Mapana Journal of Sciences
2020, Vol. 19, No. 3, 47-58
ISSN 0975-3303|https://doi.org/10.12723/mjs.54.1

Modelling Analysis of Covid-19 Infections in India and Prediction of Daily Cases in 2021

N G Puttaswamy* and M N Anandaram†

Abstract

In this paper the data for daily confirmed new cases concerning the rise and fall of the Covid-19 (aka, coronavirus) pandemic infection in India for the nine month period starting from the first March 2020 has been subjected to a non linear least square fitting analysis using Gaussian, Skewed-Gaussian, Moffat, and Voigt model functions. The fitting parameters determined by the Python software package LMFIT are then used to compare the predicted remission times of Covid-19 pandemic during 2021. It is found that while the Gaussian, Skewed-Gaussian and Moffat models predict low levels by about March/April 2021; Voigt and other models predict longer times to reach same low endemic levels.

Keywords: Covid-19 data, Least square model fitting, prediction of fall times in 2021.

1. Introduction

Covid-19 has been a global pandemic since early 2020 that led to lockdowns in India and other countries in the world. Many countries are still showing no signs of slowing down [1]. In India also daily new confirmed cases have been increasing indicating that

* Professor of Physics (Retired), Bangalore University, Bangalore, India

† Professor of Physics (Retired), Bangalore University, Bangalore, India
mnanandaram@gmail.com

Covid-19 has become an epidemic since March 2020; however, since the middle of September 2020, the number of daily new cases has been coming down. While this is good news, we can expect that such a scenario will continue so that the daily new infection cases come down to a low level. Data regarding various aspects of covid-19 infection suitable for analysis has become available from many sources. We have selected the dataset concerning daily new cases arising from coronavirus infection in India from 1st March 2020 onwards as made available by the website of OurWorldInData [2]. We plan to estimate the time in 2021 when new cases of infection peter out to negligible or low endemic levels. To enable this determination we are applying suitable modelling functions to the Covid-19 data. These modelling functions are readily available and callable from the LMFIT software which is easily installable in any Python-3.7 based computational setup like Anaconda. A short description of the LMFIT software package is given below.

1.1. The LMFIT Python Program

LMFIT [2, 3] provides a high-level interface to non-linear optimization and curve-fitting problems. It builds on and extends many of the optimization methods of scipy's optimize. Initially inspired by (and named for) extending the **Levenberg-Marquardt** method from **scipy's optimize.leastsq() module**, LMFIT now provides a number of useful enhancements to optimization and data-fitting problems, including the use of **parameter** objects instead of plain floats as variables. A parameter has a value that can be varied during the fit or kept at a fixed value. It can have upper and/or lower bounds. A parameter can even have a value that is constrained by an algebraic expression of other parameter values. The parameter can also have attributes such as a standard error and can estimate fit uncertainties. LMFIT allows ease of changing fitting algorithms. Once a fitting model is set up, one can change the fitting algorithm used to find the optimal solution without changing the objective function. LMFIT has improved estimation of confidence intervals. LMFIT has functions to explicitly explore the parametric space and determine confidence levels even for the most difficult cases. Also, LMFIT allows improved curve-fitting by extending the capabilities of scipy's curve-fit, allowing one to parametrize a modelling-function and fit

data with that model. Further LMFIT includes many more readily usable built-in model functions for common line-shapes. **It is capable of determining all the fitting parameters starting from internally guessed values**. A brief description of the Covid19 data arrays, programming considerations and modelling functions is given in the next section.

2. Covid-19 Data, Programming and Model Functions

2.1 Description of Covid-19 Data and Programming Considerations

The nine-month duration Covid-19 data for India obtained from [2] starts from 1st March 2020 and ends on 30th November 2020. There are 275 pairs of data points obtained at the rate of one pair a day. These are listed as '**x** and **y arrays**' in the program listing given in the Appendix. The elapsed day array **x** consists of days elapsed from 1st March up to the 30th November running from 1 to 275. Similarly in the **y** array there are 275 daily values of new cases of coronavirus infections spanning a range of magnitudes from zero at the start to about 100,000 at the middle of September 2020 and then slowly declining. The raw data curve as plotted can be seen in all the figures shown below (in blue) and it also shows the fluctuations present in the Covid-19 data. It is this data that is subjected to LMFIT's model curve fitting algorithm. A Python script was written on the basis of guidelines provided by the LMFIT documentation [9] to select one of the modelling functions (described below) and carry out the least square minimized curve fitting. The essential statements that do the job for each selected model are given below.

Table 2.1: LMFIT Usage Script for best fit Covid-19 data Modelling

```
X = np.linspace( 1, 275, 275);   Y = np.array( [275  Covid data values])
model = GaussianModel()   # Select a Model from the Lmfit builtins
pars = model.guess(y=Y, x=X);  #print("\nGaussian.pars =",pars,"\n")
Mdl = model.fit( y=Y, pars, x=X )  # best fit param values array out
print("Gauss Model Fit Report:\n",Mdl.fit_report(min_correl=0.25)")
y_fit = Mdl.best_fit   #  best curve_fit ready to plot over raw data
```

```
x_pred = np.linspace(275.,356.0, 82)  # day points for extrapolation
y_pred = Mdl.eval(x=x_pred)  # predicted Covid-19 infection
plt.figure(figsize=(8,6),dpi=150)  # figure drawing statements follow
```

The complete program listing is given in the Appendix; The fitted curve is plotted in all the figures (shown under each modelling function) as solid black lines stopping at the last data point; both linear and semilog (y-axis) plots are provided. The relevant best curve fitting parameters have been tabulated along with the names of the four modelling functions in Table 3.1. In addition the same parameters are used to extend and predict model curve for a specified number of days beyond the last data point; this part of each curve is shown as dashed black lines in both the linear and semilog plots. The advantage of semilog plot is that the days on which the new cases of Covid-19 infection has reached specified low values of 1000, 500, and 100 cases per day can be easily read off. These are tabulated in Table 3.2 along with corresponding dates for all the modelling functions used. Now the four better fitting models are defined below starting with the Gaussian model and each model function is followed by two plots as already explained above.

2.2 Gaussian model fitting

A model based on a Gaussian or normal distribution line-shape (aka, bell curve) [5] has the function parameters amplitude $\boldsymbol{A}$, center $\boldsymbol{\mu}$, and sigma $\boldsymbol{\sigma}$. In addition, parameters **fwhm** (full width at half maximum) and **height** are included as constraints to report fwhm and maximum peak height, respectively. The model equation is given by

$$\boldsymbol{f(x, A, \mu, \sigma) = \frac{A}{\sigma\sqrt{2\pi}} exp\left(-(x-\mu)^2/2\sigma^2\right)}$$

The full width at half maximum is $\mathbf{fwhm} = \boldsymbol{2\sigma\sqrt{2ln(2)}} \cong \boldsymbol{2.3548\sigma}$. The Gaussian model function is symmetric about the centre.

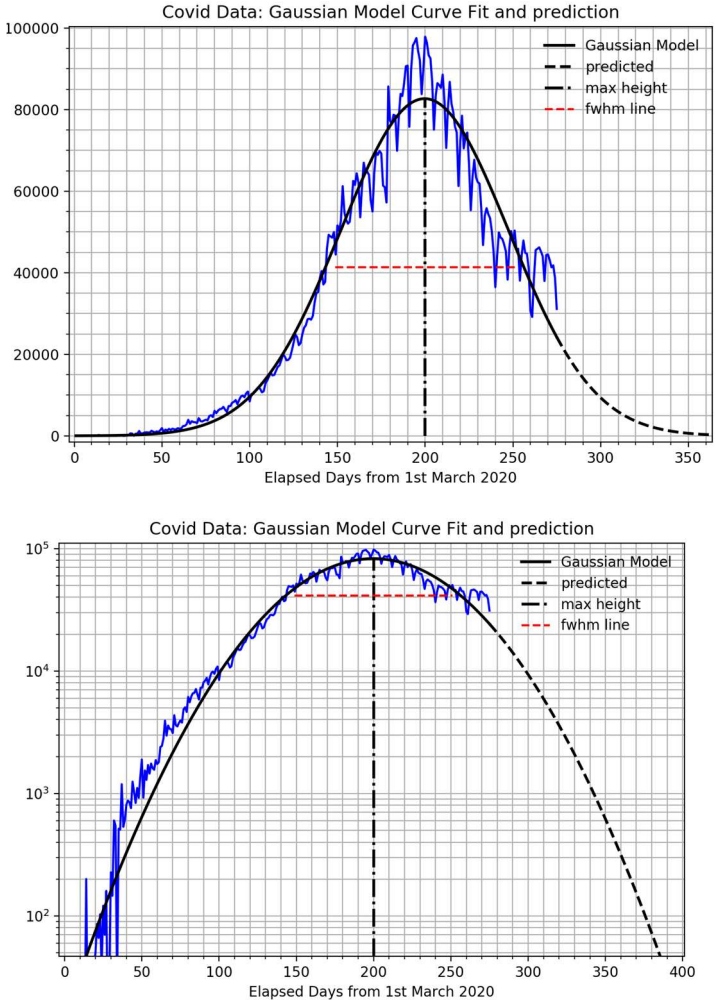

Figure 2. 2 : Gaussian modelled Covid-19 data as linear (top) and semilog plot (bottom).

2.3 Skewed-Gaussian model fitting (aka, Skew-Normal Model)

This model based on a skewed form of the Gaussian or normal distribution lines-shape [6] has the parameters **amplitude A**, **center** μ , **sigma** σ and **the skewness** constant **gamma** γ. Also the parameters **fwhm** and **peak height** are included as constraints to report full width at half maximum and maximum peak height, respectively.

$$f(x, A, \mu, \sigma) = \frac{A}{\sigma\sqrt{2\pi}} exp\left(-(x-\mu)^2/2\sigma^2\right)\{1 + erf\left(\gamma(x-\mu)/\sigma\sqrt{2}\right)\}$$

where, ***erf()*** is the error function callable from scipy modules. This Model fit gives a skewness index value of $\gamma = -1.59$.

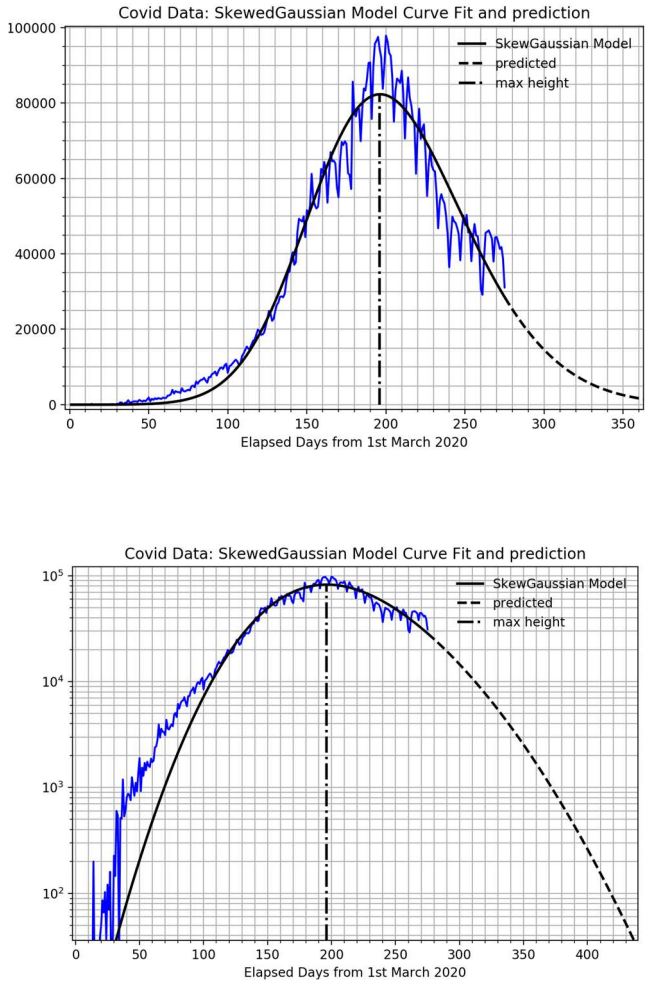

Figure 2.3 : Skewed-Gaussian modelled Covid-19 data as linear (top) and semilog plot (bottom).

2.4 Moffat model fitting

A model based on the Moffat distribution function [7] which has four parameters **amplitude (A), center (μ), a width parameter sigma (σ), and an exponent (β).** In addition, the parameters **fwhm and height** are included as constraints to report full width at half maximum and maximum peak height, respectively.

$$f(x, A, \mu, \sigma, \beta) = A\left(1 + \left((x - \mu)^2/\sigma^2\right)\right)^{-\beta}$$

The full width at half maximum is $fwhm = 2\sigma\sqrt{2^{(1/\beta)} - 1}$.

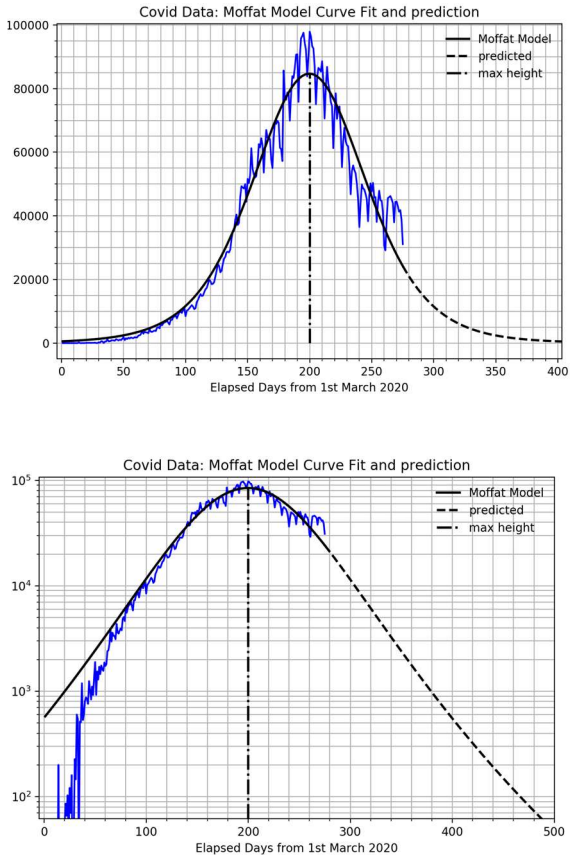

Figure 2.4: Moffat modelled Covid-19 data as linear (top) and semilog plot (bottom).

2.5 Voigt model fitting

A model based on a Voigt distribution function [8] with four Parameters: **amplitude A, center μ, sigma σ, and gamma γ.** By default, gamma is constrained to have a value equal to sigma, though it can be varied independently. In addition, arameters **fwhm** and **height** are included as constraints of fwhm and maximum peak height, respectively. The definition for the Voigt function used here is

$$f(x, A, \mu, \sigma, \gamma) = A\,|w(z)|/\left(\sigma\sqrt{2\pi}\right)$$

where, $z = (x - \mu + i\gamma)/(\sigma\sqrt{2\pi})$ and $w(z) = exp\left(-z^2\right) erfc(-iz)$ and erfc() is the complementary error function. Then $\mathbf{fwhm} \cong \mathbf{3.6013}\sigma$.

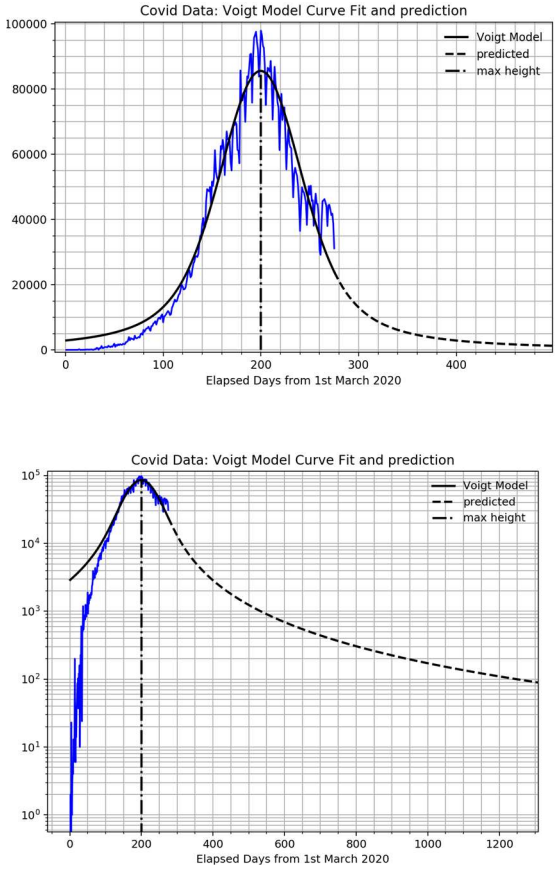

Figure 2.5 : Voigt modelled Covid-19 data as linear (top) and semilog plot (bottom).

In addition to the above, LMFIT [3] provides many more readily usable models. Some of them have also been used to model the data but are not been reported here as their predictions require more years for new coronavirus infections to reach negligible levels.

3. Discussion and conclusions

The Gaussian modelled linear plot (Fig. 2.2) has the symmetrical bell shaped profile whereas the Skewed-Gaussian modelled plot (Fig.2.3) shows longer falling times for the daily new coronavirus infection cases. This is reflected in the shorter times required for the infection to fall to 1000, 500, and 100 daily new case levels for the Gaussian model and about 30 to 40 days longer for the Skewed-Gaussian model as shown in the second and third rows respectively in Table 3.2. Thus these two models indicate that the coronavirus infection reaches around 1000 daily new infection levels by March/April 2021.

Table 3.1. LMFIT Least-Square Minimized Parameters for Selected Peak-like Models on Covid-19 Data from 1st March 2020

Model	Amplitude	height	μ	σ	fwhm	γ or β
Gausssian	9957334.4	82703.1	199.7	48.0	113.1	----------
SkewGauss	10232772.6	59223.1	158.9	68.9	162.3	$\gamma = -1.59$
Moffat	84669.1	84669.1	199.7	120.4	107.8	$\beta = 3.79$
Voigt	11881198.2	85556.5	199.7	29.0	104.4	$\gamma = 29.0$

Both the Moffat (Fig. 2.4) and the Voigt models (Fig. 2.5) of coronavirus infection also have some symmetry, but as the 3rd and 4th rows of Table 3.2 indicate, they imply much longer times are needed to reach the stated corresponding levels.

Table 3.2. Name of Model	Elapsed Days from 1st March 2020 for Covid-19 Densities (new infection cases in India) of 1000	500	100
Gaussian	343 days =	354 days =	376 days =
Dates →	Feb 07, 2021	Feb 18, 2021	Mar 11, 2021
Skewed-			
-Gaussian	372 days =	387 days =	418 days =
Dates→	Mar 10, 2021	Mar 25 2021	Apr 25, 2021
Moffat	380 days =	405 days =	465 days =

Dates →	Mar 15, 2021	Apr 10, 2021	Jun 09, 2021
Voigt	538 days =	677 days =	1255 days =
Dates →	Aug 20, 2021	Jan 07, 2022	Aug 07, 2024

Thus the Covid-19 infection levels in the Moffat model take about 3 to 4 months longer than the Gaussian model and, in the case of Voigt model, over 3 years are required. Some of the other models not reported here require even longer times to reach similar levels.

There is a possibility that the coronavirus infection may continue to exist at low levels as an endemic disease like many other viral diseases such as HCV and AIDS. It is thus necessary to study actual new infection data for the next few months to get better predictions for 2021.

References

[1] https://ourworldindata.org/coronaviruss
[2] https://ourworldindata.org/coronavirus-data-explorer?zoomToSelection=true&time=2020-02-07..latest&country=~IND&region=World&casesMetric=true&interval=daily&hideControls=true&smoothing=0&pickerMetric=total_cases&pickerSort=desc
[3] https://lmfit.github.io/lmfit-py/index.html
[4] https://lmfit.github.io/lmfit-py/builtin_models.html
[5] https://en.wikipedia.org/wiki/Normal_distribution
[6] https://en.wikipedia.org/wiki/Skew_normal_distribution
[7] https://en.wikipedia.org/wiki/Moffat_distribution
[8] https://en.wikipedia.org/wiki/Voigt_profile
[9] https://lmfit-py.readthedocs.io/en/stable/fitting.html

Appendix

```
import numpy as np
import matplotlib.pyplot as plt
From lmfit.models import GaussianModel, Skewed Gaussian Model,
MoffatModel, VoigtModel
# COVID-19 Xdata (275 days) and Ydata from 1st March to 30th Nov
2020
#Source: https://ourworldindata.org/coronavirus-data-explorer?
x = np.linspace(1.0, 275.0, 275);
```

```
# Y array numbers below are in 1000s and so multiplied by 1000 at the
end *
y = np.array([
 0.0, 0.002,0.0, 0.023,0.002, 0.001,0.003, 0.005,0.004, 0.013, 0.006,
 0.011, 0.009,0.20, 0.011,0.006, 0.023,0.014, 0.038,0.050, 0.086,0.066,
 0.103,0.037, 0.121, 0.07, 0.16, 0.01, 0.037, 0.227, 0.146,
 0.601, 0.545,0.024, 0.515,0.506,1.19, 0.533,0.605, 0.809,0.873,
 0.848,0.759, 1.248,1.034, 0.835,1.108, 0.922, 1.37,1.893, 0.924,
 1.541, 1.29, 1.707,1.453,1.753,1.607,1.561,1.873,1.738, 1.801,
 2.394,2.442, 2.806,3.932,2.963,3.587, 3.364,3.344,3.113, 4.353,3.607,
 3.524,3.763, 3.942,3.787, 4.864,5.05, 4.63,6.147, 5.553,6.198, 6.568,
 6.629, 7.113, 6.414, 5.843, 7.293, 7.30, 8.105, 8.336, 8.782,
 7.761, 8.821, 9.633, 9.889,
 9.471,10.438,10.864,8.442,10.218,10.459,10.93,11.458,11.929,11.502,
10.667,10.974,12.881,13.586,14.516,15.403,14.831,14.933,15.968,16.922,
 17.296,18.552,19.906,19.459,18.522,18.641, 19.160,20.903,22.771,24.85,
24.248,22.251,22.753,24.879,26.506,27.114,28.606,28.732,28.498,29.429,
 32.676,34.975,35.252,38.697, 40.425,37.132,37.74,45.72,49.31,48.916,
 48.611,49.981,44.457,51.596,50.294,52.783,61.242, 54.735,52.972,52.05,
 52.509,56.282,62.538,61.537,64.399,62.064,53.601,
 60.963,66.999,64.553, 64.732,64.03,57.711,55.018,64.572,69.672,68.90,
 69.876,69.239,61.408, 0.975,57.224,85.687,77.266,76.472,78.761,78.512,
 69.921,78.357,83.883, 3.341,86.432,90.632,90.802,75.809,89.706,95.735,
 96.551, 97.57,94.372, 2.071,83.809,90.123,97.894,96.424,93.337,92.605,
 86.961, 75.083,83.347, 86.508,86.052,85.362,88.60,82.17,70.589,80.472,
86.821,81.484,79.476, 5.829,74.442,61.267,72.049,78.524,70.496,73.272,
74.383, 66.732,55.342, 63.509,67.708,63.371,62.212,61.871,55.722,46.79,
 54.044, 55.839,54.366, 53.37,50.129,45.148,36.47,43.893,49.881,48.648,
 48.268,46.963,45.231, 38.310,46.253,50.21,47.638,50.356,45.674,45.903,
 38.073,44.281,47.905, 44.879,44.684,41.10,30.548,29.163,38.617,45.576,
 45.882, 46.232,45.209, 44.059,37.975,44.376,44.489,43.082,41.322,
 41.81, 38.772, 31.118 ] ) * 1000.0
#for ii in range (len(y)): print(ii,"-->", x[ii],"-->",y[ii])
# Gaussian model
model = GaussianModel()
pars = model.guess(y, x=x);  #print("\nGaussian.pars =",pars,"\n")
Mdl = model.fit(y, pars, x=x)
print("Gaussian Fit
Report:\n",Mdl.fit_report(min_correl=0.25),"#"*75,"\n")
y_fit=Mdl.best_fit;imx=np.argmax(y_fit);xmax,ymax=x[imx],y_fit[imx]
fwhm = 102.277310; hwhm, ymx_2 = fwhm/2, ymax/2
x_pred = np.linspace(275.,420.0, 146)  # points for extrapolation
y_pred = Mdl.eval(x=x_pred)  # predicted using best_fit params
```

```
plt.figure(figsize=(8,6),dpi=180)
plt.plot(x, y, 'b-')
plt.plot(x, y_fit, 'k-', lw=2,label='Gaussian Model')
plt.plot(x_pred,y_pred,"k--",lw=2,label="predicted")
plt.plot([xmax,xmax],[0.0,ymax],"k-.",lw=2,label="max height")
plt.plot([xmax-hwhm,xmax+hwhm],[ymx_2,ymx_2],"r--",label="fwhm
line")
plt.grid(which="both"); #plt.yscale("log")
plt.minorticks_on(); plt.legend(loc='best',frameon=False)
plt.xlabel("Elapsed Days from 1st March 2020")
plt.title("Covid Data: Gaussian Model Curve Fit and prediction")

# SkewedGaussian model
model = SkewedGaussianModel()
pars = model.guess(y, x=x);  #print("\nSkewedGaussian.pars
=",pars,"\n")
Mdl = model.fit(y, pars, x=x)
print("SkewGaussModel Fit Report:\n", Mdl.fit_report
(min_correl=0.25), "="*75,"\n")
y_fit = Mdl.best_fit; imx = np.argmax(y_fit); xmax,ymax =
x[imx],y_fit[imx]
x_pred = np.linspace(275.,460.0, 186)  # prediction points for
extrapolation
y_pred = Mdl.eval(x=x_pred)  # predicted using best_fit params
plt.figure(figsize=(8,6),dpi=180)
plt.plot(x, y, 'b-')
plt.plot(x, y_fit, 'k-',lw=2, label='SkewGaussian Model')
plt.plot(x_pred,y_pred,"k--",lw=2,label="predicted")
plt.plot([xmax,xmax],[0.0,ymax],"k-.",lw=2,label="max height")
plt.grid(which="both"); #plt.yscale("log")
plt.legend(loc='best',frameon=False); plt.minorticks_on();
plt.xlabel("Elapsed Days from 1st March 2020")
plt.title("Covid Data: SkewedGaussian Model Curve Fit and
prediction")
plt.show()
# Similar coding may be done for omitted LMFIT Model functions
here
```