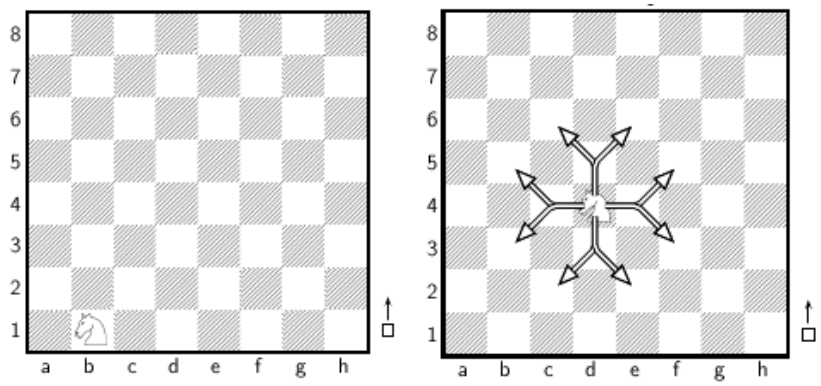
1 Le cavalier seul

Objectif

Créer un programme qui est capable de prédire les déplacements du cavalier sur un échiquier. Ce programme doit afficher la position du cavalier d'une couleur, et les positions possibles d'une autre couleur. L'utilisateur pourra ensuite rentrer de nouvelles coordonnées. La nouvelle position, ainsi que les nouvelles possibilités seront alors affichées. Prévoir également la possibilité pour l'utilisateur d'arrêter de jouer.

Quelques rappels

La position initiale du cavalier gauche est illustrée sur la figure de gauche. C'est donc à cette position que devra se trouver votre cavalier au début du jeu. Aux échecs, les déplacements du cavalier se font de deux cases dans une direction (colonne ou ligne) et d'une case dans l'autre direction (respectivement ligne ou colonne) comme illustré sur la figure de droite.



Étapes à suivre

1. Créer une matrice E de dimension (8,8) ne contenant que des zéros. Vous pouvez utiliser la fonction `zeros` (pour en savoir plus, taper `help zeros`). La représentation graphique de cette matrice E sera notre échiquier.
2. Dans la case où se trouve le cavalier, remplacer le 0 par un 1.
3. Construire une nouvelle matrice, de dimension (8,2), qui contiendra les indices (ligne, colonne) des 8 cases où le cavalier peut se déplacer.
4. Indiquer ces cases sur l'échiquer, en remplaçant les 0 de la matrice E par des 2 (par exemple) aux cases concernées. Attention, il faut vérifier que le cavalier reste sur l'échiquier !
5. Représenter graphiquement la matrice E, en utilisant la fonction imagesc. La ligne de commande à indiquer est :
   `imagesc(E) ; axis equal ; axis tight ; axis xy`
6. Demander à l'utilisateur s'il souhaite continuer à jouer.
7. Si c'est le cas, lui demander la nouvelle position qu'il choisit pour le cavalier. Vérifier qu'elle fait bien partie des positions autorisées. Sinon, lui redemander une position jusqu'à ce qu'il rentre un choix correct.
8. Le programme revient alors au début, calcule les nouveaux déplacements possibles, affiche le nouvel échiquier... ainsi de suite, jusqu'à ce que l'utilisateur décide d'arrêter de jouer !

2 Résolution d'équations linéaires

2.1 Résolution directe d'équations avec Matlab

Objectif

Transformer un système d'équations en système matriciel, et inverser la matrice des coefficients.

À faire

Soit à résoudre le système :

$$\begin{cases} x + 3y + 5z = 6 \\ 2x - y = 0 \\ 5x + 4y + 3z = 3 \end{cases}$$

1. Écrire la matrice A et la matrice Y de façon à obtenir le système sous la forme $AX = Y$.
2. Calculer X en posant $X = A^{-1}Y$. Pour calculer l'inverse d'une matrice en Matlab utiliser la fonction `inv( )`.
3. Écrire le résultat.

2.2 Par la méthode du pivot

Objectif

On veut résoudre un système d'équations linéaires par la méthode du pivot de Gauss.

À faire

Soit à résoudre le système suivant de 3 équations à 3 inconnues par la méthode du pivot :

$$\begin{cases} -0.04x + 0.04y + 0.12z = 3 \\ 0.56x - 1.56y + 0.32z = 1 \\ -0.24x + 1.24y - 0.28z = 0 \end{cases}$$

1. Définir la matrice des arguments. Pour cela on met les coefficients de x dans la première colonne, de y dans la deuxième colonne, de z dans la troisième colonne et du membre de droite dans la quatrième colonne. On obtient une matrice 3×4.
2. On choisit la première ligne comme pivot. On effectue des opérations sur les lignes et les colonnes de façon à obtenir des zéros sur la première colonne sauf à la première ligne où l'on veut obtenir 1. On stocke ce résultat dans le tableau b, qui doit avoir la forme :

$$b = \begin{pmatrix} 1 & -1 & -3 & -75 \\ 0 & -1 & 2 & 43 \\ 0 & 1 & -1 & -18 \end{pmatrix}$$

3. On choisit le deuxième ligne comme pivot et on effectue des opérations sur les lignes et les colonnes de façon à obtenir des zéros sur la deuxième colonne sauf à la deuxième ligne où l'on veut obtenir 1.
4. On choisit le troisième ligne comme pivot et on effectue des opérations sur les lignes et les colonnes de façon à obtenir des zéros sur la troisième colonne sauf à la troisième ligne où l'on veut obtenir 1. En déduire les valeurs de x, y et z.
5. Proposer un algorithme qui permette ainsi de résoudre n'importe quel système d'équations connaissant la matrice des arguments.

3 Calcul d'indice de végétation dans une image LandSat

Objectif

Nous allons créer un programme qui va analyser une image multispectrale LandSat. Cette image contient quatre canaux (initialement, elle en contient sept, mais pour plus de simplicité, nous en avons retiré trois).

Les trois premiers canaux contiennent des informations dans le visible :
- Canal 1 : Bleu (485 nm)
- Canal 2 : Vert (560 nm)
- Canal 3 : Rouge (660 nm).

Le quatrième canal contient des informations dans le Proche-infrarouge PI (830 nm). L'information contenue dans notre image s'appelle la réflectance.

L'objectif de notre programme est de calculer l'Indice de Végétation nommé NDVI (*Normalized Difference Vegetation Index*). Le NDVI est l'indice de végétation le plus communément utilisé pour estimer la quantité de végétation présente sur une surface. Cet indice est calculé comme le rapport normalisé de la différence entre la réflectance proche-infrarouge et la réflectance acquise dans la bande spectrale rouge c'est-à-dire :

$$NDVI = (R_{PI} - R_R)/(R_{PI} + R_R),$$

où R_{PI} est la valeur du canal 4 (Proche Infrarouge) et R_R la valeur du canal 3 (Rouge). Un NDVI compris entre 0.2 et 0.8 indique la présence de végétation.

Étapes à suivre

1. Charger l'image LandSat dans un tableau à une dimension, en utilisant les lignes de commande suivantes :
```
fichier='../landsat.img' ;
fid = fopen(fichier) ;
DATA= fread(fid) ;
```

2. Mettre l'image sous forme d'un tableau nommé IMG de $512 \times 512 \times 4$ cases. 512×512 est le nombre de pixels dans l'image en X et Y. Le "4" correspond au nombre de canaux. Pour cela, vous pouvez effectuer des boucles successives, ou bien utiliser la fonction `reshape` (pour en savoir plus, taper `help reshape`).

3. Visualiser la partie visible de l'image en utilisant la fonction `imagesc`. Attention, cette fonction nécessite d'avoir des composantes rouge/vert/bleu entre 0 et 1. Vous pourrez par exemple définir une nouveau tableau, appelé rgb :
```
rgb=IMG( :, :,1 :3)/max(max(max(IMG))) ;
imagesc(rgb)
```

4. Extraire l'information de deux pixels particuliers sur les 4 canaux. Vous prendrez les coordonnées suivantes pour ces deux pixels :
   x1=240 ; y1=140 ;
   x2=322 ; y2=135 ;
   Vous calculerez le NDVI pour ces deux pixels (que l'on nomme ndvi1 et ndvi2 et qui sont des scalaires). Que pouvez-vous en dire ?

5. Calculer le NDVI sur l'ensemble de l'image. On appelera alors le résultat (qui est un tableau) IV.

6. Représenter graphiquement l'indice de végétation, en utilisant à nouveau `imagesc`.