```
# Load the required packages
using ODE
using JLD
using ForwardDiff

set_bigfloat_precision(113)

# Define the system for the solver
function roberAD(x)
    k1 = parse(BigFloat,"0.04");
    k2 = parse(BigFloat,"1e4");
    k3 = parse(BigFloat,"3e7");

    return [-k1*x[1]+k2*x[2]*x[3],
    k1*x[1]-k2*x[2]*x[3]-k3*(x[2])^2,
    k3*(x[2])^2]
end

function rober(t,x)
    return roberAD(x)
end

function getJacobian(t,x)
    return ForwardDiff.jacobian(roberAD,x);
end

# Set up the initial conditions
tSpan = [zero(BigFloat);parse(BigFloat,"10.0").^collect(0:11)];
x0 = [one(BigFloat),zero(BigFloat),zero(BigFloat)];

# Set the tolerances
# ATol = RTol*1e-6
RTol = parse(BigFloat,"1e-20");
ATol = parse(BigFloat,"1e-26");

# Solve and get the solution at T = tEnd
(t,x_tmp) = ode23s(rober,x0,tSpan;
reltol=RTol,abstol=ATol,points=:specified,
jacobian = getJacobian,minstep=parse(BigFloat,"1e-8"));

x_ref = Array{BigFloat}(11,3);

for i=1:11
    x_ref[i,:] = x_tmp[i+1,1][:];
end

# Save the solution to a file
save("refSolRober.jld","x_ref",x_ref);
```