26-1-2021

Métodos Numericos para la Computación

Trabajo de curso

4ºCurso

Grupo Prácticas 11

Alejandro Daniel Herrera Cardenes
Carlos Eduardo Pacichana Bastidas
UNIVERSIDAD DE LAS PALMAS DE GRAN CANARIA

Indice

Selección de datos

Descripción

La base de datos se basa en tres tipos de ondas con 5000 instancias y 21 atributos los cuales varían entre 0 y 6.

Cada clase se genera a partir de una combinación de 2 de 3 ondas "base".

Cada instancia se genera con ruido agregado (media 0, varianza 1) en cada atributo.

5000×22 table

Var1	Var2	Var3	Var4	Var5	Var6	Var7	Var8	Var9	Var10	Var11	Var12	Var13	Var14	Var15	Var16	Var17	Var18	Var19	Var20	Var21	Var22
-1.23	-1.56	-1.75	-0.28	0.6	2.22	0.85	0.21	-0.2	0.89	1.08	4.2	2.89	7.75	4.59	3.15	5.12	3.32	1.2	0.24	-0.56	2
-0.69	2.43	0.61	2.08	2.3	3.25	5.52	4.55	2.97	2.22	2.81	1.61	1.24	1.89	1.88	-1.34	0.83	1.41	1.78	0.6	2.42	1
-0.12	-0.94	1.29	2.59	2.42	3.55	4.94	3.25	1.9	2.07	0.51	1.45	2.5	0.12	1.41	2.78	0.64	0.62	-0.01	-0.79	-0.12	0
0.86	0.29	2.19	-0.02	1.13	2.51	2.37	5.45	5.45	4.84	4.65	4.05	2.58	1.4	1.24	1.41	1.07	-1.43	2.84	-1.18	1.12	1
1.16	0.37	0.4	-0.59	2.66	1	2.69	4.06	5.34	3.53	4.82	4.79	4.3	1.84	1.73	0.21	-0.18	0.13	-0.21	-0.8	-0.68	1
0	0.77	1.32	0.29	-1.28	0.84	1.6	1.55	2.93	4.76	5.55	4.3	4.89	2.81	2.37	3.68	-0.98	0.69	0.91	-1.8	0.39	2
0.87	1.07	-0.65	1.46	0.84	2.7	3.67	2.94	3.81	5.2	8.16	3.29	4.24	2.43	0.4	1.6	0.72	0.66	0.05	-0.24	0.67	1
-0.22	-0.91	-1.18	0.35	-1.92	-1.59	1.91	0.75	1.72	2.02	3.63	3.91	2.73	4.29	4.89	2.04	1.13	-0.66	-1.33	0.41	-0.75	2
-1.11	-1.14	-0.89	0	0.53	0.44	0.24	2.15	1.64	1.75	3.92	5.68	3.39	4.24	3.81	4.56	3.18	1.51	2.9	0.14	-0.12	2
-0.75	1.1	-1.9	1.43	0.47	0.4	0.86	3.51	2.62	4.5	6.83	6.94	0.75	3.23	1.08	-0.25	0.73	-0.41	-1.5	0.46	1.47	2
0.14	-1.18	1.42	2.28	3.1	3.15	3.49	4.54	1.4	3.41	3.4	2.83	0.06	0.6	3.61	2.08	-0.83	0.55	-0.55	-0.43	-1.05	0
1.32	-0.4	-0.69	4.17	3.66	4	5.24	3.88	2.17	1.82	3.65	1.01	1.82	1.13	-0.07	0.26	0.5	1.38	1.25	-1.34	0.53	0
-0.93	2.48	1.2	2.97	2.91	3.57	3.68	4.19	3.22	3.53	2.46	2.17	0.77	0.52	2.42	-0.89	0.51	-0.39	0.82	0.14	-0.63	1
-1.06	0.59	1.01	3.33	2.05	3.2	4.7	4.21	4.73	2.22	2.67	2.79	2.05	-1.53	-1.54	0.37	-0.09	1.04	-0.08	-0.27	0.47	1
1.86	0.37	-0.35	0.74	0.84	0.21	1.97	1.52	1.85	2.39	3.92	3.76	3.27	1.61	3.08	2.78	1.58	1.68	2.61	-0.91	-0.27	2
-0.51	-0.48	0.35	-1.67	0.26	2.45	-0.09	2.03	0.79	1.42	1.13	2.52	2.06	4.5	4.28	4.66	3.3	0.38	0.75	1.76	0.37	0
1.16	-1.19	-2.26	0.63	0.32	1.51	2.11	2.58	1.03	2.01	4.04	4.55	5.65	2.74	3.12	2.67	2.01	4.12	-0.81	0.07	-0.96	2
-0.09	2.3	-0.43	0.36	0.11	-1.2	1.47	2.25	3.5	2.14	6.68	5.45	2.22	2.79	2.61	1.87	0.48	1.98	1.64	1.32	0.71	2

Dataset waveform

Matlab

Para realizar la parte básica de la práctica final hemos realizado el siguiente programa en octave que lo explicaremos paso a paso:

Para comenzar creamos la variable que va a contener todo el dataset y le pasamos mediante el método readtable el nombre del fichero, con lo que conseguimos que la variable file sea una matriz que contiene los datos del csv mencionado.

```
filename = 'waveform.csv';
M1 = readtable(filename)
```

Variable dataset

Continuamos calculando la media por columnas de dicho fichero para ello lo único que necesitamos es pasarle al método mean la matriz con todos los datos.

Una vez tenemos la media calculada, tenemos que restarle la media de cada columna a cada elemento de su misma columna.

```
media = mean(table);
XC = zeros(nrows,ncols);
for col = 1:ncols
    XC(:,col) = table(:,col)-media(col);
end
```

Media de las columnas del dataset

Calculamos la matriz de covarianza multiplicando nuestra matriz de datos por su traspuesta. Para calcular los autovectores y autovalores usamos la función ***eig*** de la covarianza y las ordenamos por sus pesos (de mayor a menor).

```
Z = (XC'*XC)./nrows;
[V,D] = eig(Z);
[d,ind] = sort(diag(D),"descend");
Ds = D(ind,ind);
Vs = V(:,ind);
```

Autovalores y autovectores del dataset

```
Ds =

   12.2549         0         0         0         0         0         0         0         0         0
         0    3.5625         0         0         0         0         0         0         0         0
         0         0    1.0764         0         0         0         0         0         0         0
         0         0         0    1.0453         0         0         0         0         0         0
         0         0         0         0    1.0286         0         0         0         0         0
         0         0         0         0         0    1.0034         0         0         0         0
         0         0         0         0         0         0    0.9964         0         0         0
         0         0         0         0         0         0         0    0.9933         0         0
         0         0         0         0         0         0         0         0    0.9580         0
         0         0         0         0         0         0         0         0         0    0.9533
```

Matriz de autovectores

Sabiendo que los autovectores con mayor valor están en la columna 1 y 2, pasamos de 5 dimensiones a 2 nuestros datos y los representamos.

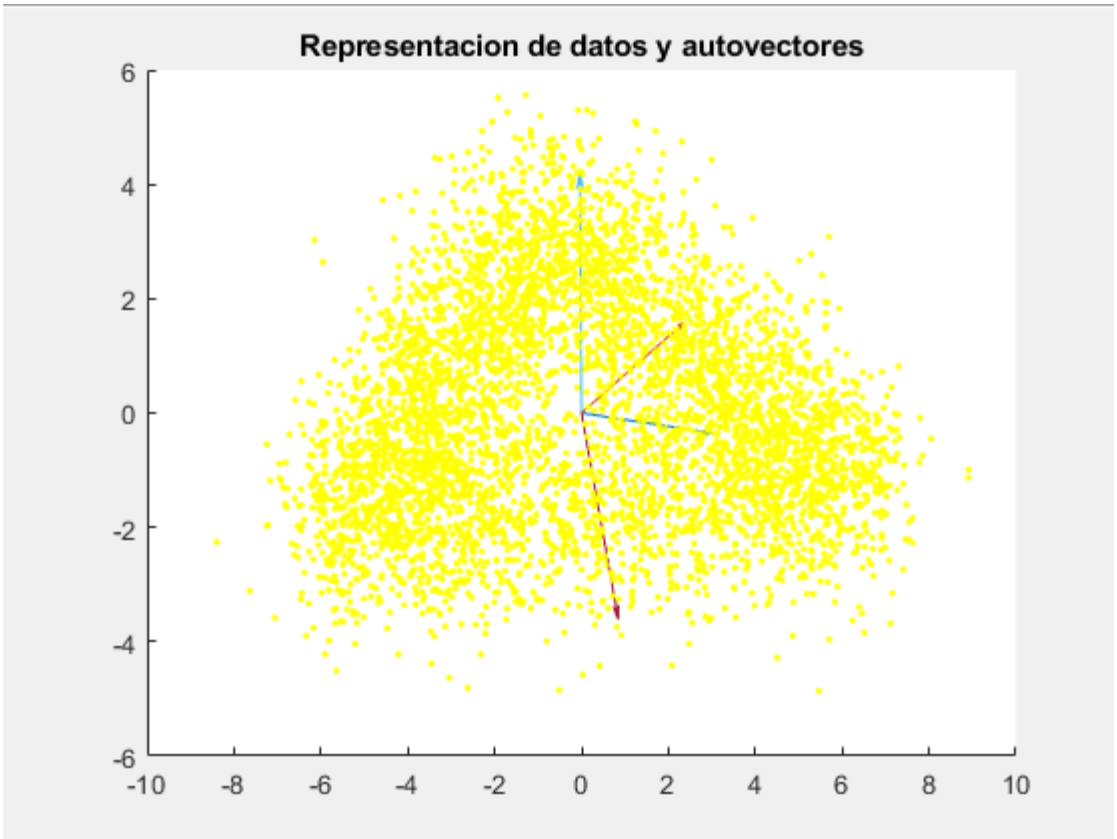
```
dataResize = XC * Vs(:,1:2);
mediaDataResize = mean(dataResize);
```

Reajustando el dataset

```
figure(1)
hold on
quiver(mediaDataResize(1), mediaDataResize(2), Vs(1,1)*50, Vs(2,1)*50);
quiver(mediaDataResize(1), mediaDataResize(2), Vs(1,2)*50, Vs(2,2)*50);
quiver(mediaDataResize(1), mediaDataResize(2), Vs(1,3)*10, Vs(2,3)*10);
quiver(mediaDataResize(1), mediaDataResize(2), Vs(1,4)*5, Vs(2,4)*5);
scatter(dataResize(:,1), dataResize(:,2),".y")
title('Representacion de datos y autovectores')
```

Código representación de datos y autovectores

Para representar los autovectores con mas peso usamos ***quiver*** el cual colocamos en el punto medio de los autovalores y representamos los puntos con ***scatter***



Gráfica de los datos de autovectores

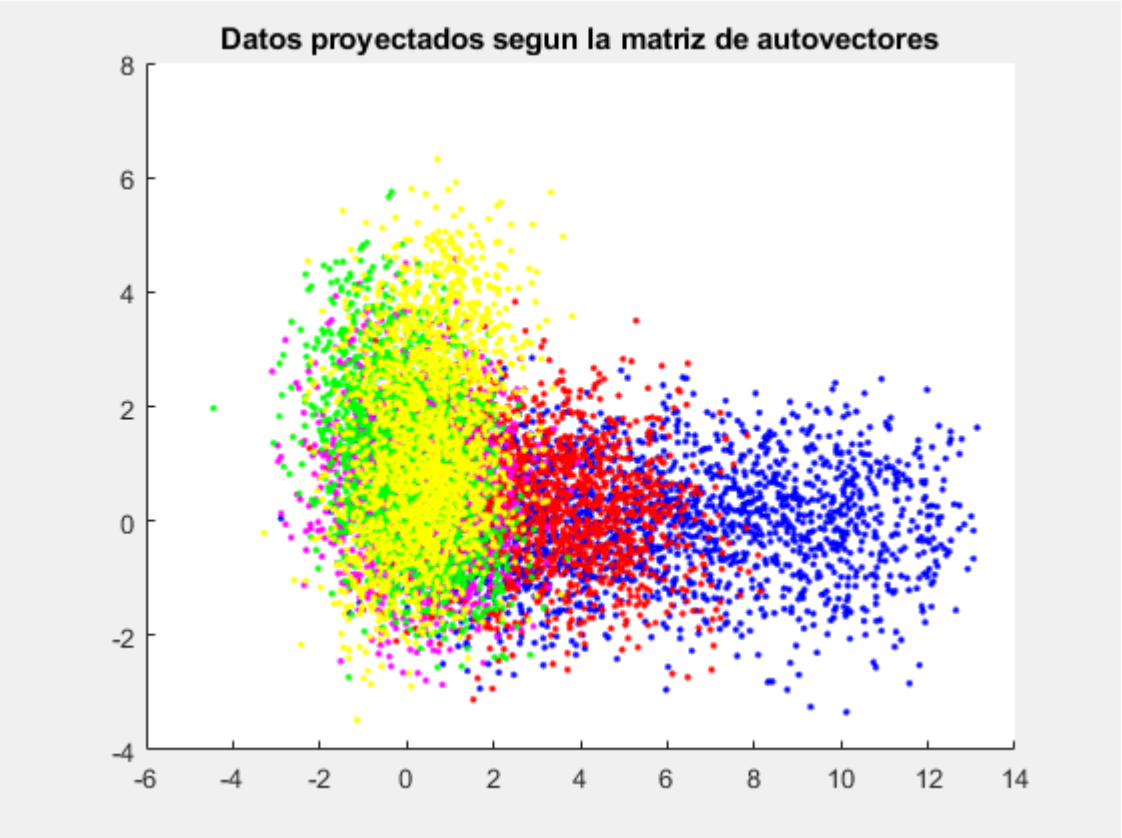
```
figure(2)

hold on;
final = table*Vs;

scatter(final(1:2000,1), table(1:2000,1),".b")
scatter(final(1:2000,2), table(1:2000,2),".r")
scatter(final(1:2000,3), table(1:2000,3),".m")
scatter(final(1:2000,4), table(1:2000,4),".g")
scatter(final(1:2000,5), table(1:2000,5),".y")
title('Datos proyectados segun la matriz de autovectores')
```

Código representación matriz de autovectores

Esta imagen ***(Gráfica proyección autovectores)*** representa el resultado de multiplicar los datos por los autovectores, podemos deducir que lo que hace es proyectar los datos en función de dichos autovectores, ya que si comprobamos la dirección de los vectores con la forma de los datos en esta última gráfica son bastante similares.



Gráfica proyección autovectores

Blas

Cargamos los datos y los centramos restando la media de cada componente generando la matriz XC.

```
const int nRow = 5000;
const int nColl = 10;
float* datas = new float[nRow*nColl];
load_csv(nRow, nColl, datas);
```

Cargando el dataset

```
const MKL_INT incx = 0;
float media;
const float a = 1;

for (int i = 0; i < nColl; i++) {
    media = -(cblas_sdot(nRow, &datas[i], nColl, &a, incx)/(float)nRow); // Sacar media
    cblas_saxpy(nRow, a, &media, incx, &datas[i], nColl); // generando una matriz XC
}
```

Generando la matriz XC

Generar la matriz de covarianza y calcular los autovalores y autovectores de ésta.

```
float* datasTrans = new float[nRow * nColl];
float* Z = new float[nColl * nColl];
float alpha = 0.0002;

mkl_somatcopy('R', 'T', nRow, nColl, a, datas, nColl, datasTrans, nRow);
cblas_sgemm(CblasRowMajor, CblasNoTrans, CblasNoTrans, nColl, nColl, nRow, alpha, datasTrans, nRow, datas, nColl, a, Z, nColl);
```

Generando la matriz de covarianza

```
const char JOBVL = 'V', JOBVR = 'V';
const int N = nColl;
const int LDA = N;
float* WR = new float[N];
float* WI = new float[N];
float* VL = new float[N];
const int LDVL = N;
float* VR = new float[N];
const int LDVR = N;
float* WORK = new float[4*N];
const int LWORK = 4*N;
int info = 0;

sgeev(&JOBVL, &JOBVR, &N, Z, &LDA, WR, WI, VL, &LDVL, VR, &LDVR, WORK, &LWORK, &info);
```

Calculando los autovalores y autovectores

Resultados

```
AutoValor: 12.254871 |  AutoVector: -0.000801 , 0.093333 , 0.175659 , 0.267644 , 0.369544 , 0.441171 , 0.521104 , 0.413065 , 0.310776 , 0.137300
AutoValor: 3.562470 |  AutoVector: 0.018812 , -0.080233 , -0.156203 , -0.255336 , -0.331683 , -0.190239 , -0.062832 , 0.222183 , 0.513726 , 0.663461
AutoValor: 1.076400 |  AutoVector: 0.337161 , -0.039311 , -0.137810 , 0.597696 , -0.188079 , 0.338605 , -0.134448 , -0.510422 , 0.107073 , 0.261624
AutoValor: 1.045260 |  AutoVector: -0.526119 , -0.352358 , 0.135337 , 0.446948 , -0.240317 , -0.392274 , 0.297208 , -0.117251 , 0.210244 , -0.151824
AutoValor: 0.953269 |  AutoVector: -0.047067 , -0.582294 , -0.265918 , -0.086143 , 0.003766 , 0.397091 , -0.347122 , 0.201439 , 0.321307 , -0.398223
AutoValor: 0.957979 |  AutoVector: -0.122476 , -0.173770 , 0.124578 , -0.394130 , 0.495964 , 0.020527 , 0.124809 , -0.646330 , 0.304899 , 0.106117
AutoValor: 1.028607 |  AutoVector: -0.161687 , -0.494157 , -0.234211 , 0.112864 , 0.334244 , -0.019244 , -0.006327 , 0.122775 , -0.551663 , 0.480145
AutoValor: 0.996417 |  AutoVector: 0.747782 , -0.366369 , 0.023315 , 0.019010 , 0.106193 , -0.423425 , 0.284650 , 0.072234 , 0.054960 , -0.160816
AutoValor: 1.003395 |  AutoVector: -0.080715 , 0.322875 , -0.797786 , 0.161610 , 0.316346 , -0.264107 , 0.061960 , -0.012221 , 0.185598 , -0.135626
AutoValor: 0.993314 |  AutoVector: 0.010533 , 0.084930 , 0.359667 , 0.322787 , 0.437301 , -0.305173 , -0.630170 , 0.164253 , 0.214084 , 0.069538
```

Datos de autovectores y autovalores

Cuda

Creamos un núcleo para generar la matriz de dos dimensiones con la que representar los datos principales, los autovectores y autovalores además de usarlo para multiplicar los datos por la matriz de autovectores.

A partir de estos resultados podemos generar un fichero .csv para cargar los datos en Matlab y representarlos gráficamente. ***(Paso 4 y 5)***

```
__global__ void multMatrixKernelFloatBlocks2(float* C, float* A, float* B, int width, int P, int Q) {
    int r = blockDim.x * blockIdx.x + threadIdx.x;
    int c = blockDim.y * blockIdx.y + threadIdx.y;

    if (r < P && c < Q) {
        float value = 0;
        for (int k = 0; k < width; k++) {
            value += A[r * width + k] * B[k * Q + c];
        }
        C[r * Q + c] = value;
    }
}
```

Núcleo para multiplicar matrices no cuadradadas

Resultados

En las imágenes inferiores podemos ver tres secciones ***(Primeros 5 valores, 5 valores en mitad de la tabla y los 5 últimos valores)*** de la matriz centrada por los dos autovectores principales.

A la izquierda podemos el resultado de la ejecución de cuda y a la derecha en Matlab.

```
0 -> | -4.243191 | -2.272764
2 -> | 3.651654 | -1.131589
4 -> | 2.577010 | -2.085503
6 -> | 2.268430 | 3.297496
8 -> | 1.086234 | 2.246893
          ...

5000 -> | -2.307247 | 1.421246
5002 -> | -0.443535 | 3.062418
5004 -> | -3.226379 | -0.321716
5006 -> | 6.110477 | -0.172599
5008 -> | -0.712568 | 2.243645
          ...

9990 -> | -3.538793 | -2.965284
9992 -> | 6.081593 | -0.069879
9994 -> | 0.946475 | 3.386845
9996 -> | -3.679379 | -1.383783
9998 -> | 3.762213 | -0.217241
```

```
ans =

   -4.2431   -2.2726
    3.6517   -1.1318
    2.5770   -2.0856
    2.2681    3.2975
    1.0861    2.2471

ans =

   -2.3074    1.4214
   -0.4438    3.0625
   -3.2265   -0.3216
    6.1104   -0.1728
   -0.7125    2.2436

ans =

   -3.5387   -2.9651
    6.0815   -0.0702
    0.9463    3.3868
   -3.6792   -1.3835
    3.7622   -0.2174
```

Resultado de matriz de autovectores por matriz centrada

En las imágenes inferiores podemos ver el resultado de la matriz de datos por la matriz de autovectores. ***(Ejecución en cuda y en matlab)***

```
| 1.263920 | 0.306615 | 0.349158 | 0.129611 | 2.063817 | -0.442289 | -1.167795 | -1.762535 | 0.661017 | -1.668197
| 9.158765 | 1.447789 | -0.666558 | -0.085983 | -0.340191 | -1.095412 | -0.778840 | -0.837634 | -0.984780 | 1.353827
| 8.084121 | 0.493876 | 0.535853 | -0.924662 | 1.528049 | 0.738473 | 0.499744 | -0.835078 | -0.093844 | -0.131977
| 7.775541 | 5.876874 | -0.648177 | 1.046735 | -0.497085 | 1.596304 | 0.233990 | 1.447576 | -0.314391 | -0.309843
| 6.593345 | 4.826272 | -1.131296 | 1.071386 | -0.430749 | -0.817954 | 1.373764 | 1.215082 | 1.130246 | -0.275641
         ...           ...           ...           ...           ...           ...           ...

| 3.199863 | 4.000625 | 0.357519 | 0.954010 | -0.117473 | 0.310796 | 1.138430 | -0.151728 | -0.057317 | -0.974271
| 5.063576 | 5.641797 | -0.395416 | 0.990699 | 0.935448 | 0.726727 | -0.721202 | -1.311631 | 1.891525 | -1.754174
| 2.280732 | 2.257663 | -0.123320 | 0.917924 | -0.837036 | 1.692025 | -1.171291 | -0.454060 | 0.040590 | 0.033270
| 11.617588 | 2.406779 | -0.309942 | 0.690616 | 0.460331 | -0.355520 | 0.706797 | 0.967561 | -1.160144 | -0.272865
| 4.794543 | 4.823024 | 1.066741 | -0.163794 | 3.633684 | -0.404917 | -0.097372 | 1.650480 | -0.616146 | 0.207873
         ...           ...           ...           ...           ...           ...           ...

| 1.968318 | -0.385905 | -0.631939 | -0.913501 | -0.812671 | 1.285972 | -0.131775 | -0.449686 | 1.312153 | 1.723930
| 11.588704 | 2.509499 | 0.950045 | -1.100874 | -0.458598 | 0.898744 | 0.023891 | -0.774256 | -0.144780 | 0.563144
| 6.453586 | 5.966223 | 2.163747 | -1.936070 | 1.542957 | 0.811212 | 0.486573 | -0.586845 | -0.240002 | -0.232797
| 1.827731 | 1.195596 | 0.130648 | -1.197492 | -0.505413 | -1.534914 | 0.322140 | 0.594337 | 0.430947 | 0.273827
| 9.269323 | 2.362137 | 0.326543 | 0.183799 | 1.151553 | 1.324114 | 0.556319 | 1.734169 | -0.094925 | 0.410436
```

Ejecución en cuda

```
ins =

    1.2639    0.3065    0.3492    0.1294    2.0639   -0.4422   -1.1677   -1.7624    0.6610   -1.6681
    9.1587    1.4473   -0.6669   -0.0859   -0.3399   -1.0956   -0.7782   -0.8371   -0.9849    1.3540
    8.0840    0.4935    0.5356   -0.9247    1.5282    0.7382    0.5002   -0.8347   -0.0940   -0.1317
    7.7751    5.8766   -0.6485    1.0470   -0.4967    1.5963    0.2342    1.4480   -0.3144   -0.3099
    6.5931    4.8261   -1.1316    1.0715   -0.4303   -0.8181    1.3739    1.2154    1.1302   -0.2757


ins =

    3.1996    4.0005    0.3573    0.9540   -0.1173    0.3108    1.1384   -0.1515   -0.0573   -0.9744
    5.0632    5.6416   -0.3955    0.9906    0.9359    0.7268   -0.7212   -1.3112    1.8916   -1.7542
    2.2805    2.2575   -0.1234    0.9180   -0.8369    1.6921   -1.1712   -0.4538    0.0407    0.0332
   11.6174    2.4063   -0.3104    0.6908    0.4607   -0.3558    0.7075    0.9682   -1.1603   -0.2727
    4.7945    4.8227    1.0667   -0.1637    3.6339   -0.4048   -0.0973    1.6509   -0.6163    0.2080


ins =

    1.9683   -0.3860   -0.6320   -0.9134   -0.8127    1.2858   -0.1317   -0.4497    1.3121    1.7241
   11.5885    2.5089    0.9496   -1.1009   -0.4583    0.8984    0.0245   -0.7737   -0.1449    0.5634
    6.4533    5.9658    2.1635   -1.9362    1.5431    0.8112    0.4867   -0.5865   -0.2400   -0.2326
    1.8278    1.1956    0.1306   -1.1976   -0.5053   -1.5350    0.3222    0.5944    0.4309    0.2739
    9.2691    2.3617    0.3263    0.1840    1.1518    1.3239    0.5567    1.7346   -0.0951    0.4107
```

Ejecución en Matlab