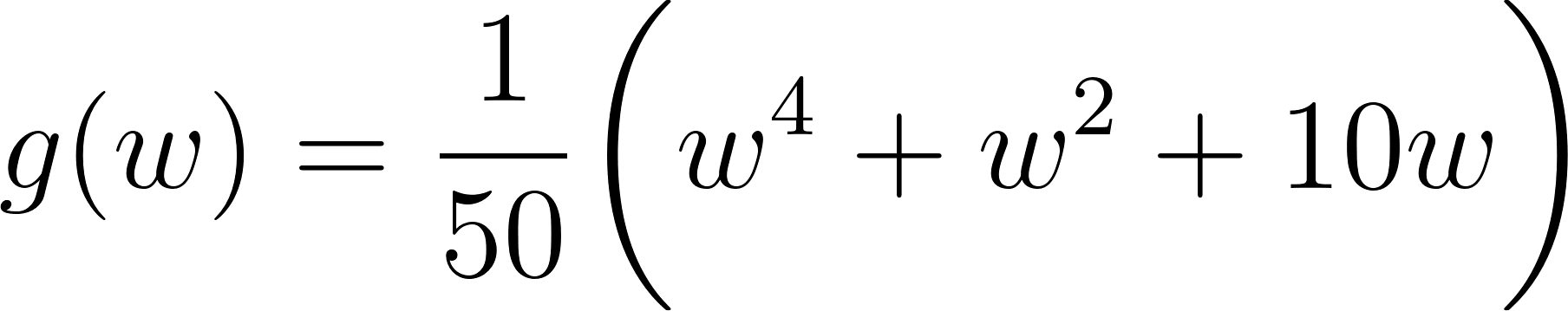
# Homework 2: Gradient Descent

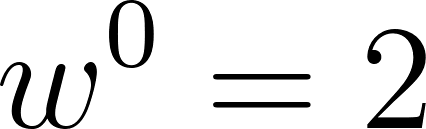
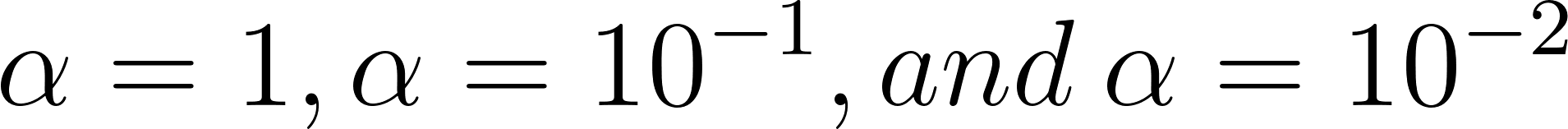
| Contact TA | Chhandak Bagchi |
| --- | --- |
| Released date | Thursday, 9/21/2023 |
| Submission deadline | Midnight, 11:59PM, Thursday, 9/28/2023 |
| Total Marks | 100 |

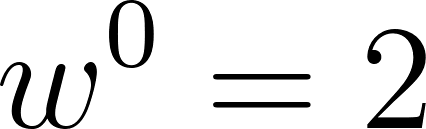
## Task Descriptions

### Task 1: Implement the gradient descent algorithm

Implement the gradient descent algorithm. Run gradient descent to minimize the following function:

[](https://www.codecogs.com/eqnedit.php?latex=g(w)%20%3D%20%5Cfrac%7B1%7D%7B50%7D%20%5Cbigg(w%5E4%20%2B%20w%5E2%20%2B10w%20%5Cbigg)#0)

Start with an initial point [](https://www.codecogs.com/eqnedit.php?latex=w%5E0%3D2#0) and run for 1000 iterations. Make three separate runs using each of the step length values [](https://www.codecogs.com/eqnedit.php?latex=%5Calpha%20%3D%201%2C%20%5Calpha%20%3D%2010%5E%7B-1%7D%2C%20and%20%5C%3A%20%20%5Calpha%20%3D%2010%5E%7B-2%7D#0).

Compute the derivative of this function by hand, and implement it (as well as the function itself) in Python using Numpy. Report the value of the function and its derivative at [](https://www.codecogs.com/eqnedit.php?latex=w%5E0%3D2#0).

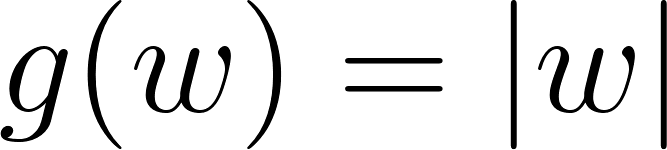
Plot the resulting cost function history plot of each run in a single figure to compare their performance. Also report which step length value works best for this particular function and initial point.

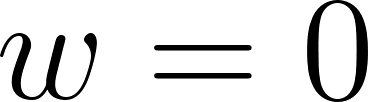
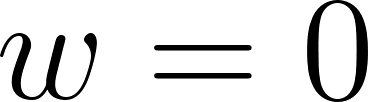
A skeleton of the desired algorithm is in [hw2.py](https://drive.google.com/file/d/1hWnq_UQNmGGxydmABkPpdyJypgnX2Jy_/view?usp=drive_link).

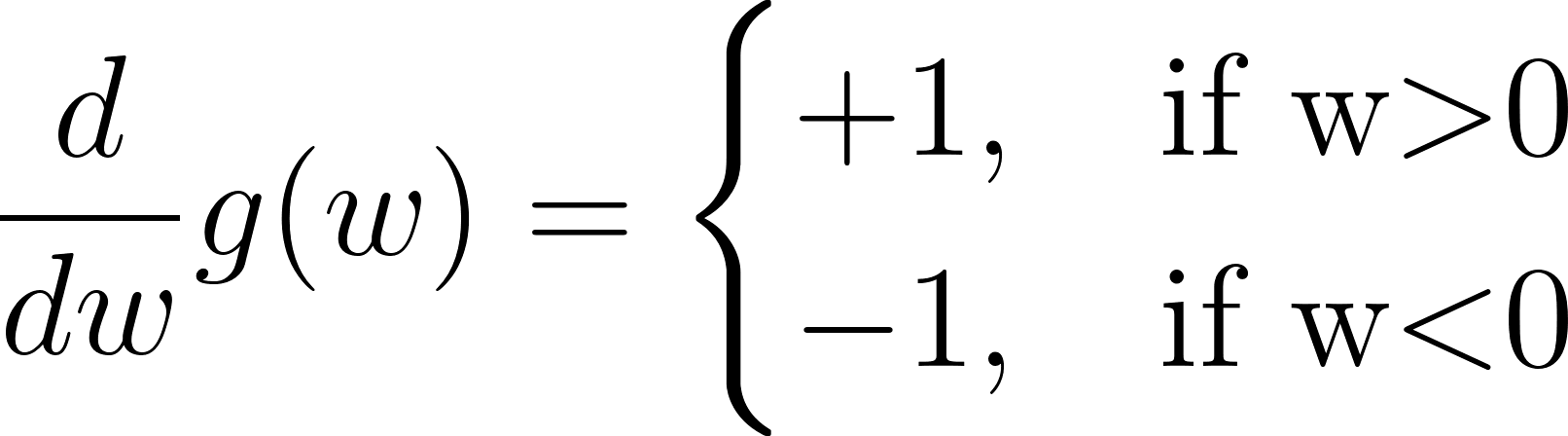
All parts marked "TO DO" are for you to construct.

### Task 2：Compare fixed and diminishing steplengths with gradient descent

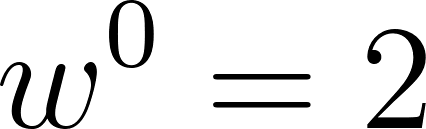
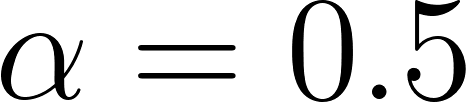
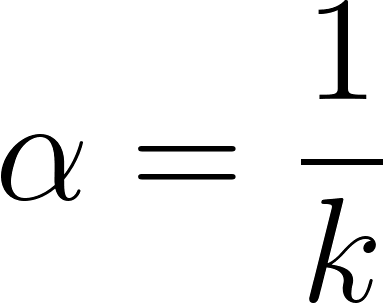
In this exercise you will compare a fixed steplength scheme and a diminishing step length rule to minimize the function

[](https://www.codecogs.com/eqnedit.php?latex=g(w)%20%3D%20%7Cw%7C#0)

Notice that this function has a single global minimum at [](https://www.codecogs.com/eqnedit.php?latex=w%3D0#0) and a derivative defined (everywhere but at [](https://www.codecogs.com/eqnedit.php?latex=w%3D0#0))

[](https://www.codecogs.com/eqnedit.php?latex=%5Cfrac%20%7Bd%7D%7Bdw%7D%20g(w)%20%3D%20%5Cbegin%7Bcases%7D%2B1%2C%20%26%20%5Ctext%7Bif%20w%24%3E%240%7D%20%5C%5C-1%2C%20%26%20%5Ctext%7Bif%20w%24%3C%240%7D%5Cend%7Bcases%7D#0)

which makes the use of any fixed step length schema problematic for gradient descent.

You will make two runs of 20 steps of gradient descent each initialized at the point [](https://www.codecogs.com/eqnedit.php?latex=w%5E0%3D2#0), the first with a fixed step length rule of [](https://www.codecogs.com/eqnedit.php?latex=%5Calpha%3D0.5#0) for each and every step, and the second using the diminishing step length rule [](https://www.codecogs.com/eqnedit.php?latex=%5Calpha%20%3D%20%5Cfrac%7B1%7D%7Bk%7D#0).

Compute the cost function history associated with each desired run and plot both to compare.

In this task, you should use Jax to automatically compute the gradients of the given function. Instructions on how to install Jax in your Conda environment can be found in this [section](#_4qr1xsr7oyp) of this document.

A skeleton of the desired algorithm is provided in hw2.py. All parts marked "TO DO" are for you to construct.

# 

# 

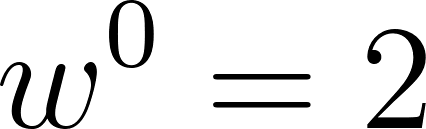
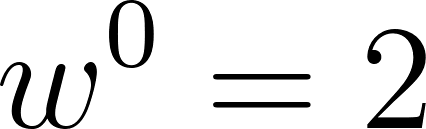
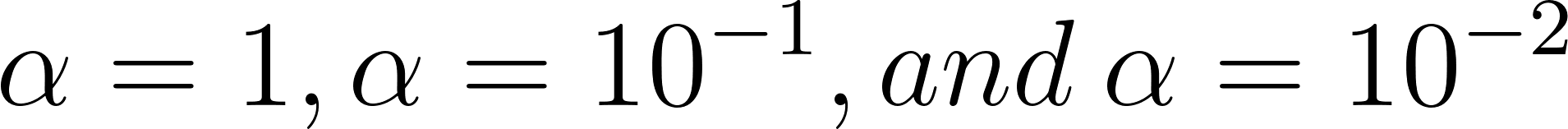
# 

# Deliverables

* **The completed source code hw2.py**
* **A pdf report**

The source code should be able to run by executing the command: python hw2.py

The PDF report should include:

* Task 1:
  + Value of the cost function (g(w)) at [](https://www.codecogs.com/eqnedit.php?latex=w%5E0%3D2#0).
  + Value of the derivative of the cost function at [](https://www.codecogs.com/eqnedit.php?latex=w%5E0%3D2#0).
  + Single figure containing the cost function history of three runs using different step length values ([](https://www.codecogs.com/eqnedit.php?latex=%5Calpha%20%3D%201%2C%20%5Calpha%20%3D%2010%5E%7B-1%7D%2C%20and%20%5C%3A%20%20%5Calpha%20%3D%2010%5E%7B-2%7D#0)).
  + Report which step length works best for this particular function and initial point.
* Task 2:
  + Single figure containing the cost function history of two runs using a fixed step length and a diminishing step length.

# Environment Configuration: Install Jax

**Please refer to the Appendix in the HW1 document to configure the environment.**

**You need to install an additional library Jax (CPU version).**

**Please install Jax using Conda.**

To install Jax follow these steps:

1. We would suggest creating a new Conda environment:conda create -n jax\_env
2. Activate the new environment: conda activate jax\_env
3. You can install Jax in your Conda environment using the following command: conda install -c conda-forge jax
4. Install matplotlib in your environment using: conda install -c conda-forge matplotlib

You can refer to the official documentation and quickstart guide for Jax for more info on the library: [Documentation](https://jax.readthedocs.io/en/latest/notebooks/quickstart.html), [Quickstart guide(Colab Notebook)](https://colab.research.google.com/github/google/jax/blob/main/docs/notebooks/quickstart.ipynb)