UERJ – Instituto de Matemática e Estatística Especialização em Aprendizado em Matemática II/2002

Cálculo e Computação – Palestra 19-8-2002

Carlos A. de Moura[†]

Instituto de Matemática e Estatística – UERJ
[†]Departamento de Análise Matemática

1 Introdução

Nas duas palestras apresentadas no semestre passado, discutimos as contribuições que os avanços na tecnologia da computação – desenvolvida dramaticamente a partir da 2^a. guerra mundial – trouxe às ciências, tecnologias, ao nosso dia-a-dia e à própria Matemática. Por outro lado, enfatizamos a contribuição e a indispensável participação da teoria matemática em questões essenciais nesse desenvolvimento. Expusemos, em particular, exemplos que demonstram a necessidade de fugir da folclórica sensação da infalibilidade, confiabilidade dos resultados obtidos com o uso dos computadores. Citamos o caso do fracasso do foguete francês *Ariane 5*, no seu primeiro lançamento. A queda, comprovou-se, foi devida a erros de programação numérica.

Uma referência para a questão dos erros numéricos pode ser obtida em
Moura,CA de (1999): **COMPUTADOR TAMBÉM ERRA**, Ciência Hoje na Escola, Vol.8, pp.16-18, Ed. SBPC.
e nas páginas
http://ime.uerj.br/~demoura/Especializ/Ariane
http://ime.uerj.br/~demoura/Especializ/Arredond

A outra face da moeda é a expectativa de que qualquer cálculo que não conseguimos efetuar, eles, os computadores, conseguem. Longe daí a realidade! Este ponto, de fato, deu origem a uma nova área de pesquisa matemática, a **Complexidade Computacional**.
Antes de iniciarmos, codificarmos, depurarmos e rodarmos um programa computacional, é indispensável termos a garantia de que o programa será finalizado em um intervalo de tempo **viável**!

2 Um problema de Álgebra Linear

Um dos problemas mais antigos, simples e ininterruptamente resolvidos – dizem mesmo que ele ocupa mais de 85% do tempo utilizado por todo o conjunto de equipamentos dedicados à chamada **Computação Científica** – é o de buscar a solução para um sistema linear de equações algébricas, de ordem N :

$$\begin{bmatrix} a_{11} & \cdots & a_{1N} \\ \cdot & & \cdot \\ \cdot & & \cdot \\ \cdot & & \cdot \\ a_{N1} & \cdots & a_{NN} \end{bmatrix} \cdot \begin{bmatrix} x^1 \\ \cdot \\ \cdot \\ \cdot \\ x^N \end{bmatrix} = \begin{bmatrix} b^1 \\ \cdot \\ \cdot \\ \cdot \\ b^N \end{bmatrix} ,$$

ou, *por extenso*,

$$\sum_{k=1}^{N} a_{\imath,k}\, x^k \;=\; b^\jmath \;,\quad \imath, \jmath \;=\; 1, \ldots,\; N \;,$$

ou, ainda mais compactamente,

$$A\, x \;=\; b \;.$$

Um dos primeiros métodos de solução que estudamos para resolver esse problema é chamado de **regra de Cramer**. Consiste numa expressão relativamente simples de formular e tem a vantagem de apresentar uma **fórmula explícita**. Esse resultado afirma que a solução do sistema formulado acima pode ser obtido por

$$x^\imath \;=\; \frac{\det A_\imath}{\det A} \;,\quad \imath \;=\; 1, \ldots,\; N \;,$$

onde a matriz $A_\imath$ é obtida da matriz A por meio da substituição da $\imath$-ésima coluna pelo vetor do segundo membro, conhecido, b .
Portanto, para se resolver esse sistema de N equações é preciso efetuar N divisões , mas antes de chegar a elas, necessitamos de $N+1$ determinantes. O ponto é justamente o cálculo

desses determinantes. Suponhamos que os vamos calcular operando a partir da definição, que também fornece uma fórmula explícita, a saber

$$\det A \;=\; \sum_{\text{permutações}} \; \prod_{k=1}^{N} \; a_{\imath(k)\,\jmath(k)} \; (-1)^{\sigma} \;,$$

onde σ é o sinal da permutação dos índices $\imath\,,\jmath$, isto é, quantas inversões eles contêm. Em resumo, cada determinante é calculado como uma soma de um certo número de parcelas, cada uma delas efetuada com o produto de N fatores. Quantas parcelas temos? Ora, são $N!$, o número de funções 1-1, de um conjunto de N elementos nele mesmo. O que será que isso acarreta em termos de cálculos a efetuar?
Vamos olhar a tabela que segue

N	N^2	N^3	2^n	$N!$
1	1	1	2	1
2	4	8	4	2
3	9	27	8	6
4	16	64	16	24
5	25	125	32	120
6	36	216	64	720
7	49	343	128	5040
8	64	512	216	40320
9	81	729	512	362880
10	100	1000	1024	3.628.800

e observar como a função **fatorial** ganha rapidamente *de muito* até mesmo da **exponencial**, que conhecemos como uma função de tão rápido crescimento que se tornou a sua menção uma expressão da linguagem, mesmo dos *mais leigos* em matemática, v.g., "tal candidato está crescendo exponencialmente".
Quais são as conseqüências para aplicarmos a regra de Cramer? Bom, já a aplicamos na prática, ou fizemos nossos alunos aplicá-la, para matrizes de ordem 2 ou 3, até aí, nada grave ocorreu. Mas em problemas da vida real, a resolução de um sistema de ordem 10.000 não é rara. Situações que exigem milhares de variáveis são bem comuns e, portanto, não é exagero exemplificarmos o que ocorre com uma matriz de ordem 100. Efetuaremos mesmo os 99 produtos necessários para obter o valor de **100!**? Antes, apliquemos resultados clássicos do Cálculo Diferencial e Integral, pedindo ajuda, para trabalhar com a fórmula de Cramer, a outra fórmula, a de Stirling.
Uma prática importante em Análise Matemática é a de **estimar**, que significa inferir conclusões quantitativas sobre entes pouco conhecidos a partir de outros, mais familiares, ou mais dóceis.

Isso que passamos a fazer.
O produto dos N fatores em $N!$ é claramente menor do que o produto obtido se tomarmos os N fatores iguais a N , ou seja,

$$N! \ < \ N^N \ , \quad \text{se} \ \ N \ \geq \ 2 \ .$$

Exercício Um desses termos está associado a funções 1-1, já o mencionamos, e o outro?
Exercício Demonstre rigorosamente a desigualdade enunciada.

É claro que essa estimativa pode ser qualificada de exagerada: se ela aparecesse também na tabela acima, teríamos, em vez de aproximadamente 3.6×10^6 para o fatorial de 10, quatro ordens de grandeza a mais, e essa situação vai se agravar, à medida que N cresce. Uma idéia seria a de *dar uma ajudazinha* ao fatorial, e é o que fazemos tentando calcular o

$$\lim_{N\to\infty} \ \frac{N^N}{N!\ e^N} \ = \ \Gamma \ .$$

Demonstra-se que passamos da medida, pois $\Gamma \ = \ 0$, ou seja, agora é N^N que precisa de uma *forcinha*. E esta é bem pequena, conforme deduziu Stirling, cujo nome ficou associado à expressão

$$\lim_{N\to\infty} \ \frac{N^N \ \sqrt{2\pi N}}{N!\ e^N} \ = \ 1 \ .$$

Vamos então, sem receio de usar desigualdades grosseiras e perder muito nas estimativas, estimar o valor de $N!$ a partir de

$$N! \ \sim \ (N/e)^N \ \sqrt{2\pi N} \ , \quad \text{para} \ \ N \ \sim \ \infty$$

da seguinte forma:

$$\text{para} \ \ N \ \sim \ \infty \ , \ N! \ \sim \ (N/e)^N \ \sqrt{2\pi N} \ > \ (N/10)^N \ \sqrt{4N} \ ,$$

o que nos dá, para $N \ = \ 100$,

$$100! \ > \ (10^{100})20 \ .$$

É um número de valor bem elevado, tão elevado que não temos uma boa idéia do que ele pode significar, em termos práticos. Quantas operações pode um computador efetuar por segundo? E por mili-segundo? E por micro-segundo? No contexto da eletrônica, da computação, dos experimentos de laboratório, necessitamos lidar com grandezas muito pequenas e muito elevadas.

Observe a terminologia introduzida para essas grandezas na tabela[1] que segue.

Prefixo	Símbolo(s)	Potência de 10	Potência de 2
yocto-	y	10^{-24} *	–
zepto-	z	10^{-21} *	–
atto-	a	10^{-18} *	–
femto-	f	10^{-15} *	–
pico-	p	10^{-12} *	–
nano-	n	10^{-9} *	–
micro-	m	10^{-6} *	–
mili-	μ	10^{-3} *	–
centi-	c	10^{-2} *	–
deci-	d	10^{-1} *	–
(nenhum)	–	10^{0}	2^{0}
deca-	D	10^{1} *	–
hecto-	h	10^{2} *	–
kilo-	k ou K **	10^{3}	2^{10}
mega-	M	10^{6}	2^{20}
giga-	G	10^{9}	2^{30}
tera-	T	10^{12}	2^{40}
peta-	P	10^{15}	2^{50}
exa-	E	10^{18} *	2^{60}
zeta-	Z	10^{21} *	2^{70}
yota-	Y	10^{24} *	2^{80}

* Não é usado para taxa de circulação de dados, normalmente
** $k = 10^3$ e $K = 2^{10}$

A contagem de operações em um computador é geralmente expressa em número de *flops*, isto é, de operações de ponto (vírgula) flutuante por segundo. Suponhamos que nosso computador atinja 1000 gigas de operações por segundo, ou mais, um bilhão de gigas, o que não se tem esperança de nem chegar perto num futuro à vista.

Exercício Pesquise na literatura qual a capacidade dos computadores atuais, em *flops*.

[1] Obtida de *http://whatis.techtarget.com*, vale a pena consultar.

Agora vamos também exagerar na outra direção, totalizando as operações que podemos efetuar em um intervalo de tempo. Suponhamos que o minuto tenha 10^2 segundos, a hora, 10^2 minutos, o dia também 10^2 horas e, finalmente, o mês 10^2 dias. Para encerrar, que o ano tenha igualmente 10^2 meses. Assim, um ano totalizaria 10^{10} segundos, o que nos permitiria apenas um volume de cálculos de 10^{28} operações . Três anos de cálculos ininterruptos, mesmo com este relógio e calendário especialíssimos – e sem *apagões* – ainda não nos bastariam. E ainda estamos esquecendo aquele fator 20 ...

Os programas disponíveis resolvem sistemas de equações lineares algébricas em muito menos tempo, evidentemente. Seguem outro caminho, o do chamado algoritmo de triangularização de Gauss. Essencialmente consiste ele em transformar as equações do sistema em outras que são equivalentes, ou seja, que apresentam as mesmas soluções do sistema original, e com as quais é mais fácil de lidar. Uma das equações tem apenas uma incógnita, o valor desta é encontrado e utilizado numa das outras equações , a que só tem duas incógnitas, sendo uma delas esta que acabamos de calcular. E assim, sucessivamente.
A grande diferença nas duas abordagens é que este é um esquema numérico **factível**: ele envolve apenas $N^3/6$, a menos de uma constante multiplicativa fixa para qualquer matriz. No caso em estudo, isso equivale a 10^6 operações algébricas, ou seja, se nosso computador tiver a taxa de 1 megaflops, em um segundo estaremos com a solução disponível.

3 Observações

- $20! = 2.432.902.008.176.640.000 \sim 2.4x10^{18}$
- $2^{20} = 2^{10\times 2} = (2^{10})^2 \sim (10^3)^2 = 10^6$
- E com relação ao item **precisão**, confiabilidade, que se pode afirmar deste algoritmo? A forma como o utilizamos para pequenos sistemas, efetuando os cálculos de maneira exata, deve ser adaptada para quando se utilizam computadores, os quais sempre **aproximam** os números com que operam. Nesse contexto, ver

  C.A. de Moura. Triangularização sem Pivoteamento: um (mau) Exemplo. Boletim "Dá licença", IM/UFF, (2002).
  Disponível em: http://ime.uerj.br/~demoura/Especializ/Arredond/pivot.pdf

- O exemplo acima já ilustra que o fato de dispormos de uma fórmula fechada, uma ex-

pressão explícita para um dado problema, como ocorre com a fórmula de Cramer, não significa necessariamente que o resolvemos. Se desejarmos a solução numérica de um sistema, essa expressão não indica um bom caminho. Situações existem, porém, onde esse é um importante – e útil – resultado (não menosprezar o Cramer!).

- Recentemente foi descoberto (inventado?) um algoritmo para determinar se um dado inteiro é ou não primo. Ele reduz grandemente a complexidade computacional até então obtida para esse problema, tendo posto em evidência a área de pesquisa da **Complexidade Computacional**. O artigo que contém esse resultado e comentários sobre o mesmo estão disponíveis em
  http://ime.uerj.br/~demoura/Especializ/AlgorPRIMOS.pdf
  http://ime.uerj.br/~demoura/Especializ/AlgorPRIMOS_1.txt

- Dois interessantes portais para consulta, sendo o primeiro só de matemática, em inglês, o segundo ligado às Secretarias Estaduais de C & T e de Educação:
  http://mathworld.wolfram.com/

  http://www.educacaopublica.rj.gov.br/