Optimizing FrED Run Conditions

Using run condition models to select optimal run conditions.

Uses auto porcessed run condition data - "Run Condition Data Summary.csv" and pickled regression models in "./Models/"

J. Cuiffi, - Penn State New Kensington

In [103]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.linear_model import LinearRegression
from sklearn.preprocessing import PolynomialFeatures
from sklearn.pipeline import Pipeline
import seaborn as sn
from mpl_toolkits.mplot3d import Axes3D
import pickle
from joblib import dump, load
from scipy.optimize import fsolve
```

In [104]:
```
# load run condition data into dataframe
# current file - 050820 11:06pm
path_local = 'C:/Users/cuiff/Dropbox/Python Common Library/python-fred/data/Reports/'
data = pd.read_csv(path_local + 'Run Condition Data Summary.csv')
data.head()
```

Out[104]:

	Run File	Feed Rate Ave (RPS)	Spool Wind Rate Set (RPS)	Spool Rate Ave (RPS)	Wind BF Rate Ave (PPS)	Heater Set (C)	Heater Temp Ave (C)	Filament Diamter Ave (mm)	Filament Std Dev (mm)	System Power Ave (W)	System Power Std Dev (W)	Cu
0	log_Manual Control__2020-04-12_12-13-50.csv	0.001000	1.00	1.000132	127.730719	90.0	90.009313	0.213487	0.016746	29.041052	1.000198	17
1	log_Manual Control__2020-04-12_17-26-48.csv	0.000312	0.25	0.250982	20.426288	90.0	90.068162	0.228383	0.044286	30.064674	1.123981	19
2	log_Manual Control__2020-04-12_17-26-48.csv	0.000312	0.50	0.499899	28.561457	90.0	90.080434	0.172805	0.012332	29.126951	1.001411	18
3	log_Manual Control__2020-04-12_17-49-04.csv	0.000500	0.25	0.254232	45.159629	90.0	89.850642	0.302968	0.022932	28.685025	1.093912	18
4	log_Manual Control__2020-04-12_17-49-04.csv	0.000500	0.50	0.499777	63.865360	90.0	90.043857	0.217002	0.014405	28.366543	0.931641	17

In [105]:
```
# load pickled models
model_path = './Models/'
# linear regression for fiber diameter - model2
# inputs ['Sqrt Feed/Spool','Heater Temp Ave (C)']
dia_mod = load(model_path + 'dia_mod_1.p')

# polynomial regression (order 2) for fiber standard deviation - model5
# inputs ['Feed Rate Ave (RPS)','Spool Rate Ave (RPS)','Heater Temp Ave (C)']
dia_std_mod = load(model_path + 'dia_std_mod_1.p')

# polynomial regression (order 2) for system power - model6
# inputs ['Feed Rate Ave (RPS)','Spool Rate Ave (RPS)','Heater Temp Ave (C)']
sys_pow_mod = load(model_path + 'sys_pow_mod_1.p')
```

Using the Models

Examples on how to use the models, and solve for unknown paramters.

In [106]:
```
# calculate a fiber diameter - single input conditions
# note the input to the model should be a 2D array with different paramters as the columns
print('Single Condition:')
feed = .001 # RPS
spool = 1.0 # RPS
temp = 80.0 # C
X = np.array([[np.sqrt(feed/spool),temp]])
print(X)
dia = dia_mod.predict(X)
print('diameter (mm) = ', dia[0])

# calculate fiber diameter - multiple input conditions
print('Multiple Conditions:')
feeds = np.array([.0005,.001,.002,.003]) # RPS
spool = 1.0 # RPS
temps = np.ones(4) * 80.0 # C
X = np.array([np.sqrt(feeds/spool),temps]).T
print(X)
dias = dia_mod.predict(X)
print('diameters (mm) = ', dias)
```

```
Single Condition:
[[3.16227766e-02 8.00000000e+01]]
diameter (mm) =  0.21282563470444538
Multiple Conditions:
[[2.23606798e-02 8.00000000e+01]
 [3.16227766e-02 8.00000000e+01]
 [4.47213595e-02 8.00000000e+01]
 [5.47722558e-02 8.00000000e+01]]
diameters (mm) =  [0.15881684 0.21282563 0.28920561 0.34781402]
```

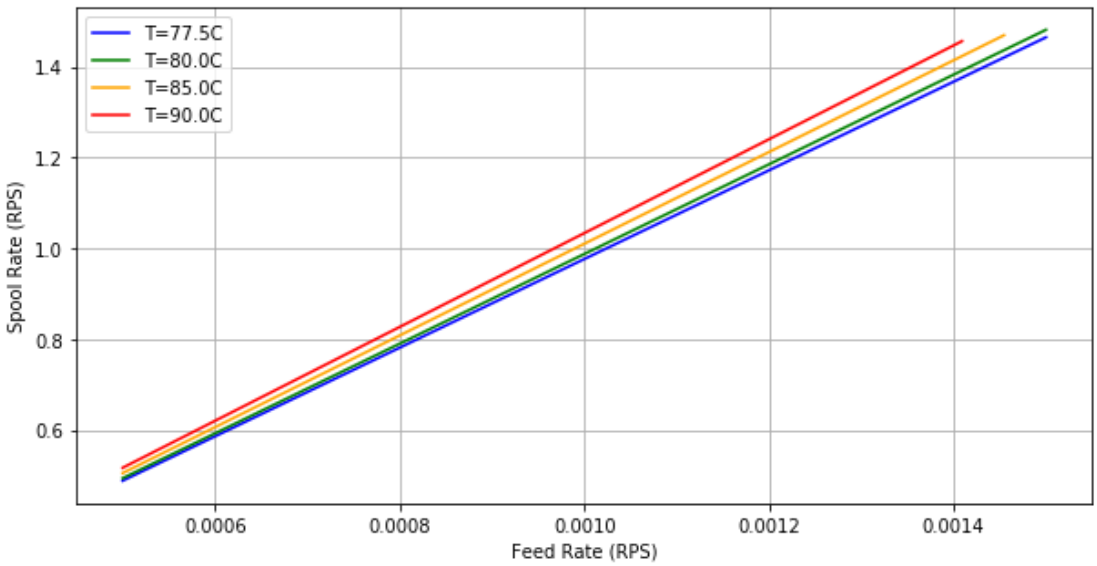
In [107]:

```
# solve for a condition given a desired diameter
# uses the fsolve fucntion - single input conditions
# define wrapper function to solve for diamter and subtract from desired diamter
def calc_dia_zero(spool, *params):
    dia_des, feed, temp = params
    X = np.array([[np.sqrt(feed/spool),temp]])
    dia_calc = dia_mod.predict(X)[0]
    return dia_des - dia_calc

# define the function arguments - constants/parameters
# (desired diamter, feed rate, temperature)
inputs = (.213,.001,80.0)
print('Desired diameter (mm) = ', inputs[0])
print('Feed rate (RPS) = ', inputs[1])
print('Temperature (C) = ', inputs[2])
# solve for spool speed, .75 is inital guess
spool = fsolve(calc_dia_zero, .75, args=inputs)[0]
print('Calculated spool speed (RPS) = ', spool)
```

```
Desired diameter (mm) =  0.213
Feed rate (RPS) =  0.001
Temperature (C) =  80.0
Calculated spool speed (RPS) =  0.9981114903219982
```

In [108]:
```
# Vizualize the options for feed and spool rate for a desired diameter at given temps
dia = .214   # mm - desired diamter
feeds = np.linspace(.0005,.005,num=100)
spools = np.zeros(feeds.size)

# when plotting -impose spool rate limitations
fig, ax1 = plt.subplots()
fig.set_size_inches(10,5)
temp = 77.5 # C - run temperature
for i, spool in enumerate(spools):
    spools[i] = fsolve(calc_dia_zero, .75, args=(dia,feeds[i],temp))[0]
ax1.plot(feeds[spools<=1.5],spools[spools<1.5],c='blue')
temp = 80.0 # C - run temperature
for i, spool in enumerate(spools):
    spools[i] = fsolve(calc_dia_zero, .75, args=(dia,feeds[i],temp))[0]
ax1.plot(feeds[spools<=1.5],spools[spools<1.5],c='green')
temp = 85.0 # C - run temperature
for i, spool in enumerate(spools):
    spools[i] = fsolve(calc_dia_zero, .75, args=(dia,feeds[i],temp))[0]
ax1.plot(feeds[spools<=1.5],spools[spools<1.5],c='orange')
temp = 90.0 # C - run temperature
for i, spool in enumerate(spools):
    spools[i] = fsolve(calc_dia_zero, .75, args=(dia,feeds[i],temp))[0]
ax1.plot(feeds[spools<=1.5],spools[spools<1.5],c='red')
ax1.legend(('T=77.5C','T=80.0C','T=85.0C','T=90.0C'))
ax1.set_xlabel('Feed Rate (RPS)')
#ax1.set_ylim(0,1.5)
ax1.set_ylabel('Spool Rate (RPS)')
ax1.grid()
print('Feed and Spool Rate Combinations for a Fiber Diameter of ' + str(dia) + 'mm')
```

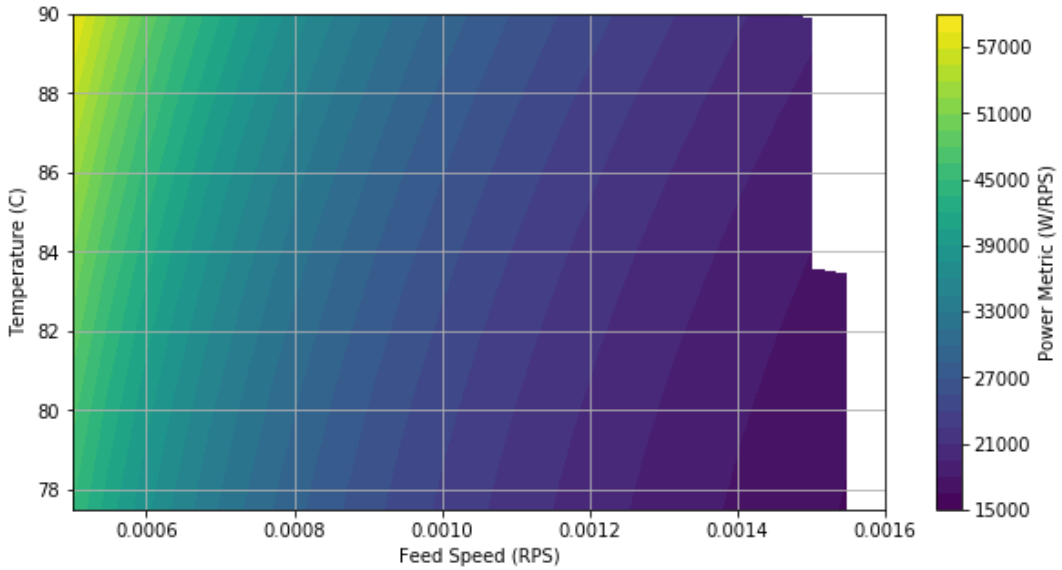
```
Feed and Spool Rate Combinations for a Fiber Diameter of 0.214mm
```



In [109]:
```
# Calculate the feed, spool and temp options for Power Metric (Power/feed rate)
dia = .214   # mm - desired diamter
feeds = np.linspace(.0005,.005,num=100)
temps = np.linspace(77.5,90,num=100)
# create meshgrid of the conditions
xx, yy = np.meshgrid(feeds, temps)
spools = np.zeros(xx.flatten().size)
# calculate spool rates for the given diameter
for i, spool in enumerate(spools):
    spools[i] = fsolve(calc_dia_zero, .75, args=(dia,xx.flatten()[i],yy.flatten()[i]))[0]
# calculate power using model
pows = sys_pow_mod.predict(np.array([xx.flatten(),spools,yy.flatten()]).T)
#print(pows/feeds)
# power metric grid
zz = np.divide(pows,xx.flatten()).reshape(xx.shape)
spools = spools.reshape(xx.shape)
```

In [110]:
```
# Vizualize the feed, spool and temp options for Power Metric (Power/feed rate)
# include only results for valid spool speeds - use masked array
zzma = np.ma.masked_array(zz, (spools > 1.55))
fig, ax1 = plt.subplots()
fig.set_size_inches(10,5)
CS = ax1.contourf(xx,yy,zzma, levels=30, alpha=1.0)#, vmin=0.15, vmax=.7)
ax1.set_ylabel('Temperature (C)')
ax1.set_xlabel('Feed Speed (RPS)')
ax1.set_xlim(.0005,.0016)
CB = fig.colorbar(CS)
CB.ax.set_ylabel('Power Metric (W/RPS)')
ax1.grid()
print('Power Metric versus Feed Speed and Temperature')
```

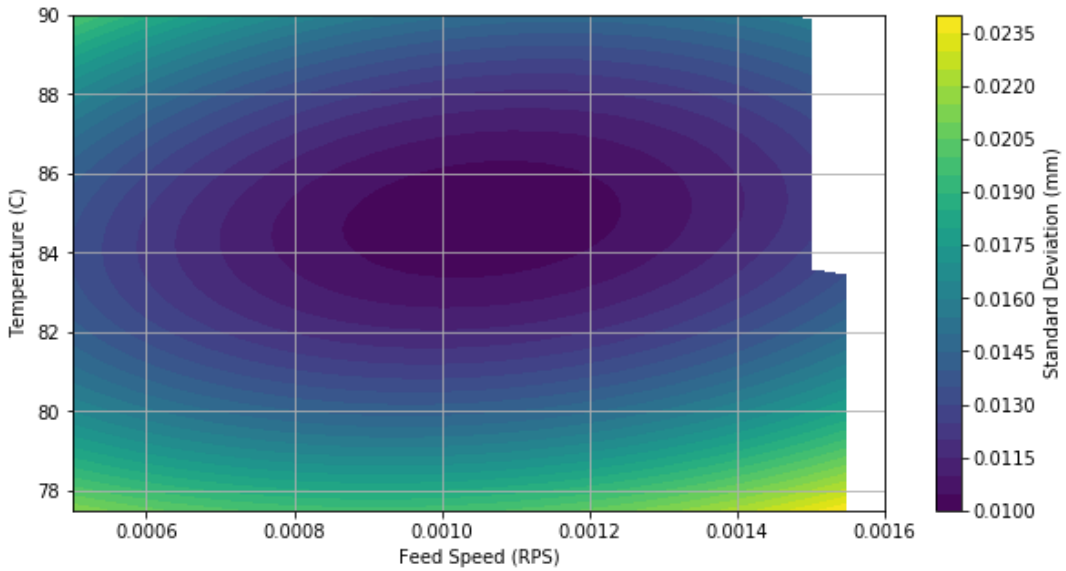
```
Power Metric versus Feed Speed and Temperature
```



In [129]:
```
# Calculate the feed, spool and temp options for Fiber Diameter Standard Deviation
dia = .214   # mm - desired diamter
feeds = np.linspace(.0005,.005,num=100)
temps = np.linspace(77.5,90,num=100)
# create meshgrid of the conditions
xx2, yy2 = np.meshgrid(feeds, temps)
spools = np.zeros(xx2.flatten().size)
# calculate spool rates for the given diameter
for i, spool in enumerate(spools):
    spools[i] = fsolve(calc_dia_zero, .75, args=(dia,xx2.flatten()[i],yy2.flatten()[i]))[0]
# calculate power using model
stds = dia_std_mod.predict(np.array([xx2.flatten(),spools,yy2.flatten()]).T)
# power metric grid
zz2 = stds.reshape(xx2.shape)
spools = spools.reshape(xx2.shape)
```

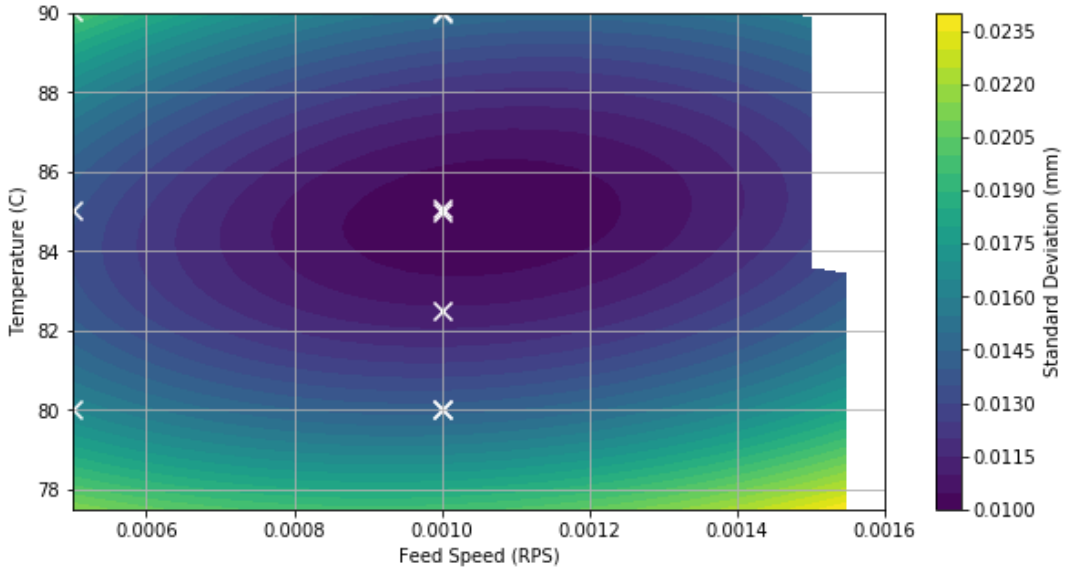
In [130]:
```
# Vizualize the feed, spool and temp options for Standard Deviation
# include only results for valid spool speeds - use masked array
zzma2 = np.ma.masked_array(zz2, (spools > 1.55))
fig, ax1 = plt.subplots()
fig.set_size_inches(10,5)
CS = ax1.contourf(xx2,yy2,zzma2, levels=30, alpha=1.0)
ax1.set_ylabel('Temperature (C)')
ax1.set_xlabel('Feed Speed (RPS)')
ax1.set_xlim(.0005,.0016)
CB = fig.colorbar(CS)
CB.ax.set_ylabel('Standard Deviation (mm)')
ax1.grid()
print('Standard Deviation versus Feed Speed and Temperature')
```

```
Standard Deviation versus Feed Speed and Temperature
```



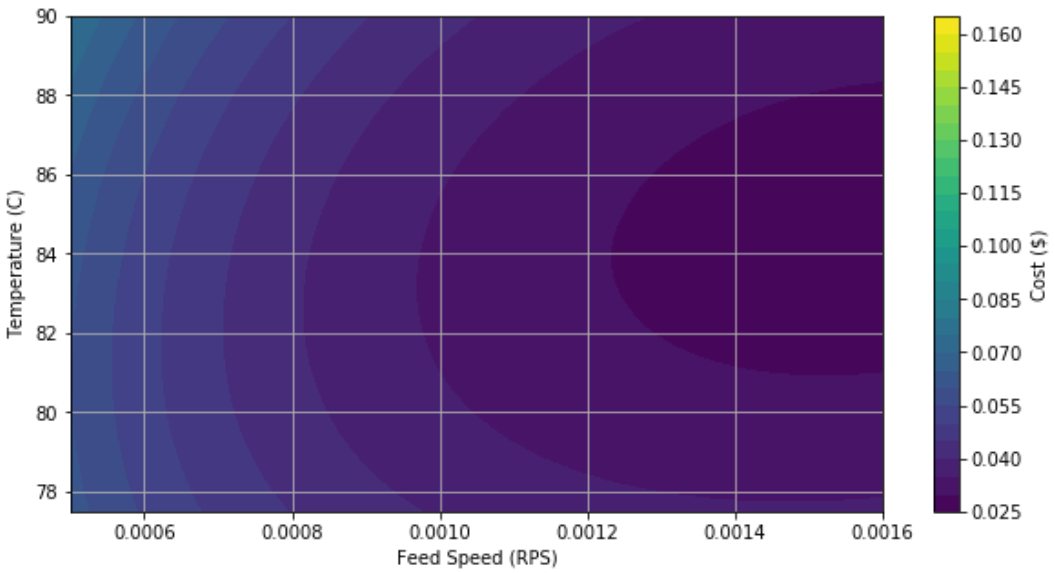
In [131]:
```
# Vizualize the feed, spool and temp options for Standard Deviation
# Add in historic run conditions - pull out ones close to dia
dia_data = data[data['Filament Diamter Ave (mm)'] < .22]
dia_data = dia_data[dia_data['Filament Diamter Ave (mm)'] > .21]
fig, ax1 = plt.subplots()
fig.set_size_inches(10,5)
CS = ax1.contourf(xx2,yy2,zzma2, levels=30, alpha=1.0)#, vmin=0.15, vmax=.7)
ax1.scatter(dia_data['Feed Rate Ave (RPS)'],dia_data['Heater Temp Ave (C)'],marker='x',c='white',s
=100)
ax1.set_ylabel('Temperature (C)')
ax1.set_xlabel('Feed Speed (RPS)')
ax1.set_xlim(.0005,.0016)
ax1.set_ylim(77.5,90.0)
CB = fig.colorbar(CS)
CB.ax.set_ylabel('Standard Deviation (mm)')
ax1.grid()
print('Standard Deviation versus Feed Speed and Temperature')
```

Standard Deviation versus Feed Speed and Temperature



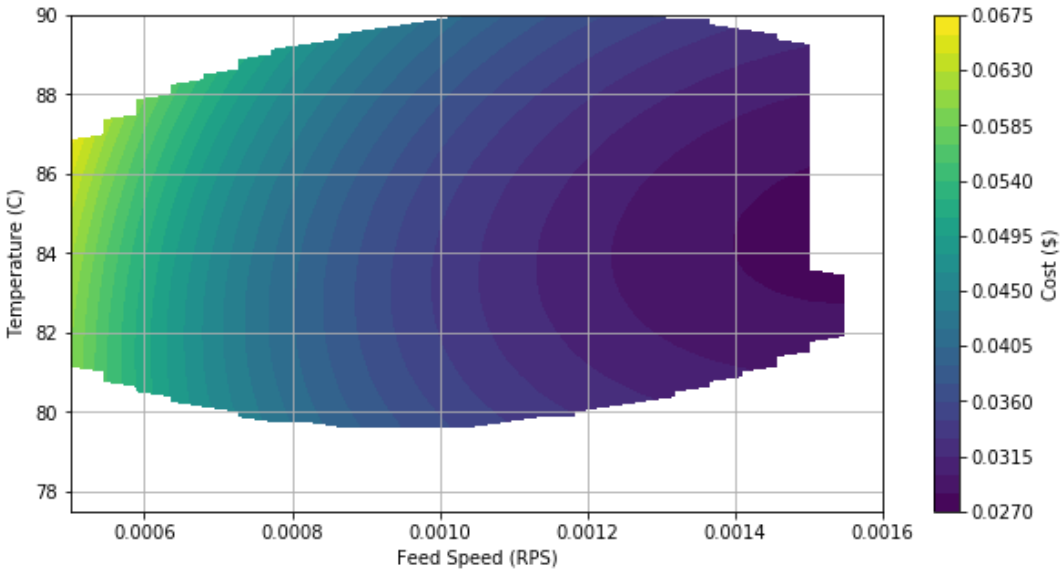
In [142]:
```
# optimize run conditions with multiple constraints -mulit-objective optimization
# define a constraint on standard deviation
std_max = .015
# assign a cost factor that converts W/RPS to $ (essentially $ per length based on power use)
pow_fac = .000001
# assign a cost factor for standard deviation (essentially $ per length cost having less quality)
std_fac = .8
# get total cost matrix
zz3 = (zz * pow_fac) + (zz2 * std_fac)

# Vizualize cost versus run conditions
fig, ax1 = plt.subplots()
fig.set_size_inches(10,5)
CS = ax1.contourf(xx2,yy2,zz3, levels=30, alpha=1.0)#, vmin=0.15, vmax=.7)
ax1.set_ylabel('Temperature (C)')
ax1.set_xlabel('Feed Speed (RPS)')
ax1.set_xlim(.0005,.0016)
ax1.set_ylim(77.5,90.0)
CB = fig.colorbar(CS)
CB.ax.set_ylabel('Cost ($)')
ax1.grid()
print('Cost Function versus Feed Speed and Temperature')
```

```
Cost Function versus Feed Speed and Temperature
```



In [143]:
```
# Vizualize cost versus run conditions with masking
zzma3 = np.ma.masked_array(zz3, ((spools > 1.55) | (zz2 > std_max)))
fig, ax1 = plt.subplots()
fig.set_size_inches(10,5)
CS = ax1.contourf(xx2,yy2,zzma3, levels=30, alpha=1.0)#, vmin=0.15, vmax=.7)
ax1.set_ylabel('Temperature (C)')
ax1.set_xlabel('Feed Speed (RPS)')
ax1.set_xlim(.0005,.0016)
ax1.set_ylim(77.5,90.0)
CB = fig.colorbar(CS)
CB.ax.set_ylabel('Cost ($)')
ax1.grid()
print('Cost Function versus Feed Speed and Temperature')
```

```
Cost Function versus Feed Speed and Temperature
```



In [ ]: