Vitesse de Convergence d'une chaine de Markov ergodique vers sa mesure stationnaire

```
> restart():
  with(linalg):
  with(stats):
  with(LinearAlgebra):
  Randomize():
  with(RandomTools):
  with(plots):
```

```
Warning, the previous binding of the name GramSchmidt has been removed and it now has an assigned value

Warning, these names have been redefined: anova, describe, fit, importdata, random, statevalf, statplots, transform

Warning, the name changecoords has been redefined
```

génération aléatoire d'une matrice stochastique (matrice de transition de la chaine)

la petite fonction...

```
> rand_stoch := proc (n)
  local v,i,j,s;
  v := Matrix(n,n);
  for j from 1 to n do
  s := 0;
  for i from 1 to n do
    v[j,i] := rand();
    s := s + v[j,i];
  end do;
  for i from 1 to n do
    v[j,i] := v[j,i]/s;
  end do;
  end do;
  v;
  end proc;
```

```
rand_stoch := proc(n)
local v, i, j, s;
    v := Matrix(n, n);
    for j to n do
        s := 0;
        for i to n do v[j, i] := rand( ); s := s + v[j, i] end do;
        for i to n do v[j, i] := v[j, i] / s end do
    end do;
    v
```

```
end proc
```

... et son application

```
> Q := rand_stoch(4):
  evalf(Q);
  Id:=Matrix(4,4,shape=identity):
```

$$\begin{bmatrix} 0.5206840165 & 0.1642503752 & 0.3022593487 & 0.01280625953 \\ 0.1744774040 & 0.07826179288 & 0.6826583913 & 0.06460241180 \\ 0.1881386595 & 0.2799814644 & 0.3290417997 & 0.2028380765 \\ 0.4572083766 & 0.08482520852 & 0.1054323232 & 0.3525340917 \end{bmatrix}$$

calcul de sa loi invariante

résolution du système

```
> M := Transpose(Id-Q):
  pi_ := LinearSolve(M, Vector(4,0), free='s');
```

$$pi\_ := \begin{bmatrix} \frac{1061428513435098349023713888023092680658852058727}{43104542830473107819658026867880611362977567 1320} s_4 \\ \frac{7139242395400154045483770181025013659319374 0793}{53880678538091384774572533584850764203721958915} s_4 \\ \frac{9391617217733183975951724438341776775700896 1567}{35920452358727589849715022389900509469147972610} s_4 \\ s_4 \end{bmatrix}$$

normalisation de la loi invariante

```
> i := 'i':
  x := solve(sum('pi_[i]',i=1..4)=1, {'s[1]', 's[2]',
  's[3]', 's[4]'}):
  pi := Transpose(subs(x, pi_)):
  evalf(pi);
```

$$[0.3326728677, 0.1790064775, 0.3532224198, 0.1350982350]$$

Tirage d'une loi sur un ensemble à quatre éléments (pour initialiser la chaine)

```
> a := rand():
  b := rand():
  c := rand():
  d := rand():
```

```
S := a+b+c+d:
t_max := 15;
mu := Matrix(4, t_max+1, 0):
mu[1,1] := a/S:
mu[2,1] := b/S:
mu[3,1] := c/S:
mu[4,1] := d/S:
evalf(Column(mu,1));
```

$$t\_max := 15$$

$$\begin{bmatrix} 0.4151179114 \\ 0.07188356699 \\ 0.3938522326 \\ 0.1191462890 \end{bmatrix}$$

calcul de la loi itérée k fois

```
> for t from 1 to t_max  do
    eta[t] :=
VectorMatrixMultiply(Transpose(Column(mu,1)),MatrixPower(Q,t)
):
end do:
evalf(eta[t_max]);
```

[0.1986515648, 0.2080541973, 0.3508591358, 0.2424351021]

Calcul de N

```
> d :=2:
  f:= (i,j) -> 4*(eta[j][i] - eta[j][i]^2)*10^(2*d):
  N_mat := evalf(Matrix(4,t_max, f)):
  N := floor(max(seq(seq(N_mat[i,j], j = 1 .. t_max ), i = 1
  .. 4)));
```

$$N := 9379$$

simulation de la loi invariante - mesures empiriques

```
> min(seq(seq(Q[i,j], j = 1 .. 4 ), i = 1 .. 4)):N_inv := 1/N:
  for t from 2 to t_max+1 do
    M := MatrixPower(Q,t-1):
    p := VectorMatrixMultiply(Transpose(Column(mu,1)), M);
    p2 := p[1]+p[2]:
    p3 := p2+p[3]:
    for i from 1 to N do
       g := Generate(float(method=uniform)):
       if g < p[1] then
        mu[1,t] := mu[1,t]+N_inv;
      elif g < p2 then
        mu[2,t] := mu[2,t]+N_inv;
      elif g < p3 then
```

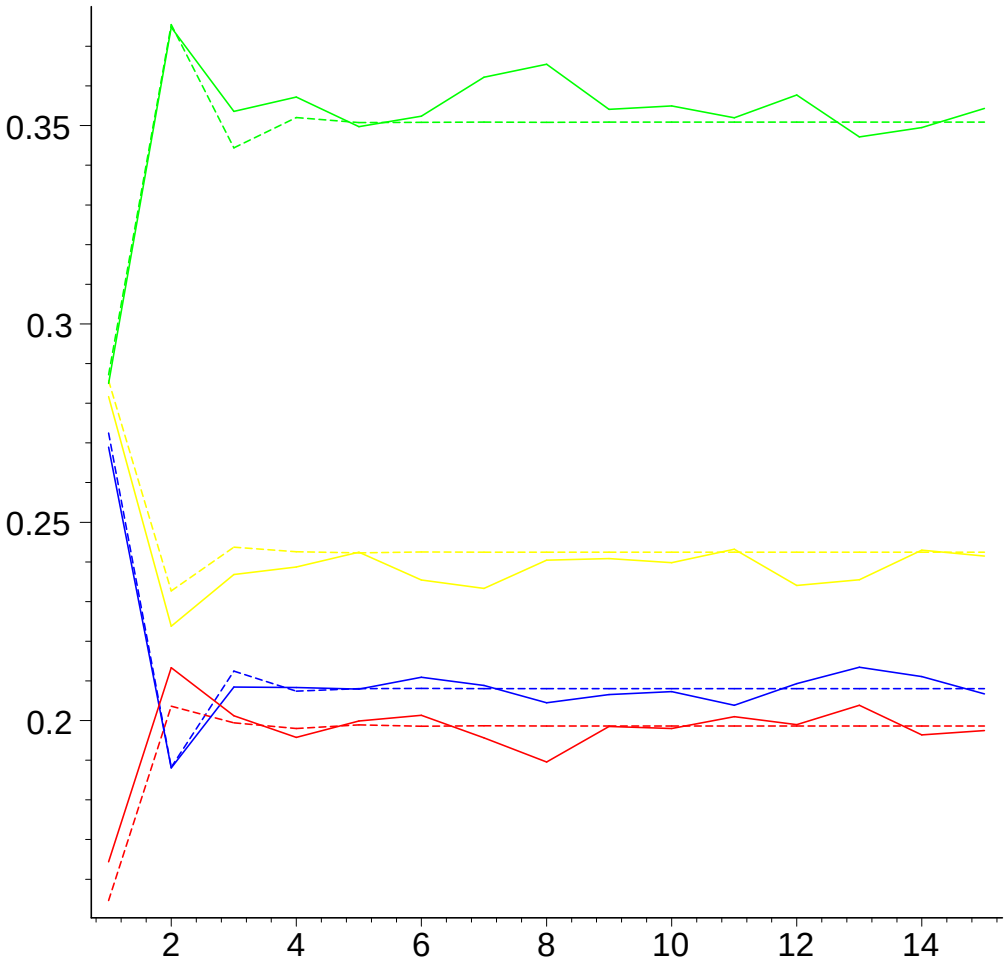
```
        mu[3,t] := mu[3,t]+N_inv;
      else
        mu[4,t] := mu[4,t]+N_inv;


      end if;
    end do:
  end do:
  evalf(Column(mu,t_max+1));
```

$$\begin{bmatrix} 0.1974624160 \\ 0.2067384583 \\ 0.3543021644 \\ 0.2414969613 \end{bmatrix}$$

représentation graphique des résultats

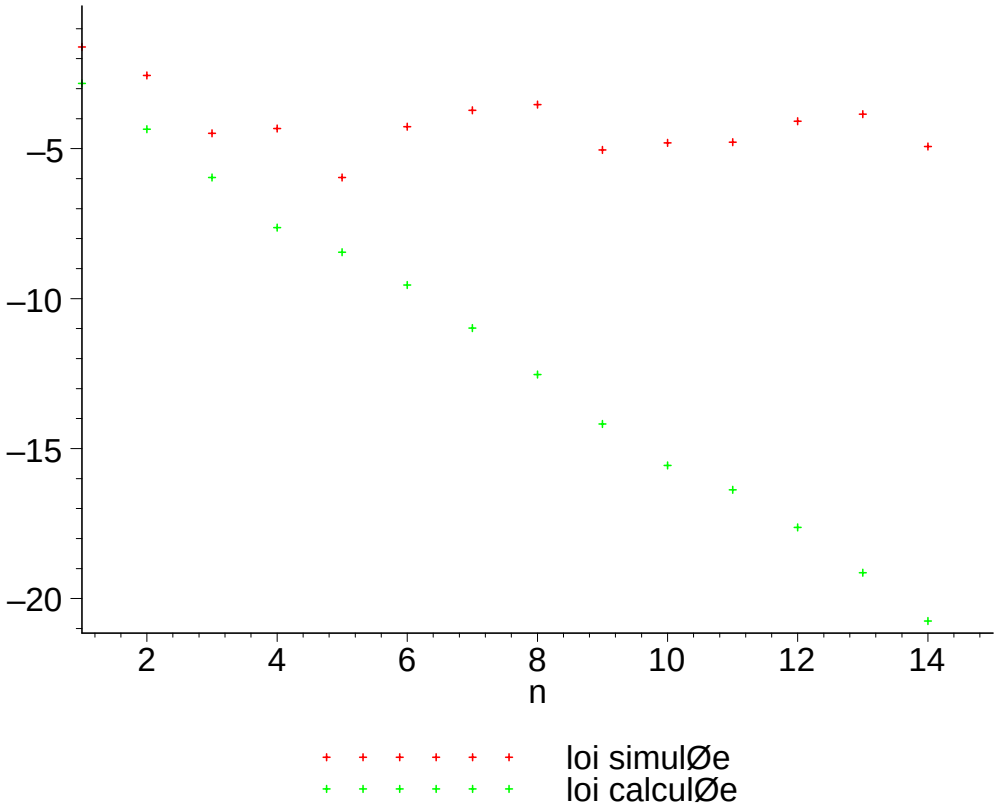
visualisation de la convergence de la loi simulée vers la loi calculée

```
> l1 := [seq( [j-1, mu[1, j]], j=2..t_max+1)]:
  l2 := [seq( [j-1, mu[2, j]], j=2..t_max+1)]:
  l3 := [seq( [j-1, mu[3, j]], j=2..t_max+1)]:
  l4 := [seq( [j-1, mu[4, j]], j=2..t_max+1)]:
  p1 := plot({l1, l2, l3, l4},
  color=[red,blue,green,yellow]):
  l5 := [seq( [j, eta[j][1]], j=1..t_max)]:
  l6 := [seq( [j, eta[j][2]], j=1..t_max)]:
  l7 := [seq( [j, eta[j][3]], j=1..t_max)]:
  l8 := [seq( [j, eta[j][4]], j=1..t_max)]:
  p2 := plot({l5, l6, l7, l8},
  color=[red,blue,green,yellow], linestyle=DASH):
  display({p1,p2});
```



visualisation de la convergence de ces deux lois vers la loi invariante

```
> f := i -> log(abs(pi[1]-mu[1,i]) + abs(pi[2]-mu[2,i]) +
  abs(pi[3]-mu[3,i]) + abs(pi[4]-mu[4,i])):
  g := i -> log(abs(pi[1]-eta[i][1]) + abs(pi[2]-eta[i][2])
  + abs(pi[3]-eta[i][3]) + abs(pi[4]-eta[i][4])):
  lf := [seq([n-1,f(n)],n=1..t_max)]:
  lg := [seq([n-1,g(n)],n=1..t_max)]:
  plot([lf, lg],n=1..t_max, style=point, legend=["loi
  simulée", "loi calculée"]);
```



regression linéaire - calcul de l'équation de la droite

```
> a:= 'a':
  u := 'u':
  a := evalf([seq(g(n),n=1..t_max)]):
  b := [seq(n, n=1..t_max)]:
  res := fit[leastsquare[[n,u]]]([b,a]):
  assign(res):
  pente := diff(u, n);
  ordonnee_0 := subs(n=0, u);
```

$$pente := -1.251218749$$

$$ordonnee\_0 := -0.9000436200$$

comparaison entre la pente de la droite et log (1 - inf P)

```
> c := 1-min(seq(seq(Q[i,j], j = 1 .. 4 ), i = 1 .. 4)):
  evalf(exp(pente)) < evalf(c);
```

$$0.2861558321 < 0.9871937405$$

Spectre de P

```
> evalf(Eigenvalues(Q));
```

$$\begin{bmatrix} 1. \\ -0.09904610971 \\ -0.2197850785 + 0.1390958000\,I \\ -0.2197850785 - 0.1390958000\,I \end{bmatrix}$$

module des valeurs propres

```
> evalf(Map(abs, Eigenvalues(Q)));
```

$$\begin{bmatrix} 1. \\ 0.09904610971 \\ 0.2601021382 \\ 0.2601021382 \end{bmatrix}$$

```
> exp(pente);
```

$$0.2563816839$$

```
>
```