Tutoraggio #1

Federico Pichi

28 marzo 2019

Esercizio 1

Si dimostri che i^i è un numero reale e si verifichi il risultato in MATLAB.

Esercizio 2

Si consideri in MATLAB la seguente operazione: $((1+x)-1)/x$ con $x \neq 0$, il cui risultato esatto è ovviamente 1 per ogni $x \neq 0$. Si commenti il risultato prendendo ad esempio $x = 1.e$-15.

Esercizio 3

Si scriva in MATLAB uno script in grado di calcolare il valore della precisione di macchina (ottenibile con il comando *eps*), ovvero tale che 1.0 + eps > 1.0.

Esercizio 4

Si scriva in MATLAB uno script in grado di calcolare la rappresentazione di un dato numero intero a secondo la base b.

Esercizio 5

Si consideri la successione

$$z_2 = 2, \quad z_{n+1} = 2^{(n-1)/2} - \sqrt{1 - \sqrt{1 - 4^{(1-n)} z_n^2}}, \quad \text{per } n = 2, 3, \dots$$

che converge a π per $n \to \infty$. Fare un plot dell'errore relativo fra π e z_n e spiegarne l'andamento.

Esercizio 6

Per il calcolo di π si può usare la seguente tecnica: si generano n coppie (x_k, y_k) di numeri casuali compresi fra 0 e 1 e di questi si calcola il numero m di punti che cadono nel primo quarto del cerchio di centro l'origine e raggio 1. Si ha che π è il limite per n che tende all'infinito dei rapporti $\pi_n = 4m/n$. Si scriva un programma MATLAB che esegua questo calcolo e si verifichi la correttezza del risultato al crescere di n.

Esercizio 7

Si costruiscano in MATLAB una matrice triangolare superiore ed una triangolare inferiore di dimensione 10 con 2 sulla diagonale principale e -3 sulla seconda sopra (rispettivamente, sotto) diagonale. Si scambino terza e settima riga e successivamente quarta e ottava colonna.

Soluzione 1

Dalla formula di eulero si ottiene la rappresentazione equivalenti in forma matriciale ed esponenziale $z = \rho e^{i\theta} = \rho(\cos\theta + i\sin\theta)$. Quindi $i = e^{i\pi/2}$, quindi $i^i = e^{-\pi/2}$. Comandi utili: *complex(x,y)*, *abs(z)*, *angle(z)*, *real(z)*, *imag(z)*.

Codice 1: Soluzione 1

```
>> 1i^1i
ans = 0.2079
```

Soluzione 2

Un fenomeno problematico si verifica quando si sommano tra loro numeri che hanno all'incirca lo stesso modulo, ma segno opposto. In tal caso il risultato della somma può essere assai impreciso e ci si riferisce a questa situazione con l'espressione *cancellazione di cifre significative*. Troviamo invece:

Codice 2: Soluzione 2

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

Soluzione 3

Partiamo da my_eps = 0.5 e consideriamo la variabile eps_1 = my_eps+1 che è sicuramente maggiore di 1. Allora dimezziamo il valore di my_eps e riconsideriamo la quantità eps_1. Dimezzeremo ogni volta il valore di my_eps fino a quando sarà vera la diseguaglianza eps_1 > 1. Non appena eps_1 ≤ 1, vuol dire che abbiamo superato il valore critico di my_eps, per cui il valore della precisione di macchina è dato dall'ultimo valore preso in esame prima che non valesse più la diseguaglianza eps_1 > 1.

Codice 3: Soluzione 3

```
n_max = 100;
n = 0 ;
my_eps = 0.5;
eps_1 = my_eps+1;
while (eps_1 > 1 && n < n_max)
  n = n+1;
  my_eps = 0.5*my_eps;
  eps_1 = my_eps+1;
end

my_eps=2.*my_eps;
fprintf(1, 'epsilon = \%12.8e con numero di iterazioni n = \%2d \n', my_eps, n)
fprintf(1, 'epsilon di macchina = \%12.8e \n', eps)
```

Soluzione 4

Per rappresentare i numeri interi $a > 0$ si usano scritture del tipo $a_1, a_2, \dots, a_n$ dpve gli a_i sono cifre del sistema di numerazione in base b, ovvero tali che: $a = \sum_{i=1}^{n} a_i b^{n-i}$. Con un gruppo di n cifre e $a_1 \neq 0$ si possono rappresentare tutti i numeri interi compresi tra b^{n-1} e $b^n - 1$, allora per rappresentare un intero $a > 0$ sono sempre necessarie $1 + \lfloor \log_b a \rfloor$ cifre.

Codice 4: Soluzione 4

```
a = 10;
b = 2;
n = 1 + floor(log(a)/log(b));
q = a;
a_vec = zeros(1,n);
```

```
for i = n:-1:1
  a_vec(n-i+1) = q - b.*floor(q/b);
  q = floor(q/b);
end
```

Soluzione 5

Questa successione è una riscrittura della più nota formula di Francois Viete (matematico francese del XVI secolo) per l'approssimazione di π. Se utilizziamo MATLAB per calcolare z_n, troveremo che l'errore relativo fra π e z_n decresce fino a $n = 16$ per poi cominciare a crescere a causa degli *errori di arrotondamento*.

Codice 5: Soluzione 5

```
z_n = 2;
n_it = 30;
err_rel = zeros(1, n_it);
n_vec = 2:n_it+1;

for n = 2:n_it
    z_np = 2^(n-1/2)*sqrt(1 - sqrt(1 - 4^(1-n)*z_n^2));
    z_n = z_np;
    err_rel(n) = abs(pi - z_n)/pi;
end

semilogy(n_vec, err_rel)
title('Errore relativo')
xlabel('n')
ylabel('err')
```

Soluzione 6

Codice 6: Soluzione 6

```
n_it_1 = 1000;
pi_vec = zeros(1, n_it_1);
err_rel_1 = zeros(1, n_it_1);
n_vec = 2:n_it_1+1;

for n = 2:n_it_1
    pi_vec(n) = pi_montecarlo(n);
    err_rel_1(n) = abs(pi - pi_vec(n))/pi;
end

figure()
semilogy(n_vec, err_rel_1)
title('Errore relativo')
xlabel('n')
ylabel('err')

function my_pi = pi_montecarlo(n)
  x = rand(n,1);
  y = rand(n,1);
  z = x.^2+y.^2;
  v = (z <= 1);
  m=sum(v);
  my_pi=4*m/n;
end
```

Soluzione 7

Codice 7: Soluzione 7

```
L = 2*eye(10)-3*diag(ones(8,1),-2);
U = 2*eye(10)-3*diag(ones(8,1),2);
r = 1:10;  r(3) = 7;  r(7) = 3;
Lr = L(r,:);
c = 1:10;  c(8) = 4;  c(4) = 8;
Lc = L(:,c);
```