Les graphiques (deuxième partie)

Dans cette deuxième partie, nous allons voir comment créer des graphiques multiples et des vecteurs.

Pour plus d'informations, se référer au site : https://matplotlib.org/tutorials/index.html

Afficher plusieurs graphiques simultanément

Pour cela, on utilise la méthode **plt.subplot(nombre de lignes, nombre de colonnes, index)** de la bibliothèque matplotlib.pyplot as plt.

Cette méthode permet de créer un emplacement en divisant la fenêtre graphique en un nombre donné de lignes et de colonnes indiqué entre parenthèses. L'index permet de renseigner la position de l'emplacement dans la fenêtre graphique. Il commence à 1 dans le coin supérieur gauche de la fenêtre graphique et augmente vers la droite puis vers le bas.

Exemple: pour une fenêtre graphique divisée en 2 lignes et 2 colonnes, les emplacements sont indexées comme suit :

1 2

3 4

Après avoir défini l'emplacement, il suffit de coder directement les caractéristiques du graphique qui apparaîtra dans l'emplacement en question.

Pour ajuster les positions des côtés des graphiques et les espacements entre ceux-ci, on utilise la méthode :

plt.gcf().subplots_adjust(left = 0.125, bottom = 0.1, right = 0.9, top = 0.9, wspace = 0.2, hspace = 0.2) de la bibliothèque matplotlib.pyplot as plt.

Les paramètres utilisés sont les suivants (les valeurs données ci-dessous sont les valeurs par défaut) :

- left = 0.125 : position du côté gauche du graphique
- right = 0.9 : position du côté droit du graphique
- bottom = 0.1 : position du côté bas du graphique
- top = 0.9 : position côté haut du graphique
- wspace = 0.2 : espacement horizontal entre deux graphiques
- hspace = 0.2 : espacement vertical entre deux graphiques

Toutefois, si la taille de la fenêtre graphique n'est pas cohérente avec les valeurs des paramètres, l'affichage sera modifié.

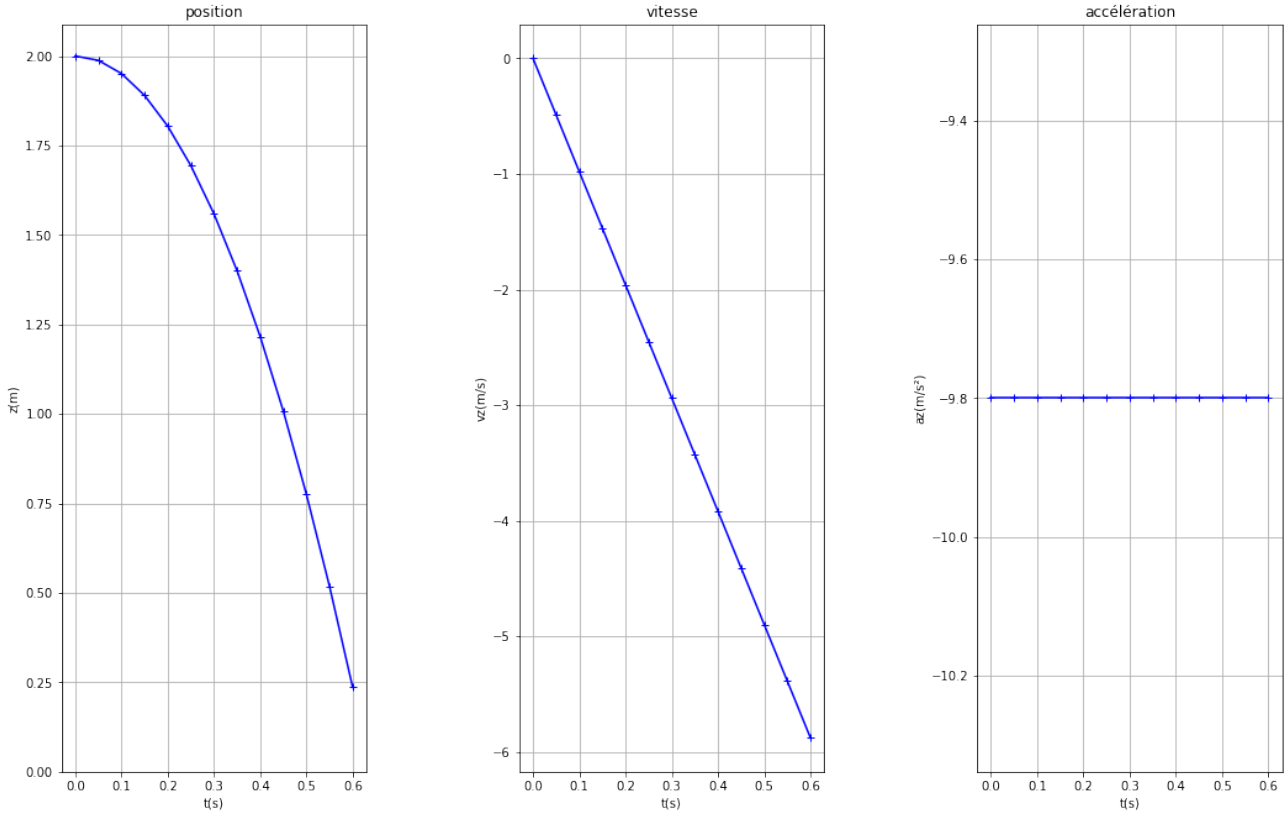
Prenons l'exemple de la chute libre d'une balle sans vitesse initiale afin d'illustrer ces différentes méthodes. Voici trois programmes permettant d'afficher les coordonnées verticales de la position, de la vitesse et de l'accélération d'une balle en fonction du temps sur trois graphiques simultanément. Selon le programme, les graphiques sont positionnés différemment.

Premier programme

In [1]:

```
import matplotlib.pyplot as plt
%matplotlib inline
import numpy as np
```

```
t=np.arange(0,0.65,0.05)
z=-4.9*t**2+2
vz=-9.8*t
az=-9.8+0*t

# création de la fenêtre graphique 1

plt.figure(1,figsize=(10,12))
plt.gcf().subplots_adjust(left = 0.125, bottom = 0.2, right = 1.5,
                          top = 0.9, wspace = 0.5, hspace = 0)

# division de la fenêtre graphique en 1 ligne, 3 colonnes,
# graphique en position 1
# puis caractéristiques de ce graphique

plt.subplot(1,3,1)
plt.plot(t,z,color='b', marker = '+')
plt.title('position')
plt.grid()
plt.xlabel('t(s)')
plt.ylabel('z(m)')
plt.ylim(0)


# division de la fenêtre graphique en 1 ligne, 3 colonnes,
# graphique en position 2
# puis caractéristiques de ce graphique

plt.subplot(1,3,2)
plt.plot(t,vz,color='b', marker = '+')
plt.title ('vitesse')
plt.grid()
plt.xlabel('t(s)')
plt.ylabel('vz(m/s)')


# division de la fenêtre graphique en 1 ligne, 3 colonnes,
# graphique en position 3
# puis caractéristiques de ce graphique

plt.subplot(1,3,3)
plt.plot(t,az,color='b', marker = '+')
plt.title('accélération')
plt.grid()
plt.xlabel('t(s)')
plt.ylabel('az(m/sš)')

plt.show()
```

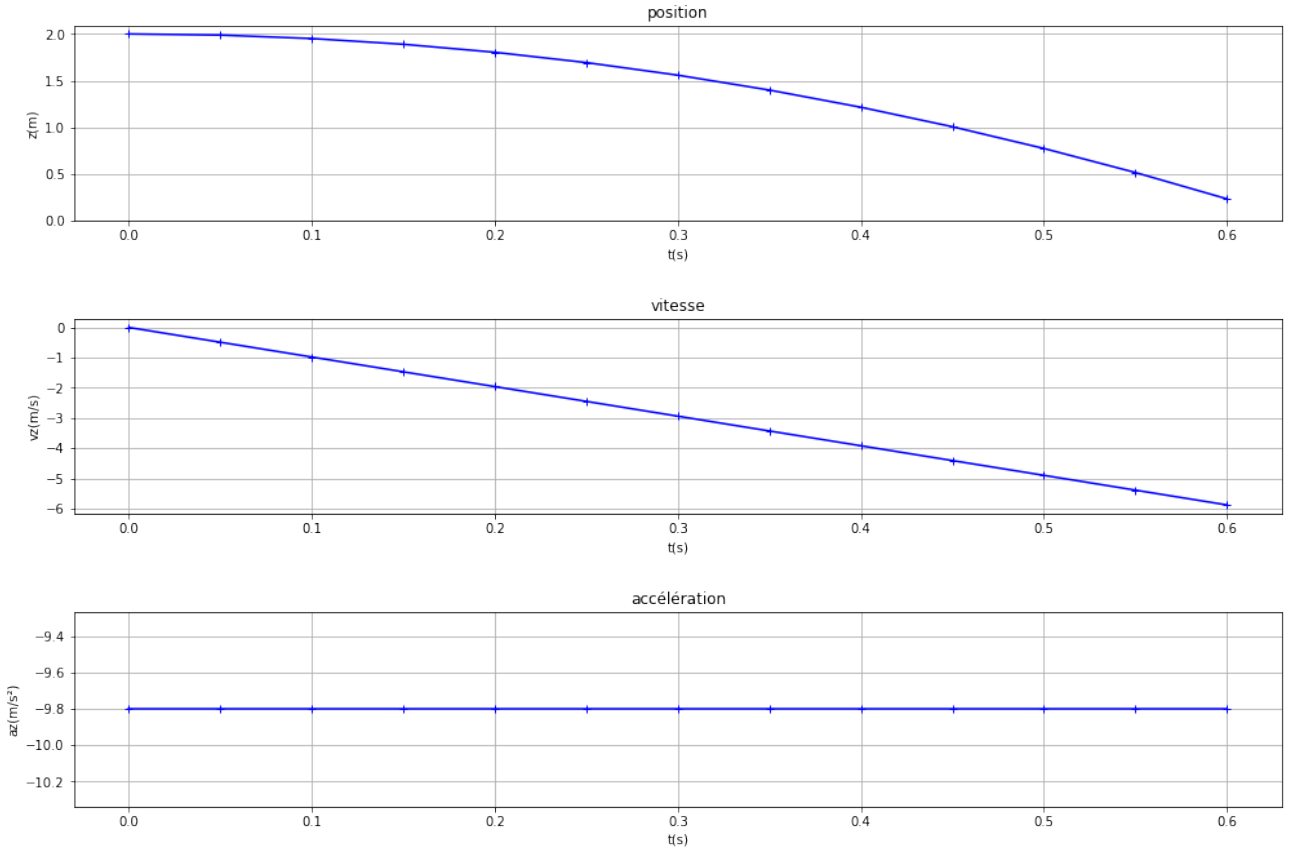


Deuxième programme

In [2]:

```
# création de la fenêtre graphique 2

plt.figure(2,figsize=(10,12))
plt.gcf().subplots_adjust(left = 0.2, bottom = 0.2, right = 1.5,
                          top = 0.9, wspace = 0, hspace = 0.5)

# division de la fenêtre graphique en 3 lignes, 1 colonne,
# graphique en position 1
# puis caractéristiques de ce graphique

plt.subplot(3,1,1)
plt.plot(t,z,color='b', marker = '+')
plt.title('position')
plt.grid()
plt.xlabel('t(s)')
plt.ylabel('z(m)')
plt.ylim(0)


# division de la fenêtre graphique en 3 lignes, 1 colonne,
# graphique en position 2
```

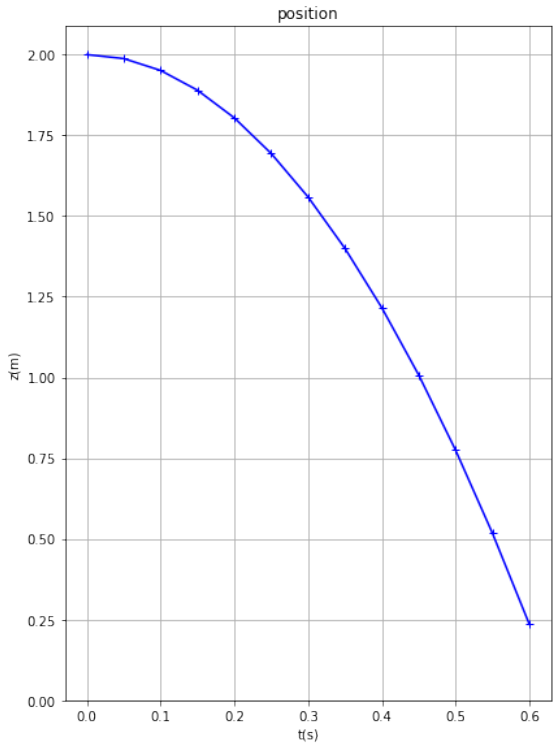
```
# puis caractéristiques de ce graphique

plt.subplot(3,1,2)
plt.plot(t,vz,color='b', marker = '+')
plt.title ('vitesse')
plt.grid()
plt.xlabel('t(s)')
plt.ylabel('vz(m/s)')


# division de la fenêtre graphique en 3 lignes, 1 colonne,
# graphique en position 3
# puis caractéristiques de ce graphique

plt.subplot(3,1,3)
plt.plot(t,az,color='b', marker = '+')
plt.title('accélération')
plt.grid()
plt.xlabel('t(s)')
plt.ylabel('az(m/sš)')

plt.show()
```

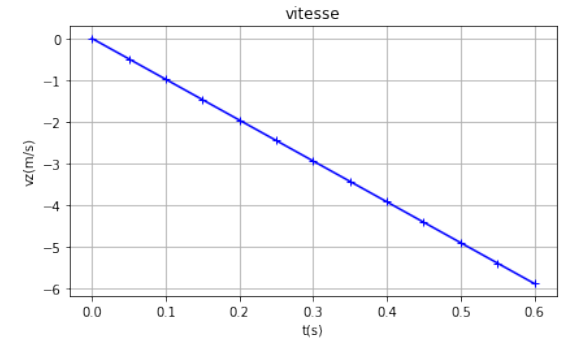


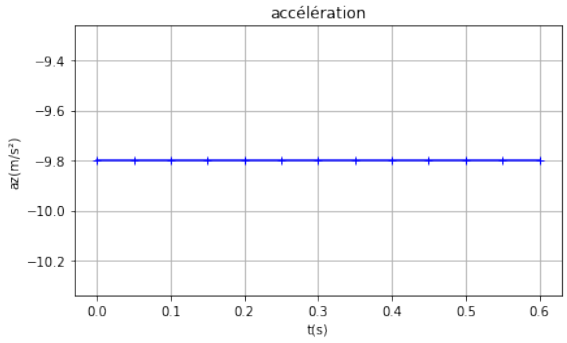
Troisième programme

In [3]:

```
# création de la fenêtre graphique 3

plt.figure(3,figsize=(10,12))
plt.gcf().subplots_adjust(left = 0.2, bottom = 0.2, right = 1.5,
                          top = 0.8, wspace = 0.5, hspace = 0.5)

# division de la fenêtre graphique en 1 ligne, 2 colonnes,
# graphique en position 1
# puis caractéristiques de ce graphique

plt.subplot(1,2,1)
plt.plot(t,z,color='b', marker = '+')
plt.title('position')
plt.grid()
plt.xlabel('t(s)')
plt.ylabel('z(m)')
plt.ylim(0)


# division de la fenêtre graphique en 2 lignes, 2 colonnes,
# graphique en position 2
# puis caractéristiques de ce graphique

plt.subplot(2,2,2)
plt.plot(t,vz,color='b', marker = '+')
plt.title ('vitesse')
plt.grid()
plt.xlabel('t(s)')
plt.ylabel('vz(m/s)')


# division de la fenêtre graphique en 2 lignes, 2 colonnes,
# graphique en position 4
# puis caractéristiques de ce graphique

plt.subplot(2,2,4)
plt.plot(t,az,color='b', marker = '+')
plt.title('accélération')
plt.grid()
plt.xlabel('t(s)')
plt.ylabel('az(m/sš)')

plt.show()
```







Afficher des vecteurs sur un graphique

Pour cela, on utilise la méthode de la bibliothèque matplotlib.pyplot as plt

plt.arrow(x, y, composante_selon_x, composante_selon_y, facecolor='r', edgecolor='r', width=0.008, head_width=0.02, length_includes_head=True)

Les paramètres utilisés sont :

- x, y : coordonnées de l'origine du vecteur dans la base utilisée dans le graphique.
- composante_selon_x, composante_selon_y : composantes (coordonnées) du vecteur
- facecolor (fc) : couleur de remplissage du vecteur
- edgecolor (ec) : couleur du contour du vecteur
- width : largeur du corps du vecteur (valeur par défaut: 0.001)
- head_width : largeur de la tête du vecteur (par défaut: 3 * width)
- head_length : longueur de la tête du vecteur (par défaut: 1.5 * head_width)
- length_includes_head : True si la tête doit être comptée dans le calcul de la longueur du vecteur (par défaut: False)

Prenons l'exemple d'un tir parabolique d'une balle et voyons comment tracer les vecteurs vitesse pour chaque position de la balle.

In [4]:

```
import matplotlib.pyplot as plt
%matplotlib inline
```

```
import numpy as np

t=np.arange(0,1.05,0.05)

z=-4.9*t**2+5*t
vz=-9.8*t+5
x=3*t
vx=3+0*t


plt.figure(4,figsize=(17,7))

plt.plot(x,z,color='b', marker = '+')
plt.title('Tracé des vecteurs vitesse')
plt.grid()
plt.xlabel('x(m)')
plt.ylabel('z(m)')

for i in range (len (z)) :
    plt.arrow(x[i],z[i],0.02*vx[i],0.02*vz[i],facecolor='r',
              edgecolor='r',width=0.008,head_width=0.02,
              length_includes_head=True)

# ici les coordonnées vx et vy sont multipliées par 0.02 afin
# d'adapter l'échelle de longueur du vecteur.

plt.show()
```