Report on the week of Nov 7

1. Writing a document about the Levenberg-Marquardt

2. Reading some pages of numerical recipes to better understand the details
3. The document will be updated in the future and become a part of my thesis

Checking the Levenberg-Marquardt

- It worked fine in the covariance matrix convergence test but failed the test of chi-square variance difference
- So, I came back to find the bug
- An important point is that it works very well with simple functions (e.g., gaussian)
- But when it comes to ARES, it fails.
- So, the issue probably comes from the function that calls ARES and calculates it's local derivatives.

Why do we need a local derivative ?

- We need the derivative of our model to calculate the Jacobian Matrix (which later becomes the covariance matrix)
- But while working with ARES, we do not have the analytical derivation
- That's why we need to compute the local derivative

$$\left(\mathbf{J}^{\mathrm{T}}\mathbf{J} + \lambda\mathbf{I}\right) dm = \mathbf{J}^{\mathrm{T}}[\mathbf{y} - \mathbf{f}(m)]$$

This is how we calculate the local derivative.

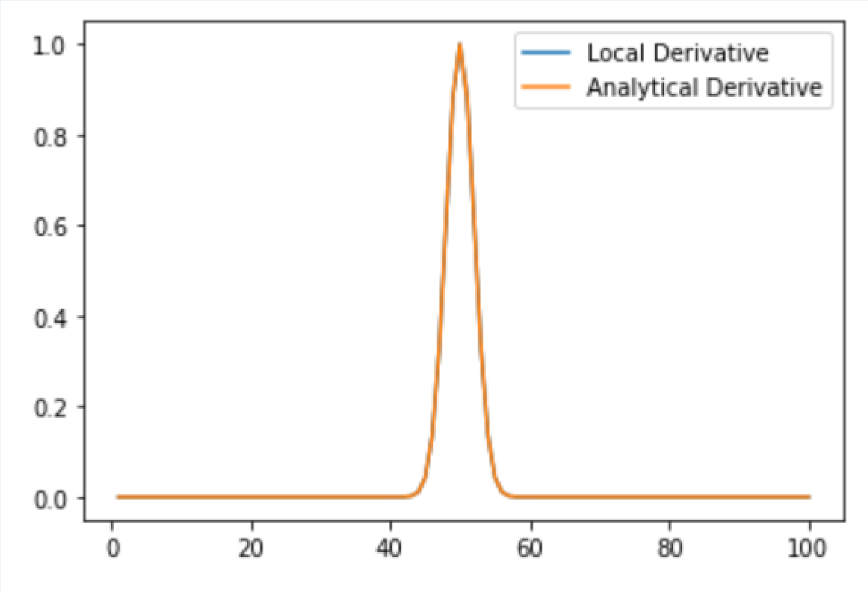
$$A_{x_i} = \frac{A(x_i + \delta x_i) - A(x_i - \delta x_i)}{2\delta x_i}$$

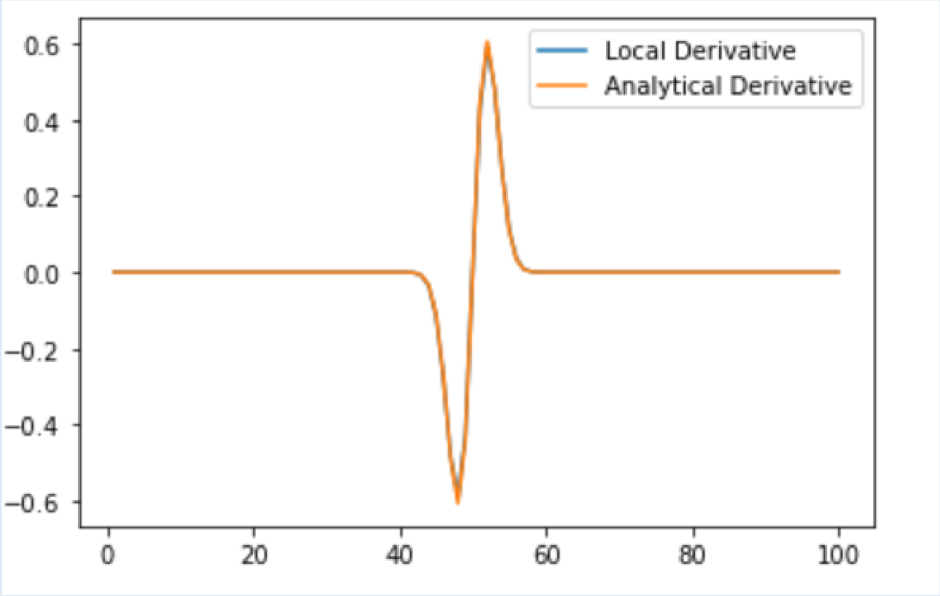
- Read about it a bit
- Tried different variations of how to improve this function
- When I reached the best version, I tried checking it with different functions that I know their analytical derivative, so I can compare these two.
- Surprisingly, I found that it does not work very well with some types of functions like periodic ones.
- But it works fine when it comes to functions similar to gaussian

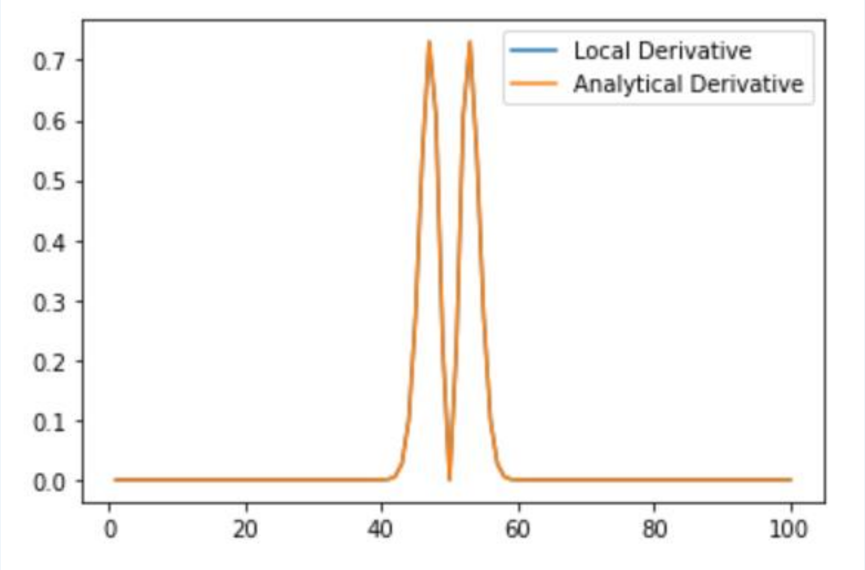
```python
def local_dev(m, x, func, d = 100):
    m = np.array(m)
    y, derivs_a = func(m , x)
    derivs = np.zeros([len(x), len(m)])
    dpars = np.zeros(len(m))
    dpars=m/d
    for i in range(len(m)):
        pars_plus = np.array(m, copy=True, dtype = 'float64')
        pars_plus[i] = pars_plus[i] + dpars[i]

        pars_minus = np.array(m, copy=True, dtype = 'float64')
        pars_minus[i] = pars_minus[i] - dpars[i]

        A_plus, a = func(pars_plus , x)
        A_minus, b = func(pars_minus , x)
        A_m = (A_plus - A_minus)/(2*dpars[i])
        derivs[:, i] = A_m
    return y, derivs_a, derivs
```

Gaussian

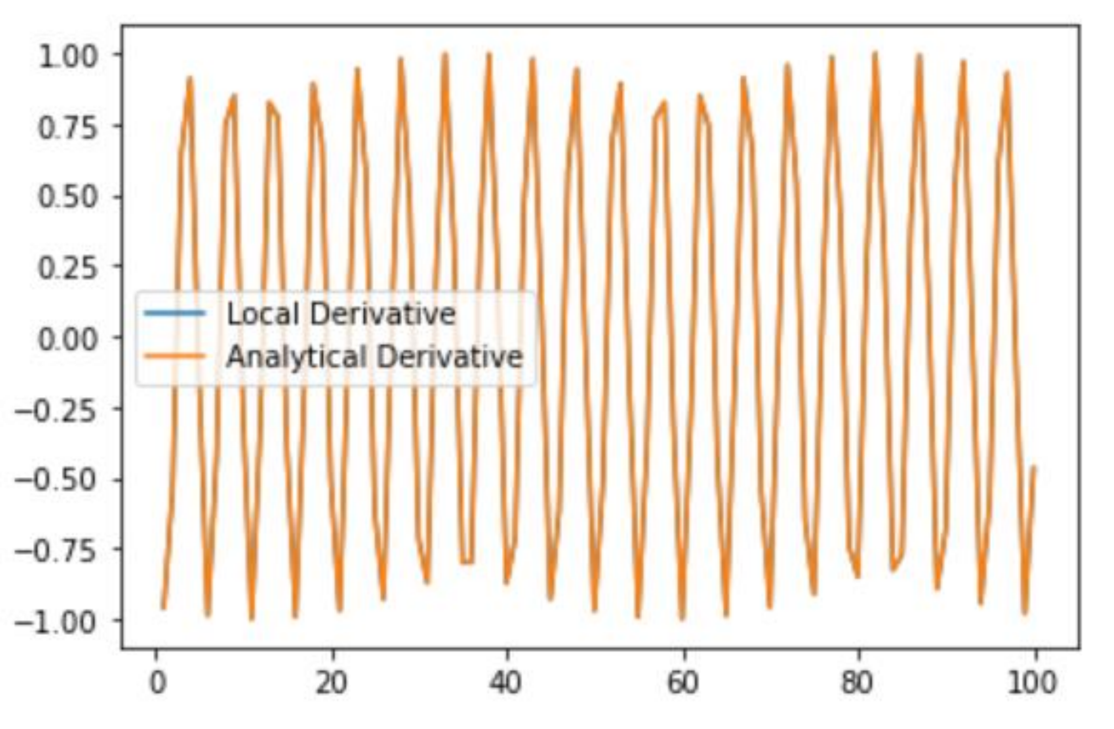
$$y = a + b \exp \frac{(x - x0)^2}{2sig^2}$$



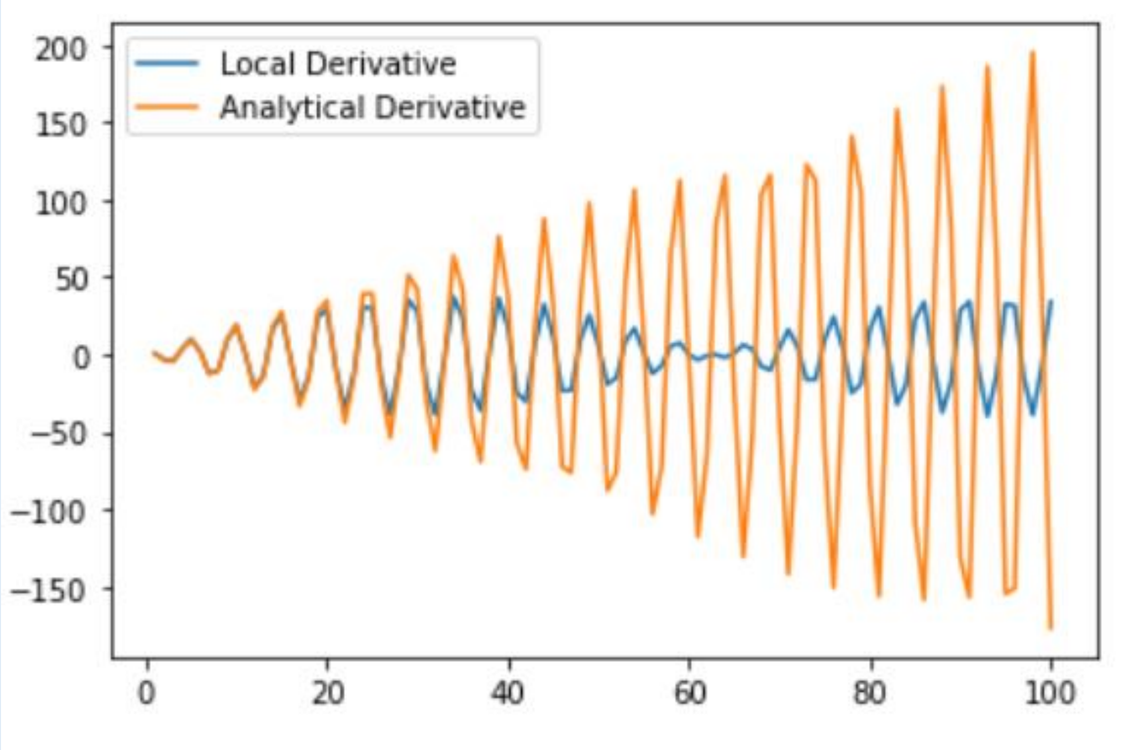




$$y = a + b \sin cx$$



Local Derivative
Analytical Derivative
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
0
20
40
60
80
100



Local Derivative
Analytical Derivative
200
150
100
50
0
−50
−100
−150
0
20
40
60
80
100