In [30]:
```
import numpy as np
import matplotlib.pyplot as plt
```

Homework

Question 1

The gradient is

$$\nabla f(x, y) = \begin{bmatrix} 2x + y + 1 \\ 4y + x - 1 \end{bmatrix}$$

Then we have

$$H_f(x, y) = \begin{bmatrix} 2 & 1 \\ 1 & 4 \end{bmatrix}$$

We now calculate the eigenvalues of the Hessian matrix.

$$\det(H_f - \lambda I) = \det(\begin{bmatrix} 2-\lambda & 1 \\ 1 & 4-\lambda \end{bmatrix}) = (2-\lambda)(4-\lambda) - 1 = \lambda^2 - 6\lambda + 7 = 0$$

Solving this quadratic equation:

$$\lambda = \frac{6 \pm \sqrt{36-28}}{2} = \frac{6 \pm \sqrt{8}}{2} = 3 \pm \sqrt{2}$$

Since both eigenvalues $3 + \sqrt{2}$ and $3 - \sqrt{2}$ are positive, the Hessian matrix is positive definite, meaning that $f(x, y)$ is a convex function, so it satisfies the conveergence theorem of gradient descent.

The optimal step size α for gradient descent is given by:

$$\alpha = \frac{2}{L + \mu}$$

where L and μ are the largest and smallest eigenvalues, respectively. Thus:

$$L = 3 + \sqrt{2}, \quad \mu = 3 - \sqrt{2}$$

Substituting these values:

$$\alpha = \frac{2}{(3+\sqrt{2}) + (3-\sqrt{2})} = \frac{2}{6} = \frac{1}{3}$$

Thus, the optimal step size for the fastest convergence is $\alpha = \frac{1}{3}$.

0th Iteration

$$x_0 = (3, 3)$$

1st Iteration

$$\nabla f(x_0, y_0) = \begin{bmatrix} 10 \\ 14 \end{bmatrix}$$

$$x_1 = x_0 - \alpha \nabla f(x_0) = \begin{bmatrix} 3 \\ 3 \end{bmatrix} - \frac{1}{3} \begin{bmatrix} 10 \\ 14 \end{bmatrix} = \begin{bmatrix} -\frac{1}{3} \\ -\frac{5}{3} \end{bmatrix}$$

2nd Iteration

$$\nabla f(x_1, y_1) = \begin{bmatrix} -\frac{4}{3} \\ -8 \end{bmatrix}$$

$$x_2 = x_1 - \frac{1}{3} \begin{bmatrix} -\frac{4}{3} \\ -8 \end{bmatrix} = \begin{bmatrix} \frac{1}{9} \\ 1 \end{bmatrix}$$

3rd Iteration

$$\nabla f(x_2, y_2) = \begin{bmatrix} \frac{20}{9} \\ \frac{28}{9} \end{bmatrix}$$

$$x_3 = x_2 - \frac{1}{3} \begin{bmatrix} \frac{20}{9} \\ \frac{28}{9} \end{bmatrix} = \begin{bmatrix} -\frac{17}{27} \\ -\frac{1}{27} \end{bmatrix}$$

Question 2

$$\nabla f(x, y) = \begin{bmatrix} 8x - 4y \\ 4y - 4x \end{bmatrix}, \quad H_f(x, y) = \begin{bmatrix} 8 & -4 \\ -4 & 4 \end{bmatrix}, \text{ positive definite}$$

0th Iteration

$$x_0 = \begin{bmatrix} 1 \\ 1 \end{bmatrix}, \quad f(x_0) = 2$$

1st Iteration

At x_0, the steepest descent direction

$$v_1 = -\nabla f(x_0) = \begin{bmatrix} -4 \\ 0 \end{bmatrix}$$

and the optimal step

$$\alpha_1 = \frac{v_1^T v_1}{v_1^T H_f v_1} = \frac{16}{\begin{bmatrix} -4 & 0 \end{bmatrix} \begin{bmatrix} 8 & -4 \\ -4 & 4 \end{bmatrix} \begin{bmatrix} -4 \\ 0 \end{bmatrix}} = \frac{1}{8}$$

so

$$x_1 = x_0 + \frac{1}{8}\begin{bmatrix}-4\\0\end{bmatrix} = \begin{bmatrix}\frac{1}{2}\\1\end{bmatrix}, \quad f(x_1) = 1$$

2nd Iteration

Having the first iteration, we use the formula

$$v_2 = v_1 - \alpha_1 H_f v_1 = \begin{bmatrix}-4\\0\end{bmatrix} - \frac{1}{8}\begin{bmatrix}8 & -4\\-4 & 4\end{bmatrix}\begin{bmatrix}-4\\0\end{bmatrix} = \begin{bmatrix}-4\\0\end{bmatrix} - \begin{bmatrix}-4\\2\end{bmatrix} = \begin{bmatrix}0\\-2\end{bmatrix}$$

Then

$$\alpha_2 = \frac{v_2^T v_2}{v_2^T H_f v_2} = \frac{1}{4}$$

so

$$x_2 = \begin{bmatrix}\frac{1}{2}\\1\end{bmatrix} + \frac{1}{4}\begin{bmatrix}0\\-2\end{bmatrix} = \begin{bmatrix}\frac{1}{2}\\\frac{1}{2}\end{bmatrix}, \quad f(x_2) = \frac{1}{2}$$

3rd Iteration

$$v_3 = v_2 - \alpha_2 H_f v_2 = \begin{bmatrix}-2\\0\end{bmatrix}$$

Then

$$\alpha_3 = \frac{v_2^T v_2}{v_2^T H_f v_2} = \frac{1}{8}$$

so

$$x_3 = \begin{bmatrix}\frac{1}{2}\\\frac{1}{2}\end{bmatrix} + \frac{1}{8}\begin{bmatrix}-2\\0\end{bmatrix} = \begin{bmatrix}\frac{1}{4}\\\frac{1}{2}\end{bmatrix}, \quad f(x_3) = \frac{1}{4}$$

Question 3

a

Given the iterative method:

$$X^{(n+1)} = X^{(n)} - a(AX^{(n)} - b) = IX^{(n)} - aAX^{(n)} - ab = [I - aA]X^{(n)} + ab$$

Thus

$$B = I - aA, \quad c = ab$$

b

Since A is symmetric positive-definite, its eigenvalues λ_i are positive.

Since $B = I - aA$, The eigenvalues of B are

$$\mu_i = 1 - a\lambda_i$$

Given $a\lambda_d < 2$, we have

$$1 - a\lambda_d > -1$$

Similarly, since $\lambda_1 > 0$

$$1 - a\lambda_1 < 1$$

Therefore, all eigenvalues μ_i of B satisfy:

$$\mu_i \in (-1, 1)$$

Since the spectral radius $\rho(B) < 1$, $BX^{(n)}$ will shrink compare to $X^{(n)}$ on all dimensions, so the iteration converges.

To find the limit $X^{(\infty)}$, we note that at convergence

$$X^{(\infty)} = BX^{(\infty)} + c$$

i.e.

$$(I - B)X^{(\infty)} = c \implies aAX^{(\infty)} = ab \implies AX^{(\infty)} = b$$

Thus, the limit is

$$X^{(\infty)} = A^{-1}b$$

c

The convergence rate depends on the spectral radius $\rho(B) = \max_i |\mu_i|$. To minimize $\rho(B)$, we set

$$|1 - a\lambda_{\min}| = |1 - a\lambda_{\max}|$$
$$1 - a\lambda_{\min} = a\lambda_{\max} - 1 \implies a = \frac{2}{\lambda_{\min} + \lambda_{\max}}$$

Therefore

$$a_{\text{opt}} = \frac{2}{\lambda_{\min} + \lambda_{\max}}$$

d

In [31]:
```
def Laplacian(n):
    M = np.zeros((n, n))
    for i in range(n):
        M[i, i] = 2
        if i > 0:
            M[i, i - 1] = -1
        if i < n - 1:
            M[i, i + 1] = -1
    return M
```

```
d = 10
n = 10000

A = Laplacian(d)
X = np.zeros(d)
b = np.ones(d)

a = 1e-2

for _ in range(n):
    X = X - a * (A @ X - b)

print(f"The 7th coordinate is {X[6]:.4f}")
```

```
The 7th coordinate is 13.9957
```

Verified ✔

Question 4

a

Suppose the corresponding eigenvectors are $v_1, v_2, \ldots, v_d$, then express x^0 as a linear combination of the eigenvectors

$$x^0 = c_1 v_1 + c_2 v_2 + \cdots + c_d v_d$$

Therefore

$$x^n = A^n x^0 = c_1 \lambda_1^n v_1 + c_2 \lambda_2^n v_2 + \cdots + c_d \lambda_d^n v_d$$

$$\langle Ax^n, x^n \rangle = \sum_{i=1}^{d} c_i \lambda_i^n v_i c_i \lambda_i^{n+1} v_i = \sum_{i=1}^{d} c_i^2 \lambda_i^{2n+3}$$

$$\|x^n\|^2 = \sum_{i=1}^{d} c_i^2 \lambda_i^{2n+2}$$

Therefore

$$\frac{\langle Ax^n, x^n \rangle}{\|x^n\|^2} = \frac{\sum_{i=1}^d c_i^2 \lambda_i^{2n+3}}{\sum_{i=1}^d c_i^2 \lambda_i^{2n+2}} = \frac{\sum_{i=1}^d c_i^2 \lambda_i^{2n+2} \lambda_i}{\sum_{i=1}^d c_i^2 \lambda_i^{2n+2}}$$

As $n \to \infty$, since $\lambda_d > \lambda_{d-1} > \cdots > \lambda_1$, the term with λ_d dominates, because the gap between term increases due to the exponential term λ_i^{2n+2}, i.e.

$$\lim_{n\to\infty} \frac{\langle Ax^n, x^n \rangle}{\|x^n\|^2} = \lambda_d$$

b

$$P_B(y) = \frac{y}{\|y\|_2}$$

c

Objective function

$$f(x) = \langle Ax, x \rangle = x^\top Ax$$

The gradient of it is

$$\nabla f(x) = 2Ax$$

and

$$H_f(x) = 2A$$

Algorithm steps:

1. **Initialization:** Choose $x^0 \in B$.
2. **Iteration:** For $k = 0, 1, 2, \ldots$:

$$x^{k+1} = P_B\left(x^k + \alpha_k \nabla f(x^k)\right) = P_B\left(x^k + 2\alpha_k Ax^k\right)$$

   where α_k is the step size.

Convergence analysis: Since $f(x)$ may be non-convex, the gradient projection method may not guarantee convergence to the global maximum.

d

1. **Initialization:** Choose a non-zero vector $x^0 \in \mathbb{R}^d$.
2. **Iteration:** For $n = 0, 1, 2, \ldots$:
   - Compute $y^{n+1} = Ax^n$.
   - Normalize $x^{n+1} = \dfrac{y^{n+1}}{\|y^{n+1}\|}$.

Convergence discussion: As shown in (a), as long as the initial vector has a non-zero component in the direction of the eigenvector corresponding to the largest eigenvalue, x^n will converge to that eigenvector, and $\langle Ax^n, x^n \rangle$ will converge to the largest eigenvalue λ_d.

Question 5

a

In [32]:

```
def energy(u):
    N = len(u) // 2
    y = u[N:]
    E = np.sum(y) / (N + 1)
    grad_E = np.zeros(2 * N)
    grad_E[N:] = 1 / (N + 1)
    return E, grad_E
```

b & c

First, compute $\frac{\partial P}{\partial x_i}$:

$$\frac{\partial P}{\partial x_i} = \sum_{k=1}^{N+1} \phi_k(u) \frac{\partial \phi_k(u)}{\partial x_i}$$

Since $\phi_k(u) = l_k(u) - h$, we have:

$$\frac{\partial \phi_k(u)}{\partial x_i} = \frac{\partial l_k(u)}{\partial x_i}$$

The length $l_k(u)$ depends on x_i only when $k = i$ or $k = i + 1$, Thus

- For $k = i$:

$$\frac{\partial l_i(u)}{\partial x_i} = \frac{x_i - x_{i-1}}{l_i}$$

- For $k = i + 1$:

$$\frac{\partial l_{i+1}(u)}{\partial x_i} = -\frac{x_{i+1} - x_i}{l_{i+1}}$$

Therefore

$$\frac{\partial P}{\partial x_i} = \phi_i(u) \frac{x_i - x_{i-1}}{l_i} - \phi_{i+1}(u) \frac{x_{i+1} - x_i}{l_{i+1}}$$

Using $\phi_i(u) = l_i - h$, we have

$$\frac{\partial P}{\partial x_i} = \left(1 - \frac{h}{l_i}\right)(x_i - x_{i-1}) + \left(1 - \frac{h}{l_{i+1}}\right)(x_i - x_{i+1})$$

Similarly for $\frac{\partial P}{\partial y_i}$

$$\frac{\partial P}{\partial y_i} = \left(1 - \frac{h}{l_i}\right)(y_i - y_{i-1}) + \left(1 - \frac{h}{l_{i+1}}\right)(y_i - y_{i+1})$$

In [33]:
```
def penalty(u):
    N = len(u) // 2
    x_i = u[:N]
    y_i = u[N:]

    # Constraint
    x = np.hstack(([0], x_i, [1]))
    y = np.hstack(([1], y_i, [1.5]))
    dx = x[1:] - x[:-1]
    dy = y[1:] - y[:-1]
    l = np.sqrt(dx**2 + dy**2)
    h = 2 / (N + 1)
    phi = l - h
    P = 0.5 * np.sum(phi**2)
```

```python
    # Gradient
    coeff = 1 - h / l
    grad_P_x = coeff[:-1] * dx[:-1] - coeff[1:] * dx[1:]
    grad_P_y = coeff[:-1] * dy[:-1] - coeff[1:] * dy[1:]
    grad_P = np.concatenate([grad_P_x, grad_P_y])

    return P, grad_P


def gradient_descent(N, epsilon, tol):
    np.random.seed(42)  # "Ultimate Question of Life, the Universe, and Everythi
    a = epsilon / 3
    u = np.random.rand(2 * N)
    for k in range(int(1e8)):
        E, grad_E = energy(u)
        P_val, grad_P = penalty(u)
        grad = grad_E + (1 / epsilon) * grad_P
        grad_norm = np.linalg.norm(grad, ord=np.inf)
        if grad_norm <= tol:
            print(f"Converged after {k} iterations with ε = {epsilon}")
            break
        u = u - a * grad
    x_i = u[:N]
    y_i = u[N:]
    x = np.hstack(([0], x_i, [1]))
    y = np.hstack(([1], y_i, [1.5]))
    return x, y


N = 20
acc = 0.001

x1, y1 = gradient_descent(N, 0.1, acc)
x2, y2 = gradient_descent(N, 0.01, acc)
```

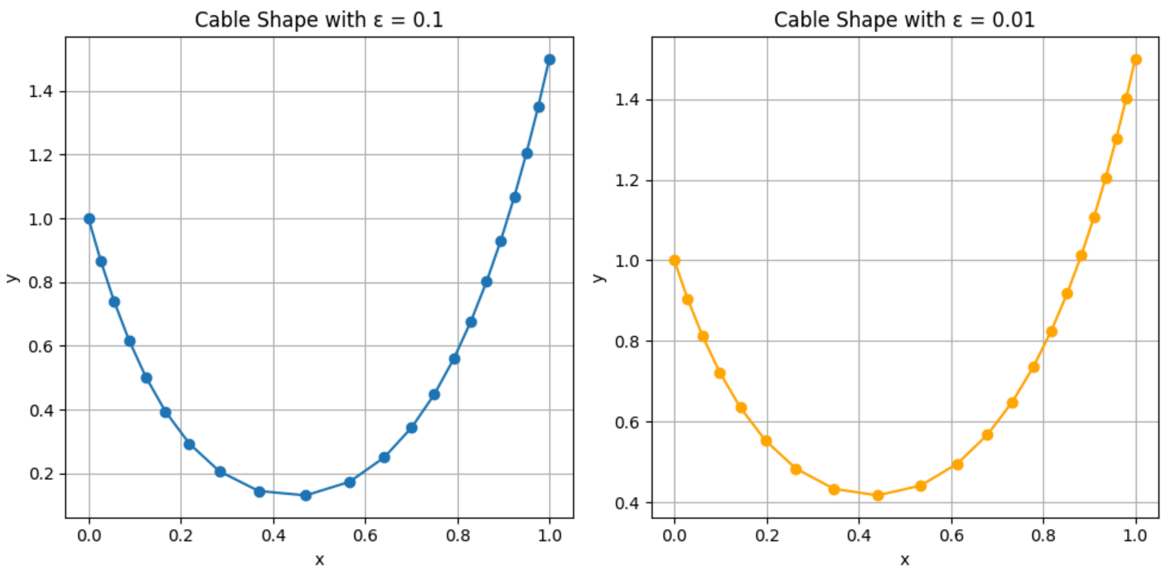
```
Converged after 816 iterations with ε = 0.1
Converged after 5584 iterations with ε = 0.01
Converged after 5584 iterations with ε = 0.01
```

Plot the result

In [34]:
```python
plt.figure(figsize=(10, 5))
plt.subplot(1, 2, 1)
plt.plot(x1, y1, marker="o")
plt.title("Cable Shape with ε = 0.1")
plt.xlabel("x")
plt.ylabel("y")
plt.grid(True)

plt.subplot(1, 2, 2)
plt.plot(x2, y2, marker="o", color="orange")
plt.title("Cable Shape with ε = 0.01")
plt.xlabel("x")
plt.ylabel("y")
plt.grid(True)

plt.tight_layout()
plt.show()
```


Cable Shape with ε = 0.1
Cable Shape with ε = 0.01
y
x
0.0
0.2
0.4
0.6
0.8
1.0
1.2
1.4