Ian Wilhite
Meen 357
A6

Task 1:

a) See comments in code.

```
def QuadraticSpline(x, fx):

    #calculate the number of segments & constants
    nsegs = len(x)-1
    nknts = len(x)
    # create empty arr A and vec B of len 3 * the number of segments
    # vec B must have 3 constants to define each quadratic section, and
    # Arr A must have the same dimension as B to create a valid system to
solve

    A = np.zeros((nsegs*3, nsegs*3))
    b = np.zeros((nsegs*3, 1))

    # fill in system for quadratic spline segments
    for i in range(1, nknts-1):
        #set up matrix ax^2 + bx + c
        A[2*i-2, 3*(i-1)] = x[i]**2
        A[2*i-2, 3*(i-1)+1] = x[i]
        A[2*i-2, 3*(i-1)+2] = 1
        A[2*i-1, 3*(i-1)+3] = x[i]**2
        A[2*i-1, 3*(i-1)+4] = x[i]
        A[2*i-1, 3*(i-1)+5] = 1

        b[2*i-2] = fx[i]
        b[2*i-1] = fx[i]

    #boundary condition for first
    A[2*nsegs-2, 0] = x[0]**2
    A[2*nsegs-2, 1] = x[0]
    A[2*nsegs-2, 2] = 1
    b[2*nsegs-2] = fx[0]

    #boundary conditions for last
    A[2*nsegs-2+1, -3] = x[-1]**2
    A[2*nsegs-2+1, -2] = x[-1]
    A[2*nsegs-2+1, -1] = 1
    b[2*nsegs-2+1] = fx[-1]
    #iterate through constants
    for i in range(1, nknts-1):
        A[(2*nsegs-2)+2 + (i-1), 3*(i-1)] = 2*x[i]
        A[(2*nsegs-2)+2 + (i-1), 3*(i-1)+1] = 1
        A[(2*nsegs-2)+2 + (i-1), 3*(i-1)+3] = -2*x[i]
        A[(2*nsegs-2)+2 + (i-1), 3*(i-1)+4] = -1
        b[(2*nsegs-2)+2 + (i-1)] = 0

    #set the inital conditions for the slope at the beginning and end
    A[(2*nsegs-2)+ 2 + nsegs-1, 0] = 1
    b[(2*nsegs-2) + 2 + nsegs-1] = 0
    # use np linalg to solve the matrix
    coeffs = la.solve(A, b)

    return coeffs #return
```

b) See code and Fig 1.

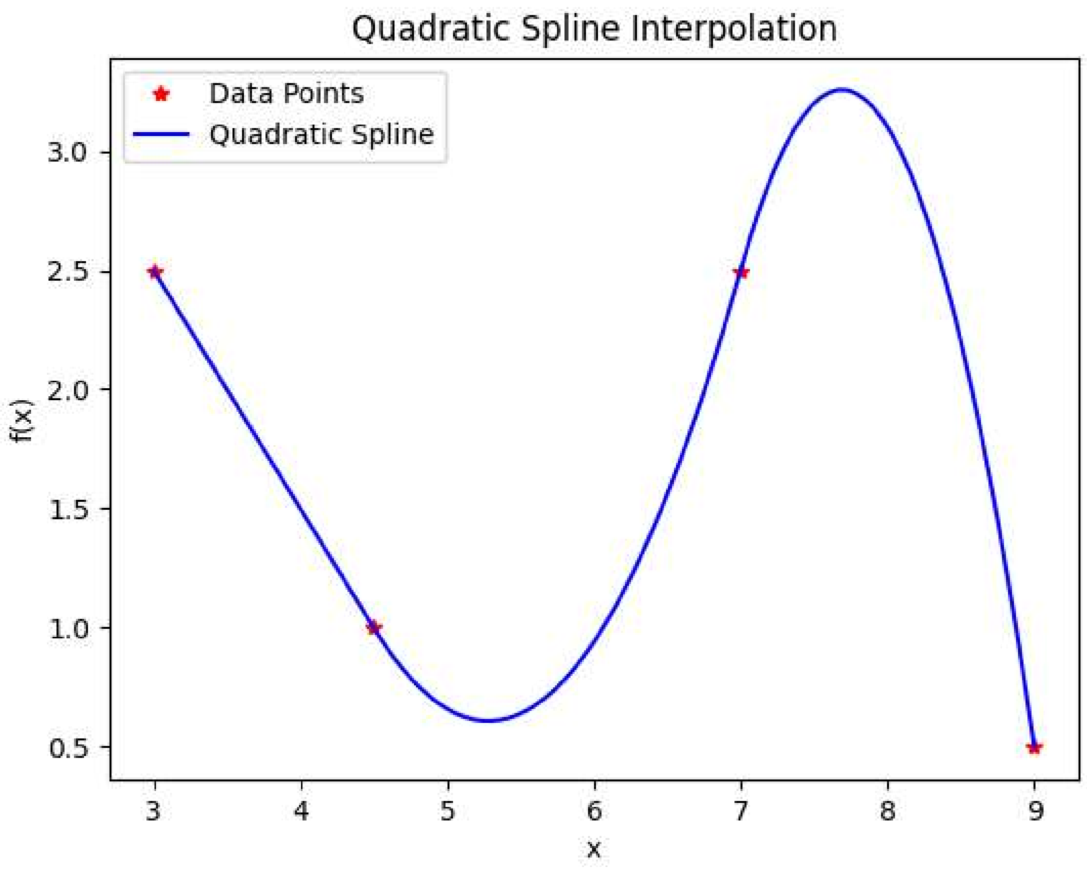
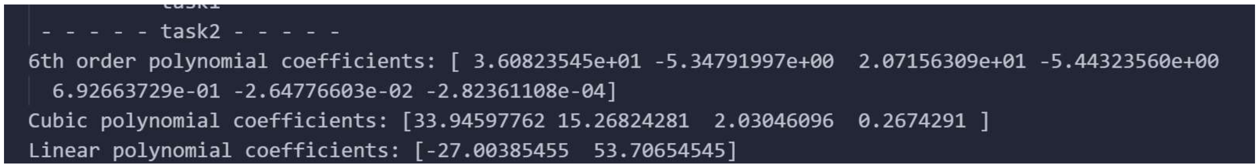


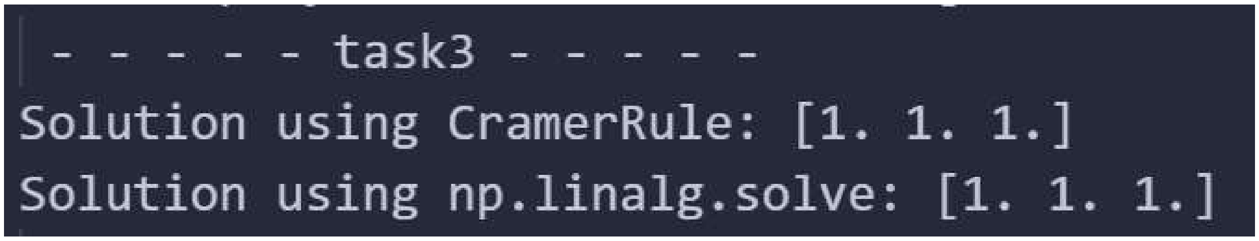
Figure 1. Quadratic Interpolation

Task 2:



```
 - - - - - task2 - - - - -
6th order polynomial coefficients: [ 3.60823545e+01 -5.34791997e+00  2.07156309e+01 -5.44323560e+00
  6.92663729e-01 -2.64776603e-02 -2.82361108e-04]
Cubic polynomial coefficients: [33.94597762 15.26824281  2.03046096  0.2674291 ]
Linear polynomial coefficients: [-27.00385455  53.70654545]
```

Task 3:



```
 - - - - - task3 - - - - -
Solution using CramerRule: [1. 1. 1.]
Solution using np.linalg.solve: [1. 1. 1.]
```

Task 4:

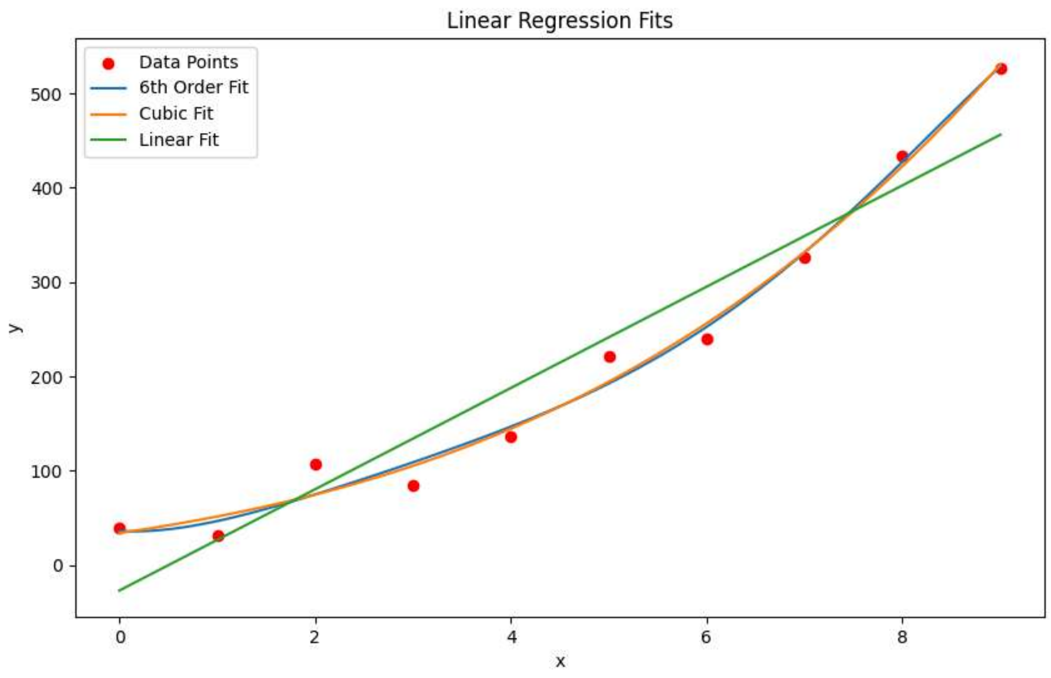


Figure 2. Linear Regressions