eman ta zabal zazu

Universidad del País Vasco Euskal Herriko Unibertsitatea

ZIENTZIA ETA TEKNOLOGIA FAKULTATEA
FACULTAD DE CIENCIA Y TECNOLOGÍA

7. Time complexity of Numerical Methods.

Jon Asier Bárcena-Petisco (University of the Basque Country UPV/EHU) - Numerical Methods I (2023/24)

Contents on this chapter

- Asymptotic notation.
- Theoretical study of time complexity of algorithms.
- Time complexity of the matrix multiplication.

7.1. Bachmann–Landau asymptotic notation.

Introduction

When studying an algorithm, it is interesting to know the time it takes to terminate in an abstract way, that do not depend on the computer. For that we need to define what we consider an elementary step (for instance, the number of additions, multiplications, etc.), and then count the number of elementary steps that are required. However, getting and exact number is not easy at all, nor the objective. Usually, it suffices to get an approximate answer when the size of the input is large.

Essentially positive sequences

Definition

A sequence $a_n \in \mathbb{R}$ is *essentially positive* if there is $N > 0$ such that $a_n \geq 0$ for all $n \geq N$.

Example

- All positive sequences are essentially positive.
- $a_n = n^2 - 4n + 1$ is essentially positive.

Bachmann-Landau $\mathcal{O}$ notation.

Definition

Let $a_n \in \mathbb{R}$ be a essentially positive sequence. We denote by $\mathcal{O}(a_n)$ the family of all sequence b_n for which there are positive real numbers $C, N > 0$ such that:

$$|b_n| \leq Ca_n, \quad \forall n \geq N.$$

In addition, by performing an abuse of notation, the fact that $b_n \in \mathcal{O}(a_n)$ is denoted by $b_n = \mathcal{O}(a_n)$.

Remark

This notation can also be used for limits and function, but throughout the course we stick at sequences.

Basic properties of the $\mathcal{O}$ notation

Exercise

Let a_n, b_n be essentially positive sequence and c_n be any sequence. Then,

1. $a_n = \mathcal{O}(a_n)$,
2. $\mathcal{O}(a_n) = \mathcal{O}(\lambda a_n) \ \ \forall \lambda > 0$.
3. If $c_n = \mathcal{O}(b_n)$ and $b_n = \mathcal{O}(a_n)$, then $c_n = \mathcal{O}(a_n)$.
4. $\mathcal{O}(b_n) \subset \mathcal{O}(a_n)$ if and only if $b_n = \mathcal{O}(a_n)$.

Definition

Let a_n, b_n be a sequence. by performing an abuse of notation, the fact that $\mathcal{O}(b_n) \subset \mathcal{O}(a_n)$ is denoted by $\mathcal{O}(b_n) = \mathcal{O}(a_n)$. This abuse of notation is very common in the literature.

Operations with sets of sequences

Definition

Let S_1, S_2 two families of sequences. Then:

1. $S_1 + S_2 := \{a_n + b_n : a_n \in S_1, b_n \in S_2\}$,
2. $S_1 - S_2 := \{a_n - b_n : a_n \in S_1, b_n \in S_2\}$,
3. $S_1 S_2 := \{a_n b_n : a_n \in S_1, b_n \in S_2\}$.

With that notation it makes sense to add or multiply sets with the big $\mathcal{O}$ notation.

Addition and multiplication with the $\mathcal{O}$ notation

Proposition

Let a_n, b_n be essentially positive sequences. Then,

1. $\mathcal{O}(a_n)\mathcal{O}(b_n) = \mathcal{O}(a_n b_n)$.
2. $\mathcal{O}(a_n) + \mathcal{O}(b_n) = \mathcal{O}(a_n + b_n)$.
3. $\mathcal{O}(a_n) - \mathcal{O}(b_n) = \mathcal{O}(a_n + b_n)$.

Remark

It is not true that $\mathcal{O}(a_n) - \mathcal{O}(b_n) = \mathcal{O}(a_n + b_n)$. For example:

$$\mathcal{O}(n) - \mathcal{O}(n) = O(n).$$

Indeed, $2n = \mathcal{O}(n)$, $n = \mathcal{O}(n)$, so the resulting set must contain the sequence $2n - n = n$.

$\mathcal{O}$ notation: proof

Proof.

Let $c_n = \mathcal{O}(a_n)$ and $d_n \in \mathcal{O}(b_n)$. Then, there are $C_1, C_2, N_1, N_2 > 0$ such that,

$$|c_n| \leq C_1 a_n \text{ if } n \geq N_1$$

and

$$|d_n| \leq C_2 b_n \text{ if } n \geq N_2$$

Thus, if we consider$N = \max\{N_1, N_2\}$, with the triangular inequality:

- $|c_n d_n| \leq (C_1 a_n)(C_2 b_n) = C_1 C_2 a_n b_n, \quad \forall n \geq N.$
- $|c_n + d_n| \leq |c_n| + |d_n| \leq C_1 a_n + C_2 b_n \leq \max\{C_1, C_2\}(a_n + b_n) \quad \forall n \geq N.$
- $|c_n - d_n| \leq |c_n| + |d_n| \leq C_1 a_n + C_2 b_n \leq \max\{C_1, C_2\}(a_n + b_n) \quad \forall n \geq N.$

Consequently, $c_n d_n \in \mathcal{O}(a_n b_n)$, $c_n + d_n = \mathcal{O}(a_n + b_n)$, and $c_n - d_n = \mathcal{O}(a_n + b_n)$. Since c_n, d_n are arbitrary sequences, this implies the three inclusions that we wanted to prove. □

Bachmann–Landau Θ notation

Definition

Let $a_n \in \mathbb{R}$ be a essentially positive sequence. We denote by $\Theta(a_n)$ the family of all essentially positive sequence b_n for which $b_n = \mathcal{O}(a_n)$ and $a_n = \mathcal{O}(b_n)$. In addition, by performing an abuse of notation, the fact that $b_n \in \Theta(a_n)$ is denoted by $b_n = \Theta(a_n)$.

Exercise

Prove that the Θ notation is an equivalence relation in the set of essentially positive sequences.

Exercise

Let a_n, b_n essentially positive sequences. Prove that:

1. $\Theta(a_n)\Theta(b_n) = \Theta(a_n b_n)$
2. $\Theta(a_n) + \Theta(b_n) = \Theta(a_n + b_n)$

Θ notation and polynomials

Proposition

Let $k \in \mathbb{N}$, and $p(x) = \sum_{r=0}^{k} c_r x^r$ for $c_0, \dots, c_{k-1} \in \mathbb{R}$ and $c_k > 0$. Then, $p(n) = \Theta(n^k)$.

Proof.

As $\lim_{n\to\infty} \frac{\sum_{r=0}^{k-1} c_r n^r}{n^k} = 0$, we have that there is $N > 0$ such that:

$$\left| \frac{\sum_{r=0}^{k-1} c_r n^r}{n^k} \right| \le \frac{c_k}{2}; \quad \text{that is,} \quad \left| \sum_{r=0}^{k-1} c_r n^r \right| \le \frac{c_k}{2} n^k.$$

Thus, if $n \ge N$:

$$p(n) \le \frac{3c_k}{2} n^k.$$

Moreover, if $n \ge N$:

$$p(n) \ge \frac{c_k}{2} n^k, \quad \text{so} \quad n^k \le \frac{2}{c_k} p(n).$$

Thus, $p(n) = \Theta(n^k)$. □

Θ notation and sums

Proposition

Let $\alpha \geq 0$. Then, $\sum_{k=1}^{n} k^{\alpha} = \Theta(n^{\alpha+1})$.

Proof.

On the one hand, if $n \geq 1$:

$$\sum_{k=1}^{n} k^{\alpha} \leq \sum_{k=1}^{n} \int_{k}^{k+1} x^{\alpha}\, dx = \int_{1}^{n+1} x^{\alpha}\, dx = \frac{(n+1)^{\alpha+1}-1}{\alpha+1} \leq \frac{2^{\alpha+1} n^{\alpha+1}}{\alpha+1}.$$

In addition,

$$\sum_{k=1}^{n} k^{\alpha} \geq \sum_{k=1}^{n} \int_{k-1}^{k} x^{\alpha}\, dx = \int_{0}^{n} x^{\alpha}\, dx = \frac{n^{\alpha+1}}{\alpha+1}.$$

Consequently,

$$n^{\alpha+1} \leq (\alpha+1) \sum_{k=1}^{n} k^{\alpha}.$$

Combining both inequalities, we obtain that: $\sum_{k=1}^{n} k^{\alpha} = \Theta(n^{\alpha+1})$. □

Corollary

Let $\alpha \geq 0$. Then, there is $C_{\alpha} > 0$ such that $\sum_{k=1}^{n} k^{\alpha} \leq C_{\alpha} n^{\alpha+1}$ for all $n \geq 1$.

7.2 Computational complexity of some basic algorithms.

Ways of measuring the computational cost

We may measure the computational complexity of a numerical method by asking how many elementary steps the optimal algorithm has. By elementary steps we may consider:

- Evaluation by a function, for instance, squaring.
- Additions, multiplications and divisions.
- Changing elements on a line.
- Looking at the binary tables and performing a small change.

The $\mathcal{O}(f(n))$ notation is used with we know the existence of an algorithm in at most $Cf(n)$ steps, and by $\Theta(f(n))$ when that $f(n)$ is optimal, when it cannot be improved with any algorithm.

Remark

Nowadays, the computational cost is measured by elementary steps in a Turing Machine, so the definition does not have any ambiguity. However, in this course we will be satisfied by answering these questions.

Number of elementary steps in matrix multiplications

Let $A, B \in \mathbb{R}^{n \times n}$. It is natural to ask the number of products between scalar numbers that are necessary to compute AB. For counting them, let us fix $i, j = 1, \dots, n$:

$$AB(i,j) = \sum_{k=1}^{n} A(i,k)B(k,j).$$

Thus, we have n multiplications and $n-1$ additions; that is, $2n-1$ elementary steps per term. Since there are n^2 terms, the number of steps is $2n^3 - n^2$, which might be shorten as $\mathcal{O}(n^3)$.

Remark

Here we have used the $\mathcal{O}$ notation, because we have not determined whether it is optimal.

Multiplication by a permutation matrix (i)

Let $P \in \mathbb{R}^{n\times n}$ be a permutation matrix induced by σ, and let $A \in \mathbb{R}^{n\times n}$. We want to compute the computational complexity of PA using the structure of permutation matrices. For that purpose, let us suppose consider the following framework. We consider an Excel file.

- In the first window there are the values of σ in a row, stored as a vector.
- In the second window there are the matrix A stored in the first n columns are row.
- In the third window we introduce the matrix PA.

Multiplication by a permutation matrix (ii)

We consider elementary steps as follow:

- Change file.
- Move a cell to the right/left/down/up.
- Click copy.
- Click paste.

We assume that in each file we start at the cell $(1, 1)$, and that in the first file we remain in the first row and first n columns, whether in the other files we remain.

Multiplication by a permutation matrix (iii)

We can see that the number of elementary steps is $\mathcal{O}(n^2)$ (that is, a sequence of upper bounds belong to such sets). Indeed, we have to proceed as follows for $i = 1, \dots, n$:

1. Look the value of $\sigma(i)$ in the first file (at most n step)
2. Change to the first element of the $\sigma(i)$ row in the second files (at most $2n$ steps).
3. Change to the first element of the i-th row in the second files (at most $2n$ steps).
4. Copy that row in the first row of the third file (at most $6n$ steps, as we have to copy, paste, move to the right in each file, and change file).

So, for $i = 1, \dots, n$ the elementary steps are at most $11n$. Thus, in total the number of steps is at most $11n^2$. Thus, multiplying by a permutation matrix has a computational cost of $\mathcal{O}(n^2)$.

Remark

Here, we can write that the computational cost is $\Theta(n^2)$ because, since a matrix has n^2 entries, we cannot do it better, as we have to consult each entry of A at least once.

Time complexity of Cholesky decomposition (i)

Let $A \in \mathbb{R}^{n\times n}$ a positive definite symmetric matrix. Let us see that the number of elementary steps in Cholesky decomposition is $\mathcal{O}(n^3)$. By elementary steps we mean adding, subtracting, multiplying, dividing and obtaining the square root.

Time complexity of Cholesky decomposition (ii)

For that purpose, we are going to estimate the number of elementary operations needed to compute the *i*-th formula:

- $L(i,1)$ requires 1 operation (one division).
- $L(i,2)$ requires 3 operations (one multiplication, one subtraction and one division).
- $L(i,3)$ requires 5 operations.
- $\vdots$
- $L(i,i-1)$ requires $2i-3$ operations ($i-2$ multiplications, $i-2$ subtractions and one division).
- $L(i,i)$ requires $2i-1$ operations ($i-1$ multiplications, $i-1$ subtractions and one square root).

Time complexity of Cholesky decomposition (iii)

Thus for the i-th row, we have at most $2\sum_{j=1}^{n} j$ elementary steps, which can be bounded by $C_1 i^2$, for a constant $C_1 > 0$ independent of i. Thus, the total number of elementary steps can be bounded by $C_1 \sum_{i=1}^{n} i^2$. Thus, there is $C_2 > 0$ independent of n such that the number of steps can be bounded by:

$$C_1 C_2 n^3.$$

Consequently, obtaining the Cholesky decomposition belongs to $\mathcal{O}(n^3)$.

Complexity of other known numerical methods

In the exercises we are going to prove the following results:

- Computing the matrix induced norm by the supreme norm is $\mathcal{O}(n^2)$.
- LUP is $\mathcal{O}(n^3)$.
- Solving $Ly = b$ or $Ux = y$ is $\mathcal{O}(n^2)$
- Computing the determinant via its definition is $\mathcal{O}(n \cdot (n!))$, but using the LUP or Cholesky decomposition is $\mathcal{O}(n^3)$.

7.3 Time complexity of matrix multiplication: Strassen algorithm.

The logical bound: n^3

By the nature of matrix multiplication in $\mathbb{R}^{n\times n}$, it looks logical that the amount of (scalar) additions and multiplications while multiplying matrices belong to $\Theta(n^3)$. After all, we are performing n^2 scalar products of different vectors of size n. Moreover, each element of A must multiply n elements of B, and A has n^2 different elements. Thus, for general matrices A and B (without any special structure such as permutation matrices), it seems logical that, to compute AB, no matter the algorithm, we need to compute at least $n \cdot n^2 = n^3$ scalar multiplications.

Proposition (The logical time complexity in a matrix product)

For any algorithm, there are two matrices $A, B \in \mathbb{R}^{n\times n}$ such that for computing AB it must perform n^3.

Identities

We have to be really careful while counting.

How many multiplications are in $a^2 + 2ab + b^2$?

The logical answer is 4, as the expression is $a \cdot a + 2 \cdot a \cdot b + b \cdot b$. It looks like no matter the algorithm, we will need 4 multiplications.

However, $a^2 + 2ab + b^2 = (a + b) \cdot (a + b)$, so we can do the same operation with one multiplication just by performing a "hidden identity", even if in the surface you need 4 multiplications, and this is done while also reducing the number of additions.

Strassen counterexample

Let us consider any matrices $A, B \in \mathbb{R}^{2\times 2}$. Let us compute the following products:

$$\begin{aligned}
\mathrm{I} &= (A(1,1)+A(2,2))(B(1,1)+B(2,2)),\\
\mathrm{II} &= (A(2,1)+A(2,2))B(1,1),\\
\mathrm{III} &= A(1,1)(B(1,2)-B(2,2)),\\
\mathrm{IV} &= A(2,2)(-B(1,1)+B(2,1)),\\
\mathrm{V} &= (A(1,1)+A(1,2))B(2,2),\\
\mathrm{VI} &= (-A(1,1)+A(2,1))(B(1,1)+B(1,2)),\\
\mathrm{VII} &= (A(1,2)-A(2,2))(B(2,1)+B(2,2)).
\end{aligned}$$

Note that we have only done 7 multiplications. Moreover,

$$AB = \begin{pmatrix} \mathrm{I}+\mathrm{IV}-\mathrm{V}+\mathrm{VII} & \mathrm{III}+\mathrm{V} \\ \mathrm{II}+\mathrm{IV} & \mathrm{I}+\mathrm{III}-\mathrm{II}+\mathrm{VI} \end{pmatrix}$$

Strassen algorithm for matrix in $\mathbb{R}^{N\times N}$, with $N = 2^n$ (i)

Let us consider any matrices $A, B \in \mathbb{R}^{N\times N}$. Let us denote $C = AB$. Let us consider:

- $A_{1,1} := A(1{:}N/2, 1{:}N/2)$, $B_{1,1} := B(1{:}N/2, 1{:}N/2)$, $C_{1,1} := C(1{:}N/2, 1{:}N/2)$
- $A_{1,2} := A(1{:}N/2, N/2+1{:}N)$, $B_{1,2} := B(1{:}N/2, N/2+1{:}N)$, $C_{1,2} := C(1{:}N/2, N/2+1{:}N)$
- $A_{2,1} := A(N/2+1{:}N, 1{:}N/2)$, $B_{2,1} := B(N/2+1{:}N, 1{:}N/2)$, $C_{2,1} := C(N/2+1{:}N, 1{:}N/2)$
- $A_{2,2} := A(N/2+1{:}N, N/2+1{:}N)$, $B_{2,2} := B(N/2+1{:}N, N/2+1{:}N)$, $C_{2,2} := C(N/2+1{:}N, N/2+1{:}N)$

Strassen algorithm for matrix in $\mathbb{R}^{N\times N}$, with $N=2^n$ (ii)

With the previous notation, we define:

$$\begin{aligned}
\mathrm{I} &= (A_{1,1}+A_{2,2})(B_{1,1}+B_{2,2}),\\
\mathrm{II} &= (A_{2,1}+A_{2,2})B_{1,1},\\
\mathrm{III} &= A_{1,1}(B_{1,2}-B_{2,2}),\\
\mathrm{IV} &= A_{2,2}(-B_{1,1}+B_{2,1}),\\
\mathrm{V} &= (A_{1,1}+A_{1,2})B_{2,2},\\
\mathrm{VI} &= (-A_{1,1}+A_{2,1})(B_{1,1}+B_{1,2}),\\
\mathrm{VII} &= (A_{1,2}-A_{2,2})(B_{2,1}+B_{2,2}).
\end{aligned}$$

If $n=1$, those are scalar multiplications, and can be computed normally. Otherwise, they are of size 2^{n-1} and are applied by using Strassen algorithm recursively. The algorithm is supposed to return:

$$AB=\begin{pmatrix}\mathrm{I}+\mathrm{IV}-\mathrm{V}+\mathrm{VII} & \mathrm{III}+\mathrm{V}\\ \mathrm{II}+\mathrm{IV} & \mathrm{I}+\mathrm{III}-\mathrm{II}+\mathrm{VI}\end{pmatrix}$$

Number of multiplications

Proposition

Let $n = 2^N$, and $A, B \in \mathbb{R}^{n\times n}$. Then, the number of multiplications of scalars that are performed in the Stress algorithm for are 7^N. Here, getting the opposite does not count as a multiplication.

Proof.

This is done by induction.When $N = 1$ (that is, $n = 2$) we have shown that we have 7 multiplications. Let us now assume that the result is true for N and prove it for $N + 1$. According to Strassen algorithm, we have to compute 7 products of matrices in $\mathbb{R}^{N\times N}$, which by the inductive hypothesis requires 7^N products. Thus, the total number of products is $7 \cdot 7^N = 7^{N+1}$, proving the result. □

Number of additions and subtractions (i)

To continue with, we are going to show that the number of additions and subtractions is also reduced.

Proposition

Let $n = 2^N$, and $A, B \in \mathbb{R}^{n \times n}$. Then, the number of additions and subtractions are performed in the Stress algorithm are at most $6 \cdot 7^n$.

Step 1: the induction. We are going to proof by induction that the number of addition and subtractions can be bounded by:

$$18 \sum_{k=0}^{N-1} \left(\frac{4}{7}\right)^k 7^{N-1}.$$

In that case, the result is obtained with:

$$C = \frac{18}{7} \sum_{k=0}^{\infty} \left(\frac{4}{7}\right)^k = \frac{18}{7} \frac{1}{1-\frac{4}{7}} = \frac{18}{3} = 6.$$

Number of additions and subtractions (ii)

Step 2: the base case. Let us now consider the base case; that is, $N = 1$ or $n = 2$. By a close look at the Strassen algorithm we show that there are 18 additions and subtractions, which is exactly $18\sum_{k=0}^{0}(4/7)^k 7^0$.

Step 3: the inductive case. Let us now suppose that the result is true for N and prove it for $N + 1$. When performing the matrix multiplication for size 2^{N+1}, we have to do 18 times an addition of size $(2^N)^2 = 4^N$.Moreover, we have to applied 7 times the Strassen algorithm with matrix of size 2^N. Thus, using the inductive hypothesis the number of additions and subtractions is given by:

$$\begin{aligned} 18 \cdot 4^N + 7 \cdot 18 \sum_{k=0}^{N-1} \left(\frac{4}{7}\right)^k 7^{N-1} &= 18 \cdot \left(\frac{4}{7}\right)^N 7^N + 18 \sum_{k=0}^{N-1} \left(\frac{4}{7}\right)^k 7^N \\ &= 18 \sum_{k=0}^{N} \left(\frac{4}{7}\right)^k 7^N \\ &= 18 \sum_{k=0}^{(N+1)-1} \left(\frac{4}{7}\right)^k 7^{(N+1)-1} . \square \end{aligned}$$

Strassen algorithm in general square matrices

We simply add 0s to the matrices to obtain a matrix of size 2^N, for $N = \lceil \log_2(n) \rceil$; that is, we approach to the closest power of 2. Then, we apply Strassen algorithm.

Computational complexity of Strassen algorithm.

Theorem (Computational complexity of Strassen algorithm)

The computational complexity of matrix multiplication in $\mathbb{R}^{n\times n}$ is $\mathcal{O}(n^{\log_2(7)})$.

Proof.

We first have to consider that by completing the matrix with 0s, we must apply Strassen algorithm for a matrix of size 2^N, for $N = \lceil \log_2(n) \rceil$. Moreover, in that case the number of arithmetical operations was bounded by 7^{N+1}. Consequently, the number of number of addition, subtractions and multiplications can 0be bounded by:

$$\begin{aligned} 7^{\lceil \log_2(n) \rceil + 1} &\leq 7^{\log_2(n)+2} = 49 \cdot 7^{\log_2(n)} \\ &= 49 \cdot (2^{\log_2(7)\log_2(n)}) = 49 \cdot n^{\log_2(7)} \end{aligned}$$

□