# Décision dans l'incertain

### La théorie du portefeuille de Markowitz (janv 2024)

# 1. Introduction

On considere $d$ actifs dont les rendements sont donnés par $(R_1, \dots, R_d)$. L'hypothèse de rendement signifie que si on détient à l'instant $0$ une quantité d'actif $i$ de valeur $V$, la valeur de cette même quantité d'actif à l'instant $T$ (égal à $T = 1$ an par exemple) sera donnée par $V(1 + R_i)$.

On suppose de plus que ces rendements ont des caractéristiques de moyenne et de variance connue. On note $\mu$ le vecteur des espérances $\mu_i = \mathbf{E}(R_i)$ et $\Gamma$ la matrice de variance covariance, où $\Gamma_{ij} = Cov(R_i, R_j)$. On note $\sigma_i^2 = \mathrm{V}ar(R_i) = \Gamma_{ii}$.

Pour construire le modèle, on suppose que la matrice de corrélation $C$, i.e. la matrice définie par

$$C_{ij} = \frac{Cov(R_i, R_j)}{\sqrt{\mathrm{V}ar(R_i)}\sqrt{\mathrm{V}ar(R_j)}},$$

est de la forme

$$C = \begin{pmatrix} 1 & \rho & \rho & \dots & \rho \\ \rho & 1 & \rho & \dots & \rho \\ \rho & \rho & 1 & \dots & \rho \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ \rho & \rho & \dots & \rho & 1 \end{pmatrix}.$$

On fixe la constante $\rho$ et le vecteur $\sigma$ des écart-types des rendements défini pour tout $i$ par

$$\sigma_i = \sqrt{\mathrm{V}ar(R_i)}.$$

Avec la matrice $C$, le vecteur $\sigma$ (vecteur colonne) et sa transposée $\sigma'$ on peut construire une matrice de variance-covariance $\Gamma$ pour le vecteur $R$ en posant (exercice: il faut vérifier que $\Gamma$ est bien une matrice symétrique et définie positive):

$$\Gamma_{ij} = \sigma_i C_{ij} \sigma_j.$$

## 1.1. Le cas à deux actifs risqués

On suppose que $d = 2$, que $\mu_1 = 5\%$ et $\mu_2 = 15\%$, que $\sigma_1 = 10\%$ et $\sigma_2 = 30\%$ et $\rho$ étant un paramètre réel, $\Gamma$ est donnée par

$$\Gamma = \begin{pmatrix} \sigma_1^2 & \rho\sigma_1\sigma_2 \\ \rho\sigma_1\sigma_2 & \sigma_2^2 \end{pmatrix}.$$

---

Question 1:
Que représente $\rho$ ? A quelle condition sur $\rho$ la matrice $\Gamma$ est la matrice de covariance d'un vecteur aléatoire ?

---

Dans la suite, on prendra $\rho = 0$.

On constitue un portefeuille de valeur initiale $X_0 = 1$ constitué d'une quantité $x_1$ d'actif $1$ et $x_2$ d'actif $2$ avec $x_1 \geq 0$, $x_2 \geq 0$ et $X_0 = x_1 + x_2 = 1$ (i.e. on répartit 1E entre le deux actifs risqués). On note $X_T$ la valeur de ce portefeuille en $T$

Vérifier que si le gain $G_T$ est défini par $G_T = X_T - X_0$,

$$\mathbf{E}(G_T) = \mu_1 x_1 + \mu_2 x_2 = \mu_2 + x_1(\mu_1 - \mu_2)$$

et

$$\mathrm{V}ar(G_T) = x.\Gamma x = \sigma_1^2 x_1^2 + \sigma_2^2 (1 - x_1)^2 + 2\rho\sigma_1\sigma_2 x_1 (1 - x_1).$$

---

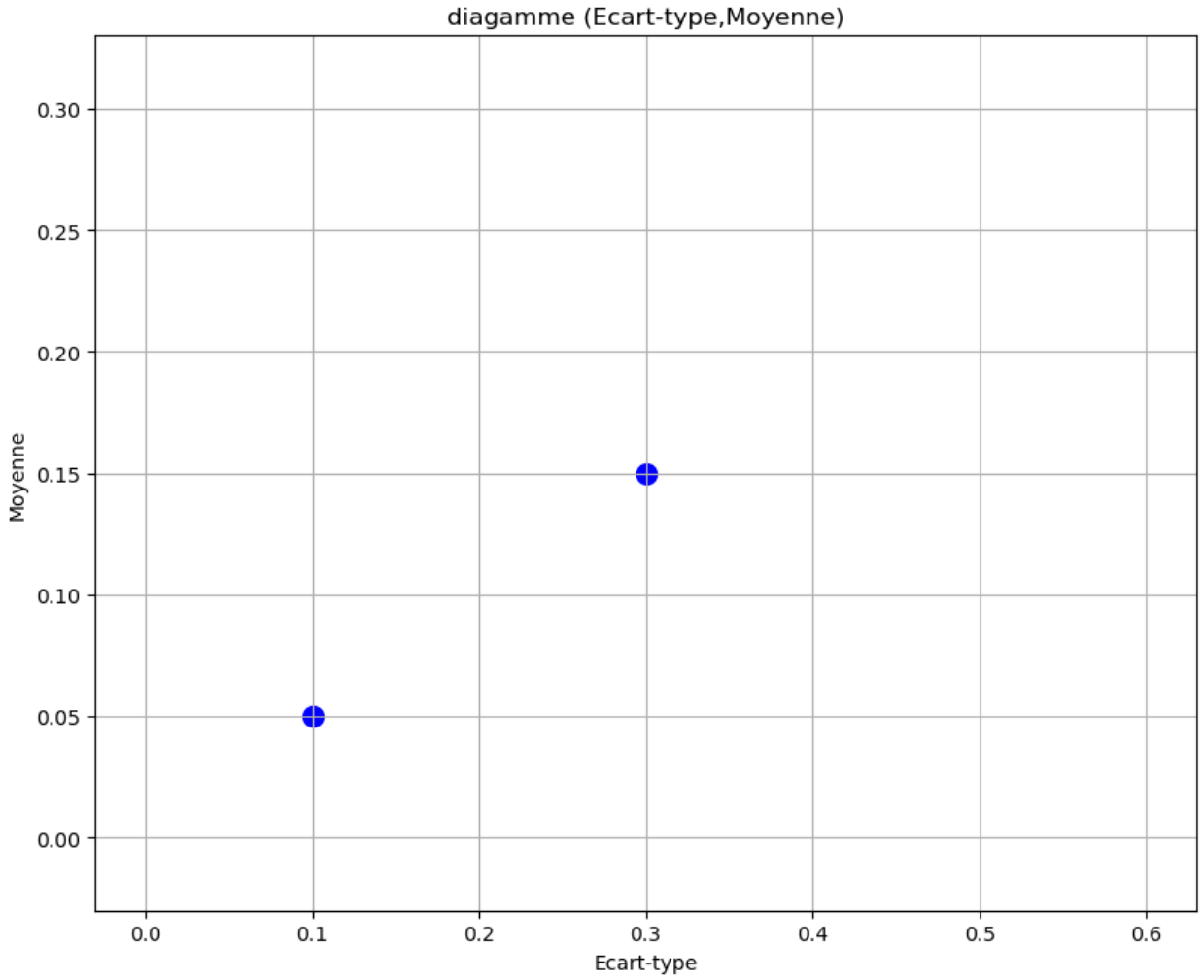
Question 2:
Tracer les caractéristiques des actifs de base dans le plan (moyenne,écart type).

---

In [14]:

```
import numpy as np
import math
import random
import matplotlib.pyplot as plt
```

```
#plt.style.use('dark_background') # Pour un background noir

# On définit les caracteristiques des actifs
d=2
# On choisit les moyennes des actifs.
mu=[0.05,0.15]
# On choisit les variances des actifs.
sigma=[0.10,0.30]

# On choisit pour matrice de corrélation: des 1 sur la diagonale, des rho ailleurs
# rho=0 : cas d'actifs décorrélés (en particulier cas indépendant)
rho=0.0

    ######  A vous de jouer  .....
    # Construire la matrice C
    # covariance = ...
    # Construire la matrice Gamma
    # Gamma = ...

# Les caractéristiques des actifs de base
moyenne_actif=mu
std_actif=sigma

# plot ################################################################
max_sigma=max(std_actif)
max_esp=max(moyenne_actif)
marge=0.03
un_inche_en_cm=2.54 # 1 inche = 2.54 cm

taille_h_cm=25
taille_v_cm=20

marker_size=100

def plot1():
    # On crée un figure dont on fixe la taille et dont on définit les axes
    fig = plt.gcf()
    fig.set_size_inches(taille_h_cm/un_inche_en_cm,taille_v_cm/un_inche_en_cm)
    plt.axis([-marge, 2*max_sigma+marge, -marge, 2*max_esp+marge])
    # On trace les points représentant les 2 actifs.
    plt.scatter(sigma,mu, s=marker_size, c='b',marker='o')
    plt.ylabel('Moyenne')
    plt.xlabel('Ecart-type')
    plt.title('diagamme (Ecart-type,Moyenne)')
    #plt.text(60, .025, r'$\mu=100,\ \sigma=15$')
    plt.grid(True)

plot1()
```




---

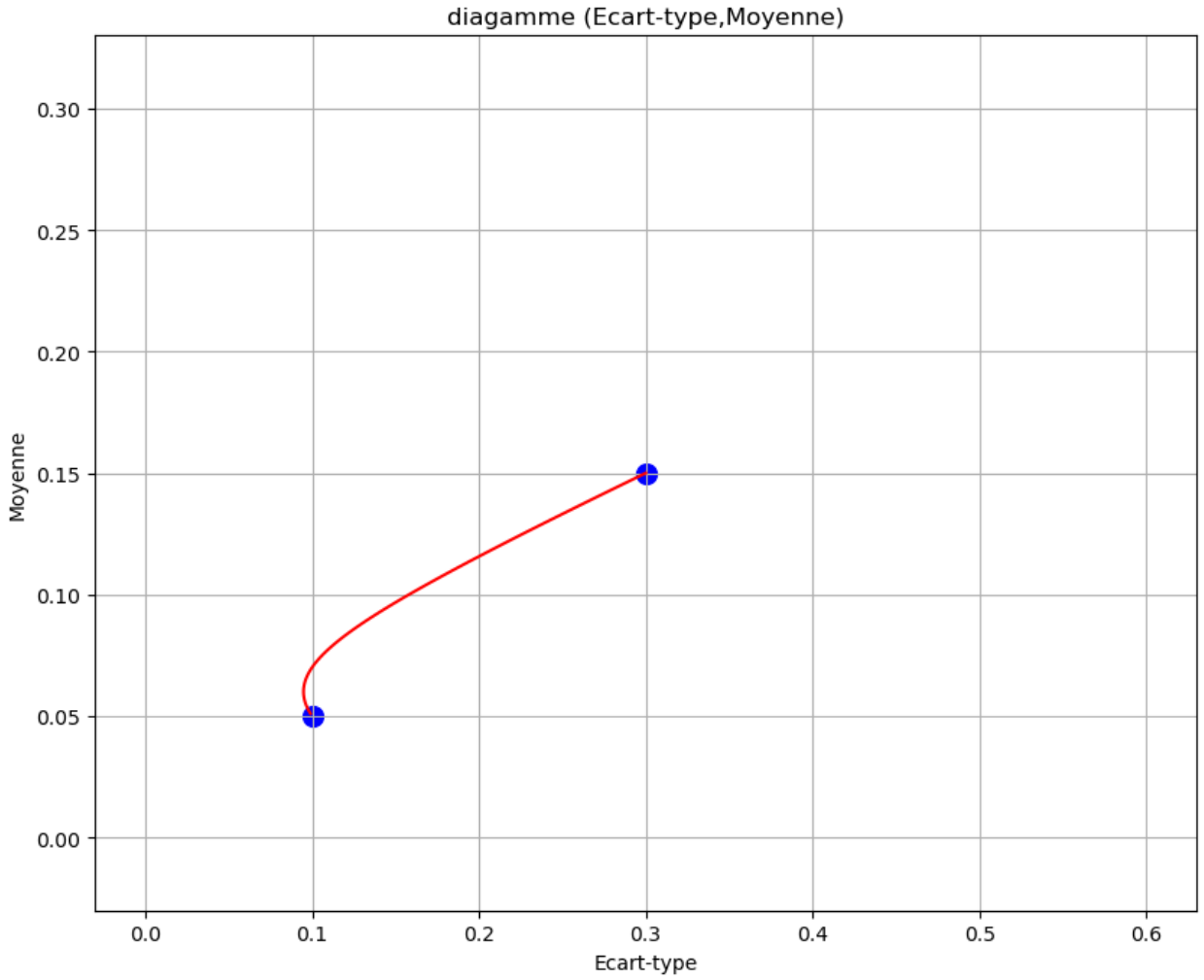
Question 3:

Tracer la courbe $x_1 \in [0,1] \rightarrow (\mathbf{E}(G_T), \sqrt{Var(G_T)})$. Vérifier que l'on peut construire un portefeuille de même variance que l'actif 1 mais dont l'espérance du rendement est supérieure à celle de cet actif. Est il rationnel d'investir dans l'actif 1, si l'on cherche à minimiser son risque ? Quel sont les portefeuilles dans lesquels il paraît rationnel d'investir ?

---

In [15]:

```
def frange(start, stop, step):
#exemple:
#for i in frange(0.5, 1.0, 0.1): print(i)
    i = start
    while i < stop:
        yield i
        i += step

####################
N=100
moyenne_x=np.zeros(N)
std_x=np.zeros(N)
i=0
for x_1 in frange(0.0,1.0,1.0/N):
    current_x = [x_1,1-x_1]  # composition du portefeuille  x_1 + x_2 = 1

    ######  A vous de jouer  .....
    # moyenne_x[i] = ...   calcul de la moyenne du rendement du porfefeuille
    # std_x[i]= ...        calcul de son écart-type
    # np.dot(u,v) calcule le produit scalaire des vecteurs u et v
    # np.dot(M,v) calcule le produit matriciel de M et du vecteur v
    # math.sqrt(x) racine carrée de x

    i=i+1

# plot ########################################################################
def plot2():
    plot1()# le plot précédent
    plt.plot(std_x,moyenne_x, 'r-')

plot2()
```




---

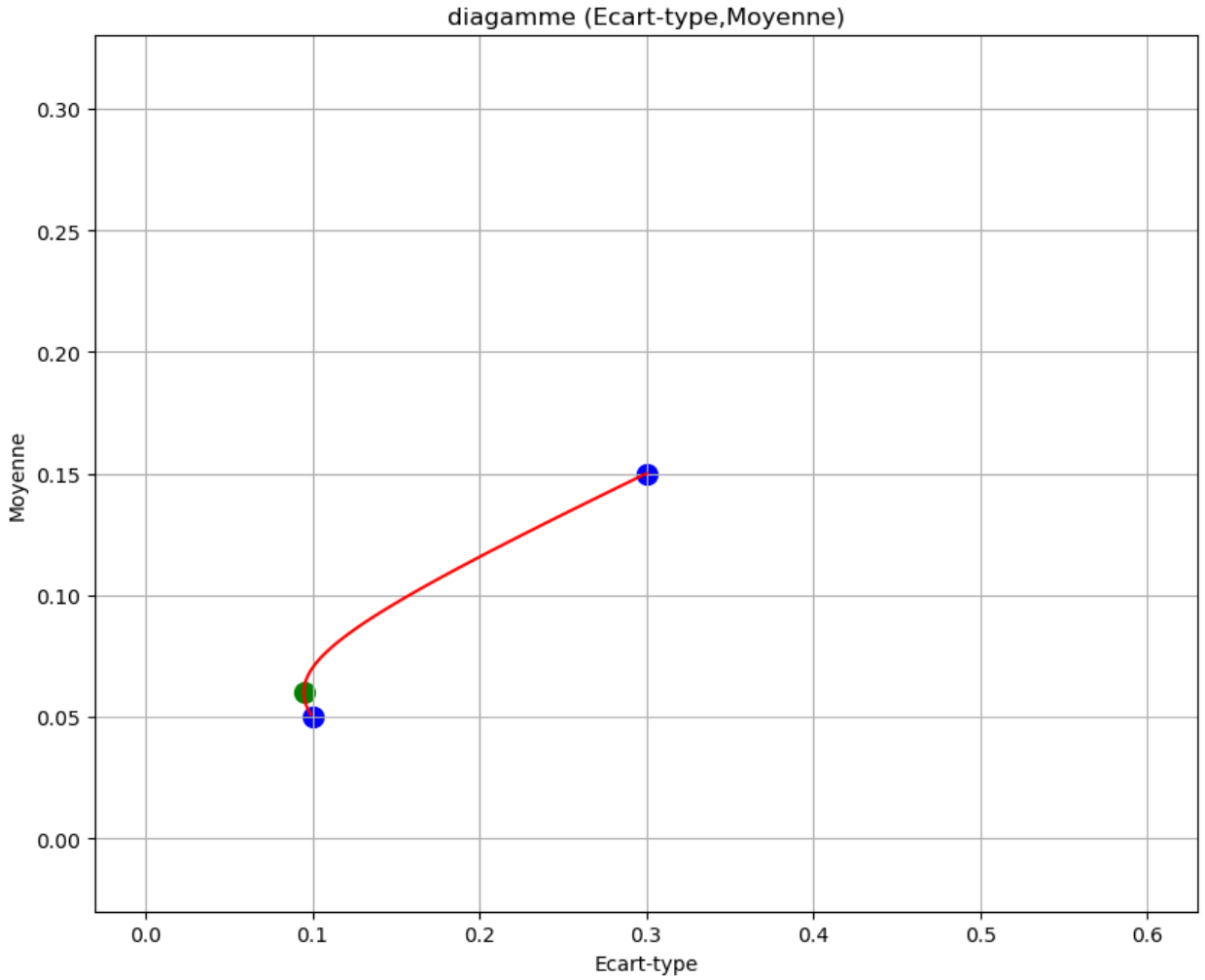
Question 4:
Vérifier que l'on peut construire un portefeuille de variance minimum (et inférieure à celle de l'actif de variance minimum).

---

C'est un exemple de l'**effet de diversification** dans la théorie des portefeuille.

In [16]:

```
# plot ##########################################################################
def plot3():
    plot2()# le plot précédent

    ######  A vous de jouer  .....
    # calcul de l'indice du portefeuille de variance minimum
    # Vecteur.argmin() renvoit l'indice du plus petit nombre du Vecteur
    # imin = ...

    # on trace ce point en vert c='g' (couleur=vert)
    plt.scatter(std_x[imin],moyenne_x[imin], s=marker_size, c='g',marker='o')

plot3()
```




Nous relaxons la condition $x_1 \geq 0, x_2 \geq 0$ tout en continuant à imposer $x_1 + x_2 = 1$ (la valeur totale de notre investissement initial reste égale à $1$ ). Nous allons faire varier $x_1$ entre $-10$ et $0$ (lorsque $x_1$ est négatif, on emprunte une quantité $|x_1|$ d'actif $1$, mais la valeur totale du portefeuille doit toujours rester égale à $1$).

---

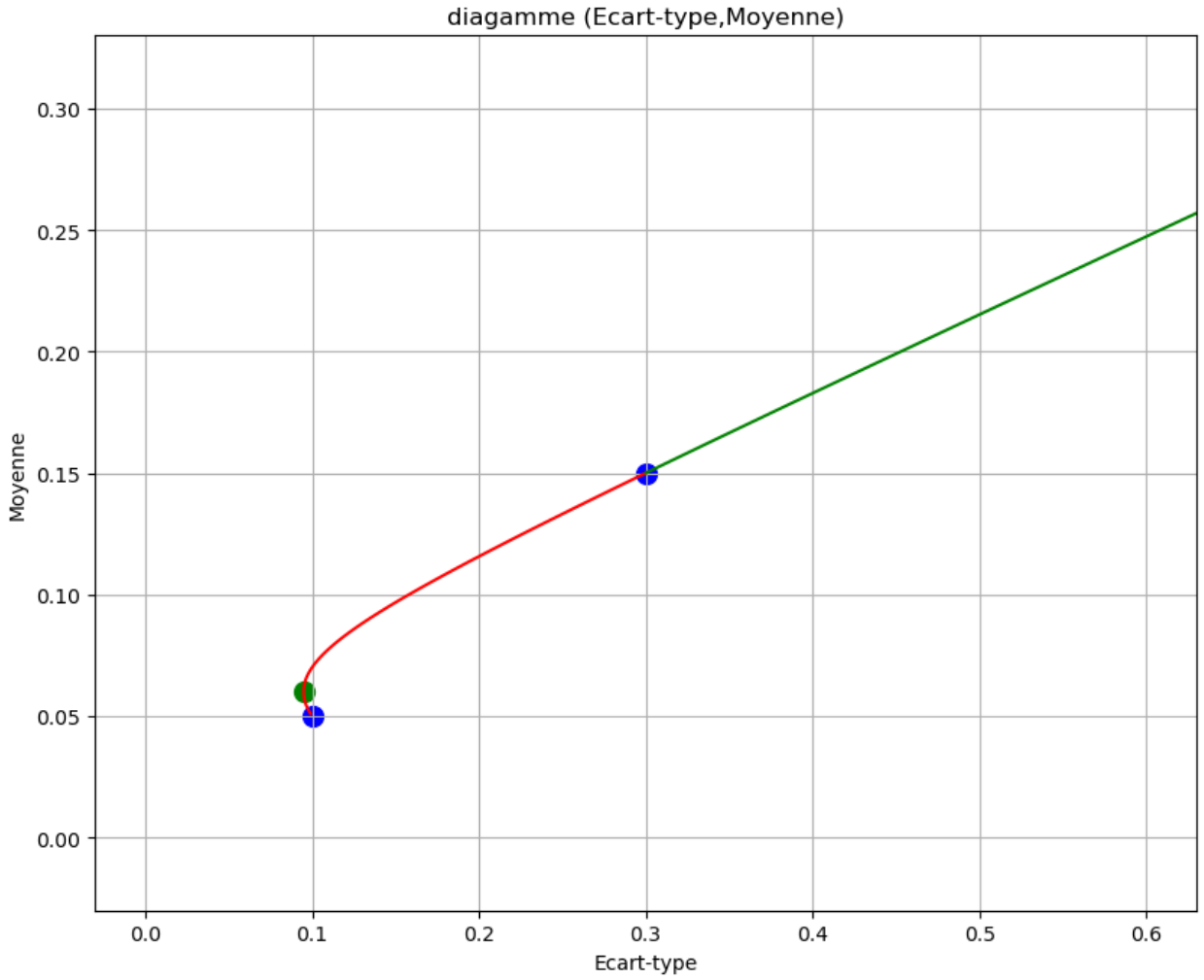
Question 5:

Tracer la courbe $x_1 \in [-5, 0] \rightarrow (\mathbf{E}(G_T), \sqrt{Var(G_T)})$. Vérifier que, si l'on accepte une variance grande, on peut constituer des portefeuilles d'espérance aussi grande que souhaitée (cet effet porte le nom d'__effet de levier__ ou leverage effect). On comprend qu'il ne faille pas en abuser !

---

In [17]:

```
# On autorise l'emprunt de l'actif 1
N=1000
moyenne2_x=np.zeros(N)
std2_x=np.zeros(N)
for i in range(0,N):

    ######  A vous de jouer  .....
    # prendre de valeurs negatives pour x_1 dans [-5,0], en déduire x_2
    # x_1 = - ...
    # x_2 = ...
    # calcul de la moyenne et de l'écart-type de ce portefeuille
    # x=[x_1,x_2]
    # moyenne2_x[i]=...
    # std2_x[i]=...

# plot ################################################################
def plot4():
    plot3()# le plot précédent
    plt.plot(std2_x, moyenne2_x,'g-')

plot4()
```




---

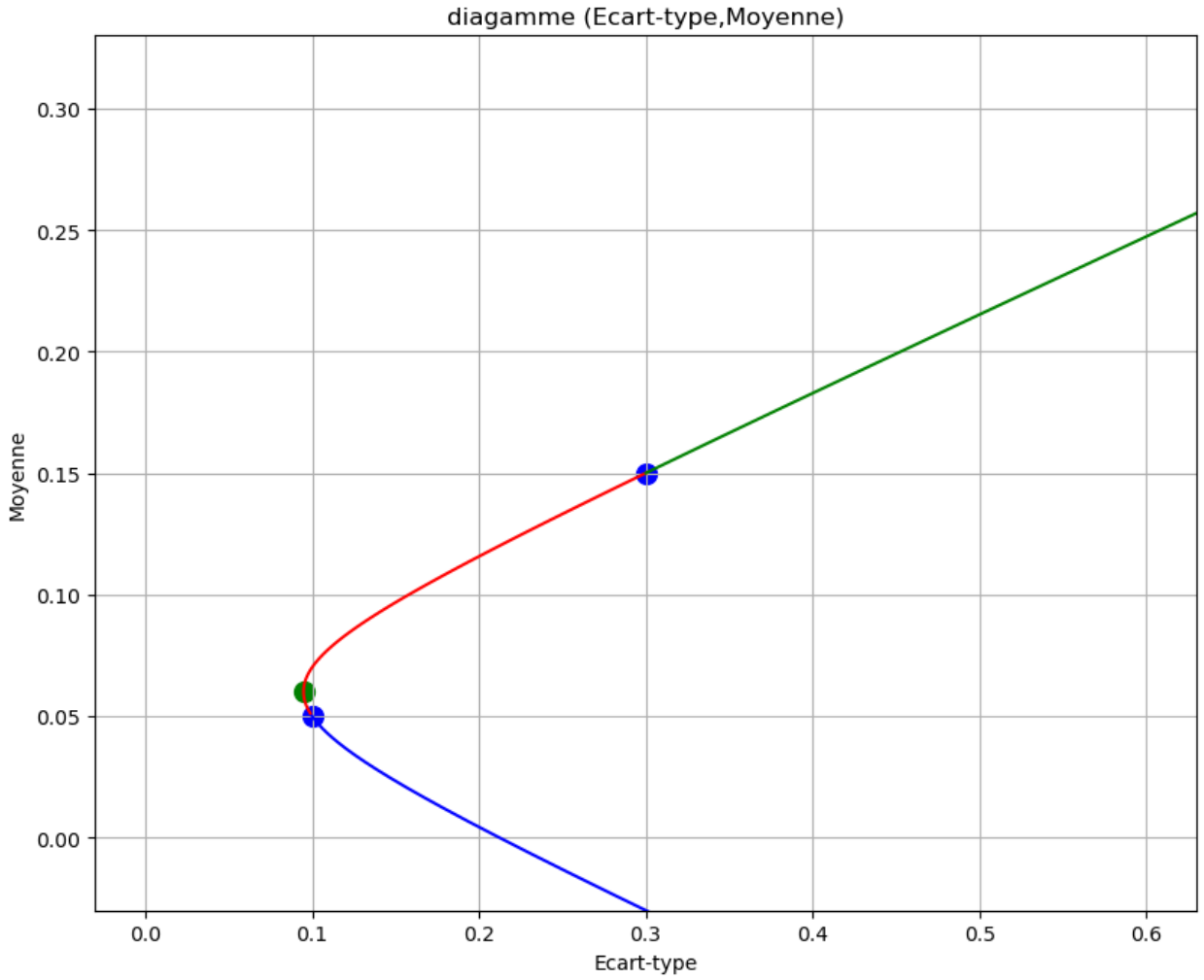
Question 6:

Tracer la courbe $x_2 \in [-5, 0] \to (\mathbf{E}(G_T), \sqrt{Var(G_T)})$. Vérifier que lorsque l'on emprunte l'actif 2 ($x_2$ négatif), l'on fait décroître l'espérance en augmentant la variance (ce qui est loin d'être optimal!).

---

In [18]:

```
# Que se passe t'il lorsque l'on emprunte l'actif 2 ?
N=1000
moyenne3_x=np.zeros(N)
std3_x=np.zeros(N)
for i in range(0,N):

    ######  A vous de jouer  .....
    # prendre de valeurs negatives pour x_2 dans [-5,0], en déduire x_1
    # x_2 = - ...
    # x_1 = ...

# plot ###########################################################################
def plot5():
    plot4() # le plot précédent
    plt.plot(std3_x, moyenne3_x,'b-')

plot5()
```




## Introduction d'un actif sans risque

Nous allons introduire un nouvel actif, l'actif sans risque, qui comme son nom le suggère aura un rendement de variance nulle (ce qui implique que ce rendement n'est pas aléatoire). On supposera que ce rendement déterministe est inférieur à tous les rendements moyens des actifs risqués (pourquoi est-ce une hypothèse raisonnable?). On prendra, ici, ce rendement égal à $0$.

---

Question 7:
Construire le vecteur de moyenne et la matrice de variance-covariance de ce nouveau vecteur des rendements.

---

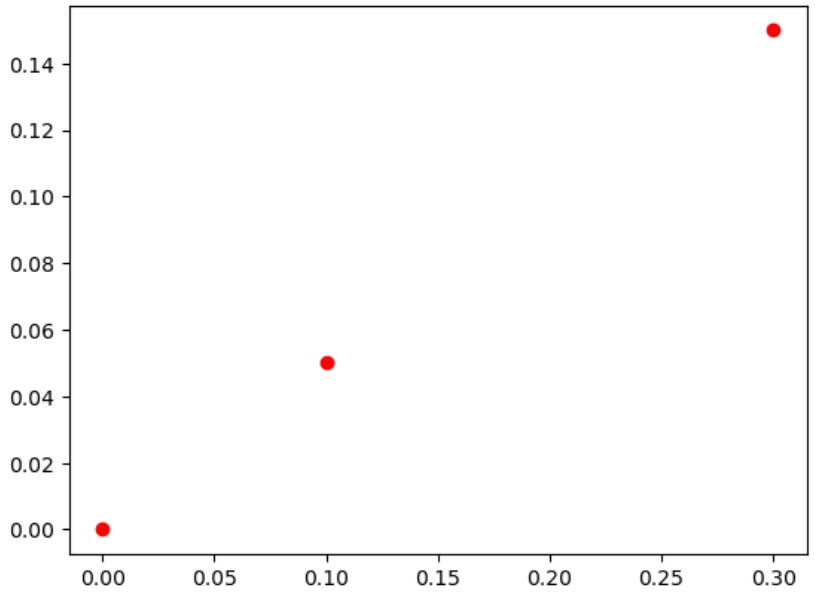
In [19]:

```
# Construction du vecteur des rendements.
# On rajoute un actif de rendement sans risque (de variance nulle).

# Pour la moyenne des rendements il suffit de rajouter le rendement sans risque
# au vecteur des rendements
r0=0
mu_3=np.append(r0,mu)
# On rajoute 0 au vecteur des variances
sigma_3=np.append(0,sigma) # variance (nulle) de l'actif sans risque

######  A vous de jouer  .....
# Construire la nouvelle matrice Gamma_3 pour les trois actifs.


print('Gamma_3=');print(Gamma_3)

# Les vecteurs moyenne et variance des actifs
moyenne_actif=mu_3
std_actif=np.sqrt(np.diag(Gamma_3))

# On peut aussi materialise les 3 actifs de base
plt.plot(std_actif, moyenne_actif, 'ro')
```

```
Gamma_3=
[[0.   0.   0.  ]
 [0.   0.01 0.  ]
 [0.   0.   0.09]]
```

Out[19]:

```
[<matplotlib.lines.Line2D at 0x7f92f697f650>]
```




On constitue des portefeuilles avec les 3 actifs (1 non risqué, 2 risqués) en tirant au hasard des coefficients $(x_1, x_2, x_3)$ dans le simplexe $\left\{(x_1, x_2, x_3) \in \mathbb{R}^3 | \forall i \in \{1, 2, 3\}\ 0 \leq x_i \leq 1 \text{ et } x_1 + x_2 + x_3 = 1\right\}$.

La fonction `simplexe(d)` fait ce travail.

In [20]:

```
def simplexe(d):
# Cette fonction tire "au hasard"  d nombre positifs
# de somme egale 1 (= dans le simplexe)
    t=np.random.rand(d-1)
    t=np.sort(t)[::-1]
    t=np.append(1,t)
    t=np.append(t,0)
    s=np.zeros(d)
    for i in range(d-1,-1,-1):
        s[i]=t[i]-t[i+1]
    return s
```

---

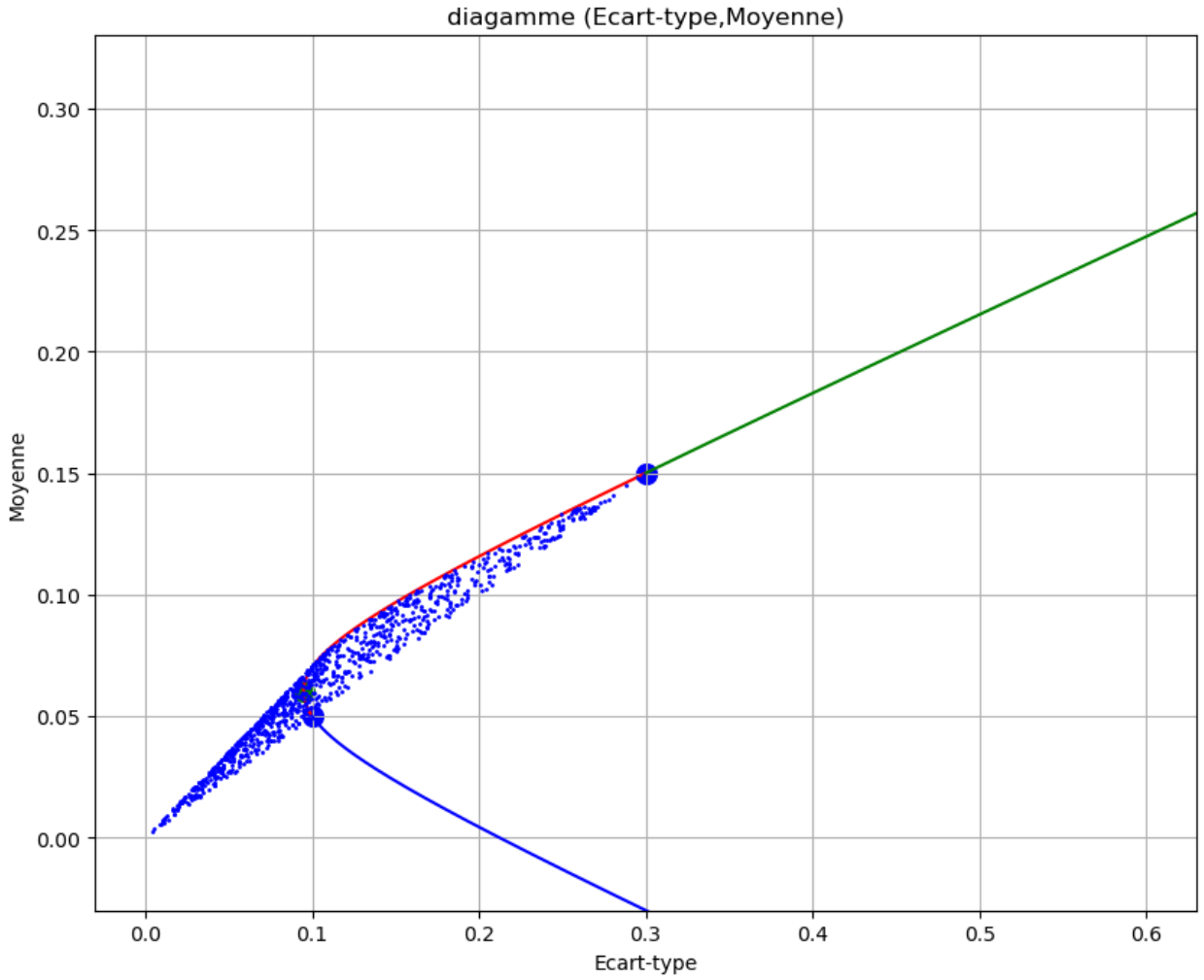
Question 8:
Matérialiser, en tirant un grand nombre de points au hasard, la nouvelle frontière efficiente.
Vérifier que :

- La nouvelle frontière efficiente étend l'ancienne par de nouveaux points "non dominés" entre l'actif sans risque et un portefeuille tangent à l'ancienne frontière $P$.
- La variance reste bornée par la variance la plus grande (tant que l'on ne fait pas d'emprunt). </font>

---

In [21]:

```
# On considère des portefeuilles *incluant l'actif sans risque*
# mais *sans emprunt*. On tire "au hasard" des portefeuilles
# dans le simplexe de dimension 3
N=1000
moyenne_d_x=np.zeros(N)
std_d_x=np.zeros(N)
for i in range(0,N):
    # tirage au hasard dans le simplexe, d=2 et d+1=3 !
    ######  A vous de jouer  .....

# plot ###########################################################################
def plot6():
    plot5()# le plot précédent
    # On rajoute en bleu ('b') les points ('.') représentant
    # les portefeuilles tirés au hasard
    plt.plot(std_d_x, moyenne_d_x,'b.',markersize=2)

plot6()
```




# Commentaire

On obtient de nouveaux points "non dominés" entre l'actif sans risque et un portefeuille tangent. La variance reste bornée par la variance de l'actifs de plus grande variance tant que l'on n'emprunte pas.

On va identifier un portefeuille particulier $P$, le "portefeuille de marché". $P$ est le portefeuille correspondant au point de tangence de la droite passant par l'actif sans risque et de l'ensemble de tous les portefeuilles a coefficients positifs de la question précédente.

Le point $P$ est caractérisé par le fait qu'il maximise la pente des droites reliant le point $(\sigma_0 = 0, r_0 = 0)$ et les points correspondants à des portefeuilles $y$ ne faisant pas intervenir d'actif sans risque.

---

Question 9:
Toujours en procédant par simulation dans le simplexe, calculer $P$ (en fait une approximation de $P$).

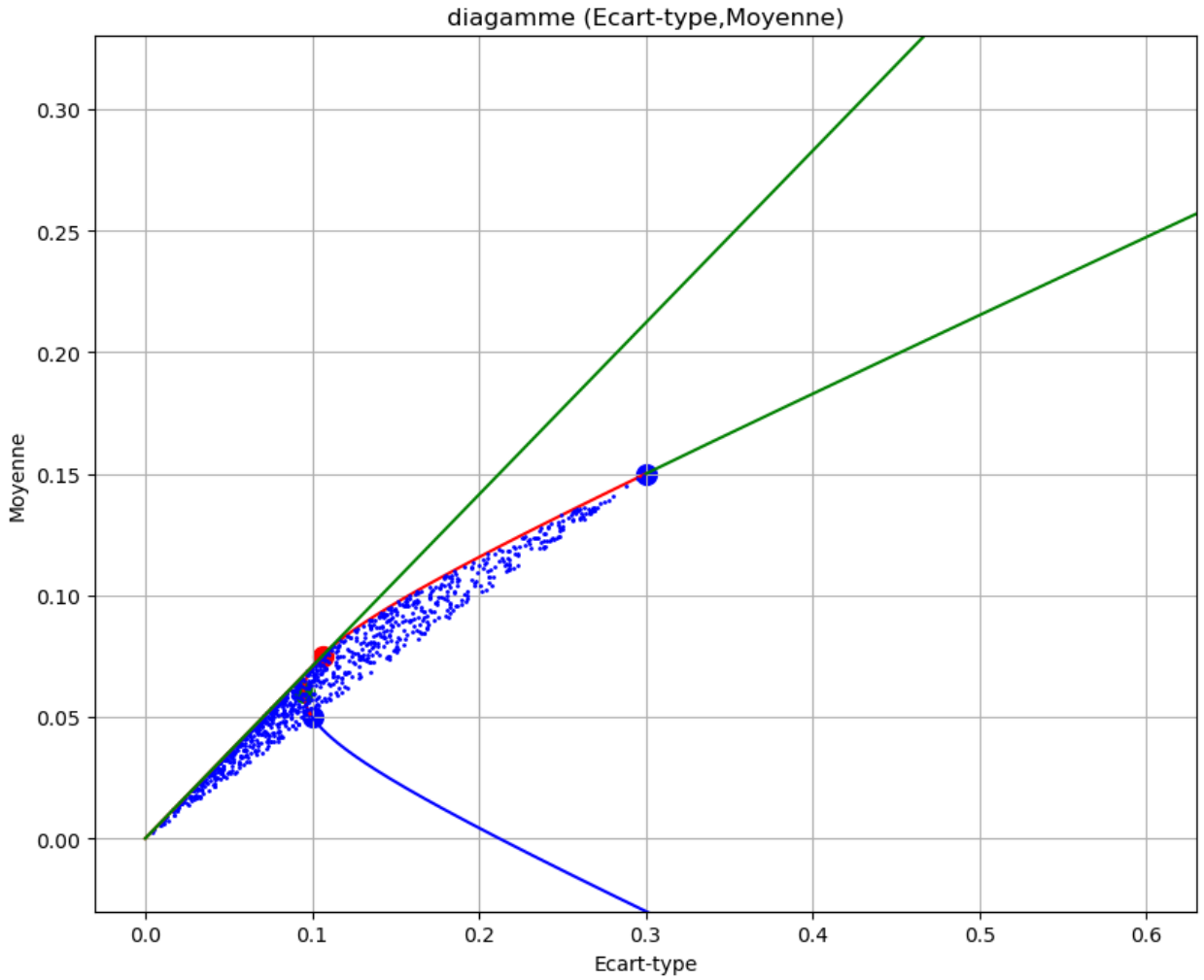
Vérifier que le portefeuille $P$ fait intervenir les $2$ actifs risqués. </font>

---

In [22]:

```
r0=mu_3[0]
sigma0=0

# On veut calculer le point P qui maximise la pente de la droite
# entre (sigma0=0, x_0=r0) et les portefeuilles sans actif sans risque.
# Pour cela, on va genere des portefeuilles sans actifs sans risque
# et calculer le max des pentes ainsi obtenues.
N=1000
moyenne_y=np.zeros(N)
std_y=np.zeros(N)
pente=np.zeros(N)
max_pente=0
for i in range(1,N):
    y = np.array([0,i/N,1-i/N]) # on  rajoute 0 en actif sans risque
    ######  A vous de jouer  .....
    # pente[i] = ....   pente associée au i-ième point

# On calcule le point P maximise la pente
imax=pente.argmax()
x_P=moyenne_y[imax]
```

```
sigma_P=std_y[imax]

# plot ##########################################################################
def plot7():
    plot6()# le plot précédent

    # Tracé du point P en rouge ('r')
    plt.scatter(sigma_P, x_P, s=marker_size, c='r', marker='o')

    # Tracé du segment "Actif sans risque -> P"
    plt.plot(np.array([sigma0,sigma_P]),np.array([r0,x_P]), 'r-')

    # Tracé de la droite "actif sans risque -> P" au dela de P
    lambd=(x_P-r0)/(sigma_P-sigma0)# pente de la droite
    sigma_infinity=2.0# arbitraire mais "grand"
    x_infinity=r0+lambd*(sigma_infinity-sigma0)
    plt.plot(np.array([sigma0,sigma_infinity]),np.array([r0,x_infinity]), 'g-')

plot7()
```




On autorise la détention d'une quantité de signe arbitraire d'actif sans risque (cela correspond soit à un emprunt, soit à un placement). Pour cela on vous suggère de tirer la quantité d'actif sans risque $x_0$ entre $[-4, 1]$ (on peut emprunter jusqu'à $4$ fois ce que l'on possède). Puis on tire, les quantités d'actifs risqués uniformément sur le simplexe $\{x_1 + x_2 = 1 - x_0\}$.

---

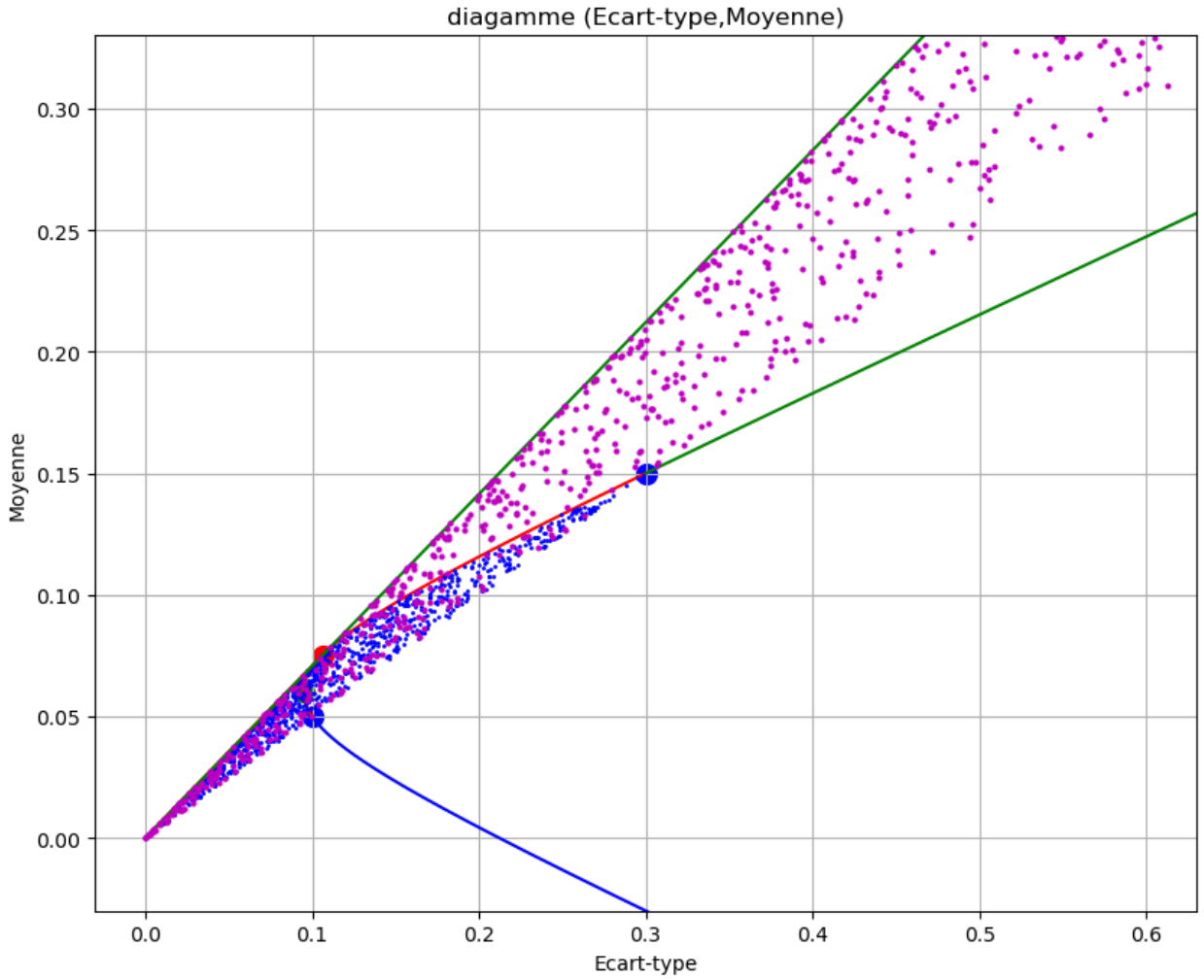
Question 10:
Tirer un grand nombre de portefeuille, calculer leurs moyennes et écarts-type, les tracer sur la figure.

---

In [23]:

```
# L'emprunt en actif sans risque est autorisé
N=1000
moyenne_x_d=np.zeros(N+1)
std_x_d=np.zeros(N+1)
for i in range(1,N):
    # On génère des portefeuilles dont la quantité
    # d'actif sans risque est uniforme sur [-4,1]
    ######  A vous de jouer  .....

# plot ##########################################################################
def plot8():
```

```
    plot7()# le plot précédent

    # Tracé des points tirés au hazard
    plt.plot(std_x_d,moyenne_x_d,'mo',markersize=2)

plot8()
```




---

Question 11:
Vérifier que:

1. l'on obtient de nouveaux points "non dominés" au delà du portefeuille tangent.
2. le rendement (mais aussi la variance) peut devenir aussi grand que souhaité: **effet de levier**.
3. un emprunt permet de construire des portefeuilles dont la moyenne des rendements est plus élevée à variance égale: **emprunter permet d'augmenter la moyenne du rendement**.
4. l'emprunt permet de construire des portefeuilles de même moyenne mais de variance inférieure: **emprunter permet de réduire le risque**. Il existe en particulier un portefeuille dont la variance est égale à celle de l'actif 2 (l'actif de rendement maximum) mais de rendement supérieur.
5. le seul point de la "frontière sans emprunt" qui n'est pas dominé par un point de la "frontière avec emprunt" est le point $P$: si l'on ne souhaite pas emprunter, "le seul point rationnel est" $P$.
6. le portefeuille $P$ fait intervenir l'ensemble des actifs de base risqués (en dehors des actifs de base dominés par d'autres actifs de base). </font>

---

# Partie optionnelle: extensions du modèle à plus de 2 actifs risqués

Le modèle de Markowitz vient d'être illustré dans le cas où l'on considère deux actifs risqués décorrélés ($\rho = 0$) et un actif sans risque. On peut évidemment généraliser l'approche au cas d'actifs corrélés et en faisant intervenir un nombre arbitraire d'actifs.

## 1. Le cas d'un corrélation non nulle

---

Question 12:
Recommencer l'expérience précédente avec des valeurs de $\rho$ non nulle. Prendre par exemple

$$\rho = -0.5 \quad \text{et} \quad \rho = 0.5$$

---

Les scripts précédents fonctionnent dans ce cas. Nous vous laissons le soin d'expérimenter par vous même.

In [24]:
```
# Matrice de covariance: des 1 sur la diagonale, des rho ailleurs
rho=0.5# -0.5
covariance=rho*np.ones([d,d])+(1-rho)*np.eye(d)
Gamma = np.matmul(np.matmul(np.diag(sigma),covariance), np.diag(sigma))
# etc ...
```

## 2. Le cas d'un nombre d'actifs risqués arbitraires

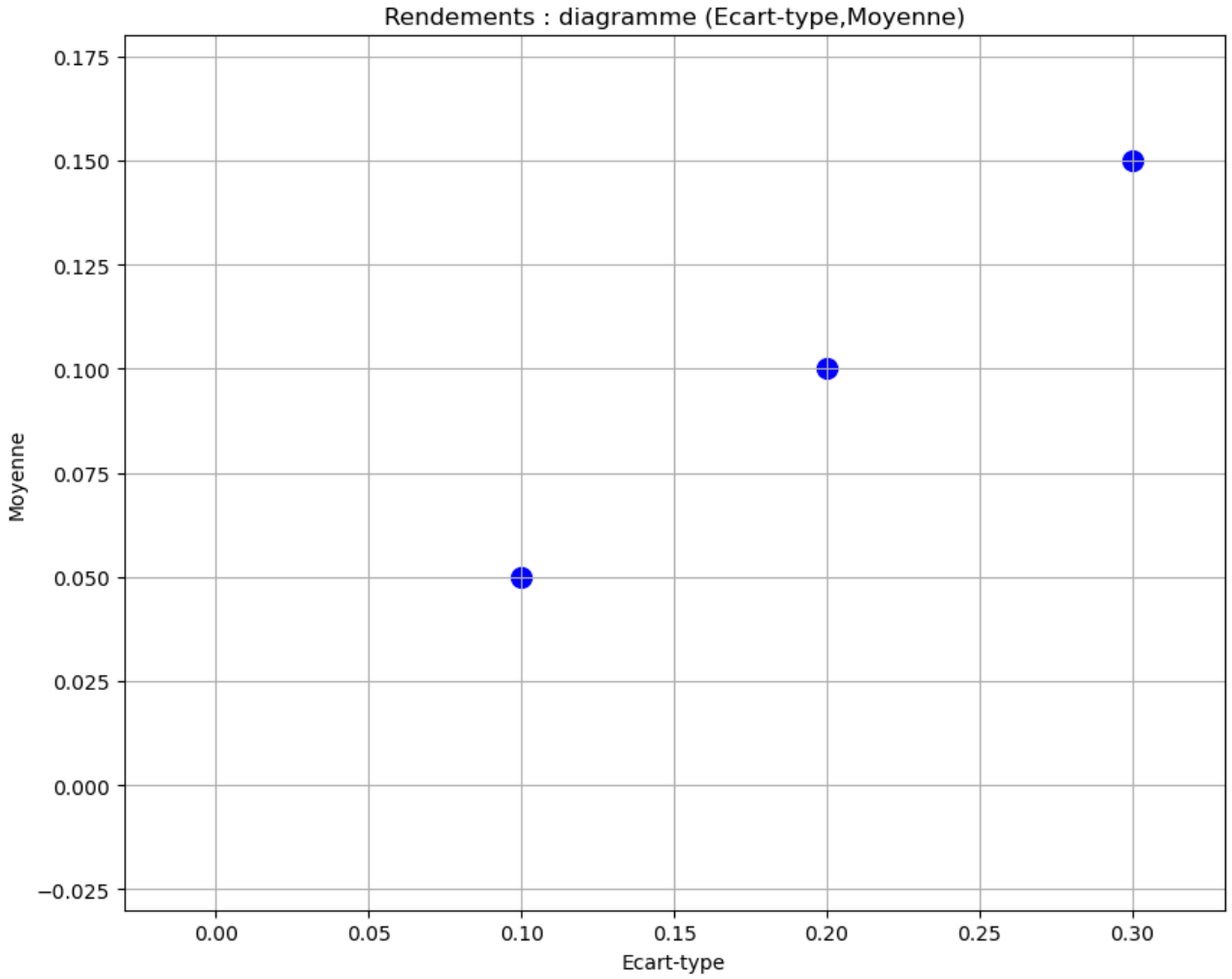
Lorsque $d > 2$ les phénomènes sont identiques mais moins explicites. On peut recommencer ce qui précède mais il faudra généraliser le choix de la matrice de variance covariance et procéder par simulation dans tous les cas.

Les programmes fournis en correction fonctionnent (le plus souvent) en dimension arbitraire.

A titre indicatif voici un exemple du cas $d = 3$.

#### Etape 1. Choix des actifs de base.

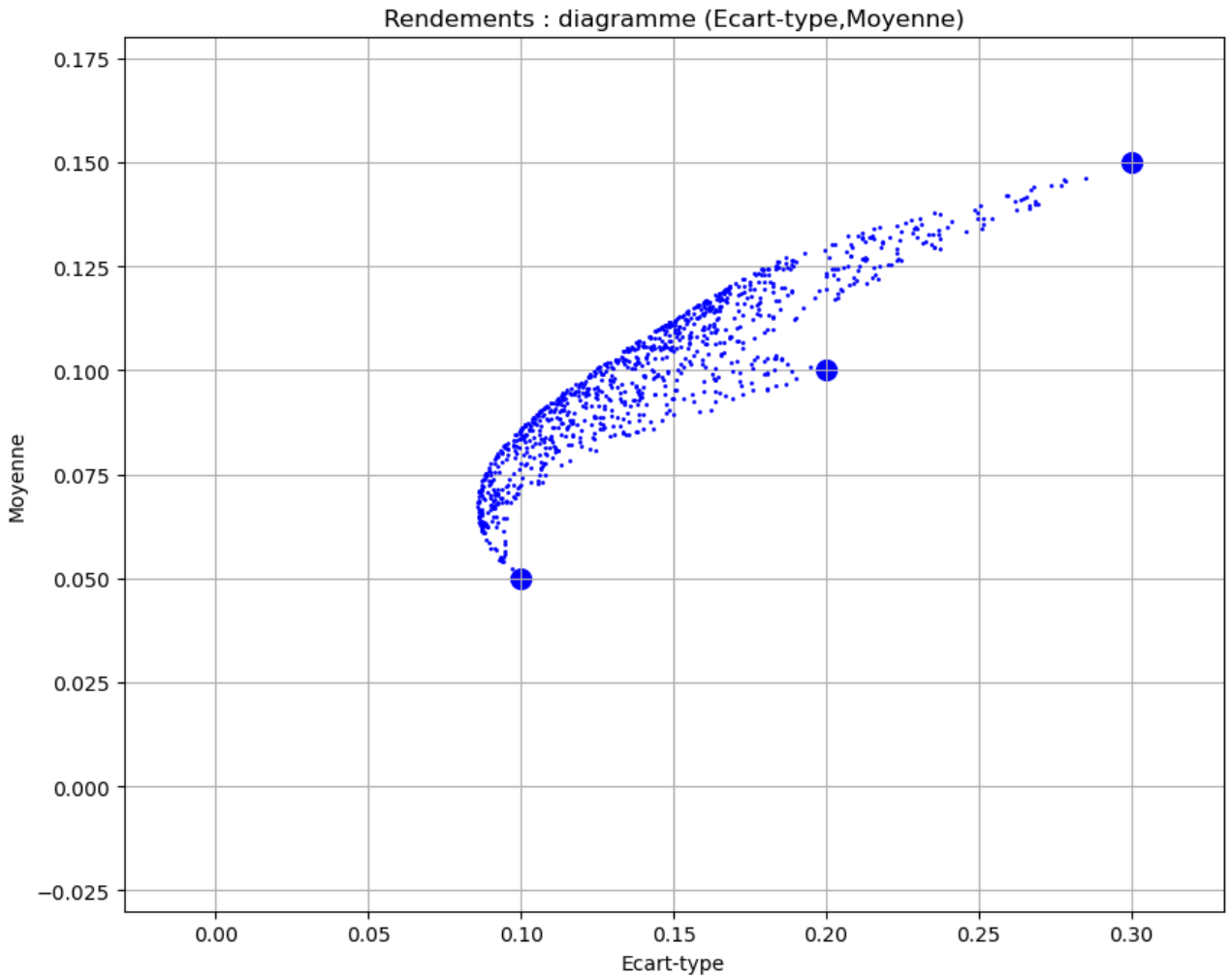
In [25]:
```
# On définit les caracteristiques des actifs risqués
d=3
rho=0.0

min_esp=0.05
max_esp=0.15
mu=np.linspace(min_esp,max_esp,d)

# On suppose que tous les actifs risqués ont une
# corrélation constante égale à |$\rho$|.
# On doit forcement avoir |$\rho >= -(1/(d-1))$|,
# sinon la matrice n'est pas une matrice de covariance (exercice!).
covariance=rho*np.ones([d,d])+(1-rho)*np.eye(d)

# On choisit un ecart type croissant en fonction de l'actif
min_sigma=0.1
max_sigma=0.3
sigma=np.linspace(min_sigma,max_sigma,d)

# La matrice de variance covariance se calcule par :
Gamma = np.diag(sigma) * covariance * np.diag(sigma)

# Les caractéristiques des actifs de base
moyenne_actif=mu
std_actif=np.sqrt(np.diag(Gamma))

# plot ##############################################################

# Tracé des actifs dans le plan (ecart-type,moyenne)
max_sigma=max(std_actif)
max_esp=max(moyenne_actif)
marge=0.03
un_inche_en_cm=2.54 # 1 inche = 2.54 cm

taille_h_cm=25
taille_v_cm=20

def plot2_1():
    # On crée un figure dont on fixe la taille et dont on définit les axes
    fig = plt.gcf()
    fig.set_size_inches(taille_h_cm/un_inche_en_cm,taille_v_cm/un_inche_en_cm)
    plt.axis([-marge, max_sigma+marge, -marge, max_esp+marge])
    # On trace les points représentant les actifs de risqués
    plt.scatter(sigma,mu, s=marker_size, c='b', marker='o')
    plt.ylabel('Moyenne')
    plt.xlabel('Ecart-type')
    plt.title('Rendements : diagramme (Ecart-type,Moyenne)')
    #plt.text(60, .025, r'$\mu=100,\ \sigma=15$')
    plt.grid(True)


plot2_1()
```




**Etape 2. Tirages des portefeuilles à coefficients positifs.**

In [26]:

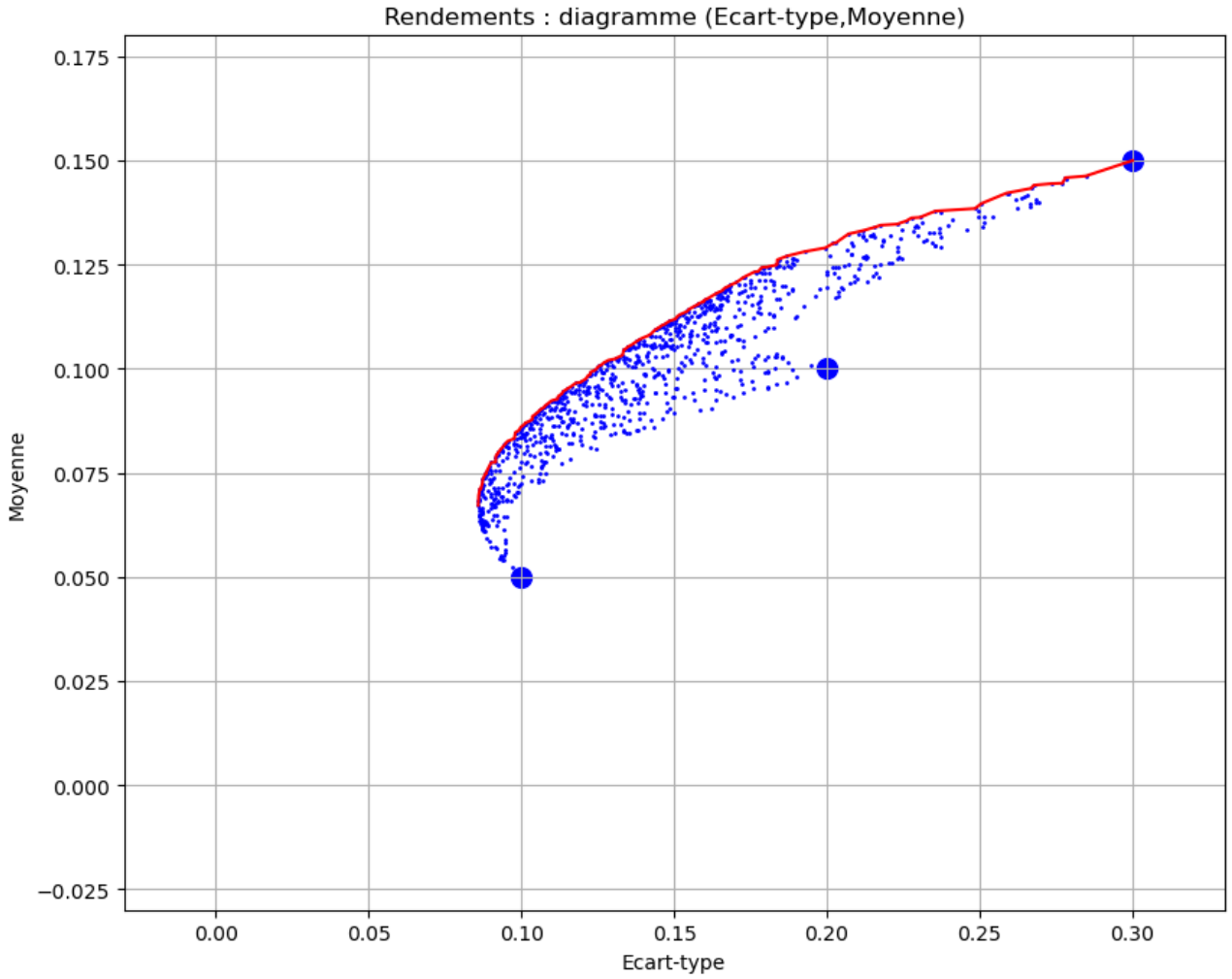
```
# On considère des portefeuilles *avec l'actif sans risque*
# mais *sans emprunt*. On les tire au hasard dans le simplexe
# de dimension 3
N=1000
moyenne1_d_x=np.zeros(N)
std1_d_x=np.zeros(N)
for i in range(0,N):

    ######  A vous de jouer  .....


    x=simplexe(d)# tirage au hasard dans le simplexe
    moyenne1_d_x[i] = np.dot(mu,x)
    std1_d_x[i]=math.sqrt(np.dot(x,np.matmul(Gamma,x)))

# plot ###########################################################################
def plot2_2():
    plot2_1()# le plot précédent
    plt.plot(std1_d_x, moyenne1_d_x,'b.',markersize=2)

plot2_2()
```




On peut estimer la frontière efficiente à partir de ces tirages. Il suffit de ne garder que les points non dominés parmis les tirages.

In [27]:

```
def frontiere_efficiente(moyenne_x,std_x):
# Calcule la fontiere efficiente
# a partir des points de R^2 ((moyenne_x(i),std_x(i)),1<=i<=N)
    ######  A vous de jouer  .....


def plot2_3():
    plot2_2();# le plot précédent
    [frontiere_m_1,frontiere_s_1] = frontiere_efficiente(moyenne1_d_x,std1_d_x) # calcul de la frontière
    plt.plot(frontiere_s_1, frontiere_m_1,'r-',markersize=2)

plot2_3()
```




## Partie optionnelle: solution des problèmes d'optimisation associés

Les complèments qui suivent sont optionnels. Ils montrent cmment calculer par diverses méthodes la frontière efficiente ainsi que le portefeuille de marché.

### Calcul du portefeuille de marché par optimisation

Le calcul du portefeuille de marché $P$, s'exprime sous la forme d'un problème d'optimisation avec contrainte.

Ce poblème se résout avec des techniques classiques implémentées dans **Python** et qui utilisent vos cours d'optimisation de 1A (passé) et de 2A (futur!).

La fonction *minimize* de *scipy* minimise une fonction, si on lui fournit une fonction *cost* qui renvoie la valeur du coût ainsi que de la dérivée du coût en fonction des paramètres. $x_0$ est la valeur initiale de l'algorithme. *minimize* renvoie la valeur de l'optimum dans f et le minimiseur dans *xopt*.

In [30]:

```
from scipy.optimize import minimize

# Choix des caracteristiques des actifs sans risque
d=3
rho=0.0
min_esp=0.05;max_esp=0.15
mu=np.linspace(min_esp,max_esp,d)
min_sigma=0.1;max_sigma=0.3
sigma=np.linspace(min_sigma,max_sigma,d)
correlation =rho*np.ones([d,d])+(1-rho)*np.eye(d)
Gamma = np.diag(sigma) * correlation * np.diag(sigma)

# Définition de la fonction de coût et de sa dérivée
# en utilisant la contrainte |$\sum_{i=1}^d \lambda_i=1$|

def cost(x):
    # Renvoie la valeur de la fonction a minimiser
    # On maximise
    #       f = (mu*lambd)^2 / lambd'*Gamma*lambd
    # sous la contrainte |$\sum_{i=1}^d \lambda_i=1$|, |$x=\lambd(2:d)$|
    # x_1 est calculé en fonction de |$\lambd$| à partir de x(2:d)
```

```
    # en utilisant la contrainte |$\sum_{i=1}^d \lambd_i = 1$|

def cost_der(x):
    # le dérivée du coût en fonction de lambda
    # x_1 est calculé en fonction de |$\lambd$| à partir de x(2:d)
    # en utilisant la contrainte |$\sum_{i=1}^d \lambd_i = 1$|

x0=np.ones(d-1)/d
xopt = minimize(cost, x0, method='BFGS', jac=cost_der, options={'disp': True})
Xopt=np.append(1-sum(xopt.x),xopt.x)
Fopt=math.sqrt(-xopt.fun)

# Est on bien entre |$0$| et |$1$| ? C'est toujours le cas pour
# Rho diagonale mais pas toujours dans le cas non diagonal
# ok = and(0<=Xopt) & and(Xopt<=1)

# Lorsque rho=0, il y a une solution explicite (exercice)
# lambda_i = alpha * mu_i/sigma_i^2, renormalisé
# On verifie ...
sigma=np.transpose(np.sqrt(np.diag(Gamma)))
x= np.divide(mu, np.multiply(sigma,sigma))
x=x/np.sum(x)# normalisation
x=np.transpose(x)
print('Lorsque rho=0, vérfiez que ce qui suit est petit, sinon vous avez un problème ...')
print(np.linalg.norm(x - Xopt)) # lorsque la matrice Rho est diagonale
                          # ça devrait etre petit
```

```
Optimization terminated successfully.
         Current function value: -0.750000
         Iterations: 7
         Function evaluations: 11
         Gradient evaluations: 11
Lorsque rho=0, vérfiez que ce qui suit est petit, sinon vous avez un problème ...
1.6104417260799285e-07
```

In [ ]: