HOMEWORK 1 Capobianco Salvatore 0124000974

Table of Contents

PUNTO 1

Definite la function Matlab che implementa la vostra funzione di riferimento e che chiamerete funrif:

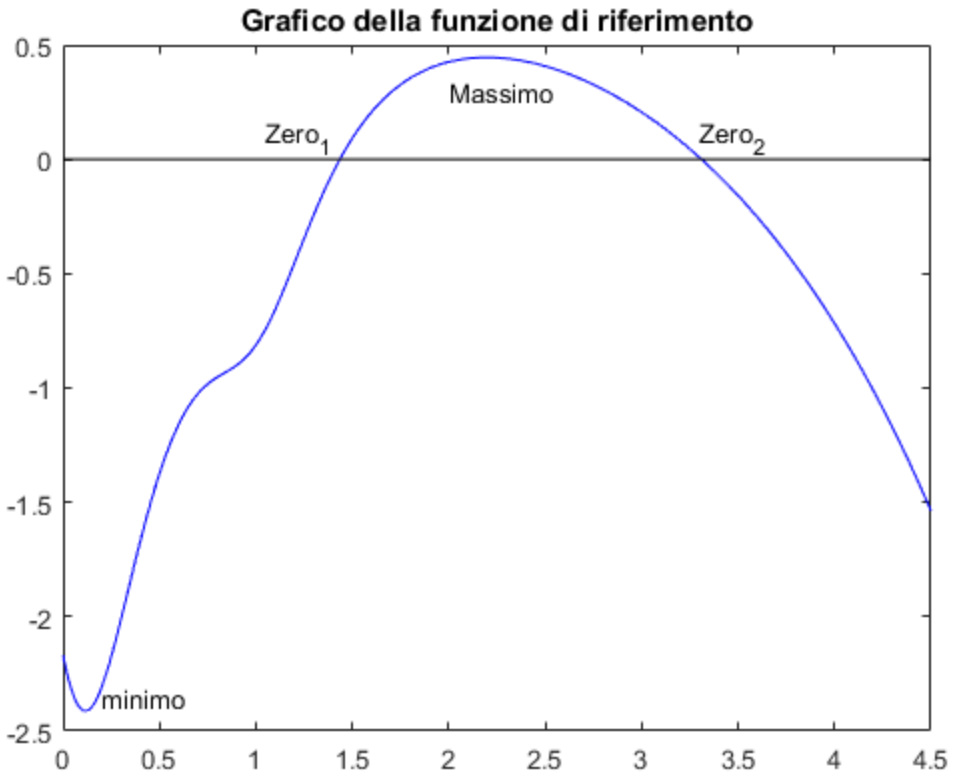
La funzione funrif è 2*(1-(2*exp(x))/(1+5.*x^2))-(1/(1+5*(x-1)^2))

```
funrif=@(x) 2*(1-(2.*exp(x))./(1+5.*x.^2))-(1./(1+5.*(x-1).^2));
```

PUNTO 2

Visualizzate il grafico della funzione di riferimento in [a,b], indicando sul grafico (comando text) i 2 zeri, il valore minimo e l'ascissa di minimo (detta punto di minimo), il valore massimo e l'ascissa di massimo (detta punto di massimo:

```
a=0; b=4.5;
x=linspace(a,b,1000);
plot(x,funrif(x),'b',[a b], [0 0], 'k')
title('Grafico della funzione di riferimento')
text(1.05,0.1,'Zero_1')
text(3.3,0.1,'Zero_2')
text(0.2,-2.35,'minimo')
text(2,0.3,'Massimo')
```



PUNTO 3

Usate la function Matlab `fzero` *per determinare i 2 zeri della vostra funzione di riferimento in [a,b], e considerate i valori calcolati da fzero come le soluzioni esatte:*

```
format long
os= [fzero(funrif, 1.5) fzero(funrif, 4)]
```

```
os =

   1.435614135758034   3.310124662006284
```

PUNTO 4

Determinate un'approssimazione dei 2 zeri, usando il metodo di bisezione (nostra function bisezione) per lo zero più piccolo e il metodo delle Secanti (nostra function Secanti per lo zero più grande, con un valore di delta_ass che garantisca, in entrambi i casi, che la parte intera e le prime 8 cifre frazionarie siano corrette:

I due metodi forniscono in output un'approssimazione della soluzione di f(x)=0, tanto più accurata quanto più è piccolo il `delta_ass`, un parametro di input. In particolare, l'ordine di grandezza di quest'ultimo indica il numero di cifre frazionarie uguali tra l'approssimazione e la soluzione esatta.

METODO DI BISEZIONE

La function `bisezione` prende in input, nell'ordine:

1. Handle della funzione;
2. Estremo sinistro e destro dell'intervallo contenente la soluzione, ovvero lo zero della funzione;
3. Maggiorazione dell'errore assoluto accettata.

```
appz1=bisezione(funrif, 0, 2, 1e-8)
```

```
appz1 =

   1.435614135116339
```

METODO DELLE SECANTI

La function `Secanti` prende in input, nell'ordine:

1. Handle della funzione;
2. Due punti di approssimazione iniziali della soluzione;
3. Maggiorazione dell'errore assoluto accettata;
4. Massimo numero di iterazioni consentito (poiché è un metodo locale, potrebbe non convergere)

```
appz2=Secanti(funrif, 3, 4, 1e-8, 20)
```

```
appz2 =

   3.310124663011075
```

PUNTO 5

Considerate sia l'errore assoluto tra l'approssimazione della bisezione e la corrispondente soluzione esatta, sia l'errore assoluto tra l'approssimazione delle Secanti e la corrispondente soluzione esatta; verificate che siano minori dell'accuratezza richiesta (delta_ass); calcolate anche il residuo per le due approssimazioni e commentate il loro valore:

```
Eass1=abs(os(1)-appz1)
Eass2=abs(os(2)-appz2)
if(Eass1 < 1e-8 && Eass2 < 1e-8)
    disp(sprintf('Verifica della maggiorazione dell errore assoluto
 positiva'));
else
    disp(sprintf('Verifica della maggiorazione dell errore assoluto
 negativa'));
end
```

```
Eass1 =

     6.416953635124401e-10
```

```
Eass2 =

    1.004791361225443e-09
```

```
Verifica della maggiorazione dell errore assoluto positiva
```

residuo su z1 = abs(funrif(os(1))-funrif(appz1))

```
res1=abs(funrif(appz1))
```

```
res1 =

    9.935474665212496e-10
```

residuo su z2 = abs(funrif(os(2))-funrif(appz2))

```
res2=abs(funrif(appz2))
```

```
res2 =

    7.720979203207357e-10
```

I residui corrispondono al valore della funzione, calcolato nelle approssimazioni degli zeri, trovate con i metodi di `bisezione` e delle `Secanti`; essi sono esattamente uguali allo zero, fino alla decima cifra dopo la virgola. E' evidente, dunque, che le approssimazioni degli zeri `appz1` e `appz2`, sono delle ottime approssimazioni.

PUNTO 6

Usate la function Matlab `fminbnd` per determinare il punto di minimo e il punto di massimo della vostra funzione di riferimento in [a,b], con due chiamate del tipo: `fminbnd(fun,sinistro,destro,optimset('TolX',1e-10))`, e considerate i valori calcolati da fminbnd come le soluzioni esatte (cioè come valori esatti del punto di minimo e del punto di massimo:

```
mi=fminbnd(funrif,a,b,optimset('TolX',1e-10))
```

```
mi =

   0.116043178079607
```

Per determinare il punto di massimo attraverso la function `fminbnd`, è necessario prima conoscere la funzione inversa della funzione di riferimento...

```
y_inv=@(x) -funrif(x);
```

...il punto di minimo di questa corrisponde a quello di massimo per la funzione di riferimento.

```
ma=fminbnd(y_inv,a,b,optimset('TolX',1e-10))
```

```
ma =

   2.196264988321584
```

PUNTO 7

Verificare che il punto di minimo e il punto di massimo sono zeri della derivata della vostra funzione di riferimento (consiglio: usare funtool per determinare l'espressione della derivata della funzione di riferimento), plottando il grafico della derivata e usando la fzero:

Per calcolare la derivata usiamo lo strumento Matlab `funtool`

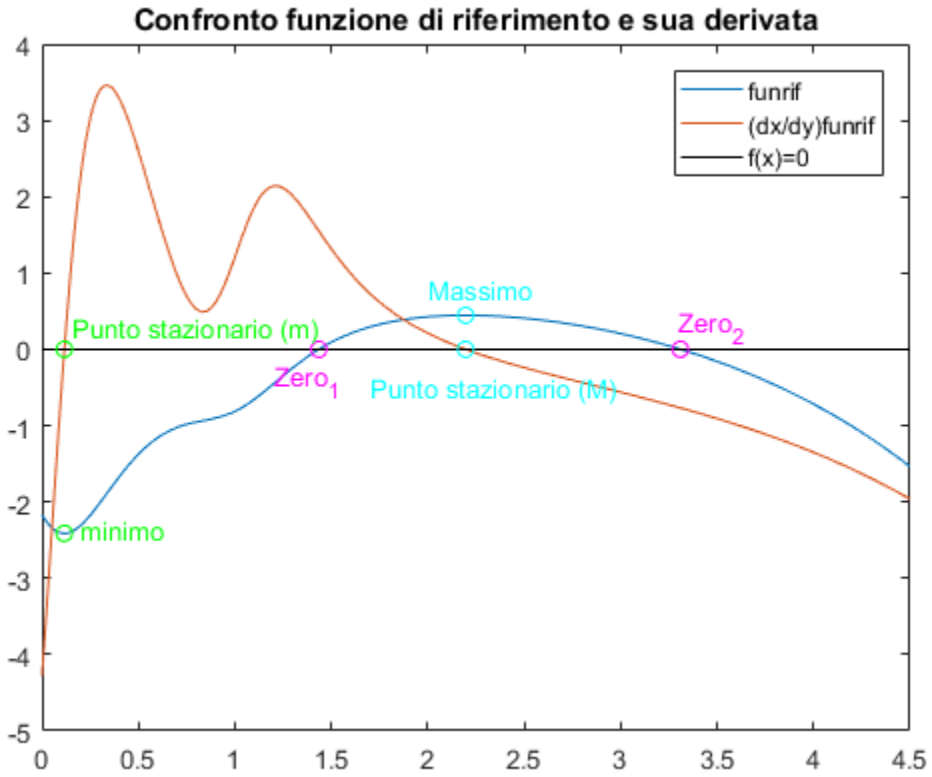
```
funrifp=@(x) (10.*x - 10)./(5.*(x - 1).^2 + 1).^2 - (4.*exp(x))./...
    (5.*x.^2 + 1) + (40.*x.*exp(x))./(5.*x.^2 + 1).^2;

osp=[fzero(funrifp, mi) fzero(funrifp, ma)];
if(abs(mi-osp(1))<1e-6)
    disp(sprintf('Il punto di minimo di f è punto stazionario della
 sua derivata:\n'));
else
    disp(sprintf('ERRORE: il punto di minimo di f non è punto
 stazionario della sua derivata:\n'));
end;
disp(sprintf('      argmin(f(x))   fzero((dx/dy)funrif(x))'));
disp([mi osp(1)]);
if(abs(ma-osp(2))<1e-6)
    disp(sprintf('Il punto di Massimo di f è punto stazionario della
 sua derivata:\n'));
else
    disp(sprintf('ERRORE: il punto di Massimo di f non è punto
 stazionario della sua derivata:\n'));
end
disp(sprintf('      argmax(f(x))   fzero((dx/dy)funrif(x))'));
disp([ma osp(2)]);
plot(x,funrif(x),x,funrifp(x),[a b], [0 0], 'k',os(1),
 funrif(os(1)),'mo',...
    os(2),
 funrif(os(2)),'mo',mi,funrif(mi),'go',ma,funrif(ma),'co',...
    osp(1),funrifp(osp(1)),'go',osp(2),funrifp(osp(2)),'co');
title('Confronto funzione di riferimento e sua derivata')
legend('funrif','(dx/dy)funrif','f(x)=0')
text(1.2,-0.4,'Zero_1','color','magenta')
text(3.3,0.3,'Zero_2','color','magenta')
text(0.2,-2.35,'minimo','color','green')
text(2,0.8,'Massimo','color','cyan')
text(mi+0.04, funrifp(osp(1))+0.3, 'Punto stazionario
 (m)', 'color', 'green');
text(ma-0.5, funrifp(osp(2))-0.5,'Punto stazionario
 (M)', 'color', 'cyan');
```

Il punto di minimo di f è punto stazionario della sua derivata:

```
   argmin(f(x))   fzero((dx/dy)funrif(x))
0.116043178079607   0.116043178143331
```

Il punto di Massimo di f è punto stazionario della sua derivata:

```
   argmax(f(x))   fzero((dx/dy)funrif(x))
2.196264988321584   2.196265003536146
```



PUNTO 8

Determinate un'approssimazione del punto di minimo e del punto di massimo, usando rispettivamente il metodo di Fibonacci search (nostra function `fminfibo`) per il minimo e il Golden search (nostra function `fmingolden`) per il massimo, con un valore di `delta_ass` che garantisca che la parte intera e le prime 5 cifre frazionarie siano corrette:

La funzione di riferimento è **unimodale** in [a,b], ovvero è una funzione con un unico punto di minimo (e un unico punto di massimo). Formalmente: **una funzione è unimodale in [a,b], se è monotona crescente fino a un certo punto (la moda) e poi monotona decrescente (o viceversa)** (fonte: Wikipedia). Ovvero esiste un unico punto p contenuto in [a,b] tale che f: è strettamente decrescente in [a,p], è strettamente crescente in [p,b], e p si dirà punto di minimo (viceversa per il punto di massimo). E' dunque possibile applicare i metodi della ricerca di Fibonacci `fminfibo` e della ricerca aurea `fmingolden`.

```
appmi=fminfibo(funrif, a, b, 1e-5)
```

```
appmi =
```

```
   0.116050105251234
```

Analogamente al **PUNTO 6** in cui si chiedeva di determinare il punto di massimo della funrif attraverso l'utilizzo di `fminbnd`, anche per ottenerne una sua approssimazione, tramite la function `fmingolden`, sfrutteremo la simmetria rispetto all'asse delle ascisse, della funzione inversa di funrif:

```
appma=fmingolden(y_inv, a, b, 1e-5)
disp(sprintf('       min/Max          appmin/appMax'))
disp([mi appmi; ma appma])
```

```
appma =

   2.196266056115877

       min/Max           appmin/appMax
   0.116043178079607   0.116050105251234
   2.196264988321584   2.196266056115877
```

PUNTO 9

Considerate l'errore assoluto tra l'approssimazione di Fibonacci search e la corrispondente soluzione esatta e poi l'errore assoluto tra l'approssimazione di Golden search e la corrispondente soluzione esatta, e verificate che sia in entrambi i casi minore dell'accuratezza richiesta (`delta_ass`):

```
Eami=abs(mi-appmi)
Eama=abs(ma-appma)
if(Eami < 1e-5 && Eama < 1e-5)
    disp(sprintf('Verifica della maggiorazione degli errori assoluti
 positiva'));
else
    disp(sprintf('Verifica della maggiorazione degli errori assoluti
 negativa'));
end
```

```
Eami =

     6.927171626841533e-06


Eama =

     1.067794292985980e-06

Verifica della maggiorazione degli errori assoluti positiva
```

PUNTO 10

Modificare opportunamente sia la function `fminfibo` sia la function `fmingolden` in modo da verificare che l'ampiezza dell'ultimo intervallo calcolato dalle due function sia quella teoricamente pre-

vista, cioè rispettivamente <= (b-a)/Fibonacci(N), e <= (b-a)(phi-1)^N, dove N è il numero di iterazioni, Fibonacci(N) è l'N-simo numero di Fibonacci e phi è la sezione aurea:

Modifico opportunamente la function `fminfibo` al fine di ottenere l'ampiezza dell' intervallo all'ultima iterazione:

```
function ampf=fminfibo1(f,a,b,delta)

...

end

ampf=hk

if(ampf<=h/fibonacci(n))

disp(sprintf('Verifica positiva, l ampiezza dell ultimo intervallo è

quella teoricamente prevista'))

else

disp(sprintf('Verifica negativa, l ampiezza dell ultimo intervallo

NON è quella teoricamente prevista'))

end

ampf=fminfibo1(funrif, a, b, 1e-5);
disp(sprintf('Ampiezza prevista:'))
(b-a)/fibonacci(fix(((log(b-a)-log(1e-5))/log(1.61803398874989))))


ampf =

     2.831871772847328e-05

Verifica negativa, l ampiezza dell ultimo intervallo NON è quella
 teoricamente prevista
Ampiezza prevista:

ans =

     1.415935886423063e-05
```

Modifico opportunamente il caso base e l'autoattivazione della function ricorsiva `fmingolden`:

```
function ampg=fmingolden1(f,a,b,delta)

if abs(b-a)<delta

ris=(a+b)/2;

ampg=abs(b-a)

...
```

```
ampg=fmingolden1(f,a,b,delta)

end

ampg=fmingolden1(y_inv, a, b, 1e-5);
phi = 1.61803398874989;
if(ampg<=(b-a)*((phi-1)^(fix(((log(b-a)-log(1e-5)))/log(phi)))))
    disp(sprintf('Verifica positiva, l ampiezza dell ultimo intervallo
 è quella teoricamente prevista'))
    else
    disp(sprintf('Verifica negativa, l ampiezza dell ultimo intervallo
 NON è quella teoricamente prevista'))
end
disp(sprintf('Ampiezza prevista:'))
(b-a)*((phi-1)^(fix(((log(b-a)-log(1e-5)))/log(phi))))


ampg =

     6.332257787899920e-06

Verifica positiva, l ampiezza dell ultimo intervallo è quella
 teoricamente prevista
Ampiezza prevista:

ans =

     1.024580832591625e-05
```

Published with MATLAB® R2016a