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

set_bigfloat_precision(113)

# Define the system for the solver
function vdpolAD(x)
    return [x[2],((1-x[1]^2)*x[2]-x[1])*1e6]
end

function vdpol(t,x)
    return vdpolAD(x)
end

function getJacobian(t,x)
    J = Matrix{BigFloat}(2,2);
    J[:,:] = ForwardDiff.jacobian(vdpolAD,x);
    return J
end

# Set up the initial conditions
tSpan = collect(zero(BigFloat):parse(BigFloat,"11.0"));
x0 = [2*one(BigFloat),zero(BigFloat)];

# Set the tolerances
Tol = parse(BigFloat,"1e-20");

# Solve and get the solution at T = tEnd
(t,x_tmp) = ode23s(vdpol,x0,tSpan;
reltol=Tol,abstol=Tol,points=:specified,
jacobian = getJacobian);

x_ref = Array{BigFloat}(11);

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

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