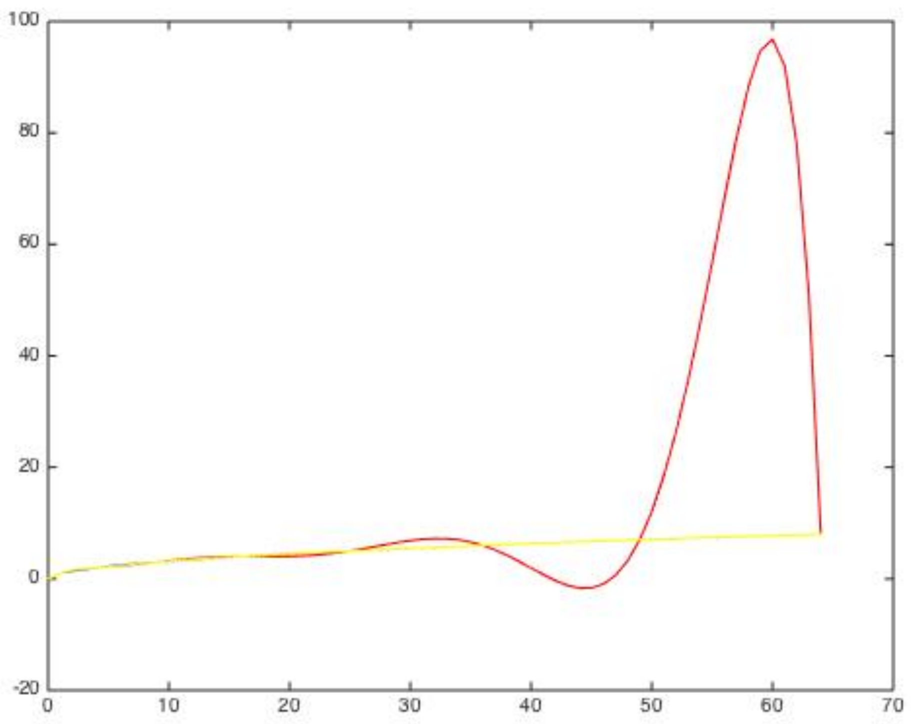
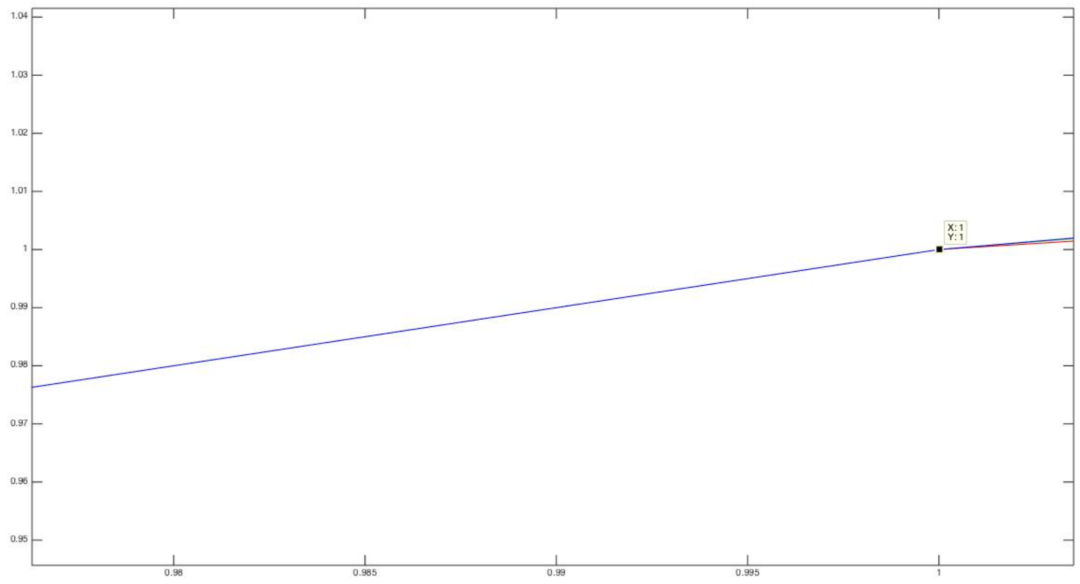
```
syms x l;
x1=[0,1,4,9,16,25,36,49,64];
y1=[0,1,2,3,4,5,6,7,8];
n=length(x1);
POLY=sym(0);
for i=1:n
    l=sym(y1(i));
     for k=1:i-1
         l=l*(x-x1(k))/(x1(i)-x1(k));
     end

     for k=i+1:n
         l=l*(x-x1(k))/(x1(i)-x1(k));
     end
     POLY=POLY+l;
end
POLY=simplify(POLY)


R=[0:1:64];
CS=spline(x1,y1,R);
p=polyfit(R,CS,3);
S=p(1)+p(2)*x+p(3)*x^2+p(4)*x^3

poly=95549/72072*R-24221063/63504000*R.^2+657859/10886400*R.^3+33983/152409600*R.^5
-13003/2395008000*R.^6+19/283046400*R.^7-2168879/435456000*R.^4-1/3048192000*R.^8;
plot(R,sqrt(R),'b',R,poly,'r',R,CS,'y')
```



The cubic spline interpolation generally approaches the real data line closer than the polynomial approximation line in total range of [0,64].
In range of [0,1], we can observe that the real data line, cubic spline interpolation line and polynomial interpolation line are very much close to each other. We can hardly judge which method is closer to real situation.



Use function of polyval, we can compare the errors of these two methods. From several data sets whose domain ranges from 0 to1, we can find that the polynomial interpolation performs better in [0,1].