Lab Assignment 02

Name : Abdelaziz Mohamed Abdelaziz

I.D : 19015941

Part 1:

Problem Statement:

The following graph represents dependency of the scaled efficiency Eγ (p) = Sγ (p)/p parameterized with γ = p^δ. The parameter δ is sampled from the interval [0, 1] referring to Amdahl' law for δ = 0 and Gustafson's law for δ = 1. The six

curves in the p–δ plane are projected iso-efficiency lines of Eγ=p^δ

(p). Obviously, we have to significantly increase the degree δ of the functional dependency of the scaling ratio γ = p^δ, in order to preserve efficiency when increasing the number of processing units p.

Answer:

//code

```
def scaled_efficiency(p, delta, f):
    return (f + (1 - f) * p ** delta) / (p * f + (1 - f) * p ** delta)
```

//Explantion

scaled_efficiency(p, delta, f):

Input: p is the number of processing units, delta is a parameter, and f is a constant.

Output: Calculates and returns the scaled efficiency using the formula (f + (1 - f) * p ** delta) / (p * f + (1 - f) * p ** delta). This function essentially computes the scaled efficiency for given input parameters.

//code

```
def create_meshgrid(start, end, num_points):
    return np.linspace(start, end, num_points)
```

//Explantion

create_meshgrid(start, end, num_points):

Input: start is the starting value, end is the ending value, and num_points is the number of points.

Output: Generates and returns a 1D array representing a linearly spaced grid of values between start and end with num_points points. This function is used to create meshgrid values for P and delta.

//code

```
def plot_3d_surface(ax, p, delta, scaled_eff):
    ax.plot_wireframe(p, delta, scaled_eff, color='blue', rstride=10, cstride=10)
```

//Explantion

plot_3d_surface(ax, p, delta, scaled_eff):

Input: ax is the Axes3D object, p, delta, and scaled_eff are the meshgrid values and scaled efficiency values.

Action: Plots a 3D wireframe surface plot of scaled efficiency on the provided Axes3D object. The surface represents the relationship between P, delta, and scaled efficiency.

//code

```
def plot_contour_lines(ax, p, delta, scaled_eff, levels):
    contour = ax.contour(p, delta, scaled_eff, levels=levels, offset=0,
cmap='coolwarm')
    ax.clabel(contour, contour.levels, inline=True, fontsize=10)
```

//Explantion

plot_contour_lines(ax, p, delta, scaled_eff, levels):

Input: ax is the Axes3D object, p, delta, and scaled_eff are the meshgrid values and scaled efficiency values, and levels are the contour levels.

Action: Plots contour lines on the provided Axes3D object based on the meshgrid values and scaled efficiency. Contour lines represent constant values of scaled efficiency.

//code

```
def set_labels_and_title(ax):
    ax.set_xlabel('Number of Processing Units P')
    ax.set_ylabel('Delta δ')
    ax.set_zlabel('Scaled Efficiency  Eγ(p)')
    ax.set_title('Functional Dependency of Scaled Efficiency')
```

//Explantion

set_labels_and_title(ax):

Input: ax is the Axes3D object.

Action: Sets labels for the x, y, and z axes, as well as the title for the plot.

//code

```
def set_axis_limits_and_ticks(ax):
    ax.set_zlim(0, 1)
    ax.set_zticks(np.arange(0, 1.2, 0.2))
    ax.set_xlim(0, 20)
    ax.set_xticks(np.arange(0, 21, 5))
```

//Explantion

set_axis_limits_and_ticks(ax):

Input: ax is the Axes3D object.

Action: Sets limits and ticks for the x, y, and z axes.

//code

```
def customize_plot(ax):
    ax.view_init(elev=25, azim=-60)
    ax.dist = 12
```

//Explantion

customize_plot(ax):

Input: ax is the Axes3D object.

Action: Customizes the plot by setting the view angle and distance.

//Code:

```
p_values = create_meshgrid(1, 20, 100)
delta_values = create_meshgrid(0, 1, 100)
p, delta = np.meshgrid(p_values, delta_values)
f = 0.1
scaled_eff = scaled_efficiency(p, delta, f)
fig = plt.figure(figsize=(10, 10))
ax = fig.add_subplot(111, projection='3d')
plot_3d_surface(ax, p, delta, scaled_eff)
efficiency_levels = np.linspace(0, 1, 11)
plot_contour_lines(ax, p, delta, scaled_eff, efficiency_levels)
set_labels_and_title(ax)
set_axis_limits_and_ticks(ax)
customize_plot(ax)
plt.show()
```

//Explantion

Generating Meshgrid Values and Calculating Scaled Efficiency:

The code generates meshgrid values for P and delta using the create_meshgrid function.

It calculates scaled efficiency using the scaled_efficiency function with the generated meshgrid values.

Creating and Customizing the 3D Plot:

It creates a 3D plot using Matplotlib and adds a subplot with projection='3d'.

The 3D surface is plotted using the plot_3d_surface function.

Contour lines are added using the plot_contour_lines function with specified levels.

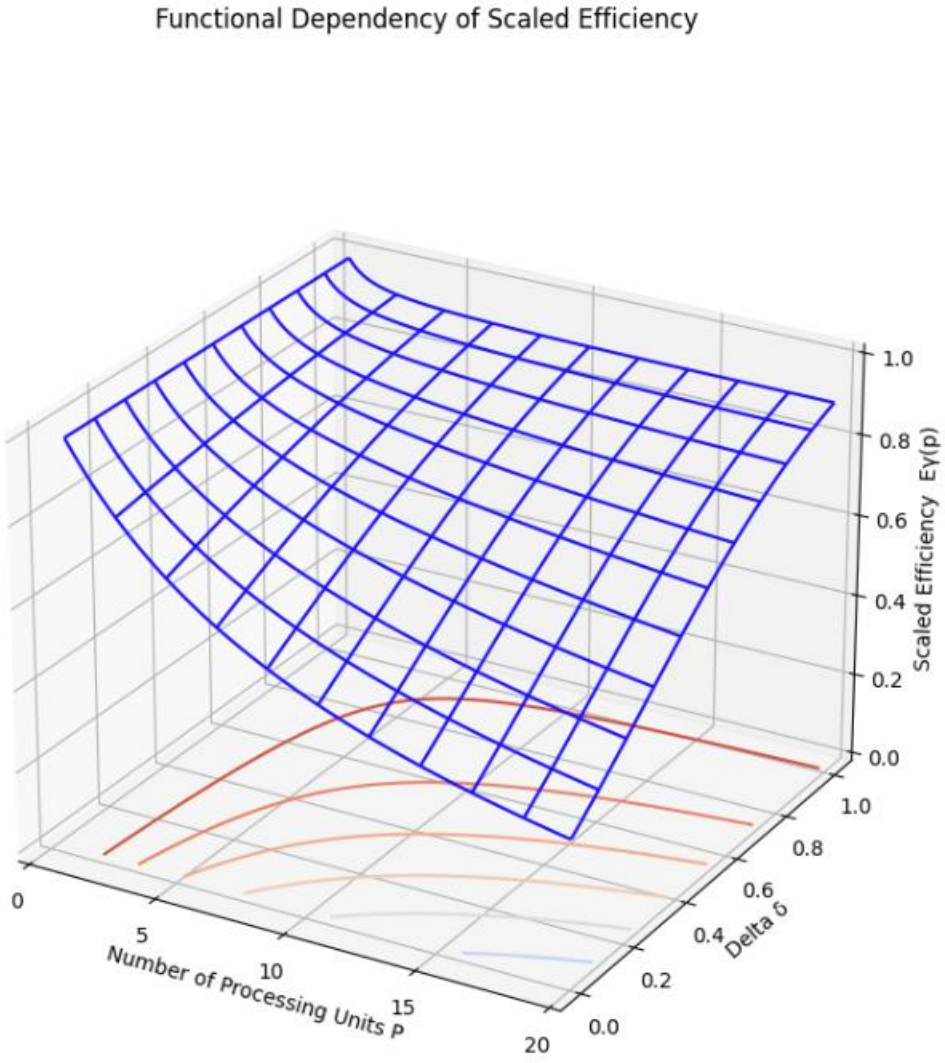
Labels, title, axis limits, and ticks are set using the set_labels_and_title and set_axis_limits_and_ticks functions.

The plot is customized with a specific view angle and distance using the customize_plot function.

Displaying the Plot:

Finally, the plot is displayed using plt.show().

Output Image :



Link To Colab:

https://colab.research.google.com/drive/1L8E-O_w8jnwPjIhUF00aNqUeF-uiwXuR?usp=sharing