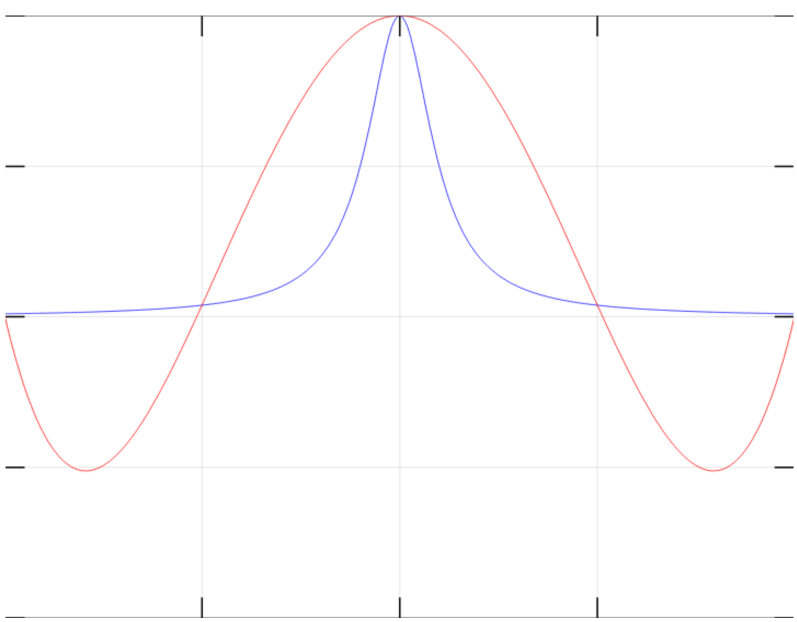
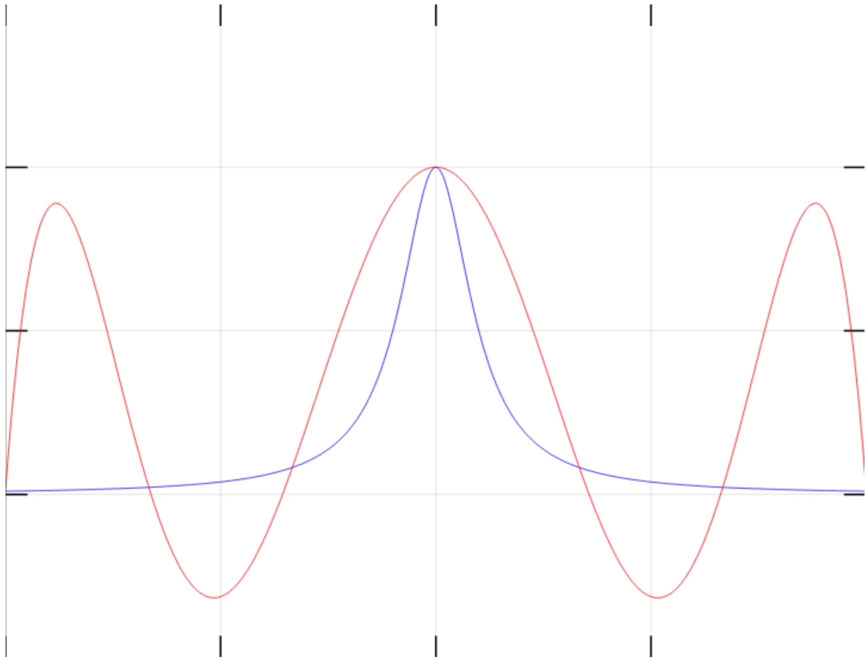
```
function d=newtondivided(f,a,b,k)
h = (b-a)/k;
t=zeros (1, k+1);
y= t;
D= zeros(k+1);

for i=1:k+1
  t(i)= a+(i-1)*h;
  y(i)= f(t(i));
endfor

tl= t;
#t πλεον για το διανυσμα της διαγωνιου
t(1)=y(1);
D(:,1)+=y';
z =zeros (1, k+1);

for j=2:k+1
    j;
    z-=z;
    for i=j:k+1
        i;
        nom= y(i)-y(i-1);
        den= tl(i)-tl(i-j+1);
        z(i)= nom/den;
        D(i, j)+= z(i)';
    endfor
    t(j)=D(j,j);
    y=z;
endfor
d=t;
endfunction


octave:4> f=@(x) 1./(1+25*x.^2)
f =

@(x) 1 ./ (1 + 25 * x .^ 2)


octave:6> newtondivided(f,-2,2,4)
D =

   0.0099        0        0        0        0
   0.0385   0.0286        0        0        0
   1.0000   0.9615   0.4665        0        0
   0.0385  -0.9615  -0.9615  -0.4760        0
   0.0099  -0.0286   0.4665   0.4760   0.2380

ans =

   9.9010e-03   2.8561e-02   4.6649e-01  -4.7601e-01   2.3800e-01
```

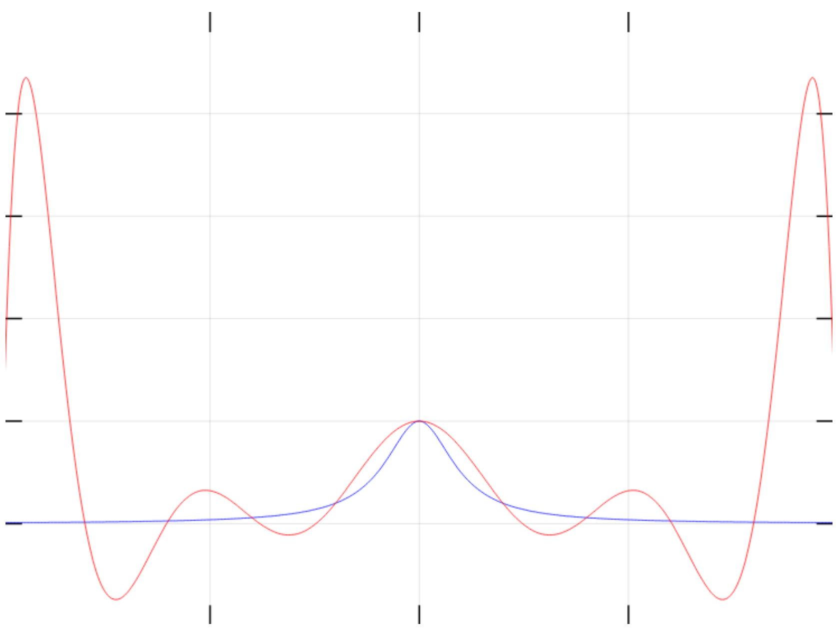
```
function plotnewton(f,a,b,k)
h = (b-a)/500;
x=zeros (1, 501);
yreal= x;
ynewton= x;

#real function
for i=1:501
  x(i)= a+(i-1)*h;
  yreal(i)= f(x(i));
endfor

#newton interp
d=newtondivided(f,a,b,k);
h2 = (b-a)/k;
t=zeros (1, k+1);
for i=1:k+1
        t(i)= a+(i-1)*h2;
endfor

#newton polynomial
for i=1:501
  product=1;
  for j=1:k
    product*=x(i)-t(j);
    ynewton(i)+=d(j+1)*product;
  endfor
endfor
ynewton+=d(1);

plot(x, yreal, "b", x, ynewton, "r"); grid

endfunction


octave:7> plotnewton(f,-2,2,4)
```



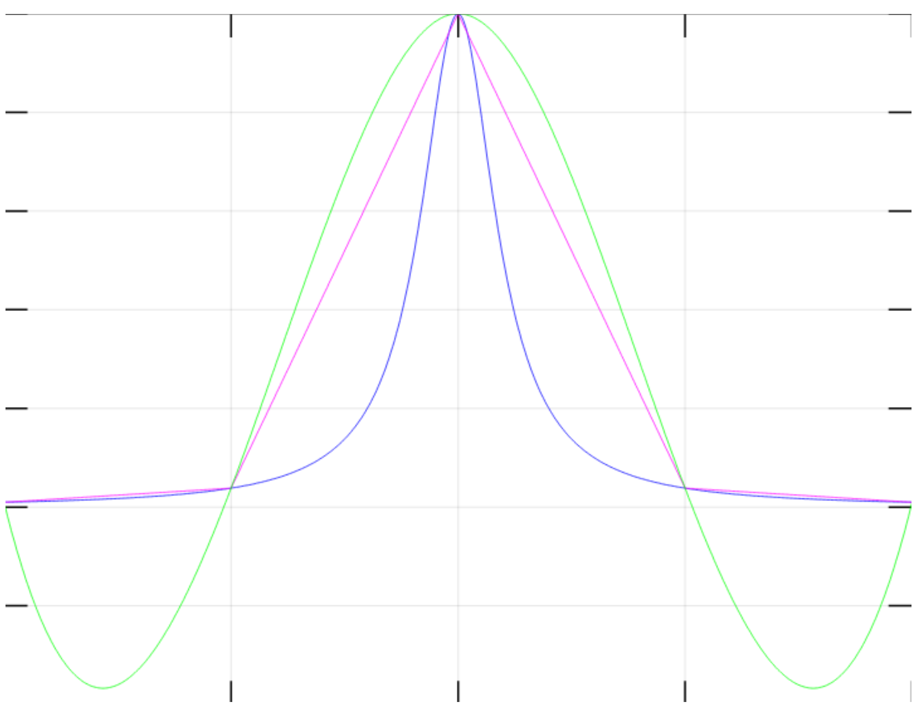
```
octave:8> plotnewton(f,-2,2,6)
```



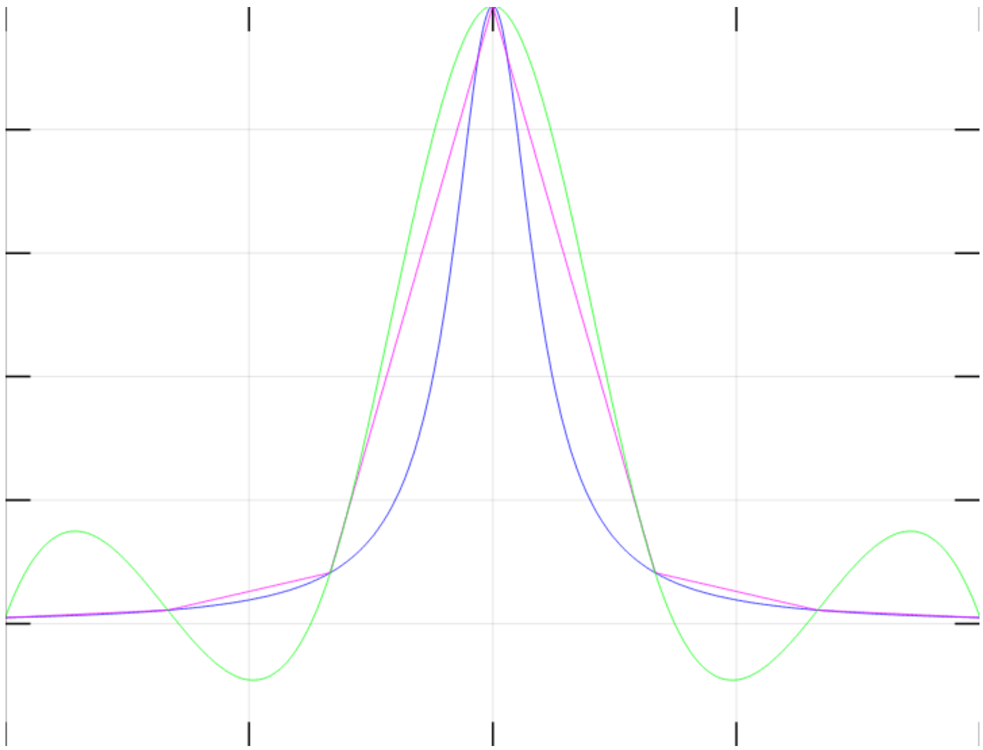
```
octave:9> plotnewton(f,-2,2,10)
```



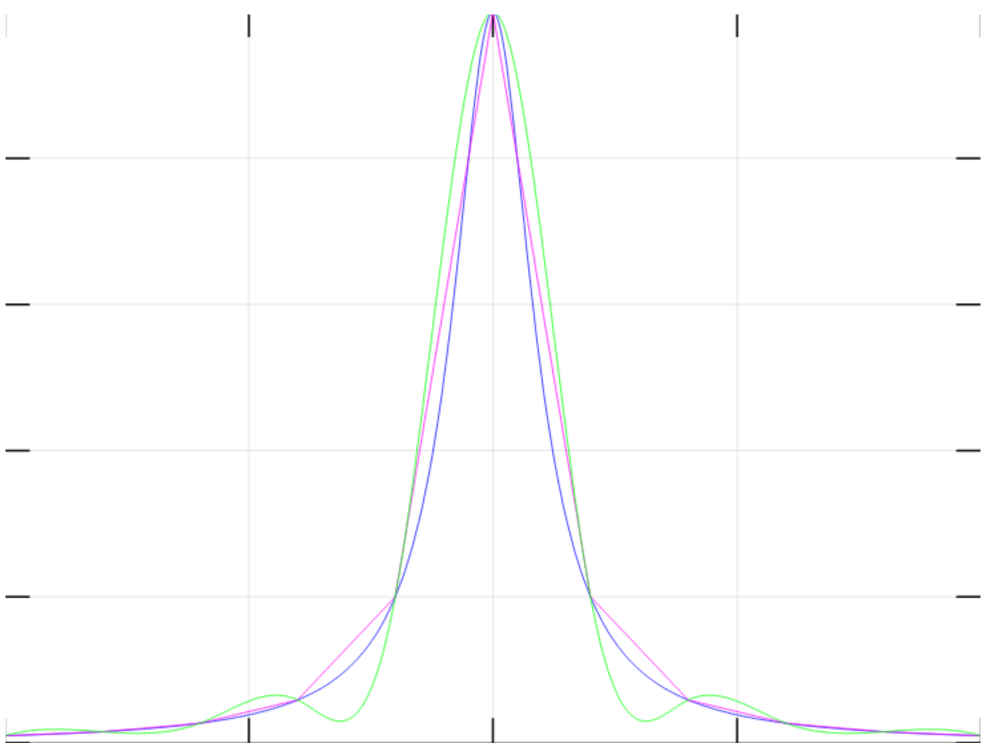
```
function plotspline(f,a,b,k)
h = (b-a)/500;
x=zeros (1, 501);
yreal= x;
ynewton= x;

#real function
for i=1:501
  x(i)= a+(i-1)*h;
  yreal(i)= f(x(i));
endfor

#interp t points
h2 = (b-a)/k;
t=zeros (1, k+1);
for i=1:k+1
      t(i)= a+(i-1)*h2;
endfor

ylinear= interp1(t,f(t),x);
yspline= interp1(t,f(t),x, "spline");

plot(x, yreal, "b", x, ylinear, "m", x, yspline, "g"); grid

endfunction

octave:3> plotspline(f,-2,2,4)
```



```
octave:4> plotspline(f,-2,2,6)
```

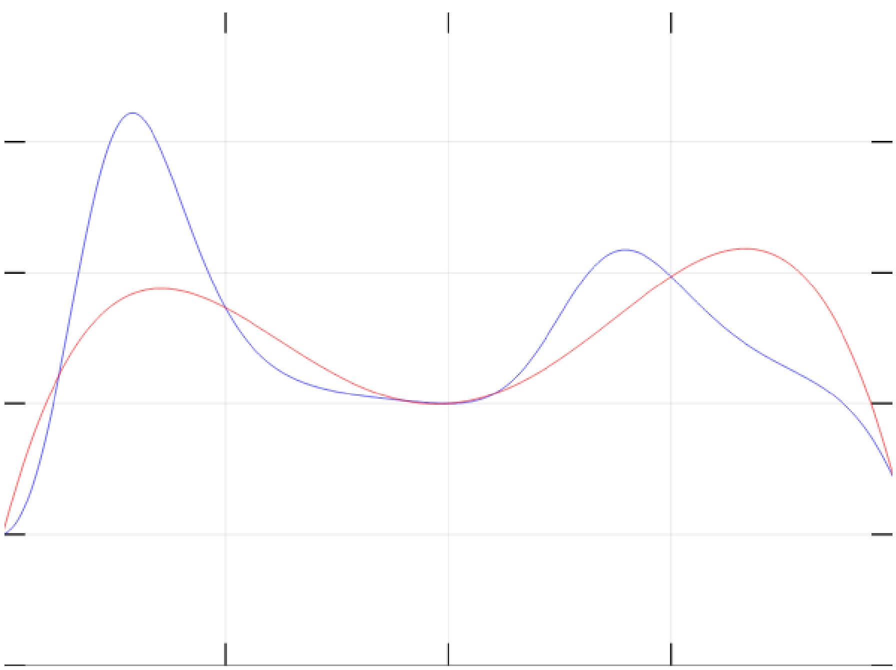


```
octave:6> plotspline(f,-2,2,10)
```

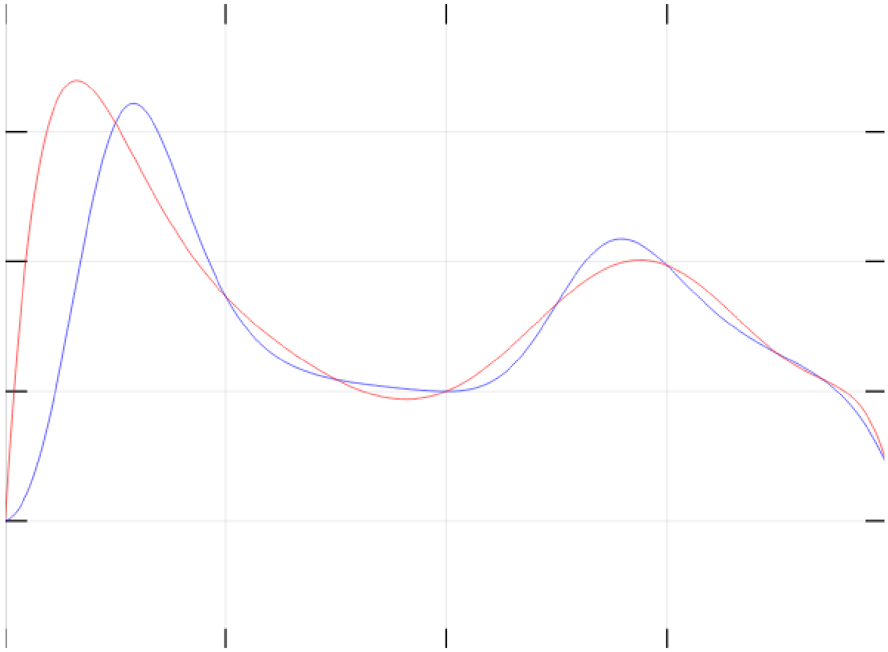


Αποτελέσματα για την $g(x) = \sin(x^2)e^{\sin(3x)}$

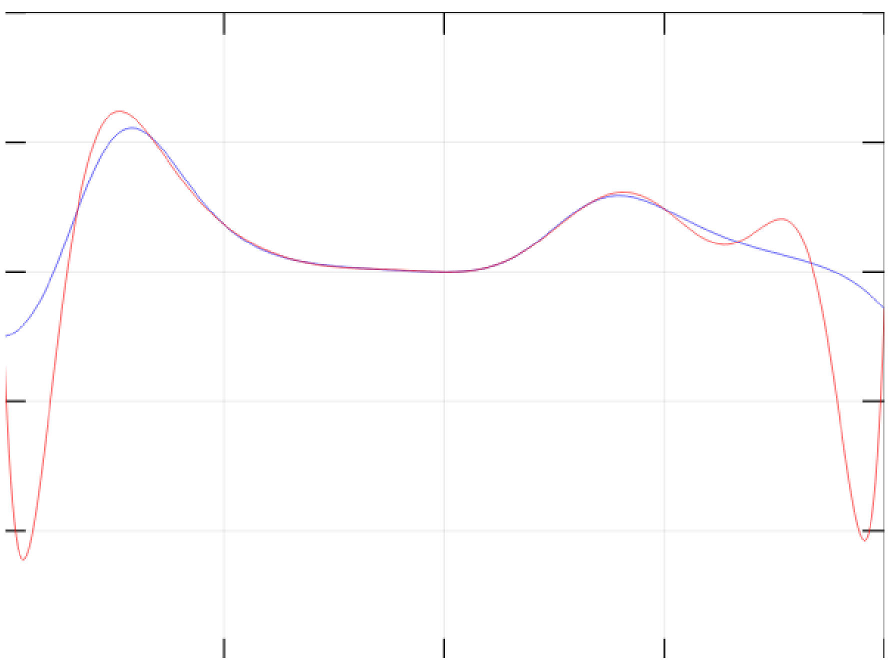
```
octave:12> g=@(x) sin(x.^2).*(e.^sin(3.*x))
g =

@(x) sin (x .^ 2) .* (e .^ sin (3 .* x))

octave:27> newtondivided(g,-2,2,4)
D =

  -1.0008         0         0         0         0
   0.7307    1.7315         0         0         0
        0   -0.7307   -1.2311         0         0
   0.9690    0.9690    0.8499    0.6937         0
  -0.5723   -1.5413   -1.2552   -0.7017   -0.3488

ans =

  -1.0008    1.7315   -1.2311    0.6937   -0.3488


octave:13> plotnewton(g,-2,2,4)
```



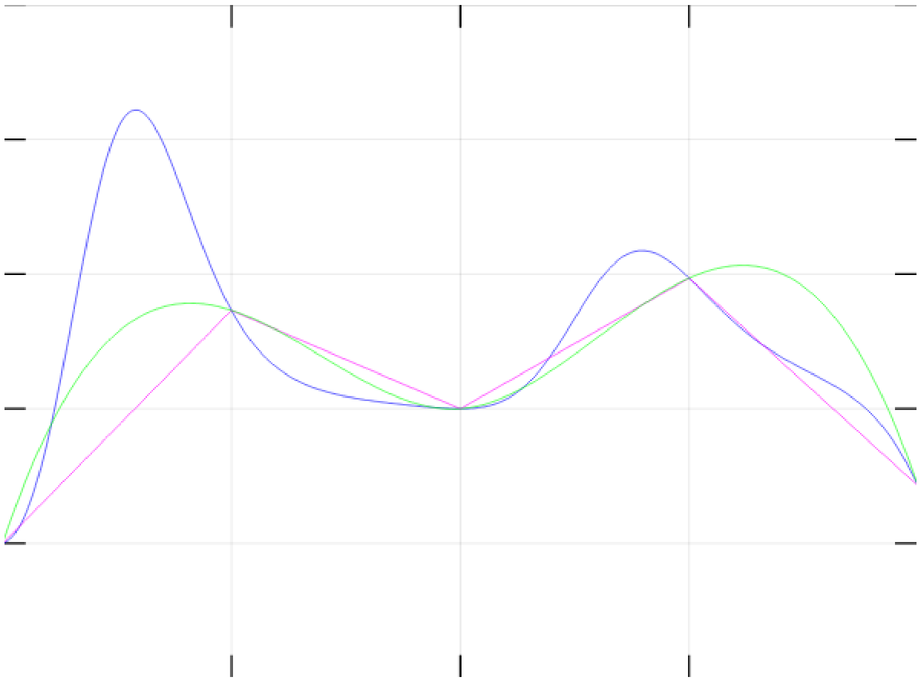
```
octave:14> plotnewton(g,-2,2,8)
```



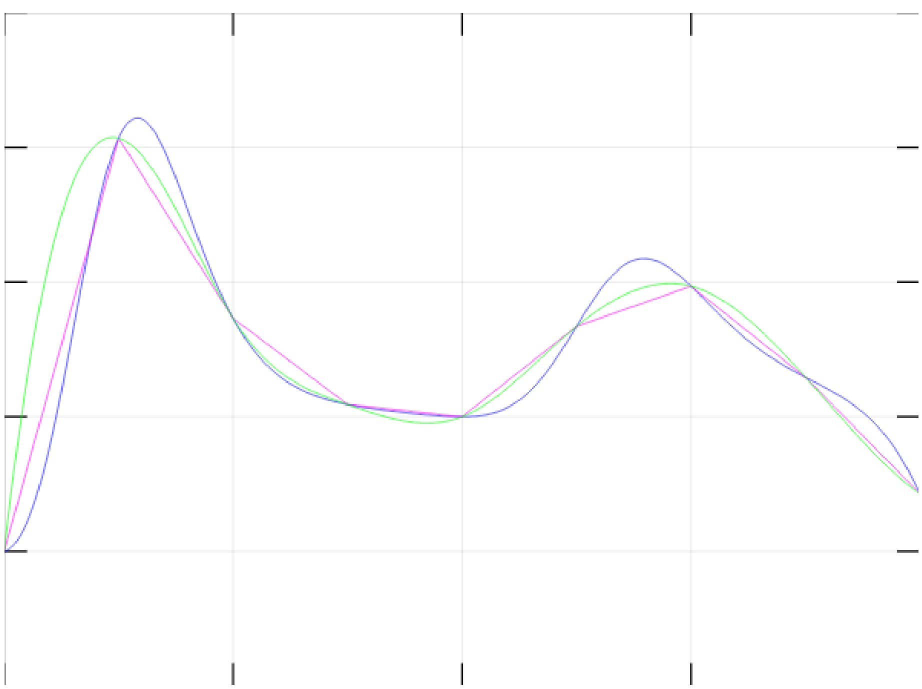
```
octave:16> plotnewton(g,-2,2,12)
```



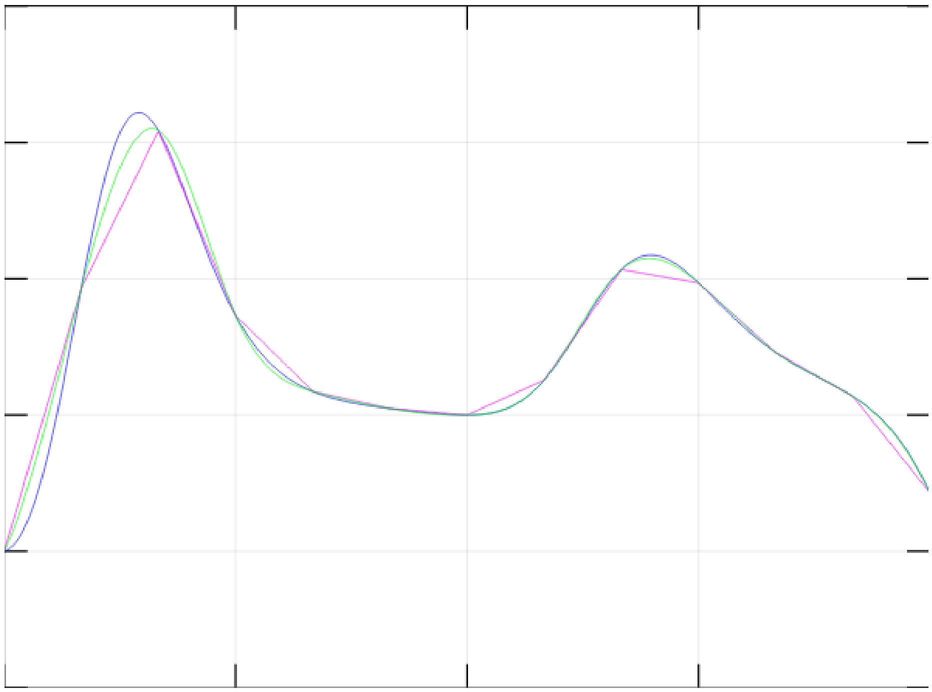
octave:23> plotspline(g,-2,2,4)



octave:24> plotspline(g,-2,2,8)



```
octave:25> plotspline(g,-2,2,12)
```



Στην συνάρτηση plotnewton() παρατηρούμε ότι με την αύξηση του βαθμού του πολυωνύμου παρεμβολής, προσεγγίζουμε καλύτερα την συνάρτηση Runge. Ωστόσο η προσέγγιση παρουσιάζει μεγάλα σφάλματα στα άκρα του διαστήματος, όπου και παρουσιάζει μεγάλες ταλαντώσεις, παρά το μεγάλο k. Με την χρήση τμηματικών πολυωνύμων, στην plotspline(), το πρόβλημα αυτό διορθώνεται και η προσέγγιση καλύπτει με μεγαλύτερη/ομοιόμορφη ακρίβεια όλο το διάστημα. Τέλος αυξάνοντας το πλήθος κόμβων για τις splines βελτιώνεται η προσέγγιση. Αντίστοιχα αποτελέσματα παρατηρούμε και για την g(x).