

Laboratorio di Calcolo Scientifico
a.a. 2002/03
SOLUZIONE HOMEWORK #1

Udine, 22 maggio 2003

Prof.ssa R. Vermiglio, Dott. Ing. D. Breda
Dipartimento di Matematica e Informatica,
Università degli Studi di Udine
vermiglio@dimi.uniud.it
dbreda@dimi.uniud.it
<http://www.dimi.uniud.it/~rossana>
<http://www.dimi.uniud.it/~dbreda>

1. Si considerino le due espressioni matematicamente equivalenti

$$r_1 = \frac{ay + b}{ay + c}, \quad r_2 = \frac{a + b/y}{a + c/y}$$

e si calcolino i loro valori per

$$a = b = \beta^{-p_{min}-t}, \quad c = 2(\beta - 1)\beta^{-p_{min}-t}, \quad y = (\beta - 1)\beta^{-1}$$

utilizzando MATLAB ovvero secondo lo standard IEEE doppia precisione basato sul sistema *floating point* $\mathcal{F}(\beta, t, p_{min}, p_{max}) = \mathcal{F}(2, 53, 1021, 1024)$ a cui si aggiungono i numeri denormalizzati, cioè con esponente $p = -p_{min} - 1$ e prima cifra nulla. Si dica quale delle due formule fornisce il risultato esatto confrontando i valori calcolati con MATLAB con il risultato teorico ottenuto analiticamente in aritmetica esatta. Si spieghi la differenza tra i valori computati dando una giustificazione analitica per il risultato errato.

sol.1 Le istruzioni MATLAB sono le seguenti:

```
>>beta=2;t=53;pmin=1021;  
>>a=beta^(-pmin-t),b=a,c=2*(beta-1)*beta^(-pmin-t),...  
y=(beta-1)*beta^(-1)  
a=  
4.9407e-324
```

```

b=
    4.9407e-324
c=
    9.8813e-324
y=
    0.5000
>>r1=(a*y+b)/(a*y+c)
r1=
    0.5000
>>r2=(a+b/y)/(a+c/y)
r2=
    0.6000

```

In aritmetica esatta si ha:

$$\begin{aligned}
 r_2 = r_1 &= \frac{ay + b}{ay + c} = \\
 &= \frac{\beta^{-p_{min}-t}(\beta - 1)\beta^{-1} + \beta^{-p_{min}-t}}{\beta^{-p_{min}-t}(\beta - 1)\beta^{-1} + 2(\beta - 1)\beta^{-p_{min}-t}} = \\
 &= \frac{2 - \frac{1}{\beta}}{2\beta - 1 - \frac{1}{\beta}} = \\
 &= \frac{2\beta - 1}{2\beta^2 - \beta - 1}|_{\beta=2} = \frac{3}{5} = 0.6
 \end{aligned}$$

per cui il valore esatto con MATLAB lo si ottiene usando la formula r_2 . L'uso della formula r_1 con i dati assegnati non fornisce il valore esatto, ma un valore approssimato con un errore di $\simeq 17\%$. Tale risultato è dovuto all'*underflow*: il valore $a = b = \beta^{-p_{min}-t} = 2^{-1074} \simeq 4.9407 \times 10^{-324}$ (che corrisponde a $realmin * eps$) è infatti il numero minimo rappresentabile con MATLAB per cui la moltiplicazione per $y = 1/2$ fornisce il valore $ay = by = 0$. Risulta perciò in aritmetica di macchina

$$r_1 = \frac{ay \oplus b}{ay \oplus c} = \frac{b}{c} = \frac{1}{2(\beta - 1)}|_{\beta=2} = \frac{1}{2} = 0.5.$$

2. Si scriva una *function* MATLAB per il calcolo del valore approssimato della derivata prima di una funzione in un punto usando le differenze finite centrate

$$f'(x) \approx \frac{f(x+h) - f(x-h)}{2h}.$$

Si testi la *function* per la funzione esponenziale $f(x) = e^x$ per $x = 1$, valutando l'errore assoluto commesso rispetto al valore esatto di $f'(x)$ per un passo variabile $h = 10^{-1}, 10^{-2}, \dots, 10^{-16}$. Si rappresentino su scala bilogarithmica l'andamento dell'errore assoluto e delle maggiorazioni dell'errore analitico e di arrotondamento (assumendo che valutando f si commette un'errore di arrotondamento maggiorato dalla precisione macchina) in funzione del passo h . La *function* deve essere del tipo

[erass,erana,erarr]=nome_function(x,h)

e deve provvedere alla rappresentazione grafica dei risultati. (*Sugg.: per determinare una maggiorazione dell'errore analitico si utilizzi lo sviluppo di Taylor per $f(x+h)$ e per $f(x-h)$ arrestandosi al terz'ordine e si faccia la differenza dei due sviluppi ricordando che esiste $\theta \in [x-h, x+h]$ tale che $f'''(\theta_+) + f'''(\theta_-) = 2f'''(\theta)$ per qualche $\theta_+ \in [x, x+h]$ e $\theta_- \in [x-h, x]$)*

Sol.2 Poichè $f \in \mathcal{C}^3[a, b]$, a, b tali che $x+h, x-h \in [a, b]$, si può applicare lo sviluppo di Taylor arrestato al terz'ordine ad $f(x+h)$ e ad $f(x-h)$

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

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

per qualche $\theta_+ \in (x, x+h)$ e $\theta_- \in (x-h, x)$. Sottraendo la seconda dalla prima si ottiene

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

per cui dividendo per $2h$ e considerando che esiste $\theta \in [x-h, x+h]$ tale che $f'''(\theta_+) + f'''(\theta_-) = 2f'''(\theta)$ si ottiene una limitazione superiore dell'errore analitico (assoluto)

$$\left| \frac{f(x+h) - f(x-h)}{2h} - f'(x) \right| \leq \frac{Mh^2}{6}$$

dove $M = \max_{\theta \in [a, b]} |f'''(\theta)|$.

Se si assume poi che l'errore (assoluto) commesso nella valutazione di f sia limitato dalla precisione macchina ϵ , l'errore inerente (assoluto) dovuto all'arrotondamento nella valutazione di f nelle differenze finite centrate è limitato da

$$\left| \frac{\text{fl}(f(x+h)) - \text{fl}(f(x-h))}{2h} - \frac{f(x+h) - f(x-h)}{2h} \right| \leq \frac{\epsilon}{h}.$$

L'errore dovuto alla sottrazione e alla divisione è indipendente da h per cui si trascurava.

L'errore (assoluto) totale risulta dunque limitato da

$$\left| \frac{\text{fl}(f(x+h)) - \text{fl}(f(x-h))}{2h} - f'(x) \right| \leq \frac{Mh^2}{6} + \frac{\epsilon}{h} = E(h)$$

Si osserva che il comportamento rispetto ad h dell'errore analitico e dell'errore inerente in seguito all'arrotondamento nella valutazione di f è diverso

per $h \downarrow 0$. Diventa interessante stimare il valore ottimale h_m (i.e. il minimo di $E(h)$): differenziando $E(h)$ secondo h e ponendo uguale a zero si osserva che il limite sull'errore totale presenta un minimo per

$$h_m = \sqrt[3]{\frac{3\epsilon}{M}}.$$

Si consideri ora $f(x) = e^x$ per $x = 1$. La derivata prima è $f'(x) = e^x$, per cui risulta $f'(1) \simeq 2.718281828459046$; la derivata terza è $f'''(x) = e^x$ per cui si prenda $M = 3.0042 \geq e^{1.1}$ come maggiorazione nell'intervallo $[0.9, 1.1]$. Considerando la precisione macchina del MATLAB, pari a $\epsilon \simeq 2.22e-16$, si ottiene $h_m \simeq 6.7372e-06$. I risultati sono riportati in tabella 1 e confermano quanto previsto.

h	erass	erana	erarr
10^{-01}	4.532735488372186e-03	5.006943373244056e-03	2.220446049250313e-15
10^{-02}	4.530492364640537e-05	5.006943373244056e-05	2.220446049250313e-14
10^{-03}	4.530466783947418e-07	5.006943373244055e-07	2.220446049250313e-13
10^{-04}	4.530565256288810e-09	5.006943373244056e-09	2.220446049250313e-12
10^{-05}	5.858735718788921e-11	5.006943373244055e-11	2.220446049250313e-11
10^{-06}	1.634576918263520e-10	5.006943373244056e-13	2.220446049250313e-10
10^{-07}	5.858691309867936e-11	5.006943373244054e-15	2.220446049250313e-09
10^{-08}	6.602751234652260e-09	5.006943373244056e-17	2.220446049250313e-08
10^{-09}	6.602751234652260e-09	5.006943373244056e-19	2.220446049250313e-07
10^{-10}	6.727365660097462e-07	5.006943373244056e-21	2.220446049250313e-06
10^{-11}	1.042949367979773e-05	5.006943373244057e-23	2.220446049250313e-05
10^{-12}	2.102696381127700e-04	5.006943373244055e-25	2.220446049250313e-04
10^{-13}	4.558641766623239e-04	5.006943373244055e-27	2.220446049250313e-03
10^{-14}	9.337648373663576e-03	5.006943373244056e-29	2.220446049250313e-02
10^{-15}	1.682980355663610e-01	5.006943373244056e-31	2.220446049250313e-01
10^{-16}	4.978357792087325e-01	5.006943373244055e-33	2.220446049250313e+00

Tabella 1: Errore nell'uso delle differenze finite centrate nel calcolo della derivata prima.

La seguente *function* di MATLAB realizza l'analisi dell'errore di approssimazione della derivata prima per $f(x) = e^x$:

```
function [erass,erana,erarr]=diffincen(x,h)

d1f=exp(x)*ones(length(h),1);           %derivata prima esatta
d1fc=(exp(x+h)-exp(x-h))./(2*h);         %derivata prima appross.
erass=abs(d1fc-d1f);                      %errore assoluto
M=max(exp(x-max(h):.001:x+max(h)));      %magg. derivata terza
erana=abs(M*h.^2/6);                      %errore analitico
```

```

erarr=eps./h;                                %errore arrotondamento
ertot=erana+erarr;                            %errore totale
loglog(h,erass,'-b');
hold on; grid on;
loglog(h,erana,'--g');
loglog(h,erarr,'--c');
loglog(h,ertot,'-.r');

```

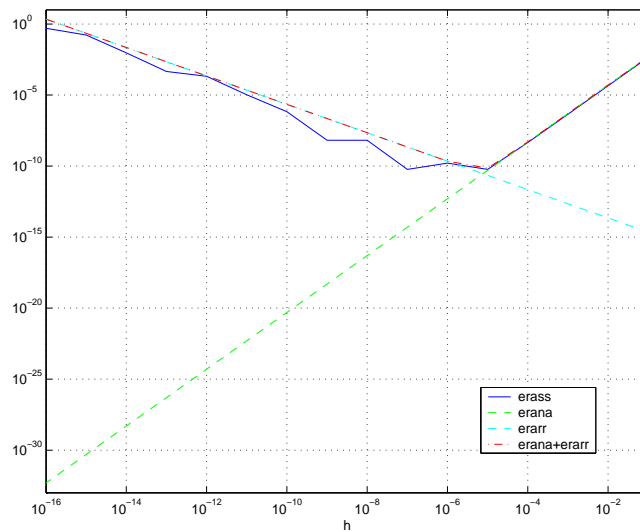


Figura 1: Errori nell'approssimazione della derivata con le differenze finite centrate al variare del passo h .

3. Si calcolino le seguenti espressioni matematicamente equivalenti

$$\sqrt{n+1} - \sqrt{n} = \frac{1}{\sqrt{n+1} + \sqrt{n}}$$

per $n = 10, 10^2, \dots, 10^{16}$ con MATLAB ovvero secondo lo standard IEEE doppia precisione. Si spieghino i risultati ottenuti e si analizzi l'errore algoritmico di entrambe le espressioni.

Sol.3 Le istruzioni MATLAB sono le seguenti

```

>>format long e
>>n=10.^[1:16]';
>>alg1=sqrt(n+1)-sqrt(n);
>>alg2=1./(sqrt(n+1)+sqrt(n));

```

e i risultati sono riportati in tabella 2.

n	$\sqrt{n+1} - \sqrt{n}$	$\frac{1}{\sqrt{n+1} + \sqrt{n}}$
10^{01}	1.543471301870203e-01	1.543471301870205e-01
10^{02}	4.987562112088995e-02	4.987562112089027e-02
10^{03}	1.580743742895763e-02	1.580743742895582e-02
10^{04}	4.999875006248544e-03	4.999875006249609e-03
10^{05}	1.581134877255863e-03	1.581134877256878e-03
10^{06}	4.999998750463419e-04	4.999998750000625e-04
10^{07}	1.581138790243131e-04	1.581138790555721e-04
10^{08}	5.000000055588316e-05	4.99999987500000e-05
10^{09}	1.581139076733962e-05	1.581138829688905e-05
10^{10}	4.999994416721165e-06	4.99999999875000e-06
10^{11}	1.581152901053429e-06	1.581138830080237e-06
10^{12}	5.000038072466850e-07	4.9999999998749e-07
10^{13}	1.578591763973236e-07	1.581138830084150e-07
10^{14}	5.029141902923584e-08	4.9999999999987e-08
10^{15}	1.862645149230957e-08	1.581138830084189e-08
10^{16}	0	5.00000000000000e-09

Tabella 2: Valori di $\sqrt{n+1} - \sqrt{n}$ e di $\frac{1}{\sqrt{n+1} + \sqrt{n}}$ ottenuti con MATLAB per $n = 10, 10^2, \dots, 10^{16}$.

I risultati attesi non dovrebbero scostarsi di molto dai seguenti valori:

$$\begin{cases} \frac{\sqrt{10}}{2} \times 10^{-k} \simeq 1.5811388 \times 10^{-k} & \text{per } n = 10^{2k-1} \text{ e } k = 1, 2, \dots, 8 \\ 5 \times 10^{-k-1} & \text{per } n = 10^{2k} \text{ e } k = 1, 2, \dots, 8 \end{cases}$$

ottenuti con la seconda formula approssimata da $\frac{1}{2\sqrt{n}}$, per cui confrontando i valori in tabella proprio la seconda formula è quella che fornisce i valori più corretti, non essendo soggetta al fenomeno della cancellazione. In particolare, per $n = 10^{16}$ la prima formula ritorna il valore zero poichè, lavorando con precisione macchina $\epsilon \simeq 2.22 \times 10^{-16}$, risulta $n+1 = n$ dato che l'aggiunta di 1 non influisce sulle cifre significative di n : la cancellazione è in questo caso totale.

Per l'analisi dell'errore algoritmico si trascuri l'errore sul dato iniziale n supponendo (com'è il caso) che sia esattamente rappresentabile secondo lo standard IEEE doppia precisione. Indicati con $\varepsilon^{(i)}$ e $\eta^{(i)}$ gli errori locali generati dalle i -esime operazioni rispettivamente nel primo e nel secondo algoritmo si ha:

$$\begin{cases} \varepsilon_{alg1} = \varepsilon^{(1)} + \frac{\sqrt{n+1}}{\sqrt{n+1}-\sqrt{n}}\varepsilon^{(2)} + \frac{1}{2}\frac{\sqrt{n+1}}{\sqrt{n+1}-\sqrt{n}}\varepsilon^{(3)} - \frac{\sqrt{n}}{\sqrt{n+1}-\sqrt{n}}\varepsilon^{(4)} \\ \varepsilon_{alg2} = \eta^{(1)} - \eta^{(2)} - \frac{\sqrt{n+1}}{\sqrt{n+1}+\sqrt{n}}\eta^{(3)} - \frac{1}{2}\frac{\sqrt{n+1}}{\sqrt{n+1}+\sqrt{n}}\eta^{(4)} - \frac{\sqrt{n}}{\sqrt{n+1}+\sqrt{n}}\eta^{(5)} \end{cases}$$

Ipotizzando poi $|\varepsilon^{(i)}| \leq \epsilon$ e $|\eta^{(i)}| \leq \epsilon$ per ogni i si ottengono le maggio-

razioni

$$\begin{cases} |\varepsilon_{alg1}| \leq \epsilon \left(1 + \frac{\frac{3}{2}\sqrt{n+1}+\sqrt{n}}{|\sqrt{n+1}-\sqrt{n}|} \right) \\ |\varepsilon_{alg2}| \leq \epsilon \left(2 + \frac{\frac{3}{2}\sqrt{n+1}+\sqrt{n}}{\sqrt{n+1}+\sqrt{n}} \right) \end{cases}$$

per cui il primo algoritmo risulta instabile per $\sqrt{n+1} \simeq \sqrt{n}$ ovvero per $n \rightarrow \infty$.

4. Si scrivano e si eseguano tre *function* MATLAB che implementino i metodi di Newton, delle secanti e di punto fisso $x_{k+1} = g(x_k) = e^{-\frac{1}{2}x_k}$ per approssimare lo zero z della funzione

$$f(x) = e^{-\frac{1}{2}x} - x.$$

In particolare si segua lo standard

```
[x,K,e]=newton(x0,TOL)
[x,K,e]=secanti(x0,x1,TOL)
[x,K,e]=puntofisso(x0,TOL)
```

dove $x_0 = x_0$ è il valore iniziale scelto (per il metodo delle secanti serve un secondo valore iniziale $x_1 = x_1$), TOL una certa precisione richiesta, x il K -vettore delle iterate x_k , $k = 1, \dots, K$, K il numero di iterazioni necessario per raggiungere la precisione TOL ed e il K -vettore dell'errore e_k , $k = 1, \dots, K$, commesso alla k -esima iterazione secondo il criterio d'arresto scelto. Si stimi infine la costante nella definizione di ordine di convergenza per ognuno dei metodi e si dia una giustificazione analitica del valore ottenuto almeno per il metodo di Newton e per quello di punto fisso.

sol.4 Applicando il metodo di Newton ad f si ottiene lo schema iterativo

$$x_{k+1} = x_k - \frac{f(x_k)}{f'(x_k)} = x_k - \frac{e^{-\frac{1}{2}x_k} - x_k}{-\frac{1}{2}e^{-\frac{1}{2}x_k} - 1} = \frac{e^{-\frac{1}{2}x_k}(x_k + 2)}{e^{-\frac{1}{2}x_k} + 2} = h(x_k).$$

Per il metodo delle secanti si ottiene invece

$$\begin{aligned} x_{k+1} &= x_k - \frac{f(x_k)(x_k - x_{k-1})}{f(x_k) - f(x_{k-1})} = \\ &= \frac{f(x_k)x_{k-1} - f(x_{k-1})x_k}{f(x_k) - f(x_{k-1})} = \\ &= \frac{e^{-\frac{1}{2}x_k}x_{k-1} - e^{-\frac{1}{2}x_{k-1}}x_k}{e^{-\frac{1}{2}x_k} - e^{-\frac{1}{2}x_{k-1}} - x_k + x_{k-1}} = s(x_k, x_{k-1}). \end{aligned}$$

Per localizzare la radice z basta trovare due valori u e v tali che $f(u)f(v) < 0$. Ad esempio per $u = 0$ e $v = 1$ si ottiene $f(u) = 1 > 0$ e $f(v) = \frac{1}{\sqrt{e}} - 1 < 0$. Dunque $z \in (0, 1)$.

Si omettono possibili implementazioni MATLAB dei metodi di Newton, delle secanti e di punto fisso in quanto potrebbero essere argomento d'esame. I risultati corrispondenti agli input $x_0 = 1$ per h e g , $x_0 = 0$ e $x_1 = 1$ per s e $TOL = 10^{-10}$ sono riportati in figura 2, 3 e 4 e in tabella 3.

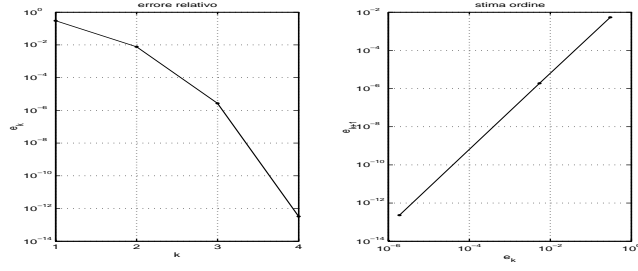


Figura 2: Errore relativo e stima dell'ordine di convergenza per il metodo di Newton h con $x_0 = 1$ ($TOL = 10^{-10}$).

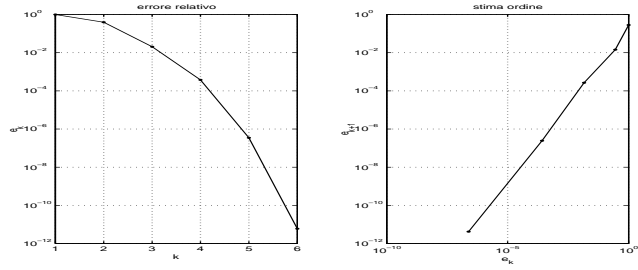


Figura 3: Errore relativo e stima dell'ordine di convergenza per il metodo delle secanti s con $x_0 = 0$ e $x_1 = 1$ ($TOL = 10^{-10}$).

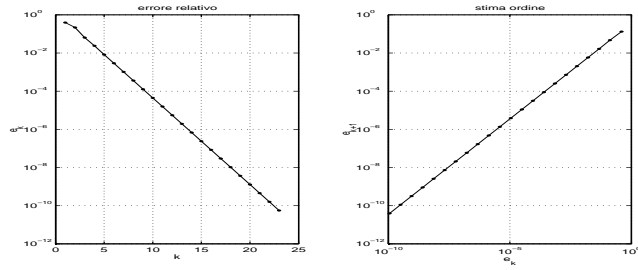


Figura 4: Errore relativo e stima dell'ordine di convergenza per il metodo di punto fisso g con $x_0 = 1$ ($TOL = 10^{-10}$).

Le costanti ottenute corrispondono ai valori $\frac{1}{2}|h''(z)|$ e $|g'(z)|$ per il metodo di Newton e per quello di punto fisso rispettivamente.

k	x_k	e_k	c
Metodo di Newton h			
1	6.980896128566958e-001	3.019103871433042e-001	5.897897076226033e-002
2	7.034655390741963e-001	5.375926217500487e-003	6.516906542800793e-002
3	7.034674224981610e-001	1.883423964623771e-006	6.503911752562572e-002
4	7.034674224983917e-001	2.307123400281200e-013	
Metodo delle secanti s			
1	1.000000000000000e+000	1.000000000000000e+000	
2	7.176332991967920e-001	2.823667008032080e-001	5.110586106353287e-002
3	7.032027058165751e-001	1.443059338021691e-002	6.502541087314707e-002
4	7.034676660965651e-001	2.649602799900119e-004	6.370911686795684e-002
5	7.034674225025867e-001	2.435939784328767e-007	6.499363691032238e-002
6	7.034674224983918e-001	4.194866676243692e-012	
Metodo di punto fisso g			
1	6.065306597126334e-001	3.934693402873666e-001	3.351531536505126e-001
2	7.384031499747310e-001	1.318724902620976e-001	3.572928626200457e-001
3	6.912860504281521e-001	4.711709954657894e-002	3.497466108998888e-001
4	7.077650963100007e-001	1.647904588184868e-002	3.524286323991108e-001
5	7.019574087066186e-001	5.807687603382128e-003	3.514887916393156e-001
6	7.039987458045500e-001	2.041337097931417e-003	3.518197966588029e-001
7	7.032805630018437e-001	7.181828027063020e-004	3.517034245594456e-001
8	7.035331503530152e-001	2.525873511715071e-004	3.517443631645359e-001
9	7.034443041760340e-001	8.884617698123876e-005	3.517299644960402e-001
10	7.034755540387092e-001	3.124986267522001e-005	3.517350290961379e-001
11	7.034645623673519e-001	1.099167135731882e-005	3.517332477174091e-001
12	7.034684285036162e-001	3.866136264352171e-006	3.517338742748792e-001
13	7.034670686525295e-001	1.359851086735198e-006	3.517336539136596e-001
14	7.034675469579210e-001	4.783053915158320e-007	3.517337314490503e-001
15	7.034673787217809e-001	1.682361401300625e-007	3.517337037093458e-001
16	7.034674378961016e-001	5.917432066571138e-008	3.517337127526007e-001
17	7.034674170824981e-001	2.081360350736361e-008	3.517337075668053e-001
18	7.034674244033440e-001	7.320845929470465e-009	3.517337147206357e-001
19	7.034674218283556e-001	2.574988333670092e-009	3.517337429090381e-001
20	7.034674227340659e-001	9.057102845488885e-010	3.517337093445346e-001
21	7.034674224154971e-001	3.185688379758744e-010	3.517336124148040e-001
22	7.034674225275485e-001	1.120513681840407e-010	3.517337795969162e-001
23	7.034674224881362e-001	3.941225124037828e-011	

Tabella 3: Approssimazione di z , errore assoluto e stima dell'ordine di convergenza per $a = \frac{1}{2}$ e $x_0 = 1$ ($TOL = 10^{-10}$).

5. Si scriva e si esegua una *function* MATLAB che associ al parametro m , $m \in [0, +\infty)$, la più piccola radice positiva dell'equazione

$$\sin x = mx$$

utilizzando la *function* predefinita *fzero*. Si studi attentamente la scelta dei valori iniziali affinché *fzero* converga alla radice positiva più piccola per $m = 0, 0.01, 0.02, \dots, 0.98, 0.99, 1, 2, 3, \dots, 10$, si rappresenti graficamente tale funzione e si commentino i risultati ottenuti.

sol.5 Una possibile implementazione è la seguente:

```
function z=mfzero(m)

for i=1:length(m)
    if m(i)<.8 x0=pi;
    else x0=m(i);
    end
    z(i)=fzero('zfun',x0,optimset('Display','final','TolX',eps),m(i));
end
plot(m,z);
```

dove *zfun* è

```
function F=zfun(x,m)
```

```
F=sin(x)-m*x;
```

Assegnato il vettore

```
>> m=[.0:.01:1,2:10]';
```

si ottiene il grafico delle radici z in funzione di m riportato in figura 5. L'istruzione di selezione del controllo *if* in *mfzero* consente di far convergere *fzero* sempre verso lo zero di *zfun* positivo più piccolo e non verso il suo simmetrico negativo, situazione che si può presentare quando $m \simeq 1$. La ragione per cui questo accade risiede nell'algoritmo su cui si basa la *fzero* di MATLAB, ovvero una combinazione dei metodi di bisezione, delle secanti e dell'interpolazione quadratica inversa. In particolare la ricerca dello zero parte con la valutazione di *zfun* in x_0 per poi cercare alternativamente a sinistra e a destra di x_0 un valore x_* in cui *zfun* cambi segno. Si può osservare tramite l'opzione *iter* di *Display* in *options* che tale ricerca procede su intervalli sempre più ampi (passi da 2 a 24, esempio con $x_0 = \pi$):

```
>> z=mfzero(.99)'
```

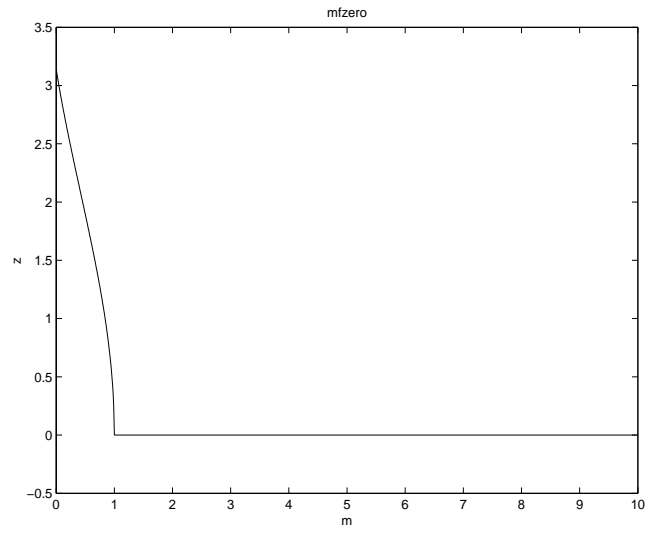


Figura 5: $mfzero$ per $m = 0, 0.01, 0.02, \dots, 0.98, 0.99, 1, 2, 3, \dots, 10$.

Func-count	x	f(x)	Procedure
1	3.14159	-3.11018	initial
2	3.05273	-2.93347	search
3	3.23045	-3.28689	search
4	3.01593	-2.86044	search
5	3.26726	-3.35992	search
6	2.96388	-2.75746	search
7	3.31931	-3.4629	search
8	2.89027	-2.61267	search
9	3.39292	-3.60768	search
10	2.78616	-2.41031	search
11	3.49702	-3.81005	search
12	2.63894	-2.13079	search
13	3.64425	-4.08956	search
14	2.43073	-1.75394	search
15	3.85245	-4.46642	search
16	2.13628	-1.27059	search
17	4.1469	-4.94976	search
18	1.71987	-0.713762	search
19	4.56332	-5.50659	search
20	1.13097	-0.214837	search
21	5.15221	-6.00552	search
22	0.298148	-0.00141609	search
23	5.98504	-6.21894	search
24	-0.879646	0.100336	search

Looking for a zero in the interval [-0.87965, 5.985]

25	-0.77065	0.0663417	interpolation
26	-0.560185	0.0232402	interpolation
27	2.71243	-2.26919	bisection
28	-0.527008	0.0187883	interpolation
29	-0.388139	0.00579113	interpolation
30	1.16214	-0.232865	bisection
31	-0.350521	0.00362861	interpolation
32	-0.288367	0.00109631	interpolation
33	0.436888	-0.00939731	bisection
34	-0.212597	-0.000528109	interpolation
35	-0.237231	-0.000153402	interpolation
36	-0.246077	1.52156e-005	interpolation
37	-0.245279	-7.72576e-007	interpolation
38	-0.245318	-3.5623e-009	interpolation
39	-0.245318	6.91114e-015	interpolation
40	-0.245318	-2.77556e-017	interpolation
41	-0.245318	0	interpolation

Zero found in the interval: [-0.87965, 5.985].

z =

-2.453178088540259e-001

Quando x_* viene trovato (in questo esempio $x_* = x_{24} = -0.879646$), l'algoritmo procede per interpolazione quadratica inversa intervallato da un passo di bisezione ogni volta che *zfun* cambia segno (ad esempio ai passi 26 e 27). Se l'intervallo $[x_*, x_0]$ contiene entrambi gli zeri simmetrici rispetto all'origine di *zfun*, *fzero* converge a quello negativo. Per evitare questo bisogna scegliere il valore iniziale x_0 in modo che la fase di ricerca di x_* si concluda con un intervallo che contiene solo lo zero positivo. Ad esempio partendo con $x_0 = m = 0.99$ si ottiene lo zero positivo:

>> z=mfzero(.99)'

Func-count	x	f(x)	Procedure
1	0.99	-0.144074	initial
2	0.961999	-0.132042	search
3	1.018	-0.156761	search
4	0.9504	-0.127248	search
5	1.0296	-0.162211	search
6	0.933997	-0.120654	search
7	1.046	-0.170115	search
8	0.9108	-0.111698	search

9	1.0692	-0.181692	search
10	0.877994	-0.099755	search
11	1.10201	-0.19887	search
12	0.8316	-0.0842738	search
13	1.1484	-0.224807	search
14	0.765989	-0.0650789	search
15	1.21401	-0.264847	search
16	0.6732	-0.042977	search
17	1.3068	-0.328377	search
18	0.541977	-0.0207266	search
19	1.43802	-0.432444	search
20	0.3564	-0.00393327	search
21	1.6236	-0.608758	search
22	0.0939543	0.000801375	search

Looking for a zero in the interval [0.093954, 1.6236]

23	0.0959653	0.000812425	interpolation
24	0.859783	-0.0934841	bisection
25	0.102546	0.000845831	interpolation
26	0.481164	-0.0135411	bisection
27	0.124806	0.000924304	interpolation
28	0.302985	-0.00158459	bisection
29	0.190449	0.000755286	interpolation
30	0.246717	-2.81382e-005	bisection
31	0.244696	1.23509e-005	interpolation
32	0.245313	1.0525e-007	interpolation
33	0.245318	-6.01591e-012	interpolation
34	0.245318	1.94289e-016	interpolation
35	0.245318	0	interpolation

Zero found in the interval: [0.093954, 1.6236].

z =

2.453178088540252e-001

Si fa osservare che la scelta euristica di x_0 implementata in *mfzero* non è l'unica possibile per ottenere sempre la convergenza verso lo zero positivo.