V12.2 à publier

Tableau des principales syntaxes pour le lycée : Scilab, Python, TI, Casio, Xcas

Contenu

[Entrées-Sorties - tests - répétitions 2](#_Toc320435469)

[Listes 3](#_Toc320435470)

[Graphiques 3](#_Toc320435471)

[Nombres aléatoires - Lois de probabilités discrètes 4](#_Toc320435472)

[Lois de probabilités continues 5](#_Toc320435473)

[Constantes - fonctions 6](#_Toc320435474)

[Matrices 7](#_Toc320435475)

Illustrations sur le site planète maths : [fiche thématique Algorithmique](http://www.ac-grenoble.fr/maths/guppy/articles.php?lng=fr&pg=99) et [logiciels](http://www.ac-grenoble.fr/maths/guppy/articles.php?lng=fr&pg=73)

# **Entrées-Sorties - tests - répétitions**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | | **Langage algorithmique** | | | | | |
|  | | **Scilab**  Certaines fonctions nécessitent l’installation du module « lycée »  **Le point virgule permet d’écrire plusieurs instructions sur la même ligne,**  **Il supprime aussi l’affichage** | **Python 2.6**  ***3.x seulement***  **Attention : en v 2.6,**  **3/2 vaut 1, et non 1,5** | **TI**  **82-84** | **Casio**  **35+**  **(non USB)** | **XCAS 0.9.4**  **Le point-virgule sépare les instructions**  **Pour plusieurs instructions le français et l’anglais sont acceptés.** | |
| Insérer un commentaire | | *// mon commentaire* | *# mon commentaire* |  |  | // mon commentaire |  |
| Saisir *a* | | *a*=input ( "donner a  ") | *a*=input("donner a ")  a=float(input("réel a ?"))  a=int(input("entier a ?")) | :Prompt A  :Input "X1=", X | "A=":?→A  dans shift PRGM | input("a= ",a) | saisir ("a= ",a) |
| Afficher *a*  *( Xcas : Unquoted : sans guillemets)* | | *a ou afficher(* *" a=* *"+string(a))*  *ou disp(* *" a= "+string(a))* | V 2.6 : print «’a= ‘, *a*  *V 3.x : print ( ‘a= ‘, a)* | :Disp "A=" , A | "A= " : A◢  dans shift PRGM | print ("a=",a)  print ("a=",a) | afficher ("a=",a)  afficher ("a=",a) |
| affectation: *a*→ *b*  *b* prend la valeur *a,* | | *b* = *a* | *b* = *a* | A  B | A→ *B*  *touche directe* | b := a | |
| Tests, logique  =,,,  et, ou, non, ou exclusif, | | == , <> , <= , >=  **&** , **|** (AltGr+6) , **~** (AltGr+2),voir | ==, != , <=, >=  and, or, not, xor | Menu 2nd TEST  =,,,  and, or, not, xor | =,,,  Dans shift PRGM REL  and, or, not  dans OPTN LOGIC | =, != , <=, >=  and, or, not, xor | |
| Bloc d'instructions | | Les blocs sont définis par la structure | Les instructions d'un bloc ont la même marge à gauche (indentation) | Le bloc est terminé par End | Le bloc est terminé par End, ifEnd, whileEnd,… | Le bloc est encadré par des accolades: {*instructions* }, **sauf** s’il est encadré par la structure, voir ci-dessous. Pour ne pas se tromper utiliser le menu **Add** | |
| Si *condition*  Alors  *Instruction*1  *Facultatif*  [ Sinon  *Intruction*2]  Fin du si | *Condition2 est la négation de condition1* | if *condition* then  *Instructions1*  else  *Intructions2*  *end*  (if et then doivent être sur la même ligne) | if *condition*:  *Instruction*1  [ else:  *Intruction*2] | :If *condition*  :Then  *Instructions*1  [:Else  *Intructions*2]  :End | If *condition*  Then *Instructions*1  [Else *Intructions*2]  IfEnd  dans PRGM COM | if ( *condition*)  then  *{Instructions1*}  [ else  *{Intructions2}]* | si (condition)  alors  *Instructions1*  [ sinon  *Intructions2 ]*  *fsi* |
| Répéter  *Instruction(s)*  Jusqu'à *condition*1 | | while *%T* then  *Instruction(s)*  if *condition*then break  end | while True*:*  *Instruction(s)*  if *condition* :  break | :Repeat *condtion*1  *Instruction(s)*  :End | Do  *Instruction(s)*  LpWhile *condition*2  dans PRGM COM | repeat  *instruction(s)*  until  *(condition*1) | repeter  *instruction(s)*  jusqu\_a *(condition*1) |
| Tant *condition*  *Instruction(s)*  Fin TantQue | | while *condition* then  *Instruction(s)*  *end* | while *condition* :  *Instruction(s)* | :While *condtion*  *Instruction(s)*  :End | While *condition*  *Instruction(s)*  Endwhile  dans PRGM COM | while (*condition)*  {*Instruction(s)*} | tantque *(condition)*  faire  *instruction(s)*  ftantque; |
| Pour i variant de 1 à *n*  Faire  *Instruction(s)*  Fin du pour | | for i=1:n  *Instruction(s)*  end | for i in range(1,n+1):  *Instruction(s)* | :For (I,1,N)  *Instruction(s)*  :End | For 1->A To 10  *Instruction(s)*  Next  dans PRGM COM | for j from 1 to n do  *instruction(s)*  end\_for  (ne pas utiliser «*i* » ) | pour j de 1 jusque n  faire  *instruction(s)*  fpour (éviter la lettre *i*) |

# Listes

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | **Scilab** | **Python 2.6**  ***3.x seulement*** | **TI**  **82-84** | **Casio**  **35+**  **(non USB)** | **XCAS 0.9.4**  **.** |
| Créer une liste | *En Scilab, les listes sont aussi appelées vecteurs, indice minimum : 1*  *l*=[5 ,8,9]  *l*(1) vaut 5,  *l*(2) vaut 8… | *l*=[5 ,8,9]  l’indice commence à zéro,  *l*[0] vaut 5, *l*[1] vaut 8,… | Les listes **L1**, **L2**, existent dans le mode  Statistique | Les listes **List 1,**  **List 2** existent dans le menu STAT | *l*:=[5 ,8,9]  l’indice commence à zéro,  *l*[0] vaut 5, *l*[1] vaut 8,… |
| Vider une liste *l*  Créer une liste vide (Scilab,  python, Xcas) | *l* =[ ] | *l* =[ ] | **ClrList** | Menu *Stat* puis *DEL-A*  *ou* {0}→ List 1 | *l* := [ ] |
| Créer et remplir une liste  de six 0, de n+1 0  Créer  *l* =[5²,7²,9²,11²,13²] | *l*= zeros (1,6)  *Avec une boucle, ou…* | *Avec une boucle et la fonction : append* | 6 **STO** **dim**(**L1**)  Seq(X^2,X,5,13,2) **L1** | 6→Dim List 1  Dans OPTN LIST  Seq(X^2,X,5,13,2) → List 1 | *l* := [0$6],  *l* := [0$**(**n+1**)**] *ou*  *l* := [0$(k=1..6)]  *l*:= seq (k^2,k,5,13,2); |
| Ajouter un élément *a*  à la fin de la liste *l* | *Si l comporte*  *déjà n éléments :*  *l* (*n*+1)=*a* | *l*.append(*a*) | *a* **STO** **L1**(I)  I étant le premier indice non encore utilisé | Dans le menu *List,* entrer directement l’élément *a* à la fin de la liste !  (inutilisable dans un programme) | *l* := append( *l*, *a*) |
| Accès à l’élément  numéro k | *l* (k) | *l* [k] | **L1**(k) | List1[k] | *l* [k] |
| Longueur d’une liste | taille(*l*) | len(*l*) | **dim** (L1) | Dim List 1  Dans OPTN LIST | length (*l*) |

# Graphiques

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | **Quelques instructions pour les graphiques** | | | | |
|  | **Scilab** | **Python 2.6 ou 3.x**  ***from graphsecondev2\_3 import\**** [télécharger](http://www.ac-grenoble.fr/maths/guppy/articles.php?lng=fr&pg=73) le module | **TI 82-84** | **Casio 35+** | **XCAS** |
| Passer en mode grahphique / mode calcul | ***automatique*** | ***fenetre****(xmin,xmax,ymin,ymax)*  ***affiche*** *à la fin du pgm* | ***DispGraph*** | ***DrawGraph***  Dans shift PRGM DISP Grph | ***DispG***  ***DispHome*** |
| Effacer l’écran  Graphique | ***Clf***  ***Attention : clear efface les fonctions !*** |  | ***ClrDraw***  *ou*  ***EffDessin*** | ***ClrGraph***  Dans shift PRGM CLR | ***ClrGraph***  ou  ***efface*** |
| Placer un point M(*x,y*)  Choisir la taille | ***plot***(*x,y*,".")  plot([1],[1],'.','MarkerSize',1) | ***point****(x,y[,couleur])* | ***Pt****-****On****(x,y[,marque])* | ***Plot*** *x,y*  Dans shift Sketch(F4) | ***point****(x,y)* |
| Tracer le segment  [AB] avec A(*xA ; yA*)  et B(*xB ; yB*) | ***plot***([*xA*, *xB*],[ *yA*, *yB*])  ***Attention à l’ordre !*** | ***segment*** *(*(*xA , yA , xB , yB [,couleur])* | ***Line****( xA , yA , xB , yB )*  *ou*  ***Ligne****( xA , yA , xB , yB )* | *F-****line*** *xA , yA , xB , yB*  Dans shift Sketch(F4) | *A :=point( xA , yA ) ;*  *B:=point( xB , yB ) ;*  *segment (A,B) ;* |
| Tracer un cercle | t=linspace(0,2\*%pi,100);  x=xO+r\*cos(t);y=yO+r\*sin(t);  plot(x,y) | ***cercle\_cr****(x,y,r[,couleur])*  ***cercle\_cp****(x,y,s,t[,couleur])* | ***Circle****(x,y,r)*  *x et y coordonnées du centre et r le rayon* | ***Circle*** *x,y,r*  *x et y coordonnées du centre, et r le rayon*  Dans shift Sketch(F4) | ***circle****(****point****(x,y),r)*  *voir autres possibilités dans l’index* |

# Nombres aléatoires - Lois de probabilités discrètes

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | **Scilab** | **Python 2.6 *3.x*** | **TI** | **Casio** | **XCAS** | |
| Nombres aléatoires | **Module Lycée** | **Avec**  **from random import\*** | **Touche MATH/ PRB** | **Run / touche OPTN / PROB** | **Le point virgule sépare les instructions** | |
| Initialisation | rand(“seed”) | seed() | **-** | **-** | srand | |
| Nombre **réel** aléatoire dans [ 0;1[ | tirage\_reel (1,0,1)  *(liste de 1 seul réel aléatoire)* | random() | **rand** | **Rand#** | rand(0,1)  ou alea(0,1) | |
| Nombre réel aléatoire dans [ *a*;*b*[ | tirage\_reel(p,a,b)  *(liste de p réels aléatoires)* | a + (b-a) x random()  *uniform(a,b) , dans [a,b]* | a + (b-a) x **rand** | a + (b-a) x **Rand#** | rand(a,b)  ou alea (a,b) | |
| Entier aléatoire dans  {a ;a+1 ; … ; b}  avec a et b entiers | tirage\_entier(p,a,b)  *(liste de p nombres entiers aléatoires)* | randint(a,b) | **rand(**a,b) | a + **Intg**((b-a+1) x **Rand#)** | a+rand(b-a+1) ou a+alea(b-a+1) | |
| Exemple : entier aléatoire dans :  {0 ;1} puis dans {1 ;2 ;3 ;4 ;5 ;6} | L= tirage\_entier (1,0,1)  *(liste de 1 seul entier aléatoire)*  L= tirage\_entier (1,1,6) | randint(0,1)  randint(1,6) | **rand**(0,1)  **randInt**(1,6) | **Intg**(2\***Rand**#)  1+ **Intg**(6\***Rand**#) | rand(2) ou alea(2)  1+rand(6) ou 1+alea(6) | |
| Remplir une liste  *l*  avec dix nombres  Aléatoires pris dans  {1 ;2 ;3 ;4 ;5 ;6} | *l*=tirage\_entier (10,1,6) | *Avec une boucle et append* | Seq(**randInt**(1,6),X,1,10,1) | Seq(1+ **Intg**(6\***Rand**#),X,1,10,1)  Seq : Dans OPTN LIST | *l* := [(1+rand(6))$(k=1..10)]  ou  seq(1+rand(6),k,1,10) | |
| Coefficients binomiaux, loi binomiale | | | | | | |
| module |  | From loi\_discrete import\*  Télécharger  Ou avec scipy [stats.binom](http://docs.scipy.org/doc/scipy/reference/generated/scipy.stats.binom.html#scipy.stats.binom) |  |  | | - |
|  | combinaison(n,p) | nCr(n,p) | *n* Combinaison *p*  *ou n* **nCr** *p*  dans MATH PRB | n nCr p dans OPTN / PROB | | nCr(n,p) |
| Loi binomiale B(n,p)  P(X=k) | loi\_binomiale(n,p,k) | binomial(n,p,k) | binomFdp(n,p,k)  *ou* binompdf(n,p,k)  dans 2nd DISTR DISTRIB | BinomialPD(k,n,p)  Dans OPTN / STAT / DIST / BINM / Bpd | | binomial(n,p,k) |
| Loi binomiale B(n,p) cumulée  P(Xk) | 1°/ c= cumsum(binomial(p,n))  2°/ afficher c(k) | binomial\_cdf(n,p,k) | binomFRép(n,p,k)  *ou* binomcdf(n,p,k) | BinomialCD(k,n,p)  Dans OPTN / STAT / DIST / BINM / Bcd | | binomial\_cdf(n,p,k) |
| Autres lois discrètes | | | | | | |
| Loi de Poisson de paramètre  P(Xx) |  | *Consulter scipy*  [*stats.poisson*](http://docs.scipy.org/doc/scipy/reference/generated/scipy.stats.poisson.html#scipy.stats.poisson) | poissonFRép(,x)  *ou* poissoncdf(,x)  dans 2nd DISTR DISTRIB | PoissonCD(x,)  Dans OPTN / STAT / DIST / POISN / Pcd | | poisson\_cdf( ,x) |

# Lois de probabilités continues

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | **Scilab** | **Python 2.6 *3.x*** | **TI** | **Casio** | **XCAS** | |
| Nombres aléatoires | **Module Lycée** | **Avec**  **from random import\***  **from [scypi.stats](http://docs.scipy.org/doc/scipy/reference/stats.html?highlight=stats" \l "scipy.stats) import\*** | **Touche MATH/ PRB** | **Run / touche OPTN / PROB** | **Le point virgule sépare les instructions** | |
| Initialisation du calcul aléatoire  Pour les simulations | rand(“seed”) | seed() |  |  | srand | |
| Lois normales | | | | | | |
| Fontions de densité N(0,1) , Et  N(, )  et |  | **[site scipy stats.norm](http://docs.scipy.org/doc/scipy/reference/generated/scipy.stats.norm.html?highlight=norm" \l "scipy.stats.norm)**  norm.pdf(x)  norm.pdf(x, )) | dans 2nd DISTR DISTRIB normalpdf(x) et  normalpdf(x, ,) | **OPTN**, puis **STAT**, puis **DIST**  NormPD(x)  NormPD(x, ) | | normald(x)  normald(mu,sigma,x) |
| Loi normale N(, )  P(a Xb) ou P(Xx) | loi\_normale(t,,)  retourne la probabilité p(X<=t) | norm.cdf(x, )) | normalFRép(a,b,, )  normalcdf(a , b , ,) | NormCD(a,b,)  Dans OPTN / STAT / DIST / NORM / Ncd | | normald\_cdf(, ,x) |
| Loi normale centrée réduite  P(a Xb) ou P(Xx) | loi\_normale(b,1,0)  – loi\_normale(a,1,0) | norm.cdf(b)- norm.cdf(a)  norm.cdf(x) | normalFRép(a,b)  ou normalcdf(a,b) | NormCD(a,b) | | normald\_cdf(mu,sigma,a,b)  normald\_cdf(mu,sigma,x) |
| Valeur de x telle que  P(Xx)=t | **cdfnor("X",6,2,0.9,0.1)**  donne le réel x tel que  P (N (6, ) ≤ x)=0, 9. | norm.ppf(t)  ou  norm.ppf(t,, ) | FracNormal( t,)  invNorm(t) | InvNormCD(t,)  Dans OPTN / STAT / DIST / NORM / invN | | normald\_icdf(mu,sigma,t) |
| Génération de 1000 nombres aléatoires suivant une loi normale N(, ) | **grand(2,3,"nor",5,4) :**  matrice 2 × 3 dont les cœf suivent N(5,) | **from** [**scypi.stats**](http://docs.scipy.org/doc/scipy/reference/stats.html?highlight=stats#scipy.stats) **import\***  norm.rvs(, ) | normAleat(,  randNorm(, ,1000) | avec seq | | *seq(randnorm(*,  *),j,1,1000)* |
| Autres lois continues | | | | | | |
| Lois exponentielle s de paramètre l | - | Dans scipy : from  [**Stats.expon**](http://docs.scipy.org/doc/scipy/reference/generated/scipy.stats.expon.html#scipy.stats.expon) import\* | - | - | | - |
| Fonction de densité | - | **from** [**scypi.stats**](http://docs.scipy.org/doc/scipy/reference/stats.html?highlight=stats#scipy.stats) **import\***  **expon.pdf(x,0,1/**l)  ou bien  **expon.pdf(x,scale=1/**l) | - | - | | calcul direct |
| P(Xx) | **loi\_exp(** l **,x)**  retourne la probabilité p(X<=x) lorsque X suit la loi exp de param l | **from** [**scypi.stats**](http://docs.scipy.org/doc/scipy/reference/stats.html?highlight=stats#scipy.stats) **import\***  **expon.cdf(x,0,1/**l)  ou bien  **expon.cdf(x,scale=1/**l) | - | - | | calcul direct |
| Génération de 1000 nombres aléatoires suivant une loi exp. De paramètre l | **grand(2,3,"exp",** l**)**  donne une matrice 2 × 3 dont les cœf suivent la loi exp de param l | **from [scypi.stats](http://docs.scipy.org/doc/scipy/reference/stats.html?highlight=stats" \l "scipy.stats) import\***  expon.rvs(0,1/ l ,size=1000)  ou bien  expon.rvs(scale=1/ l ,size=1000) | - | - | | *seq(randexp(* l*),j,1,1000)* |

# Constantes - fonctions

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | **Quelques fonctions courantes - Constantes - Définir une fonction** | | | | | |
|  | **Scilab** | **Python 2.6 *3.x seult.*** | **TI** | **Casio** | | **XCAS** |
| Racine carrée | sqrt | sqrt (from math import\*) | touche directe | touche directe | | sqrt |
| Puissance a^b | a^b | a\*\*b ou pow(a,b) (from math import\*) | a^b | a^b | | a^b |
| Valeur absolue | abs | fabs (from math import\*) | abs | Abs dans OPTN NUM | | abs |
| ln, exp | log, exp | log exp (from math import\*) | ln exp | ln exp | | ln exp |
| Partie entière(\*) , plus grand entier relatif inférieur ou égal au nombre considéré | floor | floor (from math import\*) | Int ou PartEnt dans MATH NUM | Intg dans OPTN / NUM | | floor |
| Troncature(\*), c'est-à-dire nombre sans sa partie décimale éventuelle | int | trunc (from math import\*) | iPart dans MATH NUM | Int dans OPTN / NUM | | iPart |
| Quotient division euclidienne | quotient (a,b) | *V 2.6* a/b *v 3.x a//b*  (avec a et b entiers) | Int(A/B) ou PartEnt(A/B) | Intg(A/B) | | iquo(a,b) |
| Reste division euclidienne | reste(a,b) | a%b , ou fmod(a,b) (from math import\*) | A-B\* Int(A/B) | A-B\* Intg(A/B) | | irem(a,b) |
| pi, e,  *i* | %*pi,* %*e*, %*i* | pi, e,1*j*  (from math import\*) | touche directe | touche directe | | pi, e, *i* |
| Définir une fonction  Par exemple *f*(*x*)=4*x*3+2*x*+1 | function y=f(x)  y=4\*x^3+2\*x+1  endfunction | def f(x):  return 4\*x\*\*3+2\*x+1 | Touche Y=  Y1=4\*X^3+2\*X+1 | Menu GRAPH  Y1= 4xX^3+2xX+1 | | f(x) :=4\*x^3+2\*x+1 ;  *ou pour des cas élaborés*  f(x) :={  return 4\*x^3+2\*x+1  }; |
| Saisir une fonction  Exemple de réponse | rep=input( " f(x) vaut :"+"string")  deff (‘y=f(x)’, "y=”+rep)  4\*x^3+2\*x+1 |  | :Prompt Y1  **"**4\*X^3+2\*X+1 |  | | input(f) ou saisir(f)  x -> 4\*x^3+2\*x+1 |
| puis, obtenir une image  *f*(2) , *f*(*a*) | f(2) , f(*a*) | f(2) , f(*a*) | Y1(2), Y1(A), ATTENTION :  Y**1** *se trouve dans*  VARS /Y-VARS / Function | 2→X : Y1  A→X : Y1  ATTENTION :  *Pour obtenir* Y : VARS/GRPH/F1[Y] | Y1(2)  Y1(A)  Sur 35+ USB | f(2), f(*a*) |

(\*) Les fonctions Partie entière et Troncature différent sur les nombres négatifs : si *x*=-2,31 : partie entière : -3, troncature -2, partie décimale : 0,31, partie « fractionnaire » : 0,69

# Matrices

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | ***Matrices : saisie, opérations*** | | | | |
|  | ***Scilab*** | ***Python 2.6 3.x seult.*** | ***TI*** | ***Casio*** | ***XCAS*** |
| Définir une matrice M carrée à *n* lignes et *n* colonnes, remplie de zéros |  | from numpy import\* | touche MATRX / menu Edit | ►MAT  ou Menu / Mat   * Sélectionner une matrice * Taper le nombre de lignes, par exemple 3, il apparait 3x0 * Remplacer le 0 par le nombre de colonnes | M :=matrix [*n,n*] ;  *Attention : les indices varient de 0 à n-1* |
| Saisie en ligne des coefficients de | M =[1 2 3 ;4 5 6 ;7 8 9 ]  *On sépare les coefficients par des espaces ou des virgules*  *Le point-virgule indique un passage à la ligne suivante* | M=array([(1,2,3),(4,5,6),(7,8,9)]) | [[1,2,3][4,5,6][7,8,9]][M] | Compléter la matrice affichée | M := [[1,2,3],[4,5,6], [7,8,9]] |
| Utiliser un tableur pour saisir les coefficients | - | - | Directement dans le  menu Matrix/EDIT | Directement dans ►MAT  ou Menu / Mat | Ouvrir le tableur avec Alt+t  Variable : M, nom de la matrice.  On peut alors utiliser directement M dans les calculs et programmes de la session. |
| Créer la matrice identité d’ordre *n* |  |  |  |  |  |
| Utiliser M dans une instruction | M | M | Menu Matrix/NOMS  Affichage : [M] | Mat M dans OPTN / MAT | M |
| Accéder au coefficient de la ligne *l* et de la colonne *c* de la matrice | M(*l,c*) | M[*l,c*] | [M] (*l,c*) | Mat [*l,c*] | M[*l,c*] |
|  |  |  |  |  |  |
| Afficher une matrice dans un programme | disp(M) ou afficher (M) | print M | Matrix/NOMS M : [M] Enter | Mat M◢ | afficher( M) ; |
| Dimensions de M (liste des dimensions) | size (M) ou taille(M) | M.shape |  | DIM dans ►MAT | dim(M) |
| Additionner, multiplier deux matrices A et B | A+B A\*B | A+B dot(A,B) | Affichage : [A]+[B] [A]\*[B] | Mat A+Mat B Mat A\*Mat B | A+B A\*B |
| A^n (puissance entière de A) avec n **fixé** ( par exemple A^5 ) | A^n | - | A^n | Mat A^n | A^n |
| A^n avec le **paramètre n**, A étant diagonalisable | *-* | - | - | *-* | assume n>0 ;  B := matpow(A,n) ; |
| Multiplier M par un réel *k* | *k*\*M ou M\* *k* | *k*\*M | Affichage : *k*\*[M] | *k*\*M | *k*\*M |
| Matrice inverse de M | 1/M | linalg.inv(M) | Touche *x-1* : [M]-1 | Touche *x-1* : Mat M-1 | inverse(M) |
| Résoudre A\*X=B, X vaut : | A\B  ou (1/A)\*B | linalg.inv(A)\*B | [A] -1 \*[B] | Mat A-1 \* Mat B | inverse(A)\*B |
| Multiplication élément par élément | A.\*B | A\*B |  | - | A.\*B |
| Remplir automatiquement une matrice à l’aide de formules,  par exemple |  | Voir la ressource n°300 | remplir | Fill ?  Dans OPTN / MAT |  |
|  |  |  |  |  |  |