Παράδειγμα #2
ΡΙΖΕΣ ΕΞΙΣΩΣΕΩΝ
ΕΠΙΜΕΛΕΙΑ: Ν. Βασιλειάδης

Άσκηση 1

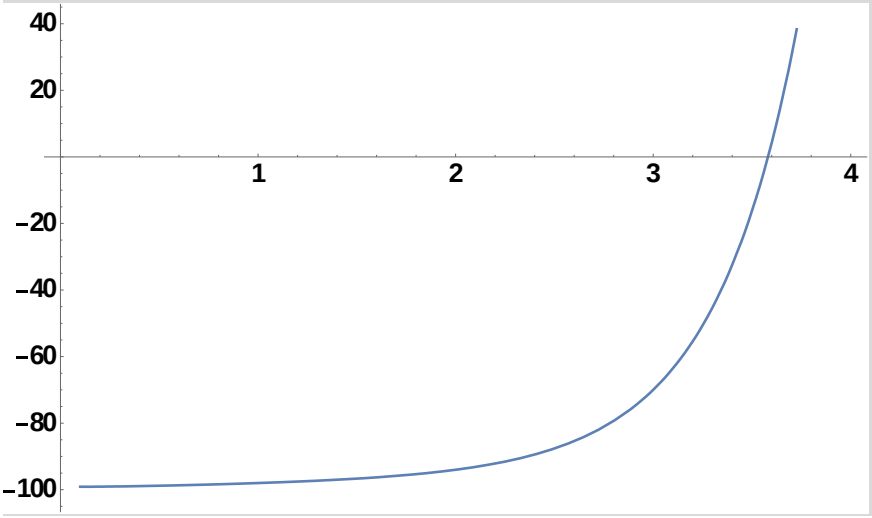
Να βρεθεί μία πραγματική ρίζα της εξίσωσης $x + x^x = 100$ με τις μεθόδους α) της διχοτόμησης β) της γραμμικής παρεμβολής γ) των διαδοχικών επαναλήψεων δ) Newton και ε) να εξετασθεί αν ο αλγόριθμος $x^{n+1} = (1-\omega)x^n + \omega F(x^n)$ βελτιώνει το ρυθμό σύγκλισης. Να βρεθεί η τιμή του ω που οδηγεί στον ελάχιστο αριθμό επαναλήψεων. Τέλος να γίνει μια σύνοψη των αποτελεσμάτων των διαφορετικών μεθόδων καθώς και το γράφημα σφάλματος (άξονας y) - επαναλήψεων (άξονας x). Σε όλες τις μεθόδους να χρησιμοποιηθεί σχετικό σφάλμα 0.01%.

Απάντηση:

Γίνεται η γραφική παράσταση της συνάρτησης για να βρεθεί προσεγγιστικά το διάστημα στο οποίο βρίσκεται η ρίζα.

Με τη χρήση της Mathematica παίρνουμε το παρακάτω διάγραμμα:

```
Plot[x+x^x-100,{x,0.1,4},AxesOrigin->{0,0},LabelStyle->
{14,GrayLevel[0],Bold}]
```



Από την γραφική παράσταση της συνάρτησης φαίνεται ότι η ρίζα βρίσκεται κοντά στο x=3.5. Με τη βοήθεια της συνάρτησης FindRoot (η οποία βρίσκει την αριθμητική τιμή της ρίζας χρησιμοποιώντας τον αλγόριθμο Newton) και αρχική εκτίμηση x=3.5 βρίσκεται η ρίζα της εξίσωσης με ακρίβεια 6 σημαντικών ψηφίων:

```
FindRoot[x + x^x-100,{x,3.5}]
{x -> 3.58128}
```

1α. Μέθοδος διχοτόμησης

Το πρόγραμμα σε Fortran είναι το παρακάτω:

```
program Bisection
implicit none
real*8:: xl,xr,rel,xold,xm,prod,err
integer:: i,maxi

print*, ' Enter xl:'
read*, xl
print*, ' Enter xr:'
read*, xr
print*, ' Enter desired relative error % :'
read*, rel
print*, ' Enter the maximum number of iterations:'
read*, maxi

open(100,file="Bisection_Results.dat")

err=100.
i=0
xold=0.
do while ((i<=maxi).and.(err>=rel))
    xm=(xl+xr)/2
    prod=f(xl)*f(xm)
    err = abs((xm - xold)/xm) * 100.

    write(*,"(I5,7ES18.8)")   i,xl,xm,xr,f(xl),f(xm),f(xr),err
    write(100,"(I5,7ES18.8)") i,xl,xm,xr,f(xl),f(xm),f(xr),err

     if (prod<0) then
        xr=xm
    else
        xl=xm
    end if
    xold=xm
    i=i+1

end do

if (err<rel) then
    print*, 'The root found is:',xm
    print*, 'The percentage relative error =',err
    print*,'The number of bisections is:',i-1
elseif (err>=rel) then
    print*, 'The root is not reached within the error limit
after the prescribed number of iterations.'
```

```
    print*, 'The approximate root =',xm
    print*, 'The percentage relative error =',err
end if

contains

real*8 function f(x)
real*8,intent(in):: x
    f=x+x**x-100
end function f

end Bisection
```

Η μέθοδος της διχοτόμησης με αρχικό διάστημα [3,4] και σχετικό σφάλμα 0.01% δίνει τα παρακάτω αποτελέσματα:

n	$\mathbf{x_l}$	$\mathbf{x_m}$	$\mathbf{x_r}$	$\mathbf{f(x_l)}$	$\mathbf{f(x_m)}$	$\mathbf{f(x_r)}$	**err (%)**
0	3.000000	3.500000	4.000000	-70.00000	-16.28822	160.00000	-
1	3.500000	3.750000	4.000000	-16.28822	45.85758	160.00000	6.66667
2	3.500000	3.625000	3.750000	-16.28822	10.15981	45.85758	3.44828
3	3.500000	3.562500	3.625000	-16.28822	-4.04712	10.15981	1.75439
4	3.562500	3.593750	3.625000	-4.04712	2.79111	10.15981	0.86957
5	3.562500	3.578125	3.593750	-4.04712	-0.69175	2.79111	0.43668
6	3.578125	3.585938	3.593750	-0.69175	1.03343	2.79111	0.21786
7	3.578125	3.582031	3.585938	-0.69175	0.16682	1.03343	0.10905
8	3.578125	3.580078	3.582031	-0.69175	-0.26347	0.16682	0.05456
9	3.580078	3.581055	3.582031	-0.26347	-0.04858	0.16682	0.02727
10	3.581055	3.581543	3.582031	-0.04858	0.05906	0.16682	0.01363
11	3.581055	3.581299	3.581543	-0.04858	0.00522	0.05906	0.00682

Επομένως ο αλγόριθμος συγκλίνει στη ρίζα: 3.58130 με σχετικό σφάλμα 0.00682% μετά από 11 διχοτομήσεις.

1β. Μέθοδος γραμμικής παρεμβολής

Το πρόγραμμα είναι το ίδιο με το προηγούμενο με μόνη αλλαγή τον υπολογισμό του xm:

```
xm=xl-f(xl)*(xr-xl)/(f(xr)-f(xl))
```

Για αρχικό διάστημα [3,4] και σχετικό σφάλμα 0.01% η μέθοδος της γραμμικής παρεμβολής δίνει τα παρακάτω αποτελέσματα:

n	x_l	x_m	x_r	$f(x_l)$	$f(x_m)$	$f(x_r)$	err (%)
0	3.000000	3.304348	4.000000	-70.00000	-44.78698	160.00000	-
1	3.304348	3.456487	4.000000	-44.78698	-23.80158	160.00000	4.40156
2	3.456487	3.526870	4.000000	-23.80158	-11.24727	160.00000	1.99562
3	3.526870	3.557944	4.000000	-11.24727	-5.00210	160.00000	0.87338
4	3.557944	3.571346	4.000000	-5.00210	-2.16297	160.00000	0.37524
5	3.571346	3.577063	4.000000	-2.16297	-0.92379	160.00000	0.15984
6	3.577063	3.579491	4.000000	-0.92379	-0.39244	160.00000	0.06783
7	3.579491	3.580520	4.000000	-0.39244	-0.16634	160.00000	0.02874
8	3.580520	3.580955	4.000000	-0.16634	-0.07044	160.00000	0.01217
9	3.580955	3.581140	4.000000	-0.07044	-0.02981	160.00000	0.00515

Άρα τελικά ο αλγόριθμος συγκλίνει στη ρίζα: 3.58114 με σχετικό σφάλμα 0.00515% μετά από 9 γραμμικές παρεμβολές.

1γ. Μέθοδος απλών επαναλήψεων

Για την μέθοδο των απλών επαναλήψεων η εξίσωση $x + x^x = 100$ πρέπει να γραφτεί σε μορφή $x = F(x)$. Άρα έχουμε διαδοχικά:

$$x + x^x = 100 \Leftrightarrow x^x = 100 - x \Leftrightarrow \ln x^x = \ln(100 - x) \Leftrightarrow x \ln x = \ln(100 - x) \Leftrightarrow x = \frac{\ln(100 - x)}{\ln x} = F(x)$$

Άρα θα ο επαναληπτικός τύπος δίνεται ως: $x^{(n+1)} = \frac{\ln(100 - x^{(n)})}{\ln(x^{(n)})} = F\left(x^{(n)}\right)$

Το πρόγραμμα σε Fortran είναι το παρακάτω:

```
program SimpleIterations
implicit none
real*8:: x0,xold,xnew,rel,err
integer:: maxi,i

print*, 'Enter initial guess x0:'
read*, x0
print*, ' Enter desired relative error % :'
read*, rel
print*, ' Enter the maximum number of iterations:'
read*, maxi

open (100,file="SimpleIterations_Results.dat")
err = 100.
i = 1
xold=x0

write(*,"(I5,4ES18.8)")   0, x0, F(x0),DerF(x0),err
write(100,"(I5,4ES18.8)") 0, x0, F(x0),DerF(x0),err
```

```
do while ((i/=maxi).and.(err>=rel))

    xnew = F(xold)
    if (xnew/=0.) then
        err = abs((xnew - xold)/xnew) * 100.
    endif
    write(*,"(I5,4ES18.8)")   i, xnew, F(xnew),DerF(xnew),err
    write(100,"(I5,4ES18.8)") i, xnew, F(xnew),DerF(xnew),err
    xold = xnew
    i = i + 1

end do

print*,'-----------------------------------------------------'

if (err<rel) then
    print*, 'The root found is:',xold,' With % relative error:
',err
    print*,'The number of iteration performed before achieving
acceptable error limit is:',i - 1
else
    print*, 'The root is not reached within the error limit
after the prescribed number of iterations.'
    print*, 'The approximate root =',xold
    print*, 'The percentage relative error =',err

endif
print*,'-----------------------------------------------------'

contains

real*8 function F(x)
real*8,intent(in)::x
F=log(100-x)/log(x)
end function f

real*8 function DerF(x)
real*8,intent(in)::x
DerF=-(Log(100 - x)/(x*Log(x)**2)) - 1/((100 - x)*Log(x))
end function DerF

end
```

Το παραπάνω πρόγραμμα και με αρχική εκτίμηση $x_0 = 3.5$, σχετικό σφάλμα $= 0.01$ και μέγιστο πλήθος επαναλήψεων $= 100$ δίνει:

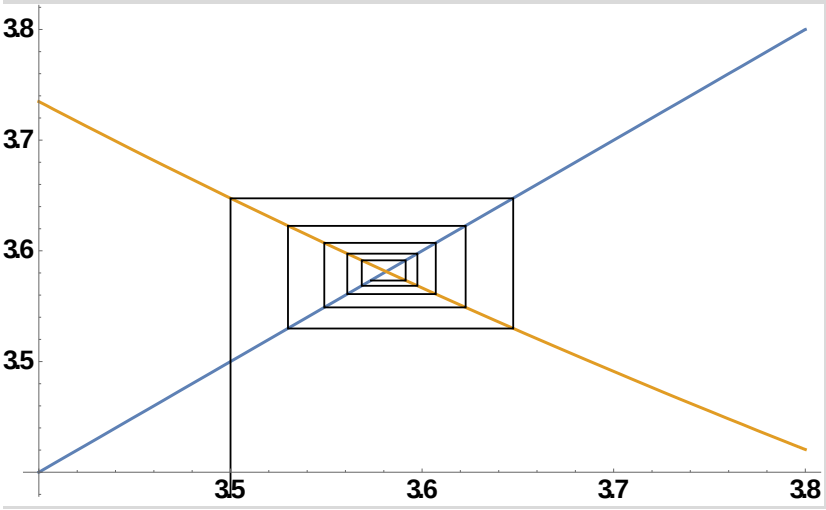
n	x^n	$F(x^n)$	$F'(x^n)$	err (%)
0	3.5000000	3.6475719	-0.8401638	-
1	3.6475719	3.5299805	-0.7558683	4.0457573
2	3.5299805	3.6226592	-0.8218718	3.3312202
3	3.6226592	3.5489757	-0.7691344	2.5583058
4	3.5489757	3.6071549	-0.8106058	2.0761898
5	3.6071549	3.5609658	-0.7775776	1.6128838
6	3.5609658	3.5974776	-0.8036197	1.2970947
7	3.5974776	3.5685165	-0.7829222	1.0149282
8	3.5685165	3.5914262	-0.7992690	0.8115743
9	3.5914262	3.5732645	-0.7862939	0.6379014
10	3.5732645	3.5876378	-0.7965523	0.5082688
11	3.5876378	3.5762473	-0.7884164	0.4006352
12	3.5762473	3.5852643	-0.7948530	0.3185039
13	3.5852643	3.5781202	-0.7897508	0.2515024
14	3.5781202	3.5837767	-0.7937889	0.1996626
15	3.5837767	3.5792957	-0.7905890	0.1578368
16	3.5792957	3.5828440	-0.7931223	0.1251928
17	3.5828440	3.5800333	-0.7911152	0.0990363
18	3.5800333	3.5822591	-0.7927044	0.0785100
19	3.5822591	3.5804961	-0.7914455	0.0621342
20	3.5804961	3.5818923	-0.7924423	0.0492392
21	3.5818923	3.5807864	-0.7916527	0.0389794
22	3.5807864	3.5816622	-0.7922780	0.0308831
23	3.5816622	3.5809686	-0.7917827	0.0244524
24	3.5809686	3.5815179	-0.7921750	0.0193708
25	3.5815179	3.5810828	-0.7918643	0.0153389
26	3.5810828	3.5814274	-0.7921104	0.0121502
27	3.5814274	3.5811545	-0.7919155	0.0096219

Επομένως ο αλγόριθμος των απλών επαναλήψεων με αρχική τιμή 3.5 συγκλίνει στη ρίζα 3.58143 με σχετικό σφάλμα 0.00962% μετά από 27 επαναλήψεις.

Με τον παρακάτω κώδικα σε Mathematica παίρνουμε γραφικά τον τρόπο σύγκλισης της μεθόδου των διαδοχικών επαναλήψεων:

```
g[x_] :=Log[100-x]/Log[x]
data=NestList[g,3.5,10];
data2=Flatten[Table[{data[[i]],data[[i]]},{i,Length[data]}]];
data2=Partition[data2,2,1];
data2[[1,2]]=0;p1=Graphics[Line[data2]];
```

```
Plot[{x,g[x]},{x,3.4,3.8}];
Show[%19,PlotLabel->None,LabelStyle->{12,GrayLevel[0],Bold}]
```



1δ. Μέθοδος Newton

Το πρόγραμμα για την μέθοδο Newton σε Fortran δίνεται παρακάτω:

```
program Newton
implicit none
real*8:: x0,xnew,rel,err,xold
integer:: maxi,i

print*, 'Enter initial guess x0:'
read*, x0
print*, ' Enter desired relative error % :'
read*, rel
print*, ' Enter the maximum number of iterations:'
read*, maxi

open (100,file="Newton_Results.dat")

err = 100.
i = 1
xold=x0

write(*,"(I5,5ES18.8)")
0,x0,f(x0),Derf(x0),Abs(Der2f(x0)/(2*Derf(x0))),err
write(100,"(I5,5ES18.8)")
0,x0,f(x0),Derf(x0),Abs(Der2f(x0)/(2*Derf(x0))),err
do while ((i/=maxi).and.(err>=rel).and.(Derf(xold)/=0.))

     xnew = xold-(f(xold)/Derf(xold))
```

```
    if (xnew/=0.) then
        err = abs((xnew - xold)/xnew) * 100.
    endif

    write(*,"(I5,5ES18.8)")
i,xnew,f(xnew),Derf(xnew),Abs(Der2f(xnew)/(2*Derf(xnew))),err
    write(100,"(I5,5ES18.8)")
i,xnew,f(xnew),Derf(xnew),Abs(Der2f(xnew)/(2*Derf(xnew))),err
    xold = xnew
    i = i + 1
end do

if ((err<rel).and.(Derf(xold)/=0.)) then
    print*, 'The root within the prescribed error limit
is:',xold
     print*, 'The percentage relative error =',err
    print*,'The number of iteration performed before achieving'
    print*,'acceptable error limit is:',i - 1
elseif((err>rel).and.(Derf(xold)/=0.)) then
    print*, 'The root is not reached within the error limit
after prescribed number of iteration.'
    print*, 'The approximate root =',xold
    print*, 'The percentage relative error =',err
elseif(Derf(xold)==0.) then
    print*, 'Newton''s method fails....Derivative equal to
zero.'
endif

contains

real*8 function f(x)
real*8,intent(in)::x
f=x**x+x-100
end function f

real*8 function Derf(x)
real*8,intent(in)::x
Derf=1 + x**x*(1 + Log(x))
end function Derf

real*8 function Der2f(x)
real*8,intent(in)::x
Der2f= x**x*(1 + Log(x))**2.+x**(x-1.)
end function Der2f

end
```

Το παραπάνω πρόγραμμα και με αρχική εκτίμηση $x_0 = 3.5$, σχετικό σφάλμα = 0.01 και μέγιστο πλήθος επαναλήψεων = 100 δίνει τα εξής αποτελέσματα:

n	$\mathbf{x^n}$	$\mathbf{f}\left(\mathbf{x^n}\right)$	$\mathbf{f'}\left(\mathbf{x^n}\right)$	$\left\|\frac{f''(x)}{2f'(x)}\right\|$	**err (%)**
0	3.50000000	-16.2882197	181.69812814	1.18324748	-
1	3.58964440	1.86333180	224.87268264	1.19483358	2.49730588
2	3.58135824	0.01832157	220.46565980	1.19377941	0.23136930
3	3.58127513	0.00000182	220.42192050	1.19376882	0.00232051

Άρα τελικά ο αλγόριθμος συγκλίνει στη ρίζα: 3.58127513 με σχετικό σφάλμα 0.00232051% μετά από 3 επαναλήψεις.

1ε. Μέθοδος απλών επαναλήψεων με παράμετρο χαλάρωσης

Ο επαναληπτικός τύπος της μεθόδου των απλών επαναλήψεων με χαλάρωση γράφεται ως:

$$x^{n+1} = G(\omega, x) = (1-\omega)x^n + \omega F\left(x^n\right)$$

Έστω x η αναλυτική λύση, το σφάλμα ορίζεται ως $\sigma^n = x^n - x$ και αντικαθιστώντας στον επαναληπτικό τύπο προκύπτει ότι

$$\sigma^{n+1} + x = (1-\omega)\sigma^n + (1-\omega)x + \omega F\left(\sigma^n + x\right)$$

Αναπτύσσοντας τον όρο $F\left(\sigma^n + x\right)$ με σειρά Taylor γύρω από το $F\left(x\right)$ και κρατώντας όρους πρώτης τάξης έχουμε

$$F\left(\sigma^n + x\right) = F\left(x\right) + \sigma^n F'\left(x\right)$$

Αντικαθιστώντας το ανάπτυγμα στην παραπάνω σχέση και κάνοντας χρήση της σχέσης

$$x = F\left(x\right) \Rightarrow x = F\left(x\right) + (1-\omega)x - (1-\omega)x \Rightarrow x = (1-\omega)x + \omega F\left(x\right)$$

προκύπτει ότι

$$\sigma^{n+1} = (1-\omega)\sigma^n + \omega\sigma^n F'\left(x\right) \Rightarrow \frac{\sigma^{n+1}}{\sigma^n} = (1-\omega) + \omega F'\left(x\right)$$

απαιτούμε το σφάλμα της $n+1$ επανάληψης να είναι όσο το δυνατόν πιο μικρό, δηλαδή το βέλτιστο ω βρίσκεται ως εξής

$$\frac{\sigma^{n+1}}{\sigma^n} = 0 \Rightarrow (1-\omega) + \omega F'\left(x\right) = 0 \Rightarrow \omega = \frac{1}{1-F'\left(x\right)}$$

Εναλλακτικά, για την εύρεση του βέλτιστου ω πρέπει να ισχύει:

$$\frac{dG(\omega, x)}{dx} = 0 \Rightarrow 1 - \omega + \omega F'\left(x\right) = 0 \Rightarrow \omega = \frac{1}{1-F'\left(x\right)}$$

Το ω υπολογίζεται για $x = x_0$ (Αν κρίνεται σκόπιμο υπολογίζεται σε κάθε επανάληψη).

Το αντίστοιχο πρόγραμμα σε Fortran είναι το ακόλουθο:

```
program Simple_Iterations_with_Relaxation
implicit none
real*8:: x0,xold,xnew,rel,err,w
integer:: maxi,i
print*, 'Enter initial guess x0:'
read*, x0
print*, ' Enter desired relative error % :'
read*, rel
print*, ' Enter the maximum number of iterations:'
read*, maxi
open (100,file="SimpleIterationsRelaxation_Results.dat")
w=1/(1-DerF(x0))
Write(*,"(A4,,ES18.8)") 'w = ',w
err = 100.
i = 1
xold=x0
write(*,"(I5,4ES18.8)")   0, x0, F(x0),DerF(x0),err
write(100,"(I5,4ES18.8)") 0, x0, F(x0),DerF(x0),err
do while ((i/=maxi).and.(err>=rel))
    xnew =(1-w)*xold+w*F(xold)
    if (xnew/=0.) then
        err = abs((xnew - xold)/xnew) * 100.
    endif
    write(*,"(I5,4ES18.8)")   i, xnew, F(xnew),DerF(xnew),err
    write(100,"(I5,4ES18.8)") i, xnew, F(xnew),DerF(xnew),err
    xold = xnew
    i = i + 1
end do
if (err<rel) then
    print*, 'The root found is:',xold,' With % relative error:
',err
    print*,'The number of iteration performed before achieving
acceptable error limit is:',i - 1
else
    print*, 'The root is not reached within the error limit
after the prescribed number of iterations.'
    print*, 'The approximate root =',xold
    print*, 'The percentage relative error =',err
endif
contains
real*8 function F(x)
real*8,intent(in)::x
F=log(100-x)/log(x)
end function f
real*8 function DerF(x)
real*8,intent(in)::x
DerF=-(Log(100-x)/(x*Log(x)**2))-1/((100-x)*Log(x))
end function DerF
end
```

Το παραπάνω πρόγραμμα και με αρχική εκτίμηση $x_0 = 3.5$, σχετικό σφάλμα = 0.01 και μέγιστο πλήθος επαναλήψεων = 100 δίνει:

n	$\mathbf{x^n}$	$\mathbf{F(x^n)}$	$\mathbf{F'(x^n)}$	**err (%)**
0	3.5000000000	3.6475719057	-0.8401637999	-
1	3.5801949836	3.5821309285	-0.7926128016	2.2399613408
2	3.5812470339	3.5812973732	-0.7920174904	0.0293766483
3	3.5812743898	3.5812757071	-0.7920020203	0.0007638581

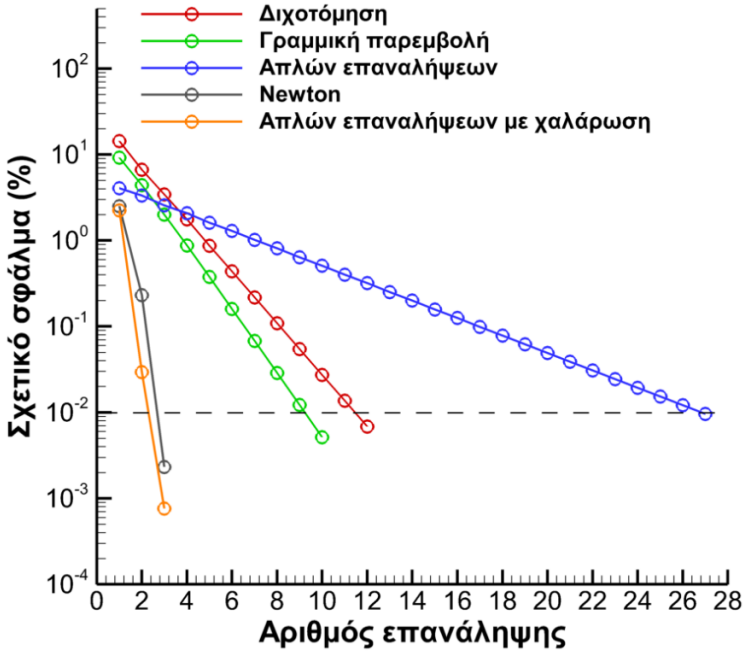
Άρα τελικά ο αλγόριθμος συγκλίνει στη ρίζα: 3.5812743898 με σχετικό σφάλμα 0.0007638581% μετά από 3 επαναλήψεις.

1στ. Σύνοψη αποτελεσμάτων

Στον παρακάτω πίνακα δίνονται συνοπτικά τα τελικά αποτελέσματα όλων των μεθόδων.

Μέθοδος	**Επαναλήψεις**	**Ρίζα**	**Σχετικό σφάλμα (%)**
Διχοτόμηση	11	3.58130	0.00682
Γραμμική παρεμβολή	9	3.58114	0.00515
Απλών επαναλήψεων	27	3.58143	0.00962
Newton	3	3.58128	0.00232
Απλών επαναλήψεων με χαλαρωση	3	3.58127	0.00076

Στην συνέχεια δίνεται το γράφημα σχετικού σφάλματος-επαναλήψεων για όλες τις μεθόδους.



Άσκηση 2

Στην μηχανική ρευστών ο συντελεστής τριβής f δίδεται από την εξίσωση $\frac{1}{\sqrt{f}} = 4\log\left(\mathrm{Re}\sqrt{f}\right) - 0.4$. Να βρεθεί ο συντελεστής τριβής για $\mathrm{Re} = 300000$.

Απάντηση:

Για την εύρεση του συντελεστή τριβής θα χρησιμοποιηθεί η μέθοδος Newton. Το πρόγραμμα σε Fortran είναι το ίδιο που δόθηκε στο υποερώτημα δ της άσκησης 1 με μόνες αλλαγές τον υπολογισμό των συναρτήσεων f, Derf και Der2f οι οποίες δίνονται παρακάτω.

```
real*8 function f(x)
real*8,intent(in)::x
real*8::Re=300000.
f=1/x-4*Log(Re*x)+0.4
end function f

real*8 function Derf(x)
real*8,intent(in)::x
Derf=-(1/x**2)-4/x
end function Derf

real*8 function Der2f(x)
real*8,intent(in)::x
Der2f=2/x**3+4/x**2
end function Der2f
```

Το παραπάνω πρόγραμμα και με αρχική εκτίμηση $x_0 = 0.01$, σχετικό σφάλμα = 0.01 και μέγιστο πλήθος επαναλήψεων = 100 δίνει:

n	$\mathbf{x^n}$	$\mathbf{f\left(x^n\right)}$	$\mathbf{f'\left(x^n\right)}$	$\left\|\frac{f''(x)}{2f'(x)}\right\|$	**err (%)**
0	1.00000E-02	6.83745E+01	-1.04000E+04	9.80769E+01	-
1	1.65745E-02	2.66872E+01	-3.88150E+03	5.84581E+01	3.96663E+01
2	2.34500E-02	7.60940E+00	-1.98909E+03	4.08155E+01	2.93198E+01
3	2.72755E-02	1.02381E+00	-1.49082E+03	3.48596E+01	1.40257E+01
4	2.79623E-02	2.39179E-02	-1.42200E+03	3.39637E+01	2.45596E+00
5	2.79791E-02	1.36553E-05	-1.42038E+03	3.39423E+01	6.01157E-02
6	2.79791E-02	4.45794E-12	-1.42038E+03	3.39423E+01	3.43608E-05

Άρα η μέθοδος Newton συγκλίνει στην λύση $f = 0.02797910$ με σχετικό σφάλμα 3.43608E-05% μετά από 6 επαναλήψεις.

Άσκηση 3

Η εξίσωση που περιγράφει την παραμόρφωση μιας ελαστικής δοκού που περιλαμβάνει ένα γραμμικά μεταβαλλόμενο φορτίο είναι $y = \frac{w_0}{120EIL}\left(-x^5 + 2L^2x^3 - L^4x\right)$. Να βρεθεί το σημείο μέγιστης παραμόρφωσης και στην συνέχεια η τιμή της ($E = 50.000\ kN/cm^2$, $I = 30.000\ cm^4$, $L = 450\ cm$, $w_0 = 1.75\ kN/cm$).

Απάντηση:

Για την εύρεση του σημείου μέγιστης παραμόρφωσης αρκεί να βρεθούν οι ρίζες που μηδενίζουν την πρώτη παράγωγο της συνάρτησης παραμόρφωσης y:

$$y'(x) = \frac{w_0}{120EIL}\left(-5x^4 + 6L^2x^2 - L^4\right)$$

Η παραπάνω εξίσωση αποτελεί μια διτετράγωνη εξίσωση και θέτοντας $a = x^2$ γράφεται ως:

$$y'(a) = \frac{w_0}{120EIL}\left(-5a^2 + 6L^2a - L^4\right)$$

Η διακρίνουσα του τριωνύμου προκύπτει $\Delta = 16L^4$ και οι ρίζες $a_1 = L^2$ και $a_2 = L^2/5$.

Άρα οι ρίζες της $y'(x) = 0$ προκύπτει να είναι $x_{1,2} = \pm L$ και $x_{3,4} = \pm L/\sqrt{5}$.

Η τιμή της παραμόρφωσης y είναι 0 cm για $x_{1,2} = \pm L$ ενώ παίρνει την τιμή $y = \mp 0.1141$ cm για $x_{3,4} = \pm L/\sqrt{5}$.

Το πρόγραμμα σε Fortran είναι το ίδιο που δόθηκε στο υποερώτημα δ της άσκησης 1 όπου έχει αλλαχθεί ο υπολογισμός των συναρτήσεων f, Derf και Der2f οι οποίες δίνονται παρακάτω. Επίσης έχει προστεθεί και η συνάρτηση Anal_F για τον υπολογισμό της τιμής της παραμόρφωσης.

```
real*8 function f(x)
real*8,intent(in)::x
real*8::L=450.
f=-5.*x**4+6.*L**2*x**2-L**4
end function f

real*8 function Derf(x)
real*8,intent(in)::x
real*8::L=450.
Derf=-20.*x**3+12.*L**2*x
end function Derf
```

```
real*8 function Der2f(x)
real*8,intent(in)::x
real*8::L=450.
Der2f=-60.*x**2+24.*L**2
end function Der2f

real*8 function Anal_F(x)
real*8, intent(IN)::x
real*8::w0=1.75,E=5.0E+4,I=3.0e+4,L=450.
Anal_F=(w0/(120*E*I*L))*(-x**5+2*(L**2)*(x**3)-(L**4)*x)
end function Anal_F
```

Το παραπάνω πρόγραμμα και με σχετικό σφάλμα = 0.01% και μέγιστο πλήθος επαναλήψεων 100 δίνει:

Αρχική εκτίμηση	Επαναλήψεις	Ρίζα	Παραμόρφωση
500	4	450.000000	0.000000
-500	4	-450.000000	0.000000
150	3	201.246118	-0.114107
-150	3	-201.246118	0.114107

Σημειώνεται πως η δοκός βρίσκεται μεταξύ $x = 0$ και $x = L$. Άρα από την φυσική σκοπιά του προβλήματος μας ενδιαφέρουν μόνο οι δύο θετικές ρίζες $x_1 = 201.25$ και $x_3 = 450.00$.

<u>Άσκηση 4</u>

Να βρεθούν με επαναληπτική μέθοδο και με ακρίβεια τεσσάρων σημαντικών ψηφίων οι πρώτες 5 θετικές ρίζες της εξίσωσης:

$$x\tan(x) = -\omega^2 \frac{h}{g},\ x \geq 0$$

όπου $g = 9.8067\ m/s^2$, $h = 5\ m$ και $\omega = 2\ rad/s$. Δικαιολογείστε την επιλογή σας ως προς την μέθοδο που θα εφαρμόσετε και βρείτε το ρυθμό σύγκλισης της μεθόδου.

Απάντηση:

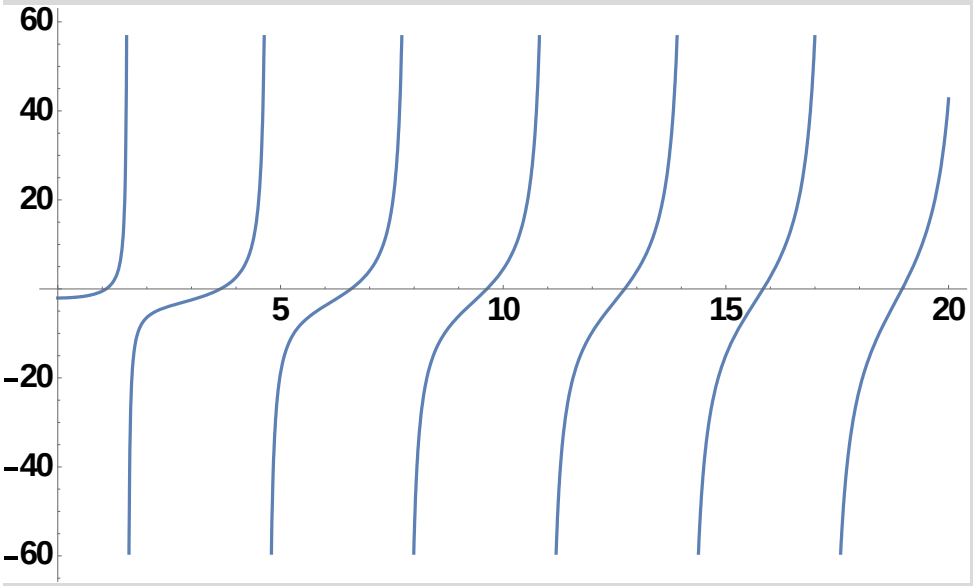
Θα γίνει χρήση της μεθόδου Newton γιατί είναι μία γρήγορη, ως προς τη σύγκλιση, μέθοδος επίλυσης εξισώσεων. Θα πρέπει όμως η αρχική προσέγγιση της ρίζας να είναι κοντά στην πραγματική.

Πρέπει να ικανοποιείται και το κριτήριο σύγκλισης: $\left|\frac{f''(x)}{2f'(x)}\right| < 1$

Επειδή ζητείται ακρίβεια τεσσάρων σημαντικών ψηφίων θα πρέπει να έχουμε σχετικό σφάλμα μικρότερο του 0.0001 ή 0.01%.

Δημιουργείται η γραφική παράσταση της συνάρτησης μέσω Mathematica ως:

```
w=2;
h=5;
g=9.8067;
f[x_]:=x*Tan[x]-w^2*h/g;
Plot[f[x],{x,0,20},LabelStyle->{14,GrayLevel[0],Bold}]
```



Το πρόγραμμα σε Fortran είναι το ίδιο που δόθηκε στο υποερώτημα δ της άσκησης 1 με μόνες αλλαγές τον υπολογισμό των συναρτήσεων f, Derf και Der2f οι οποίες δίνονται παρακάτω.

```
real*8 function f(x)
real*8,intent(in)::x
real*8::w=2,h=5,g=9.8067
f=x*Tan(x)-w**2*h/g
end function f

real*8 function Derf(x)
real*8,intent(in)::x
Derf=x/Cos(x)**2+Tan(x)
end function Derf

real*8 function Der2f(x)
real*8,intent(in)::x
Der2f=2/Cos(x)**2+2*x*Tan(x)/Cos(x)**2
end function Der2f
```

1η Ρίζα: Με αρχική προσέγγιση $x_0 = 1$ η μέθοδος συγκλίνει σε 4 επαναλήψεις στην ρίζα 1.082734 με σφάλμα 2.965436E-05%.

Παρατηρείται ότι ισχύει $\left|\frac{f''(x)}{2f'(x)}\right| > 1$. Παρόλα αυτά η μέθοδος συγκλίνει και μάλιστα ταχύτατα.

n	$\mathbf{x^n}$	$\mathbf{f(x^n)}$	$\mathbf{f'(x^n)}$	$\left\|\frac{\mathbf{f''(x^n)}}{\mathbf{2f'(x^n)}}\right\|$	**err (%)**
0	1.000000E+00	-4.820143E-01	4.982927E+00	1.758093E+00	-
1	1.096733E+00	9.809248E-02	7.211696E+00	2.087647E+00	8.820119E+00
2	1.083131E+00	2.709350E-03	6.818753E+00	2.032050E+00	1.255790E+00
3	1.082734E+00	2.185819E-06	6.807756E+00	2.030479E+00	3.669766E-02
4	1.082734E+00	1.425526E-12	6.807747E+00	2.030478E+00	2.965436E-05

2η Ρίζα: Με αρχική προσέγγιση $x_0 = 3$ η μέθοδος συγκλίνει σε 5 επαναλήψεις στην ρίζα 3.651008 με σφάλμα 9.843411E-06%.

n	$\mathbf{x^n}$	$\mathbf{f(x^n)}$	$\mathbf{f'(x^n)}$	$\left\|\frac{\mathbf{f''(x^n)}}{\mathbf{2f'(x^n)}}\right\|$	**err (%)**
0	3.000000E+00	-2.467062E+00	2.918412E+00	2.001056E-01	-
1	3.845344E+00	1.224203E+00	7.463973E+00	9.826981E-01	2.198357E+01
2	3.681329E+00	1.659561E-01	5.601579E+00	7.775923E-01	4.455318E+00
3	3.651702E+00	3.714995E-03	5.354345E+00	7.462668E-01	8.113112E-01
4	3.651008E+00	1.922274E-06	5.348806E+00	7.455496E-01	1.900374E-02
5	3.651008E+00	5.151435E-13	5.348803E+00	7.455492E-01	9.843411E-06

3η Ρίζα: Με αρχική προσέγγιση $x_0 = 5.5$ η μέθοδος συγκλίνει σε 4 επαναλήψεις στην ρίζα 6.583585 με σφάλμα 5.051368E-03%.

n	$\mathbf{x^n}$	$\mathbf{f(x^n)}$	$\mathbf{f'(x^n)}$	$\left\|\frac{\mathbf{f''(x^n)}}{\mathbf{2f'(x^n)}}\right\|$	**err (%)**
0	5.500000E+00	-7.515134E+00	9.955948E+00	8.951416E-01	-
1	6.254839E+00	-2.216773E+00	6.231513E+00	1.321204E-01	1.206808E+01
2	6.610575E+00	2.055984E-01	7.712618E+00	4.692682E-01	5.381317E+00
3	6.583917E+00	2.503299E-03	7.527335E+00	4.429875E-01	4.048867E-01
4	6.583585E+00	3.686605E-07	7.525118E+00	4.426621E-01	5.051368E-03

4η Ρίζα: Με αρχική προσέγγιση $x_0 = 9$ η μέθοδος συγκλίνει σε 3 επαναλήψεις στην ρίζα 9.633401 με σφάλμα 5.990097E-03%.

n	$\mathbf{x^n}$	$\mathbf{f\left(x^n\right)}$	$\mathbf{f'\left(x^n\right)}$	$\left\|\frac{\mathbf{f''\left(x^n\right)}}{\mathbf{2f'\left(x^n\right)}}\right\|$	**err (%)**
0	9.000000E+00	-6.110263E+00	1.038899E+01	3.560599E-01	-
1	9.588148E+00	-4.589201E-01	1.001352E+01	2.647041E-01	6.134114E+00
2	9.633978E+00	5.932379E-03	1.028053E+01	3.095776E-01	4.757128E-01
3	9.633401E+00	1.058996E-06	1.027686E+01	3.090093E-01	5.990097E-03

5η Ρίζα: Με αρχική προσέγγιση $x_0 = 12$ η μέθοδος συγκλίνει σε 4 επαναλήψεις στην ρίζα 12.72528 με σφάλμα 1.216385E-05%.

n	$\mathbf{x^n}$	$\mathbf{f\left(x^n\right)}$	$\mathbf{f'\left(x^n\right)}$	$\left\|\frac{\mathbf{f''\left(x^n\right)}}{\mathbf{2f'\left(x^n\right)}}\right\|$	**err (%)**
0	1.200000E+01	-9.669741E+00	1.621595E+01	5.741922E-01	-
1	1.259631E+01	-1.662179E+00	1.263756E+01	1.090779E-01	4.734008E+00
2	1.272784E+01	3.374772E-02	1.322841E+01	2.384796E-01	1.033380E+00
3	1.272529E+01	2.045129E-05	1.321241E+01	2.359534E-01	2.004792E-02
4	1.272528E+01	7.459366E-12	1.321240E+01	2.359518E-01	1.216385E-05

Για τον ρυθμό σύγκλισης της πρώτης ρίζας δημιουργείται το διάγραμμα του σχετικού σφάλματος (err) ανά επανάληψη (n). Παρατηρείται από την κλήση του γραφήματος η οποία αυξάνεται ανά επανάληψη ότι η σύγκλιση είναι τουλάχιστον τετραγωνική.