

TP 1 Méthodes numériques

Compte rendu Par Emmanuel Collin

Partie 1

1.1 Point fixe de Picard

On programme dans un premier temps une fonction réalisant la méthode de descente. Elle permet, à l'aide d'une suite, de déterminer la valeur de x pour laquelle on obtient une différence entre deux termes (de la suite) consécutifs pratiquement nulle (c'est-à-dire à une valeur de tolérance près) synonyme d'une annulation de la fonction.

Pour cela, 5 paramètres sont nécessaires en entrée:

- La fonction `func` que l'on va tester
- La pente p . Il existe une valeur optimale de p . Plus la différence entre la valeur optimale de p et la valeur finalement choisie de p est petite, plus la résolution a de chance de fonctionner, mais prendra plus d'itérations. Il convient donc de choisir un p optimal pour ne pas dépasser le nombre d'itérations maximal souhaité (k_{\max}) et avoir le plus petit k possible.
- Le point de départ x_0 , qui correspond au premier terme de la suite
- L'erreur ou tolérance notée ϵ pour epsilon, qui correspond au maximum toléré entre la valeur à atteindre (ici 0) et la valeur du terme de la suite, à partir duquel on considérera que le point fixe a été atteint.
- k_{\max} qui comme dit précédemment correspond au maximum d'itérations souhaité.

En ce qui concerne les paramètres de sortie, ils sont au nombre de 3:

- la valeur de x , obtenue soit après les k_{\max} itérations quand il n'y a finalement pas convergence, soit après les k (voir ci-dessous) itérations lorsqu'il y a convergence. Sa valeur est égale à sa valeur précédente à laquelle on retire p fois l'image de la valeur précédente par la fonction `func`. x vaut initialement x_0 .
- le nombre k d'itérations finalement effectuées. Lorsqu'il n'y a pas convergence, celui ci vaut

kmax, sinon il est inférieur.

- l'erreur err, qui correspond à l'écart, obtenu à la dernière itération (la k-ième), entre la valeur souhaitée (ici 0), et la valeur de $f(x)$ après la k-ième itération, soit tout simplement $f(x)$

On réitère tant que l'erreur est supérieure a la tolerance eps et que le nombre d'itérations (k) est inférieur a celui en paramètre (kmax).

On prend une première fonction telle que:

$$\text{fonction}(x) = \frac{x}{2}$$

$$\text{Donc } p \cdot \text{fonction}(x) = p \times \frac{x}{2}$$

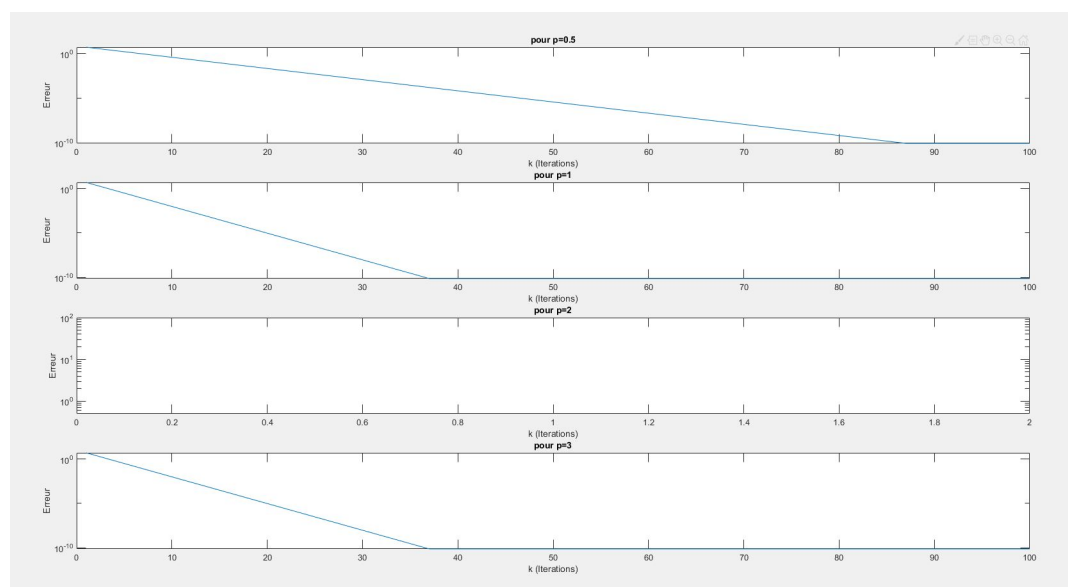
On cherche à avoir un résultat de la forme:

$$x + C$$

Avec C une constante réelle.

Ainsi, si $p = 2$, on a: $p \cdot \text{fonction}(x) = x$

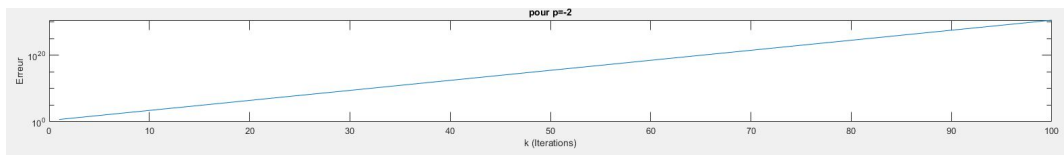
Pour diverses valeurs de p, $x_0 = 10$, et une tolérance de $1e-10$, on obtient les représentations suivantes, correspondant au log de la valeur absolue de l'erreur en fonction du nombre d'itérations:



On remarque que plus la valeur s'éloigne de 2, plus le nombre d'itérations nécessaire augmente. Pour une pente de 2, que l'on va noter p_{optimale} , seule la première erreur (voir ci-dessous) est non nulle (ce qui explique pourquoi il n'y a pas de tracé puisqu'on affiche ici le log de la valeur absolue de l'erreur):

verr							
1x100 double							
	1	2	3	4	5	6	7
1	5	0	0	0	0	0	
2							
3							

Quand p est trop éloigné de p_{optimale} , il n'y a plus convergence et dans certains cas, la valeur absolue de l'erreur devient de plus en plus élevée comme c'est le cas ci-dessous pour $p = -2$



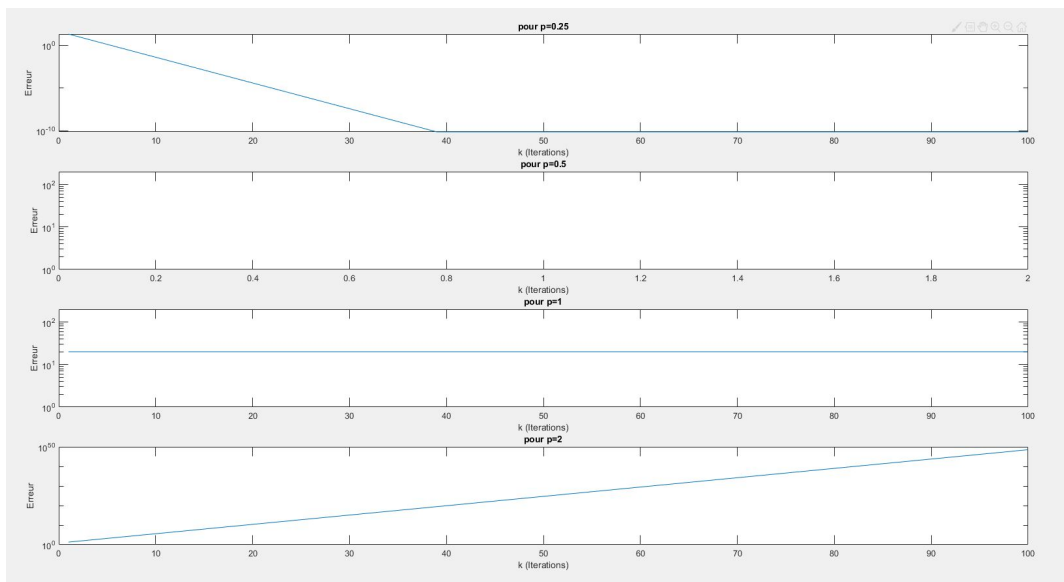
On procède de la même façon avec fonction2(x) = 2x

Si $p = 1/2$, on a $p^* \text{ fonction2}(x) = x$

Ainsi on a $p_{\text{optimale}} = 1/2$

On conserve x_0 et la tolérance.

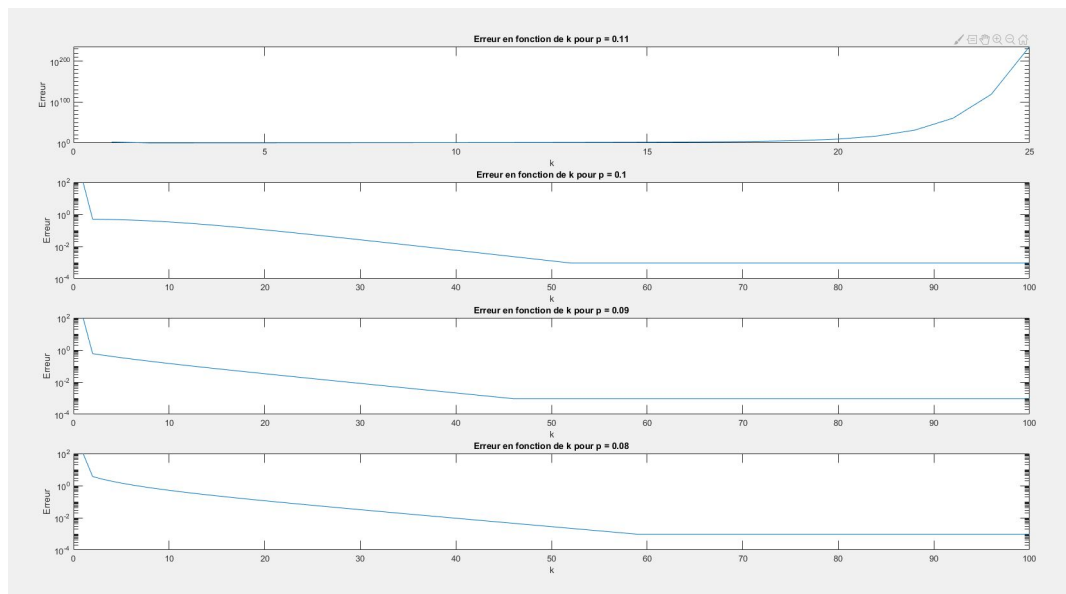
Le comportement en fonction de p (voir ci-dessous) est similaire, sauf que p_{optimale} vaut dans ce cas $1/2$ (ou 0.5):



Cette fois ci, la méthode n'est plus la même pour le calcul de la pente avec fonction3(x) = $x^*x - (1/2)$

On prend un x_0 de 10, une tolérance de 10^{-3} .

Ne sachant pas calculer p on essaye avec diverses des valeurs:

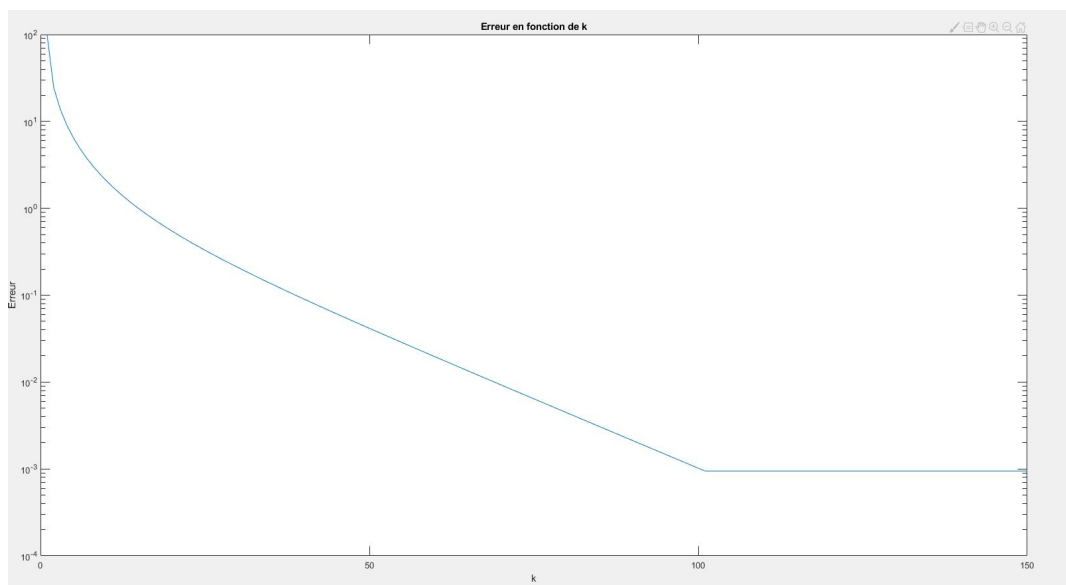


En continuant avec des tests plus poussés on obtient une valeur optimale de p avec de tels paramètres d'environ 0.0934.

1.2 Méthode de Newton

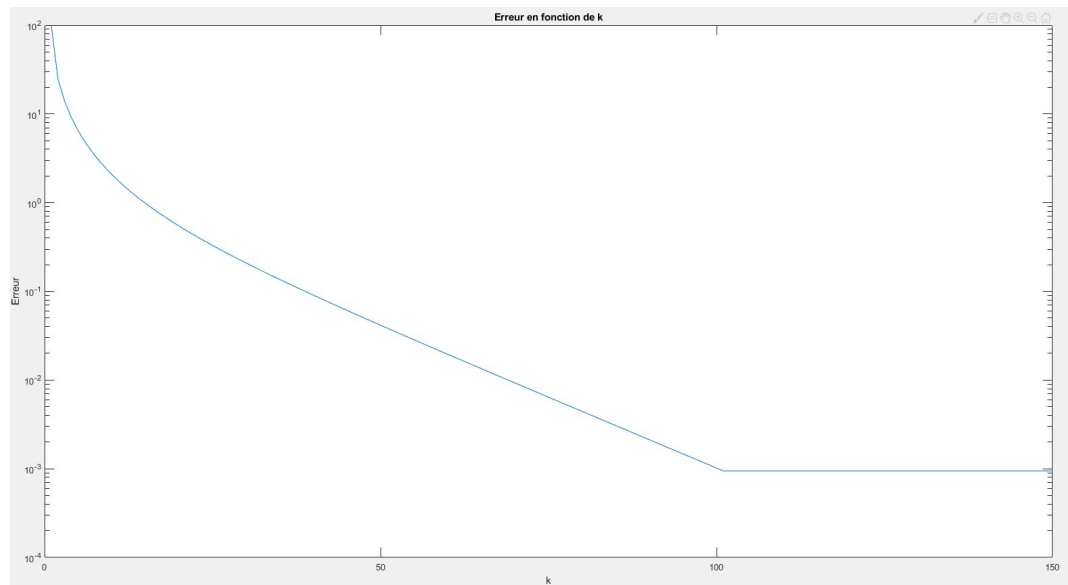
On teste cette fois-ci, toujours avec fonction $3(x) = x^*x - (1/2)$ et en paramètres $x_0 = 10$ et une tolérance de 10^{-3} , la méthode dite de Newton. Cette fois-ci, la pente vaut l'inverse de la dérivée de la fonction testée, on doit donc entrer à la place de la valeur de p , le nom de la fonction correspondant à la dérivée.

On obtient ceci:



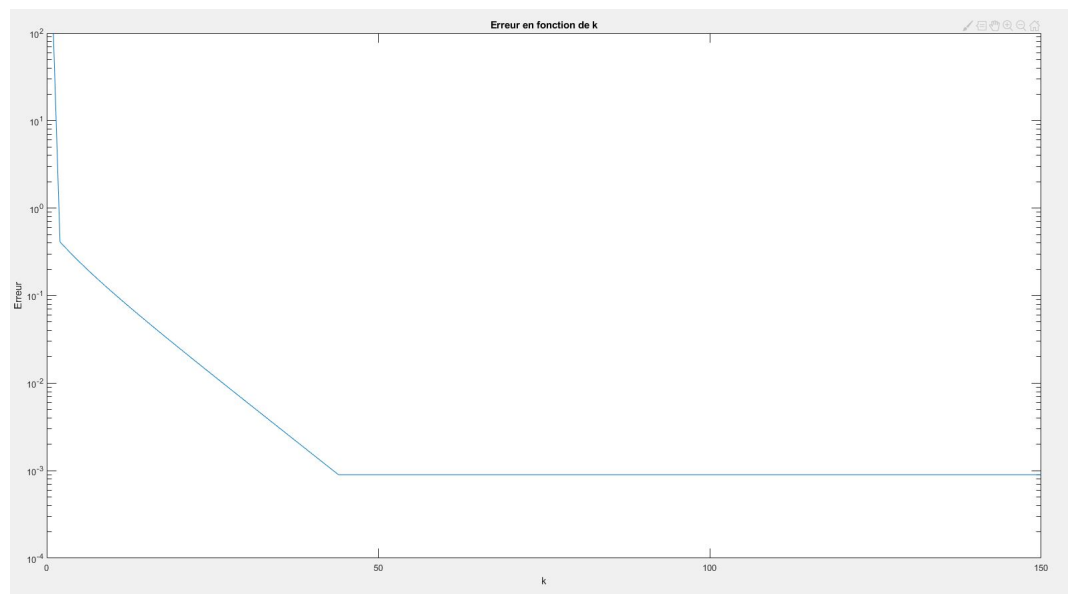
Cette fois-ci on teste la méthode des différences finies et à la différence de celle de Newton, la pente vaut l'inverse d'un coefficient, calculé à partir de deux points très proches et de la forme

$\frac{f(x+h)-f(x)}{h}$, d'où le nom de différences finies, en prenant h très petit. Avec les mêmes paramètres, on obtient ceci:

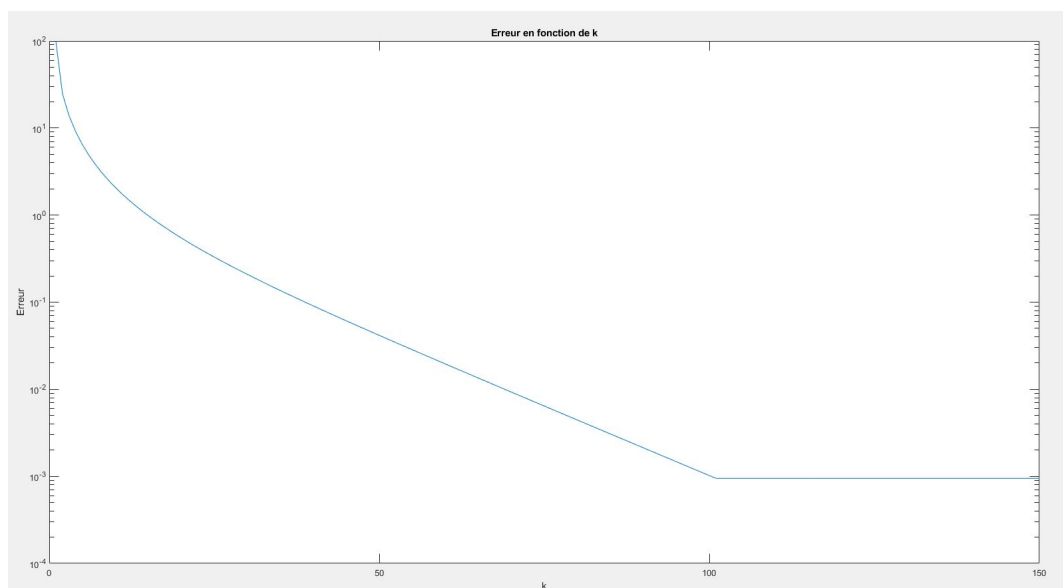


Enfin, on teste cette fois-ci la méthode de la sécante. On a maintenant une pente calculée à partir de deux points et de leur valeur associée. Elle vaut la différence entre les deux valeurs x_1 et x_2 que l'on divise par la différence de leurs images. A chaque itération, on garde le dernier x , le nouveau correspond à l'intersection entre l'axe des abscisses et de la droite passant par les images des $f(x_1)$ et $f(x_2)$ précédents.

On teste ici avec 1 et 10 comme x de départ soit un dx de 9:



Et maintenant $10 \cdot 10^{-8}$ et 10 soit un dx de 10^{-8} :



On remarque que pour cette fonction, il est préférable de prendre un x -dx davantage proche de la solution (environ 0.7077) pour avoir rapidement convergence.

Partie 2

A l'aide du principe fondamental de la dynamique et sachant que l'on approche les dérivées par des différences finies, si l'on part de l'expression de l'accélération, en intégrant successivement en y ajoutant les bonnes conditions initiales, on obtient le schéma suivant pour la position de la fusée:

$$x = x^{(0)} + \Delta t (u^{(0)} + \Delta t \frac{F(x^{(0)})}{m})$$

avec m la masse de la fusée, $F(x)$ la résultante des forces s'exerçant sur la fusée, $u^{(0)}$ la vitesse initiale, $x^{(0)}$ la position initiale, Δt le pas de temps et x la position (qui peut être un scalaire ou un vecteur)

On va considérer durant cette application que seule la force d'attraction entre la fusée et la Terre s'exerce sur la fusée.

$$\text{Ainsi } F(x) = -\frac{GMm}{||x||^3}x$$

Avec M la masse de la fusée et G la constante de gravitation telles que

$$G = 6.7e-11$$

$$M = 6e24$$

le tout dans les unités du système international

Donc, on peut remarquer dès lors que l'on a désormais une simplification si l'on injecte $F(x=x^{(0)})$

dans l'équation permettant de calculer x . En effet, la masse de la fusée n'est plus à prendre en compte:

$$x = x^{(0)} + \Delta t (u^{(0)} + \Delta t \frac{-GM}{\|x^{(0)}\|^3} x^{(0)})$$

On crée ensuite deux fonctions, chacune déclinée en deux versions.

La première est $\text{Evol}(x_0, v_0, dt)$, qui renvoie la position après dt secondes en ayant un point de départ situé en x_0 et étant à la vitesse initiale de v_0 . Cette fonction fonctionne qu'en une dimension. La déclinaison Evol3D fonctionne en 3 dimensions. Ces deux déclinaisons sont basées sur la même formule ci-dessus pour calculer les coordonnées de la fusée après un certain temps et une position et vitesse initiales données.

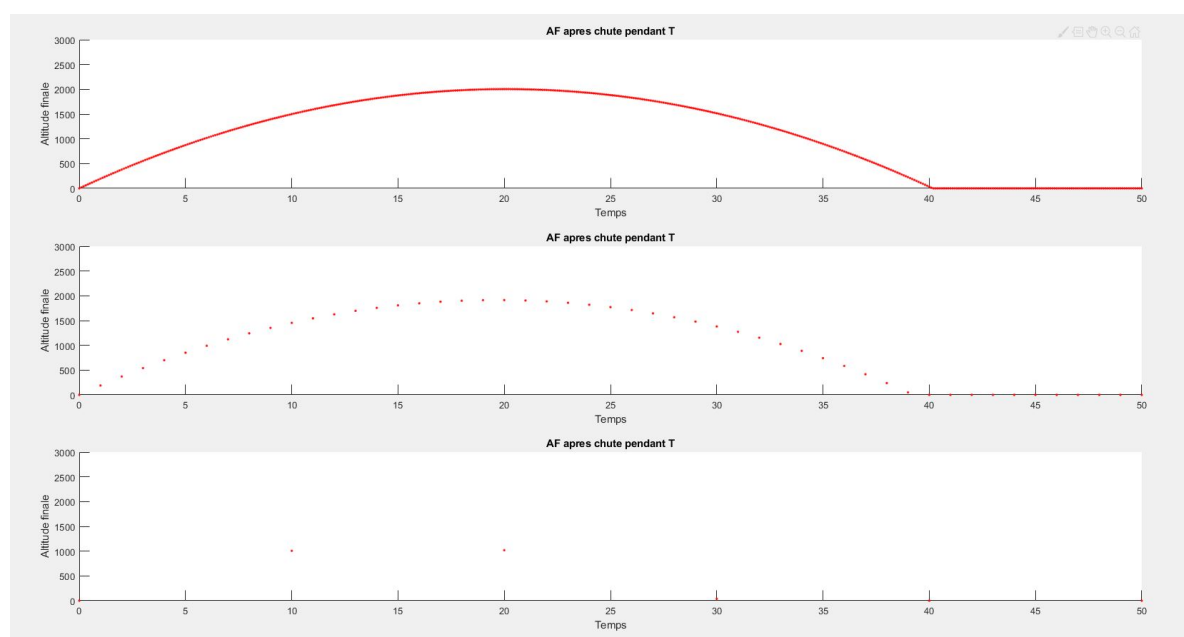
On crée ensuite la fonction $\text{TimeLoop}(x_0, v_0, T, dt)$ qui prend les mêmes paramètres en entrée que la fonction Evol , avec un paramètre supplémentaire T . Cette fonction renvoie la position de la fusée initialement en x_0 , à la vitesse v_0 , tous les dt pendant une durée totale de T .

On réalise ensuite divers tests, par exemple l'influence de dt .

On place la fusée à une altitude de $6360e3$ m, soit à la surface de la Terre. Sa vitesse initiale est de 200 m.s^{-1} .

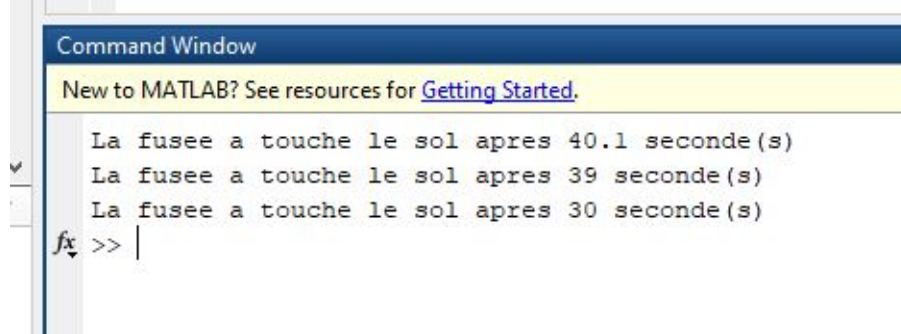
L'observation de l'altitude (par rapport à la surface de la Terre) par rapport au temps se fait sur 50 secondes. Trois intervalles d'observations ont été choisis: $1e-1$ s, 1 s et 10 s.

On obtient le résultat suivant:-



Dans le premier cas comme dans le suivant, il est possible d'estimer avec précision l'altitude après 15 secondes, alors que dans le dernier cas, cela n'est pas possible.

De plus, les temps indiqués ne sont pas les mêmes alors que seul dt change:



```

Command Window
New to MATLAB? See resources for Getting Started.

La fusée a touché le sol après 40.1 seconde(s)
La fusée a touché le sol après 39 seconde(s)
La fusée a touché le sol après 30 seconde(s)
fx >>

```

Le premier cas, avec le dt le plus petit, et un résultat de 40.1 secondes, est le plus précis des 3 et se rapproche le plus de la réalité.

Ainsi, plus dt sera petit, plus le résultat sera précis.

Cependant, en diminuant dt , la machine doit faire davantage de calculs et il faudra plus de puissance sous peine de devoir attendre très longtemps avant d'avoir un résultat.

On peut aussi calculer le temps nécessaire pour que la fusée, lâchée de 1000 mètres sans vitesse initiale, touche le sol.

On tape `TimeLoop(6360e3+1e3,0,15,1e-2)`

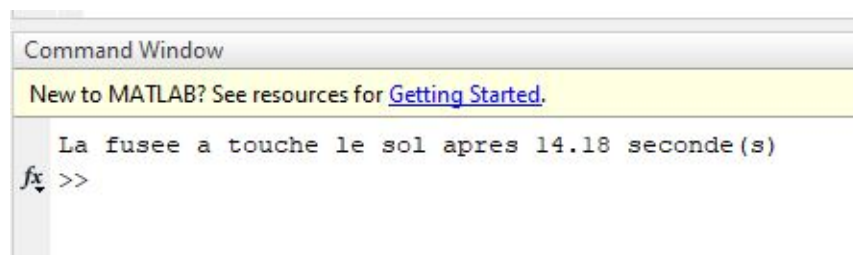
6360e3+1e3 étant le rayon de la Terre + l'altitude en mètre

0 correspond à la vitesse initiale au moment du lâcher

15 la durée d'observation en secondes

1e-2 les intervalles de temps auxquels sont calculées les altitudes

On trouve un temps de 14.18 secondes.



```

Command Window
New to MATLAB? See resources for Getting Started.

La fusée a touché le sol après 14.18 seconde(s)
fx >>

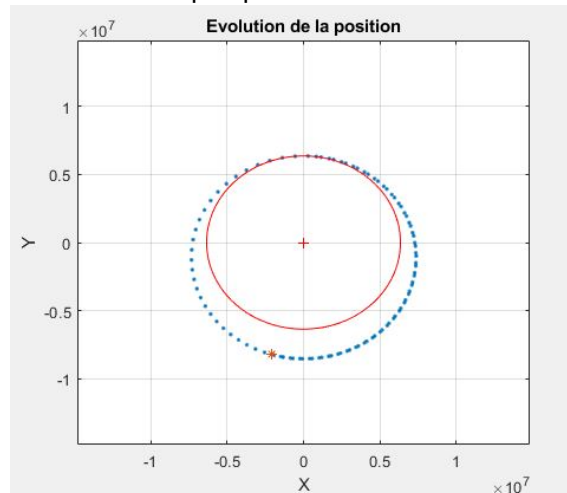
```

On peut également tester avec une vitesse initiale et en 3 dimensions avec la commande suivante:
`TimeLoop3d(0,6360e3,0,8500,500,0,500000,100)`

La fusée se trouve sur la surface de la Terre et a une vitesse initiale selon l'axe des x de 8500 m.s^{-1} , et de 500 m.s^{-1} selon l'axe des y . La mesure est effectuée sur 500000 secondes et les intervalles sont de 100 secondes (le tout est bien évidemment simulé en accéléré).

On remarque que la fusée parvient à entrer en rotation autour de la Terre sans la heurter. Cepen-

dant on remarque que l'altitude de la fusée atteint 0 m d'altitude, au niveau du point de départ.



Cela s'explique par le fait que la fusée n'est soumise uniquement à l'attraction de la Terre et aucune autre force. En effet, en réalité de nombreuses autres forces sont présentes, telles que celle générée par les réacteurs et pourraient permettre à une fusée de se mettre en orbite. Dans notre cas, 3 situations sont possibles.

Situation 1

La vitesse initiale est très grande et la fusée part à l'infini sans jamais retourner sur Terre.

On rappelle la formule permettant de calculer la position:

$$x = x^{(0)} + \Delta t (u^{(0)} + \Delta t \frac{-GM}{\|x^{(0)}\|^3} x^{(0)})$$

Dans cette situation $u^{(0)}$ prend et garde tout le temps le dessus par rapport à $\Delta t \frac{-GM}{\|x^{(0)}\|^3} x^{(0)}$, et ainsi la fusée continue sa progression sans jamais retourner vers la Terre.

Autrement dit, il faut que $u^{(0)} + \Delta t \frac{-GM}{\|x^{(0)}\|^3} x^{(0)}$ reste à tout instant positif.

$$\text{Soit } u^{(0)} > \Delta t \frac{GM}{\|x^{(0)}\|^3} x^{(0)}$$

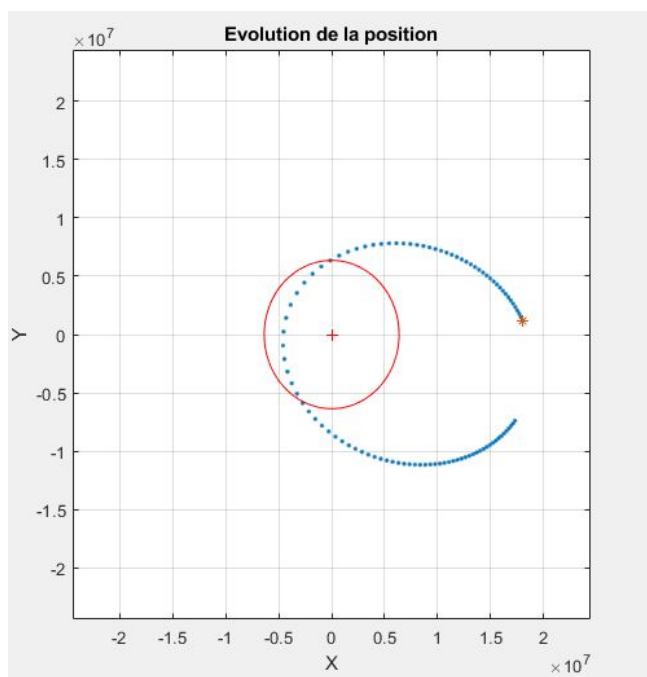
Situation 2

Cette situation est celle dans l'exemple cité précédemment illustré, où la fusée fait le tour de la Terre avant de rejoindre son point de départ sans toutefois s'écraser, et ce à l'infini.

Situation 3

Cette dernière situation est celle où la fusée s'écrase tout simplement sur terre à cause d'un mauvais angle de lancer et/ou une vitesse de lancer insuffisante. On remarque que s'il était possible de passer à travers la Terre, la fusée rejoindrait son point de lancer et effectuerait le même parcours à l'infini.

Avec la commande `TimeLoop3d(0,6360e3,0,8500,5000,0,500000,100)` (Le code a depuis été mis à jour et la fusée s'immobilise quand elle entre en collision avec la Terre) on obtient ceci:



Ainsi, en ne prenant en compte que l'attraction de la Terre sur la fusée et aucune autre force, la fusée ne peut pas réellement se mettre en orbite, sauf si l'on considère que comme dans la situation 2, où la fusée frôle la Terre sans s'écraser en revenant à chaque tour à son point de départ soit à 0 m d'altitude.

Avec les nombreux essais, on remarque que l'on parvient à cette situation lorsque l'angle de lancement est assez faible, environ 2° à 3° par rapport au sol.