Kalman filtering

April 14, 2015

1 Import utiles

```
In [1]: import numpy as np
        import scipy, scipy.linalg, scipy.signal
        from pykalman import KalmanFilter
        import matplotlib.pyplot as plt
        import pickle
        import pylab
        import csv

        %matplotlib inline
        pylab.rcParams['figure.figsize'] = (14.0, 8.0)
```

2 Etude

Paramètres du modèle

Les variables ci-dessous sont les paramètres qui décrivent le modèle, c'est à dire : * les matrices de transition A (états) et H (observations) * les covariaces Q (états) et R (observations) * l'état initial m_0 * la covariance initiale P_k

```
In [2]: osigma=0.1;

        transition_matrix = np.array([[1., 0.,0.],[1., 1.,0.],[0.,0,0.9]])
        transition_covariance = np.zeros((3,3));

        observation_matrix = np.array([[0., 1.,0.],[0., 0.,1.]])
        observation_covariance = np.eye(2)*osigma;

        initial_state_mean = np.array([1,0,10])
        initial_state_covariance = np.eye(3);
```

2.1 Filtrage à la main

Init du filtre

On initialise le filtre avec les paramètres du modèle.

```
In [3]: kf = KalmanFilter(
            transition_matrix, observation_matrix,
            transition_covariance, observation_covariance,
        )
```

Observations

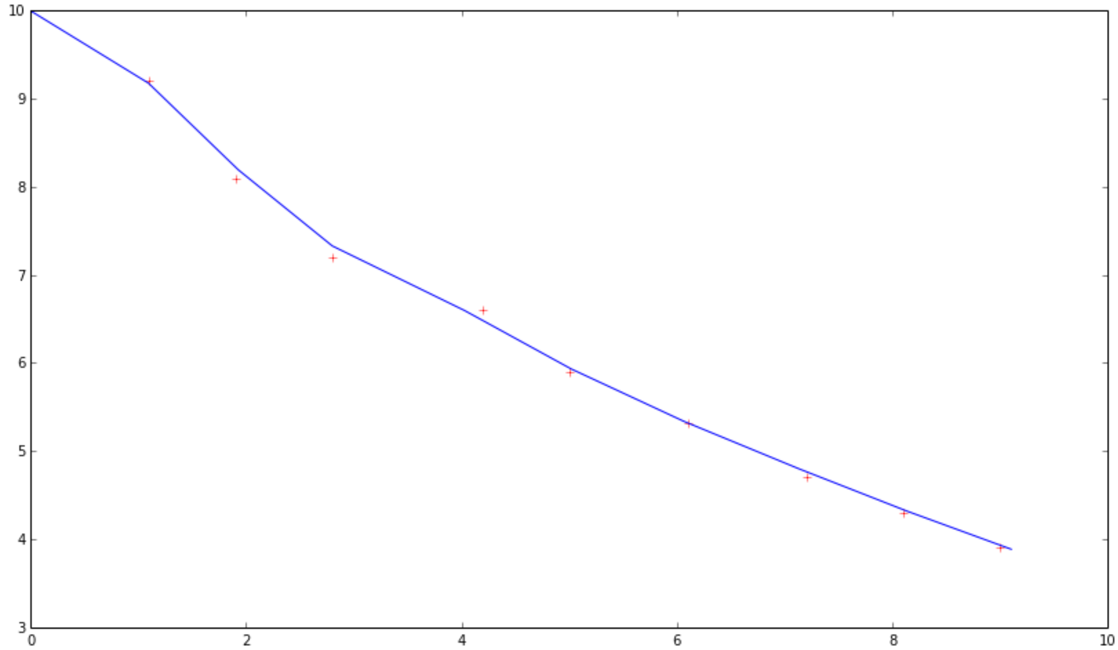
```
In [4]: observations = np.array([ [1.1,9.2],
                                  [1.9,8.1],
                                  [2.8,7.2],
                                  [4.2,6.6],
                                  [5.0,5.9],
                                  [6.1,5.32],
                                  [7.2,4.7],
                                  [8.1,4.3],
                                  [9.0,3.9]])
```

Calcul du filtrage point par point

Pour chaque point, on calcule m_k et P_k à partir de m_{k-1} et P_{k-1} et de y_k. `filter_update` réalise à la fois les tâches de prédiction et de mise à jour.

On calcule finalement les $\hat{y}_k$ filtrés en multipliant les m_k par chaque observation y_k.

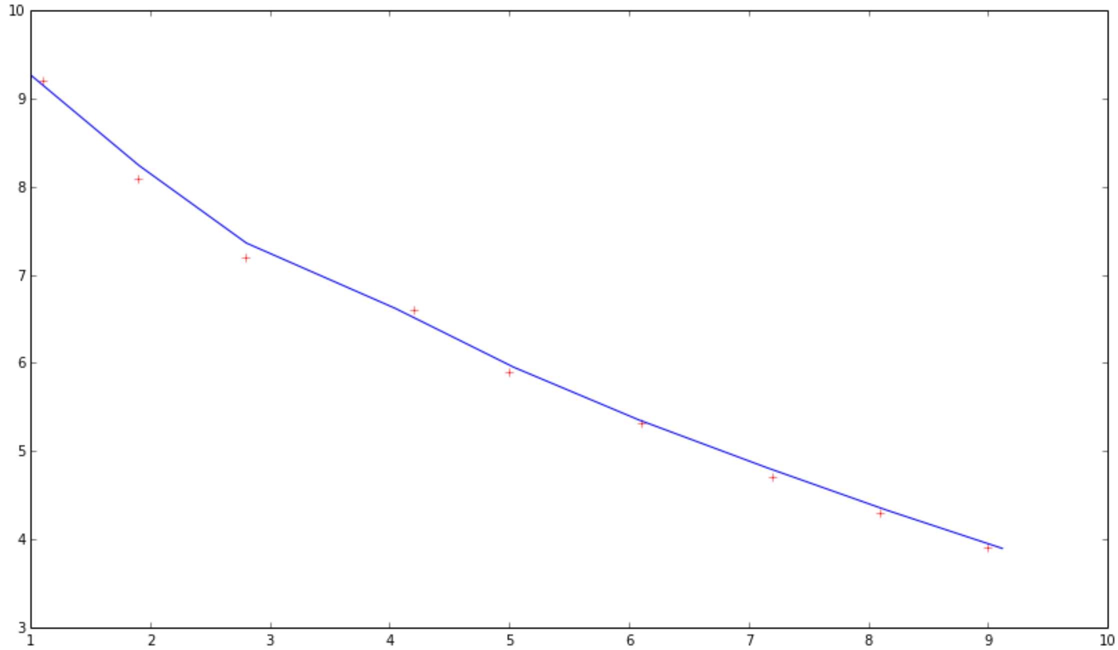
```
In [5]: # init
        hand_state_estimates = [initial_state_mean]
        hand_state_cov_estimates = [initial_state_covariance]

        # filtrage
        for anObs in observations:
            (aMean, aCov) = kf.filter_update(hand_state_estimates[-1], hand_state_cov_estimates[-1], an
            hand_state_estimates.append(aMean)
            hand_state_cov_estimates.append(aCov)

        hand_state_estimates = np.array(hand_state_estimates)

        # Calcul des positions filtrées
        hand_positions = np.dot(hand_state_estimates, observation_matrix.T)

        # Plot
        plt.figure()
        plt.plot(observations[:,0],observations[:,1], 'r+')
        plt.plot(hand_positions[:,0],hand_positions[:,1], 'b')

Out[5]: [<matplotlib.lines.Line2D at 0x13b95a58>]
```



2.2 Filtrage complet

On réalise cette fois un filtrage "complet" c'est à dire en utilisant la fonction `filter` du filtre.

```
In [6]: # Init du filtre

        kf = KalmanFilter(
            transition_matrix, observation_matrix,
            transition_covariance, observation_covariance,
            initial_state_mean=initial_state_mean, initial_state_covariance = initial_state_covariance,
        )

        # Filtrage des observations
        (filtered_state_estimates, filtered_state_cov_estimates) = kf.filter(observations)
        filtered_positions = np.dot(filtered_state_estimates, observation_matrix.T)

        # Affichage
        plt.figure()
        plt.plot(observations[:,0],observations[:,1], 'r+')
        plt.plot(filtered_positions[:,0],filtered_positions[:,1], 'b')

Out[6]: [<matplotlib.lines.Line2D at 0x1401c9b0>]
```

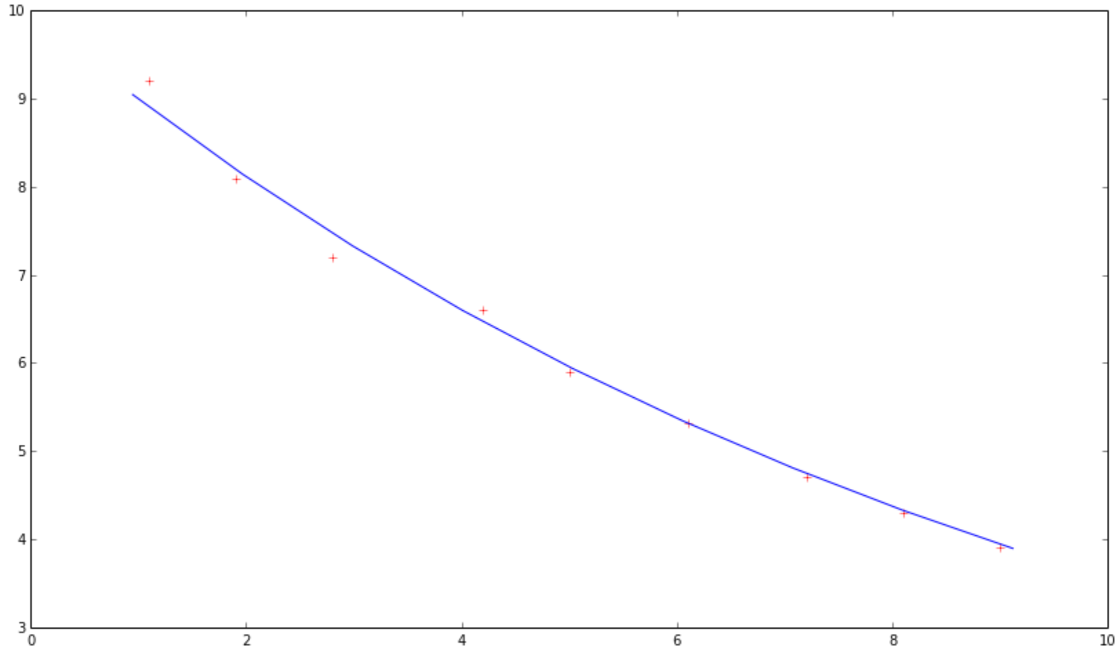


2.3 Lissage

On réalise cette fois du lissage (`smooth`) c'est à dire que l'on a) la fois une passe forward (équivalent du filtrage) et une passe backward.

```
In [7]: # Init du filtre
        kf = KalmanFilter(
            transition_matrix, observation_matrix,
            transition_covariance, observation_covariance,
            initial_state_mean = initial_state_mean, initial_state_covariance = initial_state_covarianc
        )

        # Lissage des observations
        (smoothed_state_estimates,smoothed_state_cov_estimates) = kf.smooth(observations)
        smoothed_positions = np.dot(smoothed_state_estimates, observation_matrix.T)

        plt.figure()
        plt.plot(observations[:,0], observations[:,1], 'r+')
        plt.plot(smoothed_positions[:,0], smoothed_positions[:,1], 'b')
```

```
Out[7]: [<matplotlib.lines.Line2D at 0x141ebd30>]
```



3 Pratiques

3.1 Modèle

Vous disposez du fichier `voitureObservations.csv` qui contient un enregistrement bruité de la position d'une voiture soumises à une force constante. La période d'échantillonnage est de $0.2s$.

La voiture se trouve au temps 0 à la position $(0m, 0m)$ avec une vitesse initiale $(0.75m/s, 2m/s)$, la force qui lui est appliquée correspond à une accélération de $(0.3m/s^2, 0.1m/s^2)$. Le bruit d'observation est un bruit blanc de variance 2.

3.1.1 Paramètres du modèle

$$A = \begin{bmatrix} 1 & 0 & \Delta t & 0 & 0 & 0 \\ 0 & 1 & 0 & \Delta t & 0 & 0 \\ 0 & 0 & 1 & 0 & \Delta t & 0 \\ 0 & 0 & 0 & 1 & 0 & \Delta t \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \text{ avec } \Delta t = 0.2$$

$$H = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \end{bmatrix}$$

$$Q = 0I_6$$

$$R = 2I_2$$

3.1.2 Etat initial

$$m_0 = [p_{x0}, p_{y0}, v_{x0}, v_{y0}, a_{x0}, a_{y0}] = [0, 0, 0.75, 2, 0.3, 0.1]$$

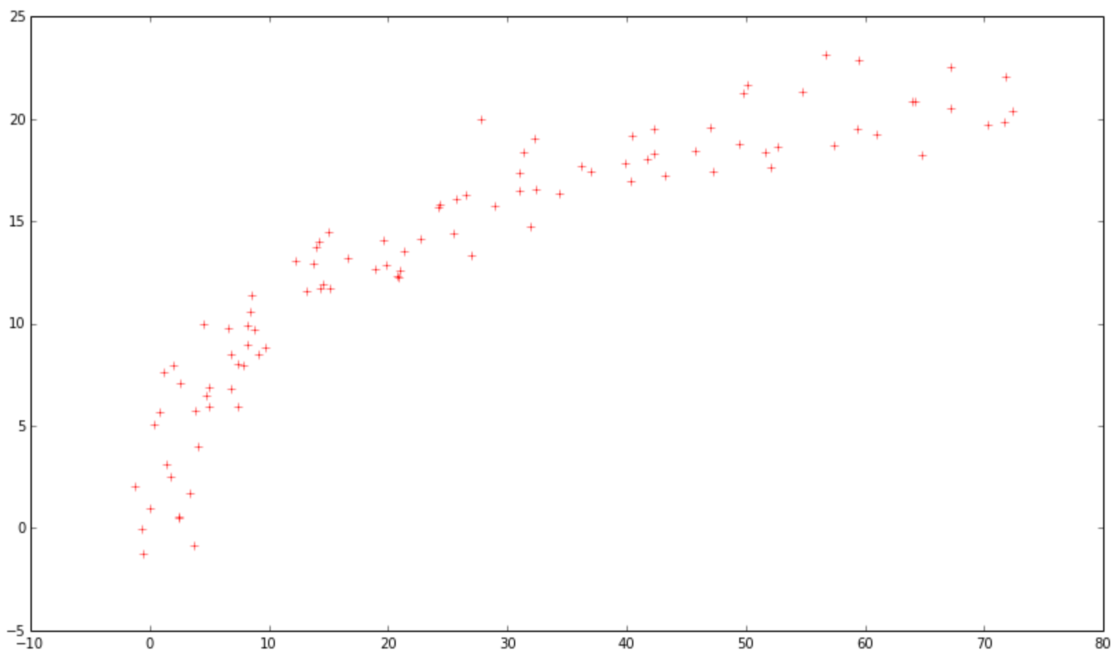
$$P_0 = I_6$$

3.2 Code

3.2.1 Chargement des données

In [8]:
```
def loadFile(filename):
    fi = open(filename, 'rb')
    reader = csv.reader(fi, delimiter=' ')
    data = []
    for row in reader:
        data.append([f for f in map(float, row)])
    return np.array(data)

observations = loadFile('voitureObservations.csv')

plt.figure()
plt.plot(observations[:,0],observations[:,1], 'r+')
```

Out[8]: [<matplotlib.lines.Line2D at 0x1463f240>]



3.2.2 Paramètres

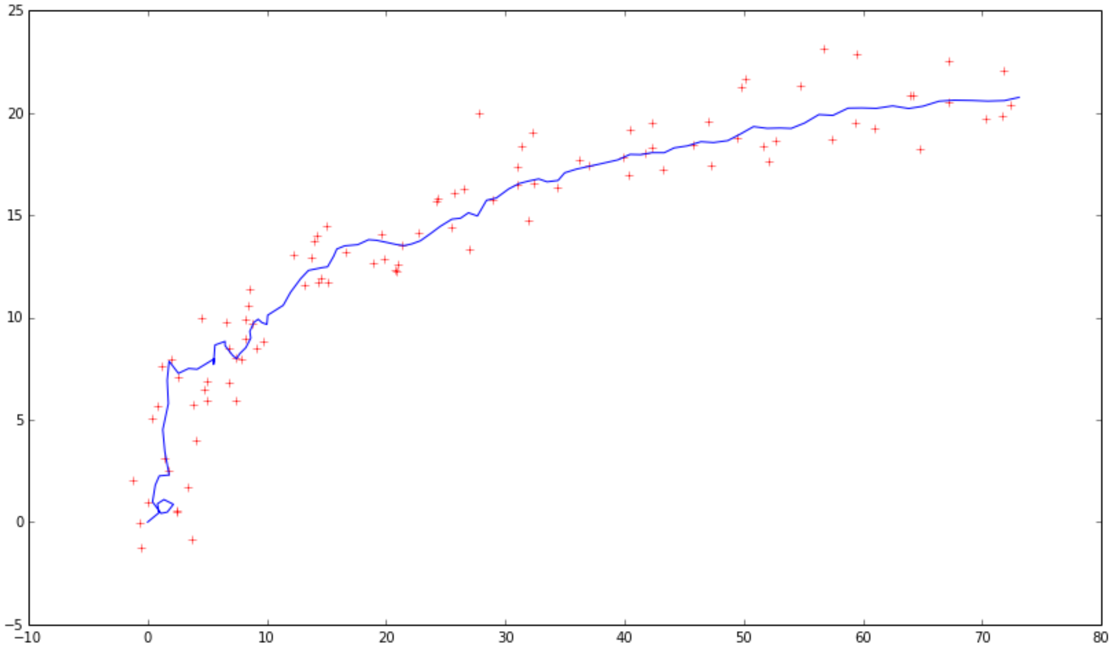
In [9]:
```
osigma = 2;

transition_matrix = np.array([
    [1., 0., .2, 0., 0., 0.],
```

```
        [0., 1., 0., .2, 0., 0.],
        [0., 0., 1., 0., .2, 0.],
        [0., 0., 0., 1., 0., .2],
        [0., 0., 0., 0., 1., 0.],
        [0., 0., 0., 0., 0., 1.]])
transition_covariance = np.zeros((6,6));

observation_matrix = np.array([
        [1., 0., 0., 0., 0., 0.],
        [0., 1., 0., 0., 0., 0.]])
observation_covariance = np.eye(2)*osigma;

initial_state_mean = np.array([0.,0.,0.75,2,0.3,0.1])
initial_state_covariance = np.eye(6);
```

3.2.3 Filtrage à la main

In [10]:
```
kf = KalmanFilter(
    transition_matrix, observation_matrix,
    transition_covariance, observation_covariance,
)

# init
hand_state_estimates = [initial_state_mean]
hand_state_cov_estimates = [initial_state_covariance]

# filtrage
for anObs in observations:
    (aMean, aCov) = kf.filter_update(hand_state_estimates[-1], hand_state_cov_estimates[-1], ar
    hand_state_estimates.append(aMean)
    hand_state_cov_estimates.append(aCov)

hand_state_estimates = np.array(hand_state_estimates)

# Calcul des positions filtrées
hand_positions = np.dot(hand_state_estimates, observation_matrix.T)

# Plot
plt.figure()
plt.plot(observations[:,0],observations[:,1], 'r+')
plt.plot(hand_positions[:,0],hand_positions[:,1], 'b')
```

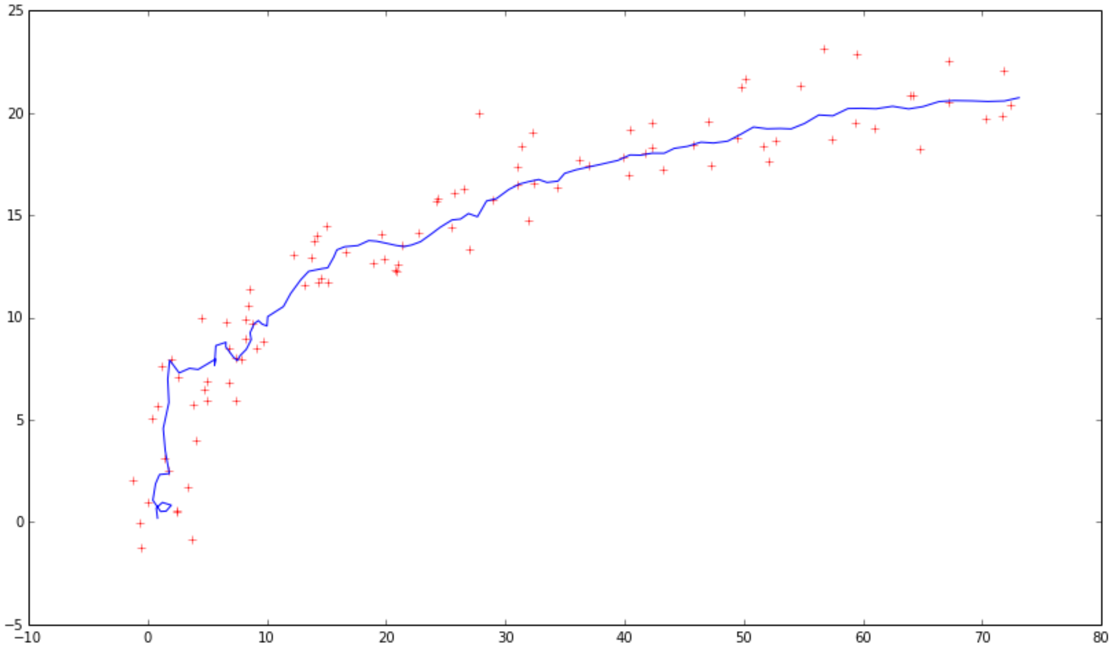
Out[10]:
```
[<matplotlib.lines.Line2D at 0x140b54a8>]
```



3.2.4 Filtrage complet

In [11]:
```
# Init du filtre

kf = KalmanFilter(
    transition_matrix, observation_matrix,
    transition_covariance, observation_covariance,
    initial_state_mean=initial_state_mean, initial_state_covariance = initial_state_covariance
)

# Filtrage des observations
(filtered_state_estimates, filtered_state_cov_estimates) = kf.filter(observations)
filtered_positions = np.dot(filtered_state_estimates, observation_matrix.T)

# Affichage
plt.figure()
plt.plot(observations[:,0],observations[:,1], 'r+')
plt.plot(filtered_positions[:,0],filtered_positions[:,1], 'b')
```

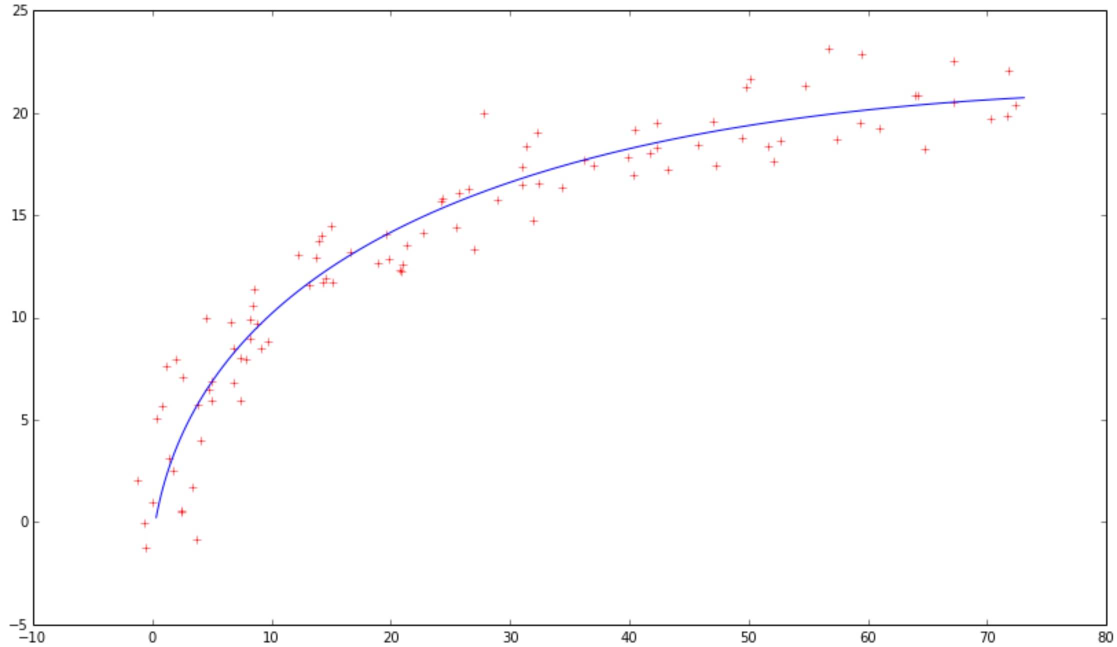
Out[11]:
```
[<matplotlib.lines.Line2D at 0x142db1d0>]
```



3.2.5 Lissage

In [12]:
```
# Init du filtre
kf = KalmanFilter(
    transition_matrix, observation_matrix,
    transition_covariance, observation_covariance,
    initial_state_mean = initial_state_mean, initial_state_covariance = initial_state_covarian
)

# Lissage des observations
(smoothed_state_estimates,smoothed_state_cov_estimates) = kf.smooth(observations)
smoothed_positions = np.dot(smoothed_state_estimates, observation_matrix.T)

plt.figure()
plt.plot(observations[:,0], observations[:,1], 'r+')
plt.plot(smoothed_positions[:,0], smoothed_positions[:,1], 'b')
```

Out[12]:
```
[<matplotlib.lines.Line2D at 0x14a595c0>]
```


25
20
15
10
5
0
−5
−10
0
10
20
30
40
50
60
70
80