Calcolo Numerico - Corso B: Laboratorio Lezione 2

Luca Gemignani <luca.gemignani@unipi.it>

11 Marzo 2020

1 Iniziare a Programmare in MatLab

Iniziamo con qualche semplice esempio.

```
function f=fact(n);
% calcola il fattoriale di n
% n dev'essere un intero
f=1;
% f fa da accumulatore: parte da 1,
% a ogni passo, lo moltiplico per k
for k=1:n
  f=f*k;
end
% ora f vale n!
end
```

Va scritto in un file chiamato `fact.m` e messo *nella cartella da cui abbiamo lanciato Matlab*; poi possiamo lanciarlo:

```
>> fact(10)
ans = 3628800
```

Esercizio 1. Scrivete una funzione `pow(x,n)` che calcoli x^n, per $n \geq 1$.

2 Calcolo dell'esponenziale

La formula di Taylor permette di esprimere una funzione come un polinomio con infiniti termini, e troncando in modo opportuno questa serie è possibile approssimare una funzione con un polinomio.

In particolare, data una funzione $f \in C^n$, cioè derivabile n-volte con derivata n-esima continua, tale che $f : (a, b) \to \mathbf{R}$. Preso un punto $x_0 \in (a, b)$, abbiamo che

$$f(x) = f(x_0) + f'(x_0)(x - x_0)| + \frac{f''(x_0)}{2!}(x - x_0)^2 + \cdots + \frac{f^{(n)}(x_0)}{n!}(x - x_0)^n + R_n(x)$$

dove $R_n(x)$ è detto resto di Taylor e nella sua forma di Lagrange è dato da

$$R_n(x) = \frac{f^{(n+1)}(\xi)}{(n+1)!}(x - x_0)^{n+1},$$

sotto l'ipotesi che $f^{(n+1)}(x)$ esista per ogni $x \in (a, b), x \neq x_0$, e con $|\xi - x_0| < |x - x_0|$.

Si può semplicemente vedere che il polinomio di Taylor per $x_0 = 0$, arrestato al termine n-esimo della funzione esponenziale è

$$1 + x + \frac{x^2}{2} + \frac{x^3}{3!} + \cdots + \frac{x^n}{n!}$$

da cui possiamo scrivere la seguente funzione per approssimare il valore dell'esponenziale in un punto x.

```
function a=myexp(x,n)
% calcola exp(x) con la serie di Taylor troncata all'n-esimo termine
  a=1; %accumulatore
  for k=1:n
    a=a+pow(x,k)/fact(k);
  end
end
```

Qualche esperimento su quanti termini servono per approssimare bene — confrontiamo con la funzione `exp(x)` di Matlab, che calcola l'esponenziale meglio di noi.

```
>> myexp(1,5)
ans = 2.7167
>> myexp(10,50)
ans = 2.2026e+04
>> exp(10)
ans = 2.2026e+04
>> format long
>> myexp(10,50)
ans = 22026.4657948067
>> exp(10)
ans = 22026.4657948067
```

Due problemi:

1. Inaccurato: prova `myexp(-20,500)`

2. Lento: con n termini, il numero di operazioni da eseguire cresce come n^2 (perché?)

Risolviamo il problema 2. introducendo un altro accumulatore:

```
function a=myexp2(x,n)
  %calcola e^x con Taylor troncato
  %ma usa solo O(n) operazioni
  t=1; %accumulatore che contiene il termine generico della sommatoria
  a=1; %accumulatore che contiene le somme parziali
  for k=1:n
    t=t*x/k;
    a=a+t;
  end
end
```

Ora va meglio:

```
>> myexp(-20,500)
ans = NaN
>> myexp2(-20,500)
ans = 5.62188447213042e-09
```

Cosa succedeva?

```
>> fact(500)
ans = Inf
>> pow(-20,500)
ans = Inf
>> Inf/Inf
ans = NaN
```

Ci sono ancora pesanti perdite di accuratezza sui numeri negativi:

```
>> myexp2(-30,500)
ans = -3.06681235635622e-05
```

Un esponenziale dovrebbe sempre essere positivo... Ci sono pesanti errori di cancellazione nella formula che abbiamo usato (perché?) La soluzione: cambiare algoritmo e sceglierne uno che non porti a errori di cancellazione che sono dovuti al fatto che per $x < 0$ la serie esponenziale è a segni alterni.

```
>> exp(-30)
ans = 9.3576e-14
>> myexp2(-30,500)
ans = -3.0668e-05
>> 1/myexp2(30,500)
ans = 9.3576e-14
>> format long
>> exp(-30)
ans = 9.35762296884017e-14
```

```
>> 1/myexp2(30,500)
ans = 9.35762296884017e-14
```

3 Calcolo della radice quadrata di un numero reale positivo

Il calcolo della radice quadrata di un numero positivo $a > 0$ si basa sull'osservazione che $\alpha = \sqrt{a}$ è soluzione dell'equazione

$$f(x) = x^2 - a = 0.$$

Per determinare unàpprossimazione della soluzione si utilizza il metodo delle tangenti o di Newton definito da

$$\begin{cases} x_0 > 0; \\ x_{k+1} = x_k - \dfrac{f(x_k)}{f'(x_k)}, \quad k \geq 0. \end{cases}$$

Ovviamente non possiamo calcolare gli infiniti termini della successione per cui abbiamo bisogno di introdurre un qualche criterio di arresto. Il primo programma fissa a priori il numero n di termini calcolati restituendo l'ultimo come approssimazione di α.

```
function x=mysqrt1(x0,a,n)
%calcola n termini della successione generata dal metodo delle tangenti
%con punto iniziale x0 per approssimare $\sqrt{a}$
 for k=1:n
 x=x0-(x0*x0-a)/(2*x0);
 % alternativamente x= (x0/2)+a/(2*x0)
  x0=x;
 end
end
```

```
>> x=mysqrt1(1,2,5)
ans = 1.414213562373095e+00
```

Si può migliorare la leggibilità e fruibilità del programma inserendo la funzione in modo più trasparente. Il modo più semplice è quello di utilizzare "anonymous functions".

> MATLAB's anonymous functions provide an easy way to specify a function. This is essential for problems that include solving a nonlinear equation, integrating or differentiating a function, minimizing a function or a solving differential equation.

La sintassi è

function_name=@(variable_name) matlab_expession

La seguente implementazione modifica il precedente programma utilizzando "anonymous functions" per definire $f(x)$ e $f'(x)$.

```
function x=mysqrt2(x0,a,n)
%calcola n termini della successione generata dal metodo delle tangenti
%con punto iniziale x0 per approssimare $\sqrt{a}$
f=@(x)x*x-a;
f1=@(x)2*x;
 for k=1:n
 x=x0-f(x0)/f1(x0);
  x0=x;
 end
end
```

```
>> x=mysqrt2(1,2,5)
ans = 1.414213562373095e+00
```

La seguente implementazione introduce f e $f1$ come parametri di input.

```
function x=mysqrt3(x0,f,f1,n)
%calcola n termini della successione generata dal metodo delle tangenti
%con punto iniziale x0 per approssimare $\sqrt{a}$
 for k=1:n
 x=x0-f(x0)/f1(x0);
  x0=x;
 end
end
```

```
>> f=@(x)x^2-2; f1=@(x)2*x; x=mysqrt3(1,f,f1,5)
ans = 1.414213562373095e+00
```

Nelle prossime lezioni adatteremo questi programmi per introdurre criteri di arresto più appropriati.

Esercizio 2. Scrivere un programma MatLab che calcola il logaritmo naturale di $x > 0$ risolvendo

$$f(z) = e^z - x = 0.$$

IL programma deve accettare in ingresso x, $x0$ ed n ed eseguire n iterazioni del metodo delle tangenti con punto iniziale $x0$ restituendo in uscita l'approssimazione x_n di $\log x$. Riportare i risultati ottenuti per $x0 = 1$, $n \in 10, 100$ e $x \in 10, 100$.