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

# Number of subdivisions of the rope
global n = 40;

# Define the system of ODEs
function rope(t,x,dx)
    n2 = n*n; # n^2
    n3by4 = convert(Int64,3*n/4); # 3*n/4

    # Force in x-direction
    Fx = 0.4;
    # Force in y-direction
    Fy = cosh(4*t-2.5)^(-4);

    # Compute required matrices
    c = -cos(x[1:n-1]-x[2:n]);
    cDiag = [1;2*ones(n-2);3];
    C = spdiagm((c,cDiag,c),(-1,0,1));

    d = -sin(x[1:n-1]-x[2:n]);
    D = spdiagm((-d,d),(-1,1));

    # Compute the inhomogeneous term
    v = -(n2+n/2-n*[1:n;]).*sin(x[1:n])-n2*sin(x[1:n])*Fx;
    v[1:n3by4] = v[1:n3by4] + n2*cos(x[1:n3by4])*Fy;

    w = D*v+x[n+1:2*n].^2;
    u = C\w;

    # Write down the system
    dx[1:n] = x[n+1:2*n];
    dx[n+1:2*n] = C*v + D*u;

    return nothing
end

# Initial Conditions
t0 = 0.0; T = 3.723; x0=zeros(2*n);

# Compute the "reference solution"
opt = OptionsODE(OPT_EPS => 1.11e-16,OPT_RHS_CALLMODE => RHS_CALL_INSITU,
OPT_RTOL => 1e-16,OPT_ATOL=>1e-16);
(t,x_ref,retcode,stats) = dop853(rope,t0, T, x0, opt);

if retcode != 1
    println("Reference solution failed")
else
    # Initialization for the loop
    # f_e = function evaluations
    f_e = zeros(Int32,89,3);
    # err = error for last step using infinity norm
    err = zeros(Float64,89,3);

    # solverNames = names of the solvers used for the plot
    solverNames = ["DOPRI5","DOP853","ODEX"];

    # Compute all the solutions
    for i=0:88

        # Set up the tolerance
        Tol = 10^(-3-i/8);

        # Set up solver options
        opt = OptionsODE(OPT_EPS => 1.11e-16,OPT_RHS_CALLMODE => RHS_CALL_INSITU,
        OPT_RTOL => Tol,OPT_ATOL => Tol);

        # Solve using DOPRI5
        (t,x,retcode,stats) = dopri5(rope,t0, T, x0, opt);
        # Check if solver was successful
        if retcode != 1
            printFlag = false;
            break;
        end
        f_e[i+1,1] = stats.vals[13];
        err[i+1,1] = norm(x_accurate[1:n] - x[1:n],Inf);

        # Solve using DOP853
        (t,x,retcode,stats) = dop853(rope,t0, T, x0, opt);
        # Check if solver was successful
        if retcode != 1
            printFlag = false;
            break;
        end
        f_e[i+1,2] = stats.vals[13];
        err[i+1,2] = norm(x_accurate[1:n] - x[1:n],Inf);

        # Solve using ODEX
        (t,x,retcode,stats) = odex(rope,t0, T, x0, opt);
        # Check if solver was successful
        if retcode != 1
            printFlag = false;
            break;
        end
        f_e[i+1,3] = stats.vals[13];
        err[i+1,3] = norm(x_accurate[1:n] - x[1:n],Inf);
    end

    # Save the plot in PNG format
    if printFlag
        savePlotPNG("RopeConvTest",f_e,err,solverNames);
    else
        println("Cannot generate plot due to solver failure")
    end
end
```