Les spécificités de Matlab

Matlab n'utilise qu'un seul type de variable, les matrices de nombres complexes. Elles n'ont pas besoin d'être déclarées au préalable : Matlab les crée et adapte leur taille au fur et à mesure de l'éxécution.
Cela a des conséquences sur le temps de calcul, notamment lors de l'utilisation de boucles, ce que je vais mettre en évidence par des exemples.

Allocation préalable des matrices utilisées dans les boucles

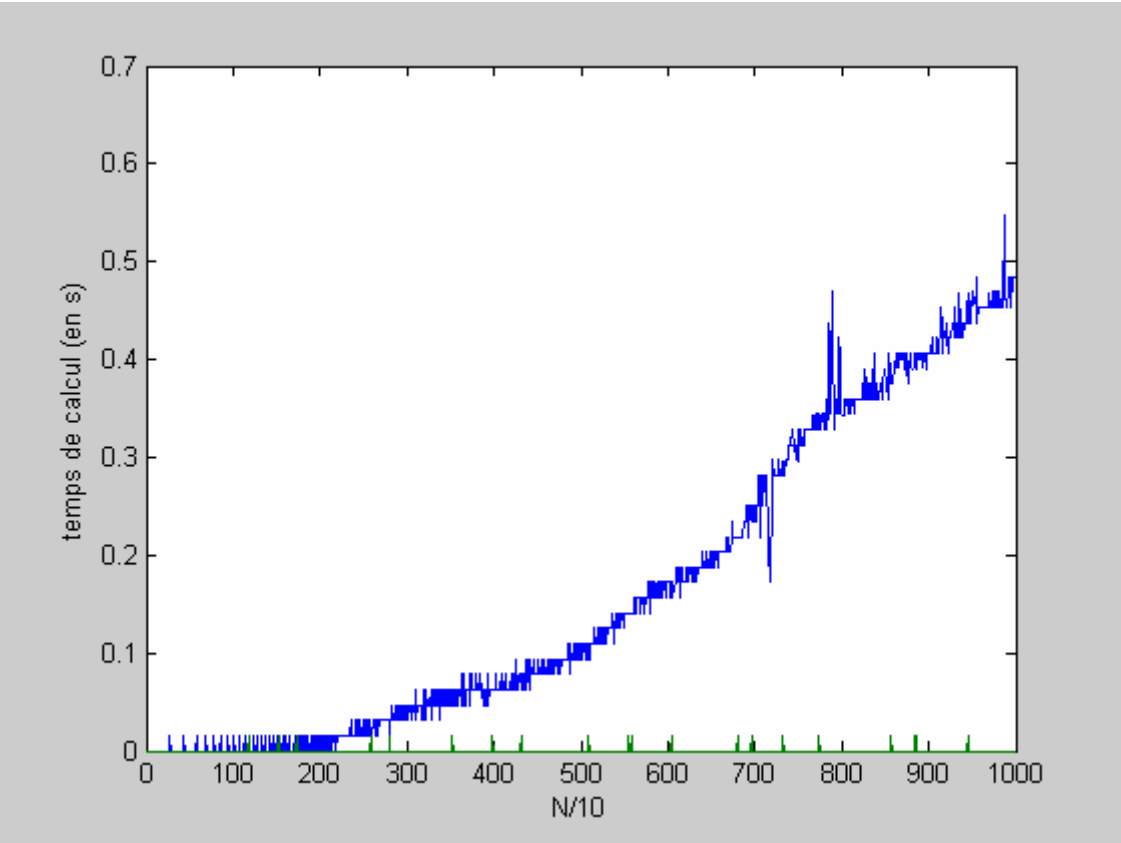
J'ai choisi un exemple très simple, la création de la séquence des premiers carrés (1 4 9 …) à l'aide d'une boucle *for*.
Les fonctions *tic* et *toc* permettent de calculer le temps mis pour chacune des deux méthodes utilisées.
J'ai réuni ce test dans une fonction *test.m*, dont voici le contenu :

```
function [t,t2]=test(N)

% calcul sans prédimensionnement de S
tic;
for i=1:N;
   S(i)=i^2;
end;
t=toc;

% calcul avec prédimensionnement de S
clear S;
tic;
S=zeros(1,N);
for i=1:N;
   S(i)=i^2;
end;
t2=toc;
```

Le fait que Matlab n'impose pas de prédimensionner les variables permet une grande flexibilité, mais qui présente ici un inconvénient. En effet, dans la première méthode, S est défini au fur et à mesure d'une boucle. Lors du premier tour, Matlab alloue la taille nécessaire, c'est-à-dire une matrice 1*1. Au deuxième tour, il doit étendre la matrice, ce qu'il ne peut faire qu'en allouant une nouvelle matrice 1*2, en recopiant laa matrice 1*1 au début de la nouvelle matrice, en complétant le nouvel élément de la matrice 1*2, et en désallouant la matrice 1*1. Cette succession d'opérations se répète à chaque tour, ce qui prend un temps considérable. La solution est simple : il suffit de prédimensionner la matrice S, ce qui est fait dans la seconde méthode.

J’ai comparé les temps de calcul pour différentes valeurs de N (de 10 à 10000 en allant de 10 en 10). Je trace ensuite les deux séries de valeurs obtenues en fonction de N. Voici le script utilisé et les courbes obtenues :

```
for N=1:1000;
        [result(N,1),result(N,2)]=test(N*10);
end ;
plot(result),
xlabel('N/10'),
ylabel('temps de calcul (en s)')
```



Analyse des résultats obtenus

Pour la syntaxe sans prédimensionnement, le temps croit nettement avec N, et les temps de calcul sont très mauvais (de l’ordre de quelques dixièmes de secondes pour N=10000).
Pour la seconde syntaxe, le temps de calcul est quasi-constant, et est très proche de zéro. Une simple instruction supplémentaire suffit donc à améliorer de façon conséquente les temps de calculs, on a donc tout intérêt à y prendre garde.

Exploitation des capacités de traitement matriciel pour éviter les boucles.

Matlab est un langage interprété : chaque ligne est lue et interprétée pendant l'exécution. Les opérations nécessaires sont effectuées par des appels de fonctions en assembleur. Si une boucle est appelée 10000 fois, l'opération d'interprétation est répétée 10000 fois. C'est la différence avec un langage compilé dans lequel chaque ligne est analysée et décodée une seule fois par le compilateur pour générer un programme exécutable en assembleur. Si une boucle est exécutée 10000 fois, seules les opérations en assembleur sont répétées. Un langage compilé est donc intrinsèquement plus rapide. L'inconvénient est qu'il faut répéter la compilation à chaque modification du code source.
Matlab est optimisé pour le calcul vectoriel et matriciel et il est souvent possible de remplacer une boucle par une opération matricielle directe. C'est un gain de temps puisque la ligne n'est interprétée qu'une seule fois.
J'ai choisi l'exemple du redressement d'un signal sinusoïdal. Une nouvelle fois, j'ai comparé deux écritures différentes.

```
function [t,t2]=test2(N)

n=0:N-1;
x=sin(2*pi*0.05*n);
% prédimensionnement des matrices :
y=zeros(1,N); z=y;

% utilisation d'une boucle for
tic;
for i=1:N;
   if x(i)>0
      y(i)=x(i);
   end;
end;
t=toc;

% utilisation d'une syntaxe matricielle
tic;
i=find(x>0);
z(i)=x(i);
t2=toc;
```

Pour N=100 000, la méthode utilisant la boucle *for* met 0.2660 secondes, alors que l'autre met 0.0320 secondes. On peut conclure qu'une syntaxe matricielle est donc à préférer chaque fois que c'est possible.