Kalman Filter

Progetto di Intelligenza Artificiale e Laboratorio, Parte 3

Emanuele Gentiletti, Alessandro Caputo

Indice

Introduzione

La seconda parte dell'esercitazione relativa al progetto di Uncertainty consiste nell'implementazione di un Kalman Filter, e nell'esecuzione di diverse simulazioni variando i parametri iniziali. In seguito descriviamo l'implementazione da noi scritta, e mostriamo, commentandoli, i dati ottenuti dalle varie prove eseguite.

Prove

Le prove vengono eseguite tramite la funzione `kalman_simulation`, che provvede a eseguire la simulazione secondo diversi parametri:

- Parametri della simulazione:
  - Numero di iterazioni
  - Stato iniziale
  - Covarianza del processo
  - Covarianza dei sensori
- Parametri del Kalman Filter:
  - Stato iniziale stimato
  - Covarianza stimata per lo stato
  - Rumore del processo stimato
  - Covarianza del rumore dei sensori stimata
- Parametri in comune:
  - Accelerazione dell'oggetto
  - Tempo tra gli spostamenti simulati

La funzione esegue la simulazione e restituisce una lista di tuple, una per ogni istante della simulazione. Ogni tupla contiene:

- Stato reale dell'oggetto
- Misurazione fatta sull'oggetto
- Stato, covarianza stato e Kalman Gain del Kalman Filter.

Inoltre, la funzione stampa un grafico che rappresenta gli stati reali e quelli predetti a ogni istante della simulazione.

Stime e covarianze corrette

```
np.random.seed(20190719)
results = kalman_simulation(
    title="Stime e covarianze corrette",
    iterations=30,
    real_state=np.array([0, 2]).reshape(2, 1),
    real_process_cov=np.diag([1000, 2]),
    real_sensor_cov=np.diag([1000, 4]),
    acceleration=1,
)
```

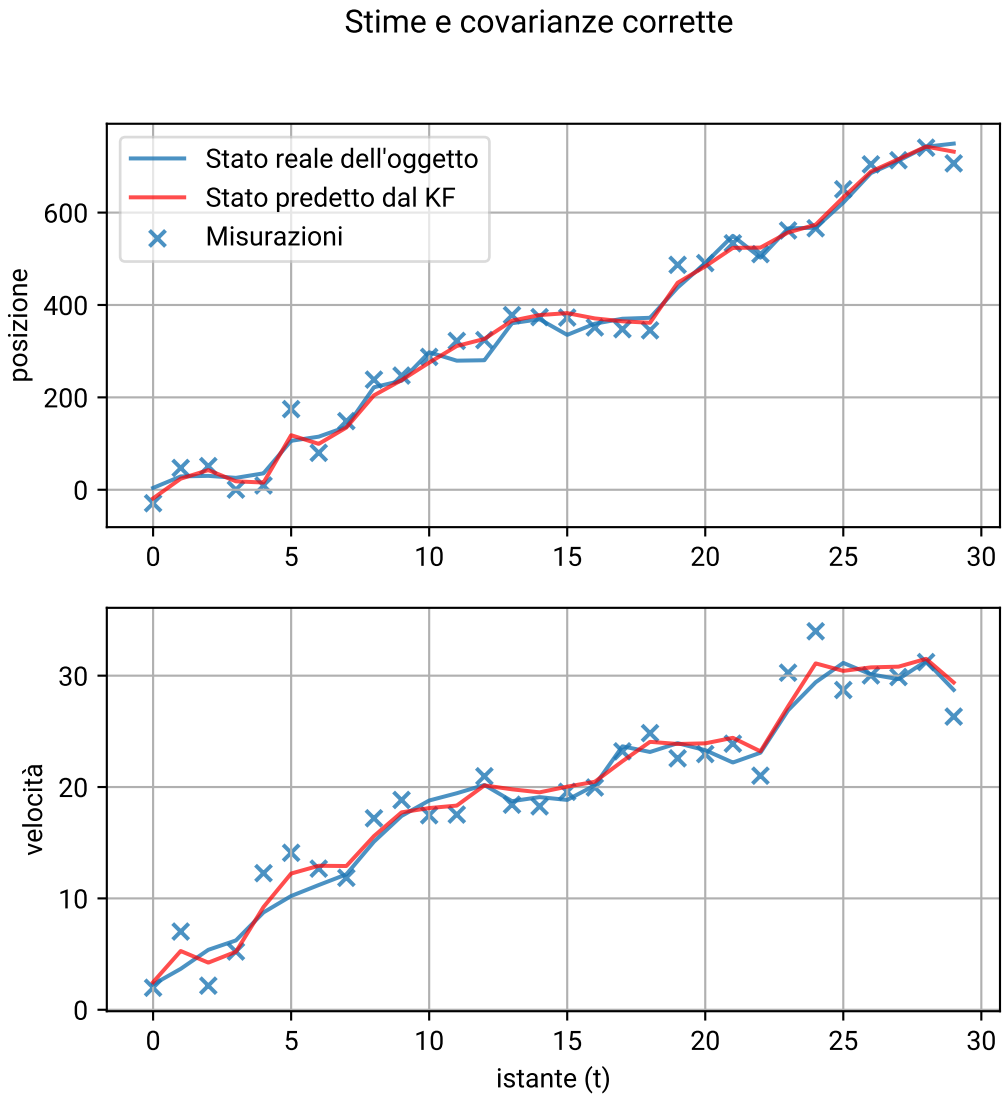

Figura 1: Risultati per il primo esperimento.

Nella prima simulazione, il Kalman Filter riceve i valori giusti riguardanti lo stato e le covarianze dello stato, del processo e dei sensori. Quando la funzione non riceve parametri riguardanti il Kalman Filter, li imposta implicitamente a quelli reali, impostando come covarianza iniziale dello stato la covarianza reale del processo.

Dal grafico, si può vedere come le misurazioni siano più rilevanti nella stima fatta nelle fasi iniziali del processo, specialmente nel caso della velocità: verso la fine, misurazioni molto sbilanciate (ad es. verso l'istante 25) fanno cambiare di poco la stima.

In seguito mostriamo covarianza dello stato e Kalman Gain nell'ultimo istante di simulazione.

```
results[-1].kalman.state_cov
```

```
## array([[6.184e+02, 4.714e-01],
##        [4.714e-01, 1.999e+00]])
```

```
results[-1].kalman.gain
```

```
## array([[6.184e-01, 1.179e-01],
##        [4.714e-04, 4.998e-01]])
```

Stime scorrette, covarianze corrette

Nel secondo esperimento, proviamo a dare al Kalman Filter stime iniziali molto sbagliate rispetto alla posizione reale. Diamo sempre lo stesso seed al generatore di numeri casuali, così che gli stati generati dal simulatore siano gli stessi di prima.

```
np.random.seed(20190719)
results = kalman_simulation(
    title="Stime scorrette, covarianze corrette",
    iterations=30,
    real_state=np.array([0, 2]).reshape(2, 1),
    real_process_cov=np.diag([1000, 2]),
    real_sensor_cov=np.diag([1000, 4]),
    kalman_state=np.array([2000,100]).reshape(2,1),
    acceleration=1,
)
```

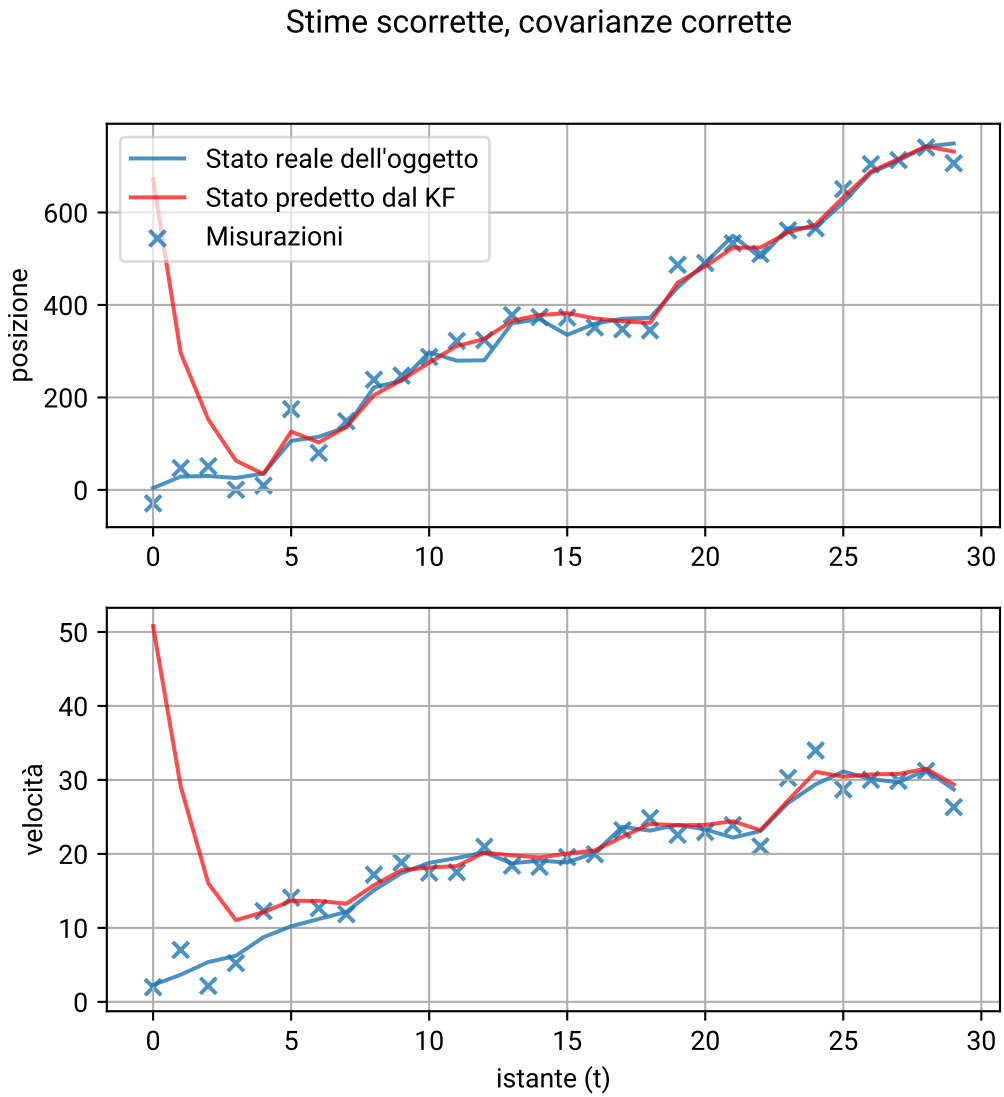

Figura 2: Risultati per il secondo esperimento.

Possiamo osservare come, nonostante ci sia una differenza enorme tra lo stato iniziale del processo e lo stato iniziale del Kalman Filter, questo converga molto presto verso lo stato reale (verso l'istante 8, la stima è già molto vicina). Possiamo verificare come le covarianze e i Kalman Gain cambino dai primi istanti di simulazione rispetto agli ultimi:

```
results[0].kalman.gain
```

```
## array([[6.668e-01, 8.329e-02],
```

```
##         [3.332e-04, 4.999e-01]])
```

```
results[-1].kalman.gain
```

```
## array([[6.184e-01, 1.179e-01],
##         [4.714e-04, 4.998e-01]])
```

```
results[0].kalman.state_cov
```

```
## array([[6.668e+02, 3.332e-01],
##         [3.332e-01, 2.000e+00]])
```

```
results[-1].kalman.state_cov
```

```
## array([[6.184e+02, 4.714e-01],
##         [4.714e-01, 1.999e+00]])
```

Stime scorrette, covarianza dello stato iniziale a 0

Nel prossimo esperimento, vogliamo continuare a dare in input al filtro la stessa stima errata dello stato iniziale, e inoltre impostare a zero la covarianza dello stato iniziale.

```
np.random.seed(20190719)
results = kalman_simulation(
    title="Stime scorrette, covarianza stato iniziale a zero",
    iterations=30,
    real_state=np.array([0, 2]).reshape(2, 1),
    real_process_cov=np.diag([1000, 2]),
    real_sensor_cov=np.diag([1000, 4]),
    kalman_state=np.array([2000,100]).reshape(2,1),
    kalman_state_cov=np.diag([0,0]),
    acceleration=1,
)
```

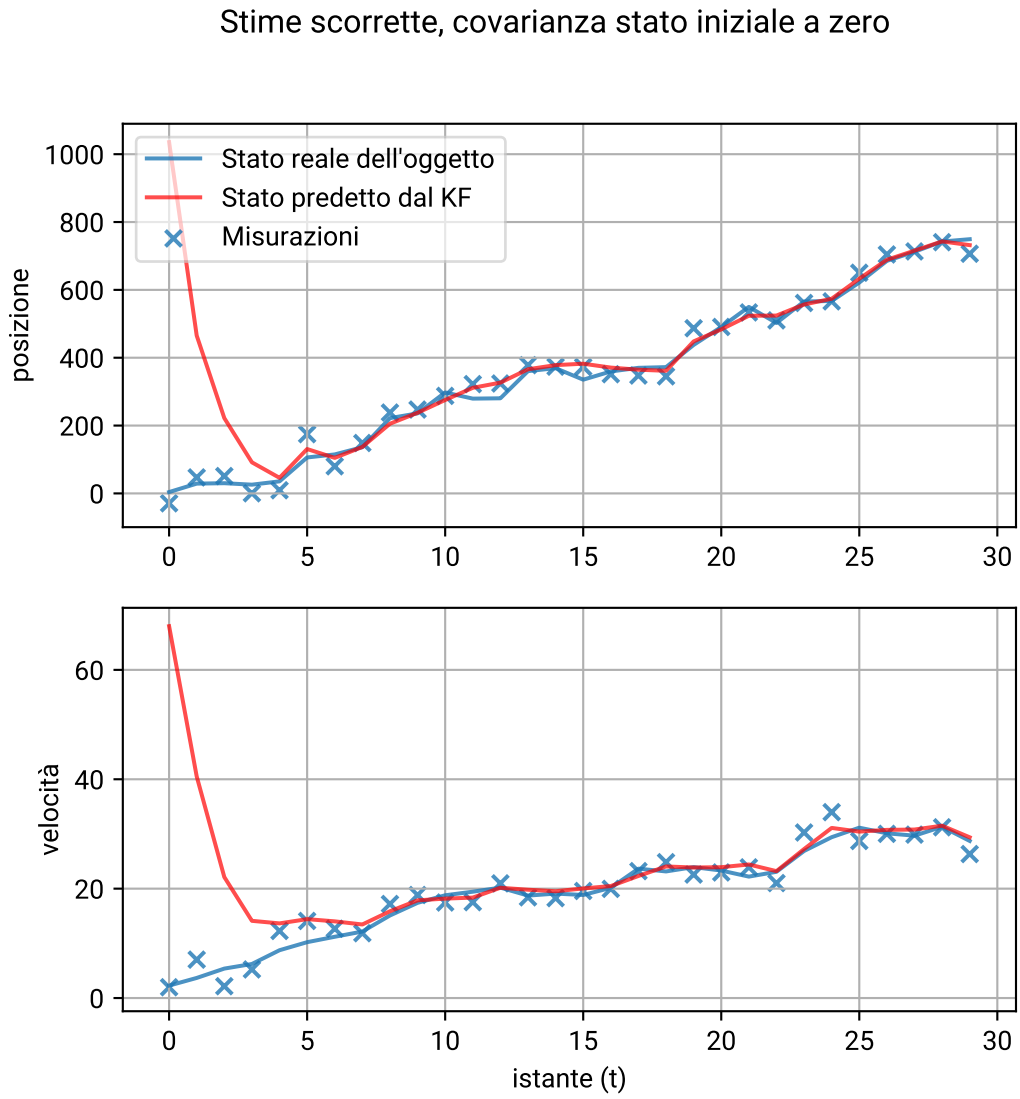

Figura 3: Risultati per il terzo esperimento.

Nonostante la stima iniziale dell'errore sia a zero, il Kalman Filter riesce comunque a convergere verso una stima accettabile più o meno nello stesso tempo. Il comportamento nel secondo e nel terzo esperimento è molto simile, e supponiamo che ciò sia dovuto al fatto che la covarianza del processo esatta data in input al Kalman Filter gli permetta comunque di dare dei pesi adeguati alle misurazioni.

Stime scorrette, covarianza dello stato iniziale a 0 e covarianza processo tendente a 0

In questo esperimento, vogliamo verificare l'ipotesi fatta prima, ovvero che una covarianza di processo ben calibrata permetta al Kalman Filter di dare delle stime corrette anche con stime di errore dello stato iniziale molto sottostimate. Per fare questo, proviamo a sottostimare anche la covarianza di processo nel Kalman Filter, aspettandoci che le predizioni in questo caso siano completamente errate.

```
np.random.seed(20190719)
results = kalman_simulation(
    title="Stime scorrette, covarianza processo vicina a 0",
    iterations=30,
    real_state=np.array([0, 2]).reshape(2, 1),
    real_process_cov=np.diag([1000, 2]),
    real_sensor_cov=np.diag([1000, 4]),
    kalman_state=np.array([2000,100]).reshape(2,1),
    kalman_state_cov=np.diag([0,0]),
    kalman_process_cov=np.diag([0.001,0.001]),
    acceleration=1,
)
```

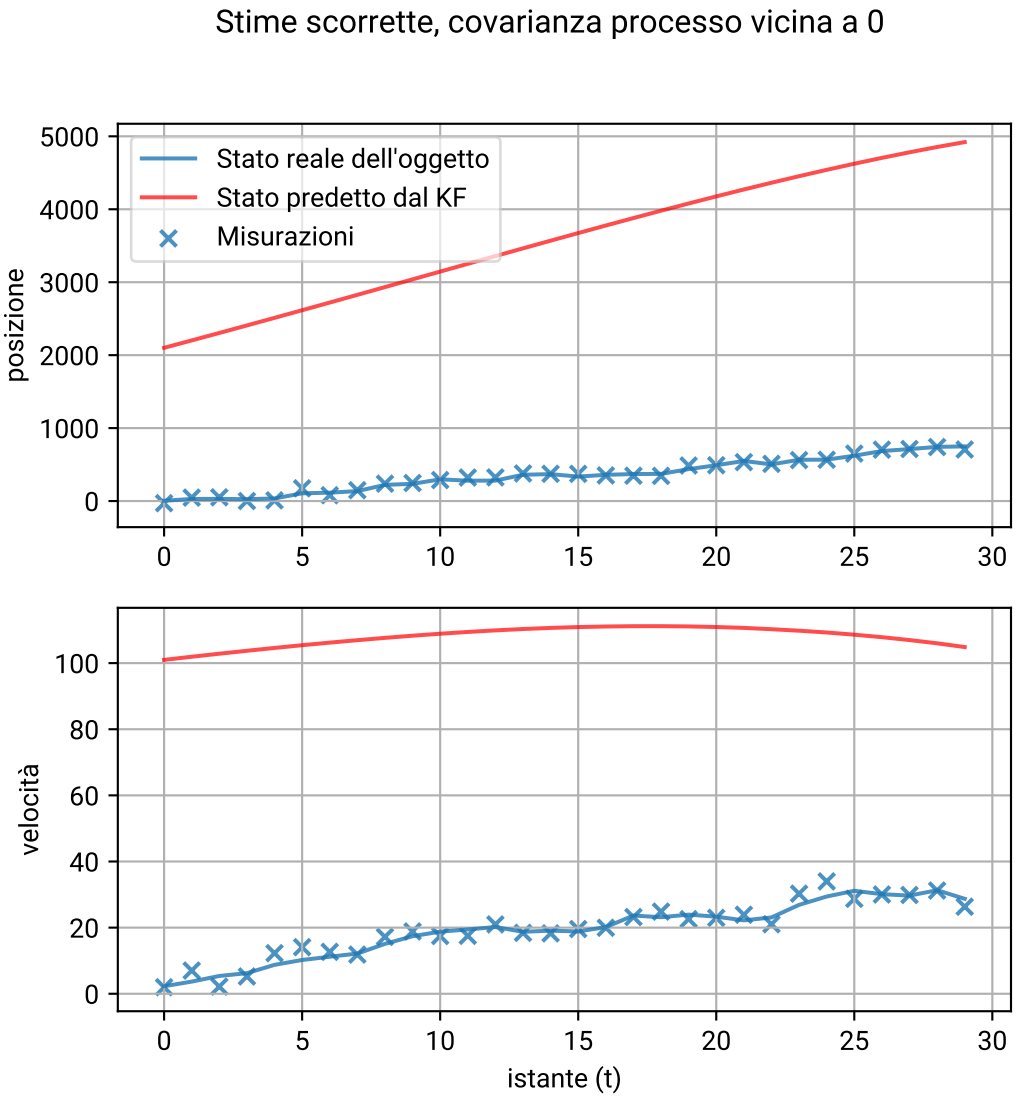

Figura 4: Risultati per il quarto esperimento.

Come ci si può aspettare, impostando la matrice di covarianza del processo verso lo 0, il Kalman Filter non è in grado di stimare correttamente quello che sta accadendo realmente. Possiamo osservare anche come le covarianze e i Kalman Gain siano cambiate nel corso dell'esecuzione rispetto agli esperimenti precedenti.

```
results[0].kalman.gain
```

```
## array([[1.000e-06, 0.000e+00],
```

```
##         [0.000e+00, 2.499e-04]])
```

```
np.identity(2) - results[-1].kalman.gain
```

```
## array([[ 9.926e-01, -9.395e-02],
##         [-3.758e-04,  9.933e-01]])
```

Essendo la covarianza di processo molto bassa, il Kalman Filter pone molto peso sulla stima del processo, e tiene quindi in conto molto poco delle misurazioni fatte. La stima, nel corso del tempo, resta vicina a quella fatta inizialmente, e cambia di troppo poco per avvicinarsi a quella reale.

Stime corrette, cov. sensori reale alta, cov. sensori stimata vicina a 0

In questo esperimento, vogliamo simulare un processo molto deterministico, dove il modello di simulazione del processo darebbe buoni risultati di per sé. Tuttavia, vogliamo che il Kalman Filter dia più peso alle osservazioni, che in questo caso faremo essere molto imprecise. Per fare ciò, diamo una covarianza al rumore dei sensori nel processo reale, e una bassa a quella stimata del Kalman Filter.

```
np.random.seed(19072019)
results = kalman_simulation(
    title="Stime scorrette, covarianza processo vicina a 0",
    iterations=30,
    real_state=np.array([0, 2]).reshape(2, 1),
    real_process_cov=np.diag([1, 1]),
    real_sensor_cov=np.diag([10000, 4000]),
    kalman_sensor_cov=np.diag([0.001,0.001]),
    acceleration=1,
)
```

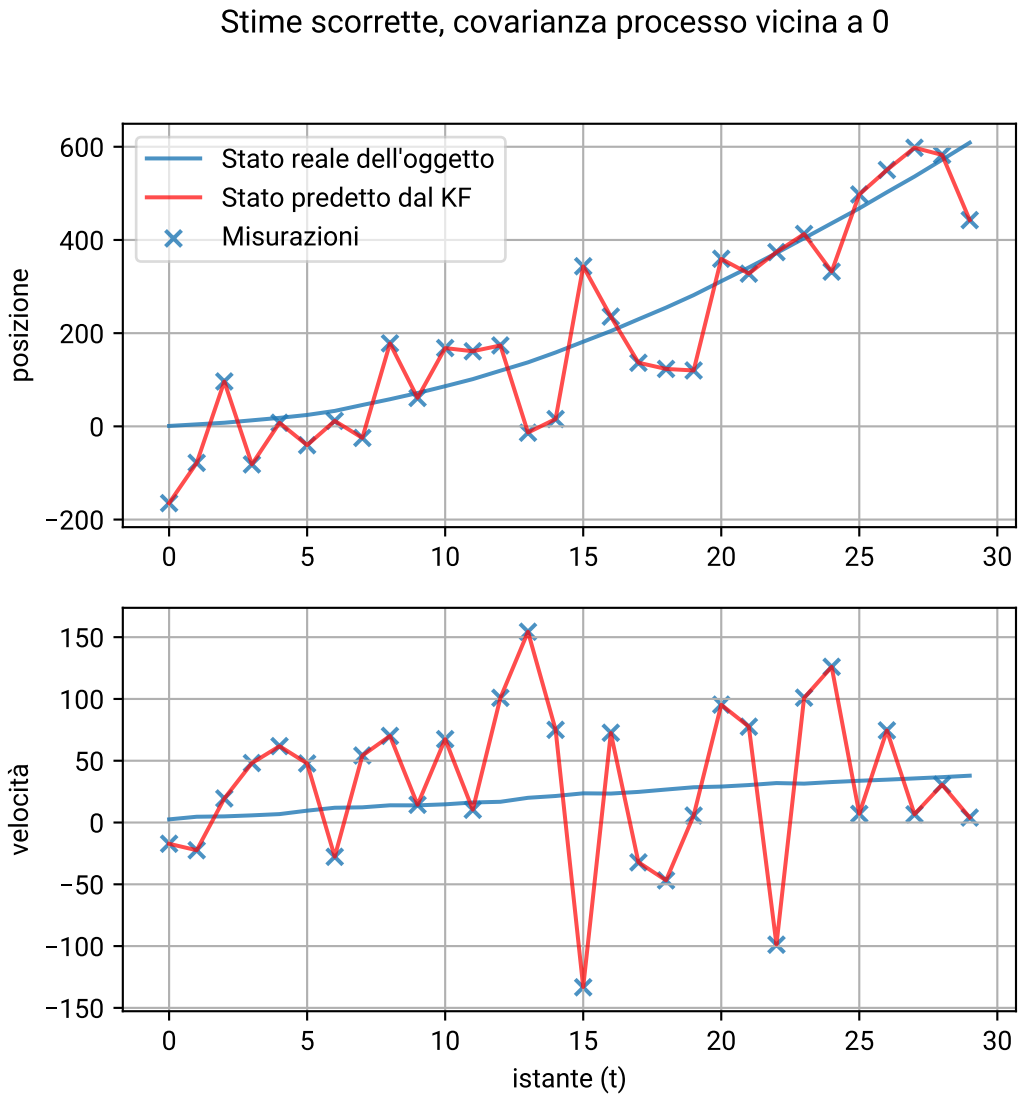

Figura 5: Risultati per il quinto esperimento.

In questo caso, la stima fatta dal Kalman Filter corrisponde quasi esattamente cone le misurazioni dei sensori, come ci aspettavamo.

Stime corrette, cov. sensori reale alta, cov. sensori e processo stimate vicina a 0

In questa prova, replichiamo l'esperimento precedente, dando in input al Kalman Filter covarianze sia dei sensori che del processo vicine allo 0.

```
np.random.seed(19072019)
results = kalman_simulation(
    title="Stime scorrette, covarianza processo vicina a 0",
    iterations=30,
    real_state=np.array([0, 2]).reshape(2, 1),
    real_process_cov=np.diag([1, 1]),
    real_sensor_cov=np.diag([10000, 4000]),
    kalman_process_cov=np.diag([0.001,0.001]),
    kalman_sensor_cov=np.diag([0.001,0.001]),
    acceleration=1,
)
```

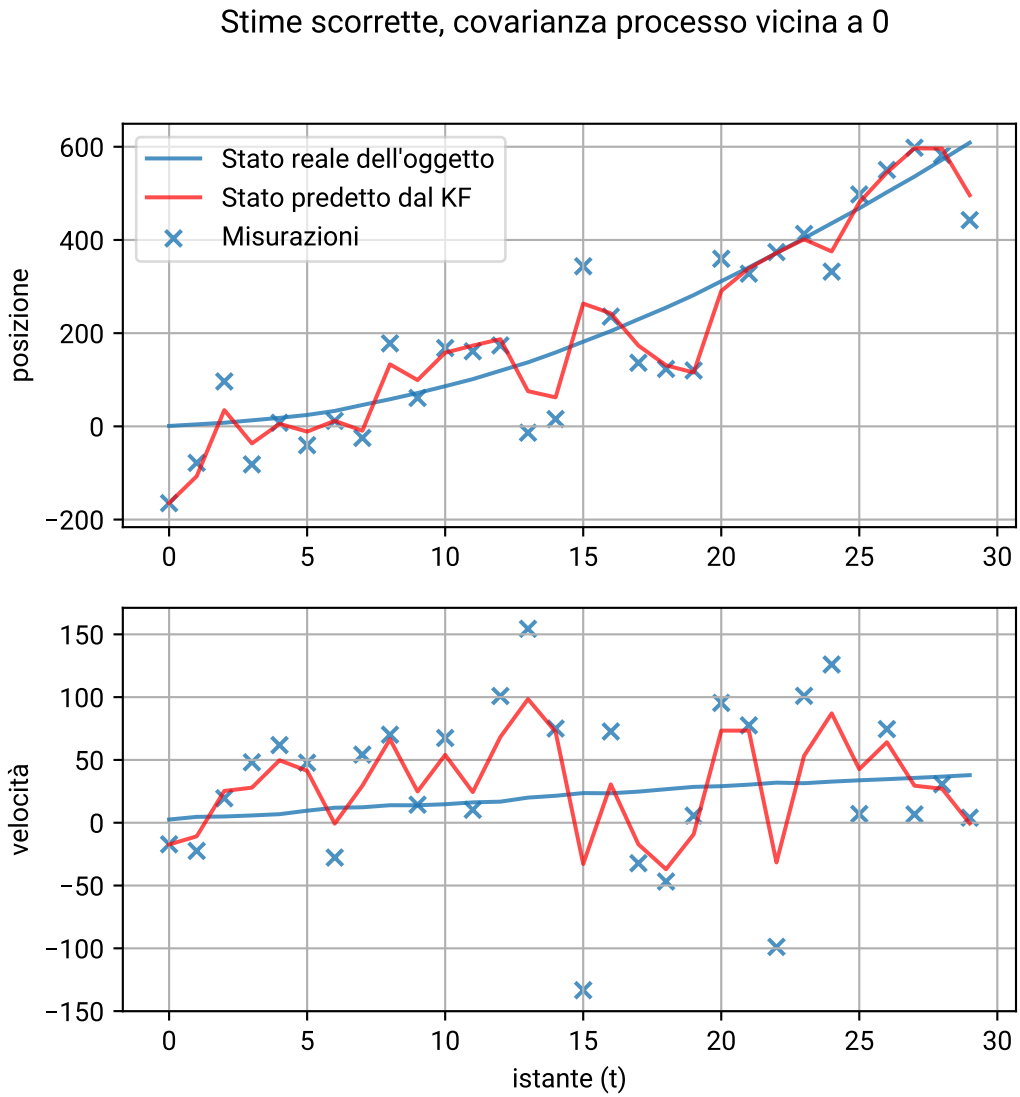

Figura 6: Risultati per il quinto esperimento.

Anche in questo caso, le stime sono molto altalenanti. Quello che si può notare rispetto all'esperimento precedente è che le misurazioni più estreme non trascinano altrettanto la stima rispetto a prima (questo si può osservare in particolare verso l'istante 15 nei due grafici, dove le misurazioni sono lontane dalla media).

```
results[15].kalman.gain
```

```
## array([[0.694, 0.079],
```

```
##         [0.079, 0.594]])
```

```
results[15].kalman.state_cov
```

```
## array([[6.944e-04, 7.932e-05],
##         [7.932e-05, 5.939e-04]])
```