Estimation of Wind Turbine Energy Generation

By Mauricio Hernandez

Goal(s):

- Collect and download data from client site from a set of wind stations using the NREL API Wind Toolkit Data Downloads.
- Get insights from wind speed and wind direction data
- Estimate the Energy Generation WT using parametric and non-parametric methods
- Know the procedure defined by the Standard IEC 61400-12-1 to measure the power performance of a WT.

Main References:

1. Vaishali Sohoni, S. C. Gupta, R. K. Nema, "A Critical Review on Wind Turbine Power Curve Modelling Techniques and Their Applications in Wind Based Energy Systems", Journal of Energy, vol. 2016, Article ID 8519785, 18 pages, 2016. https://doi.org/10.1155/2016/8519785
2. NREL Wind Toolkit. https://developer.nrel.gov/docs/wind/wind-toolkit/mexico-wtk.download/
3. IEC 61400-12-1 Ed. 2.0 b:2017. https://webstore.ansi.org/Standards/IEC/IEC6140012Ed2017?gclid=Cj0KCQiAybaRBhDtARIsAIEG3km-XBhqVUeiIgIzgOYhkRR3cSPheA_Z4vfjCvsK5kYqS70XxIkvw20aAp_rEALw_wcB

Standard IEC 61400-12-1

The standard IEC 61400-12-1 is the most accepted standard for power performance measurement of single wind turbines. It specifies the procedure to measrue the performance characteristics of single wind turbines. Its this methodology requires simultaneous measuremens of wind speed and power output at the test site to estimate the power curve.

The IEC standard uses 10-minute averaged data grouped into wind speed intervals of 0.5 m/s (bins). This 10-minute averaging of data introduces systematic averaging errors and short wind fluctuations are killed off.

Although the IEC power curve considers the wind condition of the current site it may not always be appropriate to apply to the wind conditions of other sites. The discrete model prescribed in IEC 61400-12 is simple, but a large amount of data is required to develop a reliable model.

In [1]:
```
#Import libraries
#pip install windrose
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
from datetime import datetime
import math
import seaborn as sns
from windrose import WindroseAxes
```

1. Read and Explore Wind Data

In [2]:
```
year = 2010
lon= -100.432 #-100.432
lat= 20.833  #20.833
height = 80 #80 # in m
print('Wind site location: \n Longitude: %.4f, Latitude: %.4f' % (lon, lat))
```

```
Wind site location:
 Longitude: -100.4320, Latitude: 20.8330
```

In [3]:
```
#Read data

#df_wind = pd.read_csv('./inputs/wind_site_2010.csv', index_col = False, skiprows = 1)
df_wind = pd.read_csv('./inputs/wind_speed_80m_2010.csv', index_col = False, skiprows = 1)
df_wind.insert(0, 'Date', datetime(year,1,1,0,0,0))
df_wind['Date'] = pd.to_datetime(df_wind[['Month','Day','Year', 'Hour', 'Minute']])
df_wind.head()
```

Out[3]:

	Date	Year	Month	Day	Hour	Minute	wind speed at 80m (m/s)	wind direction at 80m (deg)
0	2010-01-01 00:00:00	2010	1	1	0	0	2.80	29.62
1	2010-01-01 01:00:00	2010	1	1	1	0	0.49	200.49
2	2010-01-01 02:00:00	2010	1	1	2	0	3.28	10.00

	Date	Year	Month	Day	Hour	Minute	wind speed at 80m (m/s)	wind direction at 80m (deg)
3	2010-01-01 03:00:00	2010	1	1	3	0	3.07	328.03
4	2010-01-01 04:00:00	2010	1	1	4	0	2.42	287.20

In [4]:
```
df_wind.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 8760 entries, 0 to 8759
Data columns (total 8 columns):
 #   Column                      Non-Null Count  Dtype
---  ------                      --------------  -----
 0   Date                        8760 non-null   datetime64[ns]
 1   Year                        8760 non-null   int64
 2   Month                       8760 non-null   int64
 3   Day                         8760 non-null   int64
 4   Hour                        8760 non-null   int64
 5   Minute                      8760 non-null   int64
 6   wind speed at 80m (m/s)     8760 non-null   float64
 7   wind direction at 80m (deg) 8760 non-null   float64
dtypes: datetime64[ns](1), float64(2), int64(5)
memory usage: 547.6 KB
```

1.1 Descriptive Statistics

In [5]:
```
# change name of column
df_wind = df_wind.rename(columns={"wind speed at 80m (m/s)":"Wind_speed",
                                  "wind direction at 80m (deg)":"Wind_dir"})
df_wind_stats = df_wind.describe()
df_wind_stats
```

Out[5]:

	Year	Month	Day	Hour	Minute	Wind_speed	Wind_dir
count	8760.0	8760.000000	8760.000000	8760.000000	8760.0	8760.000000	8760.000000
mean	2010.0	6.526027	15.720548	11.500000	0.0	3.718331	145.449524
std	0.0	3.448048	8.796749	6.922582	0.0	2.513775	101.336230
min	2010.0	1.000000	1.000000	0.000000	0.0	0.050000	0.000000
25%	2010.0	4.000000	8.000000	5.750000	0.0	1.690000	53.775000
50%	2010.0	7.000000	16.000000	11.500000	0.0	3.210000	134.530000
75%	2010.0	10.000000	23.000000	17.250000	0.0	5.310000	227.957500
max	2010.0	12.000000	31.000000	23.000000	0.0	16.780000	359.990000

Questions:

1. What is the minimum, average, maximum, and standard deviation wind speed?
   Answer: 0.05, 3.71, 16.78, 2.51
2. What is the median value of the wind direction?
   Answer: 134.53
3. What is the sample size (n)?
   Answer: 8760

Save these values in the following variables:

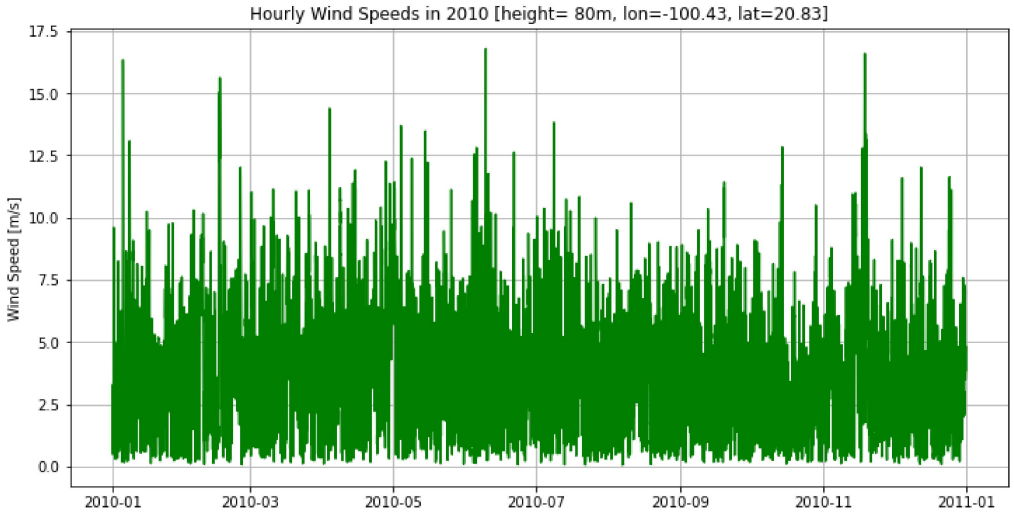
In [6]:
```
avg_wind_speed = df_wind_stats.Wind_speed['mean']
std_wind_speed =  df_wind_stats.Wind_speed['std']
min_wind_speed =  df_wind_stats.Wind_speed['min']
med_wind_dir = df_wind_stats.Wind_dir['50%']
max_wind_speed =  df_wind_stats.Wind_speed['max']

n_sample = df_wind_stats.Wind_speed['count']
```

In [7]:
```
#save descriptive statistics
df_wind_stats.to_csv('./outputs/descriptive_stats_wind_%i.csv'% height)
```

Historical Wind Speed

In [8]:
```
fig, ax = plt.subplots(figsize=(12,6))
```

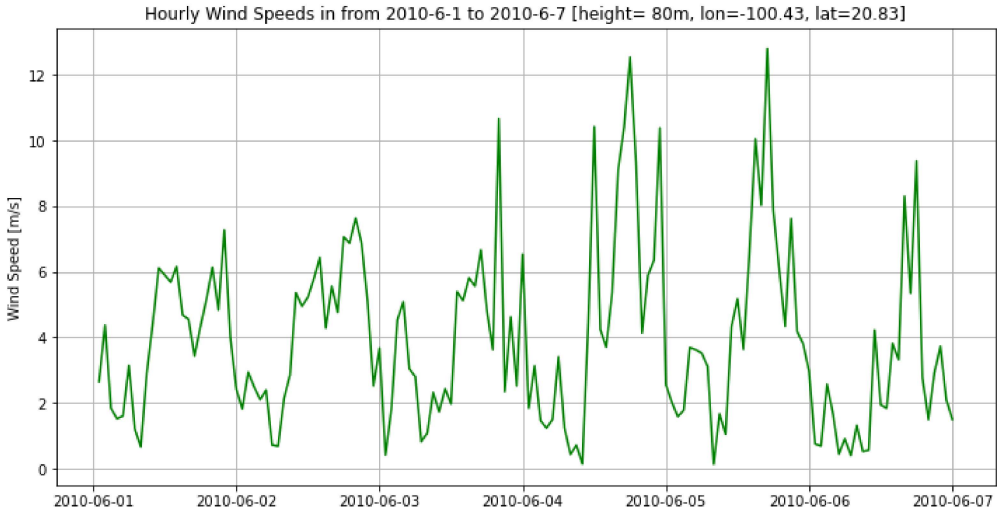
```
plt.plot(df_wind.Date, df_wind.Wind_speed, 'g',linestyle = 'solid')
plt.grid(True)
plt.title("Hourly Wind Speeds in 2010 [height= %im, lon=%.2f, lat=%.2f]" % (height, lon, lat))
plt.ylabel("Wind Speed [m/s]")
plt.savefig("./outputs/wind_speed_2010_%s.png" % height, transparent=True)
plt.show()
```



Visually check the variability of the wind speed

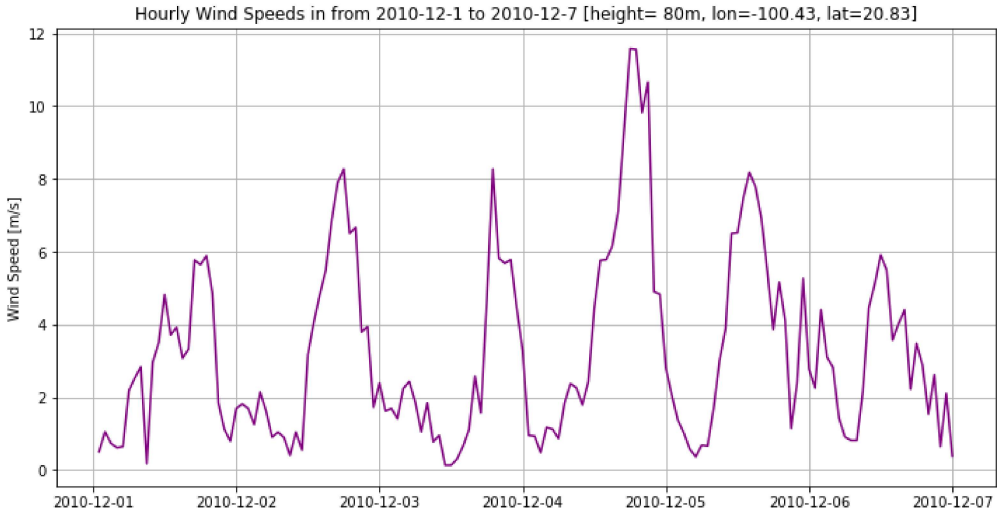
In [9]:

```
ini_date ='2010-6-1'
end_date = '2010-6-7'

date_mask = (df_wind['Date'] > ini_date) & (df_wind['Date'] <= end_date)

fig, ax = plt.subplots(figsize=(12,6))
plt.plot(df_wind.loc[date_mask].Date, df_wind.loc[date_mask].Wind_speed, 'g',linestyle = 'solid')
plt.grid(True)
plt.title("Hourly Wind Speeds in from %s to %s [height= %im, lon=%.2f, lat=%.2f]" % (ini_date, end_date, height, lon, lat))
plt.ylabel("Wind Speed [m/s]")
plt.savefig("./outputs/wind_speed_%s_date_ranges.png" % height, transparent=True)
plt.show()
```



In [10]:

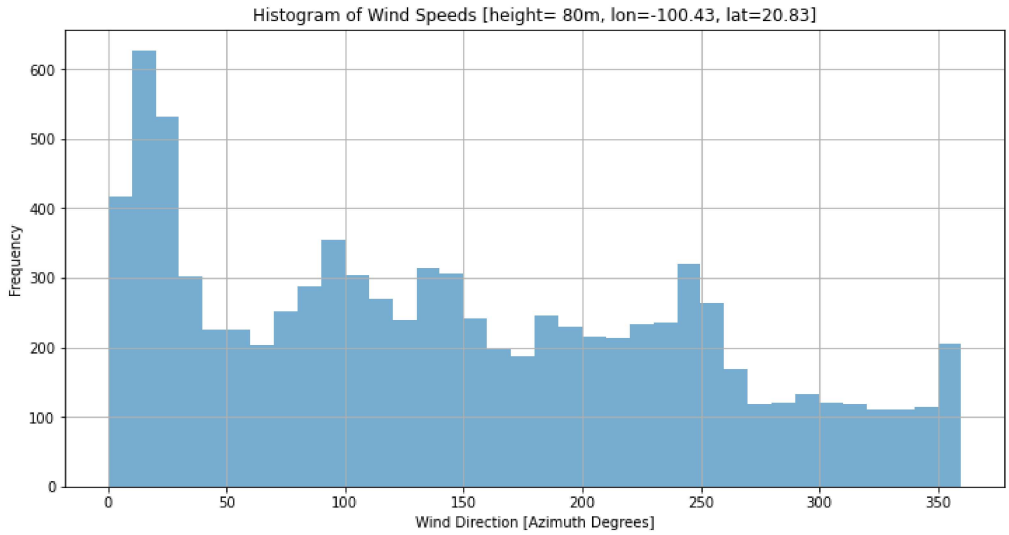
```
ini_date ='2010-12-1'
end_date = '2010-12-7'

date_mask = (df_wind['Date'] > ini_date) & (df_wind['Date'] <= end_date)

fig, ax = plt.subplots(figsize=(12,6))
plt.plot(df_wind.loc[date_mask].Date, df_wind.loc[date_mask].Wind_speed, 'purple',linestyle = 'solid')
plt.grid(True)
plt.title("Hourly Wind Speeds in from %s to %s [height= %im, lon=%.2f, lat=%.2f]" % (ini_date, end_date, height, lon, lat))
plt.ylabel("Wind Speed [m/s]")
plt.savefig("./outputs/wind_speed_%i_date_ranges.png" % height, transparent=True)
plt.show()
```



Wind Direction

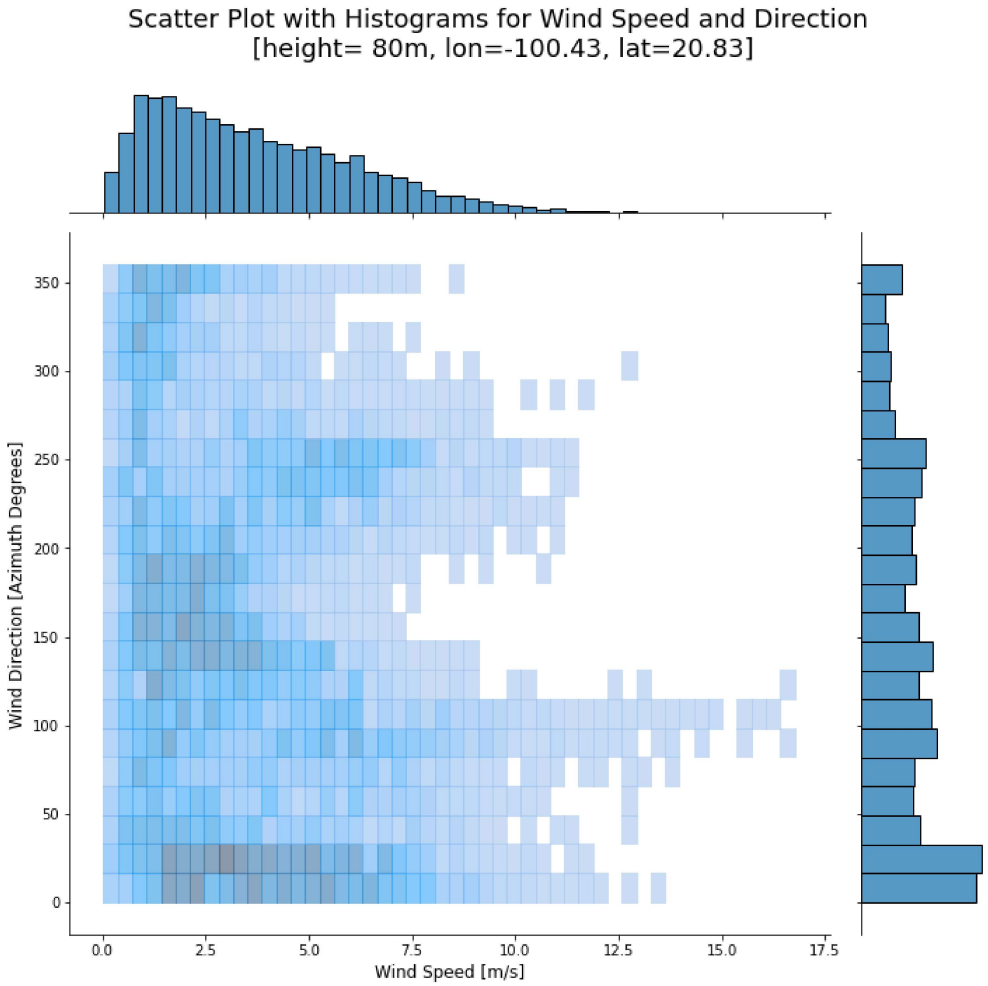
In [11]:

```
fig, ax = plt.subplots(figsize=(12,6))
df_wind.Wind_dir.plot(kind='hist', bins = 36, alpha=0.6)
plt.grid(True)
plt.title("Histogram of Wind Speeds [height= %im, lon=%.2f, lat=%.2f]" % (height, lon, lat))
plt.xlabel("Wind Direction [Azimuth Degrees]")
plt.savefig("./outputs/hist_wind_dir_%i.png" % height, transparent=True)
plt.show()
```



In [12]:

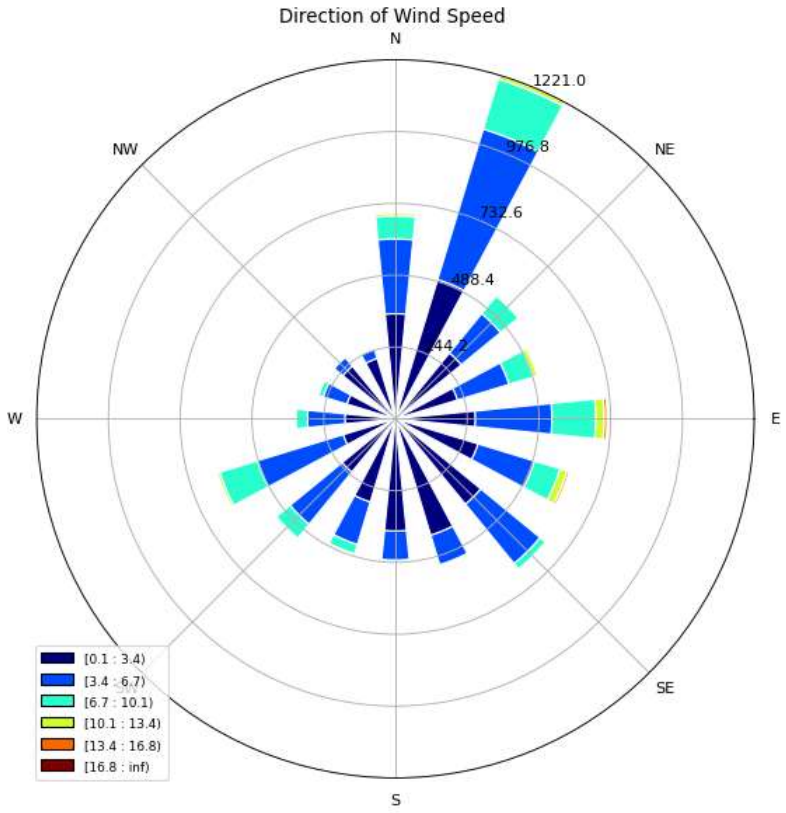
```
# plotting scatterplot with histograms for wind speed and wind direction
p = sns.jointplot(data=df_wind, x="Wind_speed", y="Wind_dir", kind ='hist', height=10)
p.fig.suptitle("Scatter Plot with Histograms for Wind Speed and Direction\n [height= %im, lon=%.2f, lat=%.2f]"
               % (height, lon, lat), fontsize=18)
p.set_axis_labels('Wind Speed [m/s]', 'Wind Direction [Azimuth Degrees]', fontsize=12)
p.ax_joint.collections[0].set_alpha(0.6)
p.fig.tight_layout()
p.fig.subplots_adjust(top=0.9) # Reduce plot size to make room for title

p.savefig("./outputs/scatter_hist_wind_dir_%i.png" % height, transparent=True)
```



Direction of Wind Speed

In [13]:

```
ax = WindroseAxes.from_ax()
ax.bar(df_wind['Wind_dir'], df_wind['Wind_speed'], normed=False, opening=0.5, edgecolor='white')

ax.set_xticklabels(['E', 'NE', 'N', 'NW',  'W', 'SW', 'S', 'SE'])
ax.set_theta_zero_location('E')
ax.set_title('Direction of Wind Speed ')
ax.set_legend()
plt.savefig("./outputs/azimuth_wind_speed_dir_%i.png" % height, transparent=True)
plt.show()
```



2. Estimating Annual Energy Generation

Turbine GEV MP R 32m 275 kW

"The GEV MP meets all the specific requirements of small grids and outperforms any turbine of its class."

Relevant Specifications:

- **Rotor diameter :** $32m$
- **Cut in wind speed:** $v_{in} = 3.5m/s$ \
- **Rated wind speed:** $v_r = 12.5m/s$ \
- **Cut off wind speed:** $v_{off} = 25m/s$ \
- **Nominal Power:** $P_{max} = 275kW$

Ref: **GEV MP C Datasheet**

2.1 Using Wind Power Equation

The theoretical Power Output of a wind turbine (WT) can be expressed by the following equation:

$$P = \frac{1}{2} C_{pmax} \rho A v^3$$

, where C_{pmax} = Betz limit (0.59), ρ = air density, A= swept area of blades, and v= wind speed

The power production of a wind turbine (WT) is affected by many parameters such as wind speed, wind direction, air density (function of temperature, pressure, and humidity) and turbine parameters.

WT can't operate at Betz limit. In real world limit is well below 0.59, values of 0.35-0.45 are common even in the best WTs.

So, the `available Power Output` of a WT can be expressed by the following equation:

$$P = \frac{1}{2} C_p \rho A v^3$$

, where C_p is the WT power efficiency, commonly expressed as η

References:

- **Wind Turbine Power Calculation, The Royal Academy of Engineering**

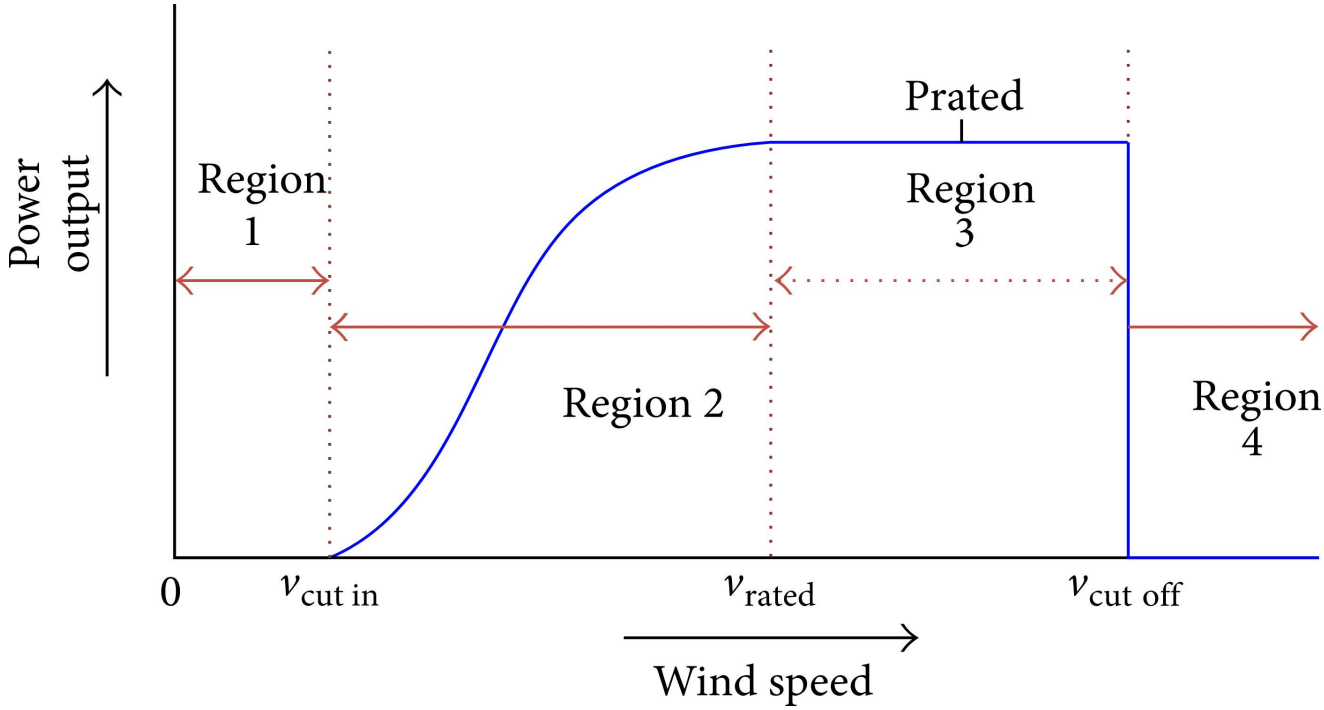
- How Wind Turbines Work, Energy and the Environment, Penn State

WT Power Curve

Wind turbines have a minimum wind speed that differs depending on the size but is typically about 4-5 m/s (about 15 km/h) and maximum wind speed above which they shut down to avoid damage, usually around 20-25 m/s (~80 km/h).

A typical power curve of a pitch regulated wind turbine is shown in the figure below.

- **"In the first region when the wind speed is less than a threshold minimum, known as the cut-in speed, the power output is zero.**
- **In the second region between the cut-in and the rated speed, there is a rapid growth of power produced.**
- **In the third region, a constant output (rated) is produced until the cut-off speed is attained. - Beyond the cut-off speed (region 4) the turbine is taken out of operation to protect its components from high winds; hence it produces zero power in this region." (Gupta, 2016)**



Calculations

In [14]:
```
## WT parameters
turbine_model ='GEV MP R 32m 275 kW'
l_rotor = 16
v_cut_in = 3.5
v_r = 12.5
v_cut_off = 25
p_max =  275 #in kW

air_density = 1.1849 # in kg/m3, https://www.gribble.org/cycling/air_density.html
c_p_max = 0.59
```

First we need to estimate C_p, by estimating the theoretical maximum power. To do so, we can use the following equation:

$$C_p = \frac{P_{max}}{P_{max(theor)}} C_{pmax}$$

In [15]:
```
blades_area = math.pi * (l_rotor**2)

p_out_theo = 0.5* c_p_max * air_density * blades_area * (v_r**3)
p_out_theo_kw = p_out_theo/1000
print ("WT Maximum Power Output [theoretical] %.3f kW" % (p_out_theo_kw))
```

```
WT Maximum Power Output [theoretical] 549.065 kW
```

In [16]:
```
c_p = (p_max / p_out_theo_kw)* c_p_max

print ("WT power efficiency: %.3f" % (c_p))
```

```
WT power efficiency: 0.296
```

In [17]:
```
p_out_avg  = c_p * air_density * blades_area * (avg_wind_speed**3)
energy_gen_year  = (p_out_avg * n_sample) /1000000

print ("WT Annual Energy Production: %.3f MWh" % energy_gen_year)
```

```
WT Annual Energy Production: 126.818 MWh
```

Finally, we estimate the Capacity Factor (CF) of the WT:

CF = Annual Energy Generated / (Nominal Power * Time Period)

In [18]:
```
cf = (energy_gen_year*1000) / (p_max*n_sample) # in Energy in MW, p_max in kWh

print ("Capacity Factor: %.3f" % (cf))
```

```
Capacity Factor: 0.053
```

Question(s):

1. What are the limitations of calculating WT Annual production by using the annual average wind speed?

In [19]:
```
#Save results in dataframe
df_results = pd.DataFrame()

results = {'method':'WT Power General Equation','min_wind_speed': min_wind_speed,
           'avg_wind_speed': avg_wind_speed, 'max_wind_speed': max_wind_speed,
           'std_wind_speed': std_wind_speed, 'median_wind_dir': med_wind_dir,
           'annual_generation': energy_gen_year,'capacity_factor':cf}

df_results = df_results.append(results, ignore_index=True)
df_results.T
```

Out[19]:

	0
annual_generation	126.818005
avg_wind_speed	3.718331
capacity_factor	0.052643
max_wind_speed	16.78
median_wind_dir	134.53
method	WT Power General Equation
min_wind_speed	0.05
std_wind_speed	2.513775

2.2 Using WT Power Curve

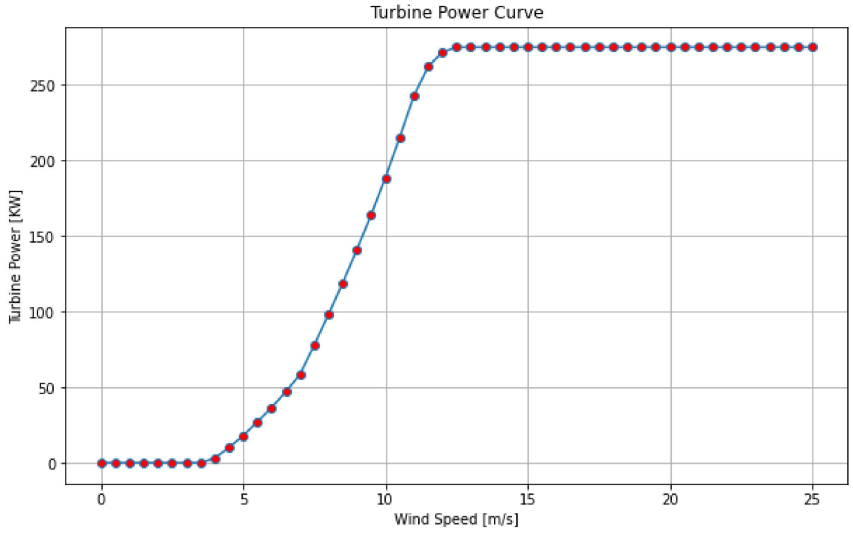
In [20]:
```
#Read power curve of turbine
df_wt_curve = pd.read_csv('./inputs/gev_275kw_32m_power_cuve.csv', index_col = False,)
df_wt_curve.tail(n=10)
```

Out[20]:

	wind_speed	power_out
41	20.5	275.0
42	21.0	275.0
43	21.5	275.0
44	22.0	275.0
45	22.5	275.0
46	23.0	275.0
47	23.5	275.0
48	24.0	275.0
49	24.5	275.0
50	25.0	275.0

In [21]:
```
fig, ax = plt.subplots(figsize=(10,6))
plt.plot(df_wt_curve.wind_speed, df_wt_curve.power_out, marker = 'o', markerfacecolor = 'r')
plt.grid(True)
plt.title("Turbine Power Curve")
plt.xlabel("Wind Speed [m/s]")
```

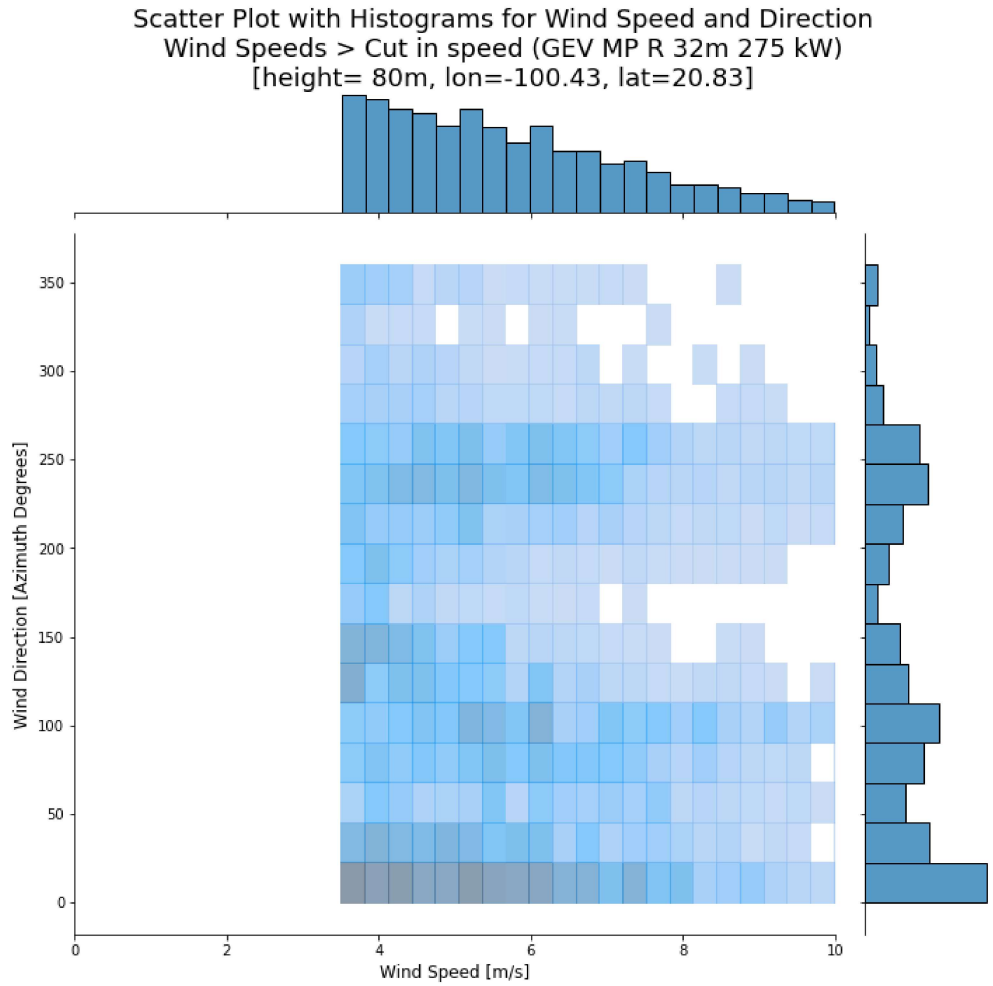
```
plt.ylabel("Turbine Power [KW]")
plt.savefig("./outputs/wt_power_curve_%s.png" % turbine_model, transparent=True)
plt.show()
```



How Wind Speed and Wind Direction looks when Wind speeds below cut_in and above cut_off are removed?

In [22]:

```
# create mask, kind of filter
speed_mask = (df_wind['Wind_speed'] > v_cut_in) & (df_wind['Wind_speed'] <= v_cut_off)

p = sns.jointplot(data=df_wind.loc[speed_mask], x="Wind_speed", y="Wind_dir", kind ='hist', height=10)
p.fig.suptitle("Scatter Plot with Histograms for Wind Speed and Direction\n Wind Speeds > Cut in speed (%s) \n[height= %im, lon=%.2f
               % (turbine_model, height, lon, lat), fontsize=18)
p.set_axis_labels('Wind Speed [m/s]', 'Wind Direction [Azimuth Degrees]', fontsize=12)
p.ax_marg_x.set_xlim(0, 10)
p.ax_joint.collections[0].set_alpha(0.6)
p.fig.tight_layout()
p.fig.subplots_adjust(top=0.9) # Reduce plot size to make room for title

p.savefig("./outputs/scatter_hist_wind_dir_nonparam_%i.png" % height, transparent=True)
```



Descriptive Statistics Keeping only Wind Speeds that are in this range

$$v_{cut\_off} < v_{wind} > v_{cut\_in}$$

In [23]:

```
speed_mask = (df_wind['Wind_speed'] > v_cut_in) & (df_wind['Wind_speed'] <= v_cut_off)
df_wind['Wind_speed_filter'] = df_wind.loc[speed_mask]['Wind_speed']
df_wind['Wind_dir_filter'] = df_wind.loc[speed_mask]['Wind_dir']
df_wind_stats_wt_curve = df_wind.describe()

avg_wind_speed_wt_curve = df_wind_stats_wt_curve.Wind_speed_filter['mean']
std_wind_speed_wt_curve = df_wind_stats_wt_curve.Wind_speed_filter['std']
min_wind_speed_wt_curve = df_wind_stats_wt_curve.Wind_speed_filter['min']
med_wind_dir_wt_curve =df_wind_stats_wt_curve.Wind_dir_filter['50%']
max_wind_speed_wt_curve = df_wind_stats_wt_curve.Wind_speed_filter['max']

df_wind_stats_wt_curve
```

Out[23]:

	Year	Month	Day	Hour	Minute	Wind_speed	Wind_dir	Wind_speed_filter	Wind_dir_filter
count	8760.0	8760.000000	8760.000000	8760.000000	8760.0	8760.000000	8760.000000	4038.000000	4038.000000
mean	2010.0	6.526027	15.720548	11.500000	0.0	3.718331	145.449524	5.921592	128.320409
std	0.0	3.448048	8.796749	6.922582	0.0	2.513775	101.336230	1.936656	96.545329
min	2010.0	1.000000	1.000000	0.000000	0.0	0.050000	0.000000	3.510000	0.090000
25%	2010.0	4.000000	8.000000	5.750000	0.0	1.690000	53.775000	4.422500	35.600000
50%	2010.0	7.000000	16.000000	11.500000	0.0	3.210000	134.530000	5.510000	104.795000
75%	2010.0	10.000000	23.000000	17.250000	0.0	5.310000	227.957500	6.930000	223.345000
max	2010.0	12.000000	31.000000	23.000000	0.0	16.780000	359.990000	16.780000	359.900000

IEC61400 Standard

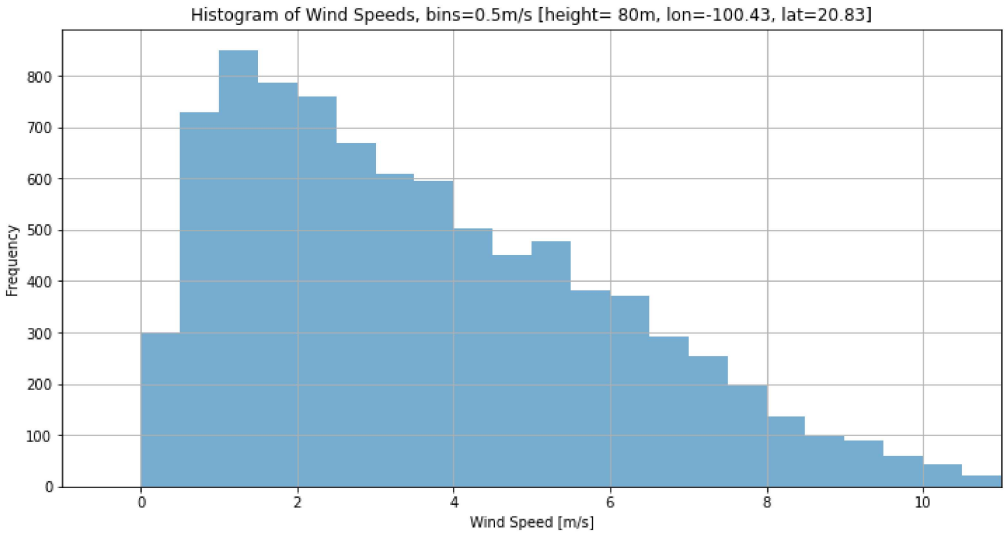
IEC61400 standard According to the IEC61400 standard, wind speeds need to be divided in bins of 0.5m/s.

> **Note: We're not obtaining the Turbine Power Curve, this is a Pseudo-procedure based on this step from the Standard**

In [24]:
```
# create bins
bins_list = np.arange(0, 26, 0.5)

bins_list
```

Out[24]:
```
array([ 0. ,  0.5,  1. ,  1.5,  2. ,  2.5,  3. ,  3.5,  4. ,  4.5,  5. ,
        5.5,  6. ,  6.5,  7. ,  7.5,  8. ,  8.5,  9. ,  9.5, 10. , 10.5,
       11. , 11.5, 12. , 12.5, 13. , 13.5, 14. , 14.5, 15. , 15.5, 16. ,
       16.5, 17. , 17.5, 18. , 18.5, 19. , 19.5, 20. , 20.5, 21. , 21.5,
       22. , 22.5, 23. , 23.5, 24. , 24.5, 25. , 25.5])
```

In [25]:
```
fig, ax = plt.subplots(figsize=(12,6))
ax = plt.hist(df_wind['Wind_speed'], bins = bins_list, alpha=0.6)
plt.xlim(-1, 11)
plt.grid(True)
plt.title("Histogram of Wind Speeds, bins=0.5m/s [height= %im, lon=%.2f, lat=%.2f]" % (height, lon, lat))
plt.xlabel("Wind Speed [m/s]")
plt.ylabel("Frequency")
plt.savefig("./outputs/hist_wind_speed_bins_%i.png" % height, transparent=True)
plt.show()
```



In [26]:
```
# What is the frequency in each bin
print(ax[0])
print(ax[1])
```

```
[301. 728. 849. 786. 760. 670. 610. 596. 501. 451. 477. 381. 371. 293.
 254. 196. 136.  99.  90.  59.  42.  21.  27.  11.  12.  11.   8.   5.
   6.   2.   1.   3.   1.   2.   0.   0.   0.   0.   0.   0.   0.   0.
   0.   0.   0.   0.   0.   0.   0.   0.   0.]
[ 0.   0.5  1.   1.5  2.   2.5  3.   3.5  4.   4.5  5.   5.5  6.   6.5
  7.   7.5  8.   8.5  9.   9.5 10.  10.5 11.  11.5 12.  12.5 13.  13.5
 14.  14.5 15.  15.5 16.  16.5 17.  17.5 18.  18.5 19.  19.5 20.  20.5
 21.  21.5 22.  22.5 23.  23.5 24.  24.5 25.  25.5]
```

In [27]:
```
df_wt_curve['n_hours'] = ax[0]
df_wt_curve['energy_gen'] = df_wt_curve['n_hours'] * df_wt_curve.power_out
energy_gen_year_wt_curve  = sum(df_wt_curve['energy_gen'])/1000 # to get value in MW

print ("WT Annual Energy Production: %.3f MWh" % (energy_gen_year_wt_curve))
```

```
WT Annual Energy Production: 165.633 MWh
```

In [28]:
```
cf_wt_curve = (energy_gen_year_wt_curve*1000) / (p_max*n_sample) # energy in MW, power in kWh

print ("Capacity Factor: %.3f" % (cf_wt_curve))
```

```
Capacity Factor: 0.069
```

Save results in dataframe

In [29]:
```
results_wt_curve = {'method':'Wind Power Curve','min_wind_speed': min_wind_speed_wt_curve,
           'avg_wind_speed': avg_wind_speed_wt_curve, 'max_wind_speed': max_wind_speed_wt_curve,
           'std_wind_speed': std_wind_speed_wt_curve, 'median_wind_dir': med_wind_dir_wt_curve,
           'annual_generation': energy_gen_year_wt_curve,'capacity_factor':cf_wt_curve}
```

```
df_results = df_results.append(results_wt_curve, ignore_index=True)
df_results.T
```

Out[29]:

	0	1
annual_generation	126.818005	165.6334
avg_wind_speed	3.718331	5.921592
capacity_factor	0.052643	0.068756
max_wind_speed	16.78	16.78
median_wind_dir	134.53	104.795
method	WT Power General Equation	Wind Power Curve
min_wind_speed	0.05	3.51
std_wind_speed	2.513775	1.936656

2.3 Using a Parametric Approach with the Power Curve

The power delivered by a WT can be expressed as:

$$P = \begin{cases} 0 & v < v_c, v > v_f \\ q(v) & v_c \leq v < v_r \\ P_r & v_r \leq v \leq v_f \end{cases}$$

The relationship between power output and wind speed of a WT between cut-in and rated speed is nonlinear (region 2). The relation can be approximated by various functions using polynomial and other than polynomial expressions.

Equations for Different Approximations of Power Curve

Model	Expressions of P and q	Parameters
Linear [12]	$q(v) = P_r \frac{(v - v_c)}{(v_r - v_c)}$	—
Quadratic [25]	$q(v) = P_r \left(\frac{v - v_c}{v_r - v_c} \right)^2$	—
Binomial [30]	$q(v) = \left(a + bv + cv^2\right) P_r$	$a = \frac{1}{(v_c - v_r)^2} \left[v_c (v_c + v_r) - 4 v_c v_r \frac{(v_c - v_r)^2}{2 v_r} \right]$ $b = \frac{1}{(v_c - v_r)^2} \left[4 (v_c + v_r) \frac{(v_c - v_r)^3}{2 v_r} - 3 (v_c + v_r) \right]$ $c = \frac{1}{(v_c - v_r)^2} \left[2 - 4 \frac{(v_n + v_r)^3}{2 v_r} \right]$
Cubic [31]	$q(v) = av^3 - bP_r$	$a = \frac{P_r}{({v_r}^3 - {v_c}^3)}$ $b = \frac{{v_r}^3}{({v_r}^3 - {v_c}^3)}$
Weibull based [6]	$q(v) = a + bv^k$	$a = \frac{P_r {v_c}^k}{({v_c}^k - {v_r}^k)}$ $b = \frac{P_r}{({v_c}^k - {v_r}^k)}$
Double exponential [32]	$P = \exp\left(-\tau_1 \exp\left(-v\tau_2\right)\right)$	τ_1 and τ_2 to be estimated
4PL [1, 33]	$P = a \left(\frac{1 + ame^{v/\tau}}{1 + ame^{v/\tau}} \right)$	$\theta = (a, m, n, \tau)$ (1) From manufacturers' curve [33] $a = P_r$ $n = e^{2sv_{ip}/(P_r - P_{ip})}$ $m = n \left(\frac{2P_{ip}}{P_r} - 1 \right)$ $\tau = \frac{(P_r - P_{ip})}{2s}$ (2) Parameters obtained by evolutionary techniques for SCADA data in [1]
4PL [34, 35]	$P = f(v, \theta) = D + \frac{(A - D)}{1 + (v/C)^B}$	$\theta = (A, B, C, D)$ A = minimum asymptote B = Hill slope C = inflection point (point of curve where the curvature changes direction) D = maximum asymptote (parameters obtained by evolutionary techniques)

Selecting a Cubic Function

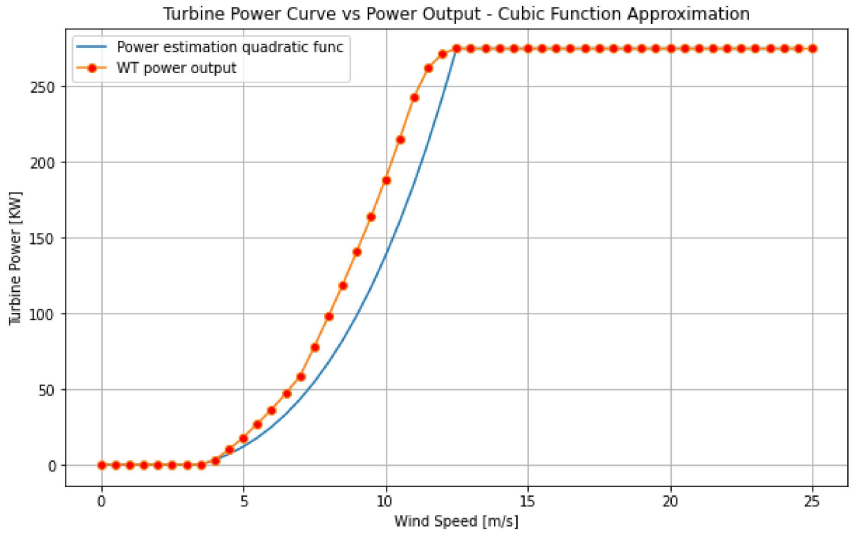
Defining cubic funtion

In [31]:
```
def func_wt_power_cubic(v, p_r, v_c=4, v_r= 12,  v_f=25):
    p_out = 0
    if (v >= v_c and v < v_r):
        a = p_r/(v_r**3-v_c**3)
        b = (v_r**3)/(v_r**3-v_c**3)
        p_out = (a*v**3)-(b*p_r) +p_r
    elif (v >= v_r and v <= v_f):
        p_out = p_r
    return p_out

print(func_wt_power_cubic(5, p_max))
```

```
10.081129807692264
```

In [32]:
```
# create an array of wind speed values from 0 to 25
v_array = np.arange(0, 25, 0.5)

#call quadratic function
q_v = [func_wt_power_cubic(v, p_max, v_cut_in, v_r, v_cut_off ) for v in v_array]

#q_v= for x in v func_quad(x.Wind_speed, p_max, v_cut_in, v_r, v_cut_off)
fig, ax = plt.subplots(figsize=(10,6))
plt.plot(v_array, q_v, label= "Power estimation quadratic func")
```

```
plt.plot(df_wt_curve.wind_speed, df_wt_curve.power_out, marker = 'o', markerfacecolor = 'r',
         label= 'WT power output')
plt.grid(True)
plt.title("Turbine Power Curve vs Power Output - Cubic Function Approximation")
#plt.xlim(0, 10)
#plt.ylim(2.5, 3.5)
plt.xlabel("Wind Speed [m/s]")
plt.ylabel("Turbine Power [KW]")
plt.legend()
plt.savefig("./outputs/wt_power_curve_cubic_approx_%s.png" % turbine_model, transparent=True)
plt.show()
```



Estimate power output using original dataframe

In [33]:
```
df_wind['P_out_cubic'] = df_wind.apply(lambda x: func_wt_power_cubic(x.Wind_Speed, p_max, v_cut_in, v_r, v_cut_off), axis=1)
```

In [34]:
```
df_wind_stats_cubic_func = df_wind.describe()

avg_wind_speed_cubic_func = df_wind_stats_cubic_func.Wind_speed_filter['mean']
std_wind_speed_cubic_func = df_wind_stats_cubic_func.Wind_speed_filter['std']
min_wind_speed_cubic_func = df_wind_stats_cubic_func.Wind_speed_filter['min']
med_wind_dir_cubic_func =df_wind_stats_cubic_func.Wind_dir_filter['50%']
max_wind_speed_wt_cubic_func = df_wind_stats_cubic_func.Wind_speed_filter['max']
```

In [35]:
```
energy_gen_year_cubic_func = sum(df_wind['P_out_cubic'])/1000  # to get value in MW

print ("WT Annual Energy Production: %.3f MWh" % (energy_gen_year_cubic_func))
```

```
WT Annual Energy Production: 135.685 MWh
```

In [36]:
```
cf_cubic_func = (energy_gen_year_cubic_func*1000) / (p_max*n_sample) # energy in MW, power in kWh

print ("Capacity Factor: %.3f" % (cf_cubic_func))
```

```
Capacity Factor: 0.056
```

In [37]:
```
results_cubic_func = {'method':'WT Power Curve- Cubic Eq. Approx.','min_wind_speed': min_wind_speed_cubic_func,
           'avg_wind_speed': avg_wind_speed_cubic_func, 'max_wind_speed': max_wind_speed_wt_curve,
           'std_wind_speed': std_wind_speed_cubic_func, 'median_wind_dir': med_wind_dir_cubic_func,
           'annual_generation': energy_gen_year_cubic_func,'capacity_factor':cf_cubic_func}

df_results = df_results.append(results_cubic_func, ignore_index=True)
df_results.T
```

Out[37]:

	0	1	2
annual_generation	126.818005	165.6334	135.684901
avg_wind_speed	3.718331	5.921592	5.921592
capacity_factor	0.052643	0.068756	0.056324
max_wind_speed	16.78	16.78	16.78
median_wind_dir	134.53	104.795	104.795
method	WT Power General Equation	Wind Power Curve	WT Power Curve- Cubic Eq. Approx.

	0	1	2
min_wind_speed	0.05	3.51	3.51
std_wind_speed	2.513775	1.936656	1.936656

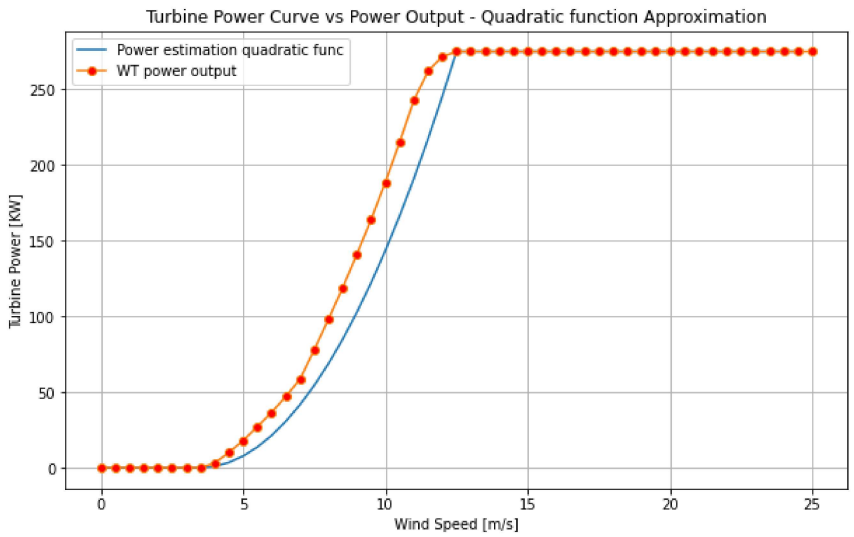
Activity: Quadratic Function

1. **Approximate the WT Power Generation Curve using a quadratic function**
2. **Estimate the Annual Energy Generation using the quadratic function from step 1.**
3. **Which method would you select to estimate the WT energy generation? Why?**

In [38]:
```
# defining quadratic function
def func_wt_power_quad(v, p_r, v_c=4, v_r= 12,  v_f=25):
    p_out = 0
    if (v >= v_c and v < v_r):
        p_out = p_r*(((v - v_c)/(v_r - v_c ))**2)
    elif (v >= v_r and v <= v_f):
        p_out = p_r
    return p_out

print(func_wt_power_quad(11, p_max))
```

```
210.546875
```

In [39]:
```
# create an array of wind speed values from 0 to 25
v_array = np.arange(0, 25, 0.5)

#call quadratic function
q_v = [func_wt_power_quad(v, p_max, v_cut_in, v_r, v_cut_off ) for v in v_array]

fig, ax = plt.subplots(figsize=(10,6))
plt.plot(v_array, q_v, label= "Power estimation quadratic func")
plt.plot(df_wt_curve.wind_speed, df_wt_curve.power_out, marker = 'o', markerfacecolor = 'r',
         label= 'WT power output')
plt.grid(True)
plt.title("Turbine Power Curve vs Power Output - Quadratic function Approximation")

plt.xlabel("Wind Speed [m/s]")
plt.ylabel("Turbine Power [KW]")
plt.legend()
plt.savefig("./outputs/wt_power_curve_quad_approx_%s.png" % turbine_model, transparent=True)
plt.show()
```



In [40]:
```
df_wind_stats_quad_func = df_wind.describe()

avg_wind_speed_quad_func = df_wind_stats_quad_func.Wind_speed_filter['mean']
std_wind_speed_quad_func = df_wind_stats_quad_func.Wind_speed_filter['std']
min_wind_speed_quad_func = df_wind_stats_quad_func.Wind_speed_filter['min']
med_wind_dir_quad_func =df_wind_stats_quad_func.Wind_dir_filter['50%']
max_wind_speed_wt_quad_func = df_wind_stats_quad_func.Wind_speed_filter['max']
```

Estimate power output using original dataframe

In [41]:

```
df_wind['P_out_quad'] = df_wind.apply(lambda x: func_wt_power_quad(x.Wind_speed, p_max, v_cut_in, v_r, v_cut_off), axis=1)
```

In [42]:
```
#Estimate annual energy generation
energy_gen_year_quad_func = sum(df_wind['P_out_quad'])/1000  # to get value in MW

print ("WT Annual Energy Production: %.3f MWh" % (energy_gen_year_quad_func))
```

```
WT Annual Energy Production: 128.016 MWh
```

In [43]:
```
##Extra: Show descriptive stats
cf_quad_func = (energy_gen_year_quad_func*1000) / (p_max*n_sample) # energy in MW, power in kWh

print ("Capacity Factor: %.3f" % (cf_quad_func))
```

```
Capacity Factor: 0.053
```

In [44]:
```
results_quad_func = {'method':'WT Power Curve- quad Eq. Approx.','min_wind_speed': min_wind_speed_quad_func,
           'avg_wind_speed': avg_wind_speed_quad_func, 'max_wind_speed': max_wind_speed_wt_curve,
           'std_wind_speed': std_wind_speed_quad_func, 'median_wind_dir': med_wind_dir_quad_func,
           'annual_generation': energy_gen_year_quad_func,'capacity_factor':cf_quad_func}

df_results = df_results.append(results_quad_func, ignore_index=True)
df_results.T
```

Out[44]:

	0	1	2	3
annual_generation	126.818005	165.6334	135.684901	128.016051
avg_wind_speed	3.718331	5.921592	5.921592	5.921592
capacity_factor	0.052643	0.068756	0.056324	0.053141
max_wind_speed	16.78	16.78	16.78	16.78
median_wind_dir	134.53	104.795	104.795	104.795
method	WT Power General Equation	Wind Power Curve	WT Power Curve- Cubic Eq. Approx.	WT Power Curve- quad Eq. Approx.
min_wind_speed	0.05	3.51	3.51	3.51
std_wind_speed	2.513775	1.936656	1.936656	1.936656

In [45]:
```
#Save wind dataframe with power out (only parametric results) in CSV file
df_wind.to_csv('./outputs/df_wind_power_%i_%i.csv' % (year, height))
```

In [46]:
```
#Save results in CSV file
df_results['year'] =  year
df_results['height'] =  height
df_results.to_csv('./outputs/wt_energy_gen_results_%i_%i.csv' % (year, height))
```

WT, Wind Speed, Wind Direction, and Height Analysis - Activity

(in Teams of 5 people):

1. **Use the client's location or your location and download the following data for the assigned year (2010-2014):**

- **windspeed_60m, windspeed_80m, windspeed_100m, winddirection_60m, winddirection_80m, winddirection_100m.**

1. **Report the descriptive statistics of the annual values of windspeeds, and power generation using the method you prefer. i.e. average wind speed at 100 meters in year x, power generation at 100 meters in year x. Use the data to answer the following questions:**

- **Does the average wind speed change as the height changes?**
- **Does the variability of the wind speed change as the height changes?**
- **Does the median value of the wind direction change as the height change?**
- **How does the power generation change as the height increases? (i.e. report the increase as a percentage as the height increases 20m or 40m)? Add a histogram plot of the power generation for one of the datasets (60m, 80m or 100m)**
- **Select one of the datasets and identify at what hour(s) of the day does the WT produces more energy. At what time(s) it produces less energy? (Hint: use the descriptive statistics grouped by hour)**
- **How does the annual energy generation and capacity factor change by height? (just report the values in a table or dataframe) and highlight the differences between the maximum and minimum values.**

In [ ]:
```

```