Robot arm

1) Implement 3-segment robot arm.
I've implemented a dynamic N-segment robot arm by slightly prettyfying initial code.

$$\frac{L_{MSE}}{\delta y'} = \frac{(y-y')^2}{\delta y'} = \frac{(y-y')^2}{\delta (y-y')} \cdot \frac{(y-y')}{\delta y'} = -2 \cdot (y-y') \cdot \frac{y'}{\delta y'}$$

```
def rotation(theta: float):
    """
    :param theta: in radians
    :return:
    """
    c = np.cos(theta)
    s = np.sin(theta)

    return np.array((
        (c, -s),
        (s, c),
    ))


def d_rotation(theta: float):
    """
    :param theta: in radians
    :return:
    """
    c = np.cos(theta)
    s = np.sin(theta)

    return np.array((
        (-s, -c),
        (c, -s),
    ))
```

2) Implemented multi-segment robot arm:

```
prev_r = None

for segment_idx in range(SEGMENT_COUNT):
    # getting rotation value
    theta = thetas[segment_idx]
    # getting rotation matrix
    r = rotation(theta)
    dr_theta_1 = d_rotation(theta)
    # calculating current segment vector by adding rotated segment template to the tip of the previous segment
    np_joints[segment_idx+1] = np.dot(r, segment) + np_joints[segment_idx]

    # STILL BLACK MAGIC FOR ME
    x = dr_theta_1 @ segment

    if segment_idx:
        x = prev_r @ x
```

```
    # is this somehow related to derivative of the loss function?
    d_theta_1 = np.sum(x * -2 * (TARGET_POINT - np_joints[-1]))
    # END OF BLACK MAGIC

    # updating and storing new rotation value for the current segment
    thetas[segment_idx] -= d_theta_1 * LEARNING_RATE

    prev_r = r

loss = np.sum((TARGET_POINT - np_joints[-1]) ** 2)
plt.title(f'loss: {loss:.4f} thetas: {tuple(round(np.rad2deg(theta)) for theta in thetas)}')
```

3) Loss of MSE from tip of robot arm (last vector) to target point:

```
loss = np.sum((TARGET_POINT - np_joints[-1]) ** 2)
```