5374 Fall '22

Numerical Linear Algebra Lecture 3

The Scientific Computing Philosophy

"Numerical analysis is the study of algorithms for the problems of continuum mathematics" - Nick Trefethen.

In mathematics we often deal with problems whose solutions we can describe very well qualitatively and even provide representation formulas for BUT which we are not able to compute or approximate in a practical way.

Here is where the theory can be used to develop algorithm that don't provide necessarily a closed and elegant formula but which provide a numerical solution in a reasonable amount of time.

Let's illustrate this philosophy with a

few examples.

Example 1 : Solving $Ax=b$ via Cramer's rule

$$\begin{pmatrix} a_{11} & \cdots & a_{1n} \\ \vdots & & \\ a_{n1} & \cdots & a_{nn} \end{pmatrix} \begin{pmatrix} x_1 \\ \vdots \\ x_n \end{pmatrix} = \begin{pmatrix} b_1 \\ \vdots \\ b_n \end{pmatrix}$$

The solution $x = \begin{pmatrix} x_1 \\ \vdots \\ x_n \end{pmatrix}$ can be obtained by computing n different determinants, where the k-th determinant is obtained by swapping the k-th column of A with the vector b.

Equivalently, Cramer's rule provides a formula for A^{-1}, obtained by computing n^2 different determinants of matrices of size $(n-1)\times(n-1)$ each.

Computing a $k\times k$ determinant requires at least $O(k!)$ operations, so we are talking about at least $n\times n!$ operations to solve $Ax=b$ via Cramer's rule.

For a system as small as $n=30$, this gives us at least 10^{31} operations.

Today, we daily solve equation $Ax=b$ where A is $n \times n$ with n in the thousand or even more

For all but the smallest matrices Cramer's rule is a no-go for solving $Ax=b$.

Note however that for $n=2$ the formula is pretty convenient:

if $A = \begin{pmatrix} a & b \\ c & d \end{pmatrix}$ and $ad-bc \neq 0$

then $$A^{-1} = \frac{1}{ad-bc} \begin{pmatrix} d & -b \\ -c & a \end{pmatrix}$$

Example 2 : The back-substitution algorithm

Recall that a matrix $A = (a_{ij})$ is called upper triangular if $a_{ij} = 0$ whenever $i > j$

$$A = \begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ 0 & a_{22} & a_{23} \cdots & a_{2n} \\ 0 & 0 & a_{33} \cdots & \\ \vdots & & & \\ 0 & \cdots & 0 & a_{nn} \end{pmatrix}$$

For upper triangular matrices it is straightforward to solve any equation $Ax=b$ via an algorithm called back-substitution.

The algorithm only works if A ~~B~~, in addition of being upper triangular, is also invertible. If one looks at the formula for the determinant of A, one see that

$$(*) \quad \det A = \sum_{\sigma \in S_n} (-1)^{\operatorname{sgn}(\sigma)} a_{1\sigma_1} a_{2\sigma_2} \cdots a_{n\sigma_n}$$

If $\sigma \in S_n$ is not $\sigma_i = i \ \forall i$, then there will be some j (depends on σ) such $j > \sigma_j$, in which case as A is upper triangular $a_{j\sigma_j} = 0$, so in $(*)$, of the $n!$ term on the right only one is non-zero, that is

$$\det A = a_{11} a_{22} \cdots a_{nn}$$

Therefore, A is invertible $\Leftrightarrow a_{ii} \neq 0 \ \forall i = 1, \ldots, n$ (when A is upper triangular)

So now that we know all the diagonal

elements of A are non-zero, we can describe the back-substitution algorithm

$$\begin{pmatrix} a_{11} & a_{12} & \cdots & & a_{1n} \\ 0 & a_{22} & \cdots & & a_{2n} \\ \vdots & & \ddots & & a_{n-2\,n} \\ & & & a_{n-1\,n-1} & a_{n-1\,n} \\ 0 & \cdots & & 0 & a_{nn} \end{pmatrix} \begin{pmatrix} x_1 \\ \vdots \\ x_n \end{pmatrix} = \begin{pmatrix} b_1 \\ \vdots \\ b_n \end{pmatrix}$$

We work our way from the bottom up in a series of steps. (n steps total)

Step 1 : $a_{nn} x_n = b_n$

So we set $x_n := \dfrac{b_n}{a_{nn}}$

Step 2 : We have the equation

$$a_{n-1\,n-1} x_{n-1} + a_{n-1\,n} x_n = b_{n-1}$$

We know x_n now, so we solve for x_{n-1}, using that $a_{n-1\,n-1} \neq 0$:

$$x_{n-1} = \frac{b_{n-1} - a_{n-1\,n} x_n}{a_{n-1\,n-1}}$$

$\vdots$

$(k \geq 2)$

Step k By now I have computed

$x_n, x_{n-1}, \dots, x_{n-k+2}$

We take the equation

$$a_{n-k+1\, n-k+1}\, x_{n-k+1} + \sum_{j=n-k+2}^{n} a_{n-k+1\, j}\, x_j = b_{n-k+1}$$

So we get

$$x_{n-k+1} = \frac{b_{n-k+1} - \sum_{j=n-k+2}^{n} a_{n-k+1\, j}\, x_j}{a_{n-k+1\, n-k+1}}$$

After step n, we have arrived at the answer $\begin{pmatrix} x_1 \\ \vdots \\ x_n \end{pmatrix}$.

Let us estimate the # of operations done to complete the algorithm. At each step the total number of products, sums, and division performed is no larger than $2n+2$. With a total of $2n^2+2n$ operations for the whole algorithm.

For upper triangular matrices:

Back-substitution : $\leq 2n^2 + 2n$ operations

$\mathcal{O}(n^2)$ operations

Cramer's rule : $\geq n \cdot n!$ operations.

Example 3 : Observe that computing the inverse of a $n \times n$ upper triangular matrix can be done with $\mathcal{O}(n^3)$ arithmetic operations, simply apply back substitution n times:

Solve $A x_k = e_k$ for $k = 1, \dots, n$

where $e_1, \dots, e_n$ are the canonical basis vectors of $\mathbb{R}^n$

$$\begin{pmatrix} 1 \\ 0 \\ 0 \\ \vdots \end{pmatrix}, \begin{pmatrix} 0 \\ 1 \\ 0 \\ \vdots \\ 0 \end{pmatrix}, \dots, \begin{pmatrix} 0 \\ \vdots \\ 1 \end{pmatrix}$$

Then the n-vectors $x_1, \dots, x_n$ provide the n columns of the inverse matrix A^{-1}.

$$A^{-1} = \left(x_1 \middle| x_2 \middle| \cdots \middle| x_n \right)$$

Note: Numerically speaking, solving $Ax=b$, while the answer might be $x=A^{-1}b$, should, as a rule of thumb, be done without computing A^{-1}.

Example 4 (the QR decomposition)

Any $n\times n$ matrix A can be factored as a product of two matrices

$$A = QR$$

where Q is an orthogonal matrix

$$(\; Q^{-1} = Q^t \;)$$

and R is upper triangular.

Later this semester we will study two algorithms, the Gram-Schmidt process and Householder orthogonalization, that take as input the matrix A and produce as output Q and R as above.

We will see these algorithm require only $O(n^3)$ arithmetic operation

Example 5 (solving $Ax=b$, again)

The QR decomposition algorithms and the back-substitution algorithm provide when combined an algorithm to solve $Ax=b$ in $O(n^3)$ time without requiring A be upper triangular.

Take $Ax=b$, A invertible

Step 1 Use Householder orthogonalization to find Q and R s.t. $A=QR$ ($O(n^3)$ arithmetic operations)

Step 2 Our equation becomes

$$QRx=b$$

Q is orthogonal, so it is invertible, and $Q^{-1}=Q^t$, so

$$Rx=Q^{-1}b=Q^tb.$$

So in this step we compute the product Q^tb

$y := Q^tb$ (cost: $O(n^2)$ arithmetic operations)

Step 3 Apply back substitution to solve

$$Rx = y$$

(cost : $O(n^2)$ arithmetic operations)

As a total, we have

$$O(n^3) + O(n^2) + O(n^2) = O(n^3)$$

operations.

(From here one could also compute A^{-1} in $O(n^3)$ operations)