CHAPTER 7

7.1 In a fashion similar to Example 7.1, n = 4, $a_0 = -20$, $a_1 = 3$, $a_2 = 14.5$, $a_3 = -7.5$, $a_4 = 1$ and t = 2. These can be used to compute

```
r = a_4 = 1

a_4 = 0

For i = 3,

s = a_3 = -7.5

a_3 = r = 1

r = s + rt = -7.5 + 1(2) = -5.5

For i = 2,

s = a_2 = 14.5

a_2 = r = -5.5

r = s + rt = 14.5 - 5.5(2) = 3.5

For i = 1,

s = a_1 = 3
```

 $a_1 = r = 3.5$

r = s + rt = 3 + 3.5(2) = 10

For i = 0, $s = a_0 = -20$ $a_0 = r = 10$ r = s + rt = -20 + 10(2) = 0

Therefore, the quotient is $x^3 - 5.5x^2 + 3.5x + 10$ with a remainder of zero. Thus, 2 is a root. This result can be easily verified with MATLAB,

7.2 In a fashion similar to Example 7.1, n = 5, $a_0 = 10$, $a_1 = -7$, $a_2 = -6$, $a_3 = 1$, $a_4 = -5$, $a_5 = 1$, and t = 2. These can be used to compute

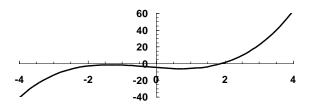
```
r = a_4 = 1
a_4 = 0
For i = 4,
s = a_4 = -5
a_3 = r = 1
```

r = s + rt = -5 + 1(2) = -3For i = 3, $s = a_3 = 1$ $a_3 = r = -3$ r = s + rt = 1 - 3(2) = -5For i = 2, $s = a_2 = -6$ $a_2 = r = -5$ r = s + rt = -6 - 5(2) = -16For i = 1, $s = a_1 = -7$ $a_1 = r = -16$ r = s + rt = -7 - 16(2) = -39For i = 0, $s = a_0 = 10$ $a_0 = r = -39$

r = s + rt = 10 - 39(2) = -68

Therefore, the quotient is $x^4 - 3x^3 - 5x^2 - 16x - 39$ with a remainder of -68. Thus, 2 is not a root. This result can be easily verified with MATLAB,

7.3 (a) A plot indicates a root at about x = 2.



Try initial guesses of $x_0 = 1$, $x_1 = 1.5$, and $x_2 = 2.5$. Using the same approach as in Example 7.2,

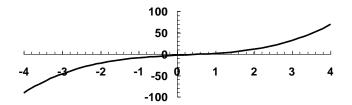
First iteration:		
f(1) = -6	f(1.5) = -3.875	f(2.5) = 9.375
$h_0 = 0.5$	$h_1 = 1$	

$$\begin{split} \delta_0 &= 4.25 & \delta_1 = 13.25 \\ a &= \frac{13.25 - 4.25}{1 + 0.5} = 6 & b = 6(1) + 13.25 = 19.25 & c = 9.375 \\ x_3 &= 2.5 + \frac{-2(9.375)}{19.25 + \sqrt{19.25^2 - 4(6)(9.375)}} = 1.901244 \\ \varepsilon_a &= \left| \frac{1.901244 - 2.5}{1.901244} \right| \times 100\% = 31.49\% \end{split}$$

The iterations can be continued as tabulated below:

i	X 3	Ea
0	1.901244	31.4929%
1	1.919270	0.9392%
2	1.919639	0.0192%
3	1.919640	0.0000%

(**b**) A plot indicates a root at about x = 0.7.



Try initial guesses of $x_0 = 0.5$, $x_1 = 1$, and $x_2 = 1.5$. Using the same approach as in Example 7.2,

$$\begin{array}{l} \underline{\text{First iteration:}}\\f(0.5) = -1 & f(1) = 1.5 & f(1.5) = 5.25\\h_0 = 0.5 & h_1 = 0.5\\\delta_0 = 5 & \delta_1 = 7.5\\a = \frac{7.5 - 5}{0.5 + 0.5} = 2.5 & b = 2.5(0.5) + 7.5 = 8.75 & c = 5.25\\x_3 = 1.5 + \frac{-2(5.25)}{8.75 + \sqrt{8.75^2 - 4(2.5)(5.25)}} = 0.731071\\\varepsilon_a = \left|\frac{0.731071 - 1.5}{0.731071}\right| \times 100\% = 105.18\%\end{array}$$

The iterations can be continued as tabulated below:

i	X 3	Ea
0	0.731071	105.1785%
1	0.720767	1.4296%

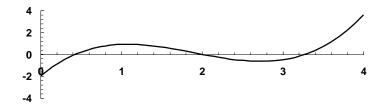
PROPRIETARY MATERIAL. © The McGraw-Hill Companies, Inc. All rights reserved. <u>No part of this Manual</u> may be displayed, reproduced or distributed in any form or by any means, without the prior written permission of the publisher, or used beyond the limited distribution to teachers and educators permitted by McGraw-Hill for their individual course preparation. If you are a student using this Manual, you are using it without permission.

2	0.721231	0.0643%
3	0.721230	0.0001%

7.4 Here are MATLAB sessions to determine the roots:

```
(a)
>> a=[1 -1 3 -2];
>> roots(a)
ans =
   0.1424 + 1.6661i
   0.1424 - 1.6661i
   0.7152
(b)
>> a=[2 0 6 0 10];
>> roots(a)
ans =
  -0.6067 + 1.3668i
 -0.6067 - 1.3668i
   0.6067 + 1.3668i
   0.6067 - 1.3668i
(c)
>> a=[1 -2 6 -8 8];
>> roots(a)
ans =
  -0.0000 + 2.0000i
 -0.0000 - 2.0000i
   1.0000 + 1.0000i
   1.0000 - 1.0000i
```

7.5 (a) A plot suggests 3 real roots: 0.44, 2 and 3.3.



Try r = 1 and s = -1, and follow Example 7.3

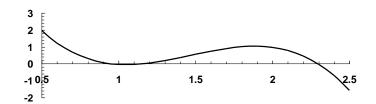
 1^{st} iteration:
 $\Delta r = 1.085$ $\Delta s = 0.887$
s = -0.1129 2^{nd} iteration:
 $\Delta r = 0.4019$ $\Delta s = -0.5565$

r = 2.487	s = -0.6694
$\frac{3^{\rm rd} \text{ iteration:}}{\Delta r = -0.0605}$ r = 2.426	$\Delta s = -0.2064$ s = -0.8758
$\frac{4^{\text{th}} \text{ iteration:}}{\Delta r = 0.00927}$ $r = 2.436$	$\Delta s = 0.00432$ s = -0.8714
$\operatorname{root}_1 = \frac{2.436 + \sqrt{2}}{}$	$\frac{.436^2 + 4(-0.8714)}{2} = 2$

$$\operatorname{root}_{2} = \frac{2.436 - \sqrt{2.436^{2} + 4(-0.8714)}}{2} = 0.4357$$

The remaining $root_3 = 3.279$.

(b) Plot suggests 3 real roots at approximately 0.9, 1.2 and 2.3.



Try r = 2 and s = -0.5, and follow Example 7.3

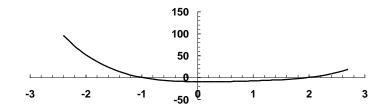
$\frac{1^{\text{st}} \text{ iteration:}}{\Delta r = 0.2302}$ $r = 2.2302$	$\Delta s = -0.5379$ s = -1.0379
$\frac{2^{\text{nd}} \text{ iteration:}}{\Delta r = -0.1799}$ $r = 2.0503$	$\Delta s = -0.0422$ s = -1.0801
$\frac{3^{rd} \text{ iteration:}}{\Delta r = 0.0532}$ $r = 2.1035$	$\Delta s = -0.01641$ s = -1.0966
$\frac{4^{\text{th}} \text{ iteration:}}{\Delta r = 0.00253}$ $r = 2.106$	$\Delta s = -0.00234$ s = -1.099

$$\operatorname{root}_{1} = \frac{2.106 + \sqrt{2.106^{2} + 4(-1.099)}}{2} = 1.1525$$

$$\operatorname{root}_2 = \frac{2.106 - \sqrt{2.106^2 + 4(-1.099)}}{2} = 0.9535$$

The remaining $root_3 = 2.2947$

(c) Plot suggests 2 real roots at approximately -1 and 2.2. This means that there should also be 2 complex roots



Try r = -1 and s = 1, and follow Example 7.3

1 st iteration:	
$\Delta r = 2.171$	$\Delta s = 3.947$
r = 1.171	s = 4.947
2 nd iteration:	
$\Delta r = -0.0483$	$\Delta s = -2.260$
r = 1.123	s = 2.688
- rd	
<u>3rd iteration:</u>	
$\Delta r = -0.0931$	$\Delta s = -0.6248$
r = 1.030	s = 2.063
Ath :	
$\frac{4^{\text{th}} \text{ iteration:}}{4^{\text{th}} 0.0288}$	0.0010
$\Delta r = -0.0288$ r = 1	$\Delta s = -0.0616$ $s = 2$
r = 1	$S \equiv Z$
$root_1 = \frac{1 + \sqrt{1^2 + 4(1^2 + 4(1^2 + 4))^2}}{2}$	$\frac{2}{2} = 2$
2	- 2
$root_2 = \frac{1 - \sqrt{1^2 + 4}}{2}$	$\overline{(2)}$
$\operatorname{root}_2 = \frac{\sqrt{2}}{2}$	= -1

The remaining roots are 1 + 2i and 1 - 2i.

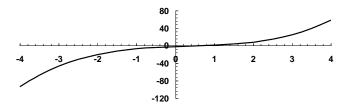
7.6 Here is a VBA program to implement the Müller algorithm and solve Example 7.2.

```
Option Explicit
```

```
Sub TestMull()
Dim maxit As Integer, iter As Integer
Dim h As Double, xr As Double, eps As Double
h = 0.1
xr = 5
eps = 0.001
maxit = 20
Call Muller(xr, h, eps, maxit, iter)
MsgBox "root = " & xr
MsgBox "Iterations: " & iter
End Sub
Sub Muller(xr, h, eps, maxit, iter)
Dim x0 As Double, x1 As Double, x2 As Double
Dim h0 As Double, h1 As Double, d0 As Double, d1 As Double
Dim a As Double, b As Double, c As Double
Dim den As Double, rad As Double, dxr As Double
x^2 = xr
x1 = xr + h * xr
x0 = xr - h * xr
Do
  iter = iter + 1
  h0 = x1 - x0
h1 = x2 - x1
  d0 = (f(x1) - f(x0)) / h0
  d1 = (f(x2) - f(x1)) / h1
  a = (d1 - d0) / (h1 + h0)
  b = a * h1 + d1
  c = f(x2)
  rad = Sqr(b * b - 4 * a * c)
  If Abs(b + rad) > Abs(b - rad) Then
    den = b + rad
  Else
    den = b - rad
  End If
  dxr = -2 * c / den
  xr = x2 + dxr
  If Abs(dxr) < eps * xr Or iter >= maxit Then Exit Do
  x0 = x1
  x1 = x2
  x2 = xr
Loop
End Sub
Function f(x)
f = x ^ 3 - 13 * x - 12
End Function
```

When this program is run, it yields the correct result of 4 in 3 iterations.

7.7 The plot suggests a real root at 0.7.



Using initial guesses of $x_0 = 0.63$, $x_1 = 0.77$ and $x_2 = 0.7$, the software developed in Prob. 7.6 yields a root of 0.715225 in 2 iterations.

7.8 Here is a VBA program to implement the Bairstow algorithm to solve Example 7.3.

```
Option Explicit
Sub PolyRoot()
Dim n As Integer, maxit As Integer, ier As Integer, i As Integer
Dim a(10) As Double, re(10) As Double, im(10) As Double
Dim r As Double, s As Double, es As Double
n = 5
a(0) = 1.25: a(1) = -3.875: a(2) = 2.125: a(3) = 2.75: a(4) = -3.5: a(5) = 1
maxit = 20
es = 0.0001
r = -1
s = -1
Call Bairstow(a(), n, es, r, s, maxit, re(), im(), ier)
For i = 1 To n
  If im(i) \ge 0 Then
    MsgBox re(i) & " + " & im(i) & "i"
  Else
    MsgBox re(i) & " - " & Abs(im(i)) & "i"
  End If
Next i
End Sub
Sub Bairstow(a, nn, es, rr, ss, maxit, re, im, ier)
Dim iter As Integer, n As Integer, i As Integer
Dim r As Double, s As Double, eal As Double, ea2 As Double
Dim det As Double, dr As Double, ds As Double
Dim r1 As Double, i1 As Double, r2 As Double, i2 As Double
Dim b(10) As Double, c(10) As Double
r = rr
s = ss
n = nn
ier = 0
ea1 = 1
ea2 = 1
Do
  If n < 3 Or iter >= maxit Then Exit Do
  iter = 0
  Do
    iter = iter + 1
    b(n) = a(n)
    b(n - 1) = a(n - 1) + r * b(n)
    c(n) = b(n)
    c(n - 1) = b(n - 1) + r * c(n)
    For i = n - 2 To 0 Step -1
      b(i) = a(i) + r * b(i + 1) + s * b(i + 2)
      c(i) = b(i) + r * c(i + 1) + s * c(i + 2)
```

```
Next i
    det = c(2) * c(2) - c(3) * c(1)
    If det <> 0 Then
      dr = (-b(1) * c(2) + b(0) * c(3)) / det
      ds = (-b(0) * c(2) + b(1) * c(1)) / det
      r = r + dr
      s = s + ds
      If r \ll 0 Then eal = Abs(dr / r) * 100
      If s \ll 0 Then ea2 = Abs(ds / s) * 100
    Else
      r = r + 1
      s = s + 1
      iter = 0
    End If
    If eal <= es And ea2 <= es Or iter >= maxit Then Exit Do
  Loop
  Call Quadroot(r, s, r1, i1, r2, i2)
  re(n) = r1
  im(n) = i1
  re(n - 1) = r2
  im(n - 1) = i2
  n = n - 2
  For i = 0 To n
    a(i) = b(i + 2)
  Next i
Loop
If iter < maxit Then
  If n = 2 Then
    r = -a(1) / a(2)
    s = -a(0) / a(2)
    Call Quadroot(r, s, r1, i1, r2, i2)
    re(n) = r1
    im(n) = i1
    re(n - 1) = r2
    im(n - 1) = i2
  Else
    re(n) = -a(0) / a(1)
    im(n) = 0
  End If
Else
  ier = 1
End If
End Sub
Sub Quadroot(r, s, r1, i1, r2, i2)
Dim disc
disc = r ^ 2 + 4 * s
If disc > 0 Then
  r1 = (r + Sqr(disc)) / 2
  r2 = (r - Sqr(disc)) / 2
  i1 = 0
  i2 = 0
Else
  r1 = r / 2
  r2 = r1
  i1 = Sqr(Abs(disc)) / 2
  i2 = -i1
End If
End Sub
```

When this program is run, it yields the correct result of -1, 0.5, 2, 1 + 0.5i, and 1 - 0.5i.

- **7.9** Using the software developed in Prob. 7.8 the following results should be generated for the three parts of Prob. 7.5:
 - (a) 3.2786, 2.0000, 0.4357
 - **(b)** 2.2947, 1.1525, 0.9535
 - (c) 2.0000, 1.0000 + 2.0000i, 1.0000 2.0000i, -1

7.10 The goal seek set up is

	A	В	C	D	E	
1	x	3.758703	Goal Seek			
2	x^3.5	102.9516	Guat Seek			
3			Set cell:	\$B\$	12	
4				S	r -	
5			To <u>v</u> alue:	80		
6	11		By changing cell	: \$B\$	51	
7						
8				OK	Cano	:el
9						_

The result is

	B2	-	∱ =B1^3.	5			
	A	В	С	D	E	F	G
1	х	3.497367	Goal See	Statue			
2	x^3.5	80.00077	dual see	N Status			
3			Goal Seekir	ng with Cell B2	2	(OK
4			found a so	ution.		Lunn	
5							Cancel
6			Target valu	ue: 80			
7			Current va	lue: 80.00	077448		Step
8							
9							Pause
10				-			

7.11 The goal seek set up is shown below. Notice that we have named the cells containing the parameter values with the labels in column A.

	A	В	C	D	E	F	G
1	Prob. 7.1	1		Goal Seek	Calcat		
2				Godi Seek			
3	g	9.8	m/s2	S <u>e</u> t cell:	F	310	N
4	c	14	kg/s		-		
5	t	8	s	To <u>v</u> alue:	[)	
6	v	35		By changing cel	l: 🔤	\$B\$8	N .
7							
8	m	50	kg		ОК	Ca	ncel
9							
10	f(v)	-3.72605					

The result is 58.717 kg as shown here:

	B10	-	<i>f</i> ∗ =g*m.	/c_*(1-EXP(-c_/	m*t))-v			
1	A	В	С	D	E	F	G	Н
1	Prob. 7.11	1		Goal Seek S	tatue			
2				Guar Seek S	latus			
3	g	9.8	m/s2	Goal Seeking	with Cell B1	10	(*****	OK
4	c	14	kg/s	found a soluti	on.		L	
5	t	8	s					Iancel
6	v	35		Target value:	0		_	
7		(2	Current value	: -3.626	556E-06		Step
8	m	58.71699	kg				-	in the second se
9			10.000					Pause
10	f(v)	-3.6E-06				-	·	s - 6

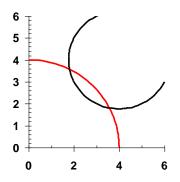
7.12 The Solver set up is shown below using initial guesses of x = y = 1. Notice that we have rearranged the two functions so that the correct values will drive both to zero. We then drive the sum of their squared values to zero by varying *x* and *y*. This is done because a straight sum would be zero if $f_1(x,y) = -f_2(x,y)$.

	C10	-	∱ =C7+C8									
		A	B	C	D	E	F	G	Н		J	K
1	Prob. 7	.12			Solver	Parameter	s				1	
2	-						<u> </u>					
3	x		1		Set Tar	get Cell:	\$C\$10				Solve	
4	y		1		Equal T	~ <u>~</u>	12		of: 0			
5					100000 000 00000 000000	anging Cells:	ax OMin	O Value	or: U	1	Close	
6			function	function ²		a second a second second						
7	f1(x,y)=	-x^2+x+0.75-	y 1.75	3.0625	\$B\$3	\$B\$4			1	Guess		
8	f2(x,y)=	x^2-y-5xy	-5	25	Subject	t to the Cons	traints:				Onlines	
9	10001000										Options	
10			sum squares	28.0625					<u>^</u>	Add		
11										hange		
12										Indingo	Reset All	
13									3	Delete		
14											Help	
15					-							- 201

The result is

8	A	В	С	D	E	F	G	Н	1	J
1	Prob. 7.12			Solver F	lesults					
2 3 4	x y	-0.188313433 0.596407992			und a solution is are satisfie		ints and optima		ports	
5 6		function	function^2	€ Kee	€ Keep Solver Solution				Answer Sensitivity Limits	
7	f1(x,y)=-x^2+x+0.75-y	0.000740524		O Re:	O Restore Original Values					0
8	f2(x,y)=x^2-y-5xy	0.00061214	3.747E-07					L		
9					к	Cancel	Save S	cenario	Т	elp
10		sum squares	9.231E-07							<u> </u>

7.13 A plot of the functions indicates two real roots at about (1.8, 3.6) and (3.6, 1.8).



PROPRIETARY MATERIAL. © The McGraw-Hill Companies, Inc. All rights reserved. <u>No part of this Manual</u> may be displayed, reproduced or distributed in any form or by any means, without the prior written permission of the publisher, or used beyond the limited distribution to teachers and educators permitted by McGraw-Hill for their individual course preparation. If you are a student using this Manual, you are using it without permission.

The Solver set up is shown below using initial guesses of (2, 4). Notice that we have rearranged the two functions so that the correct values will drive both to zero. We then drive the sum of their squared values to zero by varying x and y. This is done because a straight sum would be zero if $f_1(x,y) = -f_2(x,y)$.

	C10	▼ fx	=C7+C8									
0		А	В	С	D	E	F	G	H	I I	J	К
1	Prob. 7.13	3			Salva	r Paramete	-					
2					JULYE	i Faramete	19					
3	x		2		Set T	arget Cell:	\$C\$10				Solve	
4	y		4		_	To: O			ie of: 0	1		
5						hanging Cells:		<u>n</u> 💿 <u>V</u> ak	Je or:		Close	
6			function	function^2	EYC	nanging Celis:						
7	f1(x,y)=5-	(x-4)^2-(y-4)^2	1	1	\$B\$	3:\$B\$4			1	Guess		
8	f2(x,y)=16	-x^2-y^2	4	16	Subi	ect to the Con	ctrainte:					
9	A STREET				500	see to the con	ser dirics,				Options	
10			sum squares	17]					<u>^</u>	Add		
11										Channe		
12			1							Change	Reset A	
13										Delete		
14											Help	
15					- C							

The result is

	A	В	C	D	E	F	G	H	1	J	
1	Prob. 7.13	-		Solver I	Doculte						
2				JULVET I	lesuits						
3	x	1.805916437		Solver fo	ound a solutio	n. All constra	aints and opti	mality			
4	у	3.569117141		Conditions are satisfied.							
5									Answer		
6		function	function ²						1000		
7	f1(x,y)=5-(x-4)^2-(y-4)^2	0.000337279	1.138E-07	Restore Original Values						0	
8	f2(x,y)=16-x^2-y^2	6.86609E-05	4.714E-09	O RE:	score <u>O</u> riginal	values					
9					ж	Cancel		Scenario.		telp	
10		sum squares	1.185E-07			Cancer		Scenario.		Teib	

For guesses of (4, 2) the result is (3.5691, 1.8059).

7.14 MATLAB session:

```
>> a = poly([4 -2 1 -5 7])
a =
    1
         -5 -35 125 194 -280
>> polyval(a,1)
ans =
    0
>> polyder(a)
ans =
    5
       -20 -105
                  250 194
>> b = poly([4 - 2])
b =
    1 -2 -8
>> [d,e] = deconv(a,b)
d =
    1
         -3
              -33
                     35
```

```
e =
      0 0 0 0
   0
                             0
>> roots(d)
ans =
   7.0000
  -5.0000
   1.0000
>> conv(d,b)
ans =
   1
        -5
           -35 125 194 -280
>> r = roots(a)
r =
   7.0000
  -5.0000
   4.0000
  -2.0000
   1.0000
```

7.15 MATLAB sessions:

Prob. 7.5b:

```
>> a=[-3.704 16.3 -21.97 9.34];
>> roots(a)
ans =
    2.2947
    1.1525
    0.9535
```

Prob. 7.5c:

```
>> a=[1 -3 5 -1 -10];
>> roots(a)
ans =
    2.0000
    1.0000 + 2.0000i
    1.0000 - 2.0000i
    -1.0000
```

7.16 Here is a program written in Fortran 90:

```
PROGRAM Root
Use IMSL !This establishes the link to the IMSL libraries
Implicit None !forces declaration of all variables
Integer::nroot
```

```
Parameter(nroot=1)
Integer::itmax=50
Real::errabs=0.,errrel=1.E-5,eps=0.,eta=0.
Real::f,x0(nroot) ,x(nroot)
External f
Integer::info(nroot)
Print *, "Enter initial guess"
Read *, x0
Call ZReal(f,errabs,errrel,eps,eta,nroot,itmax,x0,x,info)
Print *, "root = ", x
Print *, "iterations = ", info
End
Function f(x)
Implicit None
Real::f,x
f = x^{*} - x^{*} - x^{*} - 2.
End
```

The output for Prob. 7.4*a* would look like

Enter initial guess
.5
root = 0.7152252
iterations = 6
Press any key to continue

The other parts of Probs 7.4 have complex roots and therefore cannot be evaluated with ZReal. The roots for Prob. 7.5 can be evaluated by changing the function and obtaining the results:

7.5 (a) 2, 0.4357, 3.279 **7.5** (b) 1.1525, 0.9535, 2.295 **7.5** (c) 2, -1

7.17
$$x_2 = 0.62, x_1 = 0.64, x_0 = 0.60$$

$$h_0 = 0.64 - 0.60 = 0.04$$
$$h_1 = 0.62 - 0.64 = -0.02$$

$$\delta_0 = \frac{60 - 20}{0.64 - 0.60} = 1000$$

$$\delta_1 = \frac{50 - 60}{0.62 - 0.64} = 500$$

$$a = \frac{\delta_1 - \delta_0}{h_1 + h_0} = \frac{500 - 1000}{-0.02 + 0.04} = 25000$$

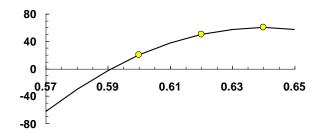
$$b = ah_1 + \delta_1 = -25000(-0.02) + 500 = 1000$$

$$c = 50$$

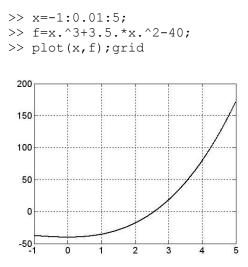
$$\sqrt{b^2 - 4ac} = \sqrt{1000^2 - 4(-25000)50} = 2449.49$$

$$t_0 = 0.62 + \frac{-2(50)}{1000 + 2449.49} = 0.591$$

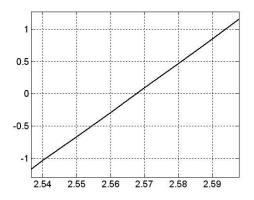
Therefore, the pressure was zero at 0.591 seconds. The result is graphically displayed below:



7.18 (a) First we will determine the root graphically



The zoom in tool can be used several times to home in on the root. For example, as shown in the following plot, a real root appears to occur at x = 2.567:



(b) The roots function yields both real and complex roots:

```
>> a=[1 3.5 0 -40];
>> roots(a)
ans =
    -3.0338 + 2.5249i
    -3.0338 - 2.5249i
    2.5676
```

7.19 (a) Excel Solver Solution: The 3 functions can be set up as a roots problems:

```
f_1(a, u, v) = a^2 - u^2 + 2v^2 = 0

f_2(a, u, v) = u + v - 2 = 0

f_3(a, u, v) = a^2 - 2a - u = 0
```

	C11	▼ fx	=SUM(C7:0	C9)						
	A	В	С	D	E	F	G	Н		J
1	Problem	7.19		Solver Da	rameters					
2				Source Pe	rameters					
3	а	1		Set Target	: Cell:	\$C\$11	E.)			Solve
4	u	1		Equal To:	O Max	O Min	Value of:	0		
5	V	1		By Chang					_	Close
6		Function	Function ^A 2		web over a					
7	Func1	2	4	\$B\$3:\$B	\$5			🔄 🖸 Gue	iss	
8	Func2	0	0	Subject to	the Constra	ints:				Continue
9	Func3	-2	4					_		Options
10		2						<u>A</u> d	d	
11		Sum Square	8					⊆har		
12									ige	Reset All
13								<u>D</u> ele	te	
14							-1			<u>H</u> elp
15	1			1						
2		-			-	-	-			
	A	В	C	D	E	F	G	Н	1	J
1	Problem	7.19		Solver Results						
2				Solver	Results					
3	а	-0.48785879		Calverd	Coursel or solut	ine All sees	to state and and	hine although		

1	Problem	7.19		Contraction and the second	
2				Solver Results	
3	а	-0.48785879		Colors for a distant All and the for a distant	
4	u	1.21398699		 Solver found a solution. All constraints and optimality conditions are satisfied. 	Reports
5	v	0.78613415			
6		Function	Function ²		Answer Sensitivity
7	Func1	0.00025561	6.534E-08	<u>Keep Solver Solution</u>	Limits
8	Func2	0.00012114	1.467E-08	Restore Original Values	<u>×</u>
9	Func3	-0.00026321	6.928E-08		
10		-		OK Cancel Save Scenario	<u>H</u> elp
11		Sum Square	1.493E-07		

If you use initial guesses of a = -1, u = 1, and v = -1, the Solver finds another solution at a = -1.6951, u = 6.2634, and v = -4.2636

(b) Symbolic Manipulator Solution:

MATLAB:

```
>> syms a u v
>> S=solve(u^2-2*v^2-a^2,u+v-2,a^2-2*a-u);
>> double(S.a)
ans =
  3.0916 + 0.3373i
  3.0916 - 0.3373i
  -0.4879
  -1.6952
>> double(S.u)
ans =
  3.2609 + 1.4108i
   3.2609 - 1.4108i
  1.2140
   6.2641
>> double(S.v)
ans =
  -1.2609 - 1.4108i
 -1.2609 + 1.4108i
  0.7860
  -4.2641
```

Mathcad:

Problem 7.19 (Mathcad)

Find(a, u, v) = $\begin{pmatrix} -1.6952 \\ 6.2641 \\ -4.2641 \end{pmatrix}$ a := 3 + i u := 3 + i v := -1 - i Given f(a, u, v) = 0 g(a, u, v) = 0 h(a, u, v) = 0 Find(a, u, v) = $\begin{pmatrix} 3.0916+ 0.3373i \\ 3.2609+ 1.4108i \\ -1.2609- 1.4108i \end{pmatrix}$

Therefore, we see that the two real-valued solutions for a, u, and v are (-0.4879, 1.2140, 0.7860) and (-1.6952, 6.2641, -4.2641). In addition, MATLAB and Mathcad also provide the complex solutions as well.

7.20 MATLAB can be used to determine the roots of the numerator and denominator:

```
>> n=[1 12.5 50.5 66];
>> roots(n)
ans =
    -5.5000
    -4.0000
    -3.0000
>> d=[1 19 122 296 192];
>> roots(d)
ans =
    -8.0000
    -6.0000
    -4.0000
    -1.0000
```

The transfer function can be written as

$$G(s) = \frac{(s+5.5)(s+4)(s+3)}{(s+8)(s+6)(s+4)(s+1)}$$

7.21

```
function root = bisection(func,xl,xu,es,maxit)
% root = bisection(func,xl,xu,es,maxit):
% uses bisection method to find the root of a function
% input:
% func = name of function
% xl, xu = lower and upper guesses
% es = (optional) stopping criterion (%)
% maxit = (optional) maximum allowable iterations
% output:
% root = real root
```

```
if func(x1) * func(xu) >0 % if guesses do not bracket a sign change
  disp('no bracket') %display an error message
 return
                        %and terminate
end
% if necessary, assign default values
if nargin<5, maxit=50; end %if maxit blank set to 50
                             %if es blank set to 0.001
if nargin<4, es=0.001; end
% bisection
iter = 0;
xr = xl;
while (1)
 xrold = xr;
 xr = (x1 + xu) / 2;
 iter = iter + 1;
 if xr \sim = 0, ea = abs((xr - xrold)/xr) * 100; end
  test = func(xl)*func(xr);
 if test < 0
   xu = xr;
  elseif test > 0
   xl = xr;
  else
   ea = 0;
  end
  if ea <= es | iter >= maxit, break, end
end
root = xr;
```

The following is a MATLAB session that uses the function to solve Example 5.3 with $\varepsilon_s = 0.0001$.

```
>> fcd=inline('(9.81*68.1/cd)*(1-exp(-0.146843*cd))-40','cd');
>> format long
>> bisection(fcd,5,15,0.0001)
ans =
    14.80114936828613
```

7.22

```
function root = falsepos(func,xl,xu,es,maxit)
% falsepos(func,xl,xu,es,maxit):
  uses the false position method to find the root of the function func
8
% input:
8
   func = name of function
8
  x1, xu = lower and upper guesses
8
  es = (optional) stopping criterion (%) (default = 0.001)
  maxit = (optional) maximum allowable iterations (default = 50)
8
% output:
8
  root = real root
if func(xl)*func(xu)>0 %if guesses do not bracket a sign change
  error('no bracket')
                      %display an error message and terminate
end
% default values
if nargin<5, maxit=50; end
if nargin<4, es=0.001; end
% false position
iter = 0;
xr = xl;
while (1)
  xrold = xr;
```

```
xr = xu - func(xu)*(xl - xu)/(func(xl) - func(xu));
iter = iter + 1;
if xr ~= 0, ea = abs((xr - xrold)/xr) * 100; end
test = func(xl)*func(xr);
if test < 0
    xu = xr;
elseif test > 0
    xl = xr;
else
    ea = 0;
end
if ea <= es | iter >= maxit, break, end
end
root = xr;
```

The following is a MATLAB session that uses the function to solve Example 5.5:

```
>> fcd=inline('(9.81*68.1/cd)*(1-exp(-0.146843*cd))-40','cd');
>> format long
>> falsepos(fcd,5,15,0.0001)
ans =
    14.80114660933235
```

7.23

```
function root = newtraph(func,dfunc,xr,es,maxit)
% root = newtraph(func,dfunc,xguess,es,maxit):
8
  uses Newton-Raphson method to find the root of a function
% input:
8
   func = name of function
9
   dfunc = name of derivative of function
   xguess = initial guess
8
   es = (optional) stopping criterion (%)
8
8
  maxit = (optional) maximum allowable iterations
% output:
8
  root = real root
% if necessary, assign default values
if nargin<5, maxit=50; end %if maxit blank set to 50
if nargin<4, es=0.001; end %if es blank set to 0.001
% Newton-Raphson
iter = 0;
while (1)
 xrold = xr;
 xr = xr - func(xr)/dfunc(xr);
  iter = iter + 1;
 if xr \sim= 0, ea = abs((xr - xrold)/xr) * 100; end
 if ea <= es | iter >= maxit, break, end
end
root = xr;
```

The following is a MATLAB session that uses the function to solve Example 6.3 with $\varepsilon_s = 0.0001$.

```
>> format long
>> f=inline('exp(-x)-x','x');
>> df=inline('-exp(-x)-1','x');
>> newtraph(f,df,0)
ans =
        0.56714329040978
```

7.24

```
function root = secant(func, xrold, xr, es, maxit)
% secant(func, xrold, xr, es, maxit):
% uses secant method to find the root of a function
% input:
8
  func = name of function
8
    xrold, xr = initial guesses
   es = (optional) stopping criterion (%)
8
8
   maxit = (optional) maximum allowable iterations
% output:
  root = real root
8
% if necessary, assign default values
if nargin<5, maxit=50; end
                            %if maxit blank set to 50
if nargin<4, es=0.001; end
                             %if es blank set to 0.001
% Secant method
iter = 0;
while (1)
 xrn = xr - func(xr) * (xrold - xr) / (func(xrold) - func(xr));
  iter = iter + 1;
  if xrn \sim= 0, ea = abs((xrn - xr)/xrn) * 100; end
  if ea <= es | iter >= maxit, break, end
 xrold = xr:
  xr = xrn;
end
root = xrn;
```

Test by solving Example 6.6:

```
>> format long
>> f=inline('exp(-x)-x','x');
>> secant(f,0,1)
ans =
        0.56714329040970
```

7.25

```
function root = modsec(func,xr,delta,es,maxit)
% modsec(func,xr,delta,es,maxit):
% uses modified secant method to find the root of a function
% input:
   func = name of function
8
8
   xr = initial guess
9
   delta = perturbation fraction
8
   es = (optional) stopping criterion (%)
  maxit = (optional) maximum allowable iterations
8
% output:
  root = real root
8
% if necessary, assign default values
if nargin<5, maxit=50; end
                            %if maxit blank set to 50
if nargin<4, es=0.001; end
                             %if es blank set to 0.001
if nargin<3, delta=1E-5; end %if delta blank set to 0.00001
% Secant method
iter = 0;
while (1)
 xrold = xr;
  xr = xr - delta*xr*func(xr)/(func(xr+delta*xr)-func(xr));
  iter = iter + 1;
  if xr \sim = 0, ea = abs((xr - xrold)/xr) * 100; end
```

```
if ea <= es | iter >= maxit, break, end
end
root = xr;
```

Test by solving Example 6.8:

```
>> format long
>> f=inline('exp(-x)-x','x');
>> modsec(f,1,0.01)
ans =
        0.56714329027265
```