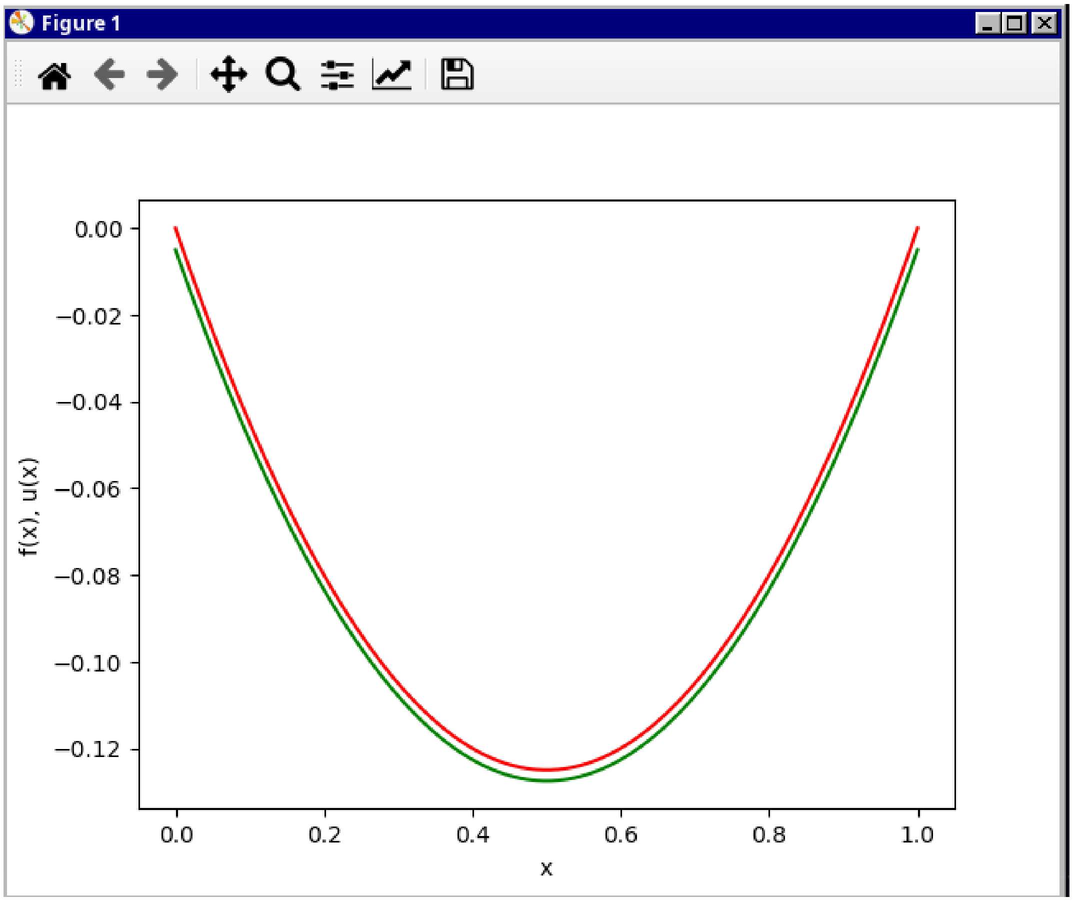
Following the lectures, one can find the Matrix A bei integrating the „Hütchen“-function for a point by using the derivative x_i – x_i-1 / h with h as stepsize.

Integrating from 0 to 1 will result in 2f/h.

```python
if __name__== "__main__":
    n = 100
    stepsize = 1/(n)
    f = 1 # a constant load f = 1
    A = -(2*np.eye(n,n) - np.eye(n, n, -1) - np.eye(n, n, 1))*f/stepsize; #does every fem matrix look like this?
    x = np.full((n),stepsize)*f;
    u = linalg.solve(A,x) #solve the system of equations
    print(u)
    xRange = np.linspace(0,1,n);
    f = (np.power(xRange,2)-xRange)*0.5;
```

Plot the solutions

```python
    plt.plot(xRange,f, "r");
    plt.plot(xRange,u, "g");
    plt.xlabel("x");
    plt.ylabel("f(x), u(x)");
    plt.show()
```



How could different boundary conditions be implemented?

I think based on different conditions the function u(x) would converge different than it does now.