```
def M0(r, C, D, m):
    return D * (m*w2(r, C, D) + (1/r) * w1(r, C, D))


r_values = np.linspace(r1, r2, 100)
M0_values = [M0(r, solution, D, m) for r in r_values]


plt.plot(r_values, M0_values, color='blue', label='Mr(r)')
plt.fill_between(r_values, M0_values, color='gray',alpha=0.5, hatch='||',edgecolor='black')
plt.xlabel('r')
plt.ylabel('M0(r)')
plt.title('Изгибающий момент в окружном направлении')
plt.grid(True)
plt.legend()
plt.show()


def σ_r(Mr_values, h):
    return 6*np.array(Mr_values)/h**2


def σ_0(M0_values, h):
    return 6*np.array(M0_values)/h**2


def σ_eqv(σ_r,σ_0):
    return np.sqrt(σ_r**2+σ_0**2-σ_r*σ_0)


plt.plot(r_values, σ_eqv(σ_r(Mr_values, h), σ_0(M0_values, h)))
plt.fill_between(r_values, σ_eqv(σ_r(Mr_values, h), σ_0(M0_values, h)), color='gray', alpha=0.5,
hatch='||', edgecolor='black')
plt.xlabel('Радиус r, м')
plt.ylabel('$σ_(экв)$')
plt.title('Эквивалентное напряжение')
plt.show()
```