Complexity

P. Danziger

1 Solving Polynomial Series Equations

We will need to manipulate some equations involving series.

Theorem 1 *Given two series $f(i)$ and $g(i)$ and a constant a:*

1. $\displaystyle\sum_{i=1}^{n}(f(i)+g(i)) = \sum_{i=1}^{n} f(i) + \sum_{i=1}^{n} g(i)$
2. $\displaystyle\sum_{i=1}^{n} af(i) = a\sum_{i=1}^{n} f(i)$

i.e. The operation of summation from the set of series to $\mathbb{R}$ is linear.

Proof:
Let $f(i)$ and $g(i)$ be two series and a be a constant.

1. $$\begin{aligned}\sum_{i=1}^{n}(f(i)+g(i)) &\\ &= (f(1)+g(1)) + (f(2)+g(2)) + \ldots + (f(n)+g(n))\\ &= [f(1)+f(2)+f(3)+\ldots+f(n)] + [g(1)+g(2)+g(3)+\ldots+g(n)]\\ &= \sum_{i=1}^{n} f(i) + \sum_{i=1}^{n} g(i)\end{aligned}$$

2. $$\begin{aligned}\sum_{i=1}^{n} af(i) &= af(1)+af(2)+af(3)+\ldots+af(n)\\ &= a(f(1)+f(2)+f(3)+\ldots+f(n))\\ &= a\sum_{i=1}^{n} f(i)\end{aligned}$$

We will make extensive use of the formulas (a is a constant)

$$\begin{aligned} a+a+\ldots+a &= \textstyle\sum_{i=1}^{n} a &&= na\\ 1+2+\ldots+n &= \textstyle\sum_{i=1}^{n} i &&= \tfrac{1}{2}n(n+1)\\ 1^2+2^2+\ldots+n^2 &= \textstyle\sum_{i=1}^{n} i^2 &&= \tfrac{1}{6}n(n+1)(2n+1)\end{aligned}$$

When confronted with a series $\sum_{i=1}^{n} f(i)$ involving powers of i we first collect coefficients in $f(i)$ so that it is expressed as a sum of powers of i, $f(i) = a_0 + a_1 i + a_2 i^2 + \ldots + a_n i^n$. Now we use the formulas for powers of summing i. We use $p_j(n)$ to represent the formula giving $\sum_{i=1}^{n} i^j$. i.e. $\sum_{i=1}^{n} i^j = p_j(n)$.

$$\begin{aligned} &\textstyle\sum_{i=1}^{n} f(i)\\ =\ &\textstyle\sum_{i=1}^{n}(a_0 + a_1 i + a_2 i^2 + \ldots + a_n i^n)\\ =\ &\textstyle\sum_{i=1}^{n} a_0 + \sum_{i=1}^{n} a_1 i + \sum_{i=1}^{n} a_2 i^2 + \ldots + \sum_{i=1}^{n} a_n i^n\\ =\ & na_0 + a_1\tfrac{1}{2}n(n+1)i + \tfrac{1}{6}a_2 n(n+1)(2n+1) + \ldots + a_n p_n(n)\end{aligned}$$

We then simplify by collecting powers of n.

2 Complexity of Gaussian Methods

When we implement an algorithm on a computer, one of the first questions we must ask is how *efficient* the algorithm is. By this we mean how many *steps* it will take in the **worst case.** This is known as the *complexity* of the algorithm.
We start by defining a *step*. For the case of Gaussian methods a basic step is either an addition or a multiplication. We differentiate between the two because on most platforms the time needed for a multiplication is much longer than that needed for an addition. Note that we are considering floating point addition and multiplication, integer addition is generally much faster.
The number of operations required to solve a system of equations by Gaussian elimination and back substitution is the same as that required for the Gauss-Jordan method, but the Gauss-Jordan method is slightly easier to count.
We consider the cost of the elementary row operations on an $m \times n$ matrix A augmented with $\mathbf{b} \in \mathbb{R}^{\mathbf{m}}$, so there are $n+1$ columns.

- $R_i \to cR_i$. Each of the $n+1$ elements of row i must be multiplied, so cost is $n+1$ multiplications.
- $R_i \to R_i + cR_j$. Each of the $n+1$ elements of row j must be multiplied, then each of these must be added to the corresponding element in row i. Thus the cost is $n+1$ multiplications and $n+1$ additions.
- $R_i \leftrightarrow R_j$. With a correct implementation using pointers this operation is effectively instantaneous.

Gaussian-Jordan

```
For Each row $i$ ($R_i$) from 1 to $n$          ($\sum_{i=1}^n$)
    If any row $j$ below row $i$ has non zero entries to the right of the first non zero entry in row $i$
        $R_i \leftrightarrow R_j$
    $R_i \to \frac{1}{c}R_i$ where $c =$ the first non-zero entry of row $i$ (the pivot).
    For each row $j > i$                         ($n - i$)
        $R_j \to R_j - dR_i$ where $d =$ the entry in row $j$ which is directly below the pivot in row $i$.
    If any 0 rows have appeared exchange them to the bottom of the matrix.
next $i$ [Matrix is now in REF]
For each non zero row $i$ ($R_i$) from $n$ to 1  ($\sum_{i=1}^n$)
    For each $j < i$                             ($n - i$)
        $R_j \to R_j - bR_i$ where $b =$ the value in row $j$ directly above the pivot in row $i$.
```

So a rough calculation counting the number of multiplications yields

$$\begin{aligned}
&\textstyle\sum_{i=1}^n[(n+1)+(n-i)(n+1)] + \sum_{i=1}^n[(n-i)(n+1)] \\
= &\textstyle\sum_{i=1}^n \sum_{i=1}^n (2n-2i+1)(n+1) \\
= &\textstyle\sum_{i=1}^n 2n^2+3n+1-2(n+1)i \\
= &\ 2n^3+3n^2+n+n(n+1)^2 \\
= &\ 3n^3+5n^2+2n
\end{aligned}$$

A similar rough calculation for the number of additions yields $\sum_{i=1} 2(n-i)(n+1) = n^3 - n$
This is only a rough calculation. If we are considering the behaviour of the algorithm for large n, the highest term will dominate. We can say that the Gauss-Jordan algorithm has *order* n^3 and write $O(n^3)$.

2.1 Detailed Calculation

It is possible to program Gauss-Jordan much more tightly, by not preforming operations whose outcome is already known by the structure of the algorithm.
Consider Gaussian elimination whilst working on row i. We know that the first $i-1$ entries of row i are zero.
We must preform $R_i \to \frac{1}{c}R_i$. but we know that the result on the i^{th} entry of the row will be 1. So there are $n+1-i$ multiplications to be done.
We must now do $R_j \to R_j - dR_i$ for each of the $n-i$ rows j below i. We know that the first $i-1$ entries are zero and the i^{th} evaluates to 0, so again there only $n+1-i$ operations to be done.

$$\begin{array}{c} 1 \\ 2 \\ \\ i-1 \\ i \\ i+1 \\ \\ j \\ \\ n \end{array}
\left(\begin{array}{cccccccc|c}
1 & \text{x} & \dots & \text{x} & \text{x} & \text{x} & \dots & \text{x} & \text{x} \\
0 & 1 & \dots & \text{x} & \text{x} & \text{x} & \dots & \text{x} & \text{x} \\
\vdots & & \ddots & & \vdots & & \ddots & \vdots & \vdots \\
0 & 0 & \dots & 1 & \text{x} & \text{x} & \dots & \text{x} & \text{x} \\
0 & 0 & \dots & 0 & c & \text{x} & \dots & \text{x} & \text{x} \\
0 & 0 & \dots & 0 & \text{x} & \text{x} & \dots & \text{x} & \text{x} \\
\vdots & & \ddots & & \vdots & & \ddots & \vdots & \vdots \\
0 & 0 & \dots & 0 & d & \text{x} & \dots & \text{x} & \text{x} \\
\vdots & & \ddots & & \vdots & & \ddots & \vdots & \\
0 & 0 & \dots & 0 & \text{x} & \text{x} & \dots & 1 & \text{x}
\end{array}\right)$$

So for each row, i, from 1 to n we must (in the worst case) preform on the downward pass.

Operation	Add's	Mult's	number of times
$R_i \to cR_i$	0	$n-i+1$	1
$R_j \to R_j - dR_i$	$n-i+1$	$n-i+1$	$n-i$

So, for this row, the total number of additions is $(n-i)(n-i+1) = n^2+n-(2n+1)i+i^2$ and the total number of multiplications is $(n-i+1)+(n-i)(n-i+1) = (n-i+1)^2$
We now consider the cost over all rows:

$$\begin{array}{clll}
\sum_{i=1}^{n} & (n^2+n & -(2n+1)i & +i^2) \\
= & n^3+n^2 & -\sum_{i=1}^{n}(2n+1)i & +\sum_{i=1}^{n} i^2 \\
= & n^3+n^2 & -(2n+1)\sum_{i=1}^{n} i & +\sum_{i=1}^{n} i^2 \\
= & n^3+n^2 & -\frac{1}{2}(2n+1)n(n+1) & +\frac{1}{6}n(n+1)(2n+1) \\
= & n^3+n^2 & -n^3-\frac{3}{2}n^2-\frac{1}{2}n & +\frac{1}{6}(2n^3+3n^2+n) \\
= & \frac{1}{3}n^3-\frac{1}{3}n & &
\end{array}$$

A similar calculation shows

$$\sum_{i=1}^{n}(n-i+1)^2 = \frac{1}{3}n^3+\frac{1}{2}n^2+\frac{1}{6}n$$

We now consider the upward pass. At each step we only have to consider the entries in the augmented part, since all other entries are zero. For each row i, from n to 1 we must preform $n-i$ additions and multiplications, one for each row above row i. We thus have $\sum_{i=1}^{n}(n-i) = n^2-\frac{1}{2}n(n+1) = \frac{1}{2}n^2-\frac{1}{2}n$ additions and multiplications in total.

$$\begin{array}{c} 1 \\ 2 \\ \vdots \\ i-1 \\ i \\ i+1 \\ \vdots \\ j \\ \vdots \\ n \end{array} \left(\begin{array}{cccccccc|c} 1 & \text{x} & \dots & \text{x} & \text{x} & 0 & \dots & 0 & \text{x} \\ 0 & 1 & \dots & \text{x} & \text{x} & 0 & \dots & 0 & \text{x} \\ \vdots & & \ddots & & \vdots & & \ddots & \vdots & \vdots \\ 0 & 0 & \dots & 1 & \text{x} & 0 & \dots & 0 & \text{x} \\ 0 & 0 & \dots & 0 & 1 & 0 & \dots & 0 & \text{x} \\ 0 & 0 & \dots & 0 & 0 & 1 & \dots & 0 & \text{x} \\ \vdots & & \ddots & & \vdots & & \ddots & \vdots & \vdots \\ 0 & 0 & \dots & 0 & 0 & 0 & \dots & 0 & \text{x} \\ \vdots & & \ddots & & \vdots & & \ddots & \vdots & \\ 0 & 0 & \dots & 0 & 0 & 0 & \dots & 1 & \text{x} \end{array}\right)$$

So the total number of **additions** is

$$\frac{1}{3}n^3 - \frac{1}{3}n + \frac{1}{2}(n^2 - n)$$

$$\boxed{= \frac{1}{3}n^3 + \frac{1}{2}n^2 - \frac{5}{6}n}$$

and the total number of **multiplications** is

$$\frac{1}{3}n^3 + \frac{1}{2}n^2 + \frac{5}{6}n + \frac{1}{2}(n^2 - n)$$

$$\boxed{= \frac{1}{3}n^3 + n^2 + \frac{1}{3}n}$$

3 Determinants and Cofactor Expansion

When we calculate the determinant of an $n \times n$ matrix using cofactor expansion we must find n $(n-1) \times (n-1)$ determinants. So (roughly) $C_n \approx nC_{n-1}$, where C_n is the complexity of finding an $n \times n$ determinant. Now $C_2 = 2$ (two multiplications).

n	2	3	4	5	...
C_n	2	$3 \cdot 2$	$4 \cdot 3 \cdot 2$	$5 \cdot 4 \cdot 3 \cdot 2$	...

We can see that $\boxed{C_n = n!}$
Given an $n \times n$ determinant to calculate, we may either use the cofactor method, with a runtime of $O(n!)$, or we may reduce the matrix using Gaussian elimination, keeping track of the effect on determinant, multiplying the diagonal entries at the end. This would be $O(n^3)$, the order of Gaussian elimination.

n	2	3	4	5	6	7
n^3	8	27	64	125	216	343
$n!$	2	6	24	120	720	5040

For small values of n the cofactor method wins, but as n grows $n!$ get very big very quickly and the cofactor method becomes impractical.

4 Exercises

1. Use the methods above to find

   (a) $\sum_{i=1}^{n} n+i$

   (b) $\sum_{i=1}^{n} (n+i)^2$

   (c) $\sum_{i=1}^{n} (2n+i+1)(n-i)$

2. Show that

   (a) $\sum_{i=1} 2(n-i)(n+1) = n^3 - n$.

   (b) $\sum_{i=1}^{n} (n-i+1)^2 = \frac{1}{3}n^3 + \frac{1}{2}n^2 + \frac{1}{6}n$

3. Repeat the procedure above to find the complexity of of Gauss-Jordan on an $m \times n$ matrix.

4. When we find the inverse of an $n \times n$ matrix A we augment $(A|I)$ and use Gauss-Jordan to find the inverse.

   (a) In the downward pass, on the i^{th} step how many operations (additions and multiplications separately) are needed

      i. to calculate $R_i \to cR_i$;

      ii. to calculate $R_j \to R_j + dR_i$ for some row below row i.

   (b) Using that there are $n-i$ rows below any row i calculate the complexity of the downward pass of this method.

   (c) Calculate the complexity of the upward pass.

   (d) Find the complexity of finding an inverse using Guass Jordan.

5. What is the complexity of finding AB for two $n \times n$ matrices A and B.

6. What is the complexity of finding AB where A is $m \times n$ and B is $n \times r$.

7. Try finding 70! on your calculator. Why does this give an error on most calculators?

8. Estimate the complexity of finding the inverse of an $n \times n$ matrix A by the adjoint method. Given your answer to part 4d which algorithm should you use?

9. From the table of n^3 vs $n!$ we see that $4^3 > n!$, yet the Gaussian method is the one that is generally recommended for calculating 4×4 determinants, why?