# TRACKING AND PREDICTING TOOL WEAR PROPAGATION USING PARTICLE FILTERING APPROACH


## Abstract

To track and predict cutting tool wear propagation based on sensing measurement, using particle filtering (PF).

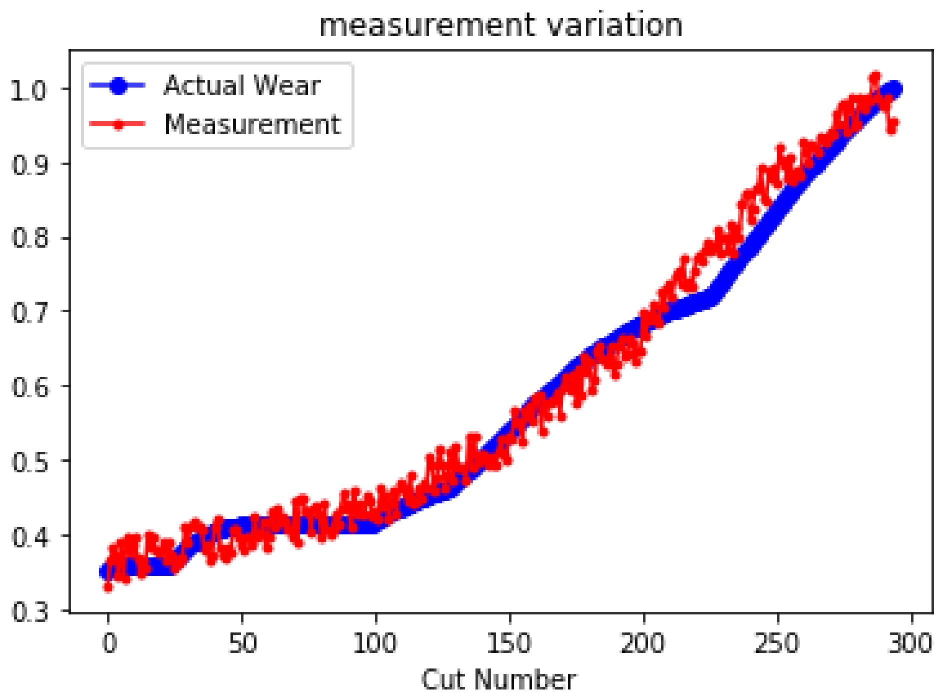


Adeniji, David O.

doad224@uky.edu

## Project 2: Objective

To track and predict cutting tool wear propagation based on sensing measurement, using particle filtering (PF).




Particle filters or Sequential Monte Carlo (SMC) methods are a set of Monte Carlo algorithms used to solve filtering problems arising in signal processing and Bayesian statistical inference. The filtering problem consists of estimating the internal states in dynamical systems when partial observations are made, and random perturbations are present in the sensors as well as in the dynamical system. The objective is to compute the posterior distributions of the states of some Markov process, given some noisy and partial observations.

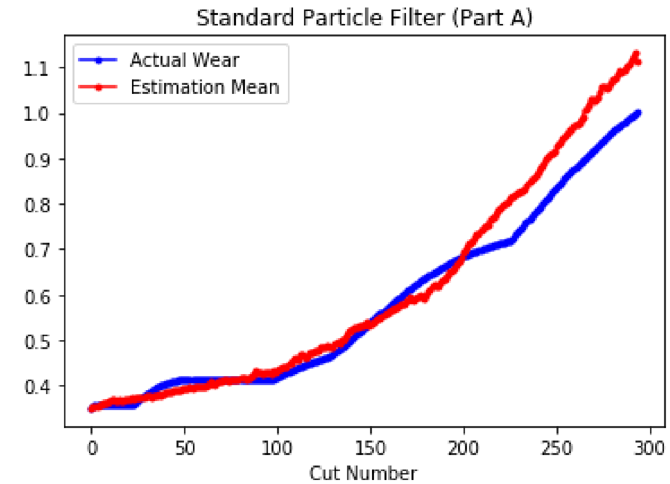
**<u>For this Study:</u>**

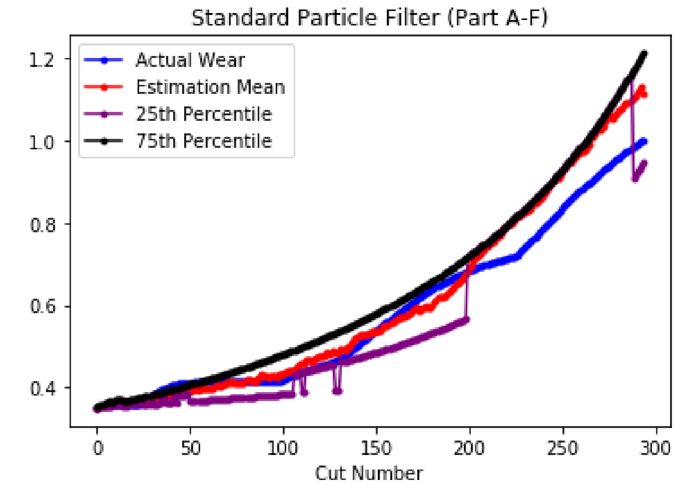
- **Standard Particle State Model:** $x[k] = [k*A*(1-B) + 0.35*(1-B)]^{1/(1-B)}$
- **Advanced Particle State Model:** $x[k] = [k*A*(1-B) + 0.35*(1-B)]^{1/(1-B)}$ + noise
- **Variance of Measurement Model** (y[k]) = 0.01
- **A and B were first sampled from two uniform distributions [0.001 0.1] and [0.1 3]**

# *Standard Particle Filter*

| ***Tool wear propagation tracking over the entire period (Part A)*** | ***Actual Wear and Estimation/Prediction Plots*** | ***Actual Wear and Estimation Plots with Percentiles*** |
|---|---|---|

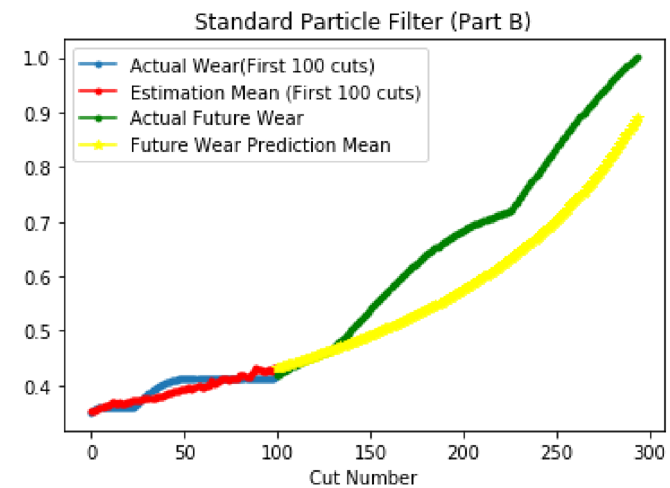
| | |
|---|---|
| ***Track Range*** | Entire Cut Period |
| ***Number of Particle*** | 600 |
| ***R*** | 0.1 |
| ***Tracking RMSE:*** | 0.0532 |

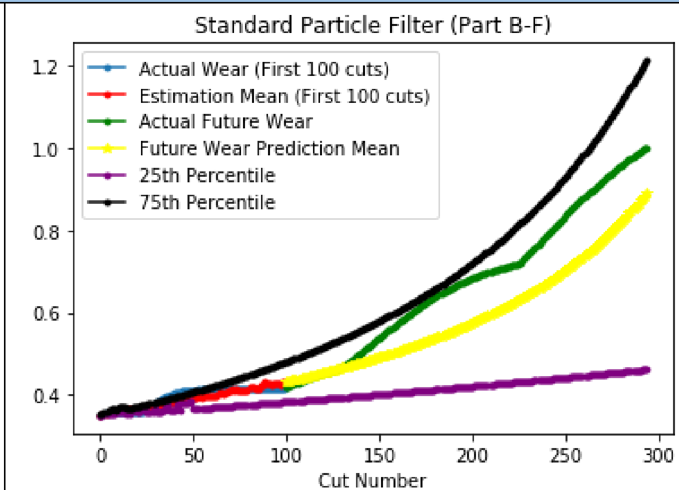


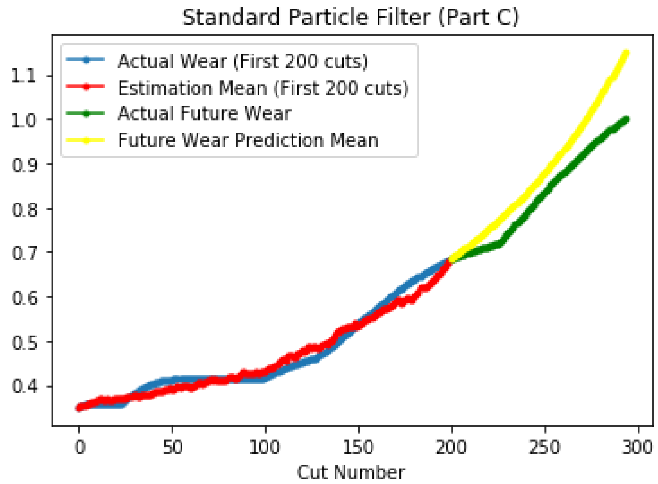
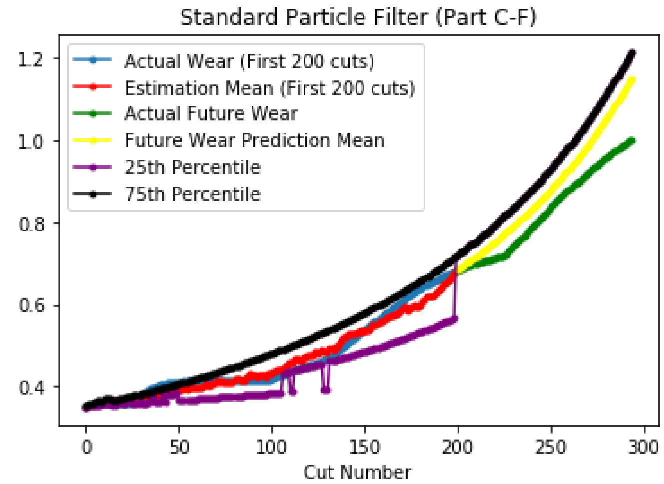




***Tool wear propagation tracking over 100 cuts period (Part B)***

| | |
|---|---|
| ***Track Range*** | First 100 Cuts |
| ***Number of Particle*** | 600 |
| ***R-Value*** | 0.1 |
| ***Tracking RMSE:*** | 0.0126 |
| ***Prediction RMSE:*** | 0.0966 |



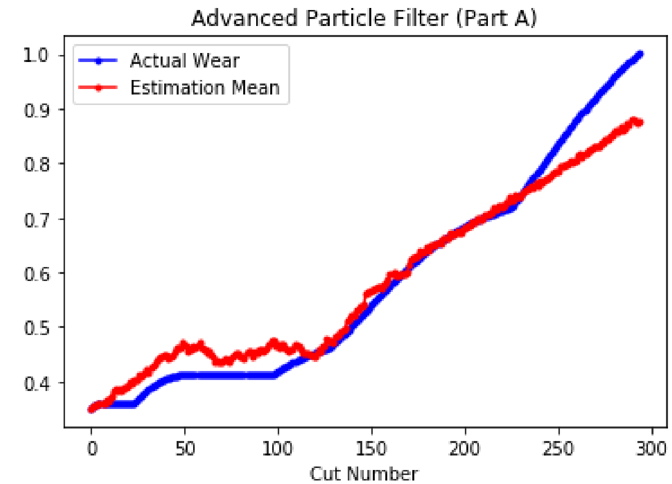



<table>
<tr><th colspan="2">Tool wear propagation tracking over 200 cuts period (Part C)</th><th></th><th></th></tr>
<tr><td>Track Range</td><td>First 200 Cuts</td><td rowspan="5">
</td><td rowspan="5">
</td></tr>
<tr><td>Number of Particle</td><td>600</td></tr>
<tr><td>R</td><td>0.1</td></tr>
<tr><td>Tracking RMSE:</td><td>0.0175</td></tr>
<tr><td>Prediction RMSE:</td><td>0.0650</td></tr>
</table>

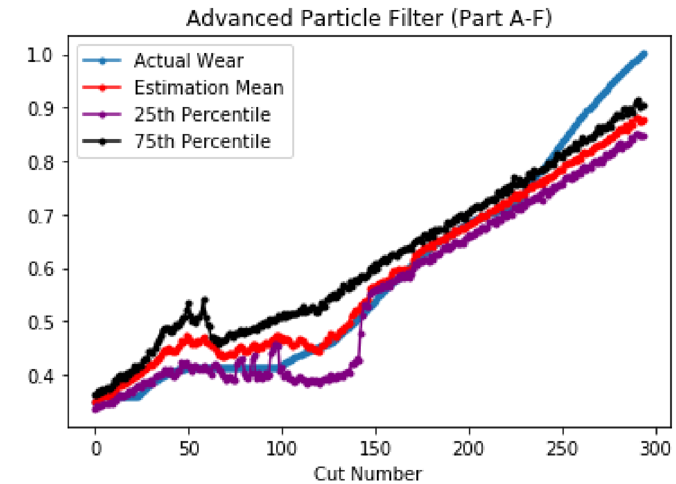
# *Advanced Particle Filter*

| ***Tool wear propagation tracking over the entire period (Part A)*** | |
|---|---|
| ***Track Range*** | Entire Cut Period |
| ***Number of Particle*** | 600 |
| ***R -Value*** | 0.1 |
| ***Tracking RMSE:*** | 0.0438 |

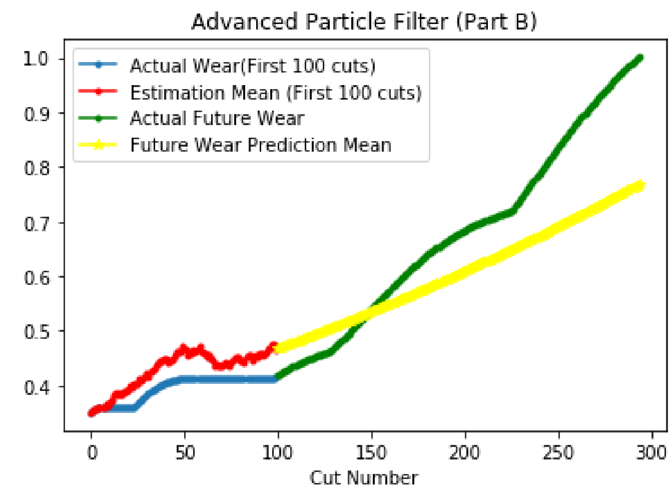
***Actual Wear and Estimation/Prediction Plots***

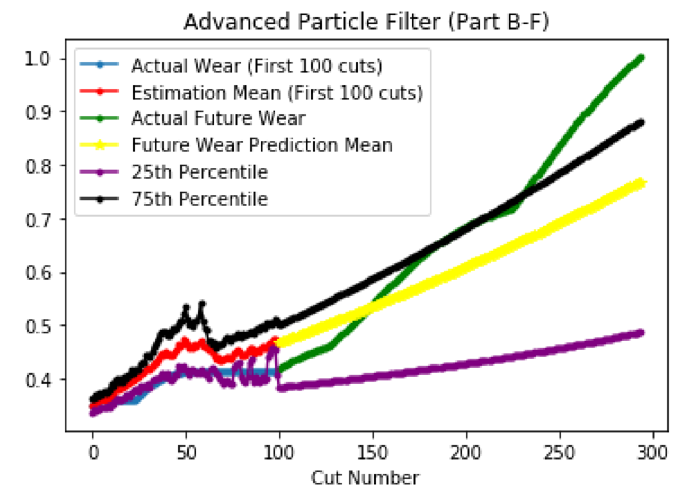


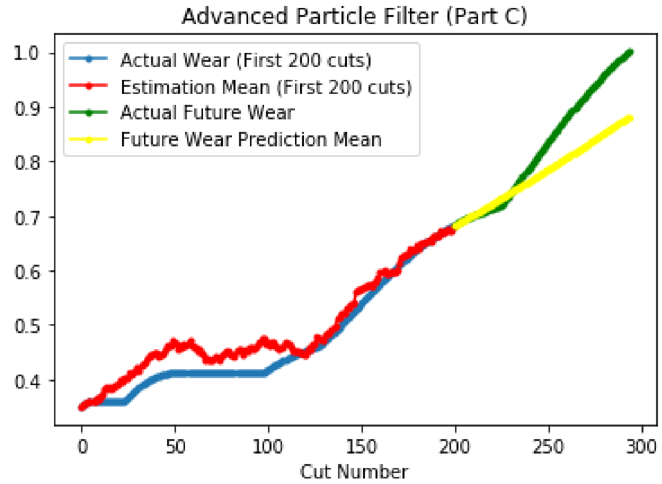

***Actual Wear and Estimation Plots with Percentiles***

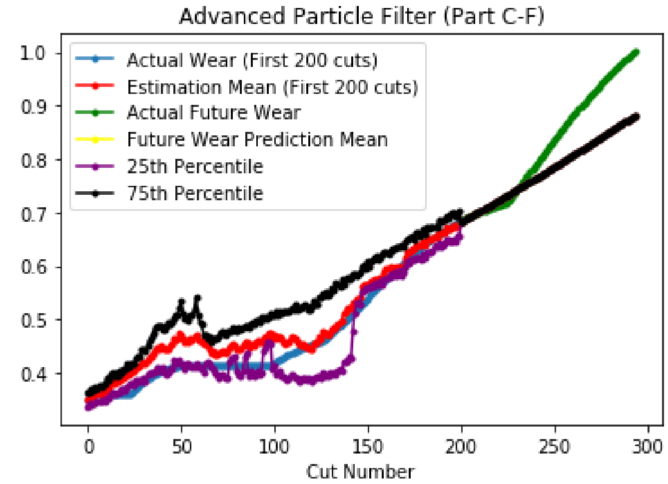



| ***Tool wear propagation tracking over 100 cuts period (Part B)*** | |
|---|---|
| ***Track Range*** | First 100 Cuts |
| ***Number of Particle*** | 600 |
| ***R -Value*** | 0.1 |
| ***Tracking RMSE:*** | 0.0381 |
| ***Prediction RMSE:*** | 0.1096 |



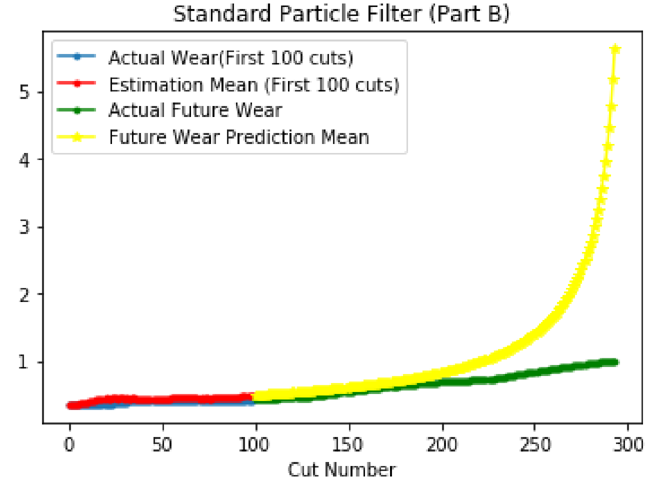
<table>
<tr><th colspan="2">Tool wear propagation tracking over 200 cuts period (Part C)</th><th></th><th></th></tr>
<tr><td>Track Range</td><td>First 200 Cuts</td><td rowspan="5">


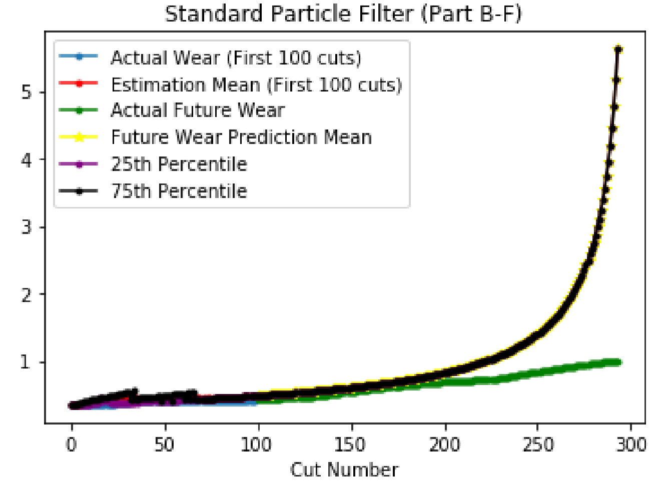
</td><td rowspan="5">


</td></tr>
<tr><td>Number of Particle</td><td>600</td></tr>
<tr><td>R -Value</td><td>0.1</td></tr>
<tr><td>Tracking RMSE:</td><td>0.0294</td></tr>
<tr><td>Prediction RMSE:</td><td>0.0654</td></tr>
</table>

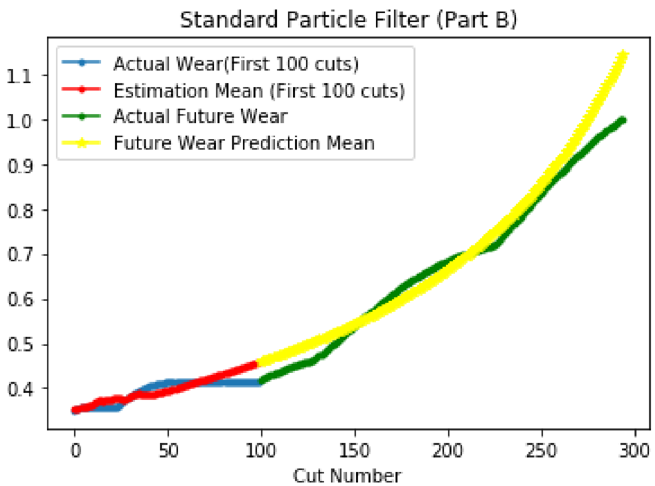
# *Selected Random Trials*

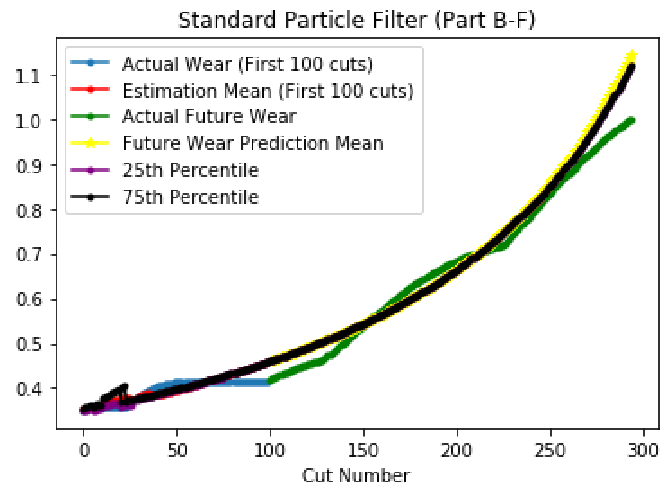
| *Tool wear propagation tracking over first 100* | | *Actual Wear and Estimation/Prediction Plots* | *Actual Wear and Estimation Plots with Percentiles* |
|---|---|---|---|
| ***Track Range*** | 100 cuts | | |
| ***Number of Particle*** | 200 | | |
| ***R-Value*** | 0.05 | | |
| ***Tracking RMSE:*** | 0.0444 | | |
| ***Prediction RMSE:*** | 0.9181 | | |

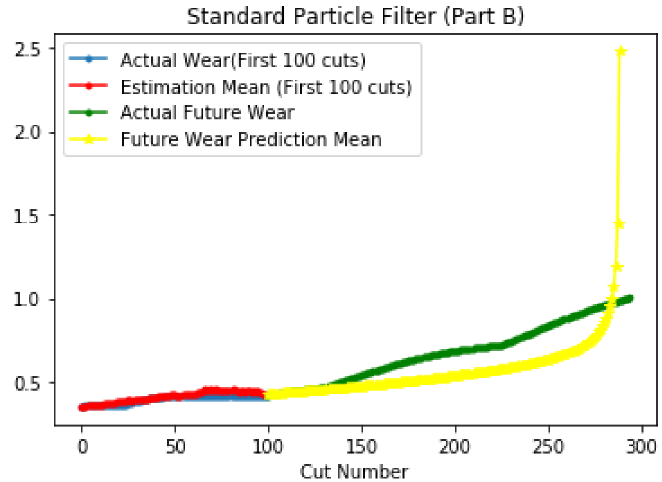
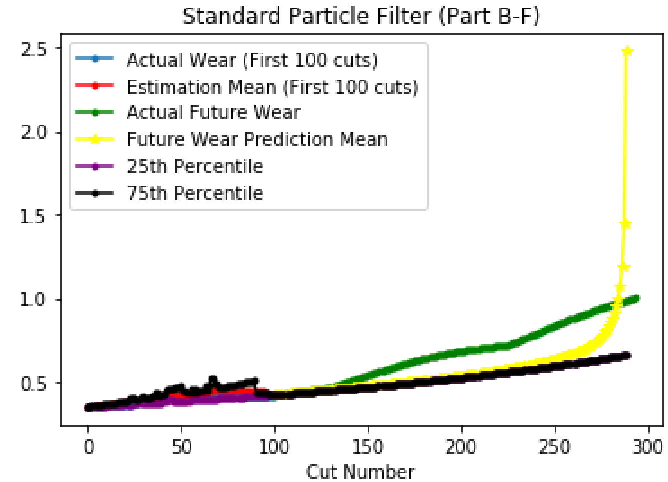



| *Tool wear propagation tracking over 100 cuts period* | |
|---|---|
| ***Track Range*** | First 100 Cuts |
| ***Number of Particle*** | 350 |
| ***R-Value*** | 0.05 |
| ***Tracking RMSE:*** | 0.0182 |
| ***Prediction RMSE:*** | 0.0409 |



<table>
<tr><th colspan="2">Tool wear propagation tracking over 100 cuts period</th><th></th><th></th></tr>
<tr><td>Track Range</td><td>100 Cuts</td><td rowspan="5">
</td><td rowspan="5">
</td></tr>
<tr><td>Number of Particle</td><td>1000</td></tr>
<tr><td>R-Value</td><td>0.05</td></tr>
<tr><td>Tracking RMSE:</td><td>0.02078</td></tr>
<tr><td>Prediction RMSE:</td><td>NAN</td></tr>
</table>

## *Part E: Analysis and Findings*

With 600 particles and a R value of 0.1, the comparison between the standard and advanced particle filter shows a negligible difference in performance as shown in the table above.

- The result is highly dependent on the R value and the random number seed. Beyond an R value of 0.3, the program does not seem to perform accurately
- The prediction RMSE significantly reduces when the tracking range was increased from the first 100 cuts to the first 200 cuts.
- The change in tracking RMSE is not proportional to the number of cuts tracked, but more change is due to the actual wear profile and estimation accuracy. This is proven by the contrasting trend between standard particle filter's first 100 and 200 tracking RMSE compared to that of advanced particle filter.
- The tracking range significantly influence the prediction capability. If the measurement was only tracked for the first 100 cuts the prediction capability will be lower compared to if it was tracked for the first 200 cuts.
- The tracking particles do not seem to update at some point
- The number of particles used significantly influence the computation time. The greater the number of particles, longer the computational time.

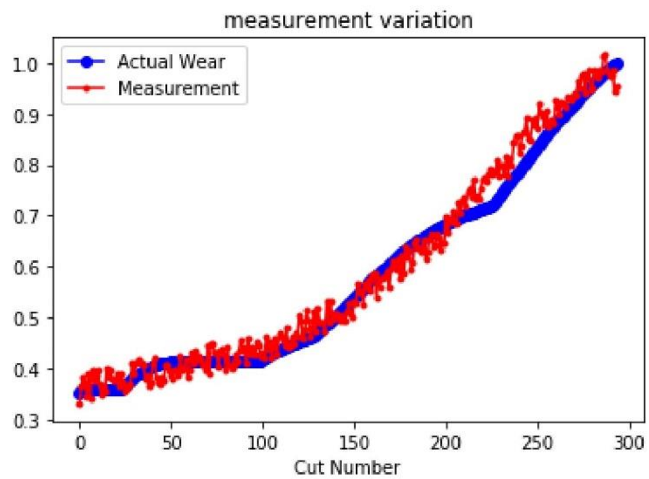
Reducing the number of particles reduces the computational time, however it also reduces the predictive capabilities of the algorithm, a high number of particles such as 1000, significantly affects the algorithm as shown in the selected trial table. An interpolation error occurred and the prediction RMSE is NAN with a poor follow trend on the plot. The predictive capability of the particle filtering algorithm is quite good for a statistical approach.

## *Part E: Difficulties*

- The major challenges was to properly select the R-values, adding the noise and setting the resampling properly, aside these class explanations and guidance were sufficient.

In [90]:
```
import numpy as np
import matplotlib.pyplot as plt
import scipy.io as sio
np.random.seed(0)
```

In [91]:
```
#load data
mat_content = sio.loadmat('Tool Wear.mat') #using pandas to load the excel
 file
x= mat_content['meas']
y = mat_content['wear']
y = y.transpose()
#R value vbased on covariance of data,try 0.01
R = 0.1
plt.plot(np.arange(294), y,'b', marker ='o', label = 'Actual Wear')
plt.plot(np.arange(294), x, 'r', marker='.', label = 'Measurement')
plt.title('system state variation')
plt.title('measurement variation')
plt.xlabel('Cut Number')
leg = plt.legend();
```




In [92]:
```
# particle filter
def PF(y, N_particle, R):
    x_estimate = np.zeros([len(y),N_particle])
    y_estimate = np.zeros([len(y),N_particle])

    P_est = np.zeros([2,N_particle])
    P_est[0,:] = np.random.uniform(0.001, 0.01, N_particle)
    P_est[1,:] = np.random.uniform(0.1, 3.0, N_particle)

    w = np.zeros(N_particle)
    for i in range(len(y)):
        #prediction, include this line only for c, not a and b
        x_estimate[i] = ((i+1)*P_est[0,:]*(1-P_est[1,:])+0.35**(1-P_est[1,
:]))**(1/(1-P_est[1,:]))
        #x_estimate[i] = np.random.normal(x_estimate[i], 0.05*np.absolute(
x_estimate[i]),N_particle) #resample, adding noise resampled particle
```

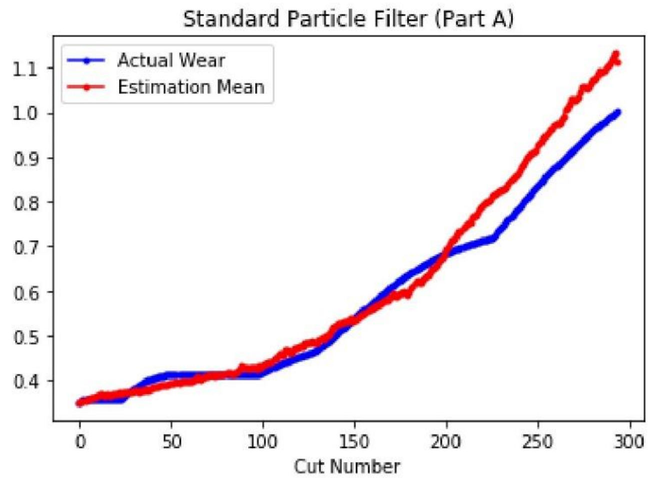
```
        #can also add noise to parameters so too if it affects performance
, try without noise to see effects of it.

        y_estimate[i,:] = x_estimate[i,:]

        #calculate weights
        w = 1/(np.sqrt(2*np.pi)*np.sqrt(R)) * np.exp(-1*(y[i]-y_estimate[i
])**2/(2*R))

        #normalize weights
        w_sum = np.sum(w)
        w = w/w_sum

        #particle resampling
        for j in range(N_particle):
            rand = np.random.rand(1)
            w_c = 0
            for k in range(N_particle):
                w_c += w[k]
                if w_c >= rand:
                    x_estimate[i,j] = x_estimate[i,k]
                    P_est[:,j] = P_est[:,k]
                    break
    return x_estimate, P_est
```

# Standard Particle Filter Part A

In [93]:
```
#tracking by particle filter
N_particle = 600
x_estimate, P_est = PF(x[:294,:], N_particle, R)#calculate 0-100 and predi
ct the remaining for a and b
x_estimate_mean = np.mean(x_estimate,1)
```

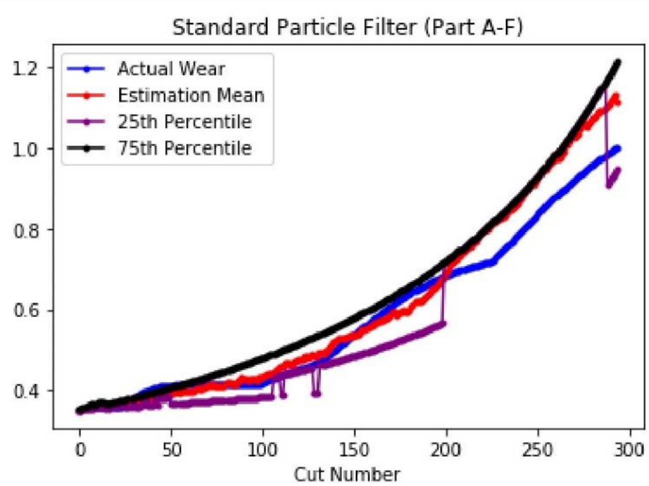
## Plot Without Percentiles

In [94]:
```
plt.title('Standard Particle Filter (Part A)')
plt.plot(np.arange(294), y[:294],'b', marker='.', label= 'Actual Wear' )
plt.plot(np.arange(294), x_estimate_mean, 'r',marker='.', label = 'Estimat
ion Mean')
plt.xlabel('Cut Number')
leg = plt.legend();
```




# Plot With Percentile

In [95]:
```
plt.title('Standard Particle Filter (Part A-F)')
plt.plot(np.arange(294), y[:294], 'b',marker='.', label= 'Actual Wear' )
plt.plot(np.arange(294), x_estimate_mean, 'r',marker='.', label = 'Estimat
ion Mean')
p1 = np.percentile(x_estimate,25,axis=1)
p2 = np.percentile(x_estimate,75,axis=1)
plt.plot(p1, 'purple',marker='.', label = '25th Percentile')
plt.plot(p2, 'black',marker='.', label = '75th Percentile')
plt.xlabel('Cut Number')
leg = plt.legend();
```




In [96]:
```
def RMSE(predictions, targets):
    return np.sqrt(np.mean((predictions-targets)**2))
print(RMSE(y[:294].squeeze(), x_estimate_mean))
```

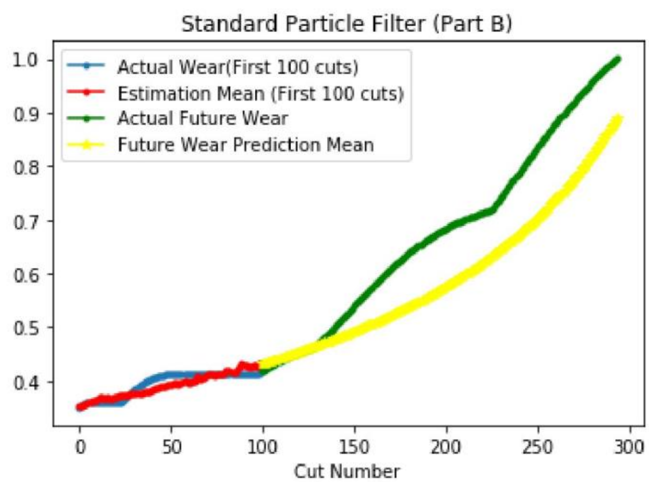
```
0.05320739971808429
```

# Standard Particle Filter Part B

In [78]:
```
#tracking by particle filter
N_particle = 600
x_estimate, P_est = PF(x[:100,:], N_particle, R)#calculate 0-100 and predi
ct the remaining for a and b
x_estimate_mean = np.mean(x_estimate,1)
x_estimate_predict = np.zeros([194,N_particle])
for i in range(294-100):
    #prediction
    x_estimate_predict[i] = ((100+i+1)*P_est[0,:]*(1-P_est[1,:])+0.35**(1-
P_est[1,:]))**(1/(1-P_est[1,:]))
    x_estimate_predict_mean = np.mean(x_estimate_predict,1)
```

## Plot Without Percentiles

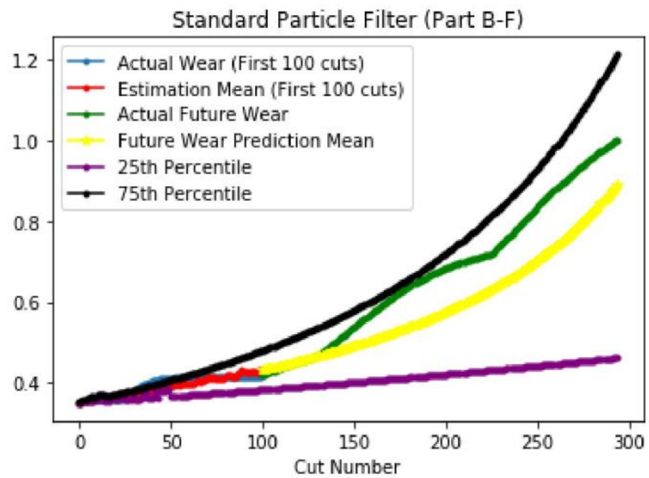
In [79]:
```
plt.title('Standard Particle Filter (Part B)')
plt.plot(np.arange(100), y[:100], marker='.', label= 'Actual Wear(First 10
0 cuts)')
plt.plot(np.arange(100), x_estimate_mean,'r',marker='.', label = 'Estimati
on Mean (First 100 cuts)')
plt.plot(np.linspace(100,293,194).transpose(), y[100:],'green', marker='.'
, label = 'Actual Future Wear')
plt.plot(np.linspace(100,293,194), x_estimate_predict_mean, 'yellow',marke
r='*', label = 'Future Wear Prediction Mean')
plt.xlabel('Cut Number')
leg = plt.legend();
```




## Plot With Percentiles

In [80]:
```
plt.title('Standard Particle Filter (Part B-F)')
p1 = np.percentile(x_estimate,25,axis=1)
p1f = np.percentile(x_estimate_predict,25,axis=1)
p1 = np.concatenate((p1, p1f), axis=0)
p2 = np.percentile(x_estimate,75,axis=1)
```

```
p2f = np.percentile(x_estimate_predict,75,axis=1)
p2 = np.concatenate((p2, p2f), axis=0)
plt.plot(np.arange(100), y[:100], marker='.', label= 'Actual Wear (First 1
00 cuts)')
plt.plot(np.arange(100), x_estimate_mean,'r',marker='.', label = 'Estimati
on Mean (First 100 cuts)')
plt.plot(np.linspace(100,293,194).transpose(), y[100:],'green', marker='.'
, label = 'Actual Future Wear')
plt.plot(np.linspace(100,293,194), x_estimate_predict_mean, 'yellow',marke
r='*', label = 'Future Wear Prediction Mean')
plt.plot(p1, 'purple',marker='.', label = '25th Percentile')
plt.plot(p2, 'black',marker='.', label = '75th Percentile')
plt.xlabel('Cut Number')
leg = plt.legend();
```




In [81]:

```
def RMSE(predictions, targets):
    return np.sqrt(np.mean((predictions-targets)**2))

print(RMSE(y[:100].squeeze(), x_estimate_mean))

RMSE(y[100:].squeeze(), x_estimate_predict_mean)
```

```
0.012581800047884124
```

Out[81]:

```
0.09662982569724343
```

# Standard Particle Filter Part C

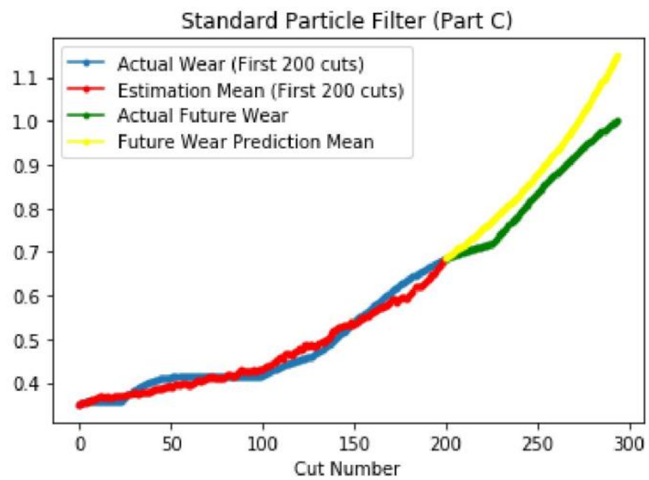
In [85]:

```
#tracking by particle filter
N_particle = 600
x_estimate, P_est = PF(x[:200,:], N_particle, R)#calculate 0-100 and predi
ct the remaining for a and b
x_estimate_mean = np.mean(x_estimate,1)
x_estimate_predict = np.zeros([94,N_particle])
for i in range(294-200):
    #prediction
    x_estimate_predict[i] = ((200+i+1)*P_est[0,:]*(1-P_est[1,:])+0.35**(1-
P_est[1,:]))**(1/(1-P_est[1,:]))
```

```
    x_estimate_predict_mean = np.mean(x_estimate_predict,1)
```

## Plot Without Percentiles

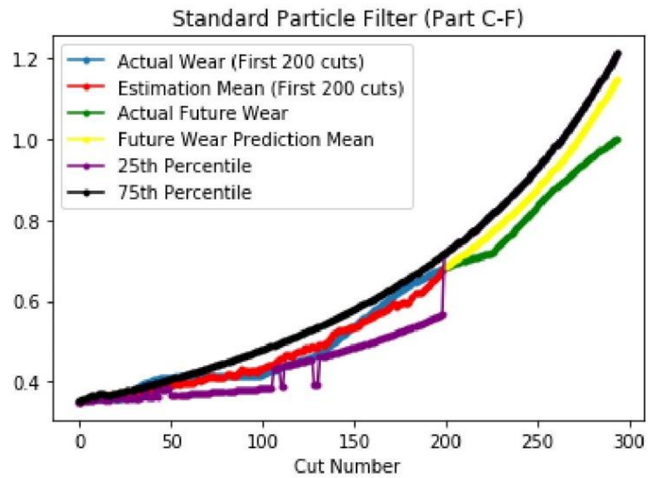
In [86]:
```
plt.title('Standard Particle Filter (Part C)')
plt.plot(np.arange(200), y[:200], marker='.', label= 'Actual Wear (First 2
00 cuts)')
plt.plot(np.arange(200), x_estimate_mean,'r',marker='.', label = 'Estimati
on Mean (First 200 cuts)')
plt.plot(np.linspace(200,293,94).transpose(), y[200:],'green', marker='.',
 label = 'Actual Future Wear')
plt.plot(np.linspace(200,293,94), x_estimate_predict_mean, 'yellow',marker
='.', label = 'Future Wear Prediction Mean')
plt.xlabel('Cut Number')
leg = plt.legend();
```




## Plot With Percentiles

In [87]:
```
plt.title('Standard Particle Filter (Part C-F)')
p1 = np.percentile(x_estimate,25,axis=1)
p1f = np.percentile(x_estimate_predict,25,axis=1)
p1 = np.concatenate((p1, p1f), axis=0)
p2 = np.percentile(x_estimate,75,axis=1)
p2f = np.percentile(x_estimate_predict,75,axis=1)
p2 = np.concatenate((p2, p2f), axis=0)
plt.plot(np.arange(200), y[:200], marker='.', label= 'Actual Wear (First 2
00 cuts)')
plt.plot(np.arange(200), x_estimate_mean,'r',marker='.', label = 'Estimati
on Mean (First 200 cuts)')
plt.plot(np.linspace(200,293,94).transpose(), y[200:],'green', marker='.',
 label = 'Actual Future Wear')
plt.plot(np.linspace(200,293,94), x_estimate_predict_mean, 'yellow',marker
='.', label = 'Future Wear Prediction Mean')
plt.plot(p1, 'purple',marker='.', label = '25th Percentile')
plt.plot(p2, 'black',marker='.', label = '75th Percentile')
plt.xlabel('Cut Number')
```

```
leg = plt.legend();
```



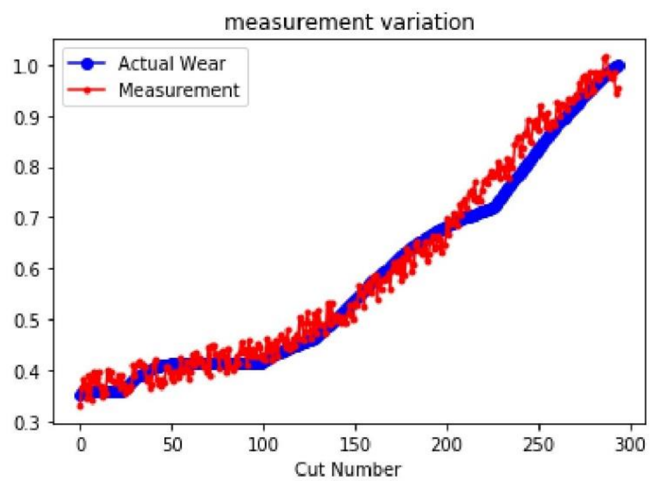

In [89]:
```
def RMSE(predictions, targets):
    return np.sqrt(np.mean((predictions-targets)**2))

print(RMSE(y[:200].squeeze(), x_estimate_mean))

RMSE(y[200:].squeeze(), x_estimate_predict_mean)
```

```
0.017546484224215578
```

Out[89]:
```
0.06495817887853901
```

In [217]:
```
import numpy as np
import matplotlib.pyplot as plt
import scipy.io as sio
np.random.seed(5)
```

In [218]:
```
#Load data
mat_content = sio.loadmat('Tool Wear.mat') #using pandas to load the excel fil
e
x= mat_content['meas']
y = mat_content['wear']
y = y.transpose()
#R value vbased on covariance of data,try 0.01
R = 0.1
plt.plot(np.arange(294), y,'b', marker ='o', label = 'Actual Wear')
plt.plot(np.arange(294), x, 'r', marker='.', label = 'Measurement')
plt.title('system state variation')
plt.title('measurement variation')
plt.xlabel('Cut Number')
leg = plt.legend();
```



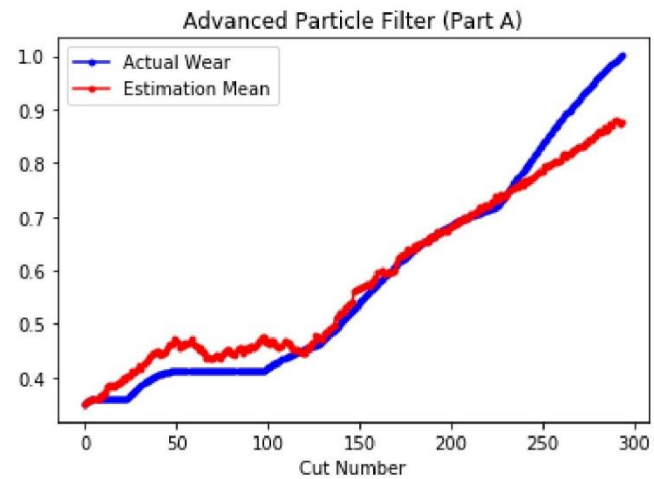
In [219]:
```
# particle filter
def PF(y, N_particle, R):
    x_estimate = np.zeros([len(y),N_particle])
    y_estimate = np.zeros([len(y),N_particle])

    P_est = np.zeros([2,N_particle])
    P_est[0,:] = np.random.uniform(0.001, 0.01, N_particle)
    P_est[1,:] = np.random.uniform(0.1, 3.0, N_particle)

    w = np.zeros(N_particle)
    for i in range(len(y)):
        #prediction, include this line only for c, not a and b
        x_estimate[i] = ((i+1)*P_est[0,:]*(1-P_est[1,:])+0.35**(1-P_est[1,:]))
**(1/(1-P_est[1,:]))
        x_estimate[i] = np.random.normal(x_estimate[i], 0.05*np.absolute(x_est
imate[i]),N_particle) #resample, adding noise resampled particle

        #can also add noise to parameters so too if it affects performance, tr
y without noise to see effects of it.
        y_estimate[i,:] = x_estimate[i,:]
        #calculate weights
        w = 1/(np.sqrt(2*np.pi)*np.sqrt(R)) * np.exp(-1*(y[i]-y_estimate[i])**
2/(2*R))
        #normalize weights
        w_sum = np.sum(w)
        w = w/w_sum
        #particle resampling
        for j in range(N_particle):
            rand = np.random.rand(1)
            w_c = 0
            for k in range(N_particle):
                w_c += w[k]
                if w_c >= rand:
                    x_estimate[i,j] = x_estimate[i,k]
                    P_est[:,j] = P_est[:,k]
                    break
    return x_estimate, P_est
```

# Advanced Particle Filter Part A

In [209]:
```
#tracking by particle filter
N_particle = 600
x_estimate, P_est = PF(x[:294,:], N_particle, R)#calculate 0-100 and predict t
he remaining for a and b
x_estimate_mean = np.mean(x_estimate,1)
```

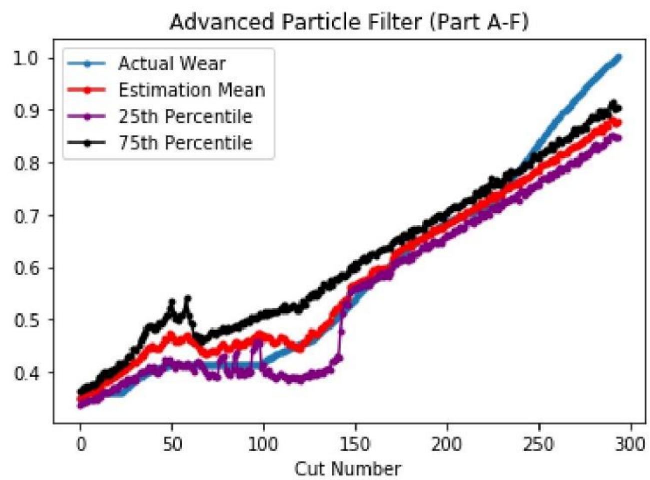
## Plot without Percentiles

In [202]:
```
plt.title('Advanced Particle Filter (Part A)')
plt.plot(np.arange(294), y[:294], 'b', marker='.', label= 'Actual Wear' )
plt.plot(np.arange(294), x_estimate_mean, 'r',marker='.', label = 'Estimation
 Mean')
plt.xlabel('Cut Number')
leg = plt.legend();
```




## Plot with Percentiles

In [203]:
```
plt.title('Advanced Particle Filter (Part A-F)')
plt.plot(np.arange(294), y[:294], marker='.', label= 'Actual Wear' )
plt.plot(np.arange(294), x_estimate_mean, 'r',marker='.', label = 'Estimation
 Mean')
p1 = np.percentile(x_estimate,25,axis=1)
p2 = np.percentile(x_estimate,75,axis=1)
plt.plot(p1, 'purple',marker='.', label = '25th Percentile')
plt.plot(p2, 'black',marker='.', label = '75th Percentile')
plt.xlabel('Cut Number')
leg = plt.legend();
```




In [205]:
```
def RMSE(predictions, targets):
    return np.sqrt(np.mean((predictions-targets)**2))
print(RMSE(y[:294].squeeze(), x_estimate_mean))
```

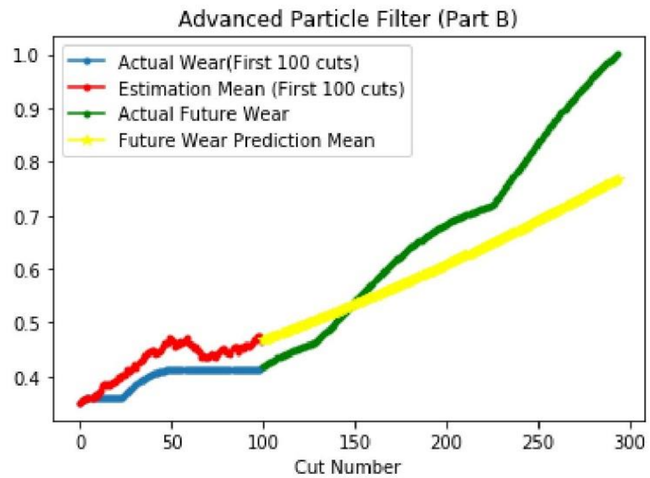
```
0.04376226646449934
```

# Advanced Particle Filter Part B

In [213]:
```
#tracking by particle filter
N_particle = 600
x_estimate, P_est = PF(x[:100,:], N_particle, R) #calculate 0-100 and predict
 the remaining for a and b
x_estimate_mean = np.mean(x_estimate,1)
x_estimate_predict = np.zeros([194,N_particle])
for i in range(294-100):
    #prediction
    x_estimate_predict[i] = ((100+i+1)*P_est[0,:]*(1-P_est[1,:])+0.35**(1-P_es
t[1,:]))**(1/(1-P_est[1,:]))
    x_estimate_predict_mean = np.mean(x_estimate_predict,1)
```

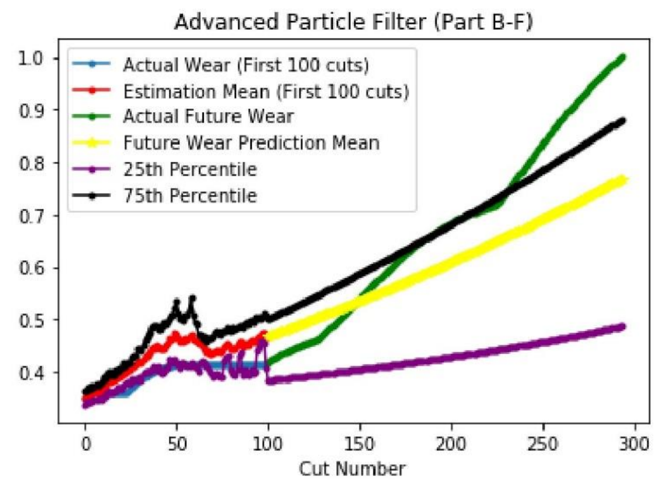
## Plot without Percentiles

In [214]:
```
plt.title('Advanced Particle Filter (Part B)')
plt.plot(np.arange(100), y[:100], marker='.', label= 'Actual Wear(First 100 cu
ts)')
plt.plot(np.arange(100), x_estimate_mean,'r',marker='.', label = 'Estimation M
ean (First 100 cuts)')
plt.plot(np.linspace(100,293,194).transpose(), y[100:],'green', marker='.', la
bel = 'Actual Future Wear')
plt.plot(np.linspace(100,293,194), x_estimate_predict_mean, 'yellow',marker=
'*', label = 'Future Wear Prediction Mean')
plt.xlabel('Cut Number')
leg = plt.legend();
```




## Plot with Percentiles

In [215]:
```
plt.title('Advanced Particle Filter (Part B-F)')
p1 = np.percentile(x_estimate,25,axis=1)
p1f = np.percentile(x_estimate_predict,25,axis=1)
p1 = np.concatenate((p1, p1f), axis=0)
p2 = np.percentile(x_estimate,75,axis=1)
p2f = np.percentile(x_estimate_predict,75,axis=1)
p2 = np.concatenate((p2, p2f), axis=0)
plt.plot(np.arange(100), y[:100], marker='.', label= 'Actual Wear (First 100 c
uts)')
plt.plot(np.arange(100), x_estimate_mean,'r',marker='.', label = 'Estimation M
ean (First 100 cuts)')
plt.plot(np.linspace(100,293,194).transpose(), y[100:],'green', marker='.', la
bel = 'Actual Future Wear')
plt.plot(np.linspace(100,293,194), x_estimate_predict_mean, 'yellow',marker=
'*', label = 'Future Wear Prediction Mean')
plt.plot(p1, 'purple',marker='.', label = '25th Percentile')
plt.plot(p2, 'black',marker='.', label = '75th Percentile')
plt.xlabel('Cut Number')
leg = plt.legend();
```




In [216]:
```
def RMSE(predictions, targets):
    return np.sqrt(np.mean((predictions-targets)**2))

print(RMSE(y[:100].squeeze(), x_estimate_mean))

RMSE(y[100:].squeeze(), x_estimate_predict_mean)
```

```
0.0380510988902974
```

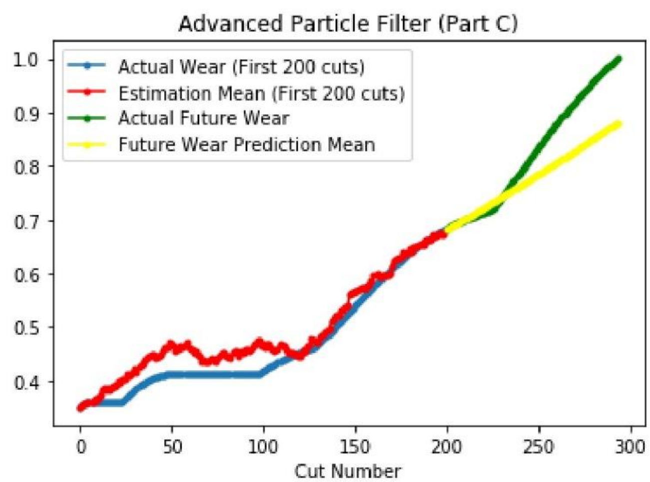
Out[216]:
```
0.10963201676899802
```

# Advanced Particle Filter Part C

In [220]:
```
#tracking by particle filter
N_particle = 600
x_estimate, P_est = PF(x[:200,:], N_particle, R)#calculate 0-100 and predict t
he remaining for a and b
x_estimate_mean = np.mean(x_estimate,1)
x_estimate_predict = np.zeros([94,N_particle])
for i in range(294-200):
    #prediction
    x_estimate_predict[i] = ((200+i+1)*P_est[0,:]*(1-P_est[1,:])+0.35**(1-P_es
t[1,:]))**(1/(1-P_est[1,:]))
    x_estimate_predict_mean = np.mean(x_estimate_predict,1)
```

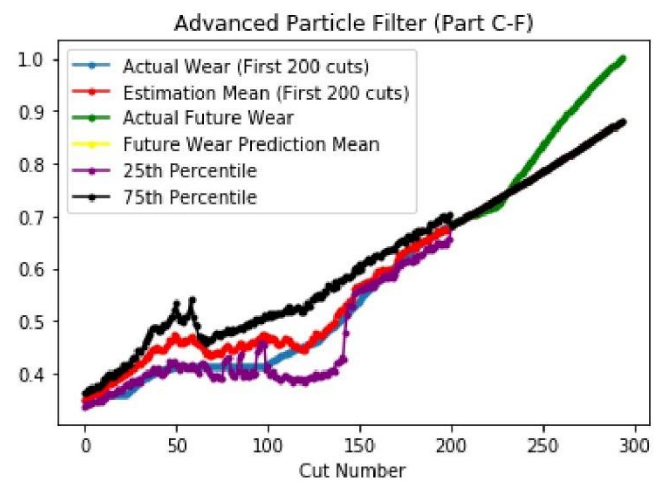
## Plot without Percentiles

In [221]:
```
plt.title('Advanced Particle Filter (Part C)')
plt.plot(np.arange(200), y[:200], marker='.', label= 'Actual Wear (First 200 c
uts)')
plt.plot(np.arange(200), x_estimate_mean,'r',marker='.', label = 'Estimation M
ean (First 200 cuts)')
plt.plot(np.linspace(200,293,94).transpose(), y[200:],'green', marker='.', lab
el = 'Actual Future Wear')
plt.plot(np.linspace(200,293,94), x_estimate_predict_mean, 'yellow',marker='.'
, label = 'Future Wear Prediction Mean')
plt.xlabel('Cut Number')
leg = plt.legend();
```




## Plot with Percentiles

In [222]:
```
plt.title('Advanced Particle Filter (Part C-F)')
p1 = np.percentile(x_estimate,25,axis=1)
p1f = np.percentile(x_estimate_predict,25,axis=1)
p1 = np.concatenate((p1, p1f), axis=0)
p2 = np.percentile(x_estimate,75,axis=1)
p2f = np.percentile(x_estimate_predict,75,axis=1)
p2 = np.concatenate((p2, p2f), axis=0)
plt.plot(np.arange(200), y[:200], marker='.', label= 'Actual Wear (First 200 c
uts)')
plt.plot(np.arange(200), x_estimate_mean,'r',marker='.', label = 'Estimation M
ean (First 200 cuts)')
plt.plot(np.linspace(200,293,94).transpose(), y[200:],'green', marker='.', lab
el = 'Actual Future Wear')
plt.plot(np.linspace(200,293,94), x_estimate_predict_mean, 'yellow',marker='.'
, label = 'Future Wear Prediction Mean')
plt.plot(p1, 'purple',marker='.', label = '25th Percentile')
plt.plot(p2, 'black',marker='.', label = '75th Percentile')
plt.xlabel('Cut Number')
leg = plt.legend();
```




In [223]:
```
def RMSE(predictions, targets):
    return np.sqrt(np.mean((predictions-targets)**2))

print(RMSE(y[:200].squeeze(), x_estimate_mean))

RMSE(y[200:].squeeze(), x_estimate_predict_mean)
```

```
0.029426061710638933
```

Out[223]:
```
0.06541192525433696
```

In [ ]: