```python
import numpy as np
import Tools as tools
import scipy.optimize as optimize

def solve(par): # Solves the model
    # Preallocating
    Vstar = np.zeros([par['T'],par['gridsize_w']])
    Cstar = np.zeros([par['T'],par['gridsize_w']])
    Astar = np.zeros([par['T'],par['gridsize_w']])


    # 1) Loop over time
    for t in range(par['T']-1,-1,-1):
        print(t)
        if t == par['T'] - 1:
            Cstar[t,:] = np.array(par['𝒢_w']).T
            Vstar[t,:] = Cstar[t,:]**(1 - par['ρ'])/(1 - par['ρ'])
        else:
            # 2) Loop over cash-on-hand
            for iw,w in enumerate(par['𝒢_w']):
                # Solving the contemporaneous decision problem
                sol = optimize.minimize_scalar(objective,bounds=[0,w+1e-04],args=(par,t,w,Vstar[t+1,:]),method='bounded',options={'xatol': 1e-4, 'maxiter': 10000})

                # Filling results
                Vstar[t,iw] = - sol.fun
                Cstar[t,iw] =   sol.x
                Astar[t,iw] =   w - Cstar[t,iw]
    return Vstar,Cstar,Astar



def objective(c_guess,par,t,w,Vstar_plus):
    # Cash-on-hand tomorrow
    w_plus = par['l'][t+1]*par['Y'] + par['R']*(w - c_guess)
    # print(w_plus.shape)

    # Interpolating over value function tomorrow
    V_temp = tools.interp_linear_1d(par['𝒢_w'],Vstar_plus,w_plus)

    # Computing the expected value function
    EV_next = par['ω'] @ V_temp

    # Computing the current value function
    V = c_guess**(1 - par['ρ'])/(1 - par['ρ']) + par['β']*EV_next
    return - V



def simulation(par,Cstar):# Simulation of the model
    # Setting the seed
    np.random.seed(2021)

    # Preallocate
    simW = np.zeros([par['T'],par['N']])            # storage for the simulation of cash-on-hand
    simC = np.zeros([par['T'],par['N']])            # storage for the simulation of consumption
    simY = np.zeros([par['T'],par['N']])            # storage for the simulation of income
    simA = np.zeros([par['T']+1,par['N']])          # storage for the simulation of savings
```