

```

1 restart;maple_mode(0);cas_setup(0,0,0,1,0,1e-10,10,[1,50,0,25],0,0,0);//radians,pas de cmplx, pas de Sqrt
  Cas_setup Vector [0,0,0,1,0,1.0000000000e-10,10,[1,50,0,25],0,0,0]
  ^M
  Warning: some commands like subs might change arguments order
2 n:=5;M:=matrix(n,n,(i0,j0)->rand(21)-10.0);//NB:rand(21) repond entre 0 et 20
  // Success
  ( 5,
    -3.0 , -7.0 , 4.0 , -8.0 , 0.0
    8.0 , -4.0 , 3.0 , 5.0 , 10.0
    0.0 , 9.0 , -4.0 , 8.0 , 0.0
    10.0 , 6.0 , 5.0 , -9.0 , 8.0
    -4.0 , 3.0 , 2.0 , 6.0 , 10.0
  )
3 maxnorm(M);colnorm(M);norm(M);
  ( 10.0 , 36.0 , 31.36877428272 )
4 cond:=M->[norm(evalf(M))*norm(1/evalf(M)),M];
  // Success
  // End defining cond
  {
    'nop';
    [norm(evalf(M))*norm(1/evalf(M)),M];
  }
  M -> }
5 cond(M);
  252.3004949579
  -3.0 , -7.0 , 4.0 , -8.0 , 0.0
  8.0 , -4.0 , 3.0 , 5.0 , 10.0
  0.0 , 9.0 , -4.0 , 8.0 , 0.0
  10.0 , 6.0 , 5.0 , -9.0 , 8.0
  -4.0 , 3.0 , 2.0 , 6.0 , 10.0
6 n:=5;:=seq(cond(matrix(n,n,(i0,j0)->rand(21)-10.0)),k=1..1000);
  // Success
  Evaluation time: 0.65
  ( Done , Done )
7 premiere methode: On prend la premiere ligne de la transpos'ee. Seconde methode avec une suite indexee par la liste
8 listecondi:=transpose(l)[0];
  [60.36801013143 8.759518296148 11.2272261816, 15.25602169235 16.42988606253 17.87716102969 49.58826935367 23.8801
9 listecondi:=seq(k[0],k=1);
  [60.36801013143 8.759518296148 11.2272261816, 15.25602169235 16.42988606253 17.87716102969 49.58826935367 23.8801
10 histogram(classes(listecondi,0,10));
  0.02
  0.015
  0.01
  0.005
  0
  0 500 1000 1500 2000
11 ml:=max(listecondi);// le max de la suite
  98441.29243204
12 ecart_type(listecondi);//ou bien: stddev donne l'ecart type
  2122.881878128

```

13

moyenne(listecondi);

158.0847871445

14

classes(listecondi,[0..50,50..100,100..200,200..400,400..floor(ml+1)]);//l'infini ne marche pas, on prend donc une borne strictement supérieure

0 .. 50 ,	796
50 .. 100 ,	102
100 .. 200 ,	59
200 .. 400 ,	27
400 .. 98442 ,	16

15

diagramme\_batons(classes(listecondi,[0..50,50..100,100..200,200..400,400..floor(ml+1)]));

0 .. 50 ,	796
50 .. 100 ,	102
100 .. 200 ,	59
200 .. 400 ,	27
400 .. 98442 ,	16

16

L'ecart type est enorme par rapport a la moyenne. On constate souvent 80 pourcent des conditionnements en dessous de 50, alors que parfois ca peut dépasser 1000

17

listetree:=sort(l,(x,y)->x[0]>=y[0]);

// Success

Done

18

listetree[0];M:=listetree[0][1];

(	98441.29243204	8.0 , -9.0 , -10.0 , 3.0 , -4.0	,	8.0 , -9.0 , -10.0 , 3.0 , -4.0	)
		-5.0 , 10.0 , -10.0 , -2.0 , -8.0		-5.0 , 10.0 , -10.0 , -2.0 , -8.0	
		5.0 , 6.0 , 1.0 , -10.0 , -2.0		5.0 , 6.0 , 1.0 , -10.0 , -2.0	
		1.0 , -2.0 , 0.0 , 0.0 , 0.0		1.0 , -2.0 , 0.0 , 0.0 , 0.0	
		-7.0 , -10.0 , 9.0 , -9.0 , -3.0		-7.0 , -10.0 , 9.0 , -9.0 , -3.0	

19

complex\_mode:=1;det(M);eigenvalues(M);

{	1, -18.0 ,	15.76131827131 0 ,	0 ,	0 ,	0
		0 ,	0.0008308359576178, 0 ,	0 ,	0
		0 ,	0 ,	-10.88411903818 0 ,	0
		0 ,	0 ,	0 ,	5.560984965459-9.765559613998*i0

20

jordan(M);//bon c'etait pas la peine!

41651.31477766	1476.895621139	10656.20712981	13047.79123569+6418.240959855*i	13047.79123569-6418.240959855*i
-29991.26974899	737.4906934134	13146.06464636	4810.857204607+6511.375870427*i	4810.857204607-6511.375870427*i
-1964.156789359	2879.511426352	-4393.328846403	-3554.012895294+12494.08267181*i	-3554.012895294-12494.08267181*i
6448.309242042	2303.98587717	1436.581326248	661.5617772698-25.89286027965*i	661.5617772698+25.89286027965*i
-3590.296761789	-4176.658740401	32790.38364009	-9156.71406895-10136.9296983*j	-9156.71406895+10136.9296983*i

21

Les cas anormaux on une valeur du determinant tout a fait normale, la matrice est donc bien inversible, en revanche, on constate une valeur propre (eventuellement complexe) proche de 0, elle n'est donc pas loin d'une matrice non inversible.

23	Digits:=4;	[0, 0, 1, 1, 0, [1e-10, 1e-15 ], 4, [1, 50, 0, 25 ], 0, 0, 0 ]	M
24	a:=-1-(0.1)^10;	0.9999999999	M
25	a-1;//il a travaille en fait avec plus de chiffres.	-1.000159954856e-10	M
26	Digits:=14;	[0, 0, 1, 1, 0, [1e-10, 1e-15 ], 14, [1, 50, 0, 25 ], 0, 0, 0 ]	M
27	a:=-1-(0.1)^15;	1.000000000000000	M
28	a-1;//il n'a toujours pas perdu le 0.1	0.000000000000000	M
29	Digits:=15;	[0, 0, 1, 1, 0, [1e-10, 1e-15 ], 15, [1, 50, 0, 25 ], 0, 0, 0 ]	M
30	a:=-1-(0.1)^16;//la il a vraiment travaille en 15 chiffres.	1.000000000000000	M
31	a-1;	0.000000000000000	M
32	Digits:=15;	[0, 0, 1, 1, 0, [1e-10, 1e-15 ], 15, [1, 50, 0, 25 ], 0, 0, 0 ]	M
33	on s'assure d'avoir une precision exacte, car sous xcas en 32 bits, moins de 14 chiffres fait la meme chose que 14 chiffres.		
34	v:=seq(rand(-10.0,10.0),j=1..n);	7.0506586770248740, 7.654587662028827, 4.204786870744286, 2.774450220440250, 7.008202482870525	M
35	b:=M*v;	[-6.991030351579, -14.73021148884, -33.16194413374, 8.243516646658, 213.6414998052]	M
36	purge(x);	No such variable x	M
37	X:=seq(x[j],j=1..n);	[x[1], x[2], x[3], x[4], x[5]]	M
38	linsolve(M*X=b,X)-v;	7.002281086022e-10, 4.55750067678e-10, 5.082204400041e-10, 4.705240845008e-10, 8.684426028422e-10	M
39	P:=eigenvectors(M);eigenvalues(M);	41651.31477766, 1476.895621139, 10656.20712981, 13047.79123569+6418.240959855*i, 13047.79123569-29991.26974899i, 737.4906934134, 13146.06464636, 4810.857204607+6511.375870427*i, 4810.857204607-1964.156789359i, 2879.511426352, -4393.328846403, -3554.012895294+12494.08267181*i, -3554.012895294-6448.309242042i, 2303.98587717, 1436.581326248, 661.5617772698-25.89286027965*i, 661.5617772698+3590.296761789i, -4176.658740401, 32790.38364009, -9156.71406895-10136.9296983*i, -9156.71406895+10136.9296983*i	M
40	v:=(transpose(P)[2]);	[10656.20712981, 13146.06464636, -4393.328846403, 1436.581326248, 32790.38364009]	M
41	b:=M*v;	[-115983.4268964, -143083.3324946, 47817.51413812, -15635.92216291, -356894.4388463]	M
42	NB une erreur de 10^-11 alors que l'on travaille avec 15 chiffres donne tout de meme un rapport 1000.		
43	linsolve(M*X=b,X)-v;	[-6.565824151039e-07, -3.301538527012e-07, -1.274747774005e-06, -1.019681803882e-06, 1.844018697739e-06]	M