DSC 210 Numerical Linear Algebra, Fall 2025

Homework Problems for Topic 4: *ML and Least Squares Problems*

Student Name (PID): Kevin Lin (A69043483)

Write your solutions to the following problems by typing them in LaTeX. Unless otherwise noted by the problem's instructions, show your work and provide justification for your answer. Homework is due via Gradescope at **11th December 2025, 11:59 PM**.
Late Policy: If you submit your homework after the deadline we will apply a late penalty of 10% per day.

Guidelines for Homework Related Questions:

(a) As a general rule, we can help you understand the homework problems and explain the material from the corresponding lectures, but we cannot give you the entire solution.

(b) Regarding debugging programming questions: We ask you to do some debugging on your own first, including printing out intermediate values in your algorithms, trying a simpler version of the problem, etc.

(c) We will not be pre-grading the homework, i.e. we won't confirm if the answer you have is correct.

AI Usage Policy:

(a) Code: You may use LLMs to debug your code; however, you may not use LLMs to generate your entire code, and code must be reviewed and tested.

(b) Writing: You may use LLMs to correct grammar, style and latex issues; however, you may not use LLMs to generate entire solutions, sentences or paragraphs. All writing must be in your own voice.

Academic Integrity Policy:

The UC San Diego Academic Integrity Policy (formerly the Policy on Integrity of Scholarship) is effective as of September 25, 2023 and applies to any cases originating on or after September 25, 2023. The university expects both faculty and students to honor the policy. For students, this means that all academic work will be done by the individual to whom it's assigned, without unauthorized aid of any kind. If violations of academic integrity occur, the same Sanctioning Guidelines apply regardless of which policy was effective for that case.

For more information on how the policy is implemented, refer to the most current procedures. Remember: When in doubt about what constitutes appropriate collaboration or resource use, please ask TAs before proceeding. It's always better to clarify expectations than to risk an academic integrity violation. Academic integrity violations can have serious consequences for your academic record, and you will get zero grades.

You can access the Homework Template using the following link: `https://www.overleaf.com/read/vfhcmsppvskp`

Question 1: Least Squares Regression (10 points)

Consider a dataset with m datapoints: $(t_i, y_i), i = 1, ..., m$. We want to minimize least squares by fitting an estimation function

$$\hat{y}(t) = at^2 + bt + c$$

on this data. This problem can be written in the form $\mathbf{Ax} = \mathbf{y}$, such that the least squares solution for $\mathbf{x}$ gives the coefficients a, b, c.

What are $\mathbf{A}$, $\mathbf{x}$ and $\mathbf{y}$ in terms of the variables above?

Solution:
We can express the least squares problem in matrix form as follows:

$$\mathbf{A} = \begin{bmatrix} t_1^2 & t_1 & 1 \\ t_2^2 & t_2 & 1 \\ \vdots & \vdots & \vdots \\ t_m^2 & t_m & 1 \end{bmatrix}, \quad \mathbf{x} = \begin{bmatrix} a \\ b \\ c \end{bmatrix}, \quad \mathbf{y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_m \end{bmatrix}$$

Question 2: Linear Regression (30 points)

Use a linear regressor to fit the data, where $\hat{y}(t) = at + b$ with a, b being scalars. Let $\mathbf{x} = \begin{bmatrix} a \\ b \end{bmatrix}$ contain the regressor coefficients. Recall that the linear algebraic formula for least squares gives $\mathbf{x} = (A^\top A)^{-1} A^\top \mathbf{y}$ with $A^\dagger = (A^\top A)^{-1} A^\top$ known as the pseudo-inverse of A.

(a) Implement 3 different ways to find the regressor coefficients using the `numpy` package and show that they agree on the random data generated in the notebook (make sure to calculate both a and b):

- i Calculate $\mathbf{x}$ directly from the least squares formula.
- ii Use the function `np.linalg.pinv` to find the values of regressor coefficients $\mathbf{x}$.
- iii Solve the problem using the built-in `numpy` function: `np.linalg.lstsq`

(b) Draw a scatter plot between $\mathbf{T}$ and $\mathbf{y}$, and overlay it with the linear regression line.

Solution: We have provided a starter code to generate the datapoints. **Do not modify the provided code block -** google colab link.

(For all the following questions, make a local copy of the colab notebook. Write the solution in the copy and submit.)

Question 3: Logarithmic Regression (30 points)

(a) Write a function `my_func_fit(T,y)`, where **T** and **y** are column vectors of the same size containing experimental data. The function should return the values for α and β which are the scalar parameters of the estimation function

$$\hat{y}(t) = \alpha.t^{\beta}$$

Hint: Minimize least squares in log-space, i.e. $\min \sum_i (\log(\hat{y_i}) - \log(y_i))^2$.

(b) Test your code on the generated sample dataset and report the coefficients. A starter code to generate the logarithmic dataset is provided in the notebook.

(c) Plot a graph between **T** vs **y**, and overlay it with the regression line.

Note: You are only allowed to use `numpy` library functions.

Solution: Provide the solution in the notebook google colab.

Question 4: Functional Regression (30 points)

(a) Write a function `my_lin_regression(f, T, y)`, where f is a list containing function objects to pre-defined basis functions and, $\mathbf{T}$ and $\mathbf{y}$ are arrays containing noisy data. Assume that $\mathbf{T}$ and $\mathbf{y}$ are the same size, i.e, $T^{(i)} \in \mathbb{R}, y^{(i)} \in \mathbb{R}$.

Return an array β which represent the coefficients of the solved problem. We are solving for the β which contains the coefficients in the regressor

$$\hat{y}(t) = \beta_0 + \beta_1 f_1(t) + \cdots + \beta_n f_n(t)$$

with f_i being the pre-defined basis functions.

(b) Write a function `regression_plot(f,T,y,beta)` which plots a graph between $\mathbf{T}$ and $\mathbf{y}$, and overlays it with the regression line.

Run the provided test scenarios provided in the notebook. First one uses $f = [\sin, \cos]$ and second $f = [\exp]$.

Note: You are only allowed to use `numpy` library functions.

Solution: Provide the solution in the notebook google colab.

(BONUS)Question 5: Ridge Regression (10 points)

Let $A \in \mathbb{R}^{n\times d}$ and $y \in \mathbb{R}^n$, where $n > d$. Assume that A is rank-deficient, i.e. $\operatorname{rank}(A) = r < d$. For $\lambda > 0$, consider the ridge regression problem

$$L_\lambda(w) = \|Aw - y\|_2^2 + \lambda\|w\|_2^2, \quad w \in \mathbb{R}^d.$$

(a) **Gradient and Hessian.**

i. Compute the gradient $\nabla L_\lambda(w)$.

ii. Compute the Hessian $H_\lambda = \nabla^2 L_\lambda(w)$.

(b) **Closed-form ridge solution via normal equations.** Show that any minimizer $w_\lambda^\star$ of L_λ satisfies the linear system

$$(A^\top A + \lambda I_d)\, w_\lambda^\star = A^\top y,$$

and hence

$$w_\lambda^\star = (A^\top A + \lambda I_d)^{-1} A^\top y.$$

(c) **SVD and spectrum of the Hessian.** Let the singular value decomposition (SVD) of A be

$$A = U\Sigma V^\top,$$

where $U \in \mathbb{R}^{n\times r}$ and $V \in \mathbb{R}^{d\times r}$ have orthonormal columns, and

$$\Sigma = \operatorname{diag}(\sigma_1, \ldots, \sigma_r), \quad \sigma_1 \geq \cdots \geq \sigma_r > 0.$$

i. Express $A^\top A$ and H_λ in terms of U, Σ, and V.

ii. Show that the eigenvalues of H_λ are

$$2(\sigma_1^2 + \lambda), \ldots, 2(\sigma_r^2 + \lambda), \underbrace{2\lambda, \ldots, 2\lambda}_{d-r \text{ times}}.$$

(d) **Ridge solution in the SVD basis and relation to the pseudo-inverse.**

i. Using the SVD of A, derive an explicit formula for $w_\lambda^\star$ of the form

$$w_\lambda^\star = \sum_{i=1}^{r} \alpha_i(\lambda)\, v_i,$$

where v_i are the columns of V, and identify the coefficients $\alpha_i(\lambda)$ in terms of σ_i and $U^\top y$.

ii. Show that, as $\lambda \to 0^+$,

$$w_\lambda^\star \to w_0^\star := A^+ y,$$

the minimum-norm least squares solution. Here A^+ is the pseudo-inverse of A.

(e) Verify the solutions with the code template provided in the notebook.

Note: You are only allowed to use `numpy` library functions.

Solution: Typewrite the solution for parts (a) to (d) in latex and code the solution to (e) in the notebook google colab.

(a) **Gradient and Hessian.**

i. We compute the gradient $\nabla L_\lambda(w)$ by taking the derivative of $L_\lambda(w)$ with respect to w:

$$\nabla L_\lambda(w) = 2A^\top(Aw - y) + 2\lambda w.$$

ii. We compute the hessian $\nabla^2 L_\lambda(w)$ by taking the derivative of the gradient with respect to w:

$$H_\lambda = 2A^\top A + 2\lambda I_d.$$

(b) **Closed-form ridge solution via normal equations.** To find the minimizer $w_\lambda^\star$ of $L_\lambda(w)$, we set the gradient to zero:

$$\nabla L_\lambda(w_\lambda^\star) = 0 \implies 2A^\top(Aw_\lambda^\star - y) + 2\lambda w_\lambda^\star = 0.$$

Rearranging gives:

$$(A^\top A + \lambda I_d)w_\lambda^\star = A^\top y.$$

Since $A^\top A + \lambda I_d$ is invertible for $\lambda > 0$, we can solve for $w_\lambda^\star$:

$$w_\lambda^\star = (A^\top A + \lambda I_d)^{-1} A^\top y.$$

(c) **SVD and spectrum of the Hessian.**

i. Using the SVD of A, we have:

$$A^\top A = V\Sigma^\top U^\top U\Sigma V^\top = V\Sigma^2 V^\top,$$

since $U^\top U = I_r$. Therefore, the Hessian can be expressed as:

$$H_\lambda = 2V\Sigma^2 V^\top + 2\lambda I_d.$$

ii. The eigenvalues of H_λ can be found by considering the action of H_λ on the basis vectors v_i:

$$H_\lambda v_i = 2(\sigma_i^2 + \lambda)v_i \quad \text{for } i = 1, \ldots, r,$$

and for the remaining $d - r$ dimensions, we have:

$$H_\lambda v_i = 2\lambda v_i \quad \text{for } i = r+1, \ldots, d.$$

Thus, the eigenvalues of H_λ are:

$$2(\sigma_1^2 + \lambda), \ldots, 2(\sigma_r^2 + \lambda), \underbrace{2\lambda, \ldots, 2\lambda}_{d-r \text{ times}}.$$

(d) **Ridge solution in the SVD basis and relation to the pseudo-inverse.**

i. Using the SVD of A, we can express $w_\lambda^\star$ as:

$$w_\lambda^\star = V(\Sigma^2 + \lambda I_r)^{-1}\Sigma U^\top y.$$

Expanding this, we get:

$$w_\lambda^\star = \sum_{i=1}^{r} \frac{\sigma_i}{\sigma_i^2 + \lambda}(u_i^\top y)v_i,$$

where u_i are the columns of U. Thus, the coefficients $\alpha_i(\lambda)$ are given by:

$$\alpha_i(\lambda) = \frac{\sigma_i}{\sigma_i^2 + \lambda}(u_i^\top y).$$

ii. As $\lambda \to 0^+$, we have:

$$\alpha_i(\lambda) \to \frac{1}{\sigma_i}(u_i^\top y),$$

which corresponds to the coefficients of the minimum-norm least squares solution:

$$w_0^\star = A^+ y = \sum_{i=1}^{r} \frac{1}{\sigma_i}(u_i^\top y)v_i.$$

Homework 4

Notes:

- Use proper LATEX formatting and notation for all mathematical equations, vectors, and matrices.
- For programming solutions: Properly add comments to your code.

Question 2: Linear Regression (30 points)

In this problem, we would like to use a linear regressor to fit the data, where $\hat{y}(t) = at + b$ with a, b being scalars. Let $\mathbf{x} = \begin{bmatrix} a \\ b \end{bmatrix}$ contain the regressor coefficients. Recall that the linear algebraic formula for least squares gives $\mathbf{x} = (A^\top A)^{-1} A^\top \mathbf{y}$ with $A^\dagger = (A^\top A)^{-1} A^\top$ known as the pseudo-inverse of A.

a) Implement 3 different ways to find the regressor coefficients using the numpy package and show that they agree on the random data generated below (make sure to calculate both a and b):

- **i.** Calculate $\mathbf{x}$ directly from the least squares formula.
- **ii.** Use the function `np.linalg.pinv` to find the values of regressor coefficients $\mathbf{x}$.
- **iii.** Solve the problem using the builtin numpy function: `np.linalg.lstsq`

b) Plot a graph between $\mathbf{T}$ and $\mathbf{y}$, and overlay it with the linear regression line.

In [2]:
```
### !!! DO NOT EDIT !!!
# starter code to generate a random least squares regression dataset with 500 point
import numpy as np
import matplotlib.pyplot as plt
from sklearn import datasets

# generate T and y
T, y =  datasets.make_regression(n_samples=500, n_features=1, n_informative=1, n_ta
print('Shape of T is:', T.shape)
print('Shape of y is:', y.shape)
```

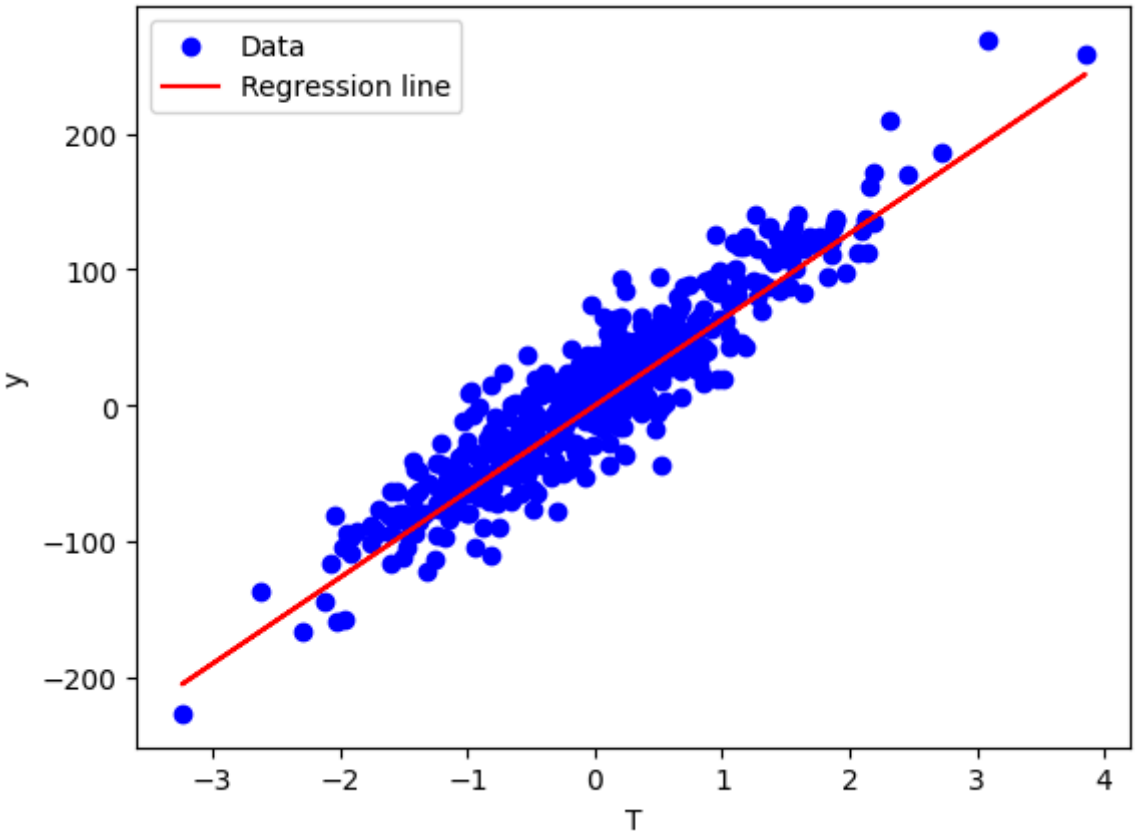
```
Shape of T is: (500, 1)
Shape of y is: (500,)
```

In [3]:
```
#######
# !!! YOUR CODE HERE !!!
# a
# i: calculate x directly using LS formula
x = np.linalg.inv(T.T @ T) @ T.T @ y
print(x)
```

```
# ii: use np.linalg.pinv to find values of regressor coefficients x
x_pinv = np.linalg.pinv(T) @ y
print(x_pinv)
# iii: use np.linalg.lstsg
x_lstsq, residuals, rank, s = np.linalg.lstsq(T, y)
print(x_lstsq)

# b
# plot the data points and the regression line
plt.scatter(T, y, color="blue", label="Data")
plt.plot(T, T * x, color="red", label="Regression line")
plt.xlabel("T")
plt.ylabel("y")
plt.legend()
plt.show()
```

```
[63.2502431]
[63.2502431]
[63.2502431]
```



Question 3: Logarithmic Regression (30 points)

a) Write a function `my_func_fit (T,y)`, where $\mathbf{T}$ and $\mathbf{y}$ are column vectors of the same size containing experimental data. The function should return the values for α and β which are the scalar parameters of the estimation function

$$\hat{y}(t) = \alpha t^{\beta}$$

.

Hint: Minimize least squares in log-space, i.e. $\min \sum_i (\log(\hat{y}_i) - \log(y_i))^2$

b) Test your code on the generated sample dataset and report the coefficients. The given piece of starter code generates a logarithmic dataset.

c) Draw a scatter plot between $\mathbf{T}$ vs $\mathbf{y}$, and overlay it with the regression line.

You are only allowed to use numpy library functions.

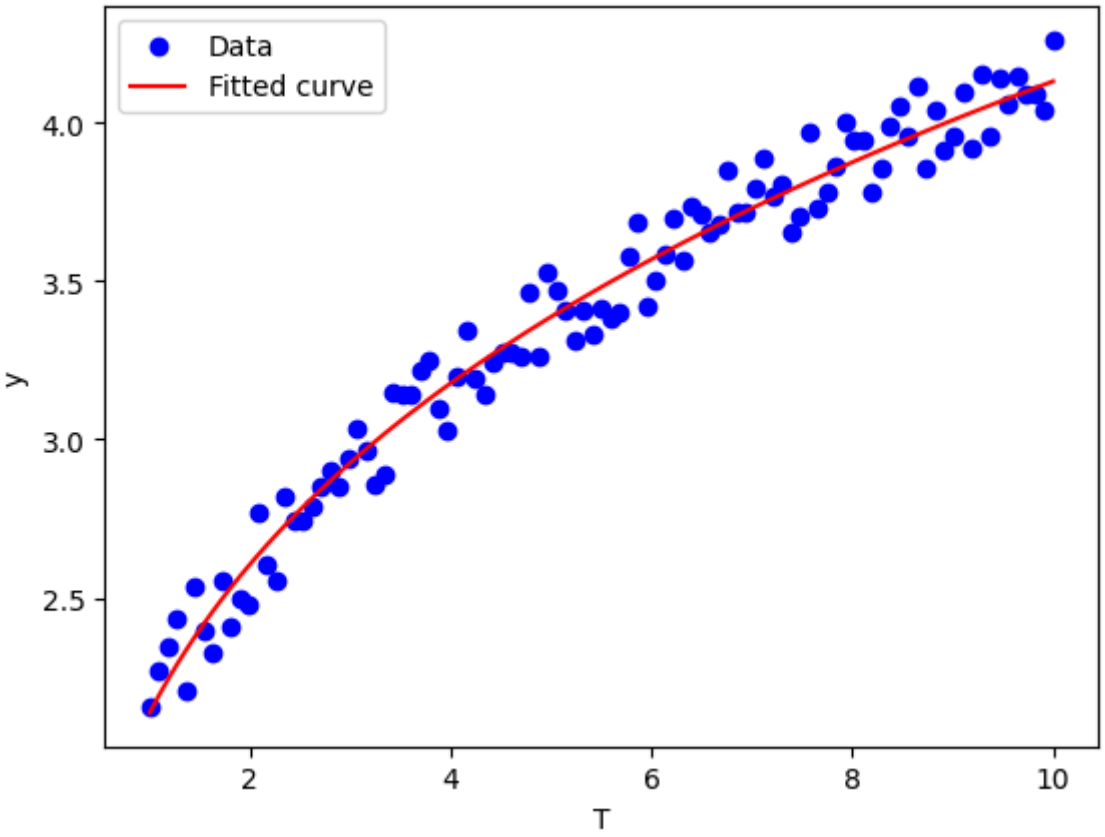
In [4]:
```
### !!! DO NOT EDIT !!!
# starter code to generate a random exponential dataset
T = np.linspace(1, 10, 101)
y = 2*(T**(0.3)) + 0.3*np.random.random(len(T))
print('Shape of T is:', T.shape)
print('Shape of y is:', y.shape)
```

```
Shape of T is: (101,)
Shape of y is: (101,)
```

In [5]:
```
#######
# !!! YOUR CODE HERE !!!
# a
def my_func_fit(T, y):
    log_T = np.log(T).reshape(-1, 1)
    log_y = np.log(y).reshape(-1, 1)
    A = np.hstack([log_T, np.ones_like(log_T)])
    coeffs, residuals, rank, s = np.linalg.lstsq(A, log_y)
    return np.exp(coeffs[1, 0]), coeffs[0, 0]

# b
a, b = my_func_fit(T, y)
print(f"Fitted parameters: a = {a}, b = {b}")
# c
y_fit = a * (T ** b)
plt.scatter(T, y, color="blue", label="Data")
plt.plot(T, y_fit, color="red", label="Fitted curve")
plt.xlabel("T")
plt.ylabel("y")
plt.legend()
plt.show()
```

```
Fitted parameters: a = 2.136909436016623, b = 0.2862605901184993
```



Question 4: Functional Regression (30 points)

a) Write a function `my_lin_regression(f, T, y)`, where `f` is a list containing function objects to pred-defined basis functions, and $\mathbf{T}$ and $\mathbf{y}$ are arrays containing noisy data. Assume that $\mathbf{T}$ and $\mathbf{y}$ are the same size, i.e, $T^{(i)} \in \mathbb{R}, y^{(i)} \in \mathbb{R}$.

Return an array `beta` which represents the coefficients of the solved problem. We are solving for β which contains the coefficients in the regressor

$$\hat{y}(t) = \beta_0 + \beta_1 f_1(t) + \cdots + \beta_n f_n(t)$$

with f_i being basis functions.

b) Also write a function `regression_plot(f,T,y,beta)` which plots a graph between $\mathbf{T}$ and $\mathbf{y}$, and overlays it with the regression line.

Run the provided test scenarios provided below. First one uses f = [sin, cos] and second f = [exp]. Your code should plot regression lines that fit the data nicely.

You are only allowed to use numpy library functions.

In [6]:
```
#######
# !!! YOUR CODE HERE !!!
def my_lin_regression(f, T, y):
```

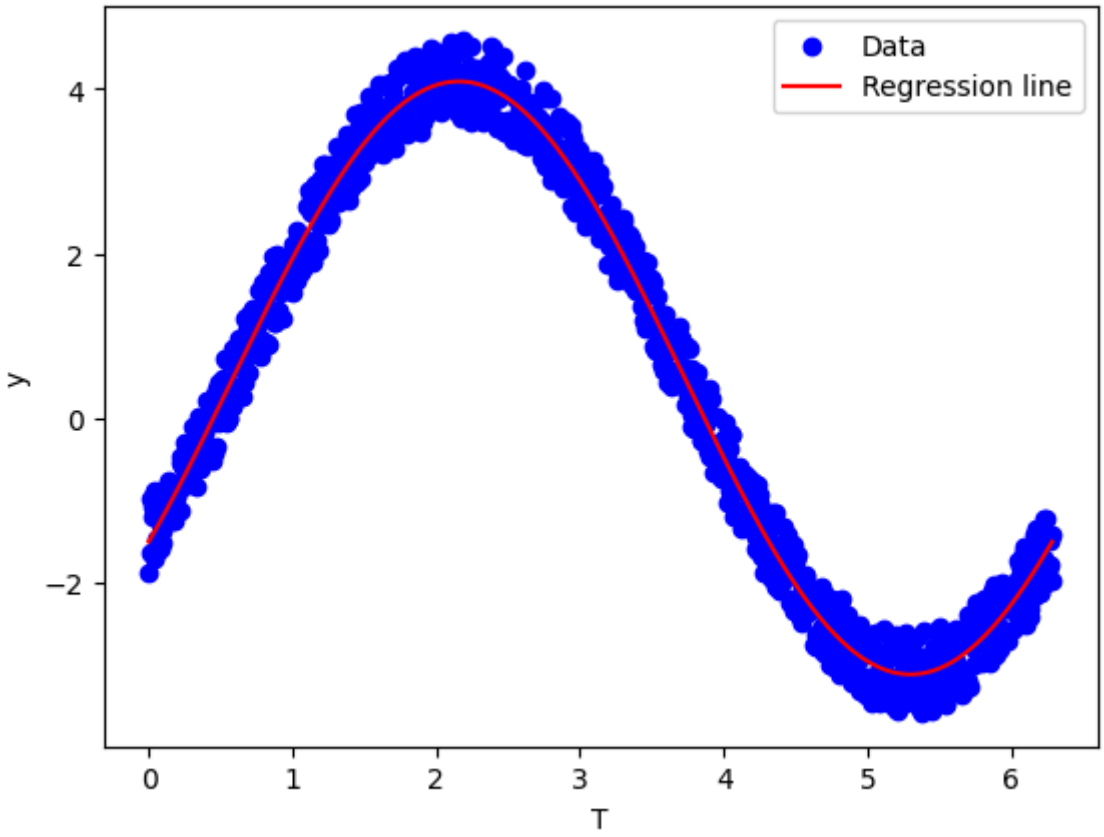
```
  n = len(f)
  m = len(T)
  A = np.zeros((m, n))
  for i in range(n):
    A[:, i] = f[i](T)
  A = np.hstack([np.ones((m, 1)), A])
  beta, residuals, rank, s = np.linalg.lstsq(A, y)
  return beta

def regression_plot(f, T, y, beta):
  plt.scatter(T, y, color="blue", label="Data")
  y_fit = np.zeros_like(y)
  y_fit += beta[0]
  for i in range(len(f)):
    y_fit += beta[i + 1] * f[i](T)
  plt.plot(T, y_fit, color="red", label="Regression line")
  plt.xlabel("T")
  plt.ylabel("y")
  plt.legend()
  plt.show()
#######
```

In [7]:

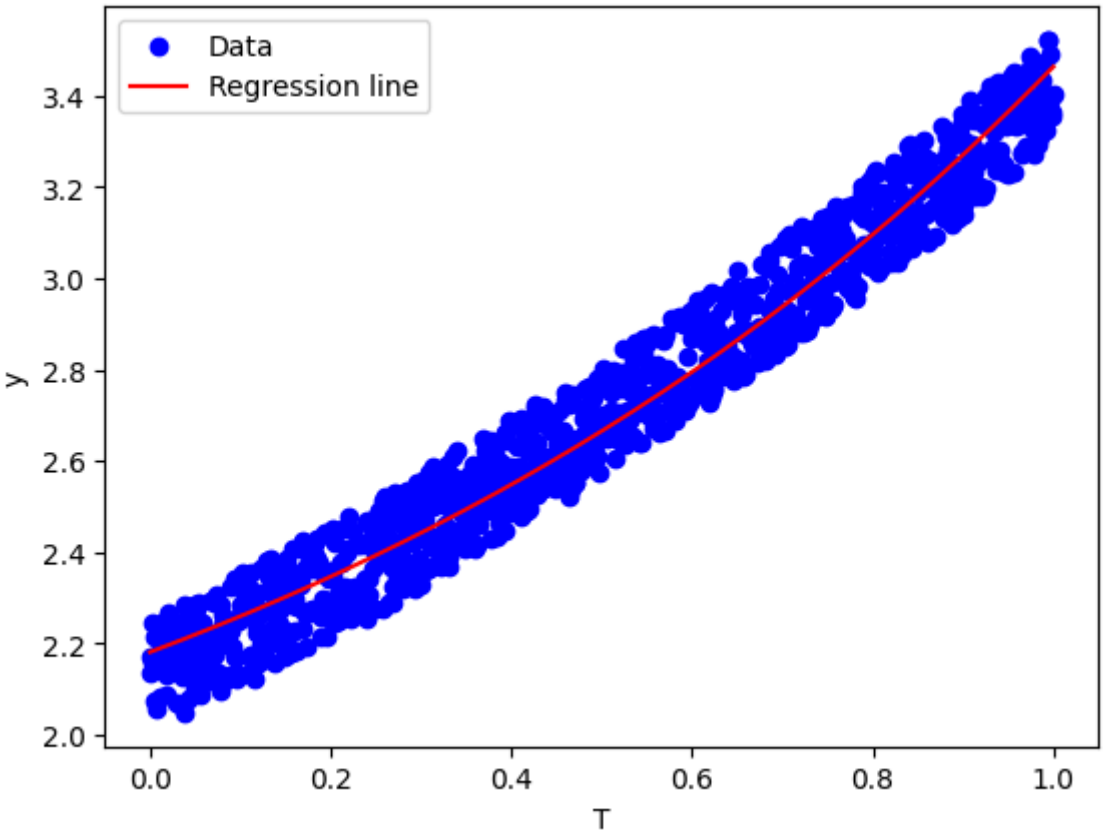
```
### !!! DO NOT EDIT !!!
### Test-1
T = np.linspace(0, 2*np.pi, 1000)
y = 3*np.sin(T) - 2*np.cos(T) + np.random.random(len(T))
f = [np.sin, np.cos] # f1 = sin, f2 = cos

beta = my_lin_regression(f, T, y)
regression_plot(f,T,y,beta)
```



In [8]:
```
### !!! DO NOT EDIT !!!
### Test-2
T = np.linspace(0, 1, 1000)
y = 2*np.exp(0.5*T) + 0.25*np.random.random(len(T))
f = [np.exp] # f1 = exp

beta = my_lin_regression(f, T, y)
regression_plot(f,T,y,beta)
```



(BONUS) Question 5: Ridge Regression (10 points)

Let $A \in \mathbb{R}^{n \times d}$ and $y \in \mathbb{R}^n$. Assume that A is rank-deficient, i.e.rank$(A) = r < d$. For $\lambda > 0$, consider the ridge regression problem

$$L_\lambda(w) = \|Aw - y\|_2^2 + \lambda \|w\|_2^2, \quad w \in \mathbb{R}^d.$$

a) Gradient and Hessian.

- **i.** Compute the gradient $\nabla L_\lambda(w)$.
- **ii.** Compute the Hessian $H_\lambda = \nabla^2 L_\lambda(w)$.

b) Closed-form ridge solution via normal equations.

Show that any minimizer $w_\lambda^\star$ of L_λ satisfies the linear system

$$(A^\top A + \lambda I_d)\, w_\lambda^\star = A^\top y,$$

and hence

$$w_\lambda^\star = (A^\top A + \lambda I_d)^{-1} A^\top y.$$

c) SVD and spectrum of the Hessian.

Let the singular value decomposition (SVD) of A be

$$A = U\Sigma V^\top,$$

where $U \in \mathbb{R}^{n\times r}$ and $V \in \mathbb{R}^{d\times r}$ have orthonormal columns, and

$$\Sigma = \mathrm{diag}(\sigma_1, \ldots, \sigma_r), \quad \sigma_1 \geq \cdots \geq \sigma_r > 0.$$

- **i.** Express $A^\top A$ and H_λ in terms of U, Σ, and V.
- **ii.** Show that the eigenvalues of H_λ are

$$2(\sigma_1^2 + \lambda), \ldots, 2(\sigma_r^2 + \lambda), \underbrace{2\lambda, \ldots, 2\lambda}_{d-r \text{ times}}.$$

d) Ridge solution in the SVD basis and relation to the pseudo-inverse.

Using the SVD of A, derive an explicit formula for $w_\lambda^\star$ of the form

$$w_\lambda^\star = \sum_{i=1}^{r} \alpha_i(\lambda)\, v_i,$$

where v_i are the columns of V, and identify the coefficients $\alpha_i(\lambda)$ in terms of σ_i and $U^\top y$. \item Show that, as $\lambda \to 0^+$,

$$w_\lambda^\star \to w_0^\star := A^+ y,$$

the minimum-norm least squares solution. Here A^+ is the pseudo-inverse of A.

e) Verify the solutions with the code template provided below.

In [9]:
```
### !!! DO NOT EDIT !!!
rng = np.random.default_rng(0)
n = 1000  # n samples
d = 6     # features
r = 4     # rank r < d

# Building a rank-deficient A
A_full = rng.normal(size=(n, r))
Q = rng.normal(size=(r, d))

A = A_full @ Q
# Verification
rank_A = np.linalg.matrix_rank(A)
print(f"A shape = {A.shape}, numerical rank = {rank_A}")

y = rng.normal(size=n)

lamda = 1.25
```

```
A shape = (1000, 6), numerical rank = 4
```

In [20]:
```
# (b) Closed-form ridge solution via normal equations.
# Calculate the w^* or w_optimal using the formula and using numpy
# Compare
I = np.eye(d)
```

```
w_optimal = np.linalg.inv(A.T @ A + lamda * I) @ A.T @ y
w_ridge_np = np.linalg.lstsq(A.T @ A + lamda * I, A.T @ y)[0]
print(np.allclose(w_optimal, w_ridge_np))
```

```
True
```

In [21]:
```
# (c) SVD and spectrum of the Hessian

# (i)
# Calculate A^TA using the formula and using numpy
# Compare
A_TA_formula = A.T @ A
A_TA_np = np.linalg.multi_dot([A.T, A])
print(np.allclose(A_TA_formula, A_TA_np))

# Calculate Hessian using the above values of A^TA
# Compare
Hessian_formula = A_TA_formula + lamda * I
Hessian_np = A_TA_np + lamda * I
print(np.allclose(Hessian_formula, Hessian_np))

# (ii)
# Calculate eigenvalues of Hessian using the formula and using numpy
# Compare
eigenvalues_formula = np.linalg.eigvals(Hessian_formula)
eigenvalues_np = np.linalg.eigvals(Hessian_np)
print(np.allclose(eigenvalues_formula, eigenvalues_np))
```

```
True
True
True
```

In [22]:
```
# (d) Ridge solution in SVD basis and relation to pseudo-inverse

# (i)
# Calculate the w_optimal_svd using
# w_optimal_svd = \sum_{i=1}^r \alpha_i(\lambda)\, v_i,
w_optimal_svd = np.zeros(d)
U, S, VT = np.linalg.svd(A, full_matrices=False)
for i in range(len(S)):
    vi = VT[i, :]
    ui = U[:, i]
    si = S[i]
    alpha_i = (si / (si**2 + lamda)) * (ui.T @ y)
    w_optimal_svd += alpha_i * vi

# Compare with w_optimal from part (b)

# (ii)
# Verify the claim
print(np.allclose(w_optimal_svd, w_optimal))
```

```
True
```
