18AIE339T-MATRIX THEORY FOR ARTIFICIAL INTELLIGENCE

Dr. Antony Sophia N

Assistant Professor

Department of Computational Intelligence

SRM Institute of Science and Technology

Course Objective

Unit I – Linear System
Understand the basic concepts of linear algebra through computer science and Engineering applications

Unit II – Matrix Calculus
Learn the basic concepts of matrix calculus

Unit III – Matrix Analysis
Perform matrix analysis for various optimization algorithms

Unit IV – Matrix Solutions
Apply the concepts of vector spaces, linear transformations, matrices and inner product spaces in engineering

Unit V – Optimization
Solve problems in computer vision using optimization algorithms with single and multi-variables for large datasets

Reference Books

1. Xian-DaZhang, "A Matrix Algebra Approach to Artificial Intelligence" , Springer, 2021
2. Xian-DaZhang, "Matrix Analysis and Applications" ,Cambridge University Press, 2017
3. Charu C.Aggarwal, "Linear Algebra and Optimization for Machine Learning" , Springer, 2020.
4. Stephen Boyd,Lieven Vandenberghe, "Introduction to Applied Linear Algebra- Vectors, Matrices, and Least Squares" , Cambridge University Press, 2018
5. "LinearAlgebra", Kenneth Hoffman and RayKunze, Prentice Hall India,2013.
6. "LinearAlgebra", Cheney and Kincaid, Jones and Bartlett learning,2014

UNIT-III - Matrix Analysis

- *Jacobian Matrix*
- *Gradient Matrix*
- *Real Matrix Differential*
- *Complex Gradient Matrices*
- *Gradient of Complex Variable function*
- *Gradient Method for smooth Convex optimization*
- *Non-smooth convex optimization*
- *Constrained Convex Optimization*

Jacobian Matrix

Jacobian Matrix

- A Jacobian matrix, often denoted as J or ∂(f)/∂(x), is a mathematical concept used in calculus and linear algebra.

- It plays a crucial role in various fields, including physics, engineering, computer graphics, optimization, and machine learning.

- The Jacobian matrix is used to represent the rate of change of a vector-valued function with respect to its input variables.

Jacobian Matrix (contd.)

- Suppose you have a vector-valued function f: $\mathbb{R}^n \rightarrow \mathbb{R}^m$, where $\mathbb{R}^n$ represents n-dimensional real numbers and $\mathbb{R}^m$ represents m-dimensional real numbers.

- The Jacobian matrix J of this function is an m × n matrix, where each element (i, j) of the matrix represents the partial derivative of the i-th component of the output vector with respect to the j-th component of the input vector.

- In other words, it contains all the partial derivatives of the components of f with respect to the input variables.

Jacobian Matrix (contd.)

Consider the function $\mathbf{f} : \mathbf{R}^2 \to \mathbf{R}^2$, with $(x, y) \mapsto (f_1(x, y), f_2(x, y))$, given by

$$\mathbf{f}\left(\begin{bmatrix} x \\ y \end{bmatrix}\right) = \begin{bmatrix} f_1(x,y) \\ f_2(x,y) \end{bmatrix} = \begin{bmatrix} x^2 y \\ 5x + \sin y \end{bmatrix}.$$

Then we have

$$f_1(x,y) = x^2 y$$

and

$$f_2(x,y) = 5x + \sin y$$

and the Jacobian matrix of $\mathbf{f}$ is

$$\mathbf{J}_{\mathbf{f}}(x,y) = \begin{bmatrix} \dfrac{\partial f_1}{\partial x} & \dfrac{\partial f_1}{\partial y} \\ \dfrac{\partial f_2}{\partial x} & \dfrac{\partial f_2}{\partial y} \end{bmatrix} = \begin{bmatrix} 2xy & x^2 \\ 5 & \cos y \end{bmatrix}$$

and the Jacobian determinant is

$$\det(\mathbf{J}_{\mathbf{f}}(x,y)) = 2xy\cos y - 5x^2.$$

Jacobian Matrix (contd.)

The Jacobian matrix of the function $\mathbf{F} : \mathbf{R}^3 \to \mathbf{R}^4$ with components

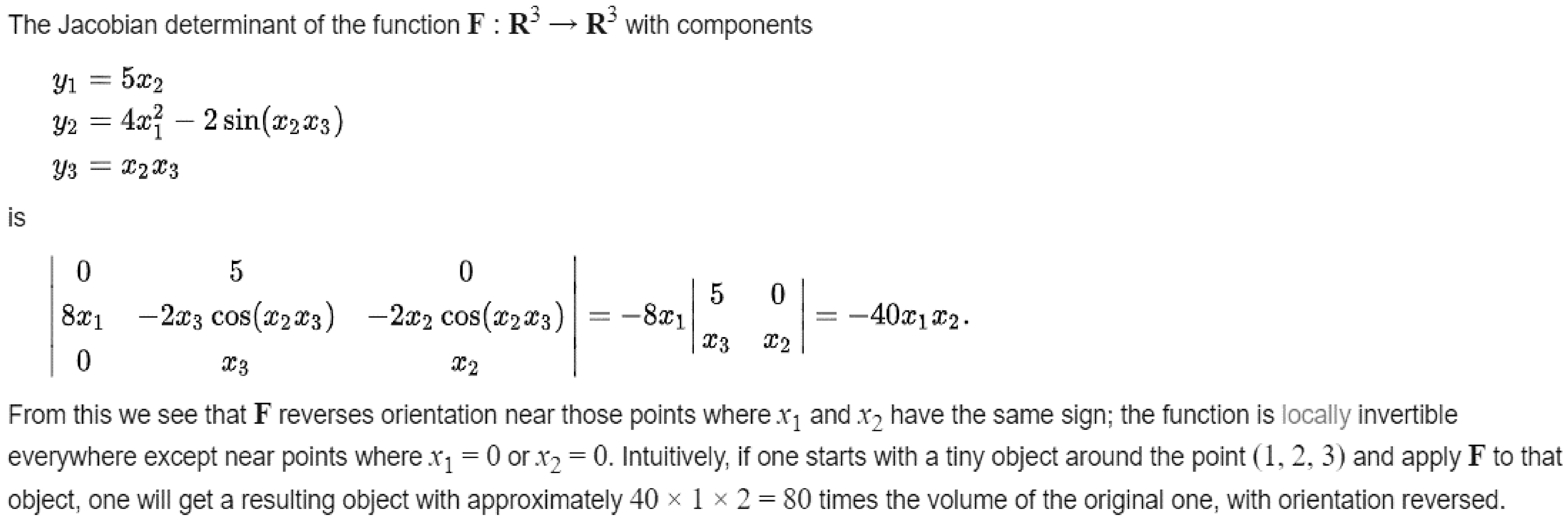
$$\begin{aligned} y_1 &= x_1 \\ y_2 &= 5x_3 \\ y_3 &= 4x_2^2 - 2x_3 \\ y_4 &= x_3 \sin x_1 \end{aligned}$$

is

$$\mathbf{J_F}(x_1, x_2, x_3) = \begin{bmatrix} \dfrac{\partial y_1}{\partial x_1} & \dfrac{\partial y_1}{\partial x_2} & \dfrac{\partial y_1}{\partial x_3} \\[1em] \dfrac{\partial y_2}{\partial x_1} & \dfrac{\partial y_2}{\partial x_2} & \dfrac{\partial y_2}{\partial x_3} \\[1em] \dfrac{\partial y_3}{\partial x_1} & \dfrac{\partial y_3}{\partial x_2} & \dfrac{\partial y_3}{\partial x_3} \\[1em] \dfrac{\partial y_4}{\partial x_1} & \dfrac{\partial y_4}{\partial x_2} & \dfrac{\partial y_4}{\partial x_3} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 5 \\ 0 & 8x_2 & -2 \\ x_3 \cos x_1 & 0 & \sin x_1 \end{bmatrix}.$$

This example shows that the Jacobian matrix need not be a square matrix.

Jacobian Matrix (contd.)



The Jacobian determinant of the function $\mathbf{F} : \mathbf{R}^3 \to \mathbf{R}^3$ with components

$$\begin{aligned} y_1 &= 5x_2 \\ y_2 &= 4x_1^2 - 2\sin(x_2x_3) \\ y_3 &= x_2x_3 \end{aligned}$$

is

$$\begin{vmatrix} 0 & 5 & 0 \\ 8x_1 & -2x_3\cos(x_2x_3) & -2x_2\cos(x_2x_3) \\ 0 & x_3 & x_2 \end{vmatrix} = -8x_1 \begin{vmatrix} 5 & 0 \\ x_3 & x_2 \end{vmatrix} = -40x_1x_2.$$

From this we see that $\mathbf{F}$ reverses orientation near those points where x_1 and x_2 have the same sign; the function is locally invertible everywhere except near points where $x_1 = 0$ or $x_2 = 0$. Intuitively, if one starts with a tiny object around the point $(1, 2, 3)$ and apply $\mathbf{F}$ to that object, one will get a resulting object with approximately $40 \times 1 \times 2 = 80$ times the volume of the original one, with orientation reversed.

Applications of Jacobian Matrix in AI

Application	Description
Gradient Descent	Used for optimizing models by computing gradients of cost functions to update model parameters.
Neural Networks	Integral in backpropagation for training deep learning models by calculating gradients for weight updates.
Reinforcement Learning	Employed in policy gradients to optimize agent policies in sequential decision-making tasks.
Natural Language Processing (NLP)	Used for analyzing word embeddings' impact on downstream NLP tasks, improving interpretability.
Optimization	Essential for solving optimization problems efficiently, providing information on objective function sensitivity.
Robotics	Models the relationship between joint and end-effector velocities, crucial for tasks like inverse kinematics.
Sensitivity Analysis	Assesses how input changes affect model or system output, aiding in understanding system robustness.
Feature Selection and Engineering	Helps understand feature impact on model output, aiding in feature selection and engineering.

Gradient Matrix

Gradient Matrix

- A "gradient matrix" is not a standard mathematical or mathematical term. However, it's possible that you might be referring to the "gradient vector" or "gradient matrix" in the context of calculus and vector calculus.

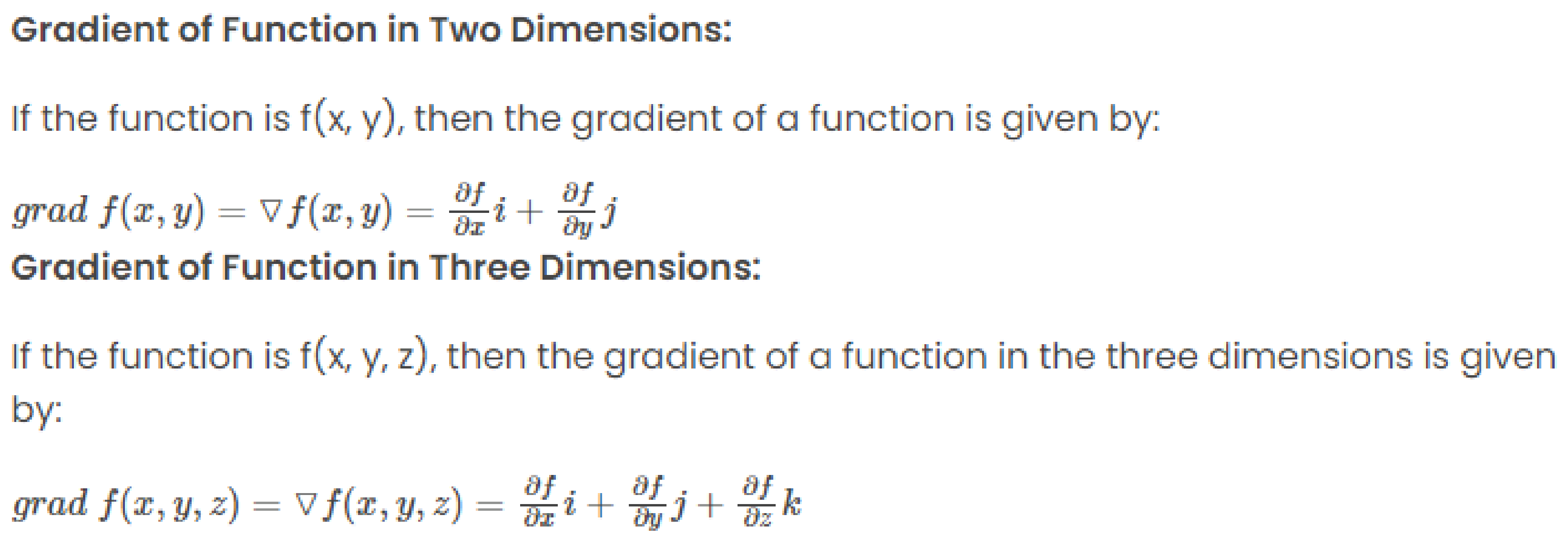
- **Gradient Vector:** In calculus, the gradient vector is a vector that points in the direction of the steepest increase of a scalar-valued function (usually a multivariable function) at a specific point. It is denoted by the symbol ∇ (nabla) and is also called the "del" operator. The gradient vector is composed of partial derivatives of the function with respect to each of its input variables. For a scalar-valued function $f(x_1, x_2, \ldots, x_n)$, the gradient vector ∇f is defined as:

$$\nabla f = [\partial f/\partial x_1, \partial f/\partial x_2, \ldots, \partial f/\partial x_n]$$

- Each component of the gradient vector represents how the function changes concerning a specific input variable.

Gradient Matrix (Contd.)

- Gradient Matrix: Sometimes, the term "gradient matrix" might be used when dealing with a vector-valued function or multiple scalar-valued functions simultaneously.
- In this case, you would have a matrix where each row corresponds to the gradient vector of a different scalar-valued function.
- For a vector-valued function $F(x_1, x_2, ..., x_n)$ that maps from $\mathbb{R}^n$ to $\mathbb{R}^m$, you can construct a gradient matrix ∇F, where each row represents the gradient vector of a different component of F.



Gradient of Function in Two Dimensions:

If the function is f(x, y), then the gradient of a function is given by:

$$grad\ f(x,y) = \nabla f(x,y) = \frac{\partial f}{\partial x} i + \frac{\partial f}{\partial y} j$$

Gradient of Function in Three Dimensions:

If the function is f(x, y, z), then the gradient of a function in the three dimensions is given by:

$$grad\ f(x,y,z) = \nabla f(x,y,z) = \frac{\partial f}{\partial x} i + \frac{\partial f}{\partial y} j + \frac{\partial f}{\partial z} k$$

Gradient Matrix (Contd.)

Example 1:

Find the gradient of function $f(x, y) = x + 3y^2$.

Solution:

Given function: $f(x, y) = x + 3y^2$

We know that,

$$grad\ f(x,y) = \nabla f(x,y) = \frac{\partial f}{\partial x} i + \frac{\partial f}{\partial y} j$$

$$grad\ f(x,y) = \nabla f(x,y) = \frac{\partial (x+3y^2)}{\partial x} i + \frac{\partial (x+3y^2)}{\partial y} j$$

$= (1+0)i + (0 + 3.\ 2y^{2-1})j$

$= i + 6yj$

Hence, the gradient of the function, $f(x, y) = x + 3y^2$ is $i + 6yj$.

Gradient Matrix (Contd.)

Example 2:

Find the gradient of the function, f(x, y, z) = sin(x)e^y In (z).

Solution:

Given function: f(x, y, z) = sin(x)e^y In (z).

As we know,

$$grad\ f(x,y,z) = \nabla f(x,y,z) = \frac{\partial f}{\partial x} i + \frac{\partial f}{\partial y} j + \frac{\partial f}{\partial z} k$$

Thus,

$$\frac{\partial f}{\partial x} = \frac{\partial}{\partial x}(sin(x))e^y\, ln(z) = cos(x)e^y\, ln(z)$$
$$\frac{\partial f}{\partial y} = sin(x)\frac{\partial}{\partial y}e^y\, ln(z) = sin(x)e^y\, ln(z)$$
$$\frac{\partial f}{\partial z} = sin(x)e^y\frac{\partial}{\partial z}ln(z) = sin(x)e^y\frac{1}{z}$$

Therefore, the gradient of the function is:

∇ f(x, y, z) = cos (x) e^y In(z) i + sin(x) e^y In(z)j + sin(x) e^y(1/z)k.

Real Matrix Differential

Real Matrix Differential

- Matrix differentiation is an important concept in machine learning and artificial intelligence (AI), especially in the context of optimization problems and gradient-based algorithms.

- When working with real-valued matrices, you often need to calculate gradients or differentials with respect to matrices.

Real Matrix Differential (Contd.)

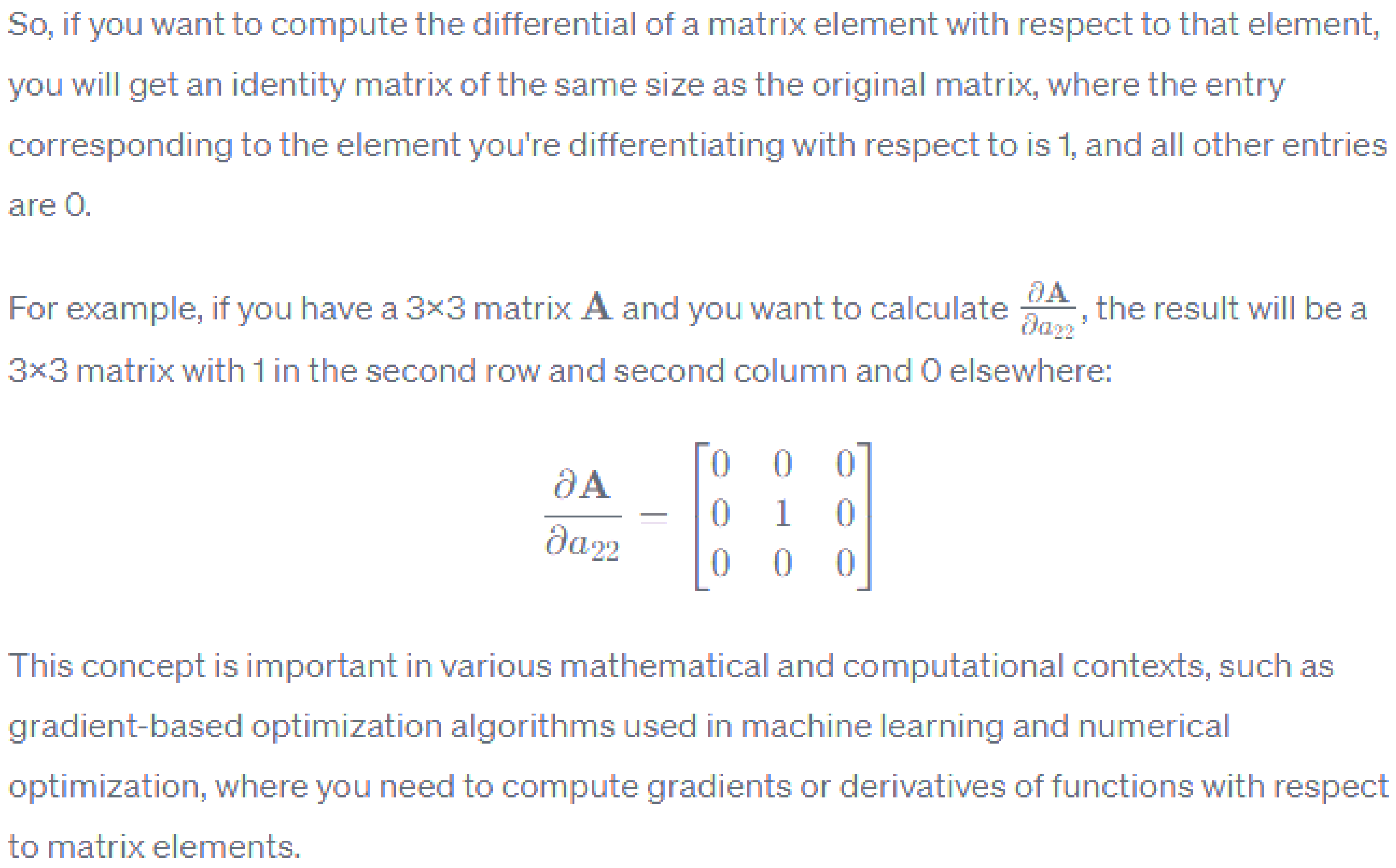
Calculating the differential of a real matrix with respect to its elements is an essential concept in matrix calculus. The notation $\frac{\partial \mathbf{A}}{\partial a_{ij}}$ represents the derivative of a matrix $\mathbf{A}$ with respect to its element a_{ij}. This derivative can be thought of as the sensitivity of the matrix to changes in that specific element.

Here's how you can calculate the differential of a real matrix:

Let $\mathbf{A}$ be an $m \times n$ matrix, and we want to compute $\frac{\partial \mathbf{A}}{\partial a_{ij}}$, where a_{ij} is an element of $\mathbf{A}$. Then, the differential of $\mathbf{A}$ with respect to a_{ij} is a matrix of the same size as $\mathbf{A}$, and its elements are given by:

$$\frac{\partial \mathbf{A}}{\partial a_{ij}} = \begin{bmatrix} 0 & 0 & \dots & 0 \\ 0 & 0 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & 1 \\ 0 & 0 & \dots & 0 \end{bmatrix}$$

Real Matrix Differential (Contd.)



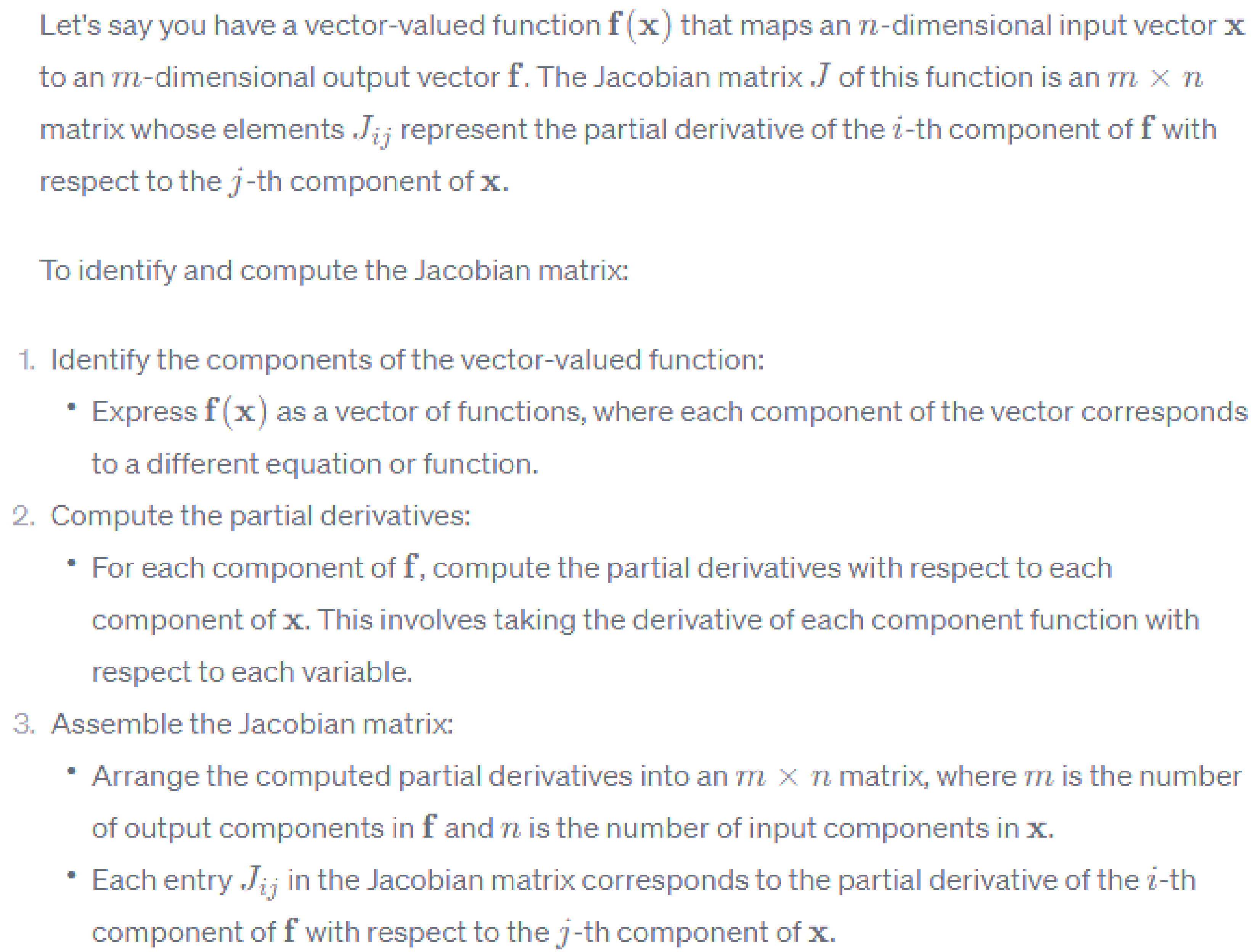
So, if you want to compute the differential of a matrix element with respect to that element, you will get an identity matrix of the same size as the original matrix, where the entry corresponding to the element you're differentiating with respect to is 1, and all other entries are 0.

For example, if you have a 3×3 matrix $\mathbf{A}$ and you want to calculate $\frac{\partial \mathbf{A}}{\partial a_{22}}$, the result will be a 3×3 matrix with 1 in the second row and second column and 0 elsewhere:

$$\frac{\partial \mathbf{A}}{\partial a_{22}} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

This concept is important in various mathematical and computational contexts, such as gradient-based optimization algorithms used in machine learning and numerical optimization, where you need to compute gradients or derivatives of functions with respect to matrix elements.

Jacobian Matrix Identification


Let's say you have a vector-valued function $\mathbf{f}(\mathbf{x})$ that maps an n-dimensional input vector $\mathbf{x}$ to an m-dimensional output vector $\mathbf{f}$. The Jacobian matrix J of this function is an $m \times n$ matrix whose elements J_{ij} represent the partial derivative of the i-th component of $\mathbf{f}$ with respect to the j-th component of $\mathbf{x}$.

To identify and compute the Jacobian matrix:

1. Identify the components of the vector-valued function:
   - Express $\mathbf{f}(\mathbf{x})$ as a vector of functions, where each component of the vector corresponds to a different equation or function.
2. Compute the partial derivatives:
   - For each component of $\mathbf{f}$, compute the partial derivatives with respect to each component of $\mathbf{x}$. This involves taking the derivative of each component function with respect to each variable.
3. Assemble the Jacobian matrix:
   - Arrange the computed partial derivatives into an $m \times n$ matrix, where m is the number of output components in $\mathbf{f}$ and n is the number of input components in $\mathbf{x}$.
   - Each entry J_{ij} in the Jacobian matrix corresponds to the partial derivative of the i-th component of $\mathbf{f}$ with respect to the j-th component of $\mathbf{x}$.

Jacobian Matrix of Real Matrix Functions

Consider the function f: $R^2 \to R^2$, with $(x, y) \mapsto (f_1(x, y), f_2(x, y))$, given by:

$$\mathbf{f}\left(\begin{bmatrix} x \\ y \end{bmatrix}\right) = \begin{bmatrix} f_1(x, y) \\ f_2(x, y) \end{bmatrix} = \begin{bmatrix} x^4 + 2y^2x \\ 5y^2 - 4xy + 1 \end{bmatrix}$$

Find the Jacobian matrix at the point $(1, 2)$.

Step 1: Calculate all the first-order partial derivatives of the function:

$$\frac{\partial f_1}{\partial x} = 4x^3 + 2y^2$$

$$\frac{\partial f_1}{\partial y} = 4xy$$

$$\frac{\partial f_2}{\partial x} = -4y$$

$$\frac{\partial f_2}{\partial y} = 10y - 4x$$

Jacobian Matrix of Real Matrix Functions

Step 2: Apply the formula of the Jacobian matrix. This function has two variables and two vector components, so the Jacobian matrix will be a 2×2 square matrix:

$$J\left(\begin{bmatrix} x \\ y \end{bmatrix}\right) = \begin{bmatrix} \frac{\partial f_1}{\partial x} & \frac{\partial f_1}{\partial y} \\ \frac{\partial f_2}{\partial x} & \frac{\partial f_2}{\partial y} \end{bmatrix} = \begin{bmatrix} 4x^3 + 2y^2 & 4xy \\ -4y & 10y - 4x \end{bmatrix}$$

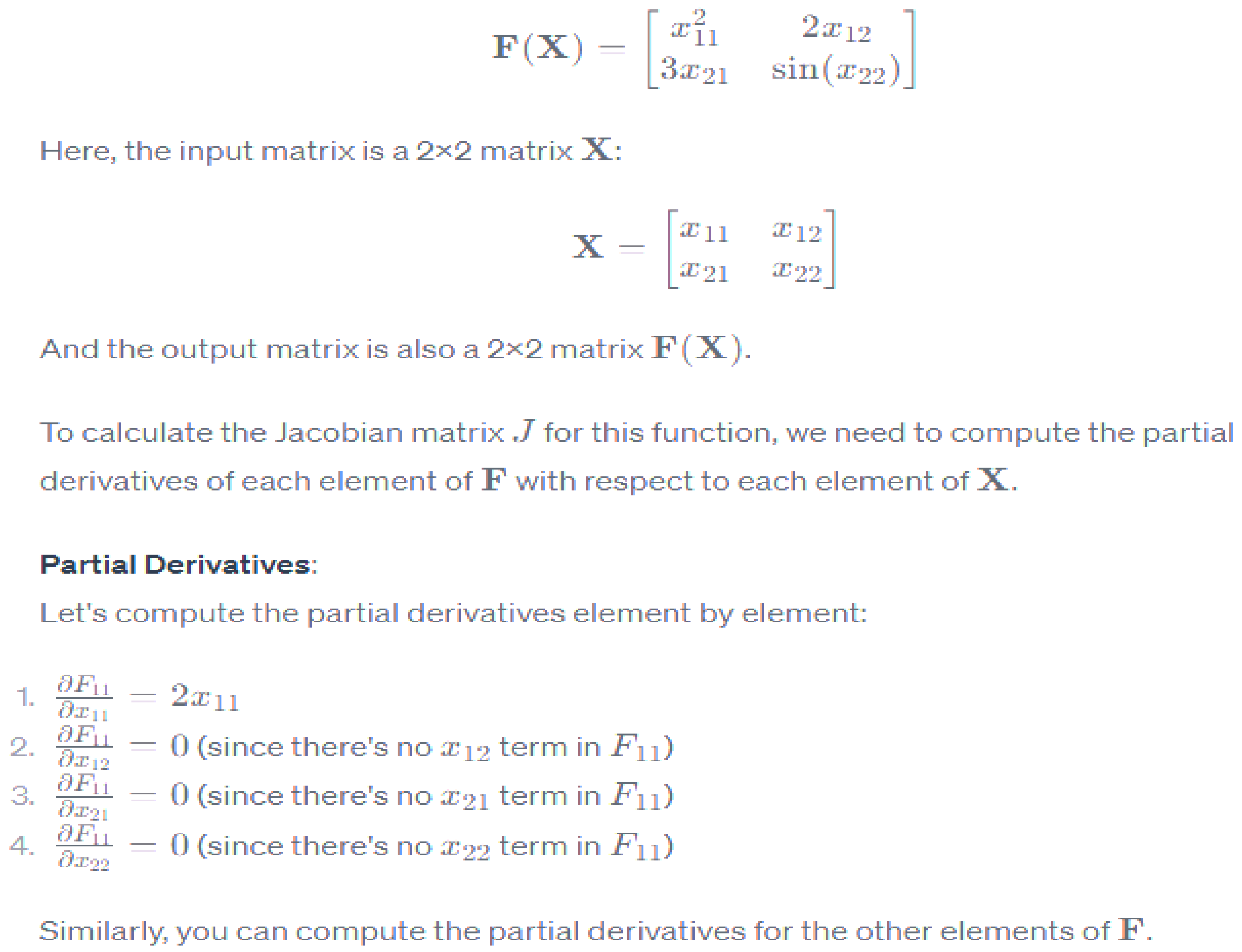
Step 3: Put the $(1, 2)$ given in the problem in the expression above:

$$\begin{bmatrix} 4 \cdot 1^3 + 2 \cdot 2^2 & 4 \cdot 1 \cdot 2 \\ -4 \cdot 2 & 10 \cdot 2 - 4 \cdot 1 \end{bmatrix}$$

Step 4: Calculate the expression:

$$\begin{bmatrix} 12 & 8 \\ -8 & 16 \end{bmatrix}$$

Jacobian Matrix of Real Matrix Functions



$$\mathbf{F}(\mathbf{X}) = \begin{bmatrix} x_{11}^2 & 2x_{12} \\ 3x_{21} & \sin(x_{22}) \end{bmatrix}$$

Here, the input matrix is a 2×2 matrix $\mathbf{X}$:

$$\mathbf{X} = \begin{bmatrix} x_{11} & x_{12} \\ x_{21} & x_{22} \end{bmatrix}$$

And the output matrix is also a 2×2 matrix $\mathbf{F}(\mathbf{X})$.

To calculate the Jacobian matrix J for this function, we need to compute the partial derivatives of each element of $\mathbf{F}$ with respect to each element of $\mathbf{X}$.

Partial Derivatives:

Let's compute the partial derivatives element by element:

1. $\frac{\partial F_{11}}{\partial x_{11}} = 2x_{11}$
2. $\frac{\partial F_{11}}{\partial x_{12}} = 0$ (since there's no x_{12} term in F_{11})
3. $\frac{\partial F_{11}}{\partial x_{21}} = 0$ (since there's no x_{21} term in F_{11})
4. $\frac{\partial F_{11}}{\partial x_{22}} = 0$ (since there's no x_{22} term in F_{11})

Similarly, you can compute the partial derivatives for the other elements of $\mathbf{F}$.

Problem 1:

Consider the function $f(x, y) = (x^2 + y^2, xy)$.

a) Find the Jacobian matrix of f at the point $(1, 2)$.

Problem 2:

Let $f(u, v, w) = (u^2, 2v - w, uvw)$.

a) Find the Jacobian matrix of f with respect to u, v, w.

Problem 3:

Consider the function $f(x, y, z) = (e^x, \sin(y), \cos(z))$.

a) Find the Jacobian matrix of f with respect to x, y, z.

Problem 4:

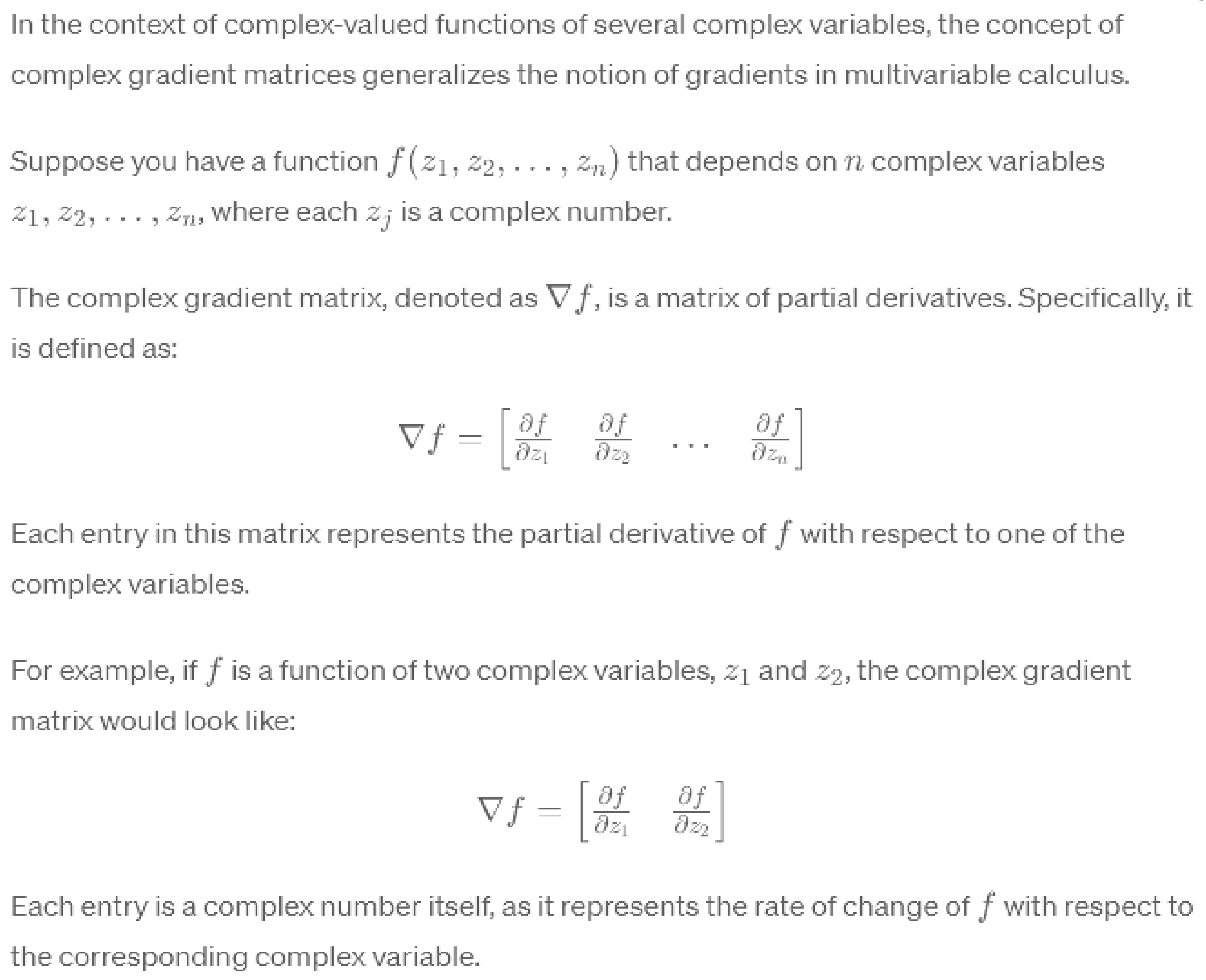
Let $f(u, v) = (u^3 - v^3, 3uv)$.

a) Find the Jacobian matrix of f at the point $(2, 1)$.

Complex Gradient Matrices

Complex Gradient Matrices

- The complex gradient matrix is a powerful tool in the study of complex analysis, especially when dealing with functions that depend on multiple complex variables.



In the context of complex-valued functions of several complex variables, the concept of complex gradient matrices generalizes the notion of gradients in multivariable calculus.

Suppose you have a function $f(z_1, z_2, \ldots, z_n)$ that depends on n complex variables $z_1, z_2, \ldots, z_n$, where each z_j is a complex number.

The complex gradient matrix, denoted as ∇f, is a matrix of partial derivatives. Specifically, it is defined as:

$$\nabla f = \begin{bmatrix} \frac{\partial f}{\partial z_1} & \frac{\partial f}{\partial z_2} & \cdots & \frac{\partial f}{\partial z_n} \end{bmatrix}$$

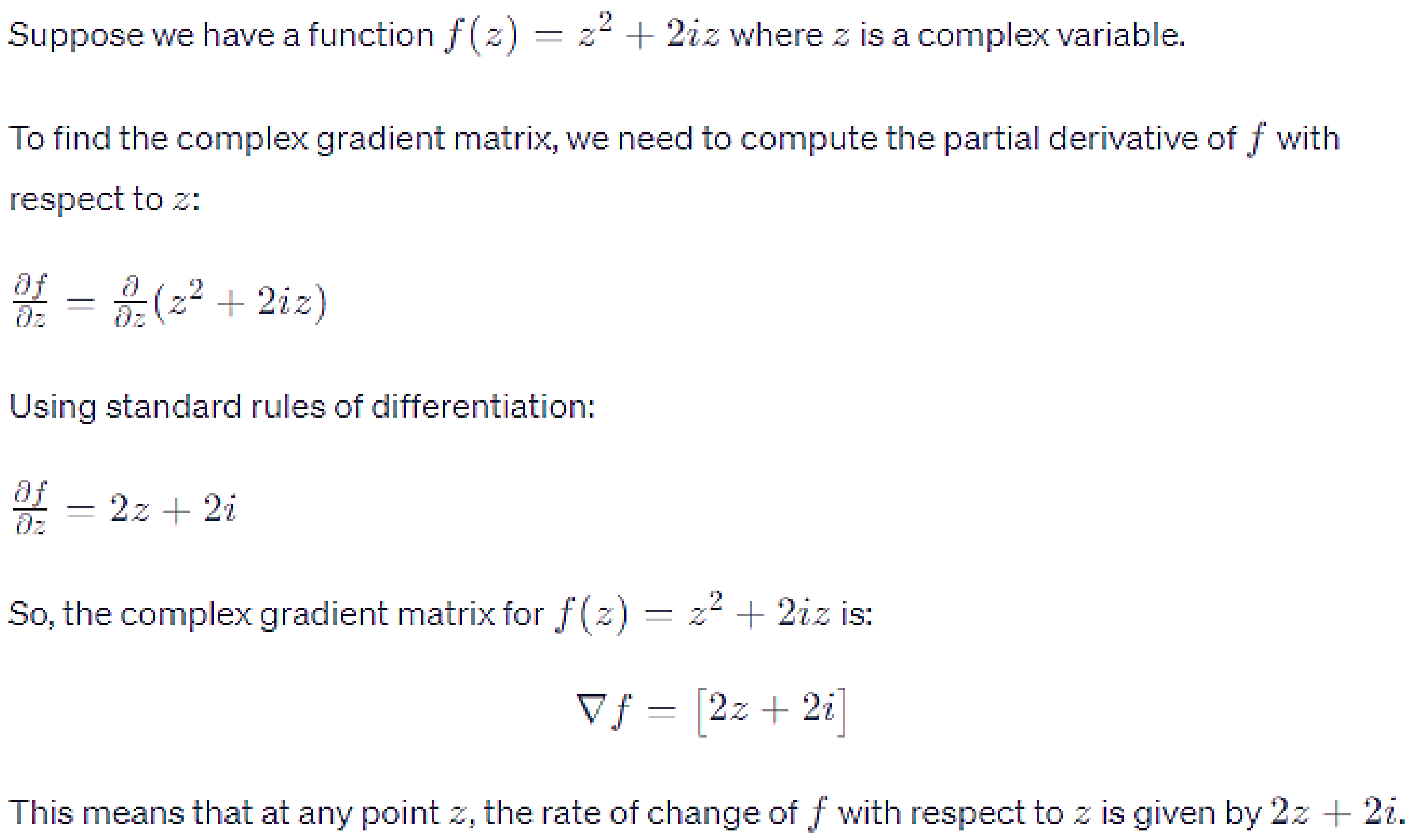
Each entry in this matrix represents the partial derivative of f with respect to one of the complex variables.

For example, if f is a function of two complex variables, z_1 and z_2, the complex gradient matrix would look like:

$$\nabla f = \begin{bmatrix} \frac{\partial f}{\partial z_1} & \frac{\partial f}{\partial z_2} \end{bmatrix}$$

Each entry is a complex number itself, as it represents the rate of change of f with respect to the corresponding complex variable.

Complex Gradient Matrices (Contd.)



Suppose we have a function $f(z) = z^2 + 2iz$ where z is a complex variable.

To find the complex gradient matrix, we need to compute the partial derivative of f with respect to z:

$$\frac{\partial f}{\partial z} = \frac{\partial}{\partial z}(z^2 + 2iz)$$

Using standard rules of differentiation:

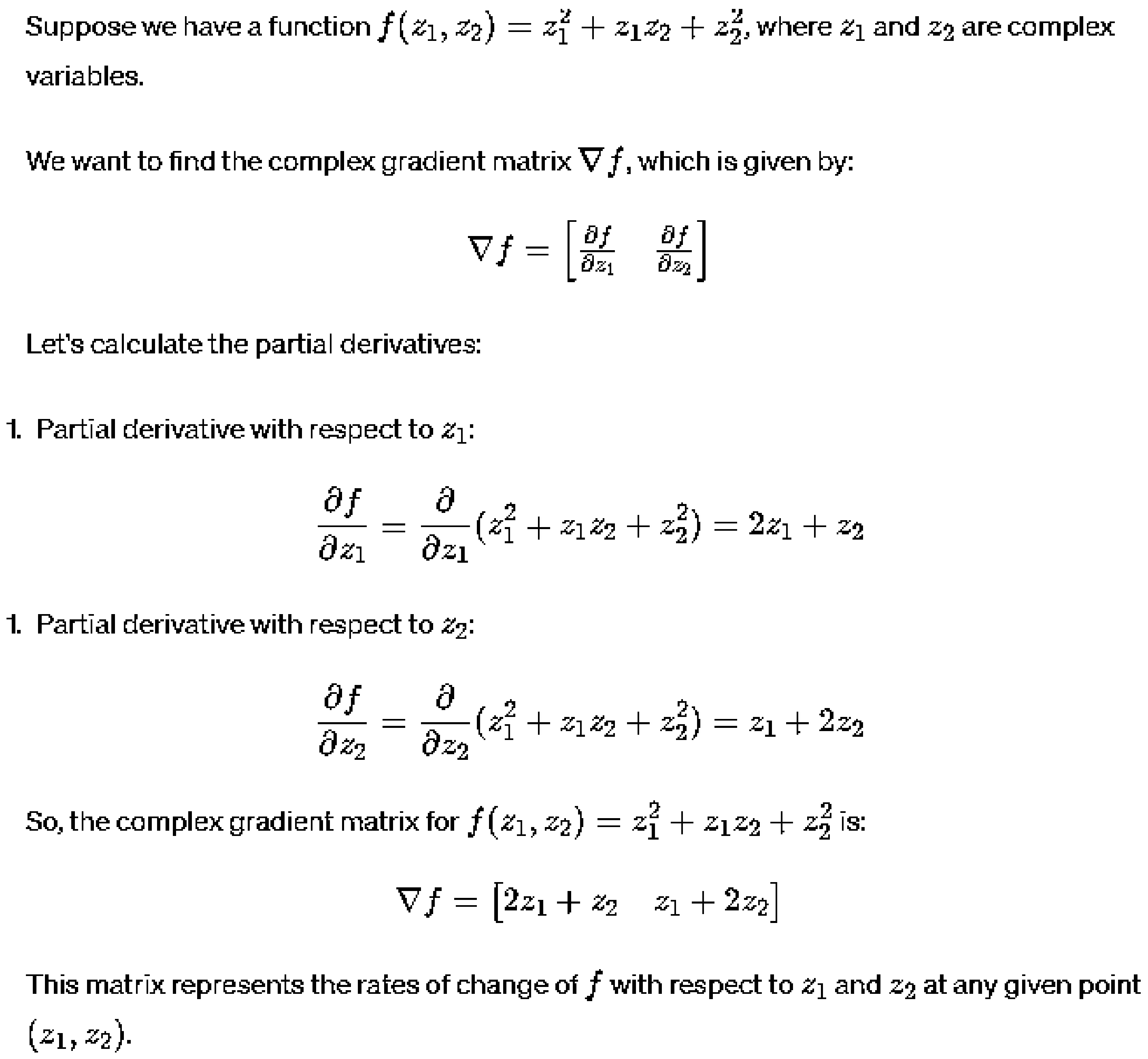
$$\frac{\partial f}{\partial z} = 2z + 2i$$

So, the complex gradient matrix for $f(z) = z^2 + 2iz$ is:

$$\nabla f = \left[2z + 2i\right]$$

This means that at any point z, the rate of change of f with respect to z is given by $2z + 2i$.

Complex Gradient Matrices (Contd.)



Suppose we have a function $f(z_1, z_2) = z_1^2 + z_1 z_2 + z_2^2$, where z_1 and z_2 are complex variables.

We want to find the complex gradient matrix ∇f, which is given by:

$$\nabla f = \begin{bmatrix} \frac{\partial f}{\partial z_1} & \frac{\partial f}{\partial z_2} \end{bmatrix}$$

Let's calculate the partial derivatives:

1. Partial derivative with respect to z_1:

$$\frac{\partial f}{\partial z_1} = \frac{\partial}{\partial z_1}(z_1^2 + z_1 z_2 + z_2^2) = 2z_1 + z_2$$

1. Partial derivative with respect to z_2:

$$\frac{\partial f}{\partial z_2} = \frac{\partial}{\partial z_2}(z_1^2 + z_1 z_2 + z_2^2) = z_1 + 2z_2$$

So, the complex gradient matrix for $f(z_1, z_2) = z_1^2 + z_1 z_2 + z_2^2$ is:

$$\nabla f = \begin{bmatrix} 2z_1 + z_2 & z_1 + 2z_2 \end{bmatrix}$$

This matrix represents the rates of change of f with respect to z_1 and z_2 at any given point (z_1, z_2).

Problem 1:

Consider the complex-valued function $f(z) = z^3 + 4iz^2 - 2\bar{z}$, where z is a complex variable and $\bar{z}$ is the complex conjugate of z.

a) Find the complex gradient matrix ∇f for this function.

b) Evaluate ∇f at the point $z = 2 + 3i$.

Problem 2:

Given the function $f(z_1, z_2) = z_1^2 + 2z_1z_2 + iz_2^2$, where z_1 and z_2 are complex numbers:

a) Compute the partial derivatives $\frac{\partial f}{\partial z_1}$ and $\frac{\partial f}{\partial z_2}$.

b) Write down the complex gradient matrix ∇f for this function.

Problem 3:

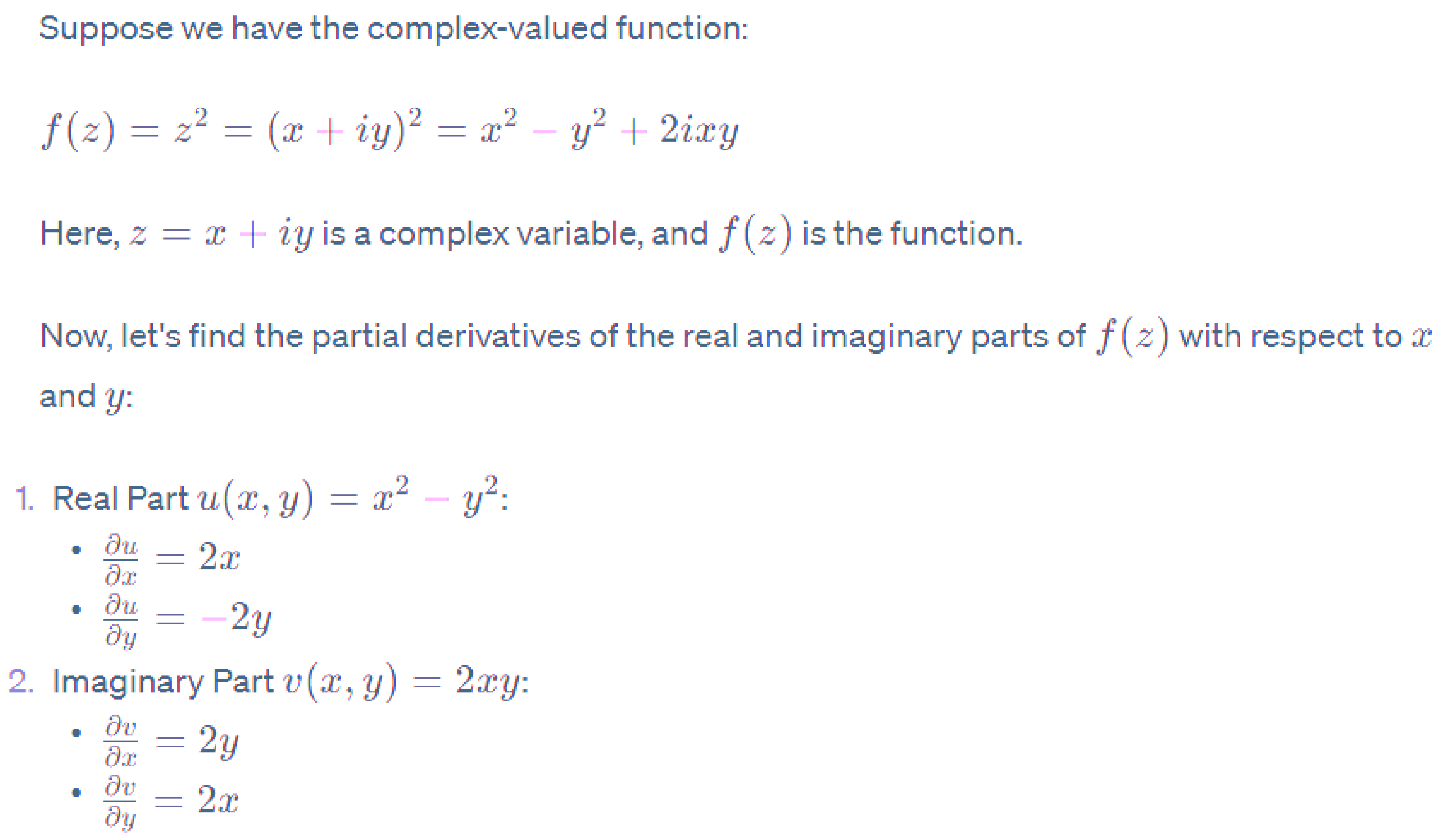
Consider the function $f(z_1, z_2, z_3) = z_1 z_2 z_3 + z_1^2 - z_2^2 - z_3^2$, where z_1, z_2, z_3 are complex variables.

a) Find the partial derivatives $\frac{\partial f}{\partial z_1}$, $\frac{\partial f}{\partial z_2}$, and $\frac{\partial f}{\partial z_3}$.

b) Write down the complex gradient matrix ∇f for this function.

Gradient of Complex Variable function

Consider an example to illustrate the concept of finding the gradient of a complex variable function.



Suppose we have the complex-valued function:

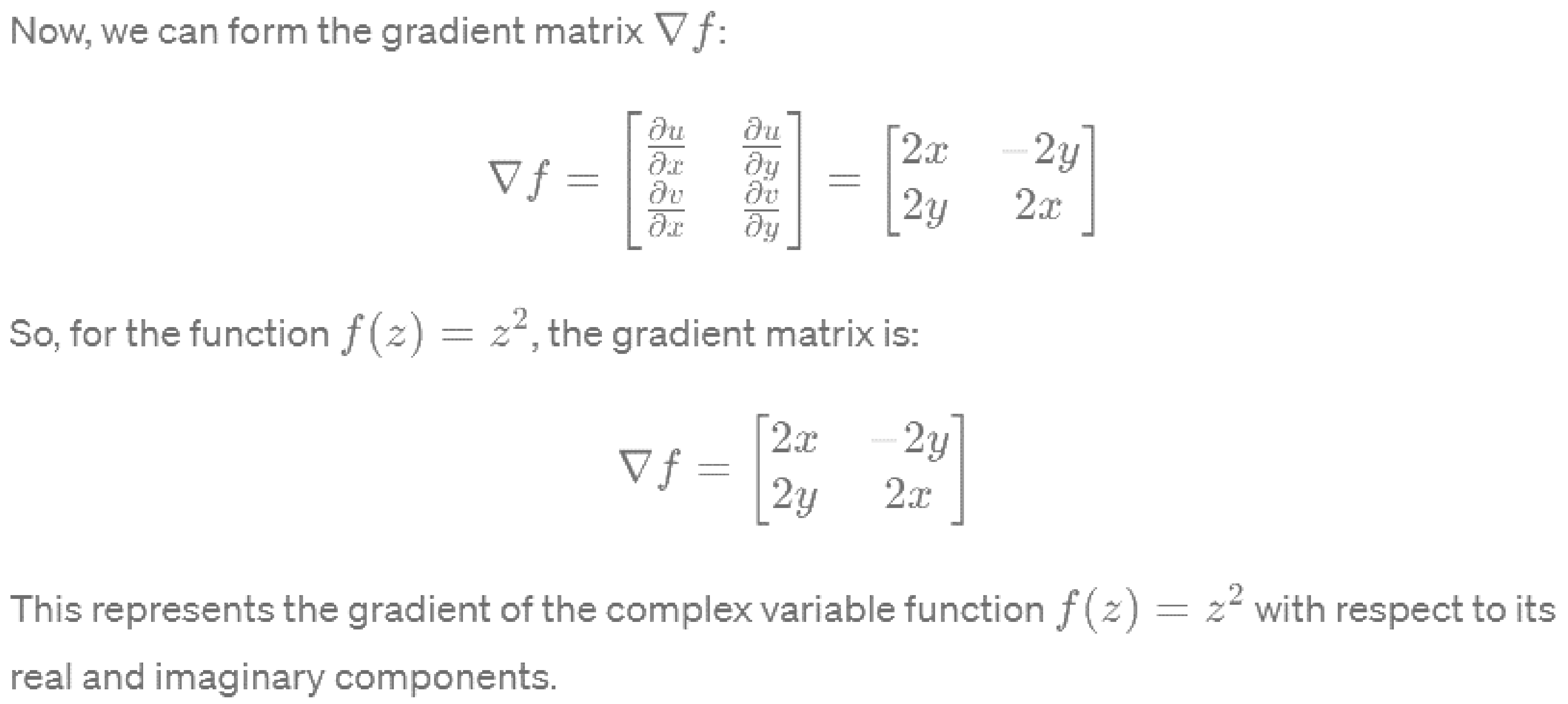
$$f(z) = z^2 = (x + iy)^2 = x^2 - y^2 + 2ixy$$

Here, $z = x + iy$ is a complex variable, and $f(z)$ is the function.

Now, let's find the partial derivatives of the real and imaginary parts of $f(z)$ with respect to x and y:

1. Real Part $u(x, y) = x^2 - y^2$:
   - $\frac{\partial u}{\partial x} = 2x$
   - $\frac{\partial u}{\partial y} = -2y$
2. Imaginary Part $v(x, y) = 2xy$:
   - $\frac{\partial v}{\partial x} = 2y$
   - $\frac{\partial v}{\partial y} = 2x$

Consider an example to illustrate the concept of finding the gradient of a complex variable function. (Contd.)



Now, we can form the gradient matrix ∇f:

$$\nabla f = \begin{bmatrix} \frac{\partial u}{\partial x} & \frac{\partial u}{\partial y} \\ \frac{\partial v}{\partial x} & \frac{\partial v}{\partial y} \end{bmatrix} = \begin{bmatrix} 2x & -2y \\ 2y & 2x \end{bmatrix}$$

So, for the function $f(z) = z^2$, the gradient matrix is:

$$\nabla f = \begin{bmatrix} 2x & -2y \\ 2y & 2x \end{bmatrix}$$

This represents the gradient of the complex variable function $f(z) = z^2$ with respect to its real and imaginary components.

Consider another example with a different complex-valued function

Suppose we have the complex-valued function:

$$f(z) = e^{iz} = e^{i(x+iy)} = e^{-y+ix} = e^{-y} \cdot (\cos(x) + i\sin(x))$$

Here, $z = x + iy$ is a complex variable, and $f(z)$ is the function.

Now, let's find the partial derivatives of the real and imaginary parts of $f(z)$ with respect to x and y:

1. Real Part $u(x, y) = e^{-y} \cdot \cos(x)$:
   - $\frac{\partial u}{\partial x} = -e^{-y} \cdot \sin(x)$
   - $\frac{\partial u}{\partial y} = -e^{-y} \cdot \cos(x)$
2. Imaginary Part $v(x, y) = e^{-y} \cdot \sin(x)$:
   - $\frac{\partial v}{\partial x} = e^{-y} \cdot \cos(x)$
   - $\frac{\partial v}{\partial y} = -e^{-y} \cdot \sin(x)$

Consider another example with a different complex-valued function (Contd.)

Now, we can form the gradient matrix ∇f:

$$\nabla f = \begin{bmatrix} \frac{\partial u}{\partial x} & \frac{\partial u}{\partial y} \\ \frac{\partial v}{\partial x} & \frac{\partial v}{\partial y} \end{bmatrix} = \begin{bmatrix} -e^{-y} \cdot \sin(x) & -e^{-y} \cdot \cos(x) \\ e^{-y} \cdot \cos(x) & -e^{-y} \cdot \sin(x) \end{bmatrix}$$

So, for the function $f(z) = e^{iz}$, the gradient matrix is:

$$\nabla f = \begin{bmatrix} -e^{-y} \cdot \sin(x) & -e^{-y} \cdot \cos(x) \\ e^{-y} \cdot \cos(x) & -e^{-y} \cdot \sin(x) \end{bmatrix}$$

This represents the gradient of the complex variable function $f(z) = e^{iz}$ with respect to its real and imaginary components.

Gradient Method for smooth Convex optimization

Gradient Method for smooth Convex optimization

- The gradient method, also known as the steepest descent method or gradient descent, is an iterative optimization algorithm used for finding the local minimum of a function. In the context of smooth convex optimization, it is particularly effective.

Gradient Method for Smooth Convex Optimization:

- **Definition:** The gradient method is an iterative optimization algorithm used to find the minimum of a smooth convex function. It relies on calculating the gradient (derivative) of the objective function and moving in the direction of steepest descent.
- **Example Problem:**

  Consider the smooth convex function:

  $f(x) = x^2$

  The gradient of this function is $f'(x) = 2x$. Using the gradient method, the update rule for finding the minimum would be:

  $x_{n+1} = x_n \quad \alpha \cdot 2x_n$

  where α is the step size.
- **Solution:**

  Let's assume an initial value $x_0 = 5$ and a step size $\alpha = 0.1$. Then the iterations would be:

  - $x_1 = 5 \quad 0.1 \cdot 2 \cdot 5 = 4$
  - $x_2 = 4 \quad 0.1 \cdot 2 \cdot 4 = 3.2$
  - $x_3 = 3.2 \quad 0.1 \cdot 2 \cdot 3.2 = 2.56$

Problem 1:

Minimize the function $f(x) = x^2 + 3x + 5$ using Gradient Descent.

Solution:

1. Calculate the derivative of $f(x)$:

   $f'(x) = 2x + 3$
2. Choose an initial guess for x_0, say $x_0 = 3$.
3. Define the learning rate α, for example, $\alpha = 0.1$.
4. Apply the gradient descent update rule:

   $x_{n+1} = x_n \quad \alpha \cdot f'(x_n)$

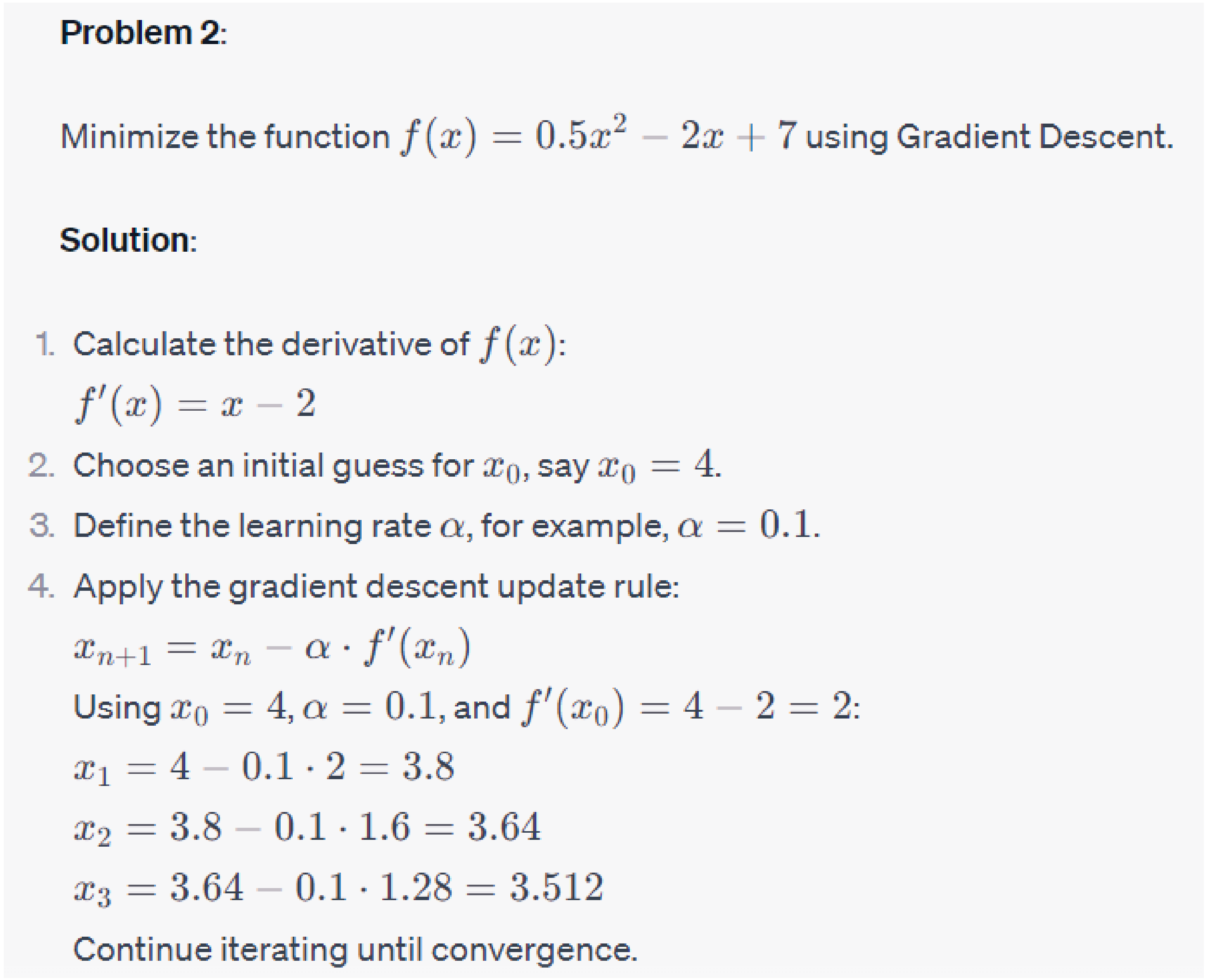
   Using $x_0 = 3$, $\alpha = 0.1$, and $f'(x_0) = 2 \cdot 3 + 3 = 9$:

   $x_1 = 3 \quad 0.1 \cdot 9 = 2.1$

   $x_2 = 2.1 \quad 0.1 \cdot 7.2 = 1.38$

   $x_3 = 1.38 \quad 0.1 \cdot 5.76 = 0.804$

   Continue iterating until convergence.



Problem 2:

Minimize the function $f(x) = 0.5x^2 - 2x + 7$ using Gradient Descent.

Solution:

1. Calculate the derivative of $f(x)$:
   $f'(x) = x - 2$
2. Choose an initial guess for x_0, say $x_0 = 4$.
3. Define the learning rate α, for example, $\alpha = 0.1$.
4. Apply the gradient descent update rule:
   $x_{n+1} = x_n - \alpha \cdot f'(x_n)$
   Using $x_0 = 4, \alpha = 0.1$, and $f'(x_0) = 4 - 2 = 2$:
   $x_1 = 4 - 0.1 \cdot 2 = 3.8$
   $x_2 = 3.8 - 0.1 \cdot 1.6 = 3.64$
   $x_3 = 3.64 - 0.1 \cdot 1.28 = 3.512$
   Continue iterating until convergence.

Non-smooth Convex Optimization:

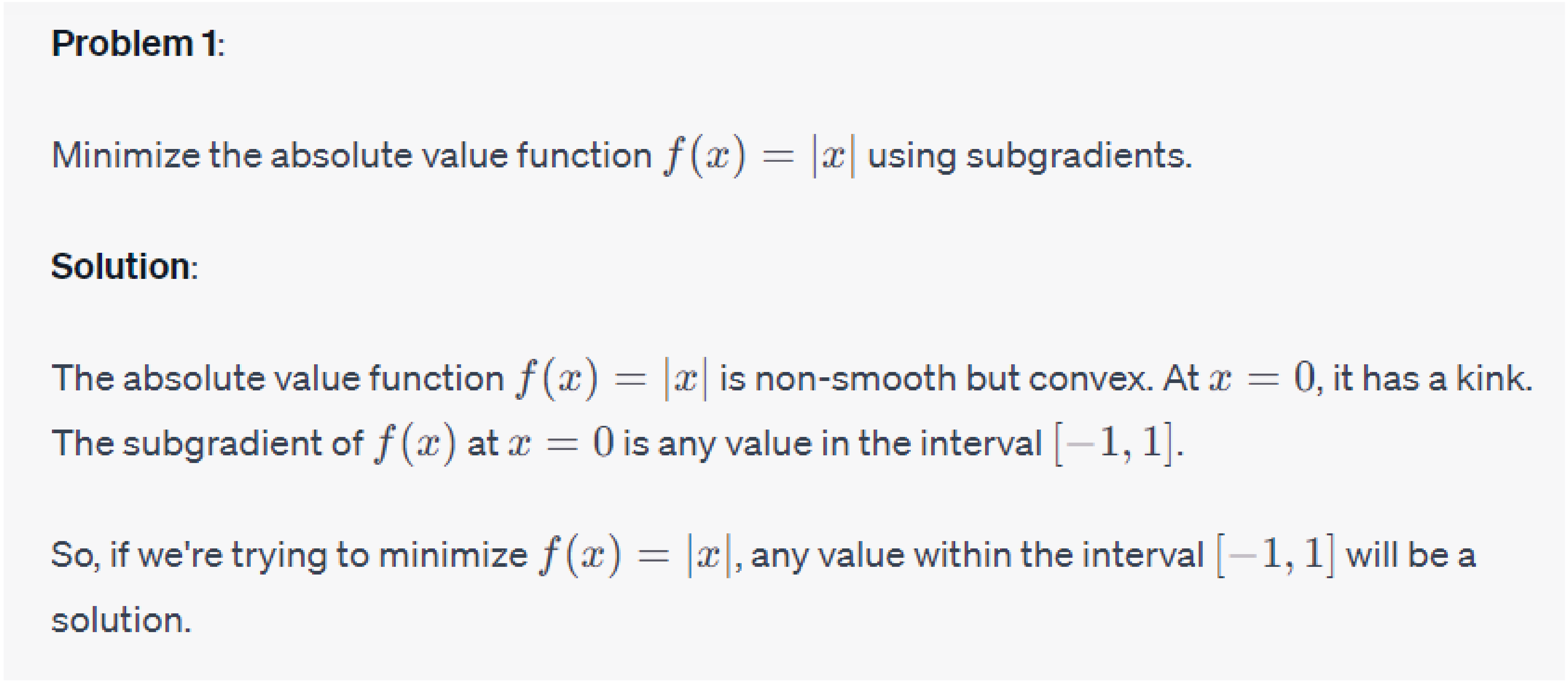
- **Definition:** Non-smooth convex optimization deals with optimizing functions that are convex but not necessarily smooth. These functions may have discontinuities or sharp points in their domain.
- **Example Problem:**

  Consider the non-smooth convex function:

  $f(x) = |x|$

  This function is convex but not differentiable at $x = 0$. It is a non-smooth convex function.
- **Solution:**

  As this is a non-smooth function, we don't use gradients but subgradients. At $x = 0$, the subgradient can be any value in the interval $[\quad 1, 1]$.



Problem 1:

Minimize the absolute value function $f(x) = |x|$ using subgradients.

Solution:

The absolute value function $f(x) = |x|$ is non-smooth but convex. At $x = 0$, it has a kink. The subgradient of $f(x)$ at $x = 0$ is any value in the interval $[-1, 1]$.

So, if we're trying to minimize $f(x) = |x|$, any value within the interval $[-1, 1]$ will be a solution.

Problem 2:

Minimize the function $f(x) = |x - 2| + 3$ using subgradients.

Solution:

The function $f(x) = |x - 2| + 3$ is non-smooth but convex. At $x = 2$, it has a kink. The subgradient of $f(x)$ at $x = 2$ is any value in the interval $[-1, 1]$.

So, if we're trying to minimize $f(x) = |x - 2| + 3$, any value within the interval $[-1, 1]$ will be a solution.

Constrained Convex Optimization:

- **Definition:** Constrained convex optimization is the process of minimizing a convex objective function subject to a set of constraints, which can be equality or inequality constraints. The goal is to find the best solution that satisfies all constraints.
- **Example Problem:**

  Consider the constrained convex optimization problem:

  Minimize $f(x) = x^2$ subject to the constraint $x \geq 1$.

  Here, the objective function $f(x) = x^2$ is convex. The constraint $x \geq 1$ is an inequality constraint.
- **Solution:**

  Since the constraint $x \geq 1$ is active, the minimum occurs at $x = 1$. So, the minimum value of $f(x)$ subject to the constraint is $f(1) = 1$.

Problem 1:

Minimize the function $f(x) = x^2 + 3x + 5$ subject to the constraint $x \geq 2$.

Solution:

1. Calculate the derivative of $f(x)$:
   $f'(x) = 2x + 3$
2. Since there is an inequality constraint $x \geq 2$, we need to check if the minimum occurs at the boundary point $x = 2$ or elsewhere.
   Evaluate $f(2) = 2^2 + 3 \cdot 2 + 5 = 15$.
3. If $f(2) = 15$ is the minimum, then $x = 2$ is the solution.
4. If not, find the critical point by setting $f'(x) = 0$:
   $2x + 3 = 0 \Rightarrow x = -1.5$.
5. Check if $x = -1.5$ satisfies the constraint $x \geq 2$. Since it doesn't, the minimum occurs at $x = 2$.

Problem 2:

Minimize the function $f(x) = 0.5x^2 - 2x + 7$ subject to the constraint $x \leq 3$.

Solution:

1. Calculate the derivative of $f(x)$:
   $f'(x) = x - 2$
2. Since there is an inequality constraint $x \leq 3$, we need to check if the minimum occurs at the boundary point $x = 3$ or elsewhere.
   Evaluate $f(3) = 0.5 \cdot 3^2 - 2 \cdot 3 + 7 = 4.5 - 6 + 7 = 5.5$.
3. If $f(3) = 5.5$ is the minimum, then $x = 3$ is the solution.
4. If not, find the critical point by setting $f'(x) = 0$:
   $x - 2 = 0 \Rightarrow x = 2$.
5. Check if $x = 2$ satisfies the constraint $x \leq 3$. Since it does, the minimum occurs at $x = 2$.

Topics

- Part – I
  - *Optimal* methods for unconstrained convex programs
    - Smooth objective
    - Non-smooth objective
- Part – II
  - *Optimal* methods for constrained convex programs
    - Projection based
    - Frank-Wolfe based
  - Prox-based methods for structured non-smooth programs

Non-Topics

- Step-size schemes
- Bundle methods
- Stochastic methods
- Inexact oracles
- Non-Euclidean extensions (Mirror-*friends*)

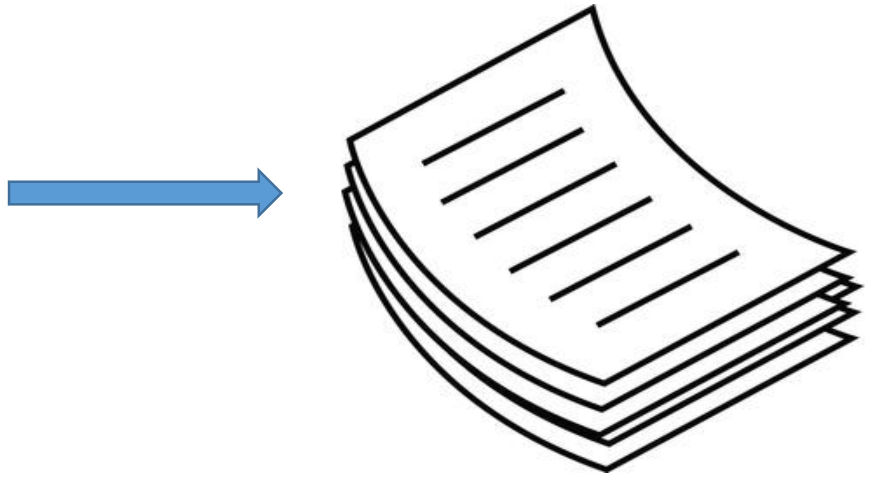
Motivation

& Example ApplicationS

Machine Learning Applications

- **Training data:** $\{ (x_1, y_1), \dots, (x_m, y_m)\}$
- **Goal:** Construct $f : X \to Y$

Machine Learning Applications

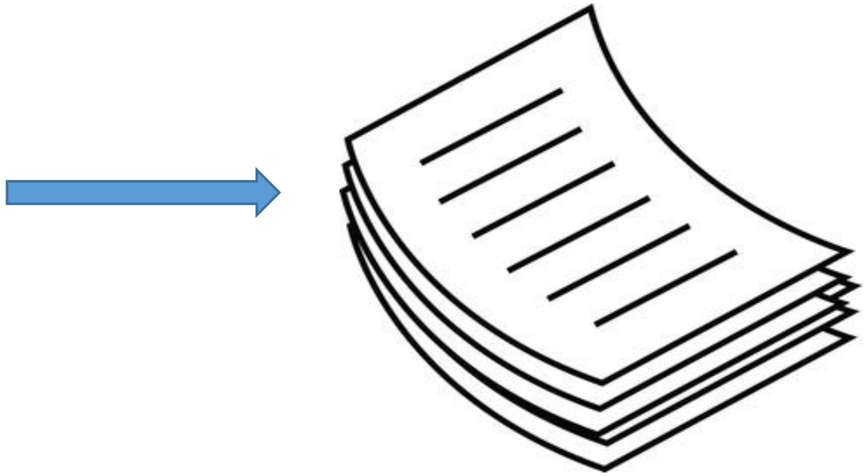


Set of Temple Images

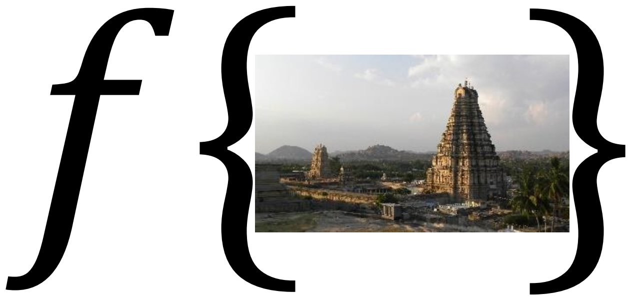
Corresponding Architecture Labels

Machine Learning Applications

Set of Temple Images



Corresponding Architecture Labels



= "Vijayanagara Style"

Machine Learning Applications

- **Input data:** $\{ (x_1, y_1), \ldots, (x_m, y_m)\}$
- **Goal:** Construct $f : X \xrightarrow{mXn} Y$
- **Model:** $f(x) = w^T \phi(x)$

Machine Learning Applications

- **Input data:** $\{ (x_1, y_1), \ldots, (x_m, y_m)\}$
- **Goal:** Construct $f : X \xrightarrow{mXn} Y$
- **Model:** $f(x) = w^T \phi(x)$
- **Algorithm:** Find simple functions that explain data

$$\min_{w \in W} \Omega(w) + \sum_{i=1}^{m} l(w^T \phi(x_i), y_i)$$

Typical Program – Machine Learning

$$\min_{w\in W} \overbrace{\Omega(w)}^{\text{Smooth/Non-Smooth}} + \sum_{i=1}^{m} \overbrace{l(w^T\phi(x_i), y_i)}^{\text{Smooth surrogate}}$$

- m, n are large
- Data term is a sum
- Domain W is restricted

Scale is the issue!

- m, n as well as no. models may run into millions!
- Even a single iteration of IPM/Newton-variants is in-feasible.
- “Slower” but “cheaper” methods are the alternative
  - Decomposition based
  - **First order methods**

First Order Methods - Overview

- Iterative, gradient-like information, $O(mn)$ per iteration ☺
- E.g. **Gradient method**, Cutting planes, Conjugate gradient
- Very old methods (1950s)

First Order Methods - Overview

- Iterative, gradient-like information, $O(mn)$ per iteration ☺
- E.g. **Gradient method**, Cutting planes, Conjugate gradient
- Very old methods (1950s)
- No. iterations:
  - Sub-linear rate ☹ .
  - But (nearly) n-independent ☺
- Widely employed in state-of-the-art ML systems
- Choice of variant depends on problem structure

Smooth un-constrained

$$\min_{w \in R^n} \sum_{i=1}^{m} (w^T \phi(x_i) - y_i)^2$$

Smooth Convex Functions

- Continuously differentiable
- Gradient is Lipschitz continuous

Smooth Convex Functions

- Continuously differentiable
- Gradient is Lipschitz continuous
  - $\|\nabla f(x) - \nabla f(y)\| \leq L\|x - y\|$
  - E.g. $g(x) \equiv x^2$ is not L-conts. over R but is over $[0,1]$ with L=2
  - E.g. $g(x) \equiv |x|$ is L-conts. with L=1

Smooth Convex Functions

- Continuously differentiable
- Gradient is Lipschitz continuous
  - $\|\nabla f(x) - \nabla f(y)\| \leq L\|x - y\|$
  - E.g. $g(x) \equiv x^2$ is not L-conts. over R but is over $[0,1]$ with L=2
  - E.g. $g(x) \equiv |x|$ is L-conts. with L=1

Theorem: Let f be convex twice differentiable. Then

$$f \text{ is smooth with const. } L \Leftrightarrow f''(x) \preccurlyeq L\, \mathrm{I}_n$$

Smooth Convex Functions

$$\min_{w \in R^n} \sum_{i=1}^{m} (w^T \phi(x_i) - y_i)^2$$

is indeed smooth!

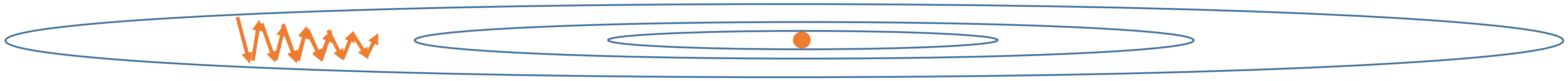
- Continuously differentiable
- Gradient is Lipschitz continuous
  - $\|\nabla f(x) - \nabla f(y)\| \leq L\|x - y\|$
  - E.g. $g(x) \equiv x^2$ is not L-conts. over R but is over $[0,1]$ with L=2
  - E.g. $g(x) \equiv |x|$ is L-conts. with L=1

Theorem: Let f be convex twice differentiable. Then

f is smooth with const. $L \Leftrightarrow \; f''(x) \preccurlyeq L\, \mathrm{I}_n$

Gradient Method [Cauchy1847]

- Move iterate in direction of instantaneous decrease
  - $x_{k+1} = x_k - s_k \nabla f(x_k)$, $\mathrm{s_k} > 0$

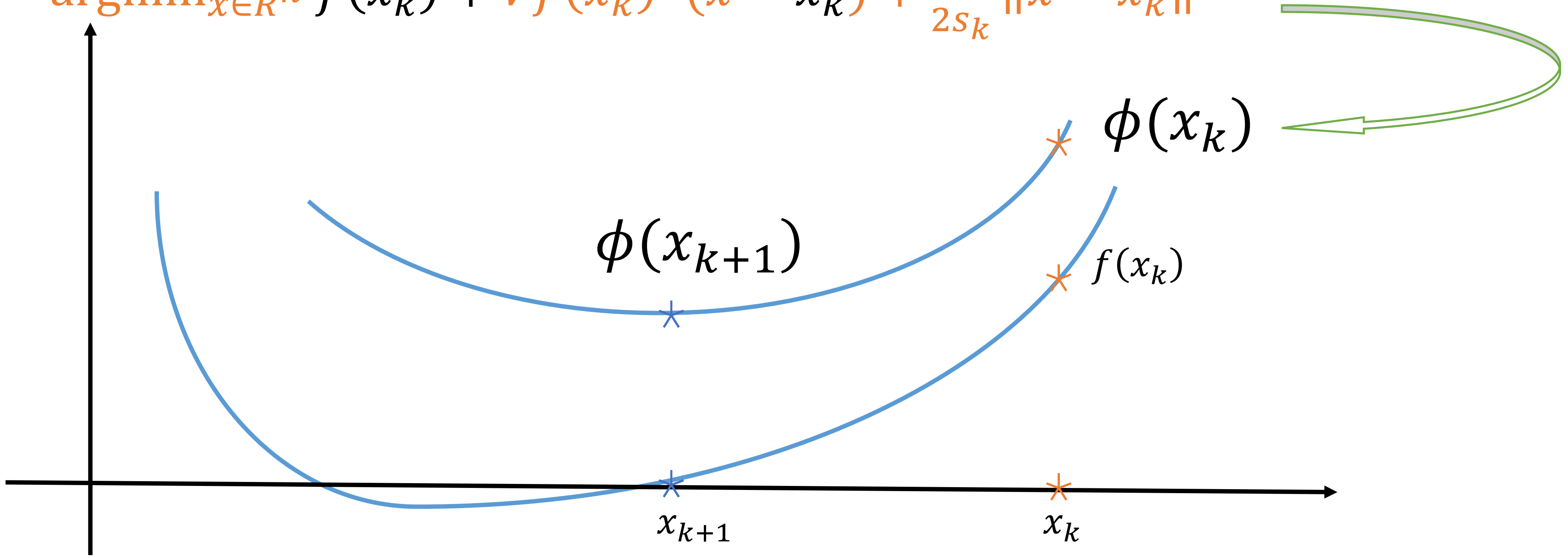


Gradient Method

- Move iterate in direction of instantaneous decrease
  - $x_{k+1} = x_k - s_k \nabla f(x_k),\ \mathrm{s_k} > 0$
- Regularized minimization of first order approx.
  - $x_{k+1} = \mathrm{argmin}_{x \in R^n}\ f(x_k) + \nabla f(x_k)^T (x - x_k) + \frac{1}{2s_k} \|x - x_k\|^2$

Gradient Method

- Move iterate in direction of instantaneous decrease
  - $x_{k+1} = x_k - s_k \nabla f(x_k),\ s_k > 0$
- Regularized minimization of first order approx.
  - $x_{k+1} = \operatorname{argmin}_{x \in R^n} f(x_k) + \nabla f(x_k)^T (x - x_k) + \frac{1}{2s_k} \|x - x_k\|^2$



THANK YOU