SME0305 - 2016

Gustavo C. Buscaglia / Roberto F. Ausas

ICMC - Ramal 738176, gustavo.buscaglia@gmail.com
ICMC - Ramal 736628, rfausas@gmail.com

Representação de números e erro de arredondamento

- Os números "**reais**" num computador são **finitos**.
- Eles se representam em **base 2**, de uma certa maneira.
- Vamos utilizar o problema de representação em base arbitrária como **exercício de programação**.

- A equação

$$x^2 = 2\ ,$$

  por exemplo, **não tem solução exata** no computador. De fato, a equação $13\,x = 15$ também não tem.

- A finitude leva a **erros de arredondamento**. Hemos de conviver com eles.

- A finitude também implica em **overflows** e **underflows**.

- A representação de ponto flutuante é a universalmente adotada nos computadores, nela

$$x = (-1)^s \underbrace{(0.a_1 a_2 \ldots a_t)}_{m} \beta^e = (-1)^s\, m\, \beta^{e-t} \qquad (1)$$

  onde β é a **base**, t o **número de dígitos** (na base β), e o **exponente** e m a **mantissa**.

- Essa representação tem a vantagem de que o erro de arredondamento satisfaz

$$\frac{|x - \mathrm{fl}(x)|}{|x|} \leq \frac{\epsilon_M}{2} \ , \tag{2}$$

  com $\epsilon_M = \beta^{1-t}$. isto é, que o **erro relativo de arredondamento** esteja **uniformemente limitado**. Isto explica a ideia comum de que "um float (single) tem 7 dígitos decimais certos", ou de que "um double tem 15 dígitos decimais certos". Na verdade a representação é binária ($\beta = 2$) e, no caso do *double*, $\epsilon_M = 2^{-52} \simeq 2.22 \times 10^{-16}$.

- Obviamente, o **erro absoluto de arredondamento** cresce de maneira aproximadamente proporcional ao número arredondado.

- Sendo que as operações são feitas também com representação

finita (algumas vezes com **dígitos de guarda** adicionados), existe uma **perda** ou **cancelamento** de "dígitos certos". Por exemplo,

```
>> x = 1.e-15; ((1+x)-1)/x
ans =
   1.1102
```

que tem um erro de 11%!

- Existem **operações que são especialmente sensíveis** aos erros de arredondamento e podem requerir algoritmos especiais:
  - Cálculo de **derivadas por diferenças**, e.g.,

$$f'(x) \simeq \frac{f(x+\delta) - f(x-\delta)}{2\delta} .$$

A escolha certa de δ depende do erro de arredondamento resultante da avaliação de f e da precisão dos cálculos.

– **Derivadas de mais alta ordem** (derivadas aproximadas de derivadas aproximadas) são ainda mais sensíveis.

– **Ortogonalização de vetores** quase linearmente dependentes. Dados dois vetores x e y, procuramos z que seja **combinação linear** de x e y, e que seja **ortogonal** a x. A solução da álgebra linear é

$$z = y - \frac{(y,x)}{(x,x)}\, x \ .$$

– Escalonamento de **matrizes mal condicionadas** (matrizes tais que $|\lambda_{\mathsf{max}}/\lambda_{\mathsf{min}}| \gg 1$, em particular matrizes “quase” singulares.

  - Integrais numéricas de funções singulares.
  - Somatória numérica de séries.
  - Et cetera...

- **Ler páginas 3 a 8 do livro de Quarteroni et al**.
- Vamos utilizar como exemplo a **Ortogonalização de Gram-Schmidt**, e mostrar como o erro de arredondamento pode ser reduzido modificando o algoritmo.

http://www.devtopics.com/20-famous-software-disasters/

8. Patriot Fails Soldiers (1991)

Cost: 28 soldiers dead, 100 injured

Disaster: During the first Gulf War, an American Patriot Missile system in Saudi Arabia failed to intercept an incoming Iraqi Scud missile. The missile destroyed an American Army barracks.

Cause: A software rounding error incorrectly calculated the time, causing the Patriot system to ignore the incoming Scud missile.

9. Pentium Fails Long Division (1993)

Cost: $475 million, corporate credibility

Disaster: Intel's highly-promoted Pentium chip occasionally made mistakes when dividing floating-point numbers within a specific range. For example, dividing 4195835.0/3145727.0 yielded 1.33374 instead of 1.33382, an error of 0.006%. Although the bug affected few users, it become a public relations nightmare. With an estimated 5 million defective chips in circulation, Intel offered to replace Pentium chips only for consumers who could prove they needed high accuracy. Eventually Intel replaced the chips for anyone who complained.

Cause: The divider in the Pentium floating point unit had a flawed division table, missing about five of a thousand entries and resulting in these rounding errors. (more)

10. Ariane Rocket Goes Boom (1996)

Cost: $500 million

Disaster: Ariane 5, Europe's newest unmanned rocket, was intentionally destroyed seconds after launch on its maiden flight. Also destroyed was its cargo of four scientific satellites to study how the Earth's magnetic field interacts with solar winds.

Cause: Shutdown occurred when the guidance computer tried to convert the sideways rocket velocity from 64-bits to a 16-bit format. The number was too big, and an overflow error resulted. When the guidance system shut down, control passed to an identical redundant unit, which also failed because it was running the same algorithm. (more)

Exemplo: Algoritmo de soma de Kahan (1965)

```
function sum=KahanSum(input)

sum = 0.0
c = 0.0
for i = 1:size(input)
  y = input(i) - c;
  t = sum + y;
  c = (t - sum) - y;
  sum = t;
end
```

Exemplo: Ortogonalização de vetores

Problema: Sejam $v^{(1)}$, $v^{(2)}$, ..., $v^{(m)}$, vetores linearmente independentes de $\mathbb{R}^n$. Seja V o espaço vetorial de todas as combinações lineares deles. Construa uma base ortogonal de V.

Qual a relevância desse problema?

Bases ortogonais permetem calcular **projeções ortogonais**.
A projeção ortogonal de $v \in \mathbb{R}^n$ sobre V é o vetor $w \in V$ que **melhor aproxima** v, i.e., que minimiza $\|w - v\|$ (norma euclidiana).
Seja $e^{(1)}$, $e^{(2)}$, ..., $e^{(m)}$ uma base ortogonal de V. Então

$$w = (e^{(1)}, v)\, e^{(1)} + (e^{(2)}, v)\, e^{(2)} + \ldots + (e^{(m)}, v)\, e^{(m)} . \quad (3)$$

E qual a relevância de calcular projeções ortogonais?

Um caso: Distância a um plano definido por três pontos.

- Pontos O, P e Q. Sejam $a = \overline{OP}$, $b = \overline{OQ}$.

  ```
  > O=[5;3;14]; P=[4;5;15]; Q=[7;7;12];
  ```

  ```
  > a=P-O; b=Q-O;
  ```

- O **plano** é dado por O mais todas as combinações lineares de a e b.

- Sejam $e^{(1)}$ e $e^{(2)}$ ortonormais, tangentes ao plano.

  ```
  [e1 e2] = calculabase1(a,b)
  ```

- Seja X um outro ponto, e seja $c = \overline{OX}$.

- A projeção Y de X sobre o plano é

  ```
  > Y = O + (e1'*c)*e1 + (e2'*c)*e2
  ```

- A distância de X ao plano é

```
> dist = norm(X-Y)
```

- O vetor normal ao plano é

```
> normal = (X-Y)/dist
```

Outro caso: Combinação de portfolios.

- O fundo de investimento 1 tem o seguinte portfolio

  ```
  >port(:,1)=[5*ones(50,1);15*ones(10,1);30*ones(20,1)]
  ```

  significando que, quando o investimento é de 1000 $, compram-se 5$ das ações 1 a 50, 15$ das ações 51 a 60 e 30$ das ações 61 a 80.

  Um portfolio é assim um vetor em $\mathbb{R}^{80}$.

- Portfolios dos outros fundos disponíveis:

  ```
  >port(:,2)=[10*ones(40,1);0*ones(10,1);20*ones(30,1)]
  >port(:,3)=[0*ones(70,1);80*ones(5,1);120*ones(5,1)]
  >port(:,4)=[10*ones(30,1);20*ones(35,1);0*ones(15,1)]
  >port(:,5)=[20*ones(50,1);0*ones(50,1)]
  ```

- Um cliente **quer realizar** o seguinte investimento:

  `target=[0:20:1580]'` (notar que `sum(target)=63200`)

  Qual é o investimento mais aproximado ao desejo do cliente que é possível conseguir combinando os 5 fundos disponíveis?

  $$(e_1, x)e_1 + \ldots + (e_m, x)e_m \longleftrightarrow \texttt{base*base'*target}$$

  ```
  > base = calculabase2(port);
  > aprox=base*base'*target
  ```

  rode exportfol.m para ver o resultado.

- Isto nos dirá qual é o investimento mais aproximado que poderemos conseguir. Para ver quanto devemos investir em cada um dos fundos, ver capítulo de **decomposição QR**.

Ortonormalizar dois vetores:

- Normalizar o primeiro: $e^{(1)} = a/\|a\|$
- Projetar b na direção de a: $p = (e^{(1)}, b)\, e^{(1)}$
- Subtrair a projeção obtida: $q = b - p$
- Normalizar: $e^{(2)} = q/\|q\|$

```
function [e1 e2] = calculabase1(a,b)
e1 = a/norm(a);
p = (e1'*b)*e1;
e2 = (b-p)/norm(b-p);
end
```

```
O=[5;3;14]; P=[4;5;15]; Q=[7;7;12];
a=P-O; b=Q-O;
[e1 e2] = calculabase1(a,b);
X=[4;4;13];
c=X-O;
Y = O + (e1'*c)*e1 + (e2'*c)*e2
%verificando que esteja no plano
d=Y-O;
err=det([a b d])

>> Y =
       5
       4
      14

>> err = 0
```

Ortonormalizar `m` vetores:

```
function B = calculabase2(A)
# Orthogonalizes the columns of A by Gram-Schmidt
[m,n]=size(A);
B = A;
for k=1:n
 V=B(:,1:k-1)'*B(:,k);
 B(:,k)=B(:,k)-B(:,1:k-1)*V;
 rkk=norm(B(:,k));
 B(:,k)=B(:,k)/rkk;
end
end
```

Ortonormalizar `m` vetores (mais preciso):

```
function B = calculabase2(A)
# Orthogonalizes the columns of A by Gram-Schmidt
# with improved round-off treatment
[m,n]=size(A);
B = A;
for k=1:n
 V=B(:,1:k-1)'*B(:,k);
 B(:,k)=B(:,k)-B(:,1:k-1)*V;
 V=B(:,1:k-1)'*B(:,k);
 B(:,k)=B(:,k)-B(:,1:k-1)*V;
 rkk=norm(B(:,k));
 B(:,k)=B(:,k)/rkk;
end
end
```