École Nationale Supérieure d'Électrotechnique, d'Électronique, d'Informatique, d'Hydraulique et des Télécommunications.

RAPPORT DU PREMIER PROJET DE RECHERCHE OPÉRATIONNELLE : PROGRAMMATION LINEAIRE.

Fait par :
BOND Tom.
MOUGENEL Yann.
AARAB Zakaria.
ROYER Thomas.
BROSSARD Thibault.
EL MAHI Moncef.

Encadré par :
Mr MARTHON Philippe.

24 octobre 2014

Sommaire

1 Introduction 2
2 La méthode du Simplexe 2
2.1 Principe de l'algorithme sur un exemple : 2
2.2 Implémentation en Matlab : 5
2.3 Tester notre algorithme 6
2.3.1 Validation du problème du brasseur 6
3 L'interface graphique 8
4 La méthode de Gabasov ou la méthode adaptée 9
4.1 Explication de l'algorithme 9
4.2 Differences avec le simplexe 9
4.3 Tests et comparaison avec le simplexe 10
4.3.1 Test de l'algorithme sur le problème du brasseur 10
4.3.2 Comparaison avec l'algorithme du simplexe 10
5 Limites de nos algorithmes 10
6 Organisation du groupe 11
7 Conclusion 11

1 Introduction

Le projet de cette année consiste à programmer en Matlab différents algorithmes de programmation linéaires : Dans notre groupe, on a choisit de travailler sur les algorithmes du Simplexe et de Gabasov, et de les tester sur des problèmes d'optimisation issus de la vie courante : (le problème du brasseur par exemple)

On réalisera une interface graphique générale permettant à n'importe quel utilisateur de tester tout problème d'optimisation, on veillera à ce qu'elle soit conviviale et simple d'utilisation.

On essaiera d'évaluer le temps de calcul en faisant varier le nombre de variables et de contraintes, de comparer les algorithmes les uns avec les autres, de mettre en œuvre leurs performances et finalement de déterminer leurs limites.

2 La méthode du Simplexe

2.1 Principe de l'algorithme sur un exemple :

Soit le système suivant contenant une fonction économique à maximiser ainsi que des contraintes à respecter.

Visualisons les étapes de l'algorithme du simplexe sur cet exemple pour mieux assimiler la partie théorique abordée dans les sections précédentes.

$$\begin{cases} max\ z & = & a+2b \\ -3a+2b & \preceq & 2 \\ -a+2b & \preceq & 4 \\ a+b & \preceq & 5 \end{cases}$$

Le but de l'exemple est donc de trouver les paramètres a et b maximisant (ou minimisant, ça dépend du problème) le coût.

Les étapes à suivre sont :

1. Ajout des variables d'écart. (Dans le cas où on a des inégalités) dans le but de transformer le système en un système d'égalités.
2. Déterminer une solution de base réalisable : Pour cela on donne à a et b les valeurs minimales qu'elles peuvent prendre. Dans cet exemple a=0 et b=0.
3. Cette solution est-elle optimale ?
   Si oui alors on arrête l'algorithme.
   sinon on trouve une solution admissible . Pour cela on suit les étapes de l'algorithme suivant jusqu'à obtenir la solution optimale. (Le calcul du nouveau tableau correspond à une itération)

   - Tableau.
   - Calcul de la variable entrante.
   - Calcul de la variable sortante.
   - Mise à jour du tableau.
   - Vérifier le critère de sortie.

La première étape consiste à ajouter les variables d'écart à notre système composé de trois inégalités. (Les variables d'écart introduites lorsqu'on a des inégalités $\preceq$ sont les vecteurs $(0, 0, ..., 1, ..., 0, 0)^T$).

Le système devient alors :

$$\begin{cases} max\ z &= a+2b \\ -3a+2b+e_1 &= 2 \\ -a+2b+e_2 &= 4 \\ a+b+e_3 &= 5 \end{cases}$$

$$\forall i \in [|1,3|] \quad e_i \succeq 0 \quad a \succeq 0 \quad b \succeq 0$$

L'étape 2 consiste à trouver la solution de base réalisable. Pour ce faire, on donne à a et b la valeur 0.

$$a \ = \ b \ = 0 \Rightarrow z \ = \ 0$$

z n'est pas maximale dans ce cas, on est obligé de faire une itération. Les vecteurs de base obtenus sont alors :

$$\begin{cases} e_1 &= 2 \\ e_2 &= 4 \\ e_3 &= 5 \end{cases}$$

On représente alors notre matrice où a et b sont les vecteurs hors-base, $e_1, e_2\ et\ e_3$ sont les vecteurs de la base. On a alors la matrice suivante :

	a	b	e_1	e_2	e_3	C	K
e_1	-3	2	1	0	0	2	
e_2	-1	2	0	1	0	4	
e_3	1	1	0	0	1	5	
z	1	2	0	0	0	0	

La deuxième étape consiste à déterminer la variable entrante qu'on appelle la colonne entrante : On commence par prendre le max des coefficients de z. Dans notre cas le maximum est 2. Par conséquent, la colonne du pivot est la deuxième colonne. Le vecteur b est alors le vecteur entrant. (les valeurs de la colonne entrante seront représentées en gras pour des raisons de lisibilité)

	a	**b**	e_1	e_2	e_3	C	K
e_1	-3	**2**	1	0	0	2	
e_2	-1	**2**	0	1	0	4	
e_3	1	**1**	0	0	1	5	
z	1	**2**	0	0	0	0	

L'étape suivante consiste à calculer les valeurs de la colonne K. Ces dernières se calculent en divisant terme à terme les valeurs de la colonne C par celles de la colonne entrante.

$$K_i = \frac{C_i}{b_i} \quad \forall i \in [|1,4|]$$

On obtient alors :

	a	**b**	e_1	e_2	e_3	C	K
e_1	-3	**2**	1	0	0	2	1
e_2	-1	**2**	0	1	0	4	2
e_3	1	**1**	0	0	1	5	4
z	1	**2**	0	0	0	0	-2

On choisit maintenant le min des $K > 0$, ce qui nous permet d'obtenir la ligne pivot. Dans notre cas le minimum positif correspond à la première ligne, donc le vecteur e_1 est la colonne ou le vecteur sortant.(Les valeurs de la colonne sortantes seront mises en gras pour des raisons de lisibilité).

	a	**b**	e_1	e_2	e_3	C	K
e_1	**-3**	**2**	**1**	**0**	**0**	**2**	**1**
e_2	-1	**2**	0	1	0	4	2
e_3	1	**1**	0	0	1	5	4
z	1	**2**	0	0	0	0	-2

Le vecteur b va donc prendre la place du vecteur e_1. L'intersection de la colonne entrante et de la colonne sortante s'appelle le pivot. Dans cet exemple, le pivot est 2.
L'étape qui suit consiste à prendre la ligne pivot (ou la ligne sortante) et a la diviser par le pivot (Le but étant d'avoir 1 à la place du pivot). Cette manipulation donne le résultat suivant :

	a	b	e_1	e_2	e_3	C	K
b	$-3/2$	1	$1/2$	0	0	1	
e_2	-1	2	0	1	0	4	
e_3	1	1	0	0	1	5	
z	1	2	0	0	0	0	

L'étape d'après consiste à annuler les coefficients de la colonne entrante sauf le pivot. Pour cela, on va faire des combinaisons linéaires grâce à la ligne du pivot résultante. On obtient alors :

	a	b	e_1	e_2	e_3	C	K
b	$-3/2$	1	$1/2$	0	0	1	
e_2	2	0	-1	1	0	2	
e_3	$5/2$	0	$-1/2$	0	1	4	
z	4	0	-1	0	0	-2	

Le max de la ligne représentant la fonction économique est 4 (toujours positif). Par conséquent, il faut refaire les étapes précédentes (réitérer) jusqu'à ce que tous les coefficients deviennent négatifs. Le résultat de ces calculs va nous donner le résultat qui suit :
2 ème Transformation :

	a	b	e_1	e_2	e_3	C	K
b	0	1	$-1/4$	$3/4$	0	$5/2$	
e_2	1	0	$-1/2$	$1/2$	0	1	
e_3	0	0	$3/4$	$-5/4$	1	$3/2$	
z	0	0	1	-2	0	-6	

Cette solution n'est toujours pas optimale puisque le maximum des valeurs prises par la fonction économique est strictement positif.

3 ème Transformation :

	a	b	e_1	e_2	e_3	C	K
b	0	1	0	$1/3$	$1/3$	**3**	
a	1	0	0	$-1/3$	$-2/3$	**2**	
e_1	0	0	1	$-5/3$	$4/3$	2	
z	0	0	0	$-1/3$	$-4/3$	**-8**	

Cette fois, toutes les valeurs prises par la fonction économique sont négatives ou nulles, cette configuration est donc la plus optimale. Les valeurs de a et b sont tout simplement les composantes de C respectivement par rapport à a et b. (Elles sont marquées en gras dans notre exemple). Et puis la valeur prise par la fonction économique (qui est la valeur maximale) est l'opposé de la composante de C par rapport au vecteur z.

Le problème d'optimisation (maximisation dans ce cas) est alors résolu. La fonction économique prend la valeur maximale en a=2 et b=3 et a comme valeur 8.

2.2 Implémentation en Matlab :

L'implémentation de l'algorithme en Matlab s'est déroulé en suivant les étapes énoncées précédemment. En partant des cas les plus généraux vers des cas de plus en plus particuliers nécessitant des traitements particuliers. tel que l'ajout des variables d'écart, des variables artificielles, la vérification de la positivité des coefficients de la colonne K ... . Ces traitements rendent l'algorithme plus puissant et plus général puisqu'il peut détecter des cas non bornés, affirmer l'existence ou pas de solution optimale ...

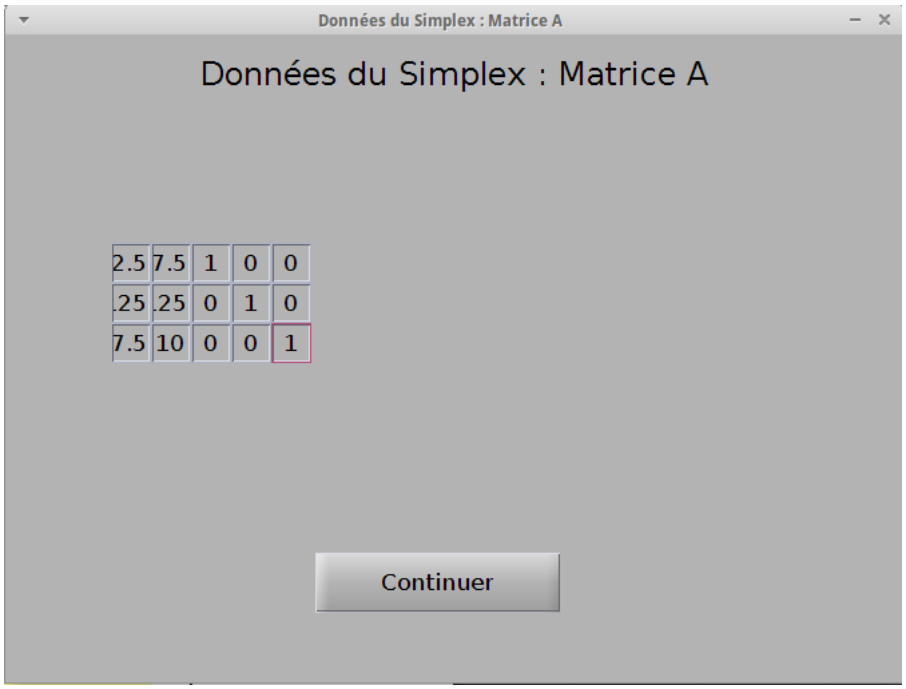
Les "classes" qui composent l'algorithme sont alors :

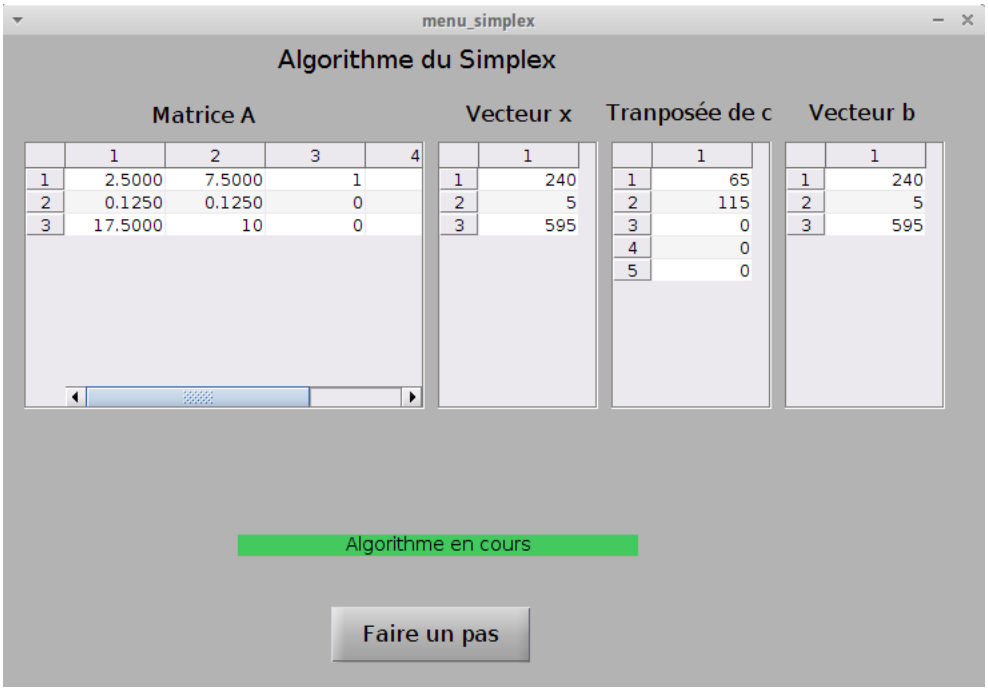
- Choix_entrante.m : Permet de choisir la variable Hors base dont le coût marginal est positif et le plus grand possible.
- Choix_sortante.m : La variable sortante est la ligne correspondant au minimum des coefficients relatifs à la division terme à terme de b_i/a_i. (Il faut faire attention puisqu'on cherche le minimum sur les coefficients positifs).
- determiner_nouvelle_base.m : Permet de déterminer la nouvelle base : (= à l'ancienne base à laquelle on retire le vecteur sortant et on ajoute le vecteur entrant).
- menu_principal.m : Permet de définir l'interface graphique. Modèle adopté : (MVC : Modèle, vue, contrôleur).
- menu_simplexe.m : La classe exécutée et qui fait appel à l'interface graphique afin de tester notre algorithme.
- pas_simulation : Permet de réaliser un pas dans la simulation. On fait alors le choix de la variable entrante, de la variable sortante et puis on détermine la nouvelle base.

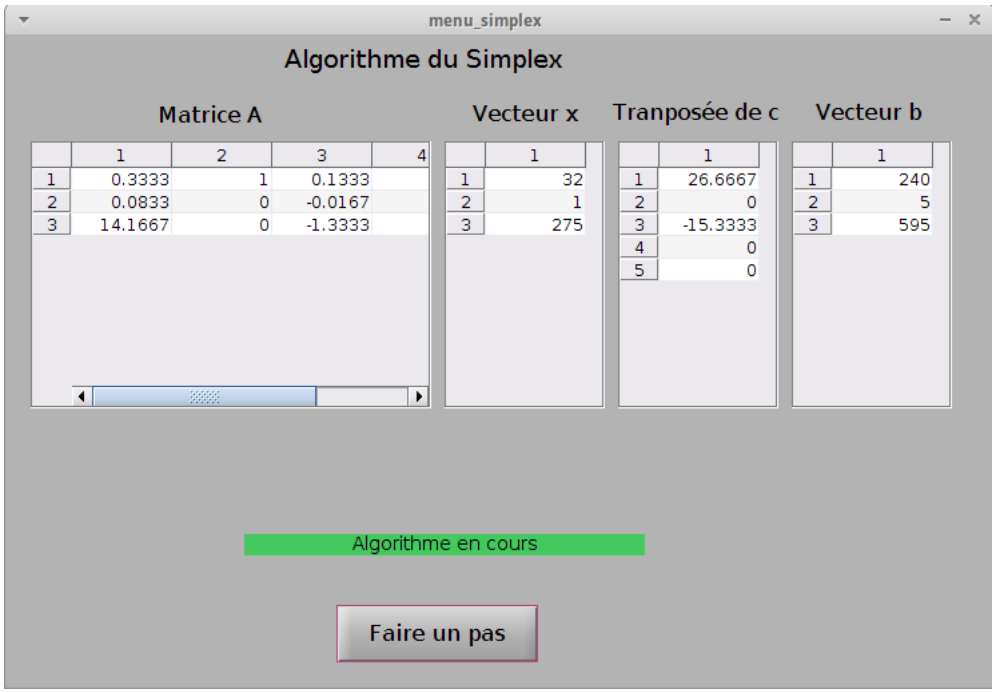
2.3 Tester notre algorithme

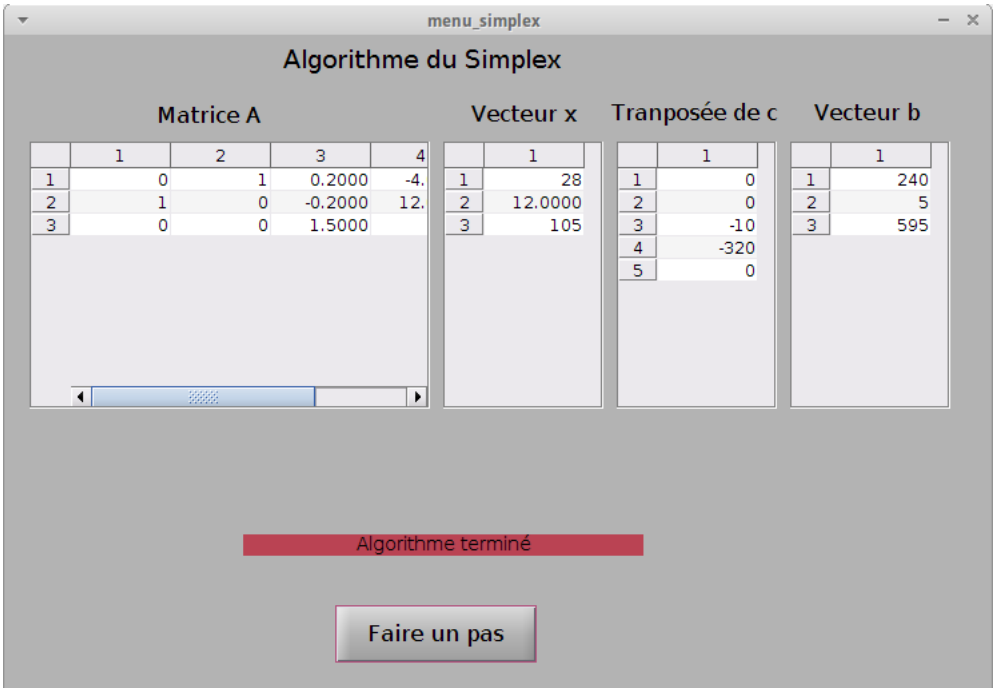
2.3.1 Validation du problème du brasseur

On commence d'abord par rentrer les données du problème : Les coefficients de la matrice A, du vecteur b ainsi que de la fonction économique à maximiser, puis on passe à l'algorithme qui fait des transformations à chaque pas de simulation jusqu'à trouver les solutions optimales.






menu_simplex
Algorithme du Simplex
Matrice A
Vecteur x
Tranposée de c
Vecteur b
1 2 3 4
1 0.3333 1 0.1333
2 0.0833 0 -0.0167
3 14.1667 0 -1.3333
1
1 32
2 1
3 275
1
1 26.6667
2 0
3 -15.3333
4 0
5 0
1
1 240
2 5
3 595
Algorithme en cours
Faire un pas


menu_simplex
Algorithme du Simplex
Matrice A
Vecteur x
Tranposée de c
Vecteur b
1 2 3 4
1 0 1 0.2000 -4.
2 1 0 -0.2000 12.
3 0 0 1.5000
1
1 28
2 12.0000
3 105
1
1 0
2 0
3 -10
4 -320
5 0
1
1 240
2 5
3 595
Algorithme terminé
Faire un pas

3 L'interface graphique

Une des difficultés de notre projet fut la gestion du temps, en particulier à cause de l'intégration d'une interface graphique correcte au projet. En effet, nous avons dû apprendre à utiliser les interfaces Matlab et les conventions de codage sont parfois assez peu conventionnelles. Elle n'a malheureusement été réalisée, par manque de temps, que pour le Simplexe, et l'écran de sélection du type d'égalités / inégalités n'est pas fonctionnel puisque le code ne gère que les égalités.

Pour résumer, nous avons utilisé l'IHM GUIDE qui permet à l'utilisateur Matlab de dessiner ses propres interfaces graphiques. Cependant, cette interface est clairement limitée. En effet, il est parfaitement impossible, par exemple, de lui indiquer qu'il faut créer dynamiquement $n * m$ cases pour remplir une matrice à l'écran et ensuite de les synchroniser directement dans une variable du workspace.

Tout cela doit se faire directement dans le code de l'interface graphique, code qui est, de par sa nature de code généré, relativement mal organisé et peu lisible. Nous avons donc utilisé à outrance via le code la propriété Callback afin de mettre à jour avec les fonctions assignin et evalin, utilisées dans l'environnement "base" afin de communiquer directement avec les scripts que nous avions créé.

La deuxième étape était de la rendre la plus ergonomique possible, par l'intégration de couleurs et en essayant de ne mettre, par fenêtre, que quelques informations nécessaires, pour ne pas perturber l'utilisateur du programme. Cela permet aussi d'afficher de manière très claire quand l'algorithme est terminé ou non, et le changement de cet état est extrêmement visuel et évident.

4 La méthode de Gabasov ou la méthode adaptée

4.1 Explication de l'algorithme

Le principe de cet algorithme est de résoudre le problème de manière algébrique. On résoud à la fois le problème primal et le problème dual. A chaque itération de l'algorithme, on trouve une direction primale et un pas primal permettant d'améliorer le critère du problème primal, puis on améliore le critère du dual en modifiant le support. On fait rentrer un nouveau vecteur dans la base, et on détermine ensuite le vecteur qui doit en sortir.
Nous avons donc découpé le problème en trois parties principales :

- L'initialisation du problème, dans laquelle nous definissons les variables (certaines variables sont également définies dans le Main) et nous mettons en place le problème.
  Nous appellons une boucle qui lance des pas de l'algorithme jusqu'a ce que le critère d'arrêt β soit inferieur à ϵ.
  Cela consiste en fait simplement a effectuer :

```
1   while beta>eps;
2       pas_simulation;
3   end
```

- La recherche de la direction primale et du pas primal, en utilisant l'algorithme détaillé en classe.
- Le calcul du nouveau support (resolution du problème dual). On utilise pour le changement de support la règle du pas court, car elle nous a paru plus simple a mettre en oeuvre, en effet elle tient en quelques lignes :

```
1 j00 = find(Jsup == j0,1);
2 Jsup(j00) = jetoile;
3 Jn(find(Jn == jetoile,1)) = j0;
4 beta = beta + sigma0*alpha0;
```

  Les notations utilisées sont les même que dans le cours.
  Pour demarrer l'algorithme sur l'exemple du brasseur, il faut lancer le fichier "Main.m". Lorsque l'algorithme se termine, on precise si la solution est optimale ou sous-optimale en regardant si β est egal a zero ou pas.

4.2 Differences avec le simplexe

Il y a plusieurs differences entre les deux algorithmes :

- Le simplexe se base sur une vision géométrique du problème, alors que Gabasov a une approche purement algébrique.
- La méthode de Gabasov rajoute en revanche une hypothèse sur les solutions possibles : le domaine réalisable doit être borné. Cette hypothèse permet de majorer l'écart à la solution optimale, et donc donne un critère d'arrêt pour l'algorithme et la possibilité d'obtenir des solutions sous-optimales.
- La méthode du simplexe se contente de traiter le problème primal, alors que celle de Gabasov traite aussi le problème dual, ce qui lui permet dans certains cas d'être plus permormante et d'éviter une complexité exponentielle. On peut intuitivement comparer les alogorithmes a deux voyageurs en montagne : le simplexe parcourt la montagne en grimpant sur les sommets, alors que Gabasov peut "couper" le chemin (en empruntant des tunnels par exemple).
- L'efficacité de la méthode du simplexe dépend énormément du point initial choisi, ce qui supose une phase d'initialisation performante du problème, ce qui n'est pas nécessaire pour la méthode de Gabasov.

4.3 Tests et comparaison avec le simplexe

4.3.1 Test de l'algorithme sur le problème du brasseur

Pour tester l'algorithme de Gabasov sur le problème du brasseur, il suffit d'exécuter le fichier Main.m .Les calculs se font automatiquement et les résultats sont stockés dans les variables concernées.

A

3x5 double

	1	2	3	4	5
1	2.5000	7.5000	1	0	0
2	0.1250	0.1250	0	1	0
3	17.5000	10	0	0	1

```
>> gabasov

compteur =

     1


compteur =

     2
```

A gamma

3x5 double

	1	2	3	4	5
1	0	1	0.2000	-4	0
2	1	0	-0.2000	12	0
3	0	0	1.5000	-170	1

A gam

5x1 double

	1
1	12
2	28
3	0
4	0
5	105

4.3.2 Comparaison avec l'algorithme du simplexe

Les deux algorithmes, pour l'exemple du brasseur que nous avons utilisé, arrivent à un resultat similaire en un même nombre d'iteration. Les resultats sont quasi-instantanés dans les deux cas, et nous ne pouvons pas comparer équitablement les temps d'execution en utilisant des fonctions matlab car il est possible que l'interface graphique, qui a été implantée pour le simplexe et pas pour gabasov, fausse les résultat en ralentissant l'execution du simplexe.

Nous n'avons malheureusement pas eu le temps de comparer les algorithmes sur d'autres exemples.

5 Limites de nos algorithmes

Pour le Simplexe, le programme fonctionne ; nous n'avons par contre pas réussi à introduire les variables d'écart d'une manière automatique, donc la contrainte est de rentrer le problème sous la forme simplexe. Ce traitement de variables d'écart à été commencé dans l'algorithme Matlab "preparation.m" mais est cependant non inclus dans le code final faute de temps (le code de préparation est tout de même présent dans le rendu final).

Pour Gabasov, le programme fonctionne également, mais il n'a pas pu être inclus dans l'interface graphique.

6 Organisation du groupe

Dès la publication du projet et la constitution des groupes, nous avons décidé de nous réunir afin de nous organiser et de répartir les tâches à faire. À la fin de cette réunion, un chef du projet (Yann MOUGENEL) et un responsable qualité (Thibault BROSSARD) ont été désignés dont la mission est de surveiller la progression du projet, le respect des deadlines, la communication entre les membres du groupe, l'information du professeur encadrant de la progression du travail ainsi que le contrôle de la qualité du travail rendu.

Pour la communication entre les membres du groupe, nous avons décidé de créer un groupe privé sur Internet nous permettant un échange instantané dans le cas de difficultés rencontrées par un membre.
Nous avons fixé des crénaux de travail individuel et collectif :

- Les créneaux de travail individuel nous permettent de réaliser les tâches individuelles.
- Les créneaux de travail collectif nous permettent de nous réunir afin de discuter de la progression du projet et de réunir les tâches faites tout en veillant à ce que le travail rassemblé soit cohérent avec la vision globale souhaitée et avec les contraintes fixées par le cahier des charges fonctionnel.

7 Conclusion

Le projet était avant tout une bonne épreuve de travail de groupe qui nous a permis de mener une réflexion à la fois globale et partagée. A l'issue de celle ci, nous avons pu développer le sens d'initiative, de critique et prendre du recul sur les choix de conception. Partir de rien nous a aussi permis de mener une profonde réflexion, d'établir une architecture propre et non imposée et donc de mieux cerner le problème afin de répondre au mieux aux critères imposés et choisir les solutions adéquates.

La gestion du timing et la répartition des tâches étaient un grand apport personnel en terme d'outil et de stratégie d'organisation.

En ce qui concerne l'apport pédagogique, le projet a balayé à peu près toutes les notions vues en cours, c'était un bon exercice où l'on a étudié de manière plus approfondie les algorithmes tout en les implémentant en Matlab, ce qui nous a permis de découvrir et mettre en pratique ce langage riche et peu manipulé l'année précédente. Le projet nous a notamment permis de rejoindre les domaines techniques des mathématiques en informatiques, permettant à chacun de s'appliquer dans la partie de son choix tout en avançant et découvrant dans l'autre domaine.

*** **Fin** ***