1)

a)

For each interpolating point x_j in [a,b], we can define an equation

P(x_j) = c_n + c_(n-1)*(x_j) + c_(n-2)*(x_j)^2 + ... + c_1*(x_j)^(n-1)

which lets us define the linear system that one would need to solve as

V*c=y

where

V is the main matrix we need to invert, written as

[
[1, x_0, x_0^2, ..., x_0^n],
[1, x_1, x_1^2, ..., x_1^n],
...
[1, x_n, x_n^2, ..., x_n^n]
]

,

c is the vector of unknown c_i values from i in {0,1,...,n}, written as

[c_n, c_(n-1), c_(n-2), ...,c_1]^T

,

and y is the vector of nth Lagrange Interpolating Polynomial evaluated at each x_j, written as

[P(x_0), P(x_1), ..., P(x_n)]^T = [f(x_0, f(x_1), ..., f(x_n)]^T *(by construction)*

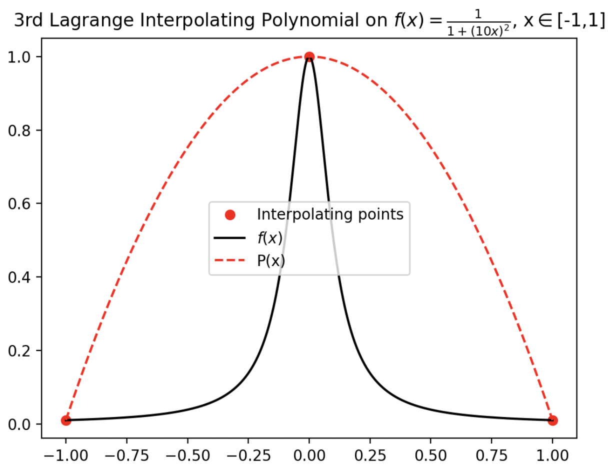
The code that would solve for the coefficients would be solving the linear system, which we could do by simply multiplying the inverse of V against vector y, which is full of knowns as for each interpolating point x_j, P(x_j) = f(x_j).

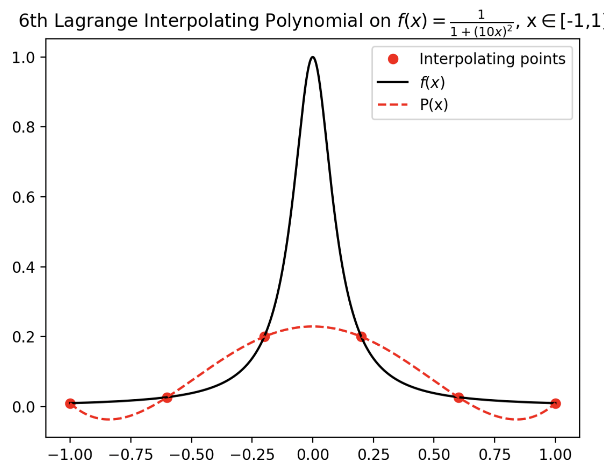
Code:

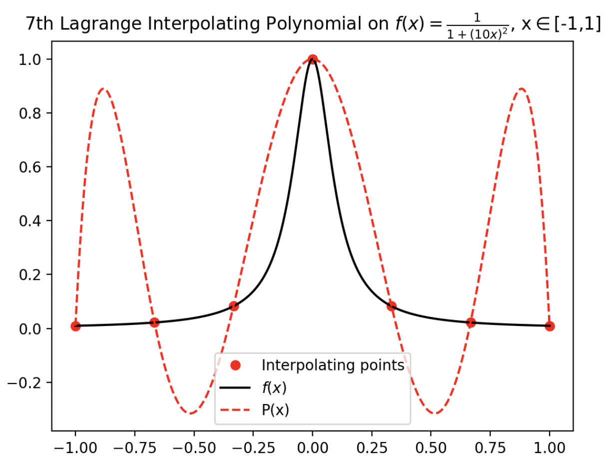
```
# create interpolating nodes, and vector y by evaluate f at those nodes
x = np.zeros(N)
for i in range(1, N + 1):
	x[i - 1] = -1 + (i - 1) * h
y = f(x)

# create matrix Vandermonde matrix
V = np.zeros((N, N))
for i, x_j in enumerate(x):
	for j in range(N):
		V[i][j] = x_j**j
# solve for c vector
c = np.matmul(np.linalg.inv(V), y)
```

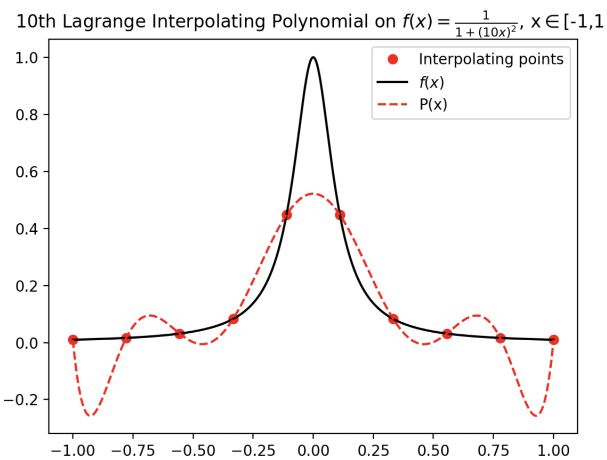
b)

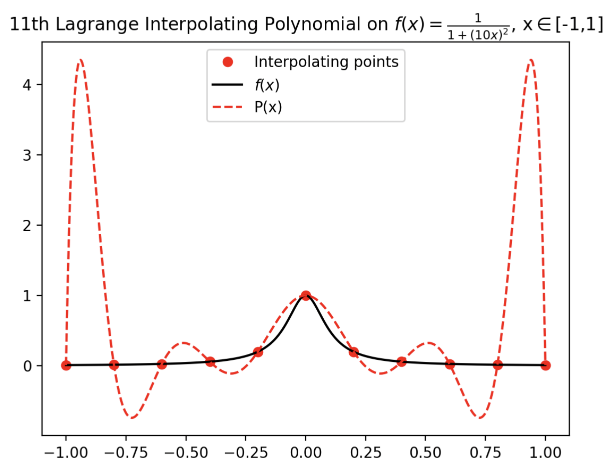
Below, we plot *P(x)* with solved *c* coefficients for several *N* as directed. As *N* increases, we can see the error gets worse at endpoints. We can also see that odd *N* predicts better at the midpoint while even *N* is better at the endpoints, which is caused by a difference in where the interpolating points lie over the interval [a,b].

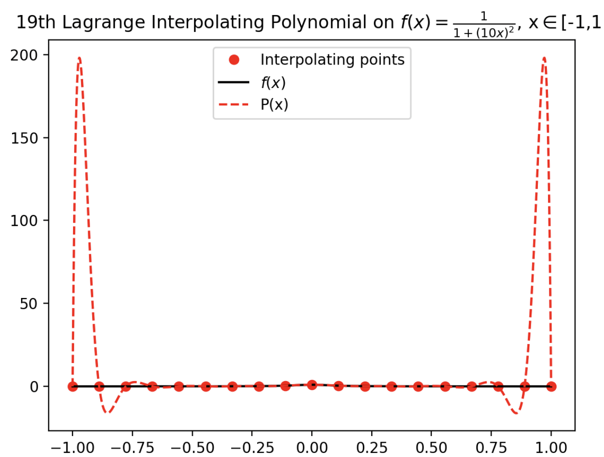






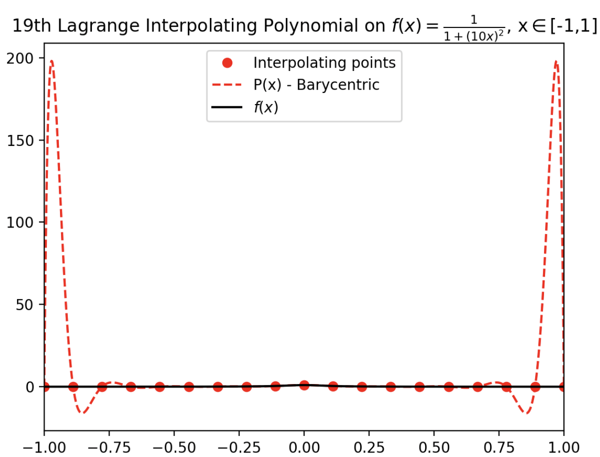


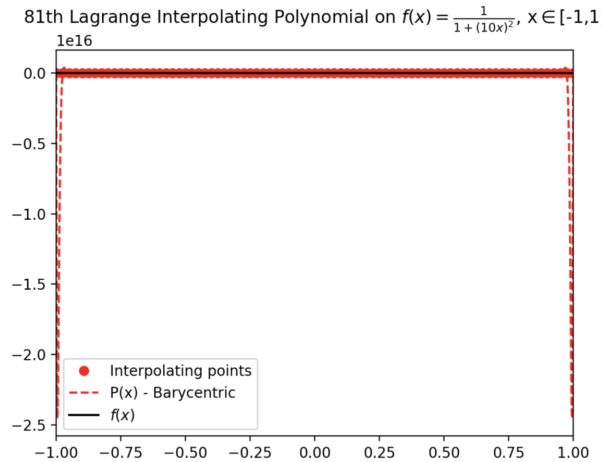




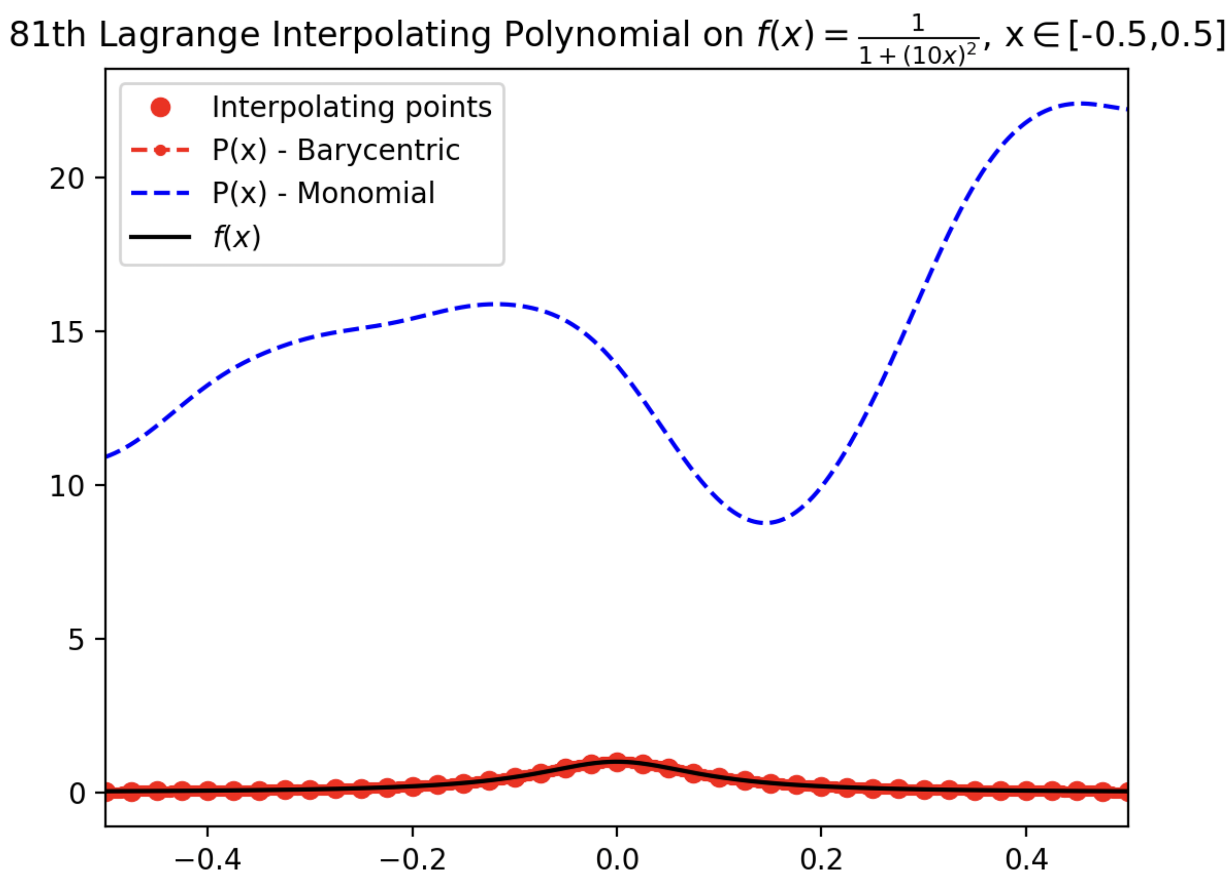
2)

Here, we see the graph for P(x) using the barycentric Lagrange interpolation formula. We see we still get bad behavior towards endpoints, which is especially apparent for high (and odd) *N*.



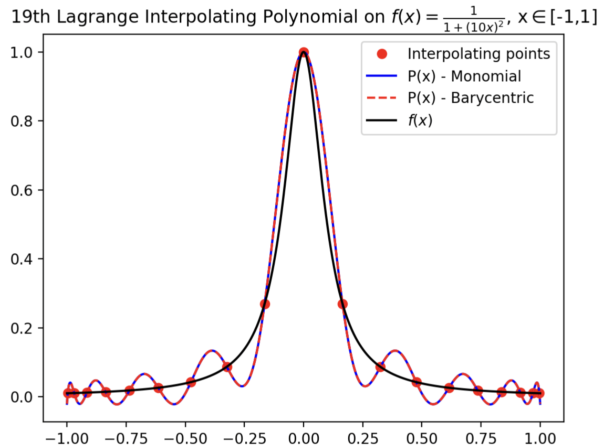


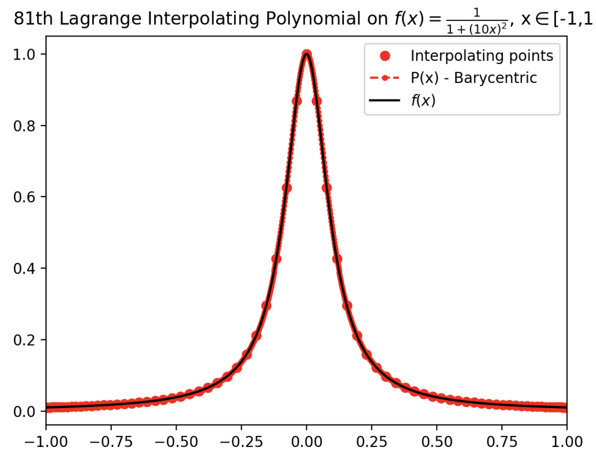
However, compared to the Monomial basis of problem 1, we see that the Barycentric polynomial is much more accurate/stable with small *x* for large number of interpolating points, *N*.



3)

Using the Chebyshev points that are clustered towards endpoints, we get far better approximations near endpoints for large N, as seen below.





We can still get the interpolation to fail, however, if we use the Monomial basis for large N. This is because the method itself is very ill-conditioned for large N (matrix inversion). We show the difference between the Monomial and Barycentric method for N=81 below.