Laboratorio computazionale numerico
Lezione 1

Federico Poloni <f.poloni@sns.it>

2009-10-07

1 Primo programma

Lanciamo Octave con il comando `octave` in una finestra di terminale (shell).

```
octave:1> 'Hello, world'
ans = Hello, world
```

2 Primi calcoli in virgola mobile

Octave utilizza la doppia precisione (8 byte per ogni numero).

```
octave:1> realmin
ans =  2.2251e-308
octave:2> realmax
ans =  1.7977e+308
octave:3> eps
ans =  2.2204e-16
octave:4> realmin/2   %'gradual underflow'
ans =  1.1125e-308
```

Octave come una calcolatrice:

```
octave:1> 1+1
ans =  2
octave:2> 10^10
ans =  1.0000e+10
octave:3> 1e10
ans =  1.0000e+10
octave:4> (1e10)^2
ans =  1.0000e+20
```

Se c'è un punto e virgola alla fine della linea, Octave esegue il calcolo ma non scrive il risultato

```
octave:1> 1+1;
octave:2>
```

Perdita di precisione da alcuni calcoli:

```
octave:1> a=1e10
a =  1.0000e+10
octave:2> b=1e4
```

```
b =  10000
octave:3> c=(a+b)^2
c =  1.0000e+20
octave:4> format long     % scrive piu' cifre
octave:5> c
c =  1.00000200000100e+20
octave:6> c - a^2 - 2*a*b - b^2
ans =  7936
```

Principalmente da sottrazioni tra due numeri grossi e molto vicini, (*errori di cancellazione*), ma anche da moltiplicazioni:

```
octave:1> a=98
a =  98
octave:2> 1 - a*(1/a)
ans =  1.1102e-16
octave:3> a=97
a =  97
octave:4> 1 - a*(1/a)
ans = 0
```

Quando succede? Controlliamo con un breve programma.

```
% file:perditaprec.m
% questo e' un commento
% octave puo' eseguire le istruzioni contenute in un file
% di testo *nella cartella in cui viene lanciato*

for k=1:300
  a=k*(1/k);
  if(a ~= 1)
    k      %scrive il valore di k
  endif
endfor
```

```
octave:1> perditaprec
k =  49
k =  98
k =  103
k =  107
k =  161
k =  187
k =  196
k =  197
k =  206
k =  214
k =  237
k =  239
k =  249
k =  253
```

Se l'output occupa più di una schermata, Octave usa il programma `less` per mostrarlo: rimpiazzando il 300 con 1000,

```
k =  49
k =  98
k =  103
```

```
k =  107
k =  161
k =  187
k =  196
k =  197
k =  206
k =  214
k =  237
k =  239
k =  249
k =  253
k =  322
k =  347
k =  374
k =  389
k =  392
k =  394
k =  412
k =  417
k =  425
k =  428
:
```

Nell'ultima riga c'è il prompt ":", che indica che stiamo visualizzando un output che occupa più di una schermata. Con i tasti freccia si scorre l'output, con il tasto `q` si esce e si ritorna al prompt di octave

Possiamo controllare che per i valori indicati a∗(1/a) è diverso da 1 (in doppia precisione).

3 Funzioni e accumulatori

```
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;
endfor
% ora f vale n!
endfunction
```

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

```
octave:1> fact(10)
ans =  3628800
```

Esercizio 1. Scrivi una funzione pow(x,n) che calcoli x^n

4 Calcolo dell'esponenziale

```
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);
  endfor
endfunction
```

Qualche esperimento su quanti termini servono per approssimare bene.

```
octave:1> myexp(1,5)
ans =  2.7167
octave:2> myexp(10,50)
ans =  2.2026e+04
octave:3> exp(10)
ans =  2.2026e+04
octave:4> format long
octave:5> myexp(10,50)
ans =  22026.4657948067
octave:6> exp(10)
ans =  22026.4657948067
```

Due problemi:

- Inaccurato: prova **exp**(−20,500)
- Lento: con n termini, il numero di operazioni da eseguire cresce come n^2

Risolviamo (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;
  endfor
endfunction
```

Ora va meglio:

```
octave:1> myexp(-20,500)
ans = NaN
octave:2> myexp2(-20,500)
ans =  5.62188447213042e-09
```

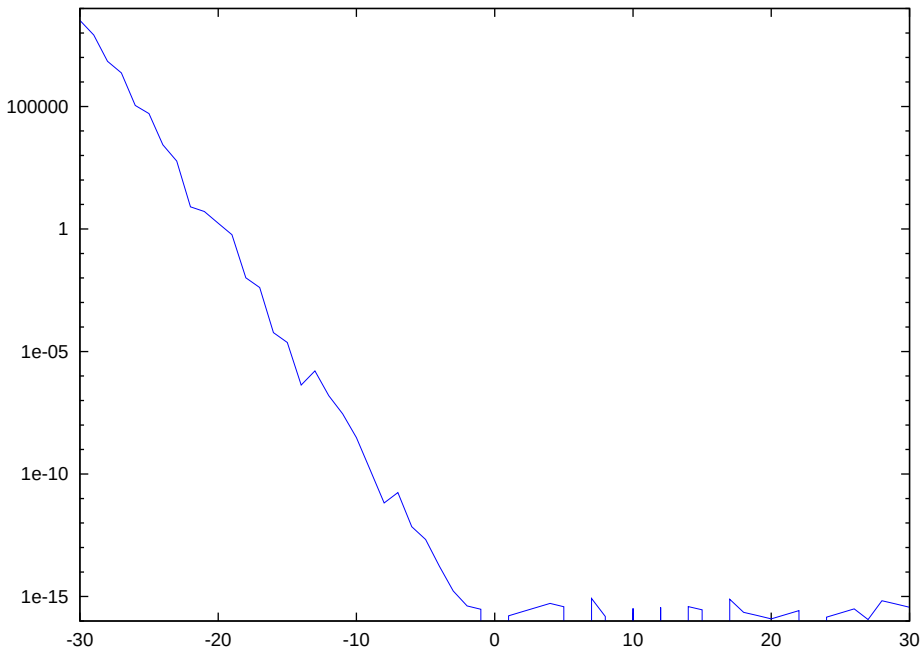
Cosa succedeva?

```
octave:27> fact(500)
ans = Inf
octave:28> pow(-20,500)
ans = Inf
octave:29> Inf/Inf
ans = NaN
```

Ci sono ancora pesanti accuratezze sui numeri negativi:

```
octave:36> myexp2(-30,500)
ans = -3.06681235635622e-05
```

Un esponenziale negativo è un brutto segno... Pesanti errori di cancellazione:



Errore relativo: $\frac{|\text{exptaylor2}(x)-e^x|}{e^x}$

La soluzione: cambiare algoritmo e sceglierne uno che non porti a errori di cancellazione

```
octave:2> exp(-30)
ans =  9.3576e-14
octave:3> myexp2(-30,500)
ans = -3.0668e-05
octave:4> 1/myexp2(30,500)
ans =  9.3576e-14
octave:5> format long
octave:6> exp(-30)
ans =  9.35762296884017e-14
octave:7> 1/myexp2(30,500)
ans =  9.35762296884017e-14
```

Esercizio 2. Scrivere una funzione myexp(x) che controlla se x è negativo o positivo, e calcola rispettivamente $1/e^{-x}$ e e^x con la serie di Taylor troncata a $n = 500$.

Se vi state annoiando.... Scrivere una funzione solve2(a,b,c) che risolve l'equazione di secondo grado $ax^2 + bx + c = 0$ in nel modo più stabile possibile (hint: ci sono al massimo sottrazioni. Una è necessaria (perché?); l'altra no). Provare su $x^2 - (10^{10} + 10^{-10})x + 1 = 0$.

Se vi state annoiando... Guardare su wikipedia l'algoritmo di sommazione di Kahan. Implementare, testare su qualche sequenza che causa errori di cancellazione.