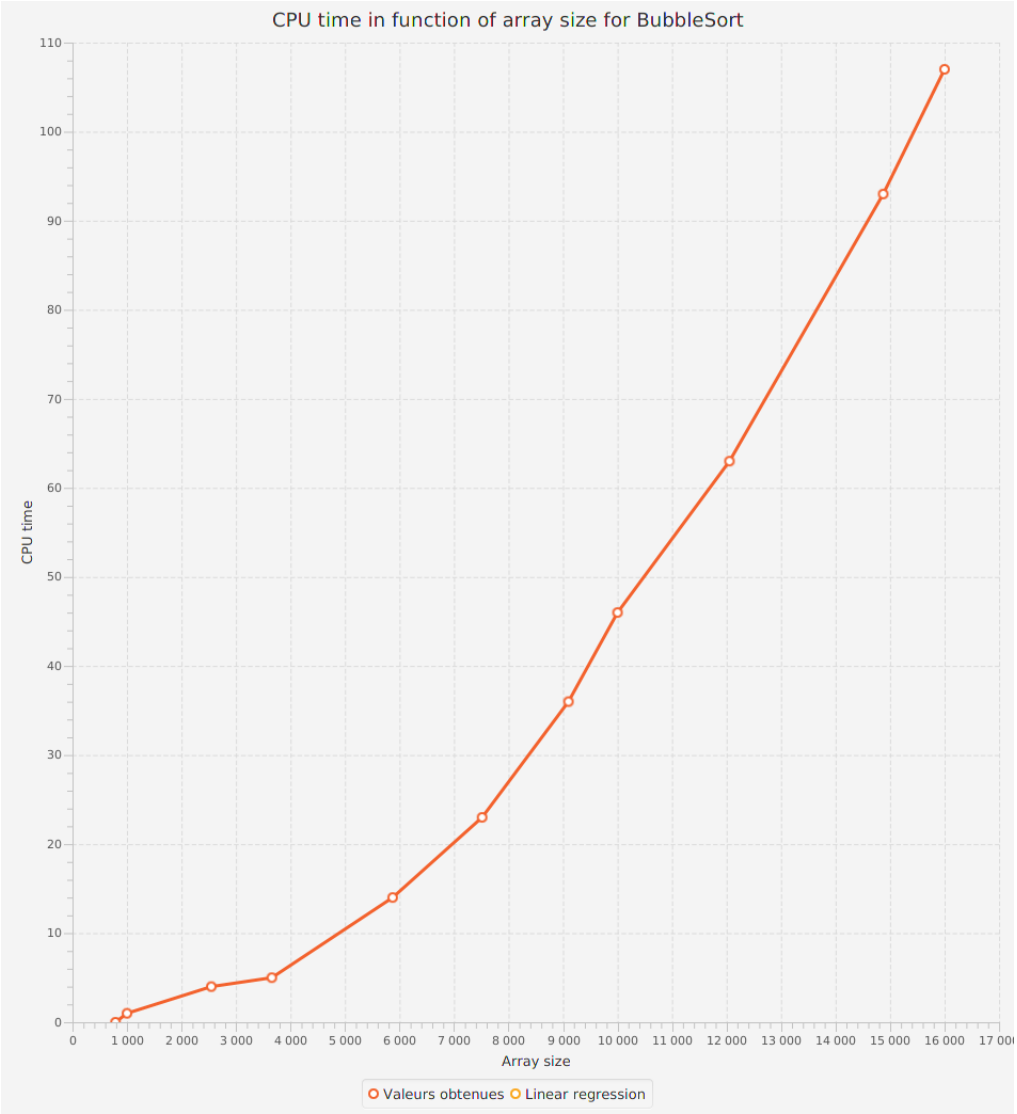
1 Première Étape

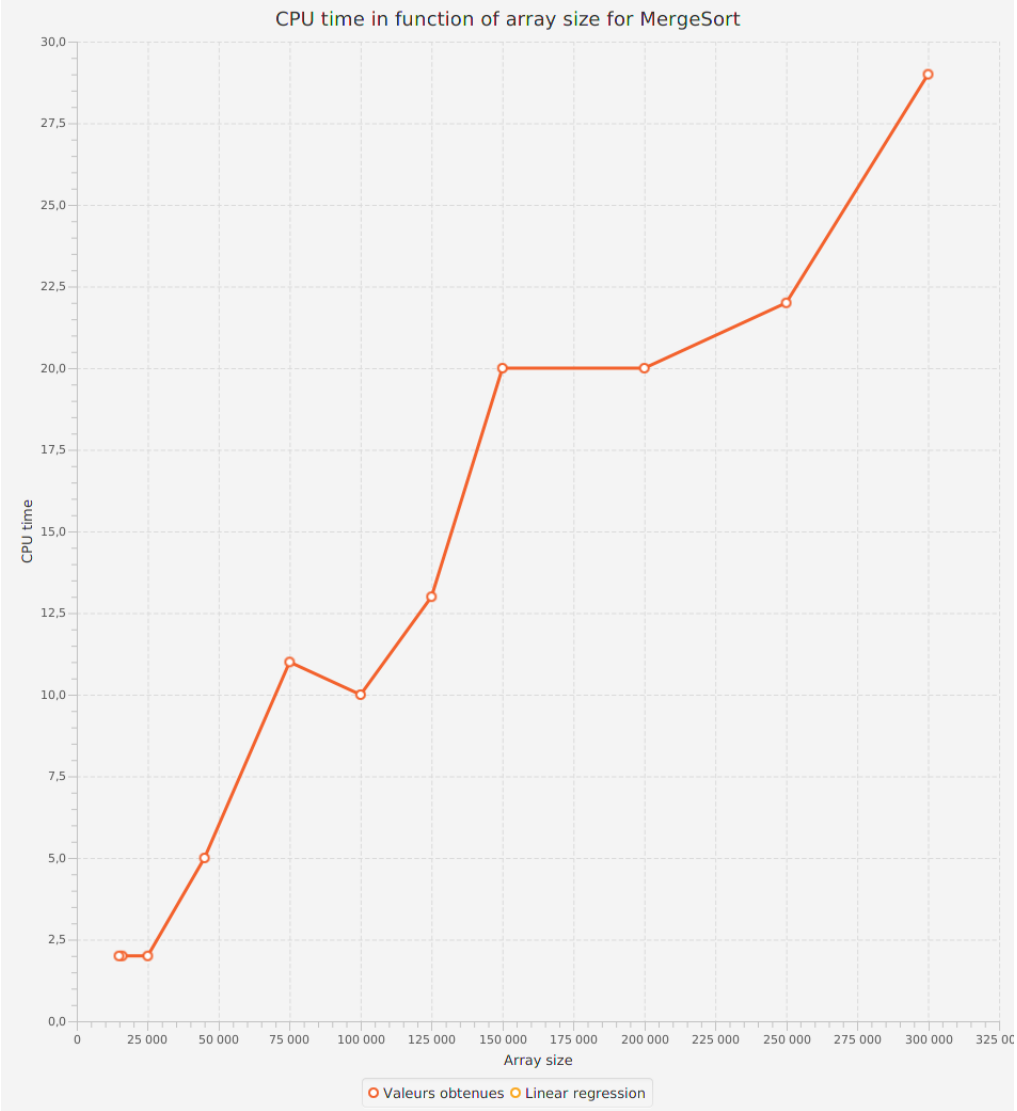
Durant cette étape je me suis concentré sur l'affichage graphique des temps CPU des différents algorithmes testés.

1.1 Bubble Sort $O(n^2)$



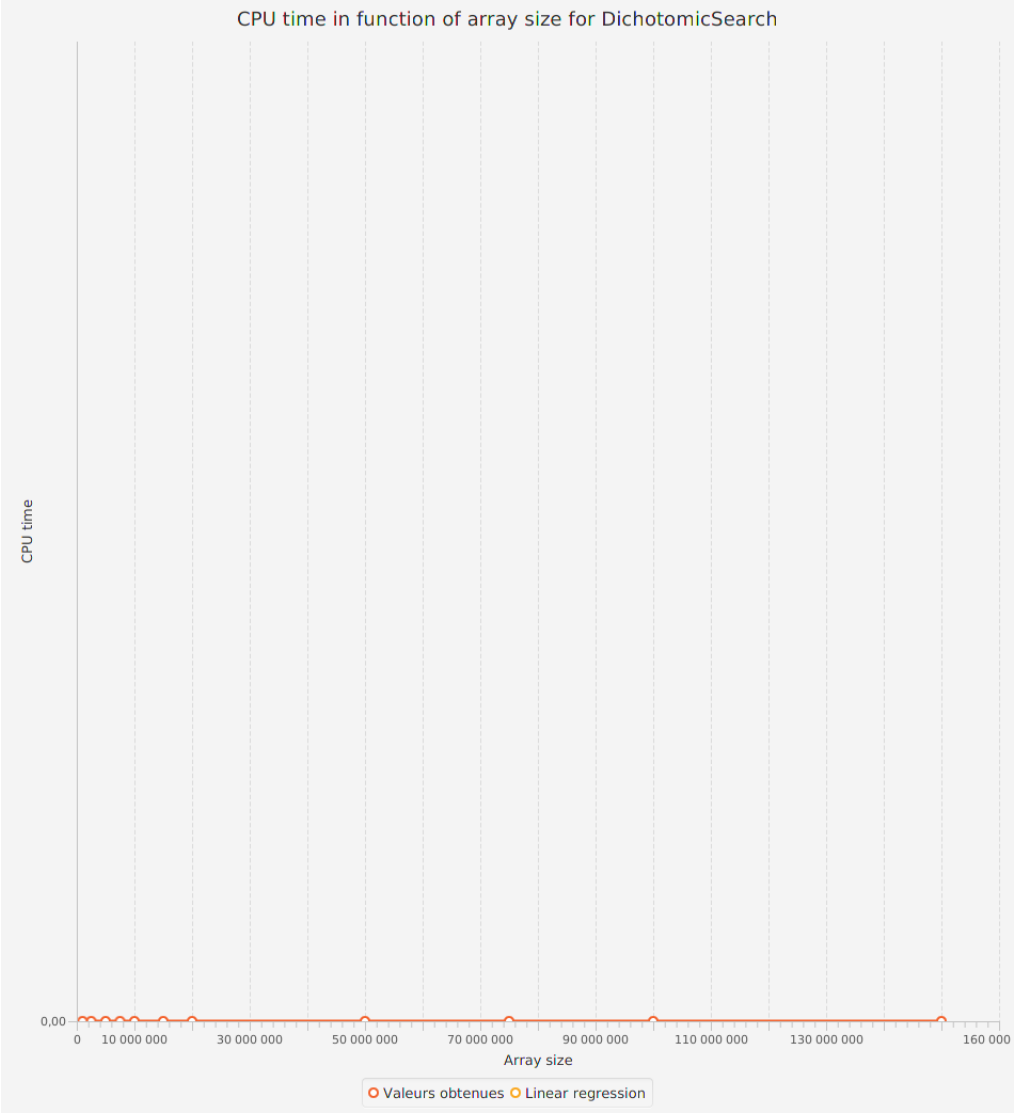
On remarque que la courbe obtenue à bien l'allure attendu ($O(n^2)$)

1.2 Merge Sort $O(nlog(n)))$



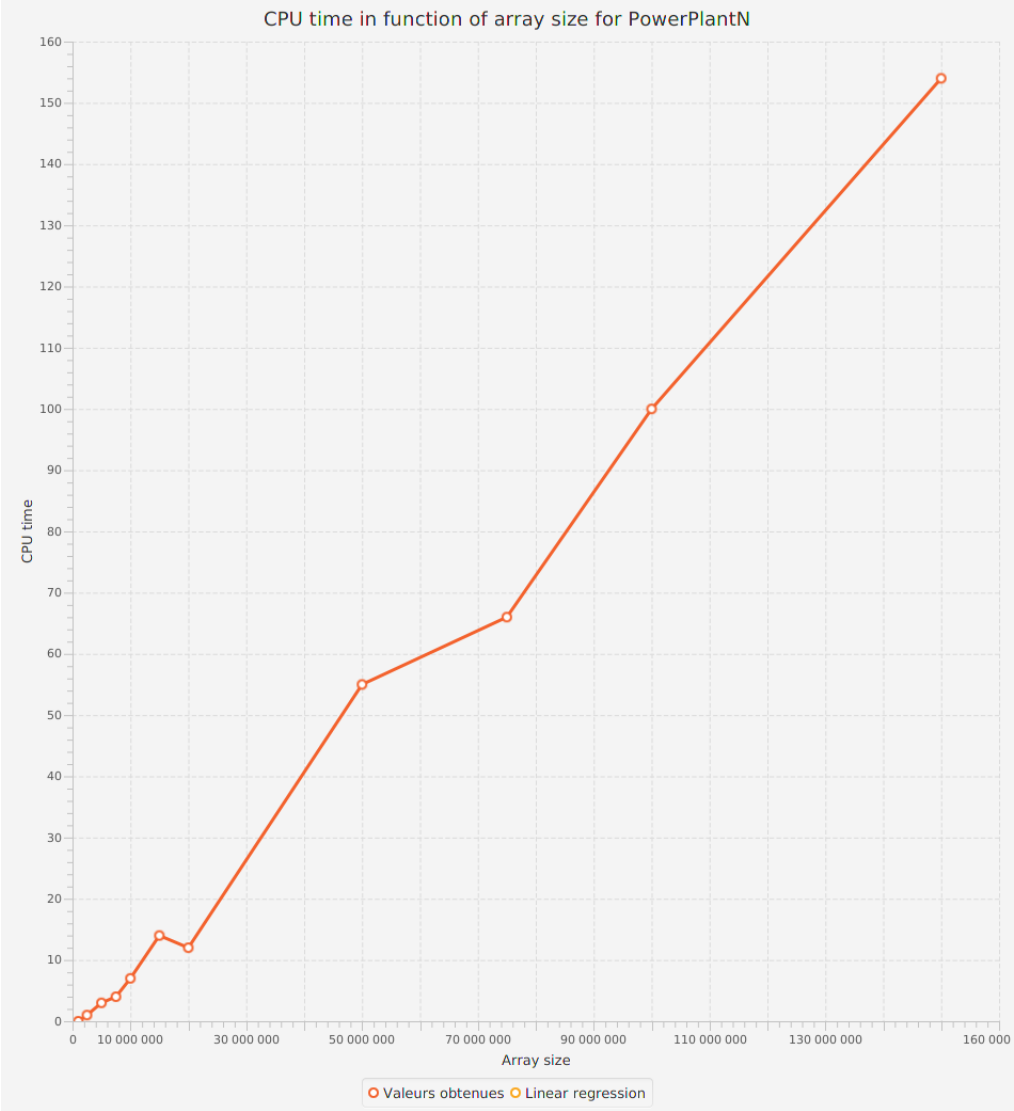
On remarque que la courbe obtenue à bien l'allure attendu ($O(nlog(n))$

1.3 Dichotomic Search $O(log(n))$



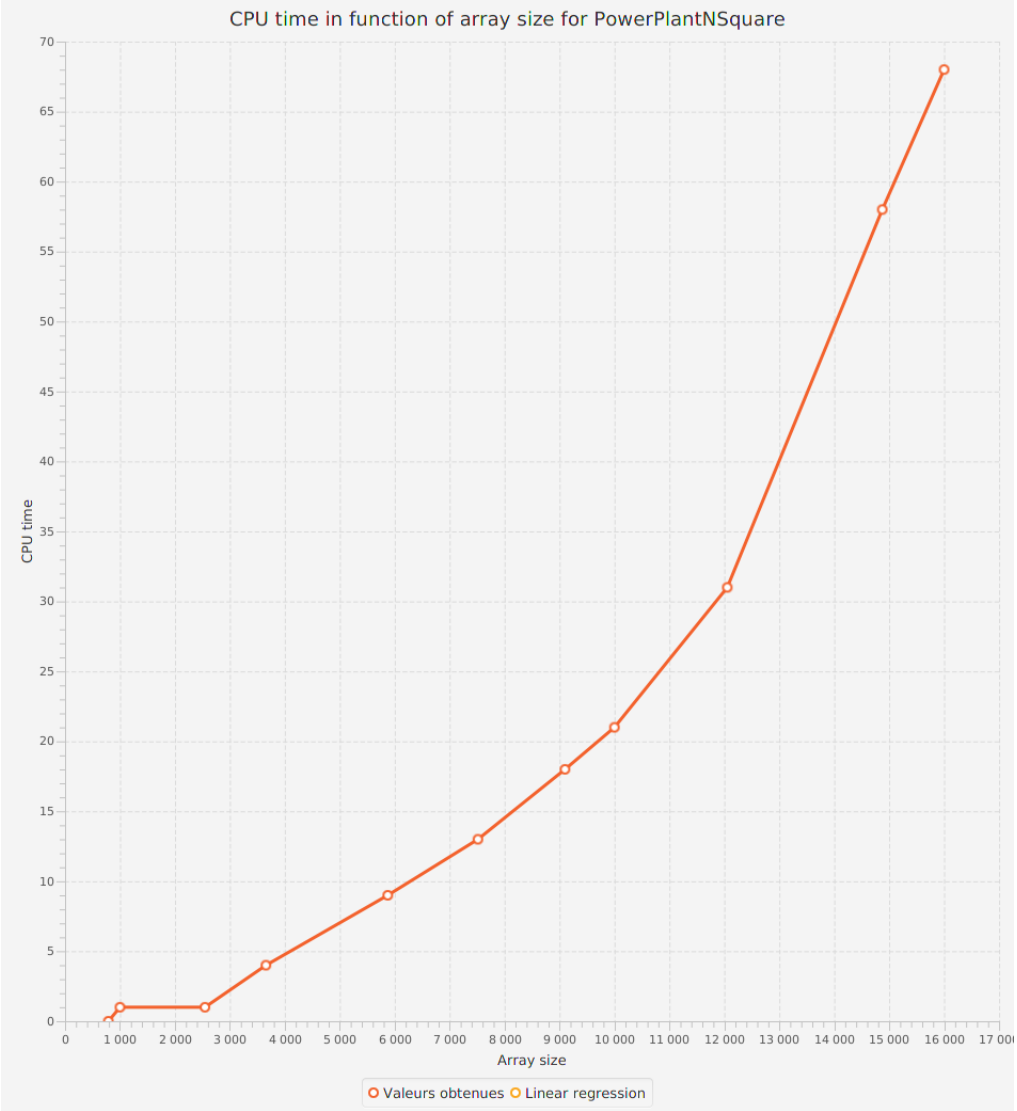
Ici un problème se pose, en effet l'algorithme testé étant en $O(log(n))$, le temps nécessaire a son execution n'est pas mesurable, en effet pour un table de taille 2 000 000 000 le programme dure moins de $1ms$.

1.4 Power Plant $O(n)$



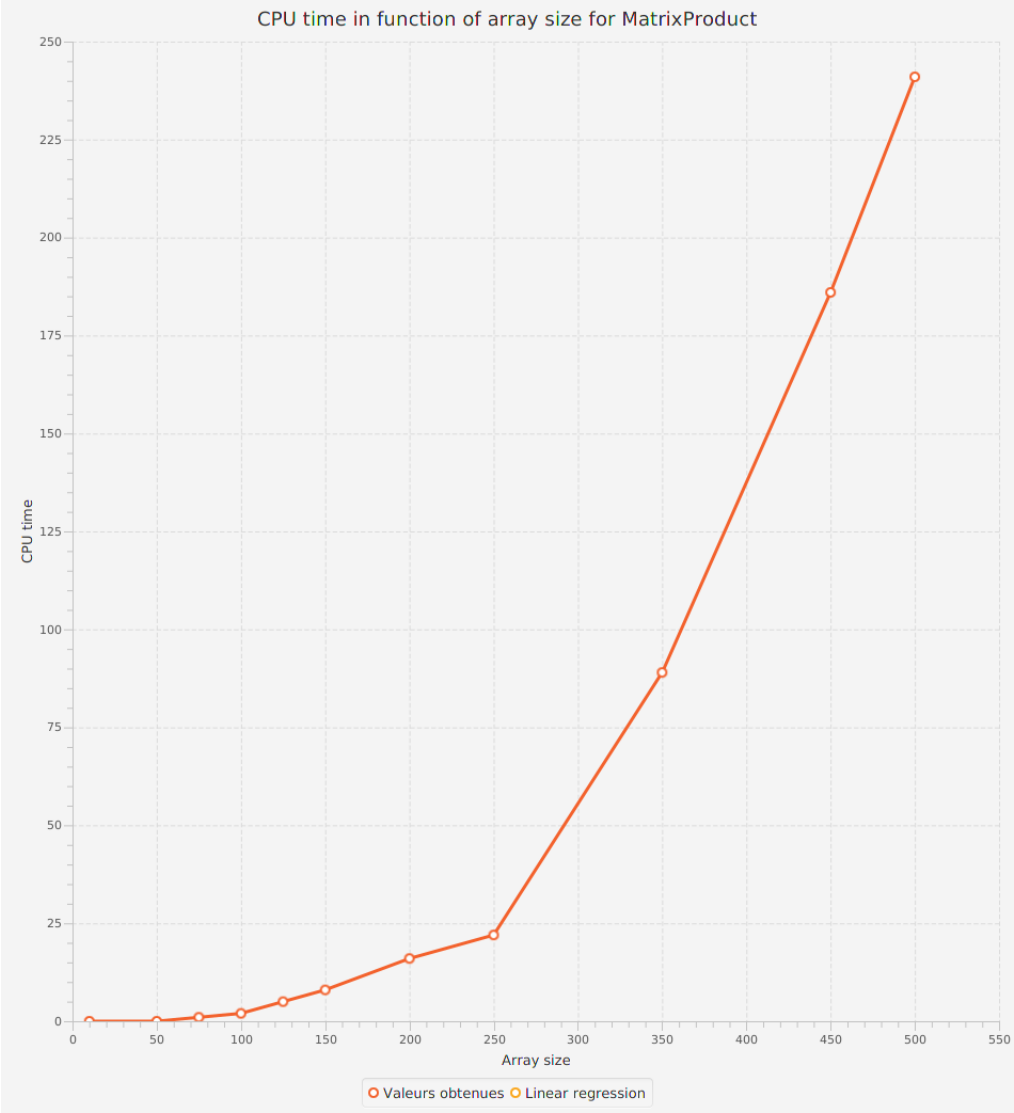
On remarque que la courbe obtenue à bien l'allure attendu ($O(n)$)

1.5 Power Plant $O(n^2)$



On remarque que la courbe obtenue à bien l'allure attendu ($O(n^2)$)

1.6 Matrix Product $O(n^3)$



On remarque que la courbe obtenue à bien l'allure attendu ($O(n^3)$)

2 Deuxième Partie

Dans cette deuxième partie, j'ai d'abord réaliser la régression linéaire des différents algorithme, puis affiché leurs résultats sans/avec scaling et avec/sans régularisation

2.1 Sans Régularisation

2.1.1 Bubble Sort $O(n^2)$

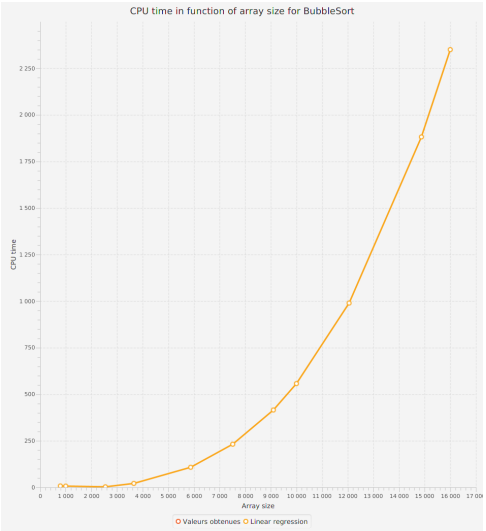


FIGURE 1 – Sans scaling

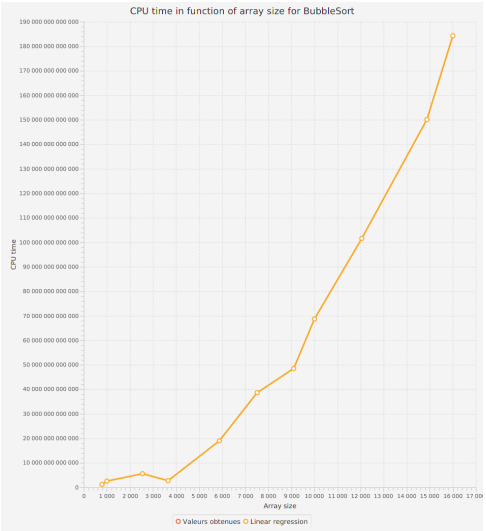


FIGURE 2 – Avec Scaling

$H(x) = 698.8915972709656 + -148.09268846735358O(log(n))) + 2.04268618079368O3O(n)) + -0.26362952370982384O(nlog(n))) + 3.159411356037367E-4O(n^2)) + -3.000054987767342E-5O(n^2log(n))) + 6.034665528924102E - 10O(n^3))$

2.1.2 Merge Sort $O(nlog(n)))$

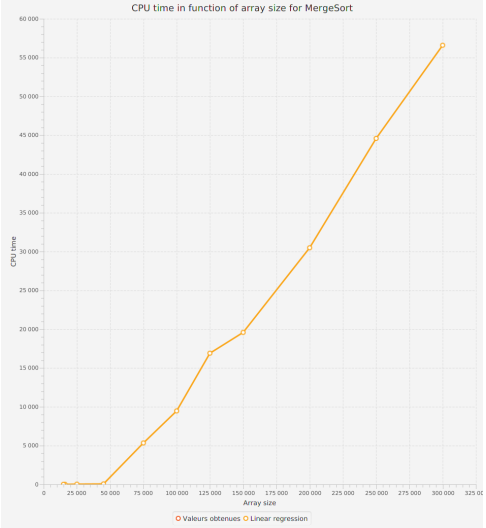


FIGURE 3 – Sans scaling

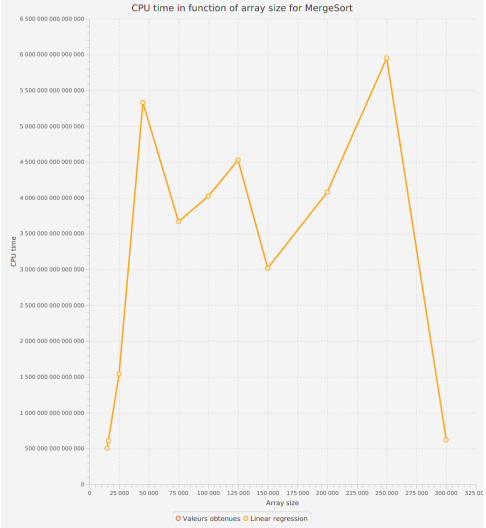


FIGURE 4 – Avec Scaling

2.1.3 Dichotomic Search $O(log(n))$

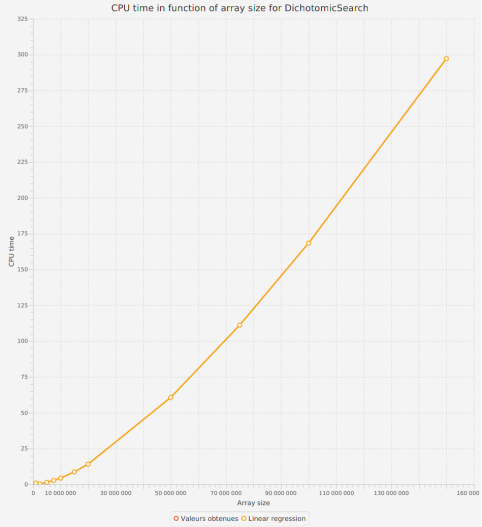


FIGURE 5 – Sans scaling

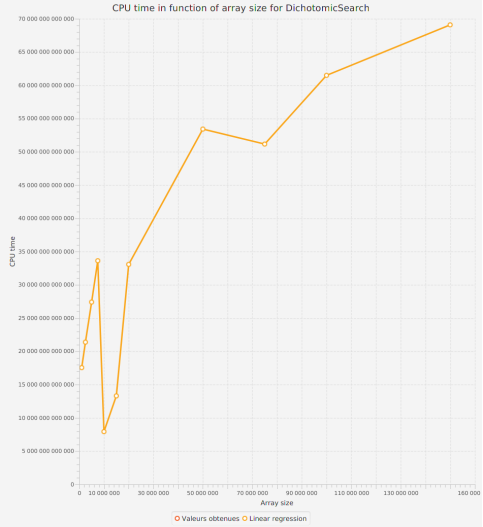


FIGURE 6 – Avec Scaling

2.1.4 Power Plant $O(n)$

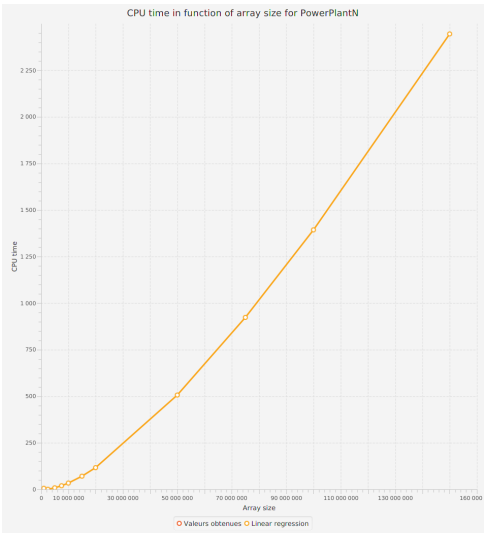


FIGURE 7 – Sans scaling

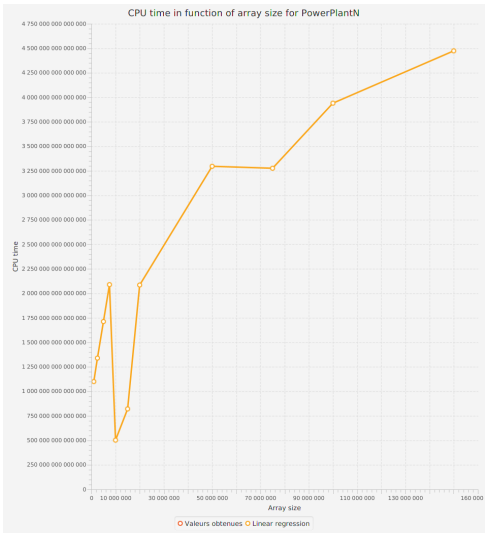


FIGURE 8 – Avec Scaling

2.1.5 Power Plant $O(n^2)$

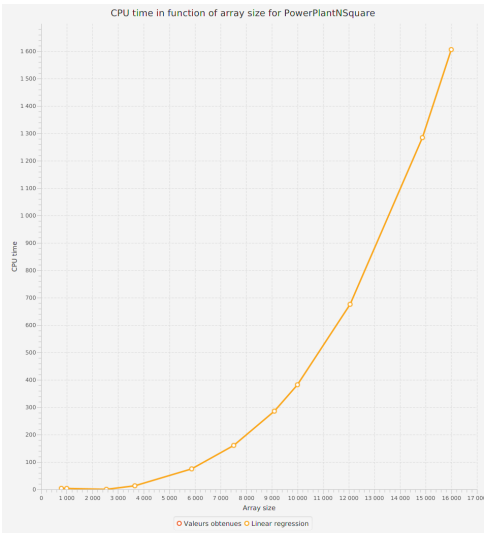


FIGURE 9 – Sans scaling

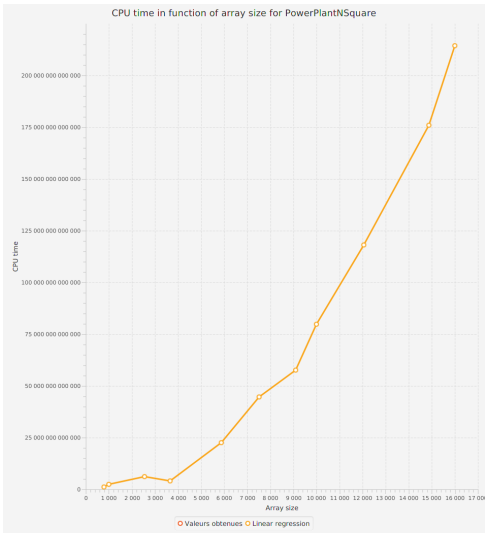


FIGURE 10 – Avec Scaling

2.1.6 Matrix Product $O(n^3)$

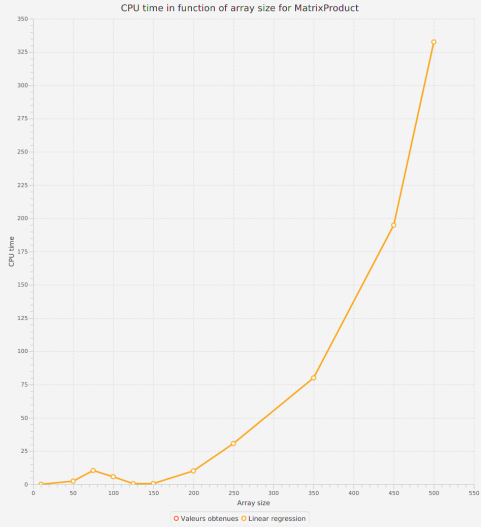


FIGURE 11 – Sans scaling

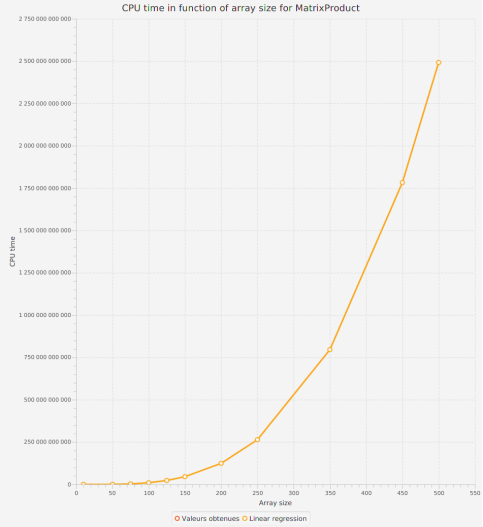


FIGURE 12 – Avec Scaling

2.1.7 Regression linéaire obtenues

SANS SCALING

—— Merge sort ——

The linear regression gives : $H(x) = 7141.655826851726 + -917.7796166948974 O(log(n))) + 0.8089920593192801 O(n)) + -0.07492421518111314 O(nlog(n))) + 5.771846820823434 E-6 O(n^2)) + -4.288045779210145 E-7 O(n^2 log(n))) + 4.6324605699258 E-13 O(n^3))$

Average complexity of the current Algortihm : $O(n)$

—— Bubble sort ——

The linear regression gives : $H(x) = 414.5926007628441 + -63.61315888911486 O(log(n))) + -0.2071383105358109 O(n)) + 0.04095703620259883 O(nlog(n))) + -1.40417155201078 E^4 O(n^2)) + 1.4068810004097543 E^5 O(n^2 log(n))) + -3.821037078188599 E^1 0 O(n^3))$

Average complexity of the current Algortihm : $O(nlog(n))$

—— Dichotomic search ——

The linear regression gives : $H(x) = 40.59673786715247 + -3.0423296450748767 O(log(n))) + 1.3937345713036534 E^5 O(n)) + -8.396535905756172 E^7 O(nlog(n))) + 2.565161112789591 E^1 3 O(n^2)) + -1.3198235426667415 E^1 4 O(n^2 log(n))) + 3.442967558179723 E^2 3 O(n^3))$

Average complexity of the current Algortihm : $O(n)$

—— PowerPlant N ——

The linear regression gives : $H(x) = 357.56172717921436 + -28.475538978469558 O(log(n))) + 2.774238285034869 E^4 O(n)) + -1.7351438081070203 E^5 O(nlog(n))) + 8.601751000340835 E^1 2 O(n^2)) + -4.51131381063441 E^1 3 O(n^2 log(n))) + 1.5643505274442346 E^2 1 O(n^3))$

Average complexity of the current Algortihm : $O(n)$

—— PowerPlant N^2 ——

The linear regression gives : $H(x) = 433.53659324347973 + -85.12150310538709 O(log(n))) + 0.7954701127455337 O(n)) + -0.09656808889303647 O(nlog(n))) + 7.316775826637922 E^5 O(n^2)) + -6.602137807343311 E^6 O(n^2 log(n))) + 7.305366713986202 E^1 1 O(n^3))$

Average complexity of the current Algortihm : $O(n)$

—— Matrix Product ——

The linear regression gives : $H(x) = 109.530443161726 + -120.6629607565701 O(log(n))) + 33.56979820970446 O(n)) + -7.834374452009797 O(nlog(n))) + 0.19197909046488348 O(n^2)) + -0.026858456595618918 O(n^2 log(n))) + 1.7692029993465752 E^- 5 O(n^3))$

Average complexity of the current Algortihm : $O(n)$

AVEC SCALING

—— Merge sort ——

The linear regression gives : $H(x) = 19.54545454545847 + -911.5904337726533 O(log(n))) + 43644.64355766773 O(n)) + -47777.954013586044 O(nlog(n))) + 25991.1073846817 O(n^2)) + -20245.193412065506 O(n^2 log(n))) + -676.7542952522635 O(n^3))$

Average complexity of the current Algortihm : $O(n)$

—— Bubble sort ——

The linear regression gives : $H(x) = 53.27272727273213 + -2000.35363150388 O(log(n))) + 139980.6954216957 O(n)) + -178024.8242712021 O(nlog(n))) + 377555.65513134 O(n^2)) + -350028.4794483185 O(n^2 log(n))) + 12693.71117207408 O(n^3))$

Average complexity of the current Algortihm : $O(n^2)$

—— Dichotomic search ——

The linear regression gives : $H(x) = 0.18181818181812015 + -10.174226371278564 O(log(n))) + 410.42379777133465 O(n)) + -379.11286762775853 O(nlog(n))) + -2152.2377136163414 O(n^2)) + 2220.9187619192526 O(n^2 log(n))) + -90.81503783489461 O(n^3))$

Average complexity of the current Algortihm : $O(n^2 log(n))$

—— PowerPlant N ——

The linear regression gives : $H(x) = 42.72727272727132 + -153.60487488331273 O(log(n))) + 33199.89365148544 O(n)) + -38602.861510157585 O(nlog(n))) + 136439.0266137123 O(n^2)) + -134304.1528491974 O(n^2 log(n))) + 3580.593799352646 O(n^3))$

Average complexity of the current Algortihm : $O(n^2)$

—— PowerPlant N^2 ——

The linear regression gives : $H(x) = 16.909090909091475 + -335.79872394911945 O(log(n))) + 19412.687231063843 O(n)) + -23904.484145641327 O(nlog(n))) + 39788.07977628708 O(n^2)) + -35896.16054391861 O(n^2 log(n))) + 982.8768238574266 O(n^3))$

Average complexity of the current Algortihm : $O(n^2)$

—— Matrix Product ——

The linear regression gives : $H(x) = 51.90909090908882 + -694.975088275969 O(log(n))) + 24242.50492787361 O(n)) + -35940.10701084137 O(nlog(n))) + 75502.83763980865 O(n^2)) + -66558.43084335327 O(n^2 log(n))) + 3705.2446922063828 O(n^3))$

Average complexity of the current Algortihm : $O(n^2)$

2.2 Avec Régularisation

2.2.1 Bubble Sort $O(n^2)$

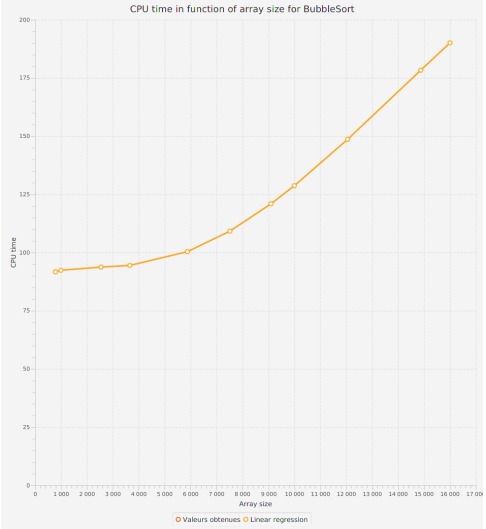


FIGURE 13 – Sans scaling

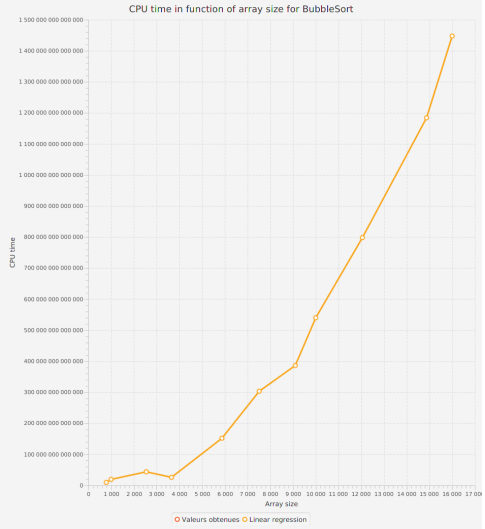


FIGURE 14 – Avec Scaling

2.2.2 Merge Sort $O(nlog(n)))$

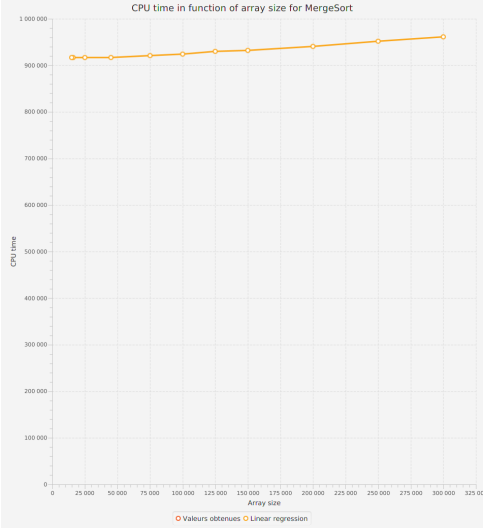


FIGURE 15 – Sans scaling

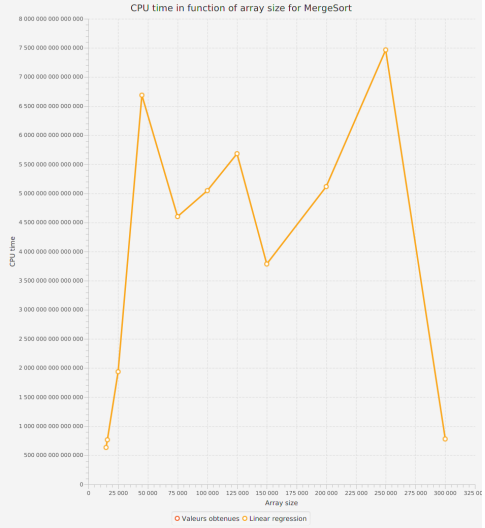


FIGURE 16 – Avec Scaling

2.2.3 Dichotomic Search $O(log(n))$

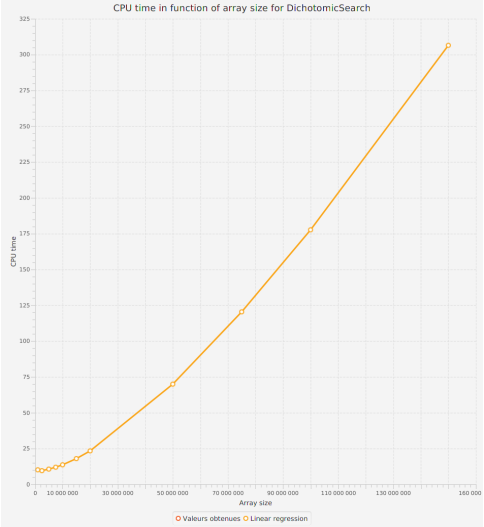


FIGURE 17 – Sans scaling

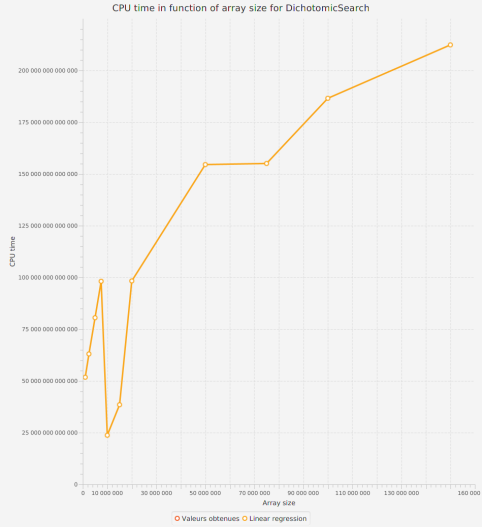


FIGURE 18 – Avec Scaling

2.2.4 Power Plant $O(n)$

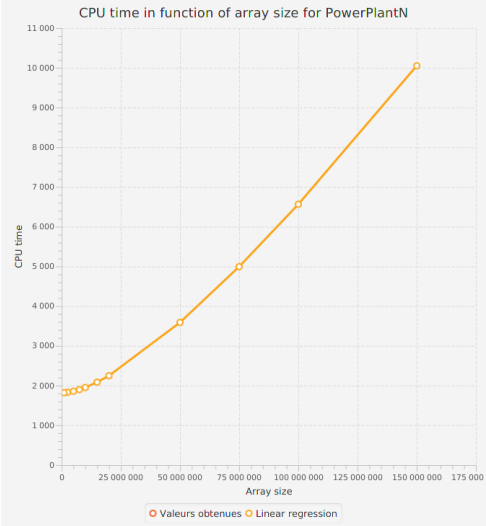


FIGURE 19 – Sans scaling

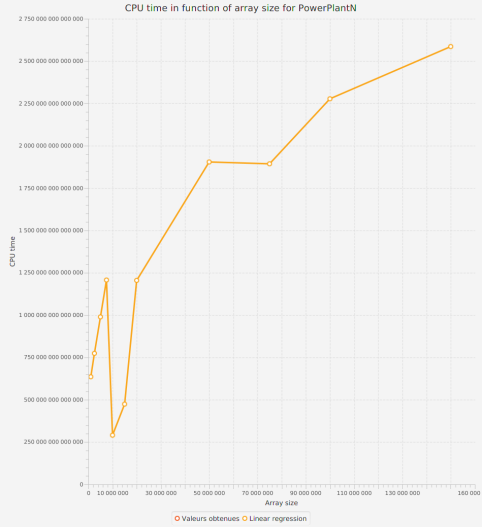


FIGURE 20 – Avec Scaling

2.2.5 Power Plant $O(n^2)$

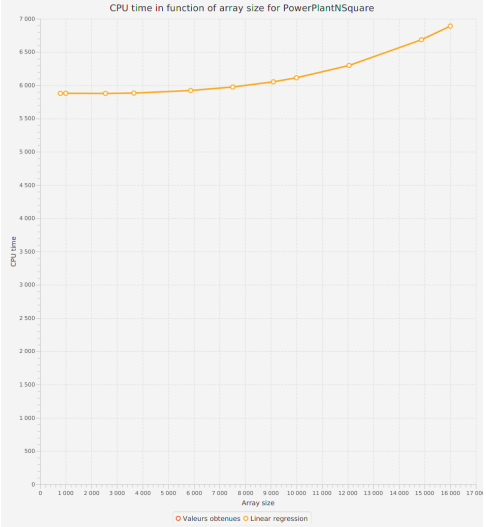


FIGURE 21 – Sans scaling

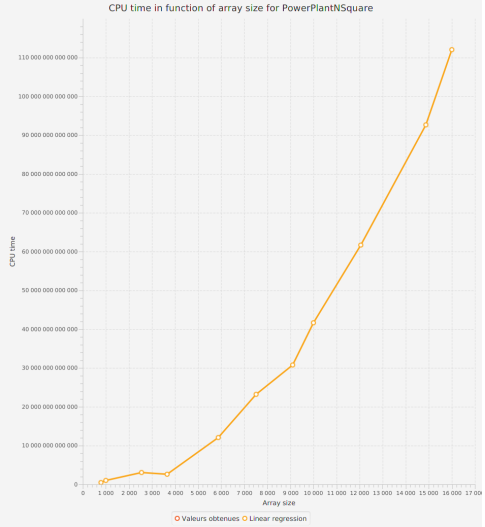


FIGURE 22 – Avec Scaling

2.2.6 Matrix Product $O(n^3)$

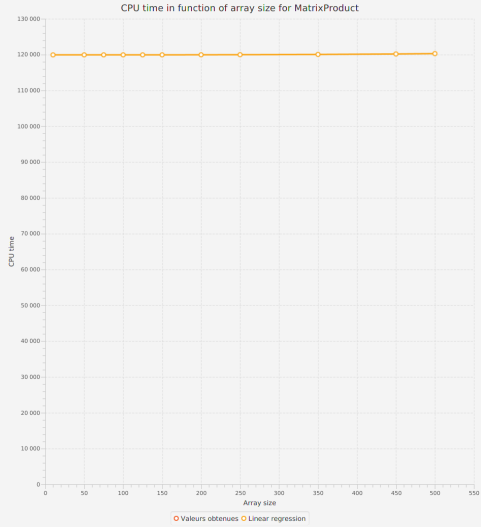


FIGURE 23 – Sans scaling

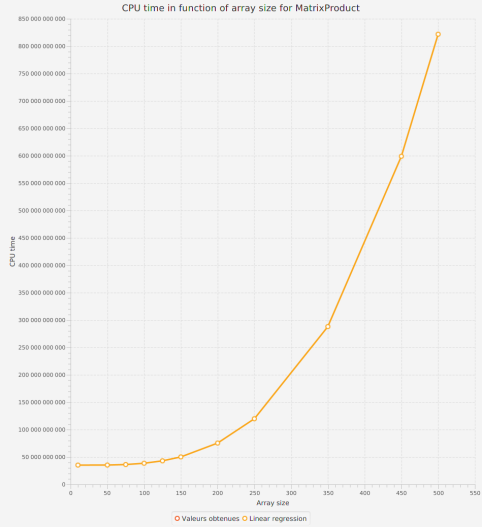


FIGURE 24 – Avec Scaling