1. a) The condition numbers are:

1. 8.4136×10^{41}
2. 5.9943×10^{15}
3. 9.3155×10^{12}
4. 1.605×10^{3}

It is very obvious that the condition numbers are decreasing from matrix with basis function 1 to matrix with basis function 4, while the number droped dramatically from matrix 1 to 2 and 3 to 4, but the drop in condition number between matrix 2 and 3 is significantly less than others.

We know that the condition number of a matrix : measures the ratio of the maximum relative stretching to the maximum relative shrinking that the matrix does to any non-zero vectors: (Heath - chapter 2) The reason for above observation is that:

. The values in each row of vandermonde matrix of $\phi_j(t) = t^{j-1}$ are growing with factors around 1000.

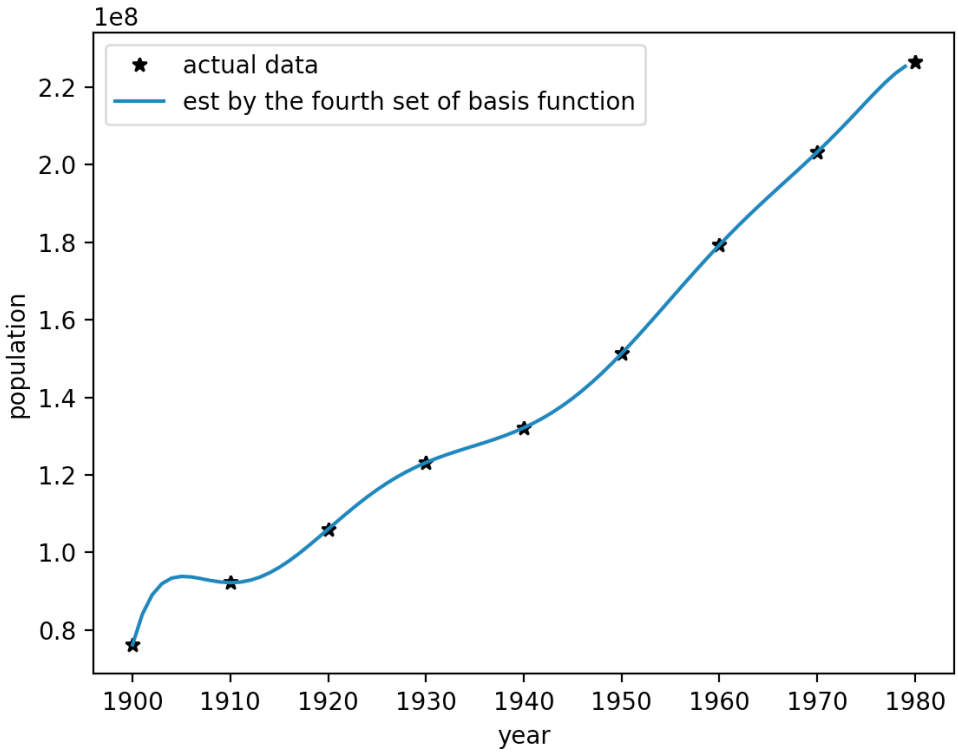
Thus, this matrix is extremely emphasizing coefficients at the back, for example:

$1a_1 + 1900a_2 + 1900^2 a_3 + 1900^3 a_4 + 1900^4 a_5 + \cdots + 1900^7 a_8$, a_8 is 1900^7 times more important than a_1.

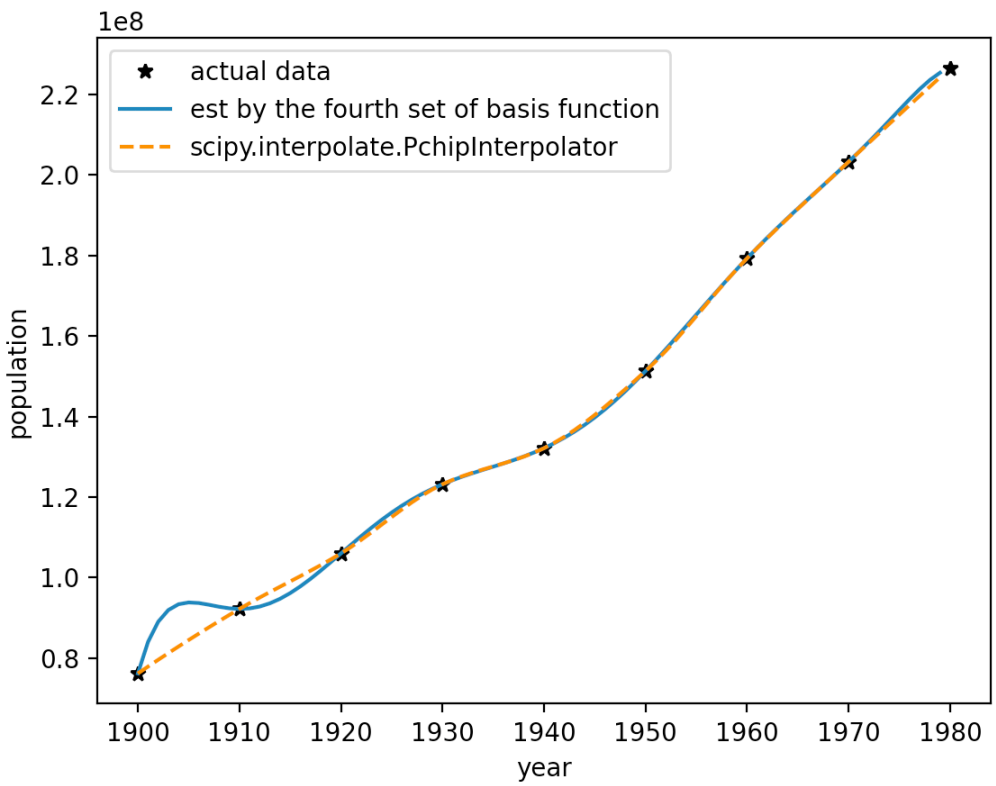
Thus, the matrix is ill conditioned and skewed toward as.

- The vandermonde matrix for $\phi_j(t) = (t-1900)^{j-1}$ and $\phi_j(t) = (t-1940)^{j-1}$ improved a little bit by subtract the t value to and mean value. So the condition number drops a lot, but still very large. The third one performs a little bit better that the second one since it uses the mean value rather that to. The absdute differences between t_0, t_n and mean are smaller than the absolute difference between t_0, t_n and t_0.
- The last vandermonde matrix improved even further by scale t values by $\frac{1}{40}$ after subtracted by 1940. All coefficients are getting much more equivalent roles, so the condition number dropped a lot again.

b)

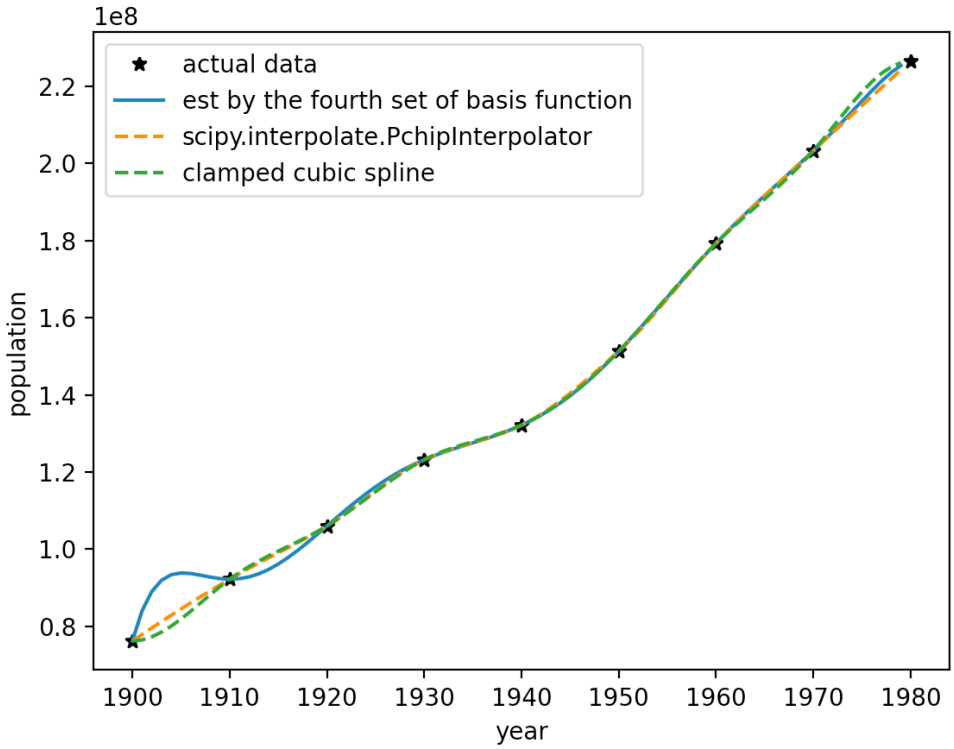


c)



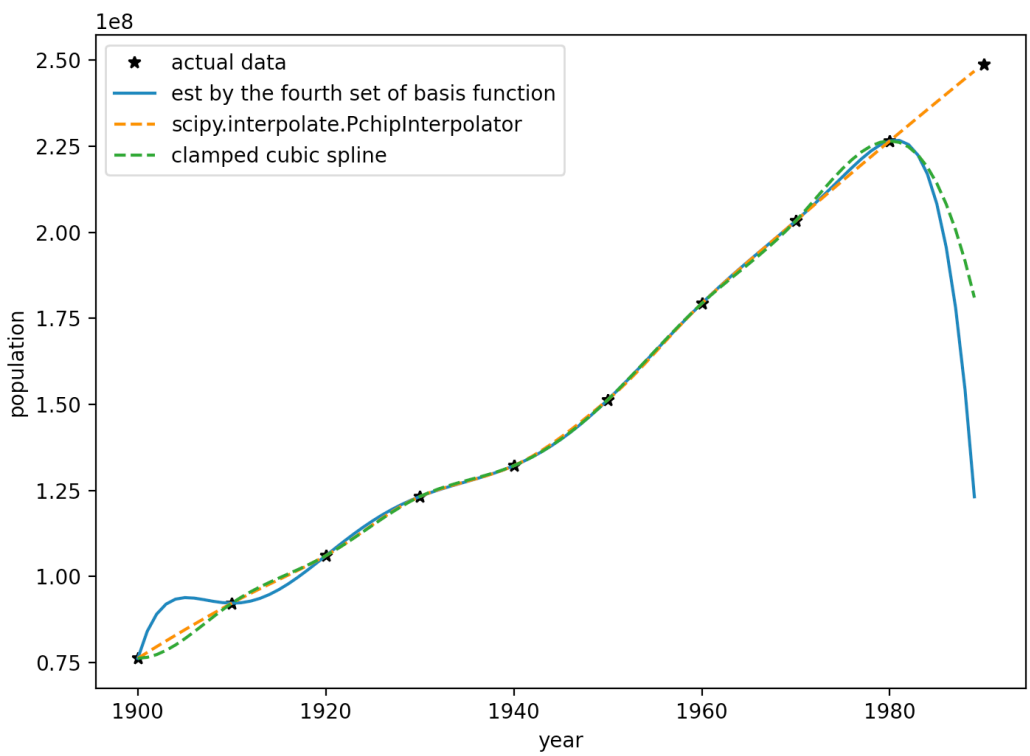
used scipy.interpolate.PchipInterpolator

d)



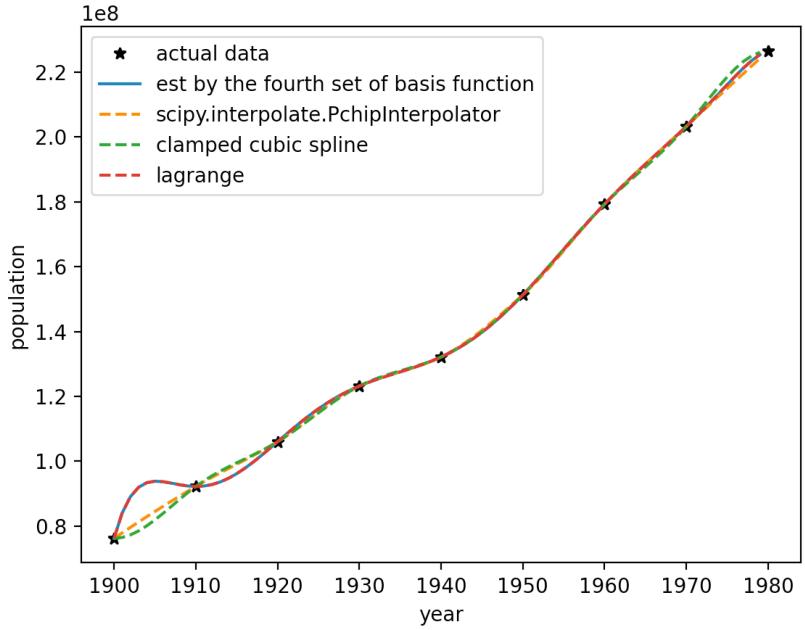
used clamped cubic spline.

e)



Since we did not take the year 1990 into consideration when calculating the coefficient, two of the lines decreases dramatically after they passed the year 1980. They are the polynomial and clamped cubic spline. The polynomial method decreases faster than the clamped cubic spline and resulted an almost 1×10^8 difference at 1990. The clamped cubic spline performed just a little bit better and also resulted a difference around 4×10^7. However, the pchip line performed amazingly well. It almost passed the actual data point on the graph without taking 1990 into consideration as well. By printing the result for pchip method, $\underbrace{24681789}_{\text{pchip}} \Longleftrightarrow \underbrace{24809873}_{\text{actual}}$, we can see both numbers are extremely close to each other.

f)




```
time for horner's evaluation scheme (polyval): 4.2551e-05
time for cubic spline: 1.9455e-05
time for lagrange: 8.9880e-05
```

For the time costs, I evaluated 80 points (every year from 1900-1980) for each method 5 times, and use the average value as their final result. As we can see, the cubic spline method has the best performance, and then the horner's evaluation scheme, the lagrange method takes the most time to compute these values.

9)

lower triangular matrix

$$
\begin{bmatrix}
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
t_2-t_1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
t_3-t_1 & (t_3-t_1)(t_3-t_2) & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
t_4-t_1 & (t_4-t_1)(t_4-t_2) & \frac{(t_4-t_3)(t_4-t_2)}{(t_4-t_1)} & 0 & 0 & 0 & 0 & 0 & \\
t_5-t_1 & (t_5-t_1)(t_5-t_2) & \frac{(t_5-t_3)(t_5-t_2)}{(t_5-t_1)} & \frac{(t_5-t_4)(t_5-t_3)}{(t_5-t_2)(t_5-t_1)} & 0 & 0 & 0 & 0 & \\
t_6-t_1 & (t_6-t_1)(t_6-t_2) & \frac{(t_6-t_3)(t_6-t_2)}{(t_6-t_1)} & \frac{(t_6-t_4)(t_6-t_3)}{(t_6-t_2)(t_6-t_1)} & \frac{(t_6-t_5)(t_6-t_4)}{(t_6-t_3)(t_6-t_2)(t_2-t_1)} & 0 & 0 & 0 & \\
t_7-t_1 & (t_7-t_1)(t_7-t_2) & \frac{(t_7-t_3)(t_7-t_2)}{(t_7-t_1)} & \frac{(t_7-t_4)(t_7-t_3)}{(t_7-t_2)(t_7-t_1)} & \frac{(t_7-t_5)(t_7-t_4)}{(t_7-t_2)(t_7-t_2)(t_7-t_1)} & \frac{(t_7-t_6)}{(t_7-t_5)(t_7-t_4)(t_7-t_2)(t_7-t_1)} & 0 & 0 & \\
t_8-t_1 & (t_8-t_1)(t_8-t_2) & \frac{(t_8-t_2)(t_8-t_2)}{(t_8-t_1)} & \frac{(t_8-t_4)(t_8-t_3)}{(t_8-t_2)(t_8-t_1)} & \frac{(t_8-t_5)(t_8-t_4)}{(t_8-t_2)(t_8-t_2)(t_8-t_1)} & \frac{(t_8-t_6)}{(t_8-t_5)(t_8-t_4)(t_8-t_2)(t_8-t_1)} & \frac{(t_8-t_7)(t_8-t_6)}{(t_8-t_5)(t_8-t_4)(t_8-t_2)(t_8-t_2)(t_8-t_1)} & 0 & \\
t_9-t_1 & (t_9-t_1)(t_9-t_2) & \frac{(t_9-t_3)(t_9-t_2)}{(t_9-t_1)} & \frac{(t_9-t_4)(t_9-t_3)}{(t_9-t_2)(t_9-t_1)} & \frac{(t_9-t_5)(t_9-t_4)}{(t_9-t_3)(t_9-t_2)(t_9-t_1)} & \frac{(t_9-t_6)}{(t_9-t_5)(t_9-t_4)(t_9-t_3)(t_9-t_2)(t_9-t_1)} & \frac{(t_9-t_7)(t_9-t_6)}{(t_9-t_5)(t_9-t_4)(t_9-t_3)(t_9-t_2)(t_9-t_1)} & \frac{(t_9-t_8)}{(t_9-t_7)(t_9-t_6)(t_9-t_5)(t_9-t_4)(t_9-t_3)(t_9-t_2)(t_9-t_1)} &
\end{bmatrix}
$$

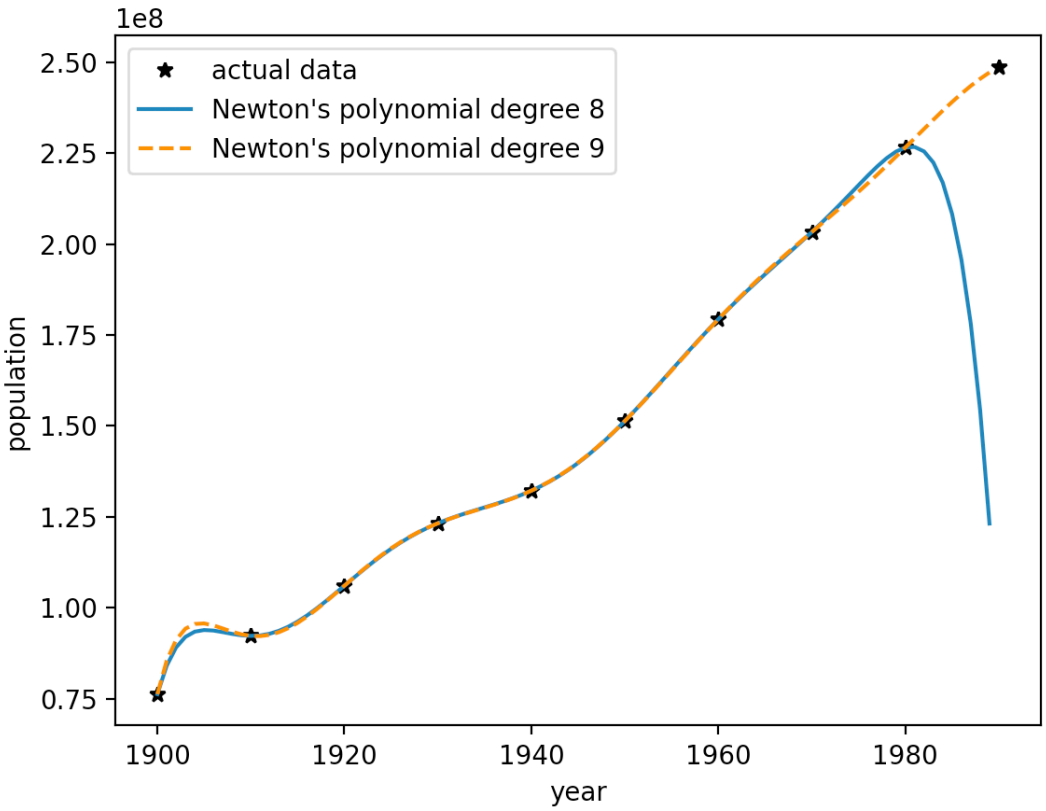
$$
\cdot \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_7 \\ x_8 \\ x_9 \end{bmatrix} = \begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \\ y_5 \\ y_6 \\ y_7 \\ y_8 \\ y_9 \end{bmatrix}
$$

$$P_8(t) = x_1 + x_2(t-t_1) + x_3(t-t_1)(t-t_2) + x_4(t-t_1)(t-t_2)(t-t_3)$$
$$+x_5(t-t_1)(t-t_2)(t-t_3)(t-t_4) + x_6(t-t_1)(t-t_2)(t-t_3)(t-t_4)(t-t_5)$$
$$+x_7(t-t_1)(t-t_2)(t-t_3)(t-t_4)(t-t_5)(t-t_6)$$
$$+x_8(t-t_1)(t-t_2)(t-t_3)(t-t_4)(t-t_5)(t-t_6)(t-t_7)$$
$$+x_9(t-t_1)(t-t_2)(t-t_3)(t-t_4)(t-t_5)(t-t_6)(t-t_7)(t-t_8)$$

I computed $P_9(t)$ by:

$$P_9(t) = P_8(t) + x_{10}(t-t_1)(t-t_2)(t-t_3)(t-t_4)(t-t_5)(t-t_6)(t-t_7)(t-t_8)(t-t_9),$$

where $x_{10} = \dfrac{y_{10} - P_8(t_{10})}{(t_{10}-t_1)(t_{10}-t_2)(t_{10}-t_3)(t_{10}-t_4)(t_{10}-t_5)(t_{10}-t_6)(t_{10}-t_7)(t_{10}-t_8)(t_{10}-t_9)}$



h)

```
coefficient for part b: [-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]
coefficient for part h: [-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]
```

```
coefficient for part b: [-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]
coefficient for round -1: [-3.15179609e+08  1.89175532e+08  6.06290162e+08 -3.42668345e+08
 -3.74614192e+08  1.82527046e+08  1.02716254e+08  4.61307824e+07
  1.32164570e+08]
coefficient for round -2: [-3.15174847e+08  1.89176686e+08  6.06281387e+08 -3.42670080e+08
 -3.74609280e+08  1.82527600e+08  1.02715340e+08  4.61307943e+07
  1.32164600e+08]
coefficient for round -3: [-3.15164444e+08  1.89160432e+08  6.06264889e+08 -3.42646044e+08
 -3.74601778e+08  1.82519822e+08  1.02713333e+08  4.61307905e+07
  1.32165000e+08]
coefficient for round -4: [-3.16058413e+08  1.89017397e+08  6.07872000e+08 -3.42371556e+08
 -3.75446667e+08  1.82365778e+08  1.02848079e+08  4.61533810e+07
  1.32160000e+08]
coefficient for round -5: [-3.10288254e+08  1.89358730e+08  5.96906667e+08 -3.43182222e+08
 -3.69080000e+08  1.82911111e+08  1.01611587e+08  4.60623810e+07
  1.32200000e+08]
coefficient for part h: [-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]
coefficient for round -7: [ 1.46285714e+08  8.12698413e+07 -1.28000000e+08 -1.56444444e+08
 -7.86666667e+07  9.55555556e+07  8.53809524e+07  5.46190476e+07
  1.30000000e+08]
```

I also ploted data with round from precisions from -1 to -7. We can see the more precise the data is, the more similar the coefficients will be. The error bound should support the observation above, since:

$$\frac{\|\Delta \text{coef}\|}{\|\text{coef}\|} \leq \text{cond}(\text{matrix}) \frac{\|\Delta y\|}{\|y\|}$$

$$\|\Delta \text{coef}\| \leq \text{cond}(\text{matrix}) \frac{\|\Delta y\|}{\|y\|} \cdot \|\Delta \text{coef}\|$$

```
delta coef from -1 to -7)
2.8180e+04
2.3897e+05
3.0932e+06
2.7586e+07
2.4190e+08
2.6428e+09
2.6428e+09
```

This shows that the norm of difference between actual data and rounded data is getting larger as the data becomes less precise. (the "distance" is getting further)

The actual norm of difference between actual data and rounded data from -1 to -7 (precision):

```
1.369136e+03
1.268440e+04
4.365432e+04
2.029584e+06
2.029584e+06
4.566257e+07
9.455073e+08
```