Deep learning and inverse problems **Exercise 8**:

Problem 1.1

$$\frac{1}{2}\|Ax-y\|_2^2 + \lambda\cdot\frac{1}{2}\cdot\|x\|_2^2$$
$$= \frac{1}{2}(Ax-y)^T\cdot(Ax-y) + \lambda\cdot\frac{1}{2}\cdot x^T\cdot x$$
$$= \frac{1}{2}\cdot x^TA^TAx - \frac{1}{2}\cdot x^TA^Ty - \frac{1}{2}\cdot y^TAx + \frac{1}{2}y^Ty + \frac{1}{2}\cdot\lambda x^Tx$$

$$\frac{\partial}{\partial x}\left[\frac{1}{2}\|Ax-y\|_2^2 + \lambda\cdot\frac{1}{2}\cdot\|x\|_2^2\right]$$
$$= A^TAx - \frac{1}{2}\cdot A^Ty - \frac{1}{2}\cdot A^Ty$$
$$\overset{!}{=} 0 \quad \text{minimize it.}$$

$$\Rightarrow (A^TA+\lambda I)\cdot x = A^Ty.$$
$$\Rightarrow \hat{x}_{2,\lambda} = (A^TA+\lambda\cdot I)^{-1}\cdot A^Ty$$
$$\underset{\overline{\overline{\hphantom{AAA}}}}{\overset{A=U\Sigma V^T}{=}} (V\Sigma^TU^T\cdot U\Sigma V^T + \lambda I)^{-1}\cdot V\Sigma^TU^T\cdot y.$$
$$= (V\Sigma^T\Sigma V^T+\lambda I)^{-1}\cdot V\Sigma^TU^T\cdot y$$
$$= (V\Sigma^T\Sigma V^T + V\cdot\lambda IV^T)^{-1}\cdot V\Sigma^TU^T\cdot y$$
$$= [V\cdot(\Sigma^T\Sigma+\lambda I)V^T]^{-1}\cdot V\Sigma^TU^T\cdot y$$
$$= (V^T)^{-1}\cdot(\Sigma^T\Sigma+\lambda I)^{-1}\cdot V^T\cdot V\Sigma^TU^T\cdot y$$
$$= V\cdot(\Sigma^T\Sigma+\lambda I)^{-1}\cdot\Sigma^TU^T\cdot y.$$

Problem 1.2

$$\|\hat{x}_{2,\lambda}(Ax+e) - x\|_2^2$$
$$= \|V\cdot(\Sigma^T\Sigma+\lambda I)^{-1}\cdot\Sigma^TU^T\cdot(Ax+e) - x\|_2^2$$
$$= \|V\cdot(\Sigma^T\Sigma+\lambda I)^{-1}\cdot\Sigma^TU^T\cdot(U\Sigma V^T x+e) - x\|_2^2$$
$$= \|V\cdot(\Sigma^T\Sigma+\lambda I)^{-1}\cdot\Sigma^T\Sigma V^Tx + V\cdot(\Sigma^T\Sigma+\lambda I)^{-1}\cdot\Sigma^TU^Te - x\|_2^2$$
$$= \|V\cdot(\Sigma^T\Sigma+\lambda I)^{-1}\cdot\Sigma^T\Sigma V^Tx + V\cdot(\Sigma^T\Sigma+\lambda I)^{-1}\cdot\lambda IV^Tx - V\cdot(\Sigma^T\Sigma+\lambda I)^{-1}\cdot\lambda IV^Tx$$
$$+ V\cdot(\Sigma^T\Sigma+\lambda I)^{-1}\Sigma^TU^Te - x\|_2^2$$
$$= \|V\cdot(\Sigma^T\Sigma+\lambda I)^{-1}\cdot(\Sigma^T\Sigma+\lambda I)V^Tx - V\cdot(\Sigma^T\Sigma+\lambda I)^{-1}\cdot\lambda IV^Tx + V\cdot(\Sigma^T\Sigma+\lambda I)^{-1}\Sigma^TU^Te - x\|_2^2$$
$$= \|\underbrace{V\cdot(\Sigma^T\Sigma+\lambda I)^{-1}\cdot\Sigma^TU^Te}_{\text{this term controls the value of reconstruction error.}} - V\cdot(\Sigma^T\Sigma+\lambda I)^{-1}\cdot\lambda IV^Tx\|_2^2$$

because of the Cauchy-Schwarz inequality
$\|u+v\| \le \|u\| + \|v\|$

$$V\cdot(\Sigma^T\Sigma+\lambda I)^{-1}\cdot\Sigma^TU^Te$$
$$= V\cdot\left(\underbrace{\begin{bmatrix}\sigma_1 & & & \\ & \sigma_2 & & \\ & & \ddots & \\ & & & \sigma_m \\ \hline & & 0 & \end{bmatrix}}_{\Sigma^T}\cdot\underbrace{\left[\begin{array}{cccc|c}\sigma_1 & & & & \\ & \sigma_2 & & & \\ & & \ddots & & 0 \\ & & & \sigma_m & \end{array}\right]}_{\Sigma} + \lambda I\right)^{-1}\cdot\underbrace{\begin{bmatrix}\sigma_1 & & & \\ & \sigma_2 & & \\ & & \ddots & \\ & & & \sigma_m \\ \hline & & 0 & \end{bmatrix}}_{\Sigma^T}\cdot U^T\cdot e$$

$$\overset{n\times n}{}\qquad \overset{n\times m}{}\qquad \overset{m\times m}{}$$

$$= V\cdot\begin{bmatrix}\frac{\sigma_1}{\sigma_1^2+\lambda} & & & \\ & \frac{\sigma_2}{\sigma_2^2+\lambda} & & \\ & & \ddots & \\ & & & \frac{\sigma_m}{\sigma_m^2+\lambda} \\ \hline & & 0 & \end{bmatrix}\cdot U^T\cdot e$$

similar to the worst-case example in lecture, a form of "SVD".

$\Rightarrow$ the worst-case perturbation is $\hat{e} = \epsilon\cdot u_{min}$, u_{min} is the left singular vector of A corresponding to the smallest singular value.

$$\|\hat{x}_{2,\lambda}(Ax+e) - x\|_2^2$$
$$= \|V\cdot(\Sigma^T\Sigma+\lambda I)^{-1}\cdot(\Sigma^TU^Te - \lambda IV^Tx)\|_2^2$$
$$\underset{\overline{\overline{\hphantom{AAAAAAA}}}}{\overset{V\text{ orthogonal \& unitary}}{=}} \|(\Sigma^T\Sigma+\lambda I)^{-1}\cdot(\Sigma^TU^Te - \lambda I\cdot V^Tx)\|_2^2$$

$$= \left\|\begin{bmatrix}0\\0\\\vdots\\\frac{\sigma_{min}}{\sigma_{min}^2+\lambda}\\\vdots\\0\end{bmatrix}\right\|_2^2 + \left\|\begin{bmatrix}\frac{\sigma_1+\sigma_1^2\lambda+\lambda^2}{\sigma_1^2+\lambda} & & & \\ & \frac{\sigma_2+\sigma_2^2\lambda+\lambda^2}{\sigma_2^2+\lambda} & & \\ & & \ddots & \\ & & & \frac{\sigma_{min}+\sigma_{min}^2\lambda+\lambda^2}{\sigma_{min}^2+\lambda} \\ & & 0 & \end{bmatrix}\cdot V^Tx\right\|_2^2$$

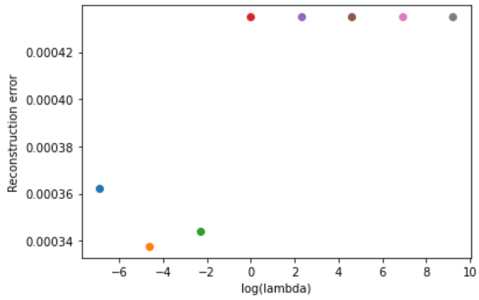
when λ increases, the reconstruction error decreases.

In [1]:
```
import numpy as np
from tqdm import tqdm
import matplotlib.pyplot as plt
from sklearn.metrics import mean_squared_error


def soft_thresholding(z, lambd):
    """
    Soft-thresholding operator.
    :param z: The input.
    :param lambd: The hyperparameter controls the soft-thresholding operator.
    :return: The result of soft-thresholding.
    """
    return np.multiply(np.sign(z), np.maximum(np.abs(z) - lambd, 0))


def g_function(A, x, y, lambd):
    """
    The objective function that to be minimized (least-squares regression problem with l1-norm regularization)
    :param A: The measurement matrix.
    :param x: The sparse vector at iteration k.
    :param y: The measurement.
    :param lambd: The hyperparameter controls the soft-thresholding operator.
    :return: The result of the objective function.
    """
    return 0.5 * np.linalg.norm(A @ x - y, ord=2) ** 2 + lambd * np.linalg.norm(x, ord=1)


def ISTA(A, y, lambd, tol):
    """
    Iterated Soft Thresholding Algorithm (ISTA).
    :param A: The measurement matrix.
    :param y: The measurement.
    :param lambd: The hyperparameter controls the soft-thresholding operator.
    :param tol: The stopping criterion.
    :return: The solution of the sparse vector, a list of iteration numbers, and a list of g_function results.
    """
    dim_x = np.shape(A)[1]
    L = np.linalg.norm(A) ** 2  # Lipschitz const
    # Initialization
    step_size = 1 / L
    # x_k = np.random.normal(0, 1, size=(dim_x, 1))
    x_k = np.zeros((dim_x, 1))
    g_k = g_function(A, x_k, y, lambd)
    ite_k = 1
    diff = 1000
    # Loop
    while abs(diff) > tol:
        gradient_k = A.T @ A @ x_k - A.T @ y
        z_k = x_k - step_size * gradient_k
        x_k_plus_one = soft_thresholding(z_k, lambd / L)
        g_k_plus_one = g_function(A, x_k_plus_one, y, lambd)
        diff = g_k_plus_one - g_k
        ite_k += 1
        x_k = x_k_plus_one
        g_k = g_k_plus_one
    x_final = x_k_plus_one
    return x_final



# Parameter
space_lambd = [1e-3, 1e-2, 1e-1, 1, 1e1, 1e2, 1e3, 1e4]
tol = 1e-4

# A: Gaussian mean 0, variance 1/500
A = np.random.normal(0, np.sqrt(1/500), size=(500, 2000))

# x: 50-sparse with mean 0, variance 1/50
x = np.random.normal(0, np.sqrt(1/50), size=(2000, 1))
zero_indices = np.random.choice(np.arange(2000), replace=False, size=int(1950))
x[zero_indices] = 0

# e: mean 0, variance 0.5/500
e = np.random.normal(0, np.sqrt(0.5/500), size=(500, 1))

# y
y = A @ x + e

# Grid search for different sparsity values
space_sparsity = [50]
search_shape = [len(space_sparsity), len(space_lambd)]
mse = np.zeros(search_shape)
i = 0
j = 0
for sparsity in space_sparsity:
    for lambd in tqdm(space_lambd):
        x_final = ISTA(A, y, lambd, tol)
        mse[i, j] = mean_squared_error(x_final, x)
        j += 1
    i += 1
```

```
100%|██████████████████| 8/8 [00:44<00:00,  5.50s/it]
```

In [2]:
```
lambda_new = np.asarray(space_lambd)
lambda_new = np.expand_dims(lambda_new, 0)
```

In [3]:
```
plt.xlabel('log(lambda)')
plt.ylabel('Reconstruction error')
plt.plot(np.log(lambda_new), mse, marker="o")
plt.show()
```



In [ ]:
```
```