To the reader: this is the computer code that appears in my textbook A MATLAB Companion to Complex Variables, published in 2016. It is placed here for your convenience so that you can copy and paste the code into a program you might be trying out without having to type it yourself. I have not included short programs that are in the book. You'll have to type those for yourself. Please tell me of any errors you find. Write to me at A_Wunsch@uml.edu or ADunsch@gmail.com Thanks ADW March 22, 2016

Chapter 1

Section 1.3.4

```
z=input('the complex value of z to be considered')
m=input('input the positive integer m')
v=zeros(1,m+1);%creates a row vector length m+1
%elements are zero.
v(1,1)=1;% the first coeff is 1, the coeff of w^m
v(1,m+1)=-z;% the constant term (the last coeff) is -z
%the following solves the equation w^m-z=0 for w=z^(1/m)
the_roots =roots(v)
prod_roots=prod(the_roots)
%the preceding should equal (-1)^m times z.
check_prod=(-1)^m*z
check_roots=sum(the_roots)
%the preceding should be zero to a good approx.
```

Chapter 2

Section 2.1

```
x=linspace(0,2,10);
y=linspace(0,7,20);
[x y]=meshgrid(x,y);
z=x+i*y;
w=z.^2;
u=real(w);
mesh(x,y,u);view(45,60); colormap(gray); colorbar
xlabel('x');ylabel('y')
title ('real part of z^2')
```

Section 2.2

```
x=linspace(-2.5,2.5,100);
y=linspace (-2.5,2.5,100);
n=linspace (-2,2,5);
[x,y]=meshgrid(x,y);
z=x+i*y;
 w=z.^2;
wi=imag(w);
 [c,h]= contour(x,y,wi,n);colormap(gray)
   xlabel('x');ylabel('y');title('Contour Lines for Im(z^2)')
   grid
   clabel(c,h)
```

Section 2.3

```
x=linspace(-4,4,1000);
y=linspace (-4,4,1000);
[x,y]=meshgrid(x,y);
z=x+i*y;
w=abs(z-i);
v=abs(z+i);
ww=(w<=2);
vv=(v<=2);
tt=ww&vv;
meshz(x,y,-tt);colormap(gray)
view(2);
hold on
x=linspace(-4,4,9);
y=x;
[x y]=meshgrid(x,y);
plot(x,y,'r');plot(y,x,'r');axis equal
title('The Intersection of the Sets |z-i|{\leq}2 and |z+i|{\leq}2 ')
xlabel('x');ylabel('y')
```

Section 2.4.1

```
x=linspace(-5,5,101);
y=linspace(-1,1,51);
[x,y]=meshgrid(x,y);
z=x+i*y;
w=sin(z+eps)./(z+eps);
wi=imag(w);
  meshz(x,y,wi);colormap gray
  grid on
  title('The Imaginary Part of sin(z)/z ')
 xlabel('x');ylabel('y')
```

Section 2.4.1.2

```
x=linspace(-1,4,150);

y=linspace(-1,1,150);

[x,y]=meshgrid(x,y);

z=x+i*y;

f=z.*(z-2).^2;

f=1./f;

f=abs(f);meshz(x,y,f);

axis([-.5 2.5 -1 1 0 30]);colormap gray

title('The Magnitude of 1/(z(z-2).^2)')

xlabel('x');ylabel('y')
```

```
view(10,15)
```

```
clf;clear
x=linspace(-1,3,100);
y=linspace (-1,1,100);
 n=[ .5 1 2];
[x,y]=meshgrid(x,y);
z=x+i*y;
w=1./(z.*(z-2).^2);
wm=abs(w);
  [c,h]= contour(x,y,wm,n);
axis equal
 grid
    hold on
   clabel(c,h);
   title('Contour Lines Showing |1/(z(z-2)^2)|')
```

Section 2.5

```
clf
x=linspace(-5,5,1000);
y=x;
[x y]=meshgrid(x,y);
figure(1)
z=x+i*y;
w=log((z-1)./(z+1));
wm=imag(w);
del=.01;
```

```
wm(abs(y)<del&-1<=x<=1)=nan;
[g h]=contour (x,y,wm,6);
clabel(g,h);grid
title('contour lines for Im log((z-1)/(z+1))')
xlabel('x');ylabel('y');
```

```
clf
x=linspace(-5,5,1000);
y=x;
[x y]=meshgrid(x,y);
z=x+i*y;
w=z.^i;
del=.01
levels=(0:5);%the values 0,1...5 for the contours
wr=real(w);wm=imag(w);
wr(abs(y)<del&-1<=x<=1)=nan;%accounts for branch cut
figure(1)
[g, h]=contour (x,y,wr,levels);
clabel(g,h);grid
title('contour lines for Re z^i')
xlabel('x');ylabel('y');
wm(abs(y)<del&-1<=x<=1)=nan;%accounts for branch cut
figure(2)
[G, H]=contour (x,y,wm,levels);
clabel(G,H);grid
xlabel('x');ylabel('y');
title('contour lines for Im z^i')
```

Chapter 3

Section 3.1

```
clear
clf
N=15;
p=ones(1,N);rp=real(p);ip=imag(p);
 for n=1:N
p(n)=1+1/(1+i)^n;
rp(n)=real(p(n));
ip(n)=imag(p(n));
b=num2str(n)
plot(rp(n),ip(n));
 text ( rp(n) , ip(n) , b , 'FontSize' , 6 );hold on
 pause(.25)
n
p(n)
 end
grid
  plot(p,'-')
  title('The First 15 Terms in the Sequence 1+1/(1+i)^n')
  hold off
```

```
clf
N=50;
p=ones(1,N);rp=real(p); ip=imag(p);
p(1)=i;
 for n=2:N
p(n)=i^p(n-1);
rp(n)=real(p(n));
ip(n)=imag(p(n));
b=num2str(n)
  plot(rp(n),ip(n),'*');
```

```
 pause(.3);% gives a .3 sec delay in plotting
 hold on
  % text ( rp(n) , ip(n) , b , 'FontSize' , 6 );hold on
n
p(n)
 end
 plot(p(1),'*');
 grid
 title('Sequence from p_n=i^{p_{n-1}}')
hold off
```

Section 3.2

```
Nmax=11;
a=0:Nmax-1;
q=1./factorial(a);
b=(2+2*i).^a;
SN=q.*b;
format long
S=(cumsum(SN)).';% this sums all the elements in the row vector SN
N=[1:Nmax]';
disp('number of terms                    series approximation        ')
[N S]
 disp('matlab computes exp(2+2i)')
exp(2+2*i)
```

```
clf
x=linspace(-2,2,100);
z=x+i*0;
w=1./cosh(z);
u=real(w);
```

```
v=imag(w);

plot(x,u)

grid

hold on

ser=1-z.^2/2+5*z.^4/4;

plot(x,ser,'--r','LineWidth',2)

ylim([-5,10]);

xlabel('x');ylabel('1/cosh(z) and its series approximation')
```

_____-

Section 3.3

```
clear

nmax=input('max number of terms in series')

for(n=1:nmax)

m=n+1;% number of points needed on curve to define the n subarcs

X=linspace(0,1,m);% this gives

%the values of X needed

Y=1-X.^2;%gives values of Y on curve

delta_x=(X(2)-X(1))/2;

x=X(1:n)+delta_x;%the n values of x coordinate where we...

% evaluate f(z)on curve (near middle of chord)

y=1-x.^2; %the values of y corresponding to the above

z=x+i*y; % the complex values of z needed on the curve

fz=cos(z);%the n values of the function on the curve

yd=Y(2:m)-Y(1:m-1);% the vertical component of the vector chord

xd=2*delta_x;%the horizontal component of the vector chord

zdchord=xd+i*yd;% the vector chord complex representation

s=fz.*zdchord; %the product of the complex vector chord...

% and the function cos z evaluated nearby on the curve
```

```
series_sum(n)=sum(s);% the sum of those products
end
number_of_terms=[1:n]';
format long
approx_value=series_sum.';
disp('number of terms in series                  series sum')
disp([number_of_terms approx_value])
disp('sin(1)-sin(i)')
disp(sin(1)-sin(i))
```

Section 4.1

```
clf
z=(1+i);
w=log(z);
figure(1);
plot(z,'*');grid on;xlabel('x');ylabel('y');
%the above plots the point 1+i in z plane,
% and labels the axes.
text(1.02,1.02,'z_0')
% above labels the above point, we use 1.02, 1.02
%and not 1,1, so as to keep the letter zo off the star *
text(1,1.6,'\fontsize {12} The z Plane')
figure(2);
plot(w,'*');grid on;xlabel('u');ylabel('v')
%above plots log(1+i) in w plane.
text(real(w)+.05,imag(w)+.05,'w_0');
```

```
%note the use of .05 in the above to keep
%the wo off the *
%labels the above point
text(0,1.7,'\fontsize{12} The w Plane')
```

```
clf
y=linspace(-5,5,11);
x=1;
z=1+i*y;
w=log(z);
figure(1)
 plot(real(z),imag(z),'.');grid on;
%plots a dot for each value used in
%z plane
 xlabel('x');ylabel('y');title('z plane');grid on
figure (2)
plot(real(w),imag(w), '.');axis equal;
%plots a dot at each value of logz
xlabel('u');ylabel('v');title('w plane')
for n=1:length(y)
     letter=char(n+64);
%the above creates the letters A, B,...
     figure(1);
     text(1.02,imag(z(n)),letter,'FontSize',10);
     % the above plots the letters A,B,
     %note that using 1.02 instead of 1 in the above
     %keeps the text off the dot in the plot.
     %a similar thing is done for figure(2)
```

```
      figure(2)

text(real(w(n))+.02,imag(w(n)),letter,'FontSize',10,'FontAngle',...
  'italic','FontWeight','bold');
      % the above plots in the w plane the
 % italicized bold A, B, C...
     % which are the images of the points in the z plane
  hold on
end;grid on
```

```
clf
y=linspace(-3,3,101);
x=1;
z1=1+i*y;
w1=log(z1);
th=linspace(-pi,pi,100);
z2=2*exp(i*th); % yields 100 points on circle of radius 2
w2=log(z2);
figure(3)
 plot(real(z1),imag(z1),'linewidth',2);hold on
 plot(real(z2),imag(z2),'linewidth',1);grid on;axis equal
xlabel('x');ylabel('y');title('z plane');grid on
figure (4)
plot(real(w1),imag(w1),'linewidth',2);hold on
plot(real(w2),imag(w2),'linewidth',1);
grid on;axis equal
xlabel('u');ylabel('v');title('w plane')
```

```
y=linspace(-3,3,6);th=linspace(-.9*pi,pi,6);
```

```
y=linspace(-3,3,6);th=linspace(-.98*pi,.98*pi,6);
z1=1+i*y; z2=2*exp(i*th);
%we will use fewer data points for the letters
%on the curves than were needed to make the curves
 for n=1:length(y)
    letter1=char(n+64);
% above creates the letters A, B,...
    letter2=char(n+96);
%above creates letters a,b, ...
     figure(3);
     text(1,imag(z1(n)),letter1,'FontSize',10);
%plots the letters A,B,
    % in the z plane on x=1;
    text(real(z2(n)),imag(z2(n)), letter2,'FontSize',10)
    % above plots the letters a,b, on the circle in the z plane
    w1(n)=log(z1(n));
    w2(n)=log(z2(n));
     figure(4);
text(real(w1(n)),imag(w1(n)),letter1,'FontSize',10,'FontAngle','italic
','FontWeight','bold');
% the above puts in the w plane the italicized bold A,B,..
 % which are the images of the points A,B,.. in the z plane
text(real(w2(n)),imag(w2(n)),letter2,'FontSize',10,'FontAngle','italic
','FontWeight','bold');
%the above puts in the w plane the italicized bold
```

```
%a,b..which are the
  %images of a,b,c in the z plane
      hold on
end;
```

Section 4.2

```
th=2*pi*linspace (0,1,1000);
z=exp(i*th);
figure (1);
plot(real(z),imag(z));axis equal;grid on
xlabel('x');ylabel('y');title 'Unit Circle z plane'
w=(z-1)./(z+i);
figure (2)
plot(real(w),imag(w));grid on;axis equal;
xlabel('u');ylabel('v');
title 'Image in the w plane of the unit circle for w=(z-1)/(z+i)'
z=1+i+2*z;
figure(3)
plot(real(z),imag(z));axis equal;grid on
xlabel('x');ylabel('y');title 'Circle |z-1-i|=2'
w=(z-1)./(z+i);
 figure(4);
 plot(real(w),imag(w));grid on;axis equal;
 xlabel('u');ylabel('v');
```

Section 4.2.1

```
w=solve('(w1-w2)*(w3-w)/((w1-w)*(w3-w2))=(z1-z2)*(z3-z)/((z1-z)*(z3-z2))','w');
zvalues= input(' z1 z2 z3 as row vector elements')
z1=zvalues(1);
```

```
z2=zvalues(2);

z3=zvalues(3);

wvalues= input(' w1 w2 w3 as row vector elements')

w1=wvalues(1);

w2=wvalues(2);

w3=wvalues(3);

% the following line substitutes the numerical values into the formula %for
w, in place of the symbols

w=subs(w,{'w1','w2','w3','z1','z2','z3'},{w1,w2,w3,z1,z2,z3});

simplify(w)

factor(w)
```

```
% the line below solves for w(z)

clear

m=solve('w=m*(x-p)/(x-conj(p))','m');

p= input(' point in upper z plane mapped to w =0   ');

x=input('point on real axis mapped to a place on unit circle   ');

w=input('image of above pt. on unit circle');

m=subs(m, {'w','x','p'},{w,x,p})

' above requires |m|=1'

 syms w z;

 w=m*(z-p)/(z-conj(p))
```

Section 4.3

```
%getting real part and imaginary part

clear

syms x y real
```

```
 z=x+i*y;
w=exp(cos(z^2))
u=real(w)
v=imag(w)
%check on harmonic behaviour
u_check=diff(u,x,2)+diff(u,y,2);
v_check=diff(v,x,2)+diff(v,y,2);
u_check=simplify(u_check)
v_check=simplify(v_check)
```

```
syms z
z=x+i*y
A0=sin(z)
A0r=real(A0)
A1=sin(1/z)
A1r=real(A1)
A2=sin(z*sin(1/z))
A2r=real(A2)
A3=sin(z^2)
A3r=real(A3)
```

```
clear
syms x y real
syms z
z=x+i*y
A1=sin(1/z)
A1r=real(A1);
A1i=imag(A1);
```

```
A2=sin(z*(A1r+i*A1i))
A2r=real(A2)
```

```
clf
x=linspace(-4,4,100);
y=linspace (-4,4,100);
n=linspace (-2,2,5);
[x,y]=meshgrid(x,y);
z=x+i*y;
 w=-log(z-1)+log(z+1);
wr=real(w);
 [c,h]= contour(x,y,wr,n);colorbar;
   xlabel('x');ylabel('y');title('Contour Lines for Log|z+1|-Log|z-1|')
   grid
   clabel(c,h);axis([-2.5 2.5 -2.5 2.5])
```

```
clf
v=linspace(-3, 3, 100);
[x,y]=meshgrid(v);
f=(x.^2-y.^2);
 figure (1)
 n=linspace(-2,2,5);
[c,h]=contour(x,y,f,n);axis equal
clabel(c,h);colorbar
hold on
%following uses coarse spacing for gradient
%which is desirable
v=linspace(-3, 3, 10);
[x,y]=meshgrid(v);
```

```
f=x.^2-y.^2;
 [gx,gy]=gradient(f);
 quiver(x,y,gx,gy);
 title('contour lines and gradient for x^2-y^2')
```

Section 4.4.1

```
clear all
syms y  x  v ux uy v1(y) v2(x) c1(x) c2(y) difference;
u=y+x^2-y^2+x;
ux=diff(u,x);
uy=diff(u,y);
%below solves du/dx=dv/dy
v1=dsolve(ux==diff(v1),y)+c1(x);
%below solves du/dy=-dv/dx
v2=dsolve(uy==-diff(v2),x)+c2(y);
v1=simplify(v1)
v2=simplify(v2)
difference=v1-v2;
difference=simplify(difference)
```

```
clf;clear
x=linspace(-6,6,200);
y=linspace (-6,6,200);
n=linspace (-2,2,9);
[x,y]=meshgrid(x,y);
z=x+i*y;
 w=z+1./z;
 wi=imag(w);
 [c,h]= contour(x,y,wi,n);colorbar;
```

```
 axis equal
 xlabel('x');ylabel('y');   grid
   clabel(c,h);
```

---

section 4.5.1

```
x=linspace (-7,7,100);
y=linspace (0, 7,100);
[x,y]=meshgrid(x,y);
z=x+i*y;
phi=-i*log(z)/(pi);
phir=real(phi);
phim=imag(phi)
values=linspace(0 ,1,6);
[g,h]=contour (x,y,phir,values,'linewidth',2);
clabel(g,h);
grid
hold
xlabel('u');ylabel('v');axis equal
x=linspace (-7,7,100);
y=linspace (0, 7,100);
[x,y]=meshgrid(x,y);
z=x+i*y;
phi=-i*log(z)/(pi);
phir=real(phi);
phim=imag(phi)
values=linspace(0 ,1,6);
[g,h]=contour (x,y,phir,values,'linewidth',2);
clabel(g,h);
```

```
grid

hold

xlabel('u');ylabel('v');axis equal

title('The Contour Lines for Real(-i/\pi Log(w)) in Upper Half Plane')
```

```
clf

clear

u=linspace (-4,4,100);

v=linspace (0, 4,100);

[u,v]=meshgrid(u,v);

w=u+i*v;

phi=-i*log(w-1)/(pi)+i*log(w+1)/(pi);

phir=real(phi);

phim=imag(phi);

[g,h]=contour (u,v,phir,[ .2 .3 .4 .5],'linewidth',2);

clabel(g,h);

xlabel('u');ylabel('v');axis equal

title('The Contour Lines for Re(-i/\pi Log(w-1)+i/\pi Log(w+1)) in Upper Half
Plane')

grid
```

Section 4.5.2

```
clf

x=linspace (0,3,100);

y=linspace (0,3,100);

[x,y]=meshgrid(x,y);

z=x+i*y;

phi=-i*log(z.^2-1)/(pi)+i*log(z.^2+1)/(pi);
```

```
phir=real(phi);
phim=imag(phi);
values=[ 0 .1 .2 .3 .6 .9 1];
[g,h]=contour (x,y,phir,values,'linewidth',2);
clabel(g,h);
xlabel('x');ylabel('y');axis equal
title('The Equipotentials for Fig 4.5-6')
grid
```

```
clf
r=linspace (0,3,100);
theta=linspace (0,pi/2,100);
[r,theta]=meshgrid(r,theta);
z=r.*exp(i*theta);
phi=-1i*log(z.^2-1)/(pi)+1i*log(z.^2+1)/(pi);
phir=real(phi);
phim=imag(phi);
values=[ 0 .1 .2 .3 .6 .9 1];
x=real(z);y=imag(z);
[g,h]=contour (x,y,phir,values,'linewidth',2);
clabel(g,h);
xlabel('x');ylabel('y');axis equal
grid
```

Section 4.5.3

```
clf
x=linspace (-1.1,1.1,100);
y=linspace (-.1, 3,100);
```

```
[x,y]=meshgrid(x,y);
z=x+i*y;
phi=2/pi*sin(pi/2*z);% this is the complex potential
phim=imag(phi);
values=[0 .5 1 3 5 10 15 20];
[g,h]=contour (x,y,phim,values,'linewidth',2);
clabel(g,h);
xlabel('x');ylabel('y');xlim([-2 2]);ylim=([0 3]);
 axis equal
title('Streamlines for Flow Into a Closed Channel')
grid
```

Section 4.5.4

```
clf
x=linspace (-3,3,100);
y=linspace (-.1,5,100);
[x,y]=meshgrid(x,y);
z=x+i*y;
phi=log((z+i)./(z-i));
phir=real(phi);
values=[0 .1 .5 1 2 5];
[g,h]=contour (x,y,phir,values,'linewidth',2);hold on
clabel(g,h);
xlabel('x');ylabel('y');axis equal
title('The Equipotentials for Line Source Above a Ground Plane')
plot(0,1,'o')
```

```
grid
```

```
clf;
clear;
x = linspace(-3, 3, 100);
y = linspace( 0, 5, 100);
[x, y] = meshgrid(x, y);
z = x + i*y;
phi    = log((z+i)./(z-i));
phim   = imag(phi);
del=.05
phim(abs(x) < del & y < 1) = nan; % added this line
values = [-2 -1, -.5,-.3 0,.3, .5, 1 2];
[g,h]  = contour(x,y,phim,values,'linewidth',2);
hold on
clabel(g, h);
xlabel('x');
ylabel('y');
axis equal
title('The Streamlines for a Line Source Above a Ground Plane')
plot(0, 1, 'o')
grid
```

Section 5.2.2

```
a=zeros(1,11);
%above creates a row vector of zeros;
%there are 11 elements in the row vector
%because of powers 0 thru 10
```

```
a(1,1)=(1-i);%coeff of z^10
a(1,11)=2; %coeff of z^0
a(1,8)=(1+i)%coeff of z^3
syms z
check=poly2sym(a,z)
% above checks that we have correct polynomial
rts=roots(a)%gives the roots
```

Section 5.2.4

```
syms w z
format short
a= [1 (-1) (-1+i) (1-2*i)  i]
% note the use of parens in the above to avoid errors
Roots_of_poly=roots( a)
% above gives roots of given polynomial
k=1;
while k<6
    ('the order of the derivative')
    k
    a=polyder(a)
    rts=roots(a)
    A=poly2sym(a,z)
    % the above is the polynomial in z
    k=k+1;
end
```

Section 5.3

```
theta=linspace(0,2*pi,1000);
```

```
%above establishes angles on a circle
z=3/2*exp(i*theta);
%above,are  the points on a circle, radius 3/2,center origin
w=z.^4+i*z.^3+4+i;
plot(w);grid
coeffs=[1 i 0 0 (4+i)]
%above are coeffs of the polynomial we are given
the_roots=roots(coeffs)
abs_values=abs(the_roots)
```

```
% use of rectangular contour, princ of the argument
cntr=input('real part center of rectangle')
cnti=input('imag part center of rectangle')
ht=input('height of rectangle')
wdth=input('width of rectangle')
x1=linspace(-wdth/2,wdth/2,1000);x2=linspace(wdth/2,-wdth/2,1000);
y1=linspace(-ht/2,ht/2,1000);  y2=linspace(ht/2,-ht/2,1000);
%the preceding places 1000 points on the 4 sides of rectangle
zbot=x1-i*ht/2;%z on bottom  side of rectangle;
ztop=x2+i*ht/2;%z on top of rectangle
zrt= wdth/2+i*y1;%z on right side of rectangle
zlft=-wdth/2+i*y2;%z on left side of rectangle.
z=[zbot zrt ztop zlft]+cntr+i*cnti;%this centers the rectangle where desired
%enter the function w(z) below
w=sin(z)-sinh(z);
plot(w,'linewidth',2);grid;hold on
plot(0,0,'*')
```

```
x1=linspace(-wdth/2,wdth/2,4);x2=linspace(wdth/2,-wdth/2,4);
```

```
y1=linspace(-ht/2,ht/2,4);  y2=linspace(ht/2,-ht/2,4);
zbot=x1-i*ht/2;%z on bottom  side of rectangle;
ztop=x2+i*ht/2;%z on top of rectangle
zrt= wdth/2+i*y1;%z on right side of rectangle
zlft=-wdth/2+i*y2;%z on left side of rectangle.
z=[zbot zrt ztop zlft]+cntr+i*cnti;


w=(sin(z)-sinh(z)).^2;
('z     and          w')
disp([z.' w.'])
```

Section 5.4

```
clf;clear all
R=1.2;
y=linspace(R,-R,1000);
theta=linspace(-pi/2,pi/2,1000);
z1=i*y;%gives points on the diameter
z2=R*exp(i*theta);%gives points on the arc
w1=9*z1.^5+z1.^4+2;% the function at points on the diameter
  w2=9*z2.^5+z2.^4+2;%the function at points on the arc
u1=real(w1);v1=imag(w1);u2=real(w2);v2=imag(w2);
plot(u1,v1);hold;
plot(u2,v2); grid;axis equal
plot(0,0,'o');%puts a o at the origin
roots([9 1 0 0 0 2])
%the above tells us where the roots are
```

Chapter 6

Section 6.3.2

```
syms t tau
  assume(t,'real')
 assume(tau,'real')
 %f(t)=1/(t^2+1)and g(t)=1/(t^2+9)our funcs of t.
 ftau=1/((tau)^2+1);
gtau=1/((t-tau)^2+9);
h=ftau*gtau;
the_conv=int(h,tau, -inf,inf)
% the above is the convolution of two functions
syms w
assume(w,'real')
Fourier_of_conv=fourier(the_conv,t,w)
%the above is the Fourier transform the convolution of the two
functions
F=fourier(ftau,tau,w)
G=fourier(1/(t^2+9),t,w)
prod_transforms=simplify(F*G)
%the above is the product of the Fourier transforms of each function
check_me=prod_transforms-Fourier_of_conv
% the above should work out to zero, it checks to see that the
%product
%of the transforms is the transform of their convolution.
```

Section 6.5.1

```
syms t
g=input('g is equal to   ')
G=fourier (g);
syms pi
```

```
h=1/(pi*t);
H=fourier(h);
C=ifourier (G*H);
assume(x,'real')
The_Hilbert=simplify(C)
```

```
function [output ] = hlbtrn( func )
% this gives you the hilbert transform of the
%function that is the argument of hlbtrn.
%the input is a function of t, the output is a function of x.
  syms  t w h x
   H=@(f)fourier(f(t),t,w);%this creates a nested  in line function
 % of the function f(t)
f(t)=func; % this establishes that f(t) is the argument you fed in
%when calling hlbtrn
h=H(f);% this  evaluates the Fourier transform
% of the input f(t)or func
%(2*heaviside(-w) - 1)*i is the Fourier transform of 1/(pi*t)
output=simplify(ifourier(h*(2*heaviside(-w) - 1)*i));
end
% the end is needed because you have a nested function within the one we
% created
```

Chapter 7, Coda

```
q=1;
while q>0
c=input('the complex value of c to be checked')
tic
```

```
nmax=10e6;
% this is the max number of interations, but
%the iterations will stop if |zn|>2
n=1;
z=0;
while(abs(z)<=2)


 z=z^2+c;
    n=n+1;
    if n>nmax
        break
    end
end
if n<=nmax
disp('the value of zn that causes early termination if n<nmax')
    z
end
  disp( 'number of iterations used')
   n-1
   if n-1==nmax
      disp( 'the number c is in the set')
   else
      disp( 'the number c is not in the set')
   end
toc
   q=input('q is neg to stop')
end
```

```
x=[0:100];%x assumes the integer values from 0 to 100

y=(1-(-1).^x)/2;%at even x values this is zero,at odd x is one

r=(.9).^((x-1)/2);%at x=1,3,5,...gives values of 1,.9,.9^2,,.9^3....

y=y.*r;  %multiplies y at the peaks by 1,.9,.9^2,.9^3 etc

xx=(1-sqrt(.9).^(x+1))/(1-sqrt(.9));% this is the partial sum of the series

%1+a+a^2+a^2...,a=sqrt(.9) each term in the sum is sqrt(.9) times the one
preceding it, alternate partial sums differ by .9

 plot(xx,y,'linewidth',2);grid
```

```
q=1;

while q>0

xo=input('x coord center of box=')

yo=input('y coord center of box=')

dx=input('half width of box, x direction=')

dy=input('half width of box, y direction=')

nx=input('number of x divisions’)

ny=input('number of y divisions')

tic

cr=linspace(xo-dx,xo+dx,nx);

ci=linspace(yo-dy,yo+dy,ny);

[Cr,Ci]=meshgrid(cr,ci);

c=Cr+i*Ci;%this creates a grid of complex numbers for c

nmax=200;% we use 200 iterations of the recursion relation

j=1;

z=zeros(size(c));%this starts off z, for each value of c, at the value zero

while j<=nmax;%iterates the expression below nmax times;%200 here;
```

```
z=z.*z+c;

j=j+1;

end

ck=abs(z)<=2;%puts a symbol of 1 in the matrix where |z|<=2 ; otherwise puts
%0

dk=1.0*ck;%converts symbolic elements to numerical in above matrix.

p=pcolor(cr,ci,dk);

set(p,'EdgeColor','none');colormap(gray);grid;%use gray for a black and white
%picture

set(gca,'layer','top'); %required if you hope to see the grid

axis([xo-dx xo+dx yo-dy yo+dy])

q=input('negative q to stop')

end
```

```
clf;clear

q=1;

while q>0

xo=input('x coord center of box=')

dx=input('half width of box, x direction=')

yo=input('y coord center of box=')

dy=input('half width of box, y direction=')

nx=input('number of x divisions')

ny=input('number of y divisions, preferably use odd=')

tic

cr=linspace(xo-dx,xo+dx,nx);

ci=linspace(yo-dy,yo+dy,ny);

[Cr,Ci]=meshgrid(cr,ci);

c=Cr+i*Ci;

figure(1)
```

```
z=zeros(size(c));
for j=1:20
    z=z.*z+c;
end
D=abs(z);
d=(D<=2);%this gives a matrix having ones for those values of c
%that are apparently in the Mandelbrot set after 20 iterations
grid;axis equal;
w=z;
for j=21:1000
    w=w.*w+c;
end
 DD=abs(w);
    dd=(DD<=2);%this gives a matrix having ones for those values of c
    %that are taken to be in the Mandelbrot set after 1000 iterations
    p=pcolor(cr,ci,d/2+dd/2);%the matrix d/2+dd/2 will have ones at those
    %values of c that are in the matrix after 1000 iterations
    % it will have the value 1/2 at those values of c that are in the
    % matrix after 20 iterations but not after 1000; these values are not
    %in the Mandelbrot set ; values of c that are eliminated after just 20
    %iterations are represented by zeros
    set(p,'EdgeColor','none')
    colormap(hot);hold on
    grid on;axis equal
    xlim([xo-dx xo+dx])
    ylim([yo-dy yo+dy])
    set(gca,'layer','top');% this allows you to see the grid
    toc
```

```
   q=input('choose a negative q to stop')
end
```

```
clear;clf
q=1


 format long
 while q>0
xo=input('x coord center of box=')
yo=input('y coord center of box=')
dx=input('half width of box, x direction=')
dy=input('half width of box, y direction=')
%below use odd numbers for nx and ny to include center of box
%in your calculations
nx=input('number of x divisions')
ny=nx; %this is numberof y divisions
% the two lines below give the real and imaginary parts of c.
tic
cr=linspace(xo-dx,xo+dx,nx);
ci=linspace(yo-dy,yo+dy,ny);
[Cr,Ci]=meshgrid(cr,ci);
c=Cr+i*Ci;%this creates a grid of complex numbers for c
nmax=1000;% we use 1000 iterations of the recursion relation
j=1;
z=zeros(size(c));%this starts off z, for each value of c, at the value zero
while j<=nmax;%iterates the expression below nmax times;
z=z.*z+c;
j=j+1;
```

```
end
ck=abs(z)<=2;%puts a symbol of 1 in the matrix where |z|<=2 ; otherwise puts
%0
dk=1.0*ck;%converts symbolic elements to numerical in above matrix.
[rows,cols,vals] = find(dk);%this finds the nonzero elements in dk


for k=1:length(rows)
    locations=c(rows(k),cols(k))
    figure(1)
    plot(locations,'k.'); hold on
    %the above is optional and will plot the points
    %you just found as dots


    %the "locations" are points in the complex plane lying in the
    % mandelbrot set
    end
toc
q=input('negative q to stop')
 end
```

.