a)

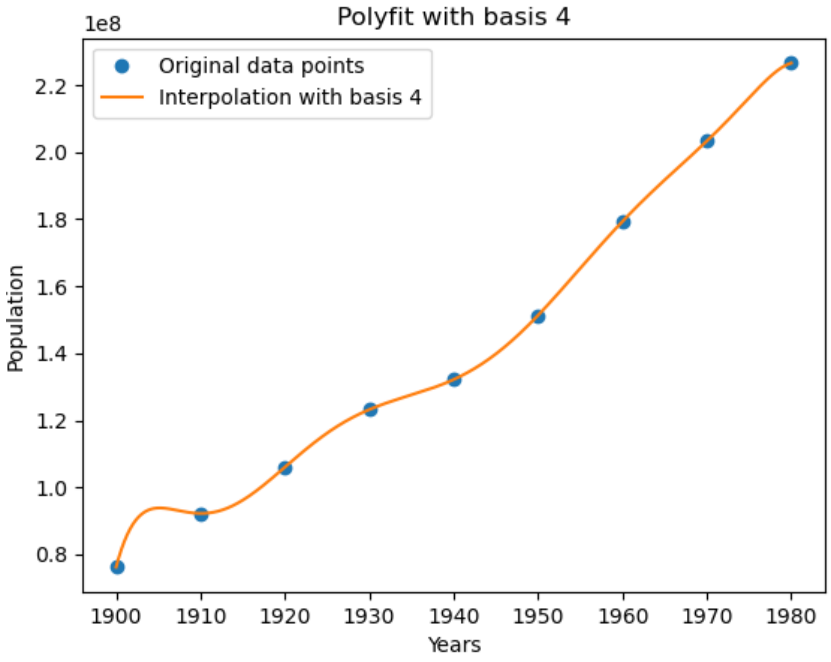
The conditional numbers computed from the Vander matrices for each basis are shown below:

Basis	Basis 1	Basis 2	Basis 3	Basis 4
Condition Num	1.0997e+42	5.9943e+15	9.3155e+12	1.6054e+03

We can see that basis 4 has the best conditional number from the chart above. This is the case because as the degree of the polynomial increases, the column of the bander matrix become nearly linear dependent and hence the polynomial coefficient becomes highly sensitive. Therefore, if we change the basis of our input, we can obtain a matrix that is less ill-conditioned.

b)

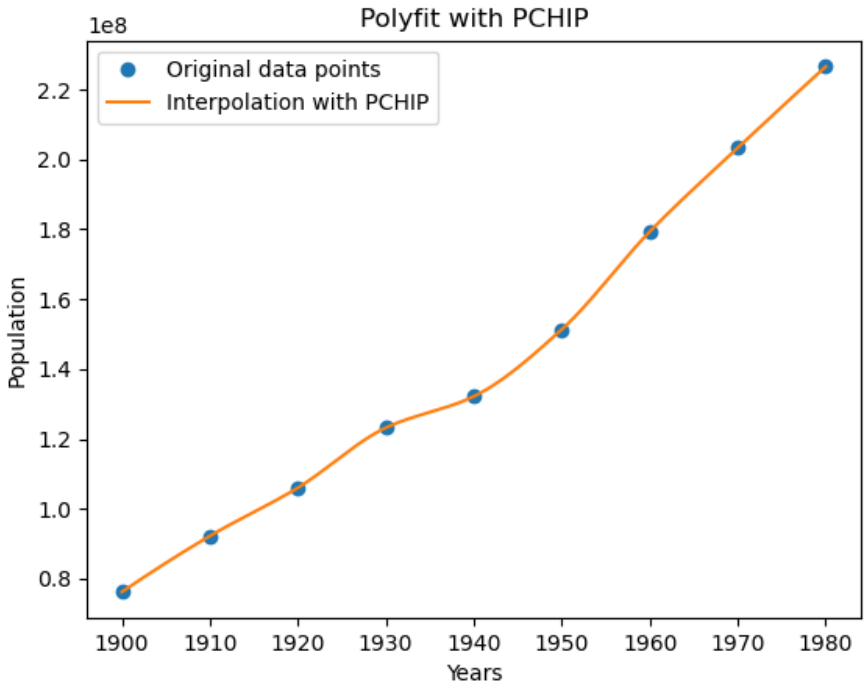
The degree 8 polynomial fitting result is shown below:



For this question, I used the basis4 from last question to form the Vander matrix and solve the coefficients using np.linalg.solve(). Then, I use NumPy's polval() to obtain the interpolated result since polyval() uses Horner's rule to calculate polynomial values by default.

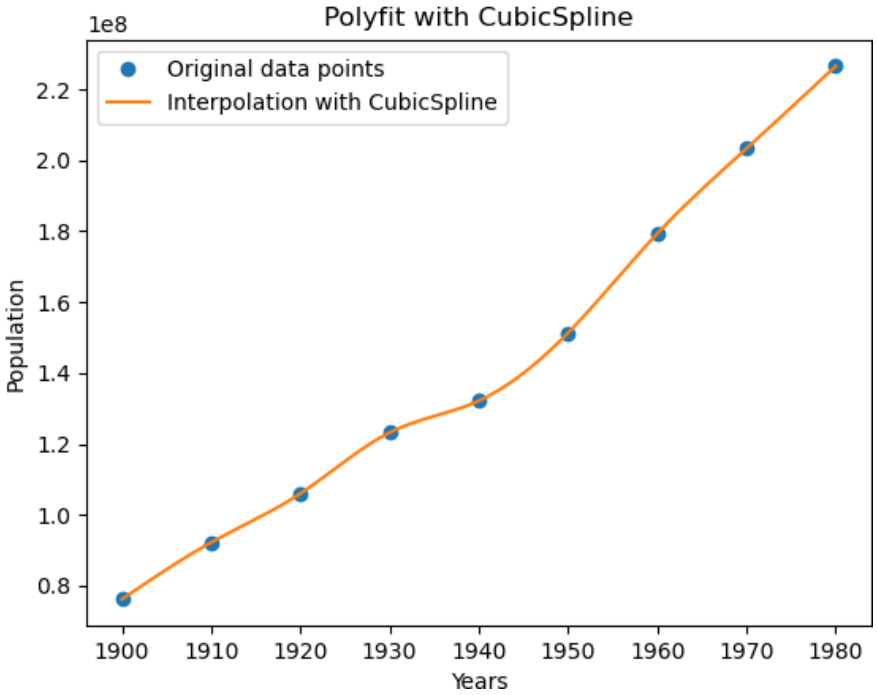
c)

The interpolation result for Hermit cubic interpolation is shown below:



In this question, I used the function scipy.interpolate.pchip() to interpolate this result. Compare to the plot in part(b), we see that Hermit cubic interpolation's plot is much smother than the plot produced by degree 8 poly fitting, especially during 1900 to 1910.

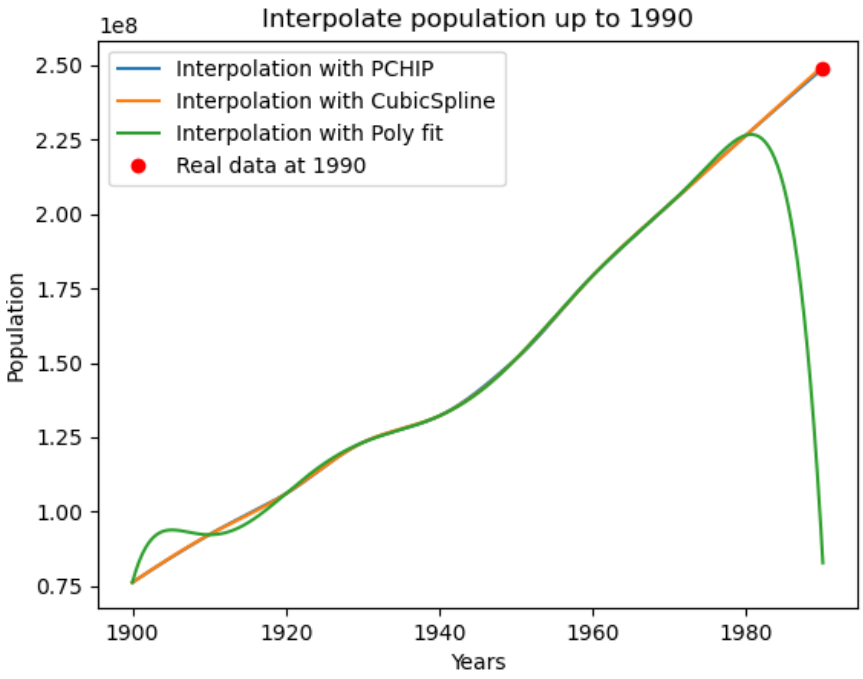
d)



Result for cubic spline interpolation is shown in the plot above. In this question, I used SciPy's CubicSpline() function to calculate the interpolation. The boundary condition I used in my implementation is "natural", which means that the second derivative at the curve ends are zero.

e)

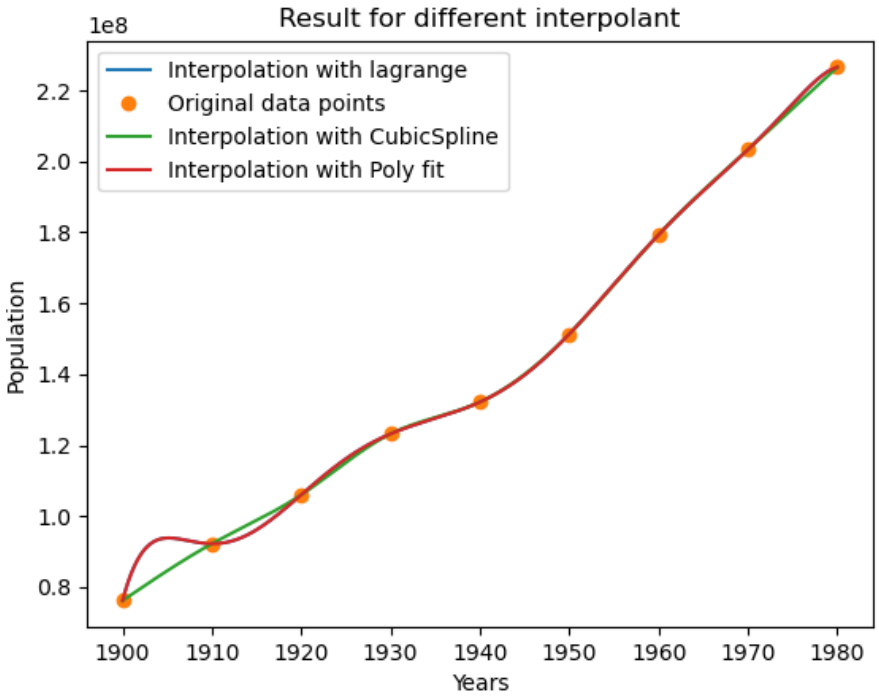
The result for different interpolation method outside of data range is shown below:



From the above plot we can see that Hermit cubic and CubicSpline interpolation produces nearly identical curves. Hermit cubic and CubicSpline made a good prediction outside of the range of datapoints we provided. Hermit cubic interpolation predict a population of 249032123 while cubic method predicts a population of 249782367. However, the polynomial fit fails spectacularly outside the interpolation domain we have given. This might be due to poly fit method will encounter numerical instability near the boundary of the datapoints. Which means that polynomial fitting is only suitable for interpolating areas that lies within the given datapoints.

f)

The interpolation result the interpolation method we used in this question is shown below:

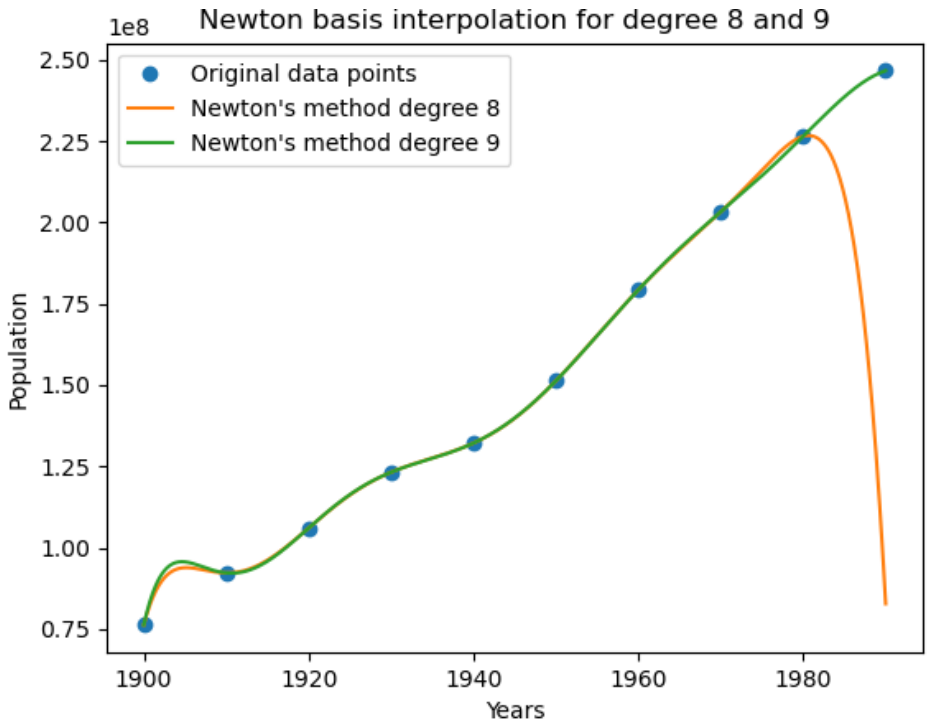


In this question, the time measure here is only the evaluation of the interpolation, not the construction of the interpolation. In my code, the evaluation time of the algorithm is as follows:

Lagrange interpolation methods took 0.0714 ms to evaluate, the cubic method took 0.0450 to evaluate and the polynomial fit took 0.0525 to evaluate. From the result for my implementation, the Lagrange method have the highest evaluation time among all the other methods.

g)

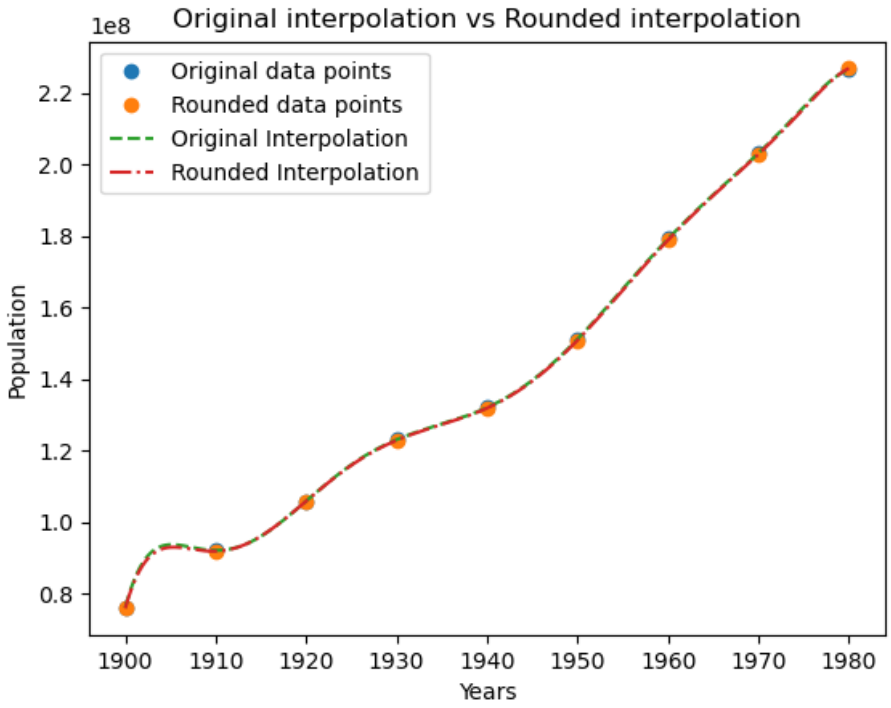
In this question, I use the lower triangle matrix to solve for the coefficient. The result is shown below:



From the plot above we can see that the newton interpolation for degree 9 is similar to degree 8, but since we are using 10 data points instead of 9, it interpolate the interval 1980 – 1990 correctly instead of going downwards like degree 8 polynomial did. On the other intervals, degree 8 and degree 9 polynomial looks very similar to each other.

g)

The interpolation result for this part in shown below:



In my implementation I used the same technic and basis as part b. The polynomial on the plot suggests that their coefficient should be very similar with each other. By printing out the actual value of the coefficients, we can confirm that this is the case. The coefficient for population before rounding is:

[-3.15180235e+08 1.89175576e+08 6.06291250e+08 -3.42668456e+08

-3.74614715e+08 1.82527130e+08 1.02716315e+08 4.61307656e+07

1.32164569e+08]

And the coefficient for population values after rounding is:

[-2.94196825e+08 1.86920635e+08 5.70311111e+08 -3.38488889e+08

-3.56755556e+08 1.81111111e+08 1.00141270e+08 4.59571429e+07

1.32000000e+08]

Which they are very similar to each other, thus confirming what the plot have told us. The similar coefficient can be explained in terms of error bounds: If we think of the population vector after rounding as b + error_b, then the relative changes in A can be determined by the following equation:

$$\frac{||\ \text{error\_coeffecient}\ ||}{||\ \text{coeffecient}\ ||} \leq \text{Cond(A)} \frac{||\ \text{error\_b}\ ||}{||\ \text{b}\ ||}$$

Which explained why the changes in the coefficient changes are small (since the percentage in b is small).