Table of Contents


```
% hw 2 problem 1
```

PART (a) Poly_interp

```
A = -1;
B = 1;

% 5 points
N = 5;
[x,y] = FuncGen(A,B,N);
[xp,yp] = FuncGen(A,B,20*N);

figure(1);
title('actual function with 5 points')
plot(xp,yp,'k',x,y,'g*')
xlabel('x')
ylabel('y')
legend('y(x)','exact')
set(gca,'fontsize', 36)

p1 = NumericalRecipes.Poly_interp(x,y,2);   % M = 2
p2 = NumericalRecipes.Poly_interp(x,y,3);   % M = 3
p3 = NumericalRecipes.Poly_interp(x,y,4);   % M = 4
p4 = NumericalRecipes.Poly_interp(x,y,5);   % M = 5
for i=1:length(xp)
    [ypi1(i),p1] = p1.interp(xp(i));
    [ypi2(i),p2] = p2.interp(xp(i));
    [ypi3(i),p3] = p3.interp(xp(i));
    [ypi4(i),p4] = p4.interp(xp(i));
end
figure(2);
plot(xp,yp,'k',x,y,'g*',xp,ypi1,xp,ypi2,xp,ypi3,xp,ypi4)
title('different order poly-interpolation vs. actual function with N=5')
xlabel('x')
ylabel('y')
legend('y(x)','exact','M=2','M=3','M=4','M=5')

figure(3);
plot(xp,yp,'k',x,y,'g*',xp,ypi1,'r')
title('M = 3rd order poly-interpolation vs. actual function with N=5')
xlabel('x')
ylabel('y')
```

```
legend('y(x)','exact','fit')

err1 = ypi1 - yp;
err2 = ypi2 - yp;
err3 = ypi3 - yp;
err4 = ypi4 - yp;
figure(4)
plot(xp,err1,xp,err2,xp,err3,xp,err4)
title('error function with different order M with N=5')
xlabel('x')
ylabel('error')
legend('M=2','M=3','M=4','M=5')

% Comment for N = 5
% order cannot be 1 since it's M-1 so M has to take on values >= 2
% order also cannot exceed the number of points N
% when M = 1, we get the best fit from the 5 points, as we can see in
% figure 2 and 4 (especially the error function)


% 20 points
N2 = 20;
[x2,y2] = FuncGen(A,B,N2);
[xp2,yp2] = FuncGen(A,B,20*N2);

p5 = NumericalRecipes.Poly_interp(x2,y2,3);   % M = 3
p6 = NumericalRecipes.Poly_interp(x2,y2,5);   % M = 5
p7 = NumericalRecipes.Poly_interp(x2,y2,10);   % M = 10
p8 = NumericalRecipes.Poly_interp(x2,y2,12);   % M = 12
p9 = NumericalRecipes.Poly_interp(x2,y2,13);   % M = 13
p10 = NumericalRecipes.Poly_interp(x2,y2,15);   % M = 15
for i=1:length(xp2)
    [ypi5(i),p5] = p5.interp(xp2(i));
    [ypi6(i),p6] = p6.interp(xp2(i));
    [ypi7(i),p7] = p7.interp(xp2(i));
    [ypi8(i),p8] = p8.interp(xp2(i));
    [ypi9(i),p9] = p9.interp(xp2(i));
    [ypi10(i),p10] = p10.interp(xp2(i));
end
figure(5);
plot(xp2,yp2,'k',x2,y2,'g*',xp2,ypi5,xp2,ypi6,xp2,ypi7,xp2,ypi8,xp2,ypi9,xp2,ypi10)
title('different order poly-interpolation vs. actual function with N=20')
xlabel('x')
ylabel('y')
legend('y(x)','exact','M=3','M=5','M=10','M=12','M=13','M=15')

figure(6);
plot(xp2,yp2,'k',x2,y2,'g*',xp2,ypi8,'r')
title('M = 12th order poly-interpolation vs. actual function with N=20')
xlabel('x')
ylabel('y')
legend('y(x)','exact','fit')

err5 = ypi5 - yp2;
```

```
err6 = ypi6 - yp2;
err7 = ypi7 - yp2;
err8 = ypi8 - yp2;
err9 = ypi9 - yp2;
err10 = ypi10 - yp2;
figure(7)
plot(xp2,err5,xp2,err6,xp2,err7,xp2,err8,xp2,err9,xp2,err10)
title('error function with N = 20 and different order M')
xlabel('x')
ylabel('error')
legend('M=3','M=5','M=10','M=12','M=13','M=15')

figure(8)
plot(xp2,err8)
title('error function with N = 20, M = 12')
xlabel('x')
ylabel('error')

% Comment for N = 20
% when M = 12, we get the best fit from the 20 points, as we can see in
% figure 5-8. The lower orders of M does not fit very accurately, yet an
% order greater than 12, like 13, 15, 20, they have high discrepencies near
% the x = -1 and +1
```

PART (b) Coefficients

```
% I used polcof to find the coefficients of the
% interpolating polynomials. I calcuated two sets of functions:
% one with N=5 points and the other N=20 points
% the y value of the function generated from the coefficents are stored in
% ans and ans2

% then I plotted the function from the coefficients along with the exact
% function and the polynomial interpolation with the 5 or 20 points
% respectably


% 5 coefficients
[xc,yc] = FuncGen(-1,1,5);
format long
cp = NumericalRecipes.polcof(xc,yc);
% cp = NumericalRecipes.polcoe(xc,yc)
ans = cp(5);
for i = 4:1
    ans = ans .* xp + cp(i);
end

figure(9)
plot(xp,ans,xp,yp,x,y,'g*')
title('function with 5 coefficients vs. actual function')
xlabel('x')
ylabel('y')
legend('function with coefficients','y(x)','exact')
```

```
% 20 coefficients
[xc2,yc2] = FuncGen(-1,1,20);
format long
cp2 = NumericalRecipes.polcof(xc2,yc2);
ans2 = cp2(20);
for i = 19:1
    ans2 = ans2 .* xp2 + cp2(i);
end

figure(10)
plot(xp2,ans2,xp2,yp2,x2,y2,'g*')
title('function with 20 coefficients vs. actual function')
xlabel('x')
ylabel('y')
legend('function with coefficients','y(x)','exact')
```

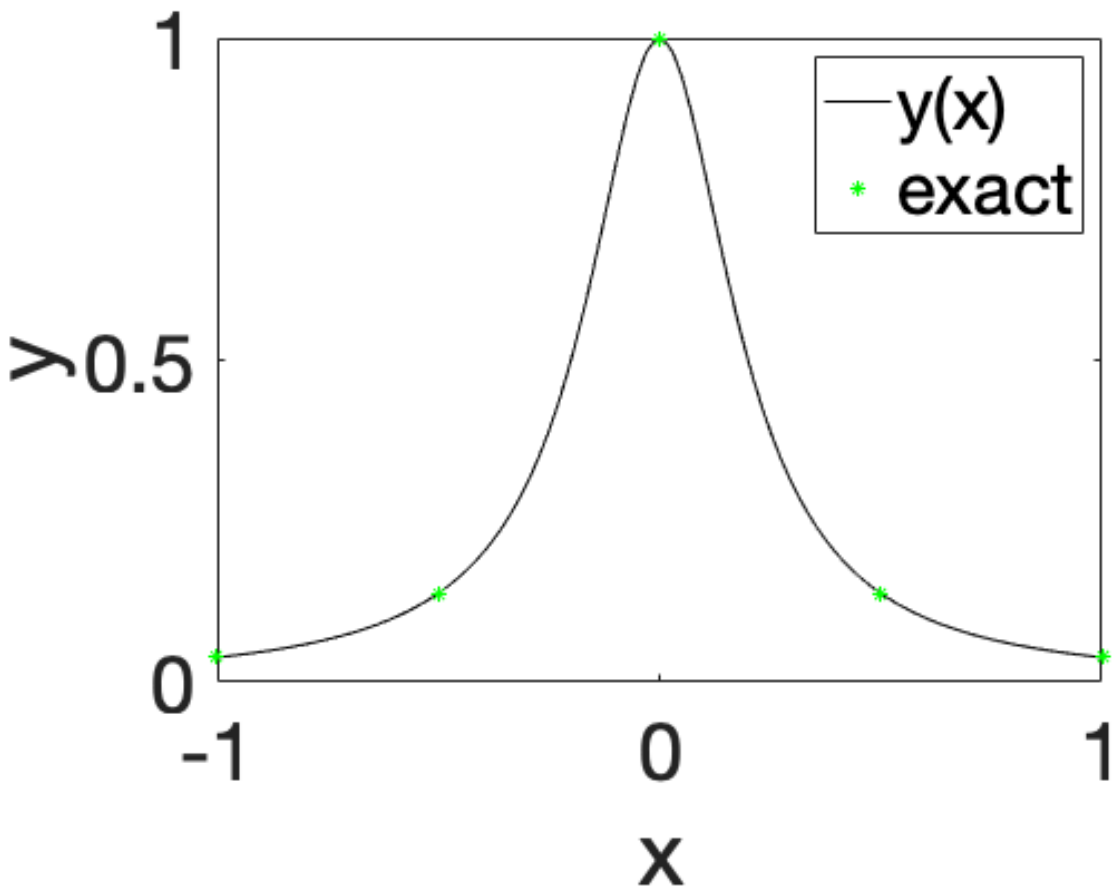
PART (c) Rat_interp

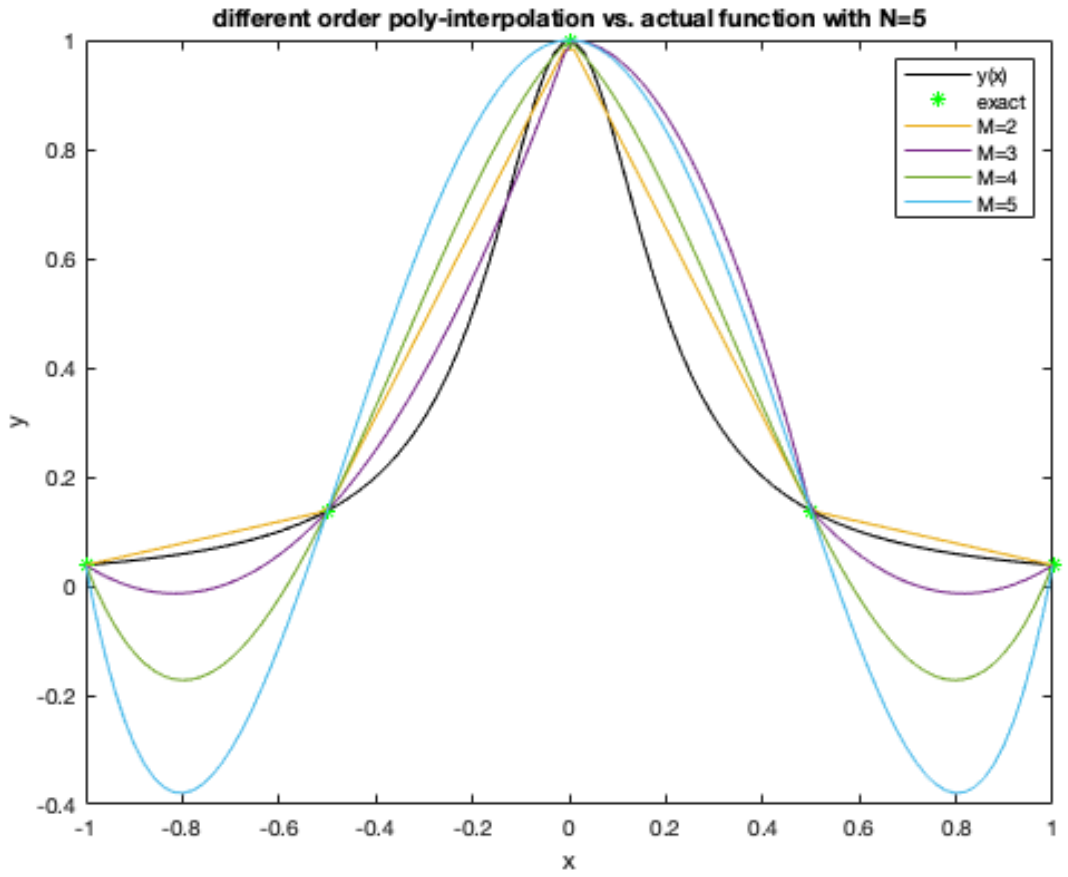
```
% 5 points
pr1 = NumericalRecipes.Rat_interp(x,y,2);   % M = 2
pr2 = NumericalRecipes.Rat_interp(x,y,3);   % M = 3
pr3 = NumericalRecipes.Rat_interp(x,y,4);   % M = 4
pr4 = NumericalRecipes.Rat_interp(x,y,5);   % M = 5


% Interpolate
for i=1:length(xp)
    [ypr1(i),pr1] = pr1.interp(xp(i));
    [ypr2(i),pr2] = pr2.interp(xp(i));
    [ypr3(i),pr3] = pr3.interp(xp(i));
    [ypr4(i),pr4] = pr4.interp(xp(i));
end
figure(11);
plot(xp,yp,'k',x,y,'g*',xp,ypr1,xp,ypr2,xp,ypr3,xp,ypr4)
title('different order rational-interpolation vs. actual function with N=5')
xlabel('x')
ylabel('y')
legend('y(x)','exact','M=2','M=3','M=4','M=5')

error1 = ypr1 - yp;
error2 = ypr2 - yp;
error3 = ypr3 - yp;
error4 = ypr4 - yp;
figure(12)
plot(xp,error1,xp,error2,xp,error3,xp,error4)
title('error function with Rat_interp N = 5')
xlabel('x')
ylabel('error')


% 20 points
```

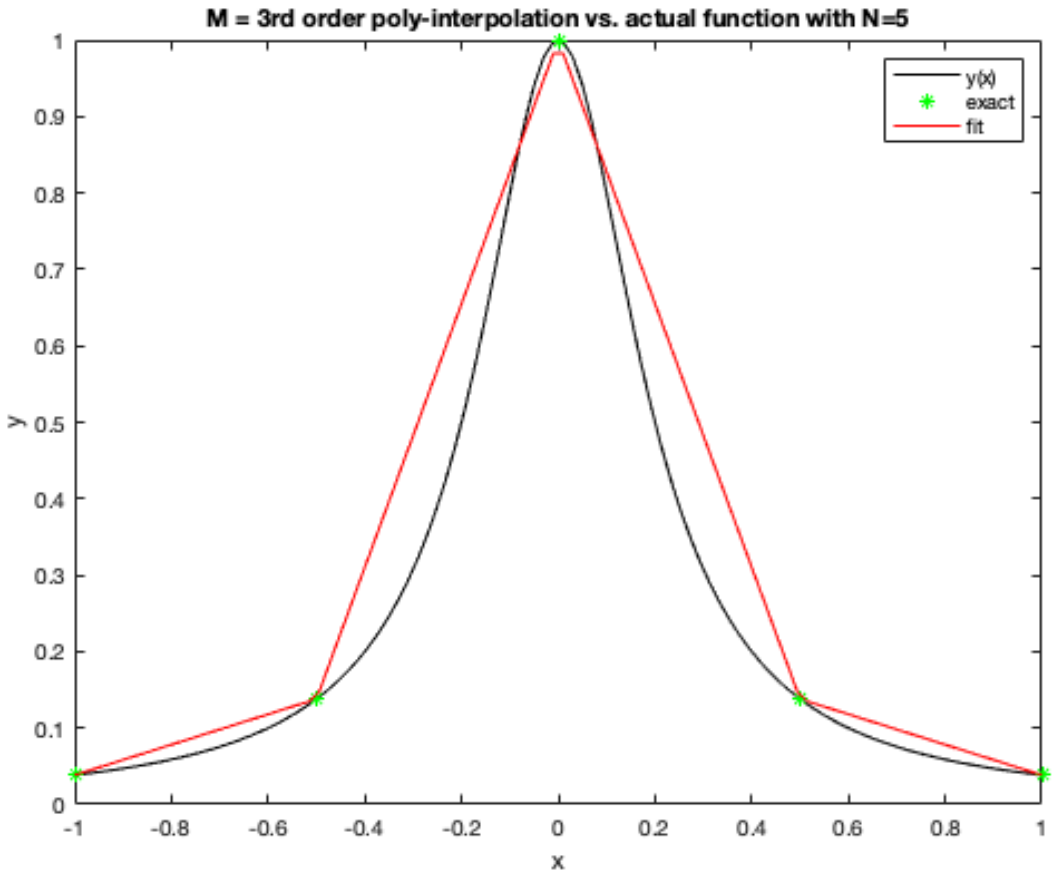
```
pr5 = NumericalRecipes.Rat_interp(x2,y2,3);   % M = 3
pr6 = NumericalRecipes.Rat_interp(x2,y2,5);   % M = 5
pr7 = NumericalRecipes.Rat_interp(x2,y2,10);   % M = 10
pr8 = NumericalRecipes.Rat_interp(x2,y2,12);   % M = 12
pr9 = NumericalRecipes.Rat_interp(x2,y2,13);   % M = 13
pr10 = NumericalRecipes.Rat_interp(x2,y2,15);   % M = 15
for i=1:length(xp2)
    [ypr5(i),pr5] = pr5.interp(xp2(i));
    [ypr6(i),pr6] = pr6.interp(xp2(i));
    [ypr7(i),pr7] = pr7.interp(xp2(i));
    [ypr8(i),pr8] = pr8.interp(xp2(i));
    [ypr9(i),pr9] = pr9.interp(xp2(i));
    [ypr10(i),pr10] = pr10.interp(xp2(i));
end
figure(13);
plot(xp2,yp2,'k',x2,y2,'g*',xp2,ypr5,xp2,ypr6,xp2,ypr7,xp2,ypr8,xp2,ypr9,xp2,ypr10)
title('different order rational-interpolation vs. actual function with N=20')
xlabel('x')
ylabel('y')
legend('y(x)','exact','M=3','M=5','M=10','M=12','M=13','M=15')

figure(14);
plot(xp2,yp2,'k',x2,y2,'g*',xp2,ypr8)
title('M = 12th order rational-interpolation vs. actual function with N=20')
xlabel('x')
ylabel('y')
legend('y(x)','exact','M=12')


error5 = ypr5 - yp2;
error6 = ypr6 - yp2;
error7 = ypr7 - yp2;
error8 = ypr8 - yp2;
error9 = ypr9 - yp2;
error10 = ypr10 - yp2;
figure(15)
plot(xp2,error5,xp2,error6,xp2,error7,xp2,error8,xp2,error9,xp2,error10)
title('error function with Rat interp N = 20')
xlabel('x')
ylabel('error')
legend('M=3','M=5','M=10','M=12','M=13','M=15')

figure(16)
plot(xp2,error8)
title('error function with Rat interp N = 20 and M = 12')
xlabel('x')
ylabel('error')


% Comment
% It's obvious to see that the function fit performs a lot better when N is
% 20 than is 5. The functions almost all overlapp with the original
% function when N is 20. This is because when you give more points, and
% hence can use higher order, the interpolation uses more points to plot
```

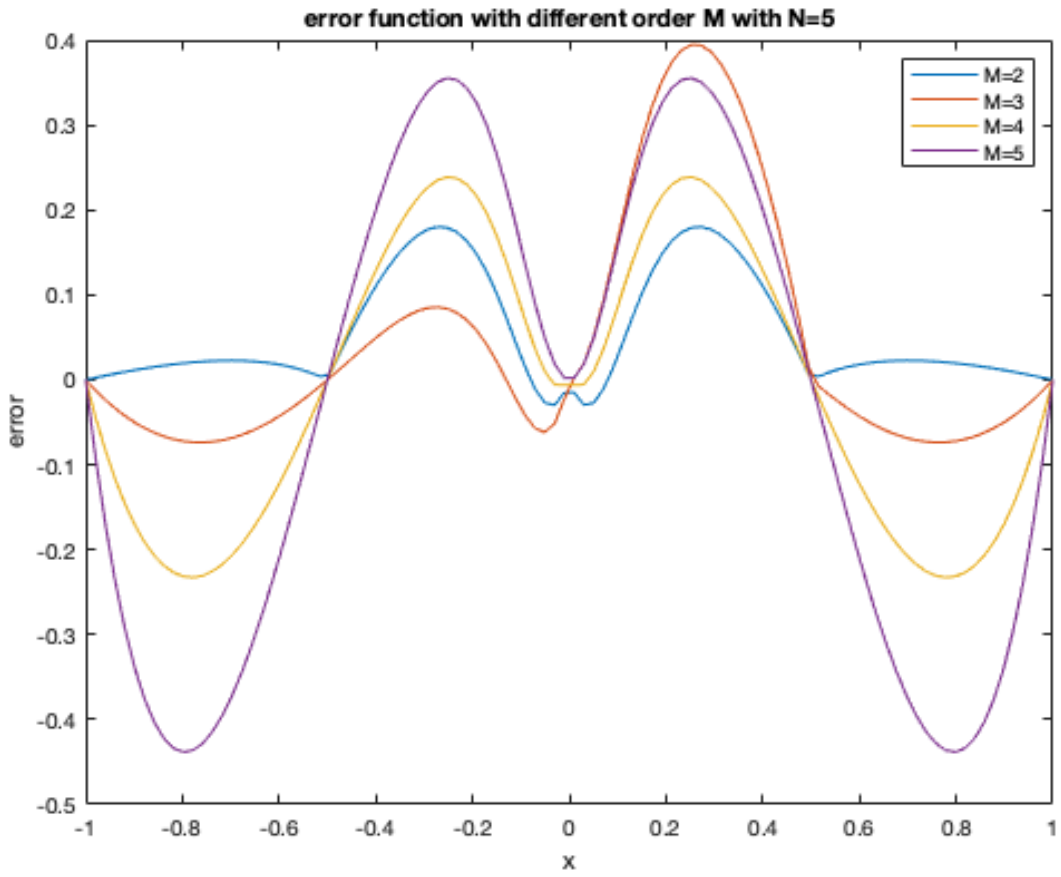
```
% more accurate function.
% Overall, rational interpolation does a much better job than poly
% interpolation, mainly because the original function is more simialr to a
% rational function than a polynomial.
% As we can see from the error function for order M = 12 and N = 20, the
% error range for polynomial is 10^-3 and the error range for rational is
% only 10^-8.
```

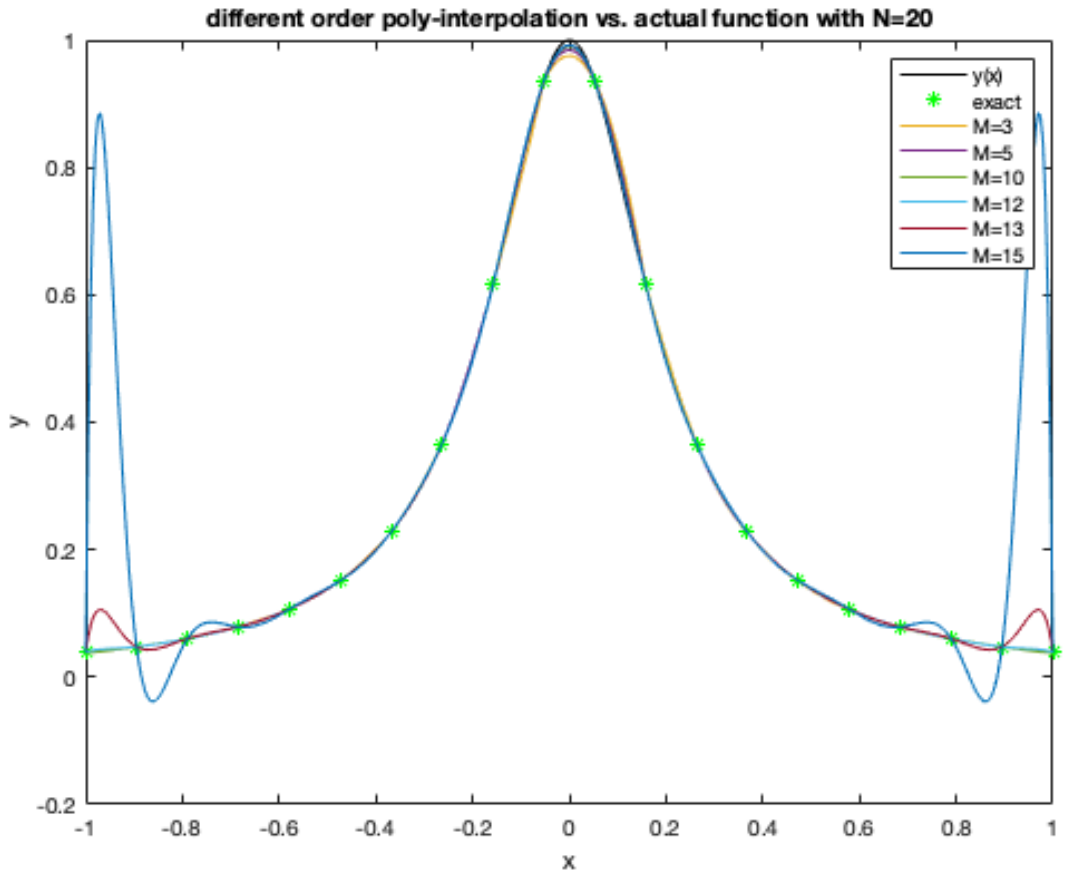
```
function [x,y] = FuncGen(a,b,n)
    x = a:(b-a)/(n-1):b;
    y = 1./(1 + 25 .* x.^2);
end
```

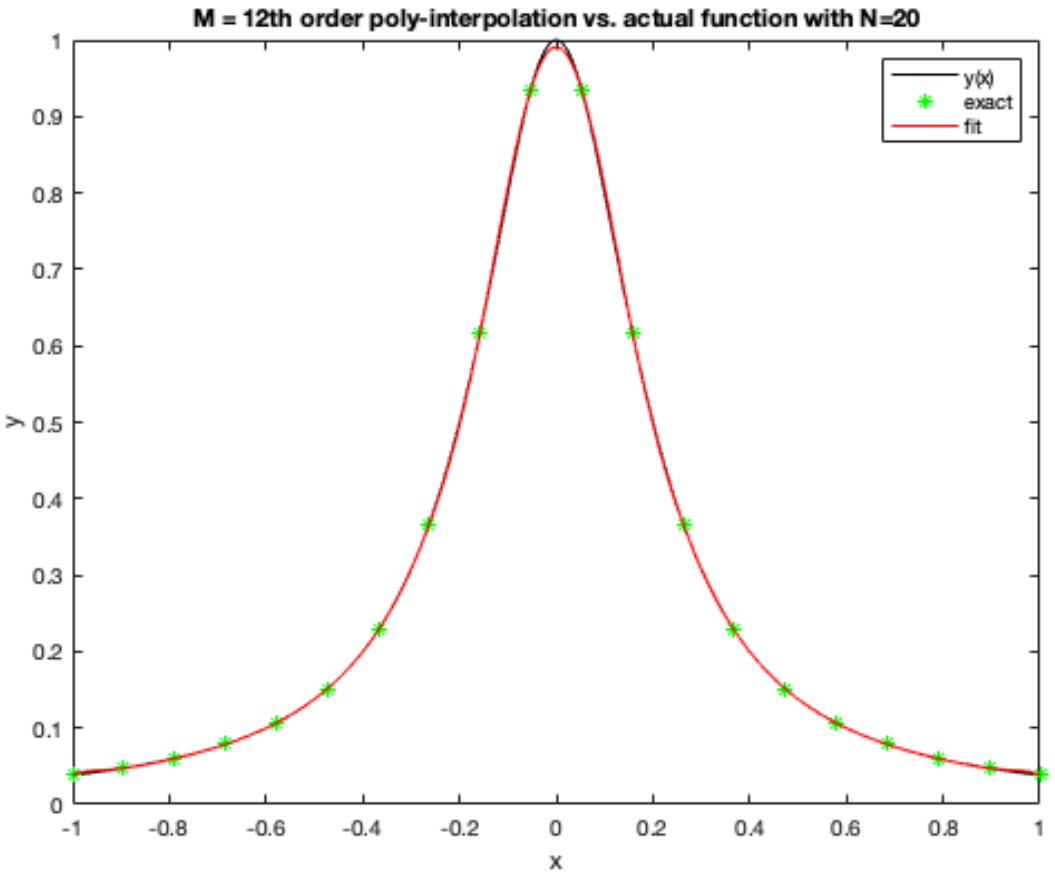


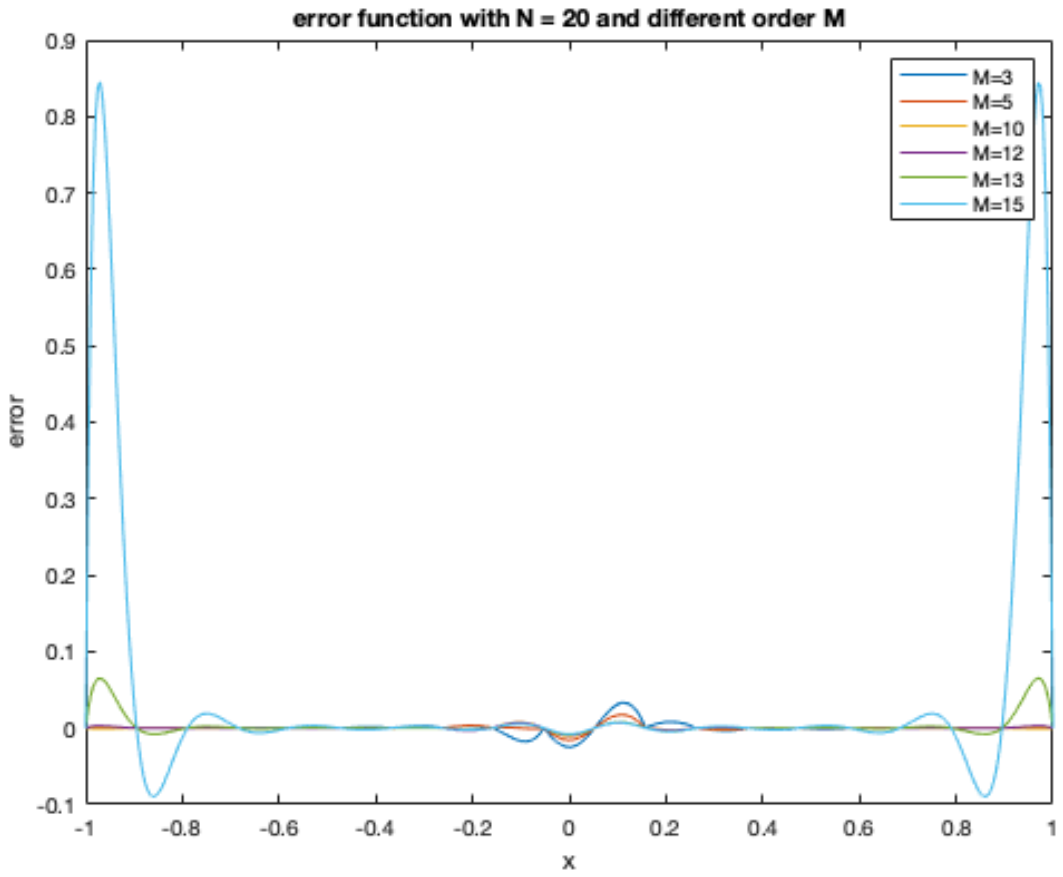

different order poly-interpolation vs. actual function with N=5
y(x)
exact
M=2
M=3
M=4
M=5
y
x
1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-1
-0.8
-0.6
-0.4
-0.2
0
0.2
0.4
0.6
0.8
1

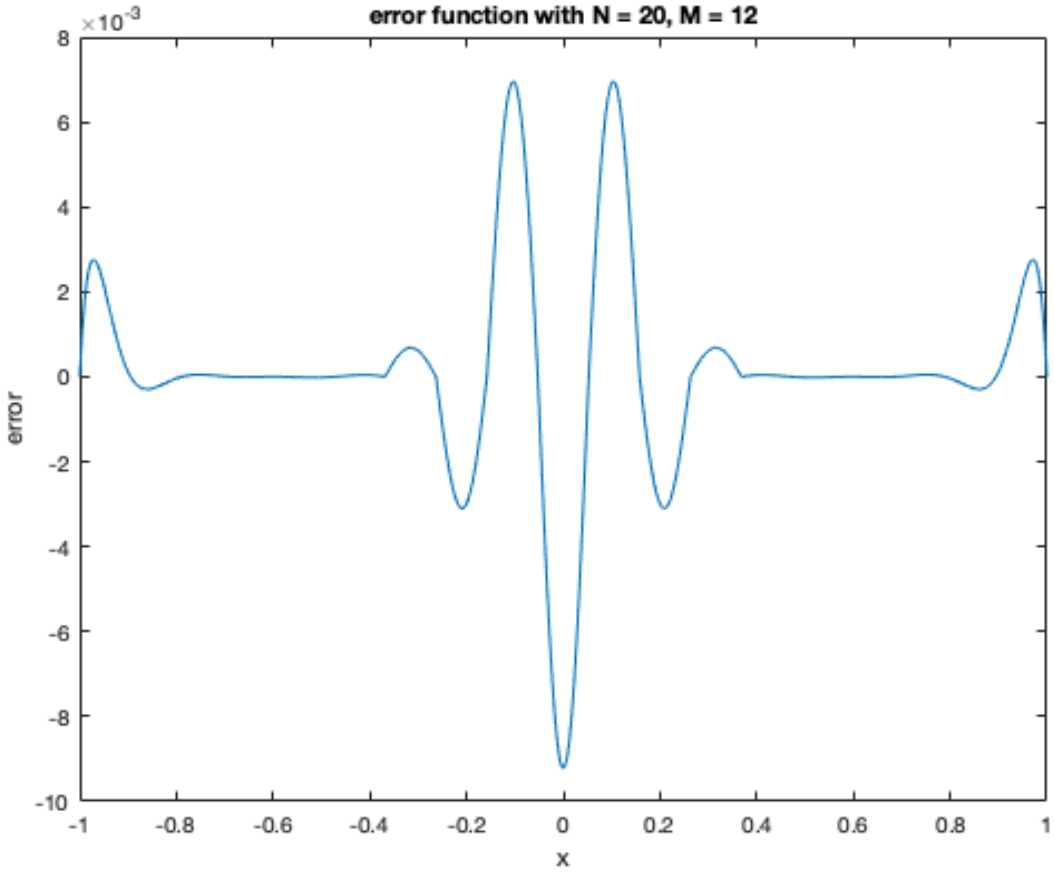

M = 3rd order poly-interpolation vs. actual function with N=5
y(x)
exact
fit
y
x
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
-1
-0.8
-0.6
-0.4
-0.2
0
0.2
0.4
0.6
0.8
1

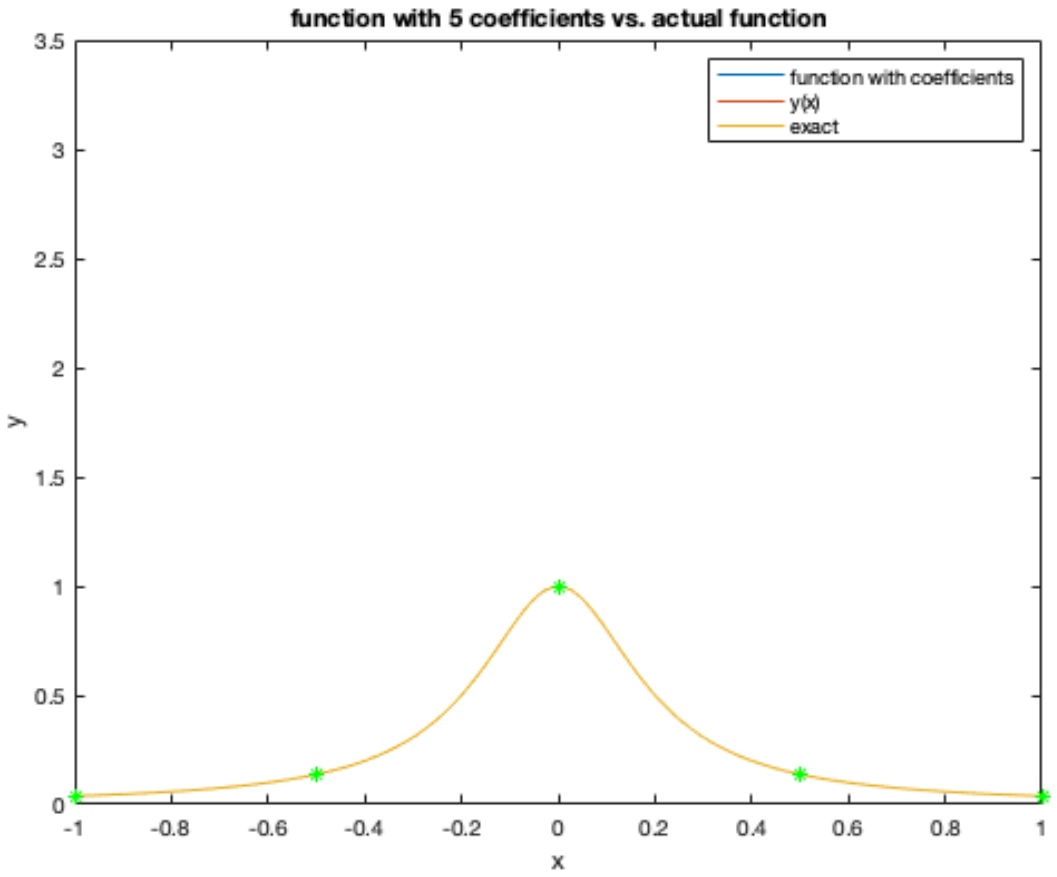

error function with different order M with N=5
M=2
M=3
M=4
M=5
error
x
0.4
0.3
0.2
0.1
0
-0.1
-0.2
-0.3
-0.4
-0.5
-1
-0.8
-0.6
-0.4
-0.2
0
0.2
0.4
0.6
0.8
1

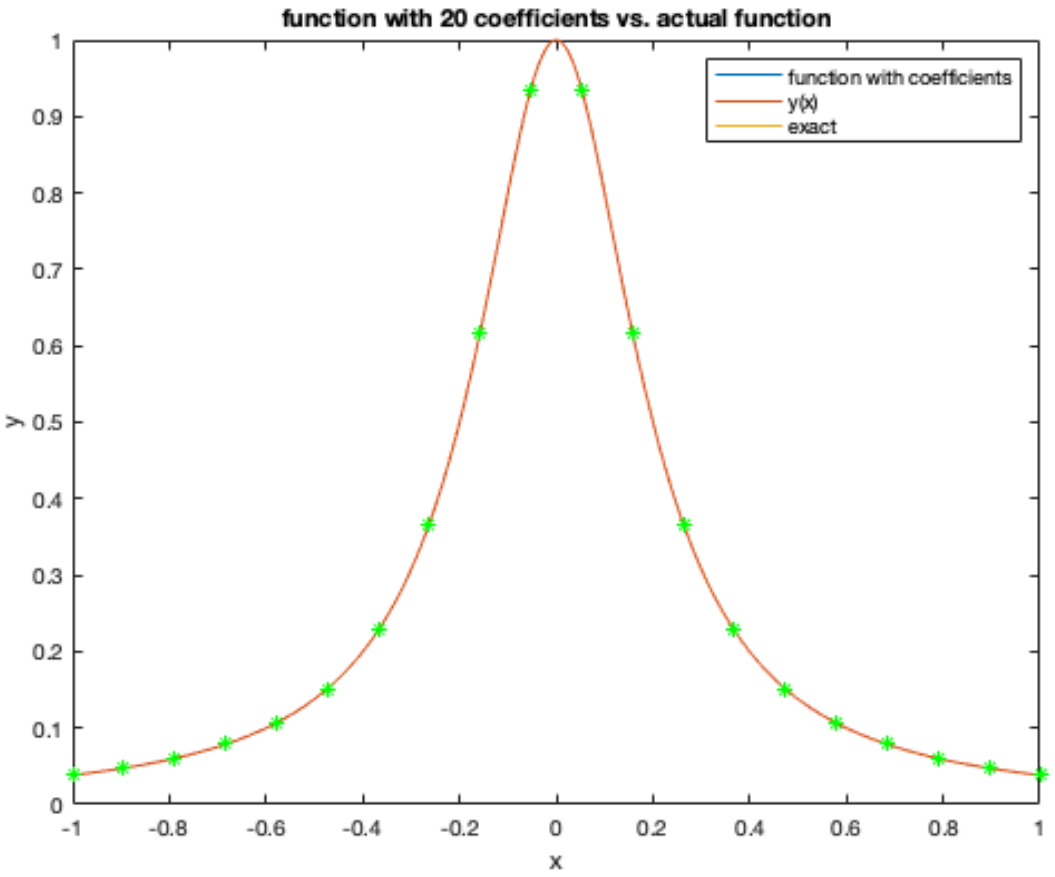

different order poly-interpolation vs. actual function with N=20
y(x)
exact
M=3
M=5
M=10
M=12
M=13
M=15
y
x
1
0.8
0.6
0.4
0.2
0
-0.2
-1
-0.8
-0.6
-0.4
-0.2
0
0.2
0.4
0.6
0.8
1

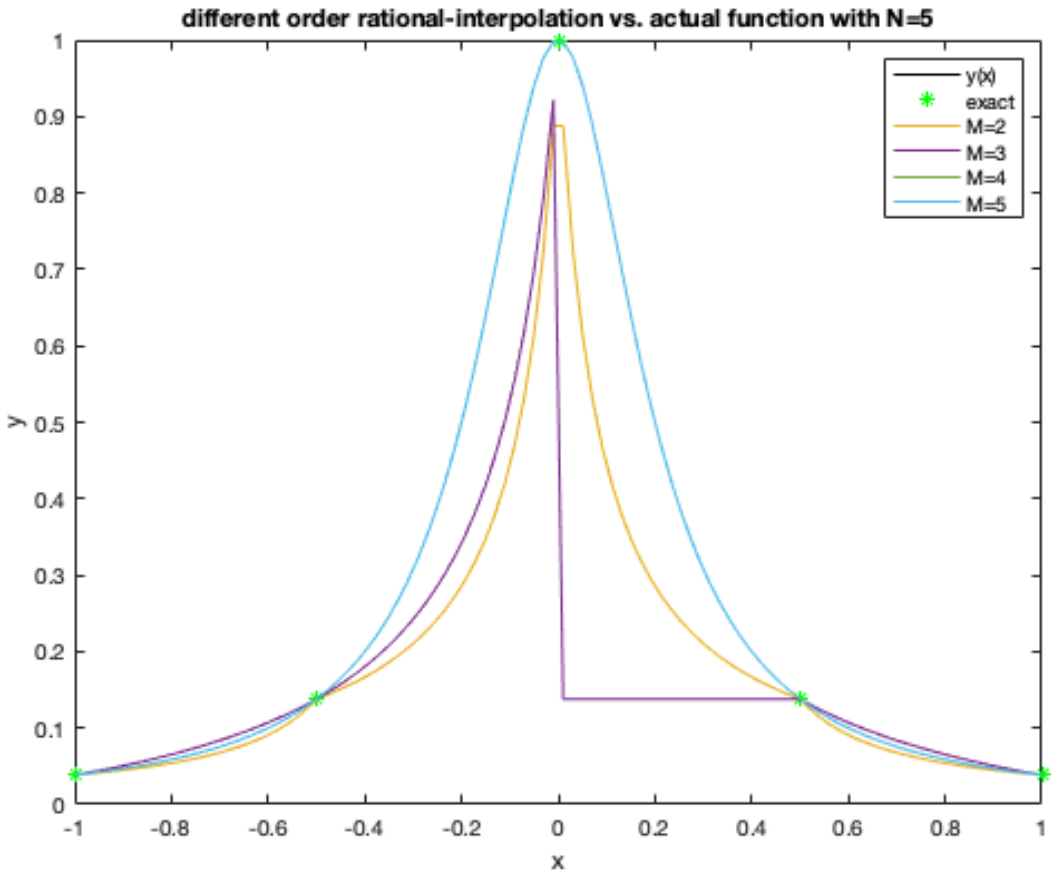

M = 12th order poly-interpolation vs. actual function with N=20
y(x)
exact
fit
y
x
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
-1
-0.8
-0.6
-0.4
-0.2
0
0.2
0.4
0.6
0.8
1

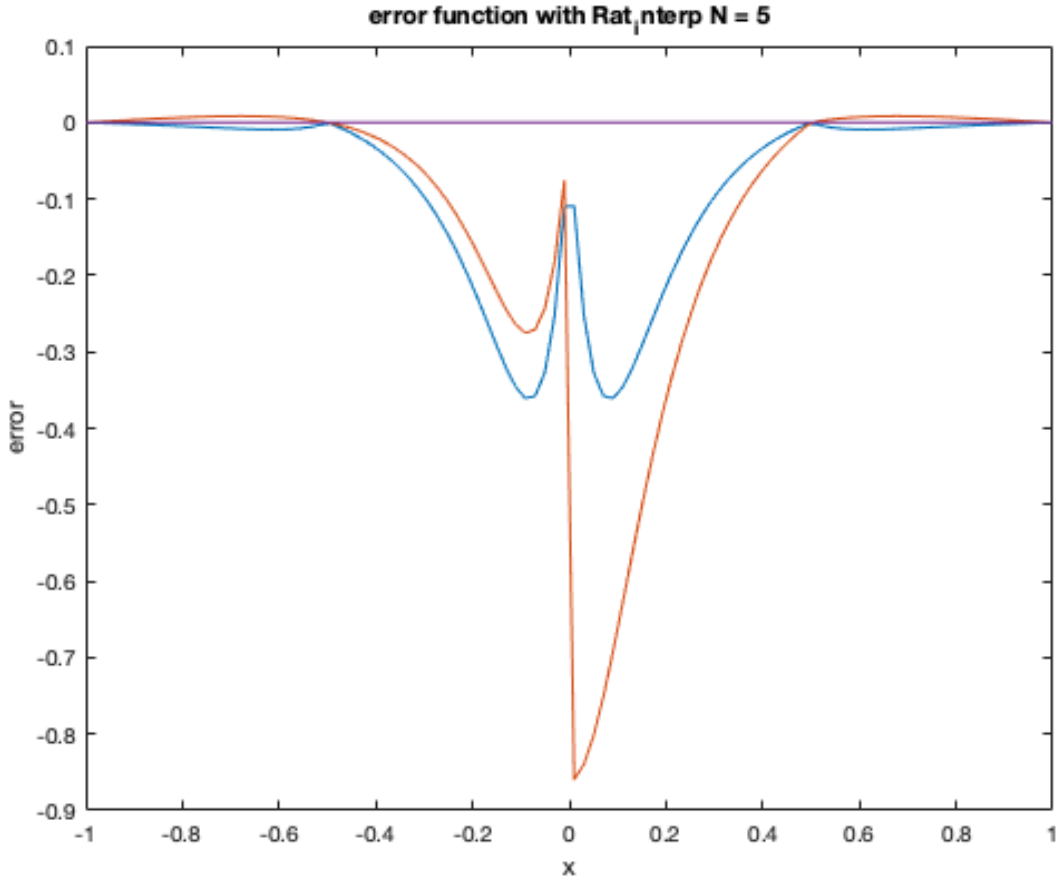

error function with N = 20 and different order M
M=3
M=5
M=10
M=12
M=13
M=15
error
x
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
-0.1
-1
-0.8
-0.6
-0.4
-0.2
0
0.2
0.4
0.6
0.8
1

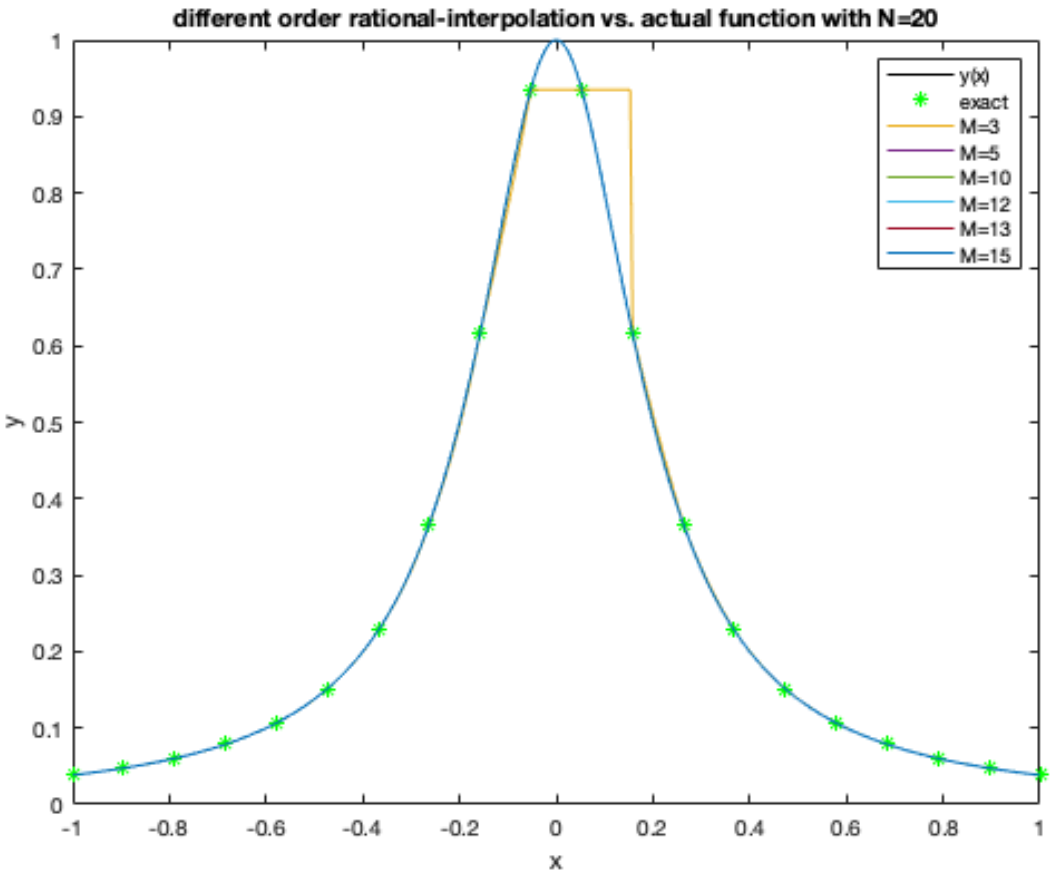

error function with N = 20, M = 12
×10^-3
8
6
4
2
0
-2
-4
-6
-8
-10
error
-1
-0.8
-0.6
-0.4
-0.2
0
0.2
0.4
0.6
0.8
1
x

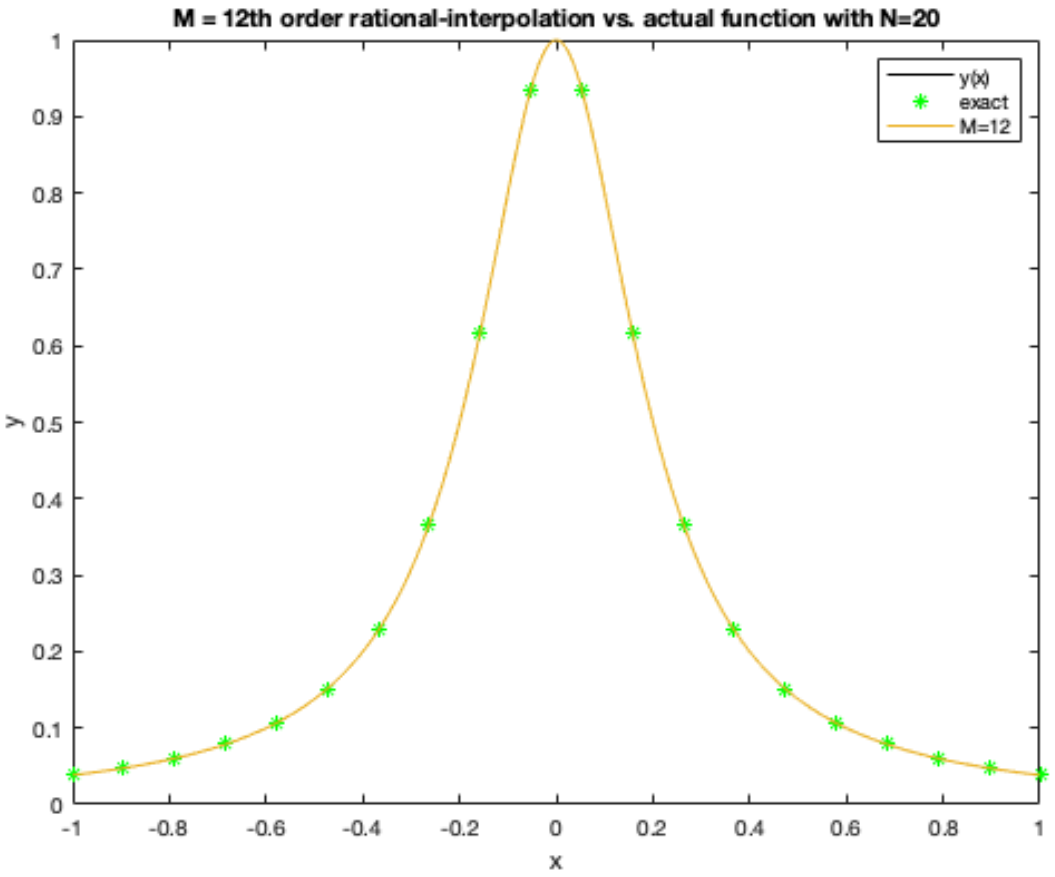

function with 5 coefficients vs. actual function
function with coefficients
y(x)
exact
y
x
3.5
3
2.5
2
1.5
1
0.5
0
-1
-0.8
-0.6
-0.4
-0.2
0
0.2
0.4
0.6
0.8
1

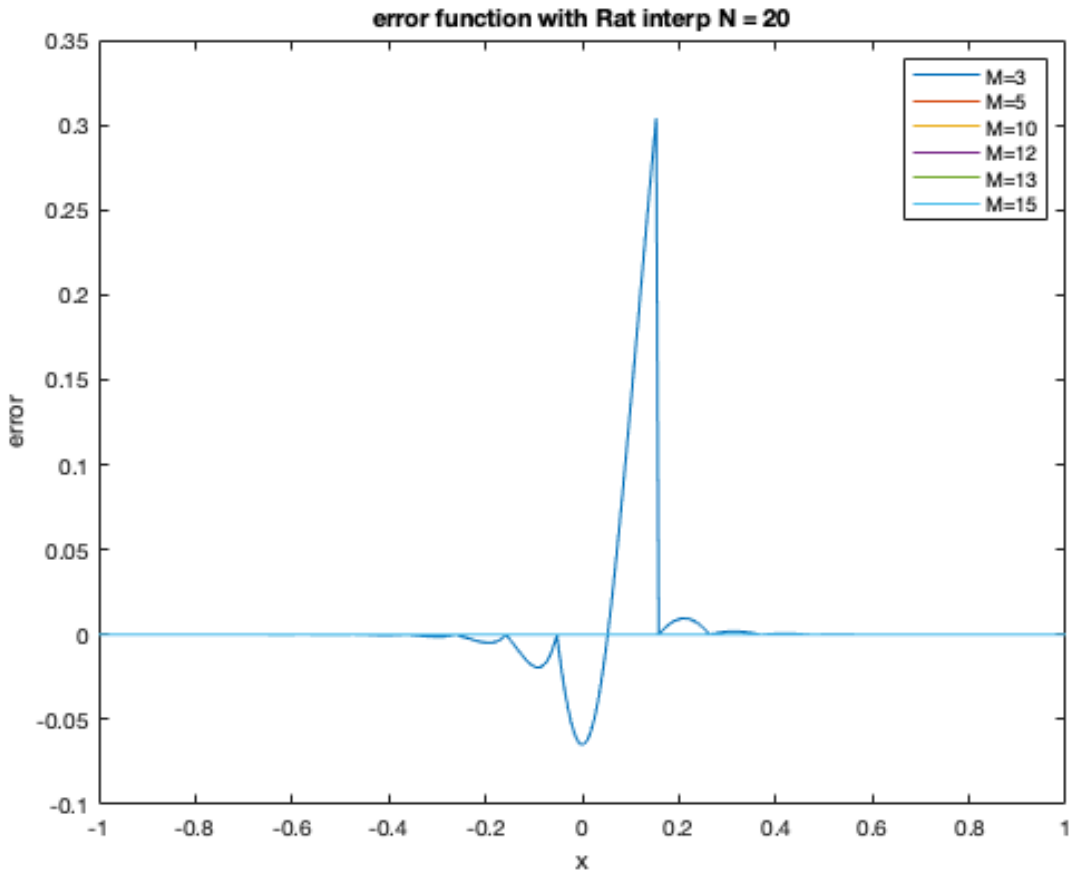

function with 20 coefficients vs. actual function
function with coefficients
y(x)
exact
y
x
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
-1
-0.8
-0.6
-0.4
-0.2
0
0.2
0.4
0.6
0.8
1

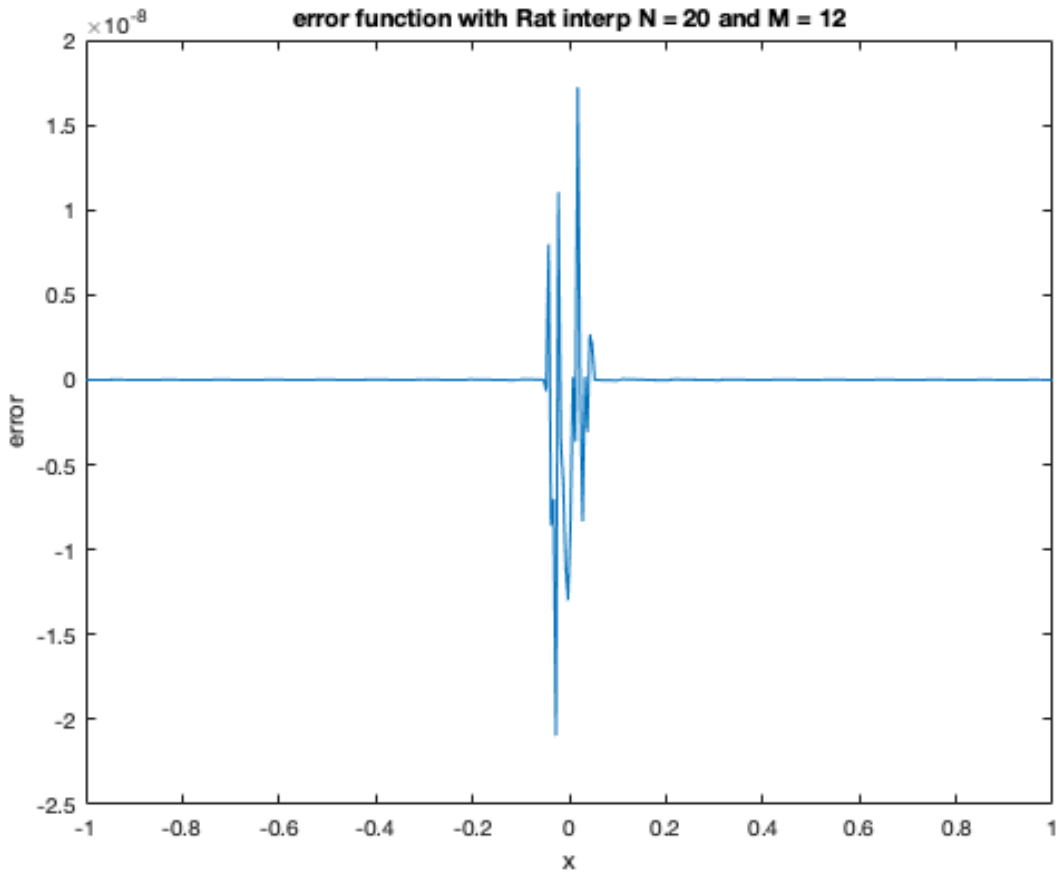

different order rational-interpolation vs. actual function with N=5
y(x)
exact
M=2
M=3
M=4
M=5
y
x
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
-1
-0.8
-0.6
-0.4
-0.2
0
0.2
0.4
0.6
0.8
1


error function with Rat_Interp N = 5
0.1
0
-0.1
-0.2
-0.3
-0.4
-0.5
-0.6
-0.7
-0.8
-0.9
error
-1
-0.8
-0.6
-0.4
-0.2
0
0.2
0.4
0.6
0.8
1
x


different order rational-interpolation vs. actual function with N=20
y(x)
exact
M=3
M=5
M=10
M=12
M=13
M=15
y
x
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
-1
-0.8
-0.6
-0.4
-0.2
0
0.2
0.4
0.6
0.8
1


M = 12th order rational-interpolation vs. actual function with N=20
y(x)
exact
M=12
y
x
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
-1
-0.8
-0.6
-0.4
-0.2
0
0.2
0.4
0.6
0.8
1


error function with Rat interp N = 20
M=3
M=5
M=10
M=12
M=13
M=15
error
x
0.35
0.3
0.25
0.2
0.15
0.1
0.05
0
-0.05
-0.1
-1
-0.8
-0.6
-0.4
-0.2
0
0.2
0.4
0.6
0.8
1



Published with MATLAB® R2022b