TP 6

14 mai 2012

L'objectif de ce tp est l'implémentation de fonctions permettant de traiter de manière relativement générique les problèmes d'optimisation linéaire de première ou de seconde espèce.

1 Retour sur la méthode du simplexe

Exercice 1

1. La fonction `STANDARD1`

```
function M=STANDARD1(A,d,c)
    s=size(A);
    //nombre de contraintes de A s(1)
    Ineg=s(1);
    D=diag(ones(1,Ineg));
    funcout=[c,zeros(1,Ineg+1)];
    M=[A,D,d;funcout];
endfunction
```

2. La fonction `DANTZIG1`

```
function Index=DANTZIG1(M)
    Index=zeros(2,1);
    //premier critère de Dantzig:la colonne max des couts marginaux
    [Max,k]=max(M($,1:($-1)));
    Index(1)=k;
    //deuxième critère de Dantzig: la ligne minimum des ratio
    I=find(M(1:$-1,k)>%eps);
    lengI=length(I);
    Index(2)=I(1);
    minimum=M(I(1),$)/M(I(1),k);
    if (lengI>1)
        for i=2:lengI
            ratio=M(I(i),$)/M(I(i),k);
            if(ratio<minimum)
                minimum=ratio;
                Index(2)=I(i);
            end
        end
    end
    printf('Pivot à l''indice (%i,%i)\n',k,Index(2));
endfunction
```

3. La fonction `SIMPLEXE1`

```
function M=SIMPLEXE1(A,d,c)
    printf('Algorithme du simplexe\n');
```

```
    //Formulation standard
    M=STANDARD1(A,d,c);
    NIterMax=1000;
    NIter=0;
    tol=%eps;
    Max=max(M($,1:($-1)));
    while (Max>tol & (NIter<NIterMax))
        //Choix du pivot
        Id=DANTZIG1(M);
        //PIVOTAGE
        M=PIVOTGJ(M,Id(1),Id(2));
        NIter=NIter+1;
        Max=max(M($,1:$-1));
    end
    //[sol,val]=EXTRAIRESOL(M);
endfunction
```

4. On peut générer une matrice aléatoire (un vecteur) de n lignes et m colonnes avec des entrées entières dans l'intervalle [a,b] avec la commande :

   `floor((b-a)*rand(n,m)+a)`

Exercice 2

1. La fonction `STANDARD2`

```
function M=STANDARD2(A,d,c,Ineg)
    if (size(d,1)<size(d,2))
        d=d';
    end
    if (size(c,1)>size(c,2))
        c=c';
    end
    s=size(A,1);
    alpha=zeros(s,1);
    D=diag(ones(1,s));
    for i=1:s
        if(Ineg(i)==-1)
            D(i,i)=-1;
            alpha(i)=1;
            end
    end
    funcout=[c,zeros(1,s+2)];
    funcoutaux=zeros(1,s+size(A,2)+2);
    funcoutaux(1,$-1)=-1;
        N=[A,D,alpha,d;funcoutaux;funcout];
    for j=1:s
        if(Ineg(j)==-1)
            funcoutaux=funcoutaux+N(j,:);
        end
    end
    M=[A,D,alpha,d;funcoutaux;funcout];
endfunction
```

2. La fonction DANTZIG2

```
function Index=DANTZIG2(M,phase)
    Index=zeros(2,1);
    tol=1e-8;
    //premier critère de Dantzig:la colonne max des couts marginaux
    if (phase==1)
        [Max,k]=max(M($-1,1:($-2)));
        if(Max<tol)//maximum atteint dans la première phase, index(1)=0
                   //donne l'indication que max atteint
            Index(1)=0;
        else //sinon on doit pivoter et on fait entrer la k ième colonne
            Index(1)=k;
        end
    else
        [Max,k]=max(M($,1:($-2)));
        if(Max<tol)//maximum atteint dans la deuxième phase,
                   // index(1)=0 donne l'indication que max atteint
            Index(1)=0;
        else
            Index(1)=k;//sinon on doit pivoter et on fait entrer la
                       //k ième colonne
        end
    end
    //deuxième critère de Dantzig: la ligne minimum des ratio
    if(Index(1)>0)//si le maximum n'est pas atteint, on cherche la
                  //variable sortante
        if(phase==1)
            I=find(M(1:$-2,k)>%eps);
        else
            I=find(M(1:$-2,k)>%eps);
        end
        if(length(I)==0)//il n'y a aucune ligne pour laquelle le coefficient
                        //est strictement positif
            Index(2)=0;
            if(phase==1)//si on ne peut pas pivoter le problème
                        //de la phase 1 n'est pas borné.
                printf('Problème non réalisable\n');
            else//si on ne peut pas pivoter le problème de la phase
                //2 n'est pas borné.
                printf('Problème non borné\n');
            end
        else
            lengI=length(I);
            Index(2)=I(1);
            minimum=M(I(1),$)/M(I(1),k);
            if (lengI>1)
                for i=2:lengI
                    ratio=M(I(i),$)/M(I(i),k);
                    if(ratio<minimum)
```

```
                        minimum=ratio;
                        Index(2)=I(i);
                    end
                end
            end
            printf('Pivot à l''indice (%i,%i)\n',k,Index(2));
        end
    end
endfunction
```

3. La fonction SIMPLEXE2

```
function M=SIMPLEXE2(A,d,c,ineg)
    printf('Algorithme du simplexe\n');
    //Formulation standard
    M=STANDARD2(A,d,c,ineg);
    NIterMax=1000;
    NIterPh1=0;
    NIterPh2=0;
    tol=1e-8;
    //première phase
    print('Phase1\n');
    Id=DANTZIG2(M,1);
    while (Id(1)>0 & Id(2)>0 & (NIterPh1<NIterMax))
        NIterPh1=NIterPh1+1;
        //pivotage
        M=PIVOTGJ(M,Id(1),Id(2));
        //Choix du pivot
        Id=DANTZIG2(M,1);
    end
    print('Nombre d''itérations à la phase 1:%i\n',NIterPh1);

    if(abs(M($-1,$))<tol)
        print('Phase2');
        //deuxième phase
        Id=DANTZIG2(M,2);
        while (Id(1)>0 & Id(2)>0 & (NIterPh2<NIterMax))
            NIterPh2=NIterPh2+1;
            //pivotage
            M=PIVOTGJ(M,Id(1),Id(2));
            //Choix du pivot
            Id=DANTZIG2(M,2);
        end
        print('Nombre d''itérations à la phase 2:%i\n',NIterPh2);
    end
endfunction
```