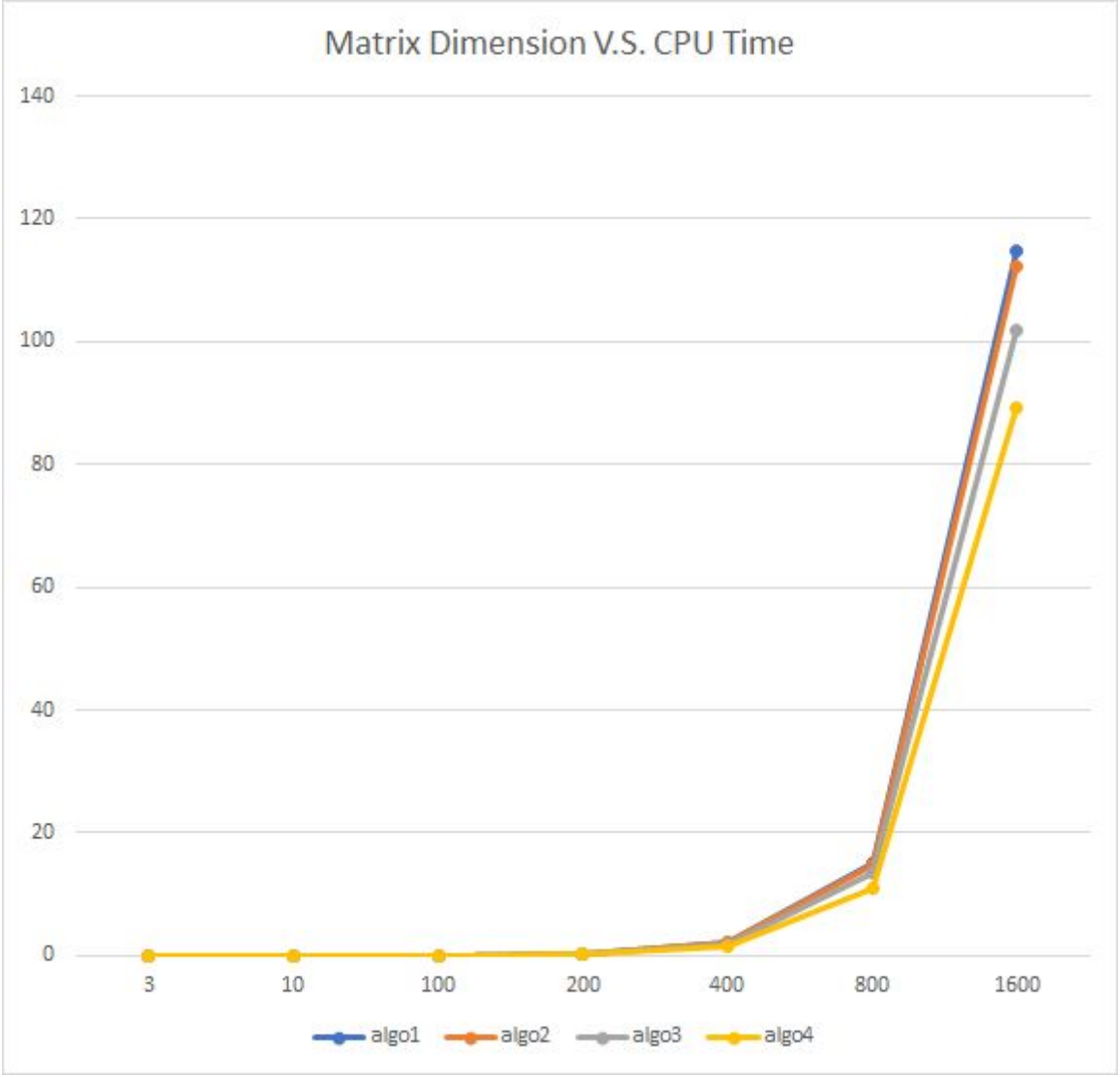
106033233 周聖諺

1. CPU Time vs Dimension:
   Generally, the time complexity of all 4 algorithms is $O(n^3)$. As a result, there is no dramatic difference between the 4 algorithms. Following is the result:

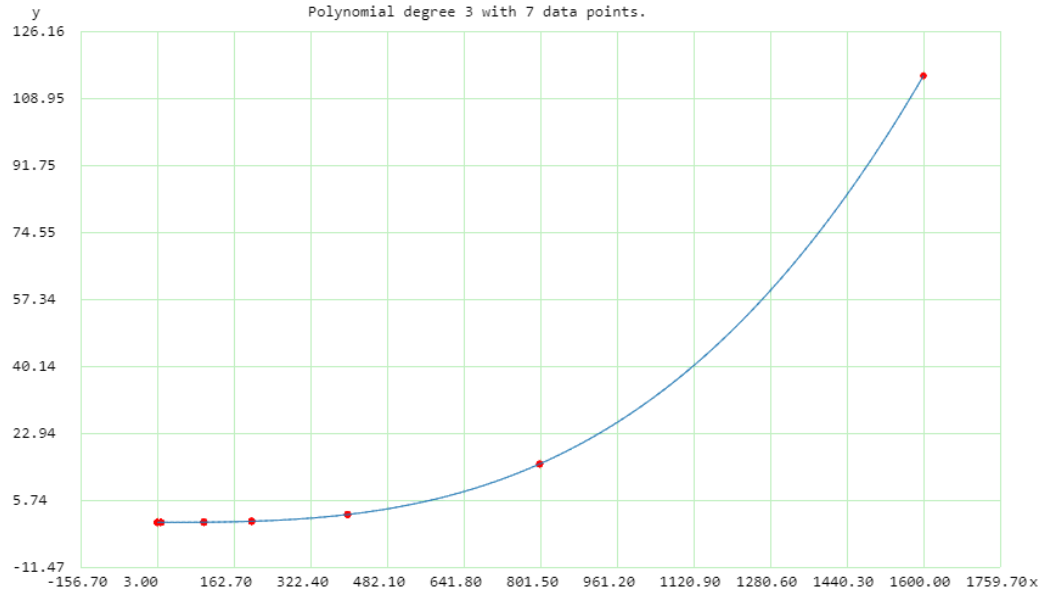
	3	10	100	200	400	800	1600
algo1	0.002	0.002	0.047	0.297	1.994	14.988	114.687
algo2	0.002	0.002	0.047	0.285	1.988	14.827	112.283
algo3	0.002	0.002	0.044	0.256	1.756	13.487	101.697
algo4	0.002	0.002	0.039	0.222	1.504	11.123	89.169

Unit: sec



(1) Algorithm 1: Classic Gram-Schmidt

Classic Gram-Schmidt is the slowest one because it conducts 5 vector multiplication in a double loop.



Fitting Curve:

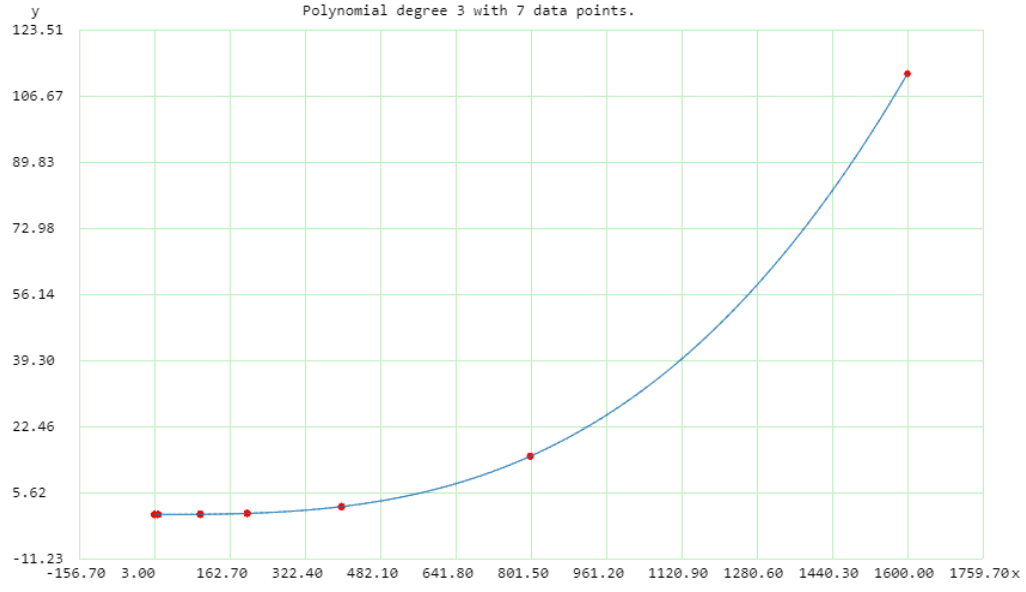
$$f(x) = 8.72 * 10^{-3} + -1.97 * 10^{-4} * x + 2.39 * 10^{-6} * x^2 + 2.65 * 10^{-8} * x^3$$

Complexity:

$$f(n) = 5n^3/2 + 7n^2/2 = O(n^3)$$

(2) Algorithm 2

Modified Gram-Schmidt algorithm is the second slowest one because it still conducts 5 vector multiplication in a double loop. However, it eliminates the minus operation in the classic algorithm. It is a little bit faster than the classic algorithm. The coefficient of x^3 terms is smaller than the classic Gram-Schmidt algorithm.



Fitting Curve:

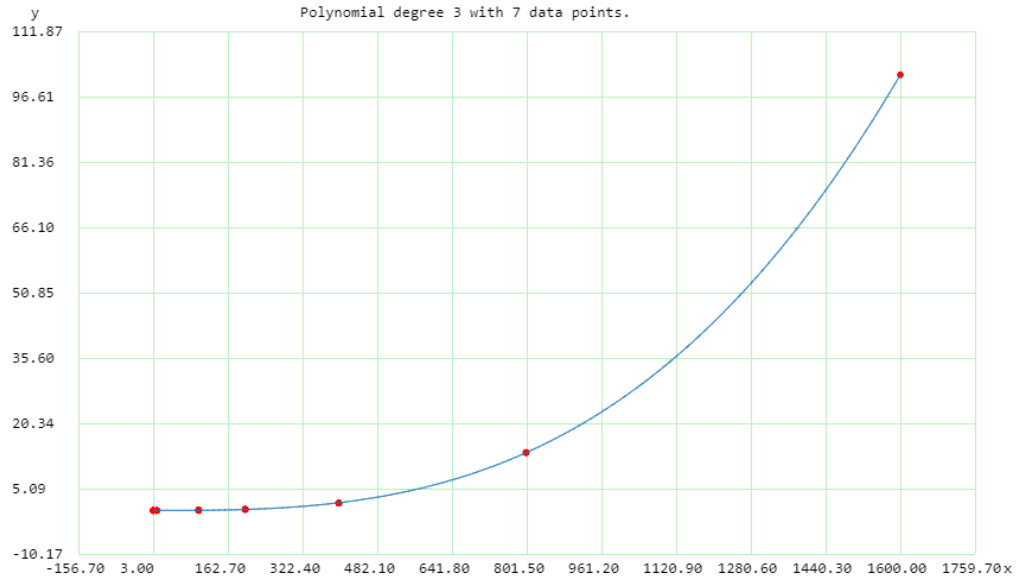
$f(x) = 1.04 * 10^{-2} + -3.30 * 10^{-4} * x + 3.04 * 10^{-6} * x^2 + 2.56 * 10^{-8} * x^3$

Complexity:

$f(n) = 5n^3/2 + 5n^2/2 = O(n^3)$

(3) Algorithm 3

The algorithm 3 is faster than algorithm 2 because it moves G_i to the right-hand side and two integers multiply first. It eliminates one more multiplication in the double loop. The coefficient of x^3 terms is also smaller than algorithm 2.



Fitting Curve:

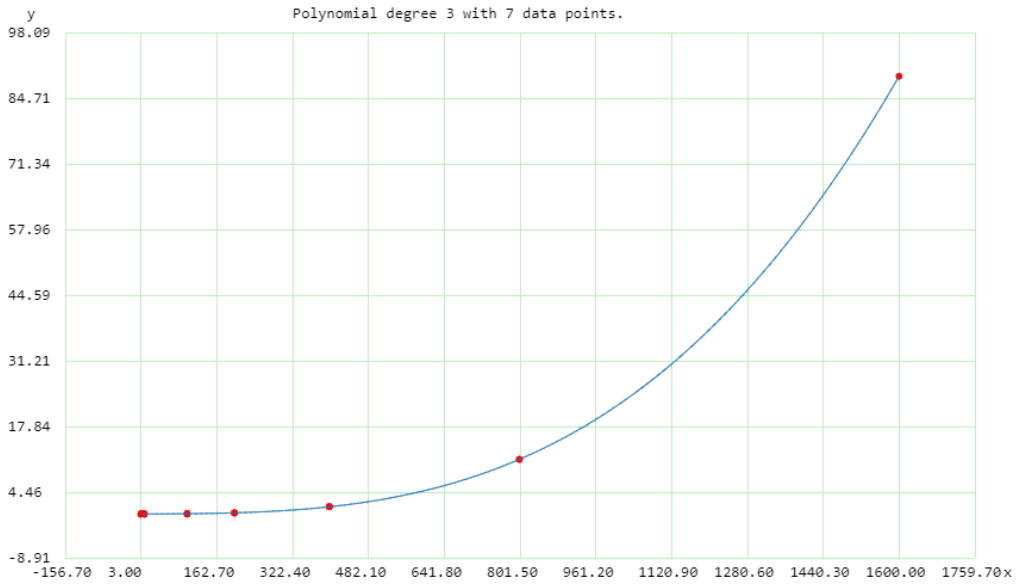
$f(x) = 1.97 * 10^{-2} + -5.93 * 10^{-4} * x + 3.44 * 10^{-6} * x^2 + 2.29 * 10^{-8} * x^3$

Complexity:

$f(n) = 2n^3 + 2n^2 = O(n^3)$

(4) Algorithm 4

The algorithm 4 is the fastest because it moves $G_i^T G_i$ to the outer loop. It eliminates one more multiplication in the double loop. The coefficient of x^3 terms is the smallest.



Fitting Curve:

$$f(x) = -1.25 * 10^{-2} + 6.29 * 10^{-4} * x - 1.19 * 10^{-6} * x^2 + 2.22 * 10^{-8} * x^3$$

Complexity:

$$f(n) = 3n^3/2 + 5n^2/2 = O(n^3)$$

2. Sigma vs Dimension:
   Actually, the most interesting thing is Sigma of the classic Gram-Schmidt algorithm becomes an explosion when the dimension = 100. The following is the result:

	mat3	mat4	mat5	mat6	mat7	mat8	mat9
algo1	1.78E-15	2.09E-10	5.76E+08	2.48E+10	3.55E+11	4.72E+13	3.84E+14
algo2	0	3.54E-13	5.62E-10	4.76E-09	4.10E-08	4.20E-07	3.92E-06
algo3	0	1.23E-12	2.62E-09	2.72E-08	3.22E-07	3.42E-06	4.21E-05
algo4	0	1.23E-12	2.62E-09	2.72E-08	3.22E-07	3.42E-06	4.21E-05

Besides, when I printed the first row of G for the classic algorithm and algorithm 1. The gap between the two series gets larger as the series grows.

Classic Algorthim Output Series:
200 -4.86685 0.25088 0.254019 0.257207 0.260444 0.263731 0.267067 0.270452 0.273887 0.277372 0.280906 0.284489 0.288121 0.291801 0.295529 0.299302 0.303131 0.306967 … -352.61 -362.025 -370.495 -378.008 -384.555 … -206.386 -203.7 -202.066

(Marked blue are the SAME.)

Algorthim 2 Output Series:
200 -4.86685 0.25088 0.254019 0.257207 0.260444 0.263731 0.267067 0.270452 0.273887 0.277372 0.280906 0.284489 0.288121 0.291801 0.295529 0.299304 0.303124 0.30699 … 0.439097 0.440846 0.442301 0.443441 0.444245 … 0.0768289 0.0580262 0.0389494 0.0196059

Classic Algorithm Minus Algorithm 2 Series:
0 0 6.39488e-14 7.89369e-14 -6.39544e-13 2.04481e-12 -5.44087e-12 1.655e-11 -5.83128e-11 2.11428e-10 -7.69545e-10 2.80358e-09 -1.02069e-08 3.71392e-08 -1.35094e-07 4.91272e-07 -1.78602e-06 6.4912e-06 -2.35853e-05 … -161.384 -179.13 -196.682 -213.934 -230.79 … -206.444 -203.739 -202.085

As the series show, the computation error propagates and become bigger and bigger. The reason is the binary system cannot represent a decimal number precisely. It would remain a small deviation (Rounding Errors). The Gram-Schmidt algorithm always takes the previous computation result to compute the next row. As a result, the deviation would be propagated and become larger. Another reason is when calculating the orthogonal span, the computer cannot represent precisely numbers. It causes the span is not a real orthogonal matrix and it causes deviation of the project of the next vector. To fix the error, the modified Gram-Schmidt algorithm replaces A_k^T with G_k^T to enhance the numeric stability. The formula of Modified Gram-Schmidt is

$$\mathbf{u}_k = \mathbf{v}_k - \operatorname{proj}_{\mathbf{u}_1}(\mathbf{v}_k) - \operatorname{proj}_{\mathbf{u}_2}(\mathbf{v}_k) - \cdots - \operatorname{proj}_{\mathbf{u}_{k-1}}(\mathbf{v}_k),$$