MINF0402 - Février 2020 - *Durée 1h00*

L'usage de tout document, calculatrice, téléphone, boulier ou divination est interdit, toutes les réponses doivent être justifiées. N'oubliez pas - d'écrire votre nom, vos prénoms, de numéroter chacune de vos copies, d'y reporter votre numéro de place et d'en indiquer le nombre total (-1pt par omission). Pour l'évaluation de la copie, il sera tenu compte de la rigueur, de la qualité et de la propreté de la rédaction.

ic v0.7 - corr 1.3

Exercice 1 (Lab in the Sky)

Que donne à l'affichage, l'exécution à la console scilab des commandes successives suivantes, c'est à dire que contiennent x, y, a ... après ces instructions ou que s'affiche t-il à l'écran ? Attention : certaines instructions génèrent une erreur. (*rappel : la sortie de la première instruction est un vecteur (ou tableau) ligne*)

```
> x=1:3
> y=2:2:6
> a=x+y
> b=x*y
> c=x.*y
> d=x*y'
> z=[x y]
> w=[x;y]
> f=x(5)
> x(5)=10
> A=[3 4 5 6;2 1 7 8; 12 10 11,9]
> B=A([3 1],[2 4])
> C=zeros(B)
> D=matrix(1:9,3,3)
> E=diag(D)
> F=diag(E)
> t=linspace(-10,10,11)
> plot(t,100-t.*t);plot(t,100-t.*t,'r*');
```

Exercice 2 (Du coté de Sherlock)

Résoudre les systèmes suivants : (NB : toutes les solutions sont entières)

$$(S_1)\begin{cases} x+2y-2z+t=-2\\ -3x-8y+6z-2t=10\\ 2x+4y-3z+t=-4\\ -x-4y+4z-t=10\end{cases}\qquad (S_2)\begin{cases} x+2y-z-2t=-5\\ 2x+3y-4z-4t=-14\\ -2x-5y+z+5t=10\\ -x-y+3z+t=8\end{cases}\qquad (S_3)\begin{cases} x+2y-z+2t=7\\ -2x-7y+z-t=-10\\ x-y-z+7t=12\\ -x-5y+2z+3t=-3\end{cases}$$

Exercice 3 (Algo est lent ?)

On cherche à résoudre un système linéaire de la forme $AX=B$ de la forme suivante :

$$\begin{pmatrix} a_{11} & a_{12} & a_{13} & a_{14} & \cdots & \cdots\cdots & \cdots & a_{1n}\\ a_{21} & a_{22} & a_{23} & a_{24} & \cdots & \cdots & \vdots & a_{2n}\\ 0 & a_{32} & a_{33} & a_{34} & \cdots & \cdots & \vdots & a_{3n}\\ 0 & 0 & a_{43} & a_{44} & \cdots & \cdots & \vdots & \vdots\\ \vdots & \vdots & \ddots & \ddots & \cdots & \cdots & \vdots & \vdots\\ \vdots & \vdots & \ddots & \ddots & \ddots & \cdots & \vdots & \vdots\\ \vdots & \vdots & \ddots & \cdots & 0 & a_{(n-1)(n-2)} & a_{(n-1)(n-1)} & a_{(n-1)(n)}\\ 0 & \cdots & \cdots & \cdots & \cdots & 0 & a_{n(n-1)} & a_{nn}\end{pmatrix}\begin{pmatrix} x_1\\ x_2\\ x_3\\ \vdots\\ \vdots\\ \vdots\\ x_{n-1}\\ x_n\end{pmatrix}=\begin{pmatrix} b_1\\ b_2\\ b_3\\ \vdots\\ \vdots\\ \vdots\\ b_{n-1}\\ b_n\end{pmatrix}$$

A est donc une matrice (n,n) "ressemblant" à une matrice triangulaire supérieure, mais avec donc des termes supplémentaires - a priori non nuls - juste en dessous de la diagonale principale.

Par rapport à une matrice triangulaire supérieure, on a donc les termes : $a_{21}, a_{32}, a_{43}, ..., a_{(n-1)(n-2)}, a_{n(n-1)}$ devant a priori être considérés comme non nuls.

Ecrire un algorithme de résolution “à la Gauss” **tenant compte de la forme particulière de la matrice** en pseudo code, on expliquera le fonctionnement. C'est en fait très simple et usuel : la première étape transformera le système en un système triangulaire supérieure qu'il suffira de résoudre dans la seconde étape. (On suppose bien entendu qu'il n'y a pas de problème de résolution : on peut effectuer la réduction de Gauss, sans devoir effectuer de permutation de lignes).

Il doit être clair qu'on ne vous demande pas de réécrire ici simplement - de mémoire - l'algorithme de Gauss général vu en TD ce qui - le cas échéant - ne compterait pour rien, mais de l'adapter dans ce cas spécifique

— • • —