{2}.$$

The reason we like dissipative schemes is that if we are not doing a good job with the high frequencies anyway, why not kill them.

Remark 7.2. A dissipative scheme is always stable.

How can we make a non-dissipative scheme dissipative? This calls for another example. Crank–Nicolson, which is second order accurate.

$$\frac{v_m^{n+1}-v_m^n}{k}+a\frac{v_{m+1}^{n+1}-v_{m-1}^{n+1}+v_{m+1}^n-v_{m-1}^n}{4h}=0$$

So adding a fourth derivative in there will not affect the order of accuracy of the approximation, since fourth derivatives get ignored anyway. When we do the Fourier analysis the fourth derivative will bring a $\sin^4\frac{\theta}{2}$.

$$\frac{v_m^{n+1}-v_m^n}{k}+a\frac{v_{m+1}^{n+1}-v_{m-1}^{n+1}+v_{m+1}^n-v_{m-1}^n}{4h}+C\epsilon\frac{v_{m-2}^n-4v_{m-1}^n+6v_m^n-4v_{m+1}^n+v_{m+2}^n}{h^4}=0$$

Now select C appropriately so that the fourth derivative only brings $\sin^4\frac{\theta}{2}$ into the picture, without any weird powers of k and h: $C=\frac{h^4}{16k}$. Then after some simplification

$$g(\theta)=\frac{1-\epsilon\sin^4\frac{\theta}{2}-i\frac{a\lambda}{2}\sin\theta}{1+i\frac{a\lambda}{2}\sin\theta}$$

implying

$$\begin{aligned}
|g(\theta)|^2 &= \frac{1+\left(\frac{a\lambda}{2}\sin\theta\right)^2-2\epsilon\sin^4\frac{\theta}{2}+\epsilon^2\sin^8\frac{\theta}{2}}{1+\left(\frac{a\lambda}{2}\sin\theta\right)^2}\\
&= 1-\frac{\epsilon\sin^4\frac{\theta}{2}-\overbrace{\epsilon\sin^4\frac{\theta}{2}(1-\epsilon\sin^4\frac{\theta}{2})}^{>0 \text{ for } \epsilon<1}}{1+\left(\frac{a\lambda}{2}\sin\theta\right)^2}\\
&\le 1-\frac{\epsilon\sin^4\frac{\theta}{2}}{1+\left(\frac{a\lambda}{2}\sin\theta\right)^2}.
\end{aligned}$$

If we now restrict $|a\lambda|$ (say) ≤ 10, then $1+\left(\frac{a\lambda}{2}\sin\theta\right)^2\le 26$, and

$$|g|^2\le 1-\frac{\epsilon}{26}\sin^4\frac{\theta}{2}.$$

We want a bound on $|g|$, not $|g|^2$:

$$|g|^2\le 1-\frac{\epsilon}{26}\sin^4\frac{\theta}{2}\le|g|^2\le 1-\frac{\epsilon}{26}\sin^4\frac{\theta}{2}+\left(\frac{\epsilon}{52}\right)^2\sin^8\frac{\theta}{2}=\left(1-\frac{\epsilon}{52}\sin^4\frac{\theta}{2}\right)^2,$$

so

$$|g|\le 1-\frac{\epsilon}{52}\sin^4\frac{\theta}{2}.$$

The scheme is all of a sudden dissipative of order 4 (since $2r=4$). Although Crank–Nicolson is stable for all $a\lambda$ we cannot make it dissipative without restricting $a\lambda$.

The exact same trick works for Leap-frog.

8. Dispersion

In this section we investigate whether in the numerical solution of $u_t + au_x = 0$ different frequencies travel with the same speed a as they should. They, of course, do not and we will see that in fact travel with speed $\alpha(h\xi) \approx a$.

Look for a solution to

$$u_t + au_x = 0, \qquad u(0,x) = f(x)$$

(which has a unique solution $u(t,x) = f(x-at)$) using separation of variables

$$u(t,x) = g(t)e^{ix\xi},$$

assuming $u(0,x) = g(0)e^{ix\xi} = e^{ix\xi}$ =periodic wave (here we assume $g(0) = 1$). Then

$$u_t + au_x = g'(t)e^{ix\xi} + ag(t)i\xi e^{ix\xi} = (g'(t) + ai\xi g(t))e^{ix\xi} = 0.$$

Since $|e^{ix\xi}| = 1$ we have $g'(t) + ai\xi g(t) = 0$, which implies $g(t) = e^{-iat\xi}g(0)$. Insert back and get

$$u(t,x) = g(0)e^{-iat\xi}e^{ix\xi} = e^{i(x-at)\xi},$$

since $g(0) = 1$.

Therefore the initial condition is translated with speed a for all ξ.

Definition 8.1 (Dispersion)**.** The phenomenon of waves with different frequencies moving with different speeds is called **dispersion**.

Return now to the solution of the difference equation. Take Lax–Friedrichs:

$$v_m^{n+1} = \frac{1}{2}(v_{m+1}^n + v_{m-1}^n) - \frac{a\lambda}{2}(v_{m+1}^n - v_{m-1}^n).$$

Separation of variables: $v_m^n = g^n e^{imh\xi}$ and substitute above to get

$$g = \cos(h\xi) - ia\lambda\sin(h\xi),$$

so the solution is

$$v_m^n = (\cos(h\xi) - ia\lambda\sin(h\xi))^n e^{imh\xi},$$

which looks nothing like $e^{i(x-at)\xi}$. Let

$$g(h\xi) \equiv \rho e^{-i\omega} = \rho\cos\omega - \rho i\sin\omega = \cos(h\xi) - ia\lambda\sin(h\xi).$$

Therefore $\tan\omega = a\lambda\tan(h\xi)$, $\quad \rho^2 = \cos^2(h\xi) + a^2\lambda^2\sin^2(h\xi)$. For $|h\xi| \le \pi/2$ we have

$$v_m^n = (\rho e^{-i\omega})^n e^{imh\xi} = \rho^n e^{imh\xi - i\omega n} = \rho^n e^{i(mh - \omega n/\xi)\xi} = \rho^n e^{i(x - \frac{\omega n}{\xi}\frac{t}{nk})\xi} = \rho^n e^{i(x - \frac{\omega}{k\xi}t)\xi} = \rho^n e^{i(x-\alpha(h\xi)t)\xi},$$

where $x = mh, t = nk$ and

$$\alpha(h\xi) \equiv \frac{\omega}{k\xi} = \frac{\arctan(a\lambda\tan(h\xi))}{\lambda h\xi}$$

(Recall $\tan\epsilon \approx \epsilon$ and $\arctan\epsilon \approx \epsilon$ so $\alpha \approx \frac{\arctan(a\lambda\tan(h\xi))}{\lambda h\xi} \approx a$.)

We have

$$v_m^n = |g(h\xi)|^n \cdot e^{i(x-\alpha(h\xi)t)\xi}.$$

Definition 8.2 (Phase speed)**.** The quantity $\alpha(h\xi)$ is called **phase speed**, and is the speed at which waves of frequency ξ are propagated by the difference scheme.

Once again, waves with different frequencies travel with different speeds. Thus we say that the scheme is *dispersive*. We want the scheme to be dispersive as little as possible (i.e., $\alpha(h\xi) \approx a$), so that the numerical solution looks like the exact solution.

Time to study the Taylor series for $\alpha(h\xi)$ to obtain a better estimate of the closeness to a.

$$\begin{aligned} \tan z &= z + \frac{1}{3}z^3 + O(z^5) \\ \arctan z &= z - \frac{1}{3}z^3 + O(z^5) \end{aligned}$$

Let $z = h\xi$

$$\begin{aligned} \alpha(z) &= \frac{\arctan(a\lambda \tan z)}{\lambda z} \\ &= \frac{a\lambda \tan z}{\lambda z} - \frac{(a\lambda \tan z)^3}{3\lambda z} + \dots \\ &= a \cdot \frac{z + z^3/3 + \dots}{z} - \frac{a^3\lambda^3}{\lambda}\frac{z^3}{3z} + \dots \\ &= a\left(1 + (1 - a^2\lambda^2)\frac{(h\xi)^2}{3} + \dots\right) \end{aligned}$$

So, if ξ is given and h is small, then the wave speed is slightly higher than a, and the high frequencies travel fastest. Let's look at some special cases.

Take $h\xi = \pi/2$. Then $\omega = \pi/2$ and $\rho = |a\lambda|$, so

$$v_m^n = |a\lambda|^n e^{imh\xi} \cdot e^{-in\pi/2} = |a\lambda|^n e^{i(x\xi - \pi n/2)} = |a\lambda|^n e^{i(x - t/\lambda)\xi}$$

(since $n\pi/2 = (t/k)(h\xi) = t\xi/\lambda$). So the speeds can be quite different. Exact $= a$; Computed $= 1/\lambda = \frac{a}{a\lambda}$, so it is not a good idea to take $a\lambda$ small. The closer to the stability limit (i.e., the closer to 1) the better.

9. Stability of the Jacobi's method

When solving $Ax = b$ the idea is to split $A = B + C$ so $Ax = b$ can be written as $(B + C)x = b$, or

$$x = -B^{-1}Cx + B^{-1}b$$

which we iterate as

$$x^{k+1} = -B^{-1}Cx^k + B^{-1}b.$$

The success of this approach hinges on the choice of B so that the error $e^k = x^k - x$ decreases at every step. Since $e^k = (-B^{-1}C)^k e^0$, we must have $\rho(-B^{-1}C) < 1$, where ρ is the spectral radius. How many steps would it take to reduce the error in half? We will use $1/e$ instead of $1/2$ for simplicity.

$$\rho^m = 1/e$$

implies $m = -1/\ln \rho$.

Consider now the Jacobi's method to solving $u_{xx} + u_{yy} = f(x, y)$, which is discretized as

$$4v_{ij} - v_{i-1,j} - v_{i+1,j} - v_{i,j-1} - v_{i,j+1} = h^2 f_{ij} \tag{9.1}$$

Over a square mesh with step size h and N unknowns v_{ij}, $i, j = 1, \ldots, N$.

Let

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

be $N \times N$.

In matrix form the equation (9.1) can be written as $T_{N\times N} v = b$. Jacobi's method

$$v_{ij}^{k+1} = (v_{i-1,j}^k + v_{i+1,j}^k + v_{i,j-1}^k + v_{i,j+1}^k + h^2 f_{ij})/4$$

can be written as $v^{k+1} = (I_{N\times N} - T_{N\times N}/4)v^k + h^2 f/4$, where

$$T_{N\times N} = \begin{bmatrix} T_N + 2I_N & -I_N & & \\ -I_N & T_N + 2I_N & \ddots & \\ & \ddots & \ddots & -I_N \\ & & -I_N & T_N + 2I_N \end{bmatrix}.$$

9.1. The eigenvalues of $T_{N\times N}$.

The analysis here follows Demmel [1], Chapters 6.3 and 6.5.

If λ_i, z^i are the eigenvalues of and eigenvectors of T_N, then the eigenvalues and eigenvectors of $T_{N\times N}$ are $\lambda_i + \lambda_j$ for all i, j and the eigenvector corresponding to $\lambda_i + \lambda_j$ is

$$z = \begin{bmatrix} z_1^i z^j \\ z_2^i z^j \\ \vdots \\ z_N^i z^j \end{bmatrix}.$$

Proof: We first note that $T_N z^i = \lambda_i z^i$ means

$$\begin{aligned} 2z_1^i - z_2^i &= \lambda_i z_1^i \\ -z_1^i + 2z_2^i - z_3^i &= \lambda_i z_2^i \\ &\vdots \\ -z_{N-1}^i + 2z_N^i &= \lambda_i z_N^i. \end{aligned}$$

Now

$$(T_{N\times N})z = \begin{bmatrix} T_N + 2I_N & -I_N & & \\ -I_N & T_N + 2I_N & \ddots & \\ & \ddots & \ddots & -I_N \\ & & -I_N & T_N + 2I_N \end{bmatrix} \begin{bmatrix} z_1^i z^j \\ z_2^i z^j \\ \vdots \\ z_N^i z^j \end{bmatrix}$$

$$= \begin{bmatrix} (T_N + 2I_N)z_1^i z^j + I_N z_2^i z^j \\ -I_N z_1^i z^j + (T_N + 2I_N)z_2^i z^j - I_N z_3^i z^j \\ \vdots \\ -I_N z_{N-1}^i z^j + (T_N + 2I_N)z_N^i z^j \end{bmatrix}$$

$$= \begin{bmatrix} z_1^i T_N z^j + 2z_1^i z^j + z_2^i z^j \\ -z_1^i z^j + T_N z_2^i z^j + 2z_2^i z^j - z_3^i z^j \\ \vdots \\ -z_{N-1}^i z^j + T_N z_N^i z^j + 2z_N^i z^j \end{bmatrix}$$

$$= \begin{bmatrix} \lambda_j z_1^i z^j + (2z_1^i + z_2^i)z^j \\ \lambda_j z_2^i z^j + (-z_1^i + 2z_2^i - z_3^i)z^j \\ \vdots \\ \lambda_j z_N^i z^j + (-z_{N-1}^i + 2z_N^i)z^j \end{bmatrix}$$

$$= \begin{bmatrix} \lambda_j z_1^i z^j + \lambda_i z_1^i z^j \\ \lambda_j z_2^i z^j + \lambda_i z_2^i z^j \\ \vdots \\ \lambda_j z_N^i z^j + \lambda_i z_N^i z^j \end{bmatrix}$$

$$= (\lambda_i + \lambda_j)z$$

So the eigenvalues of $T_{N\times N}$ are $\lambda_i + \lambda_j$ where $\lambda_i = 2 - 2\cos\frac{\pi i}{N+1}$.

9.2. **Speed of convergence of Jacobi.** Since Jacobi splits $T_{N\times N}$ as $4I - (4I - T_{N\times N})$ and the transition matrix is $(4I)^{-1}(4I - T_{N\times N}) = I - T_{N\times N}/4$.

The largest eigenvalue of the latter is the largest of $|1 - (\lambda_i + \lambda_j)/4|$, namely

$$\rho = |1 - (\lambda_1 + \lambda_1)/4| = \cos\frac{\pi}{N+1} \approx 1 - \frac{\pi^2}{2(N+1)^2}.$$

(Recall that $\cos x = 1 - x^2/2! + x^4/4! + \ldots$)

How many steps would it take Jacobi to decrease the error by a certain factor, say e^{-1}? We must have $\rho^m = e^{-1}$, i.e.,

$$\left(1 - \frac{\pi^2}{2(N+1)^2}\right)^m = e^{-1}$$

meaning

$$m \ln\left(1 - \frac{\pi^2}{2(N+1)^2}\right) = -1.$$

Using that for small x, $\ln x \approx 1 - x$ we get

$$m \approx \frac{2(N+1)^2}{\pi^2} = \mathcal{O}(N^2).$$

9.3. **Gauss–Seidel and SOR.** It can be proven that for Gauss–Seidel with red-black ordering the spectral radius of the transition matrix is

$$\rho_{GS} = \cos^2 \frac{\pi}{N+1} = (\rho_J)^2,$$

i.e., Gauss–Seidel will take half the steps that Jacobi takes. This matches our experiments.

For SOR with parameter $w = \frac{2}{1+\sin\frac{\pi}{N+1}}$ one can prove that

$$\rho_{SOR} = \frac{\cos^2\frac{\pi}{N+1}}{\left(1+\sin\frac{\pi}{N+1}\right)^2} \approx 1 - \frac{2\pi}{N+1}$$

so SOR will take $\mathcal{O}(N)$ steps only to converge which once again confirms our observations.

10. Solving the Heat Equation Using the Fast Fourier Transform [1]

One can solve the heat equation $T_{N\times N}\cdot x = b$ using the Fast Fourier Transform in $\mathcal{O}(N^2\log N)$ time.

We established that the eigenvector matrix of $T_{N\times N}$ is

$$V = [z_{jk}Z]_{j,k=1}^{N},$$

where

$$Z = \left[\sin\frac{(j+1)(k+1)\pi}{N+1}\right]_{j,k=0}^{N-1}$$

is the eigenvector matrix of T_N. Since the norm of each column of V is $\frac{N+1}{2}$ and V is symmetric,

$$V^{-1} = \frac{1}{(\frac{N+1}{2})^2}V.$$

The eigenvalue matrix of $T_{N\times N}$ is $\Lambda_{N\times N} = \operatorname{diag}(4-2\cos\frac{j\pi}{N+1}-2\cos\frac{k\pi}{N+1})_{j,k=1}^{n}$.

Thus $T_{N\times N}\cdot x = b$ means $V\Lambda_{N\times N}V^{-1}x = b$, i.e.,

$$x = \frac{1}{(\frac{N+1}{2})^2}V\Lambda_{N\times N}^{-1}Vb.$$

The key point here is that using the FFT we can multiply by V in $\mathcal{O}(N^2\log N)$ time.

10.1. The Discrete Fourier Transform.

For the rest of this section $i = \sqrt{-1}$.

The matrix of the $N\times N$ Discrete Fourier Transform (DFT) is the Vandermonde matrix

$$\Phi_N = [w^{jk}]_{j,k=0}^{N-1},$$

where $w = e^{-\frac{2\pi}{N}} = \cos\frac{2\pi}{N} - i\sin\frac{2\pi}{N}$ is a principal Nth root of unity.

Its utility comes from:

(1) $\Phi^{-1} = \frac{1}{N}\bar{\Phi}$.
(2) Z is the imaginary part of $-\Phi_{2N+2}(2:N+1, 2:N+1)$; also $Z^{-1} = \frac{1}{N}Z$ since Z is symmetric.
(3) One can multiply by Φ in $\mathcal{O}(N\log N)$ time (vs. $\mathcal{O}(N^2)$ for conventional matrix-vector multiplication).

Proof of (1): Since $\bar{w} = w^{-1}$ and $w^N = 1$,

$$(\Phi\bar{\Phi})_{lj} = \sum_{k=0}^{N-1}\phi_{lk}\bar{\phi}_{kj} = \sum_{k=0}^{N-1}w^{lk}\bar{w}^{kj} = \sum_{k=0}^{N-1}w^{k(l-j)}.$$

This is the sum of N ones if $l = j$ and a geometric sum if $l\neq j$ with value $\frac{1-w^{N(l-j)}}{1-w^{l-j}} = 0$.

Part (2) is obvious. Part (3) is the Fast Fourier Transform (FFT).

10.2. The FFT.

Assume for simplicity that $N = 2^m$.

If a is the vector $a = [a_0,\ldots,a_{N-1}]^T$. Forming the product Φa means we need to evaluate the polynomial

$$a(x) = a_0 + a_1x + \cdots a_{N-1}x^{N-1}$$

at all N roots of unity $x = w^j$, $j = 0,1,\ldots,N-1$.

If we write

$$\begin{aligned} a(x) &= a_0 + a_1x + \cdots a_{N-1}x^{N-1} \\ &= (a_0 + a_2x^2 + a_4x^4 + \cdots) + x(a_1 + a_3x^2 + a_5x^4 + \cdots) \\ &= a_{\text{even}}(x^2) + x\cdot a_{\text{odd}}(x^2). \end{aligned}$$

Thus we need to evaluate two polynomials a_{even} and a_{odd} at $(w^j)^2$, $j = 0, 1, \ldots, N-1$. The computational savings come from the recognition that $(w^j)^2$, $j = 0, 1, \ldots, N-1$ are just $N/2$, not N, points, just repeated twice since $w^{2j} = w^{2(j+\frac{N}{2})}$.

Thus computing the FFT of a vector of size N is the same as computing two FFTs of size $N/2$ and combining the results with $N/2$ multiplications and N additions.

Here is the algorithm:

```
function FFT(a)
if length(a)=1
    return a
else
    a1=FFT(a_even)
    a2=FFT(a_odd)
    w = e^(2πi/N)
    u = [w^0, w^1, ..., w^(N/2-1)]
    return [a1 + u.*a2, a1 - u.*a2]          ... the multiplication is meant componentwise
end
```

We used the fact that $w^{j+N/2} = w^j$.

For the cost $C(N)$ of the algorithm, we assume the powers of w are precomputed and stored, so we have

$$C(N) = 2C\left(\tfrac{N}{2}\right) + \tfrac{3N}{2} = 4C\left(\tfrac{N}{4}\right) + 2\tfrac{3N}{2} = 8C\left(\tfrac{N}{8}\right) + 3\tfrac{3N}{2} = \cdots = \log_2 N \cdot \tfrac{3N}{2}.$$

10.3. **Multiplication by V.** We now explain how to perform the multiplication Vx in $\mathcal{O}(N \log N)$ time. Since V is $N^2 \times N^2$ and x is $N^2 \times 1$, we partition x into sections of size N:

$$x = \begin{bmatrix} x^1 \\ x^2 \\ \vdots \\ x^N \end{bmatrix},$$

where each

$$x^i = \begin{bmatrix} x_1^i \\ x_2 \\ \vdots \\ x_N^i \end{bmatrix}$$

is an $N \times 1$ vector (for a total of $N^2 \times 1$ for x). Let $y^i = Zx^i$, $i = 1, 2, \ldots, N$. Thus

$$\begin{aligned} Vx = [z_{jk}Z]_{j,k=1}^N \cdot x &= \begin{bmatrix} z_{11}Z & z_{12}Z & \cdots & z_{1N}Z \\ z_{21}Z & z_{22}Z & \cdots & z_{2N}Z \\ \vdots & \vdots & \ddots & \vdots \\ z_{N1}Z & z_{N2}Z & \cdots & z_{NN}Z \end{bmatrix} \cdot \begin{bmatrix} x^1 \\ x^2 \\ \vdots \\ x^N \end{bmatrix} \\ &= \begin{bmatrix} z_{11}Zx^1 + z_{12}Zx^2 + \cdots + z_{1N}Zx^N \\ z_{21}Zx^1 + z_{22}Zx^2 + \cdots + z_{2N}Zx^N \\ \vdots \\ z_{N1}Zx^1 + z_{N2}Zx^2 + \cdots + z_{NN}Zx^N \end{bmatrix} \\ &= \left[\begin{array}{c} z_{11}y^1 + z_{12}y^2 + \cdots + z_{1N}y^N \\ \hline z_{21}y^1 + z_{22}y^2 + \cdots + z_{2N}y^N \\ \hline \vdots \\ \hline z_{N1}y^1 + z_{N2}y^2 + \cdots + z_{NN}y^N \end{array}\right] \\ &= \left[\begin{array}{c} z_{11}y_1^1 + z_{12}y_1^2 + \cdots + z_{1N}y_1^N \\ z_{11}y_2^1 + z_{12}y_2^2 + \cdots + z_{1N}y_2^N \\ \vdots \\ z_{11}y_N^1 + z_{12}y_N^2 + \cdots + z_{1N}y_N^N \\ \hline z_{21}y_1^1 + z_{22}y_1^2 + \cdots + z_{2N}y_1^N \\ z_{21}y_2^1 + z_{22}y_2^2 + \cdots + z_{2N}y_2^N \\ \vdots \\ z_{21}y_N^1 + z_{22}y_N^2 + \cdots + z_{2N}y_N^N \\ \hline \vdots \\ \hline z_{N1}y_1^1 + z_{N2}y_1^2 + \cdots + z_{NN}y_1^N \\ z_{N1}y_2^1 + z_{N2}y_2^2 + \cdots + z_{NN}y_2^N \\ \vdots \\ z_{N1}y_N^1 + z_{N2}y_N^2 + \cdots + z_{NN}y_N^N \end{array}\right]. \end{aligned}$$

The trick now is to recognize that the N entries in positions $1, N+1, 2N+1, 3N+1, ...$ are just

$$Z \cdot \begin{bmatrix} y_1^1 \\ y_1^2 \\ \vdots \\ y_1^N \end{bmatrix} = \begin{bmatrix} z_{11} & z_{12} & \cdots & z_{1N} \\ z_{21} & z_{22} & \cdots & z_{2N} \\ \vdots & \vdots & \ddots & \vdots \\ z_{N1} & z_{N2} & \cdots & z_{NN} \end{bmatrix} \cdot \begin{bmatrix} y_1^1 \\ y_1^2 \\ \vdots \\ y_1^N \end{bmatrix} = \begin{bmatrix} z_{11}y_1^1 + z_{12}y_1^2 + \cdots + z_{1N}y_1^N \\ z_{21}y_1^1 + z_{22}y_1^2 + \cdots + z_{2N}y_1^N \\ \vdots \\ z_{N1}y_1^1 + z_{N2}y_1^2 + \cdots + z_{NN}y_1^N \end{bmatrix}.$$

Similarly, we get the entries in positions $2, N+2, 2N+2, \ldots$.

If `Zb(x)` is the MATLAB routine that takes a vector x as an input and returns Zx, then the multiplication by V by an $N^2 \times 1$ vector x is done as follows:

```
function y=ZNNx(x)
N=√length(x)
for i=1:N
    y((i-1)*N+1:i*N)=Zb(x((i-1)*N+1:i*N))
end
for i=1:N
    y(i:N:end)=Zb(y(i:N:end))
end
```

10.4. **The right hand side of $T_{N\times N}x = b$.** We saw that solving the heat equation in 2D means solving $T_{N\times N}x = b$ for a certain right hand side b. We will establish its exact form.

Say, we're solving the heat equation on $[0,1]\times[0,1]$, subdivided into $N+1$ parts so that in both the x and y direction we have $N+2$ points—2 for boundary conditions and N unknown. Say all data is in a $(N+2)\times(N+2)$ matrix u with the unknown values in $u(2:N+1, 2:N+1)$. The boundary conditions are $u(:,1), u(:,N+2), u(1,:)$ and $u(N+2,:)$.

Here are all our equations for the $u'_{ij}s$:

$$
\begin{aligned}
4u_{22} - u_{21} - u_{23} - u_{12} - u_{32} &= 0\\
4u_{23} - u_{22} - u_{24} - u_{13} - u_{33} &= 0\\
&\vdots\\
4u_{2,N} - u_{2,N-1} - u_{2,N+1} - u_{1,N} - u_{3,N} &= 0\\
4u_{2,N+1} - u_{2N} - u_{2,N+2} - u_{1,N+1} - u_{3,N+1} &= 0\\
\hline
4u_{32} - u_{31} - u_{33} - u_{22} - u_{42} &= 0\\
4u_{33} - u_{32} - u_{34} - u_{23} - u_{43} &= 0\\
&\vdots\\
4u_{3,N} - u_{3,N-1} - u_{3,N+1} - u_{2,N} - u_{4,N} &= 0\\
4u_{3,N+1} - u_{3N} - u_{3,N+2} - u_{2,N+1} - u_{4,N+1} &= 0\\
\hline
&\vdots\\
\hline
4u_{N2} - u_{N1} - u_{N3} - u_{N-1,2} - u_{N+1,2} &= 0\\
4u_{N3} - u_{N2} - u_{N4} - u_{N-1,3} - u_{N+1,3} &= 0\\
&\vdots\\
4u_{N,N} - u_{N,N-1} - u_{N,N+1} - u_{N-1,N} - u_{N+1,N} &= 0\\
4u_{N,N+1} - u_{NN} - u_{N,N+2} - u_{N+1,N+1} - u_{N+1,N+1} &= 0\\
\hline
4u_{N+1,2} - u_{N+1,1} - u_{N+1,3} - u_{N2} - u_{N+2,2} &= 0\\
4u_{N+1,3} - u_{N+1,2} - u_{N+1,4} - u_{N3} - u_{N+2,3} &= 0\\
&\vdots\\
4u_{N+1,N} - u_{N+1,N-1} - u_{N+1,N+1} - u_{NN} - u_{N+2,N} &= 0\\
4u_{N+1,N+1} - u_{N+1,N} - u_{N+1,N+2} - u_{N,N+1} - u_{N+2,N+1} &= 0
\end{aligned}
$$

Careful inspection reveals that the right hand side is then

$$T_{N\times N}x = \left[\begin{array}{c} u_{12}+u_{21} \\ u_{13} \\ \vdots \\ u_{1N} \\ \hline u_{1,N+1}+u_{2,N+2} \\ \hline u_{31} \\ 0 \\ \vdots \\ 0 \\ \hline u_{3,N+2} \\ \hline \vdots \\ \hline u_{N,1} \\ 0 \\ \vdots \\ 0 \\ \hline u_{N,N+2} \\ \hline u_{N+2,2}+u_{N+1,1} \\ u_{N+2,3} \\ \vdots \\ u_{N+2,N} \\ u_{N+2,N+1}+u_{N+1,N+2} \end{array}\right]$$

References

1. J. W. Demmel, *Applied Numerical Linear Algebra*, SIAM, Philadelphia, 1997. MR MR1463942 (98m:65001)