```julia
# There is a bug with the Dense Output for SEULEX solver
# This bug is only seen when we have explicitly defined Jacobian

# Load all the required packages
using ODEInterface
using ForwardDiff
@ODEInterface.import_huge
loadODESolvers();

# Define the right-hand function for automatic differentiation
function vdpolAD(x)
    return [x[2],((1-x[1]^2)*x[2]-x[1])*1e6]
end

# Define the system for the solver
function vdpol(t,x,dx)
    dx[:] = vdpolAD(x);
    return nothing
end

# Define the Jacobian function using AD
function getJacobian(t,x,J)
    J[:,:] = ForwardDiff.jacobian(vdpolAD,x);
    return nothing
end

# Initial conditions
t0 = 0.0; T = 11.0; x0 = [2.0,0.0];

# Store the solutions at various time steps
global solCollection = zeros(10,2);

# Define the output function to extract function values at output time steps
function outputfcn(reason,told,t,x,eval_sol_fcn,extra_data)
    if reason == OUTPUTFCN_CALL_STEP
        T = [1.0:10.0;];
        for i=1:10
            if told < T[i] <= t
                solCollection[i,:] = eval_sol_fcn(T[i]);
                return OUTPUTFCN_RET_CONTINUE
            end
        end
    end
    return OUTPUTFCN_RET_CONTINUE
end

# Get "reference solution"
Tol = 1e-14;
# for Tol < 1e-14 we get the error "TOLERANCES ARE TOO SMALL"
opt = OptionsODE(OPT_EPS=>1.11e-16,OPT_RTOL=>Tol, OPT_ATOL=>Tol,
OPT_RHS_CALLMODE => RHS_CALL_INSITU,
OPT_JACOBIMATRIX => getJacobian,
OPT_OUTPUTFCN => outputfcn,
OPT_OUTPUTMODE => OUTPUTFCN_DENSE);
```

```
(t,x,retcode,stats) = seulex(vdpol,t0, T, x0, opt);
x_dense = [solCollection;x'];

 # Get non dense output at all the output times
 opt = OptionsODE(OPT_EPS=>1.11e-16,OPT_RTOL=>Tol, OPT_ATOL=>Tol,
 OPT_RHS_CALLMODE => RHS_CALL_INSITU,
 OPT_JACOBIMATRIX => getJacobian);

 T = [1.0:11.0;];

 x_nonDense = Array{Array{Float64,1}}(11);
 for i=1:11
     (t,x_nonDense[i],retcode,stats) = seulex(vdpol,t0, T[i], x0, opt);
 end

 # Problem occurs at T = 4.0
 # The second component is wrong
 [x_nonDense[4][:] x_dense[4,:]']
```