Esercizio 1.1 *Sia* $x = e \approx 2.7183 = \tilde{x}$. *Si calcoli il corrispondente errore relativo* ε_x *e il numero di cifre significative* k *con cui* $\tilde{x}$ *approssima* x. *Si verifichi che*

$$|\varepsilon_x| \approx \frac{1}{2}10^{-k}.$$

Soluzione: Ricordiamo la definizione di *errore relativo*

$$\varepsilon_x = \frac{\tilde{x} - x}{x}$$

Il numero di Nepero e in MatLab rappresentato tramite la funzione *exp(1)* è:

$$2.718281828459046$$

Applicando la definizione abbiamo:

$$|\varepsilon_x| = \frac{|2.7183 - 2.718281828459046|}{|2.718281828459046|} = 6.684936331611679 * 10^{-6}$$

Adesso si verifica che:

- $\tilde{x} = 2.7183$ approssima e con $k = 5$ cifre significative
- $\frac{1}{2}10^{-k} = 0.5 \times 10^{-5} \approx 6.68 \times 10^{-6} = |\varepsilon_x|$

Esercizio 1.2 *Usando gli sviluppi di Taylor fino al secondo ordine con resto in forma di Lagrange, si verifichi che se* $f \in C^3$ *, risulta*

$$f'(x) = \phi_h(x) + O(h^2)$$

dove

$$\phi_h(x) = \frac{f(x+h) - f(x-h)}{2h}$$

Soluzione: Sia $f \in C^3$ e sia $f_T(x)$ l'approssimazione al secondo ordine di f mediante il polinomio di Taylor con resto di Lagrange centrato nel punto x_0. Ricordando che:

$$P_n(x) = \sum_{k=0}^{n} \frac{f^{(k)}(x_0)}{k!}(x - x_0)^k$$

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

Abbiamo:

$$f_T(x) = P_2(x) + R_{2,x_0}(x)$$

$$f_T(x) = f(x_0) + f'(x_0)(x - x_0) + \frac{f''(x_0)}{2}(x - x_0)^2 + \frac{f'''(c)}{6}(x - x_0)^3$$

Quindi, considerando il rapporto incrementale che definisce $f'(x)$:

$$f_T(x + h) = f(x) + f'(x)h + \frac{1}{2}f''h^2 + \frac{f'''(c)}{6}h^3$$

$$f_T(x - h) = f(x) - f'(x)h + \frac{1}{2}f''h^2 + \frac{f'''(c)}{6}h^3$$

Si noti che tutte derivate che compaiono in $f_T(x)$ esistono dato che $f \in C^3$. Si procede ora a mostrare che $f'(x) = \frac{f(x+h)-f(x-h)}{2h} + O(h^2)$ sostituendo f con f_T nel rapporto incrementale.

$$f'(x) = \frac{f_T(x + h) - f_T(x - h)}{2h}$$

$$= \frac{f(x) + f'(x)h + \frac{1}{2}f''(x)h^2 + \frac{f'''(c)}{6}h^3 - f(x) + f'(x)h - \frac{1}{2}f''(x)h^2 + \frac{f'''(c)}{6}h^3}{2h}$$

$$= \frac{2hf'(x) + \frac{f'''(c)}{3}h^3}{2h}$$

$$= f'(x) + \frac{\frac{f'''(c)}{3}h^3}{2h}$$

$$= f'(x) + \frac{f'''(c)}{6}h^2$$

Esprimiamo il termine che rappresenta il resto di Lagrange $\frac{f'''(c)}{6}h^2$ tramite la notazione $O(h^2)$ ed otteniamo la tesi.

$$\frac{f(x + h) - f(x - h)}{2h} = f'(x) + O(h^2)$$

Esercizio 1.3 *Utilizzando Matlab, si costruisca una tabella dove, per* $h = 10^{-j}$, $j = 1, ..., 10$ *e per la funzione* $f(x) = x^4$ *si riporta il valore di* $\phi_h(x)$ *definito nell'esercizio 1 in* $x = 1$. *Commentare i risultati ottenuti.*

Soluzione:

```
function [] = es1_32(x, itmax)
    % -x: valore passato alla funzione
    % -itmax: massimo numero di iterazione
    format long e
    i = 1;
    while (i<=itmax)
        h = 10^-i;
        f = ((x+h)^4-(x-h)^4)/(2*h);
        str = sprintf('x%d =', -i);
        disp(str), disp(f)
        i = i+1;
    end
end
```

Funzione che produce i seguenti risultati:

h	$\phi_h(1)$
10^{-1}	4.040000000000002
10^{-2}	4.000400000000004
10^{-3}	4.000003999999723
10^{-4}	4.000000039999230
10^{-5}	4.000000000403681
10^{-6}	3.999999999948489
10^{-7}	4.000000000115023
10^{-8}	4.000000003445692
10^{-9}	4.000000108916879
10^{-10}	4.000000330961484

Si nota che all'aumentare di i, quindi al diminuire di h, ϕ_h diminuise che sta a significare un aumento di precisione del risultato approssimato.

Esercizio 1.4 *Si dia una maggiorazione del valore assoluto dell'errore relativo con cui* $x + y + z$ *viene approssimato dall'approssimazione prodotta dal calcolatore, ossia* $(x \oplus y) \oplus z$ *(supporre che non ci siano problemi di overflow o di underflow). Ricavare l'analoga maggiorazione anche per* $x \oplus (y \oplus z)$ *tenendo presente che* $x \oplus (y \oplus z) = (y \oplus z) \oplus x$.

Soluzione: Ricordiamo che l'apposimazione $(x \oplus y) \oplus z$ sarà equivalente al seguente errore:

$$\varepsilon_1 = \frac{(x\varepsilon_x + y\varepsilon_y) + z\varepsilon_z}{(x+y)+z}$$

Chiamando ε_{max} il massimo tra ε_x, ε_y e ε_z otteniamo:

$$\varepsilon_1 = \frac{|(x+y)| + |z|}{|(x+y)+z|}\varepsilon_{max}$$

Per lo stesso motivo $x \oplus (y \oplus z)$ sarà :

$$\varepsilon_2 = \frac{|x| + |(y+z)|}{|x+(y+z)|}\varepsilon_{max}$$

Esercizio 1.5 *Eseguire le seguenti operazioni in Matlab:*

$x = 0$; $count = 0$;

$while\ x \sim 1 = 1$, $x = x + delta$, $count = count + 1$, end

dapprima ponendo delta = 1/16 e poi ponendo delta = 1/20. Commentare i risultati ottenuti e in particolare il non funzionamento nel secondo caso.

Soluzione: L'algoritmo viene eseguito correttamente se poniamo
$delta = 1/16$.
Invece ponendo $delta = 1/20$ il codice va in loop:
Questo è dovuto perchè la rappresentazione di 1/20 (0.05) in binario equivale a $0.00\overline{0011}$. Essendo *delta* periodico, deve essere approssimato. Questo errore di approssimazione comporta a diventare sempre più grosso dopo ogni iterazione. L'errore di approssimazione fa ottenere un valore diverso da 1. Ecco perchè non sarà mai uguale a 1.

Esercizio 1.6 *Verificare che entrambe le seguenti successioni convergono a* $\sqrt{3}$*, (riportare le successive approssimazioni in una tabella a due colonne, una per ciascuna successione),*

$$x_{k+1} = (x_k + \tfrac{3}{x_k})/2, \qquad x_0 = 3;$$
$$x_{k+1} = (3 + x_{k-1}x_k)/(x_{k-1} + x_k), \qquad x_0 = 3;\ x_1 = 2.$$

Per ciascuna delle due successioni, dire quindi dopo quante iterazioni si ottiene un'approssimazione con un errore assoluto minore o uguale a 10^{-12} *in valore assoluto.*

Soluzione: Entrambe le successioni convergono a $\sqrt{3}$. Notiamo che la prima successione converge più velocemente; infatti dopo cinque iterazioni abbiamo come risultato 1.7320508075689:

Prima successione:

k	$x(k)$
0	3.0000000000000
1	2.0000000000000
2	1.7500000000000
3	1.7321428571429
4	1.7320508100147
5	1.7320508075689

Seconda successione:

k	$x(k)$
0	3.0000000000000
1	2.0000000000000
2	1.8000000000000
3	1.7368421052632
4	1.7321428571429
5	1.7320509347060
6	1.7320508075723
7	1.7320508075689

Per rispondere alla seconda domanda ricordiamo la definizione di valore assoluto:

$$\Delta x = \tilde{x} - x$$

Dato che si tratta di una successione, parliamo di errore assoluto di convergenza commesso ad ogni passo dell'iterazione, con x_k risultato intermedio ed x valore da approssimare ($\sqrt{3}$):

$$\Delta x_k = x_k - x$$

Per svolgere i conti è stato utilizzato il seguente script di Matlab dove è stato calcolato il valore assoluto ad ogni iterazione:

```
clc;
format long
f1 = @(x) (x + 3/x)/2; % Prima successione
f2 = @(x0, x1) (3 + x0*x1)/(x0 + x1); % Seconda successione
x = sqrt(3); % Valore da convergere

xk = 3; %Innesco per la successione f1

k = 1;
e = 10^(-12);
diff = abs(xk - sqrt(3));

disp('Prima successione: ');
fprintf('k = 0\t\tx(0) = %.13f\t err = %.15f', xk, diff);

while ((e <= diff) && xk ~= sqrt(3))
   xk = f1(xk);
```

```matlab
    diff = abs(xk - sqrt(3));
    fprintf('\nk = %d\t\tx(%d) = %.13f\t err = %.15f', k, k, xk, (diff));
    k = k + 1;
end

fprintf(successione:);
x0 = 3; %innesco per succeccione f2
x1 = 2; %innesco per successione f2
xk = 2; %innesco che cambiera nel ciclo
k = 2;
diff = x0 - sqrt(3);
fprintf('k = 0\t\tx(0) = %.13f \t err = %.15f', x0, diff);
diff = abs(x1 - sqrt(3));
fprintf('\nk = 1\t\tx(1) = %.13f \t err = %.15f', x1, diff);
while ((e <= diff) && (x0 ~= sqrt(3)))
    xk = f2(x0, x1);
    diff = abs(xk - sqrt(3));
    fprintf('\nk = %d\t\tx(%d) = %.13f\t err = %.15f', k, k, xk, (diff));
    x0 = x1;
    x1 = xk;
    k = k + 1;
end
```

```
Prima successione:
k = 0           x(0) = 3.0000000000000   err = 1.267949192431123
k = 1           x(1) = 2.0000000000000   err = 0.267949192431123
k = 2           x(2) = 1.7500000000000   err = 0.017949192431123
k = 3           x(3) = 1.7321428571429   err = 0.000092049573980
k = 4           x(4) = 1.7320508100147   err = 0.000000002445850
k = 5           x(5) = 1.7320508075689   err = 0.000000000000000

Seconda successione:
k = 0           x(0) = 3.0000000000000   err = 1.267949192431123
k = 1           x(1) = 2.0000000000000   err = 0.267949192431123
k = 2           x(2) = 1.8000000000000   err = 0.067949192431123
k = 3           x(3) = 1.7368421052632   err = 0.004791297694281
k = 4           x(4) = 1.7321428571429   err = 0.000092049573980
k = 5           x(5) = 1.7320509347060   err = 0.000000127137164
k = 6           x(6) = 1.7320508075723   err = 0.000000000003379
k = 7           x(7) = 1.7320508075689   err = 0.000000000000000>>
```