UNIVERSIDAD DE GRANADA

ALGORITMICA

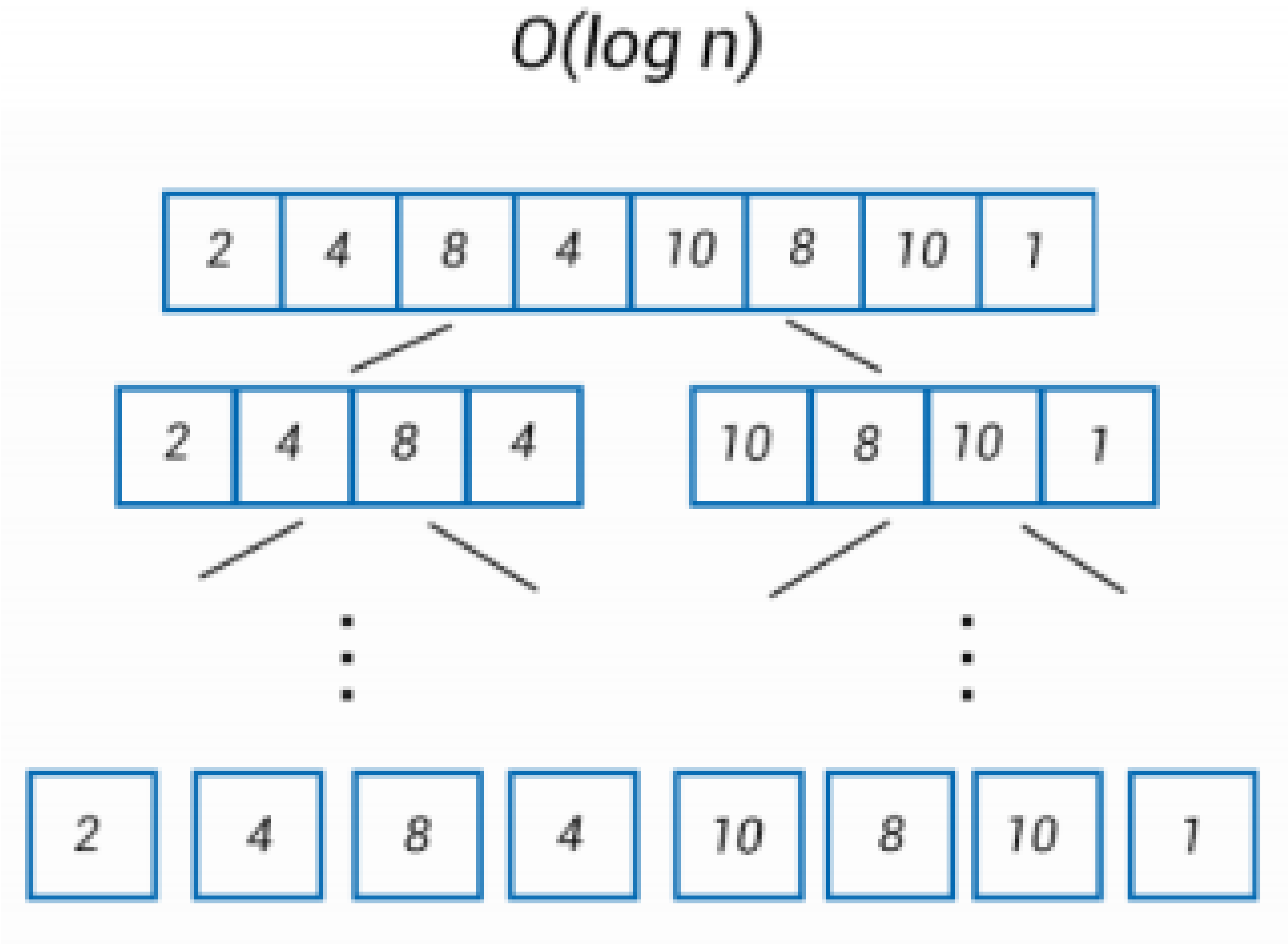
PRÁCTICA 2: ALGORITMOS DIVIDE Y VENCERÁS

INDICE

1. ALGORITMO CLASICO

```
bool yaExiste(vector<int> v, int x){
  bool encontrado = false;
  for(int i = 0; i < v.size() && !encontrado; i++)
    if(v[i] == x)
      encontrado = true;
  return encontrado;
}

vector<int> eliminarRepeticiones(vector<int> v){
  vector<int> v_res;
  for(int i = 0; i < v.size(); i++)
    if(!yaExiste(v_res, v[i]))
      v_res.push_back(v[i]);

  return v_res;
}
```

2. ALGORITMO DyV

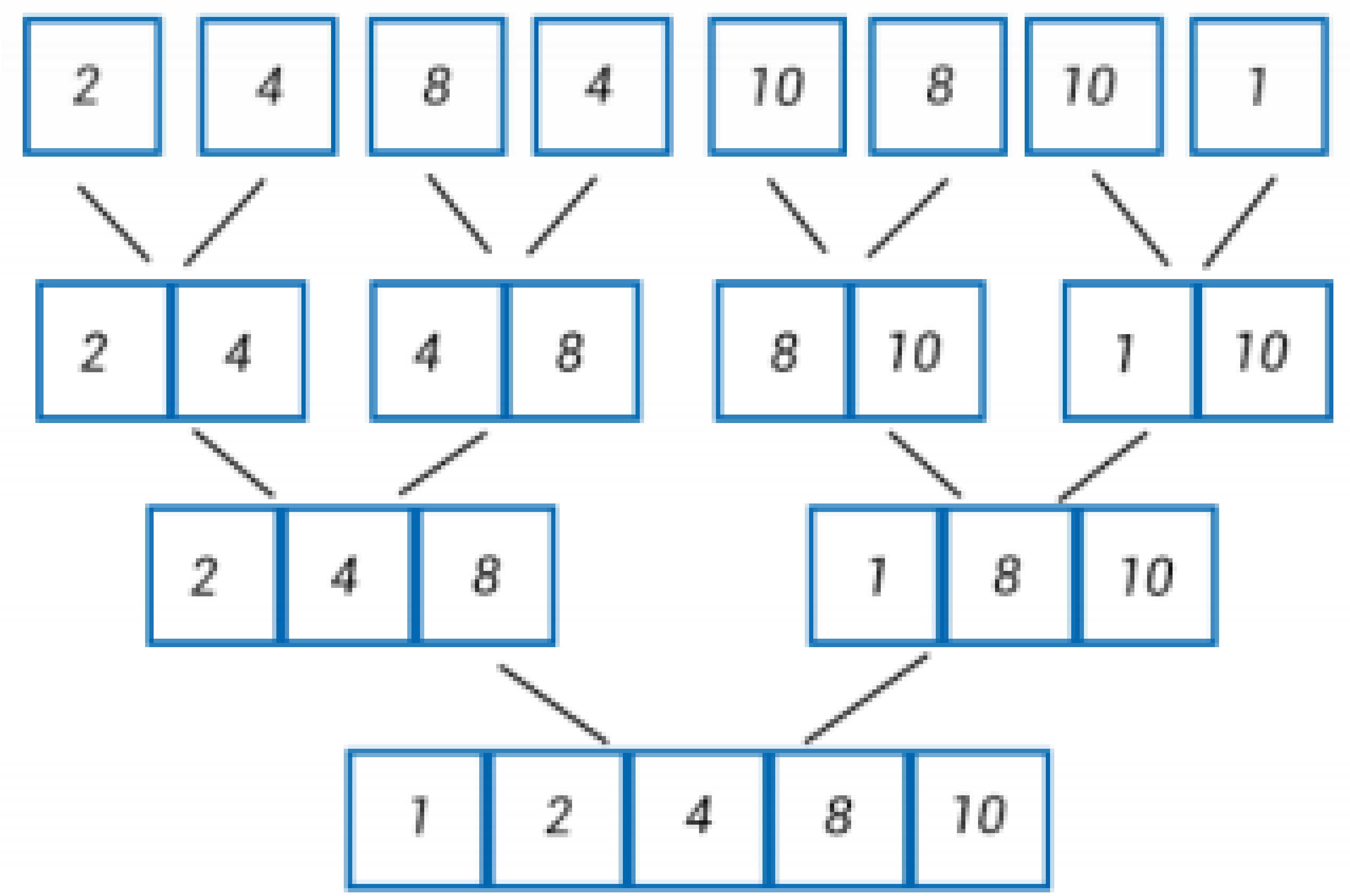


2. ALGORITMO DyV

```
vector<int> eliminarRepeticiones(vector<int> v){
    if (v.size() != 1){

        int m = (v.size())/2;
        vector<int> v1;
        vector<int> v2;

        for(int i = 0; i < m; i++)
            v1.push_back(v[i]);
        for(int i = m; i < v.size(); i++)
            v2.push_back(v[i]);


        v1 = eliminarRepeticiones(v1);
        v2 = eliminarRepeticiones(v2);


        v = eliminar(v1, v2);
    }
    return v;
}
```

2. ALGORITMO DyV

$O(n)$



2.ALGORITMO DyV

```
vector<int> eliminar(vector<int> v1, vector<int> v2){

    int i = 0, j = 0;
    int n1 = v1.size();
    int n2 = v2.size();

    vector<int> tmp;

    while (i < n1 && j < n2) {
        if (v1[i] < v2[j]){
            tmp.push_back(v1[i]);
            i++;
        }
        else if(v1[i] > v2[j]){
            tmp.push_back(v2[j]);
            j++;
        }
        else if(v1[i] == v2[j]){
            tmp.push_back(v1[i]);
            i++;
            j++;
        }

    }
    while (i < n1){
        tmp.push_back(v1[i]);
        i++;
    }

    while (j < n2){
        tmp.push_back(v2[j]);
        j++;
    }
    v1.clear();
    v2.clear();
    return tmp;
}
```

Ejemplo de ejecución:

```
pablorom@pablorom-MacBookPro:~/practicasAlgoritmica/p2$ ./eliminarDyv 20
Vector antes de eliminar duplicados
10 7 15 7 8 8 3 6 5 11 4 17 15 9 16 15 19 18 6 11
Componentes 20 tiempo 0.000138

 Vector despues de eliminar duplicados
3 4 5 6 7 8 9 10 11 15 16 17 18 19
```

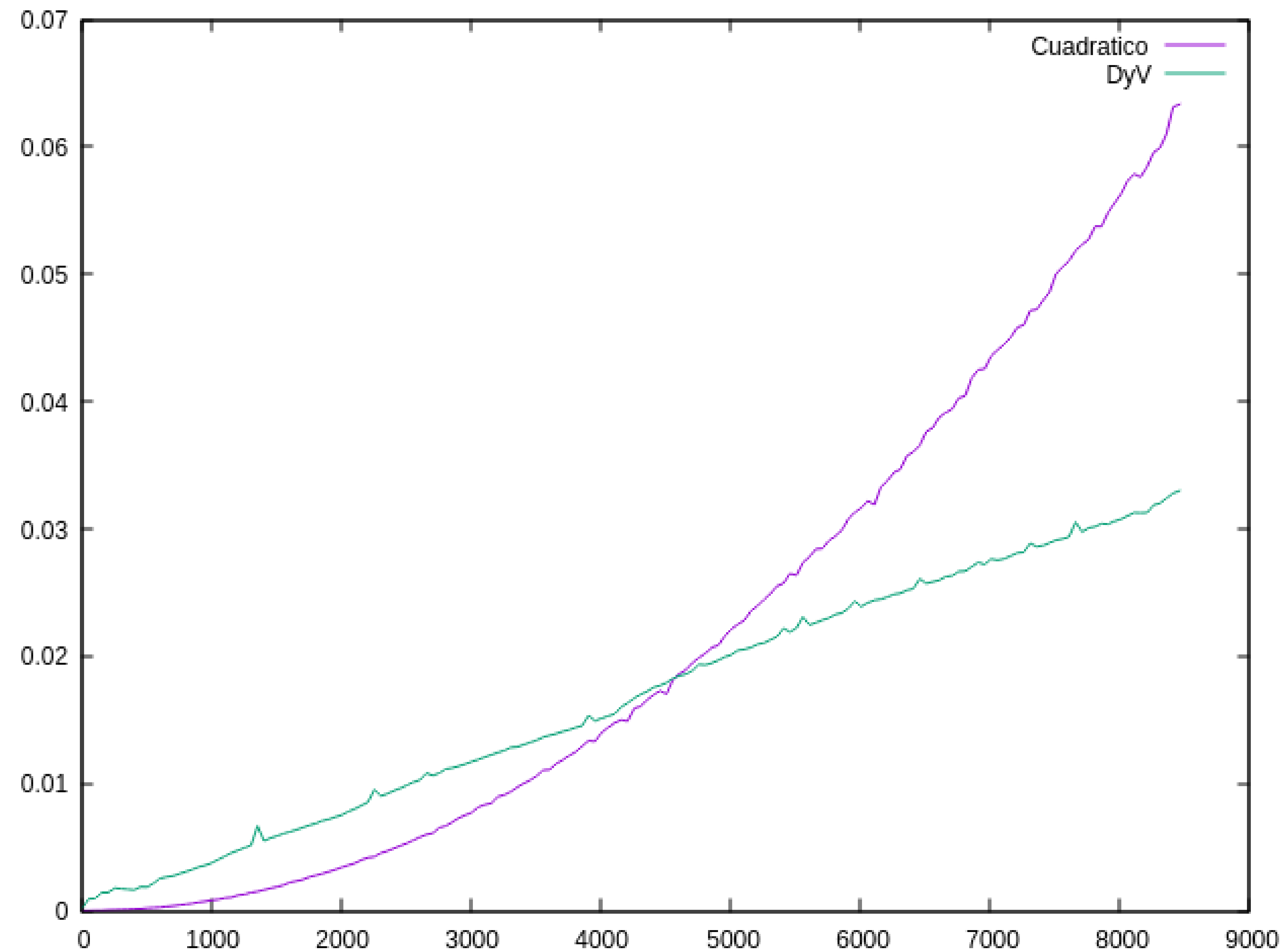
3. ESTUDIO EMPIRICO

Componentes	Tiempos (seg.)
10	8.00E-06
60	1.50E-05
110	2.40E-05
160	3.90E-05
910	0.000666
960	0.000737
1010	0.000828
1060	0.000898
2010	0.003419
2060	0.003575
2110	0.003747
2160	0.004002
3010	0.007718
3060	0.008124
3110	0.008337
3160	0.008451
4010	0.014003
4060	0.01438
4110	0.014768
4160	0.014997
4210	0.014892
5010	0.02214
5060	0.022494
5110	0.02283
5160	0.023568
6060	0.032193
6110	0.031904
6160	0.033283
6210	0.033774
7060	0.044064
7110	0.044512
7160	0.045044
7210	0.045835
8310	0.05992
8360	0.061036
8410	0.063139
8460	0.063353

n^2

Componentes	Tiempos (seg.)
10	1.92E-04
60	9.28E-04
110	9.72E-04
160	1.43E-03
210	1.43E-03
1010	0.003766
1060	0.004018
1110	0.004295
1160	0.004549
1210	0.00474
2110	0.007998
2160	0.008272
2210	0.008526
2260	0.009536
2310	0.009013
3060	0.011876
3110	0.01208
3160	0.012256
3210	0.012421
3260	0.012594
4010	0.015116
4060	0.015291
4110	0.015474
4160	0.016008
4210	0.016328
5010	0.020114
5060	0.020469
5110	0.020546
5160	0.020686
5210	0.020949
6010	0.023897
6060	0.024195
6110	0.024398
6160	0.02447
6210	0.024656
7010	0.027643
7060	0.027541
7110	0.027662
7160	0.027883
7210	0.028122
8260	0.03187
8310	0.032021
8360	0.032422
8410	0.032794

n log n



4. ESTUDIO HIBRIDO

Algoritmo cuadrático

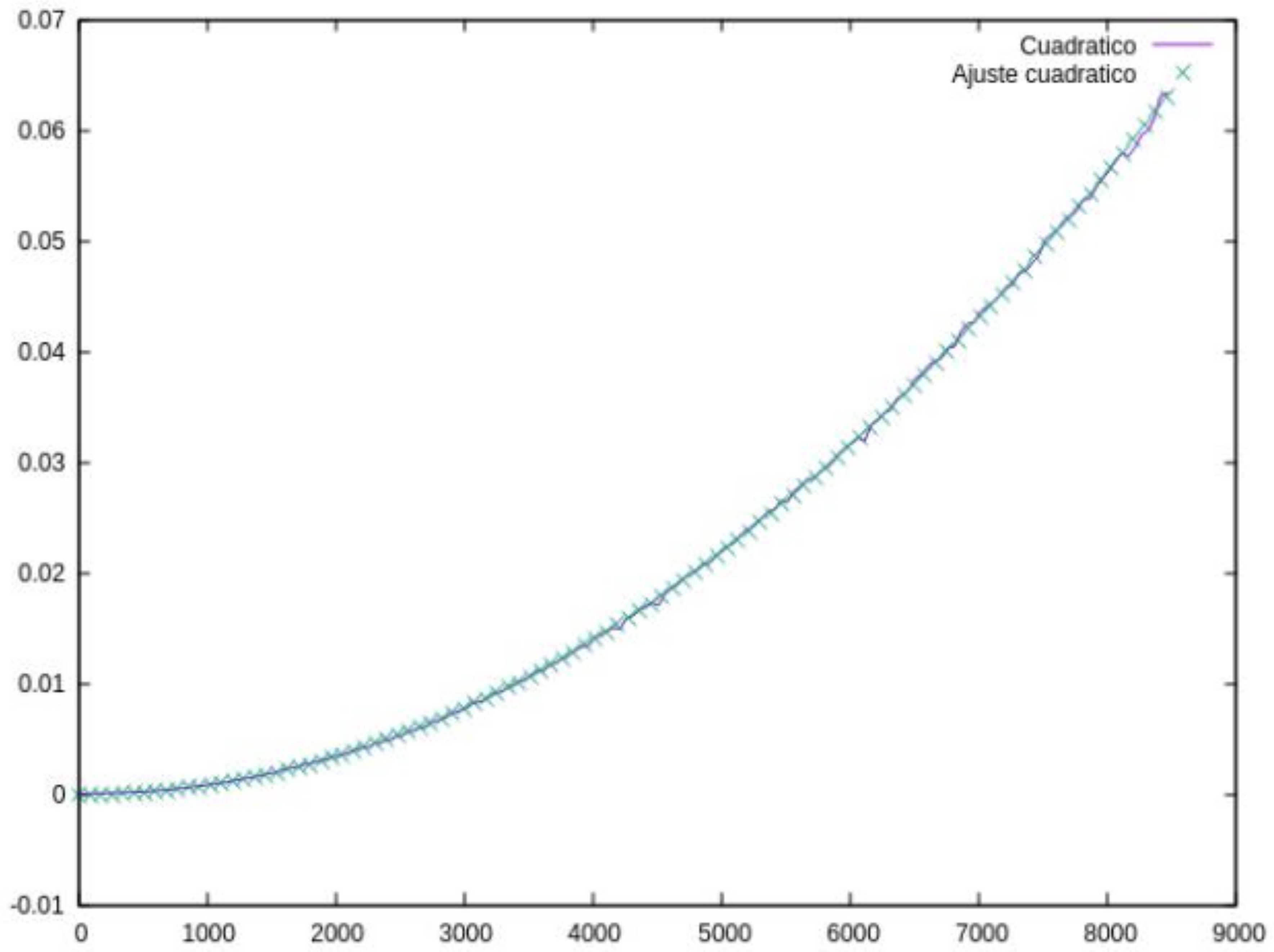
function used for fitting: f(x) = a0*x*x+a1*x+a2

After 11 iterations the fit converged.
final sum of squares of residuals : 9.82641e-06
rel. change during last iteration : -4.25933e-11

Final set of parameters	Asymptotic Standard Error	
a0 = 8.86647e-10	+/- 3.455e-12	(0.3897%)
a1 = -5.3324e-08	+/- 3.023e-08	(56.69%)
a2 = -5.00882e-05	+/- 5.542e-05	(110.7%)

correlation matrix of the fit parameters:

	a0	a1	a2
a0	1.000		
a1	-0.968	1.000	
a2	0.743	-0.864	1.000



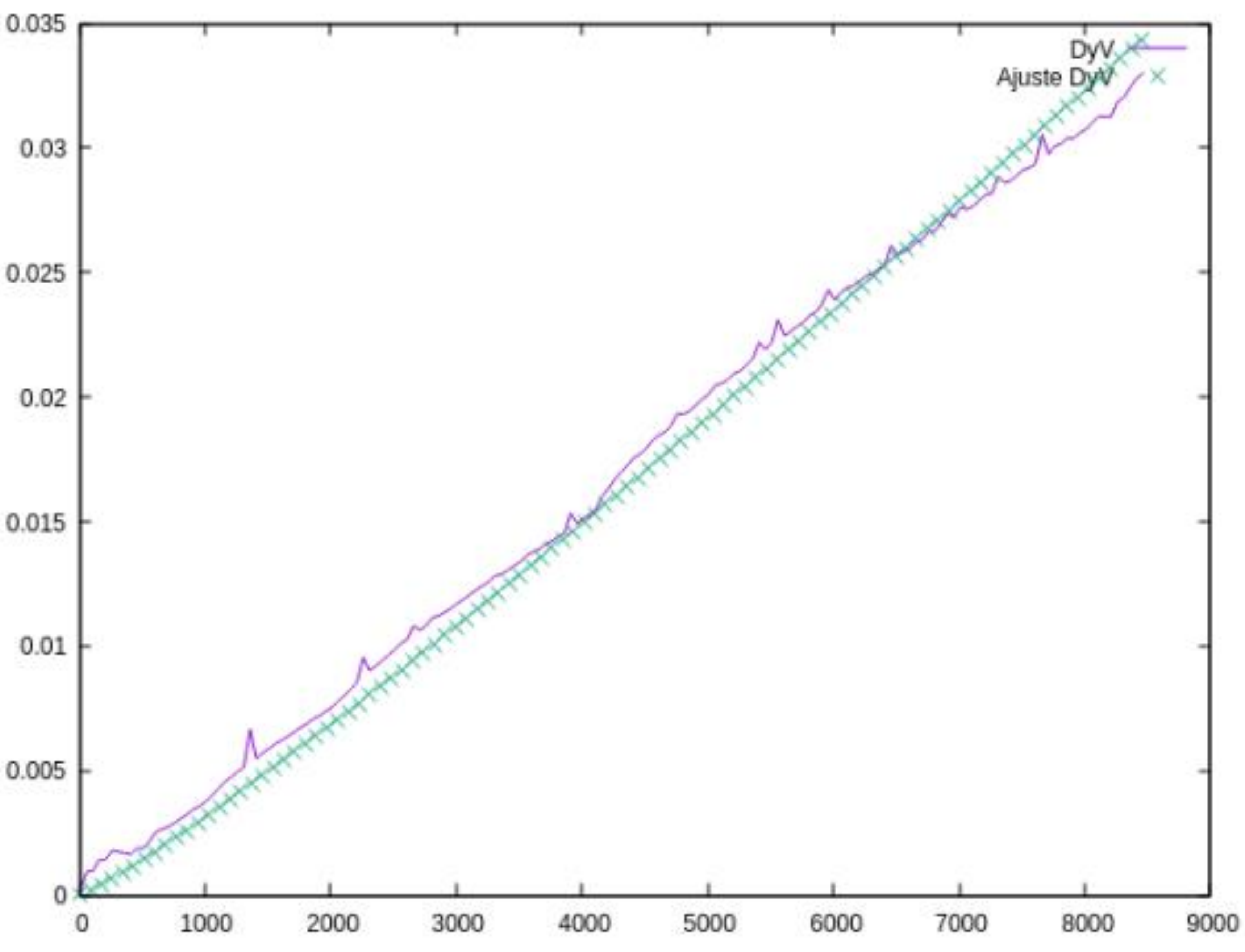
4. ESTUDIO HIBRIDO

Algoritmo DyV

function used for fitting: f(x) = a0*x*log(x)
fitted parameters initialized with current variable values

After 3 iterations the fit converged.
final sum of squares of residuals : 0.000134044
rel. change during last iteration : -5.4938e-11

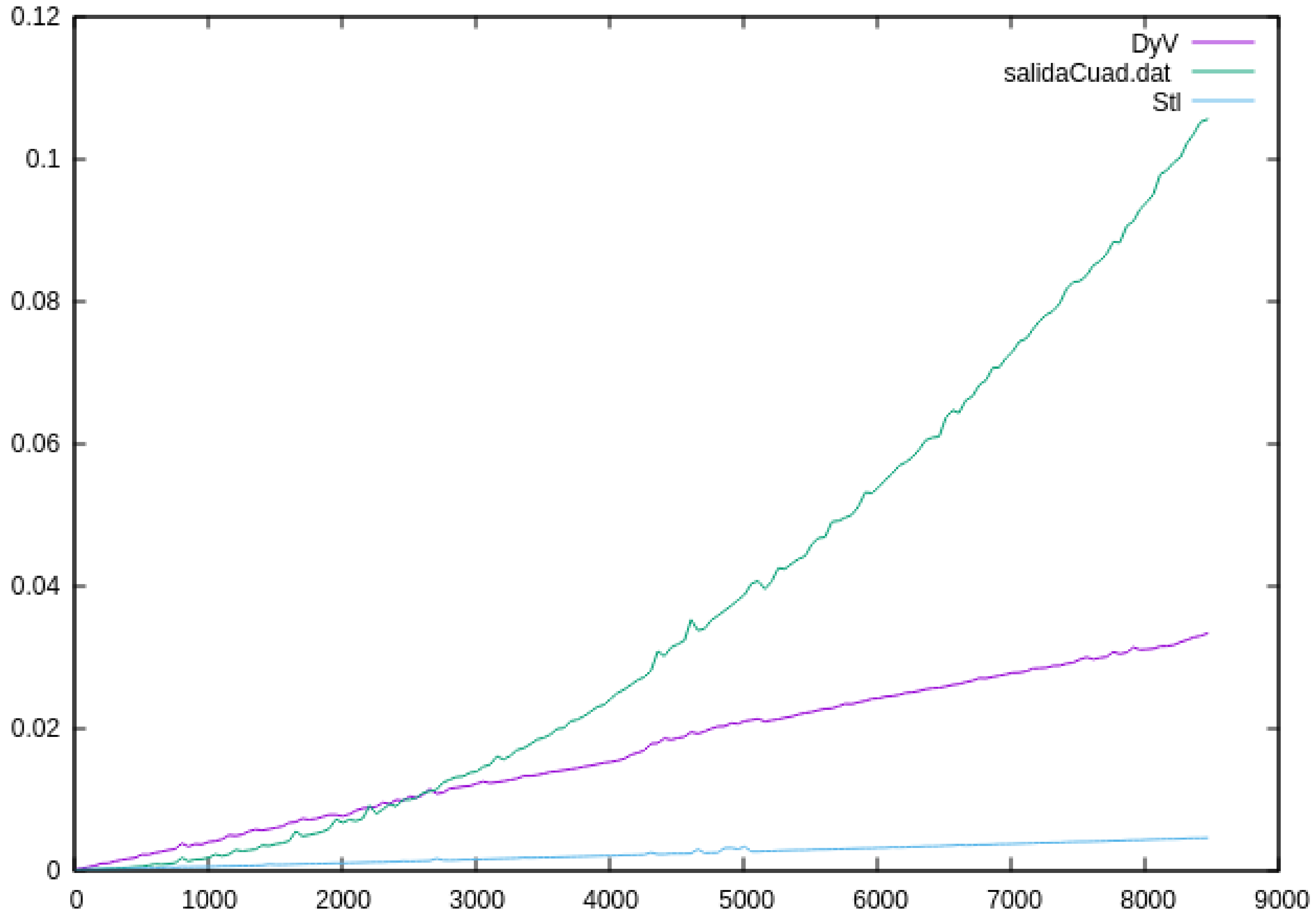
Final set of parameters		Asymptotic Standard Error	
a0	= 4.49249e-07	+/- 1.601e-09	(0.3563%)



5. ANEXO: Uso de la stl

Algoritmo stl

```
for(vector<int>::iterator it = v.begin(); it != v.end(); it++)
  tmp.insert(*it);
for(set<int>::iterator it = tmp.begin(); it != tmp.end(); it++)
  v2.push_back(*it);
```



GRACIAS POR SU ATENCIÓN

Ignacio Martínez Rodríguez
Álvaro García Jaén
Pablo Robles Molina
Práxedes Martínez Moreno
Francisco José González García