Jigar Zanzarukiya - 201801224
Hiren Chaudhary - 201801438
Naman Dave - 201801439
Deep Patel - 201801443

PROPOSAL

1) **Newton Raphson Method, Finding maxima or minima of a function, then methods to find maxima and minima explicitly, and then implementing Gradient Descent using Newton Raphson Method (Matlab):** Newton Raphson Method is used to find roots of a function by iterative manner, we can use the same technique to find maxima or minima of a function. Furthermore, we can also modify this method to find only the minima as well as the maxima. And then onwards we implement the most widely used optimizer for DL newralnets, i.e. Gradient Descent which is also a slight change in Newton Raphson Method with a few assumptions and some additional parameters which are called as hyperparameters.
2) **F1 race car tuning (Using Secant Method):** Racing isn't just speed, however it is the most important when the road is straight but in normal F1 races the roads are curvy. So the car's efficiency also depends upon two other factors; 1) Acceleration, 2) Handling. Now an F1 race car is finely tuned when the max-speed efficiency equals the handling efficiency times 10 plus acceleration times 0.23. Our goal is to find the point where the tuning happens (if multiple tuning points then find the optimum one) (Matlab). As the efficiency curves' function may not be differentiable at all the points, so we can not user methods involving derivatives, therefore we use Secant Method
3) **Comparison of Polynomial Interpolation (Lagrange and Newton's interpolation) and Polynomial Regression by accuracy, sensitivity and other comparison matrices:** Comparison of completely fitting curve vs. best fitting hyperplane (Python: ML). First we consider some data points and we fit our polynomial using these methods and then we predict the datapoint with a given set of attributes that will give us the accuracy. Also adding errors will give the approx idea about the sensitivity of methods.
4) **Cholesky Decomposition Compression of a symmetric matrix vs. splitting compression of symmetric matrix. Computational time comparison using GPU operations (Matlab/Python: GPU):** Compression of a symmetric matrix can be done using two methods; 1) Cholesky Decomposition, 2) Consider only the lower triangle with diagonal (Splitting). Now retrieving original matrices, we have one method for the Cholesky Decomposition which is to perform matrix multiplication. While retrieving splitted matrix, we have three methods; 1) Two Loops, 2) Add matrix with its transpose and half the diagonal, 3) Decompose the compressed matrix with two matrices; a lower triangle matrix with zeros on diagonal and a diagonal matrix. Add the lower triangular matrix with its transpose and add the diagonal matrix. Now the main performance criteria we consider here is the time taken by these methods, because the time taken also depends upon the hardware selection, as the Cholesky decomposition takes GPU/TPU as a primary hardware which is optimized for matrix multiplications and the rest may or may not use GPU, our goal is to compare all the methods on retrieving time and give the final results.

max_speed(x)

$$= 235\left(\left(\frac{e^{0.1(x-50)}}{e^{0.1(x-50)}+1}\right) + 0.05sin(x) - \left(0.1e^{-|x-10|} + 0.2e^{-|x-21|} + 0.3e^{-|x-35|} + 0.3e^{-|x-50|} + 0.2e^{-|x-80|}\right)\right)$$

```
max_handling_x = 60;
handling = @(x) (5.519) * ((x <= max_handling_x) .*
exp(0.2*(x-max_handling_x/2.3))./(exp(0.2*(x-max_handling_x/2.3)) + 1) + (x
> max_handling_x) .* exp(-0.07*(x - max_handling_x))) -
(0.2*exp(-abs(x-10)) + 0.2*exp(-abs(x-21)) + 0.3*exp(-abs(x-35)) +
0.3*exp(-abs(x-50)) + 0.2*exp(-abs(x-80)));
```

$$x_n = x_{n-1} - \frac{g(x_{n-1})}{g'(x_{n-1})}$$

$$x_n = x_{n-1} - \frac{f'(x_{n-1})}{f''(x_{n-1})}$$

$$e_{n+1} = \frac{e_n^2}{2}\left|\frac{f'''(x_n)}{f''(x_n)}\right|$$

handling(x)

$$= 5.519\left(\begin{cases}\left(\frac{e^{0.2(x-max_{handling})}}{e^{0.2(x-max_{handling})}+1}\right) & \text{if } x \le max_{handling} \\ e^{-0.07(x-max_{handling})} & \text{if } x > max_{handling}\end{cases} - \left(0.1e^{-|x-10|} + 0.2e^{-|x-21|} + 0.3e^{-|x-35|} + 0.3e^{-|x-50|} + 0.2e^{-|x-80|}\right)\right)$$

acceleration(x)

$$= \frac{1}{1.04\,x\,10^{-3}}\left(1 - e^{-0.15x} + 10^{-7}e^{e^{0.01x}} + 0.02sin(x)\right)$$

$$f(x) = (x-2)^2$$

$$f(x) = x^3 + (x-3)^2$$

$$f(x) = x^4 - 8x^2 + 3$$

$$X_{n+1} = X_n - \frac{f'(X_n)}{|f''(X_n)|}$$

$$X_{n+1} = X_n - \frac{f'(X_n)}{-|f''(X_n)|}$$

$$X_{n+1} = X_n - LR * f'(X_n)$$

$$\Omega = \frac{1}{\pi} Re \int_0^{\pi} log\left(\frac{e^{e^{it}} + e^{it}}{e^{e^{it}} - e^{it}}\right) dt$$

$$e = \frac{\pi}{\int_0^1 \frac{sin(\pi x)}{x^x (1-x)^{1-x}} dx}$$