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

# Define the system of ODEs
function threebody(t,x,dx)
    mu = 0.012277471; ms = 1 - mu;
    r1 = vecnorm(x[1:2]-[-mu,0]);
    r2 = vecnorm(x[1:2]-[ ms,0]);

    dx[1] = x[3];
    dx[2] = x[4];
    dx[3] = x[1] + 2*x[4] - ms*(x[1]+mu)/r1^3 - mu*(x[1]-ms)/r2^3;
    dx[4] = x[2] - 2*x[3] - ms*     x[2]/r1^3 - mu*     x[2]/r2^3;

    return nothing
end

# Flag to check if all solvers were successful
printFlag = true;

# Initial conditions
t0 = 0.0; T = 17.06521656015796; x0=[0.994, 0.0, 0.0, -2.001585106379082];

# Get "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(threebody,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(threebody,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_ref[1:2] - x[1:2],Inf);

        (t,x,retcode,stats) = dop853(threebody,t0, T, x0, opt);
        if retcode != 1
            printFlag = false;
            break;
        end
        f_e[i+1,2] = stats.vals[13];
        err[i+1,2] = norm(x_ref[1:2] - x[1:2],Inf);

        (t,x,retcode,stats) = odex(threebody,t0, T, x0, opt);
        if retcode != 1
            printFlag = false;
            break;
        end
        f_e[i+1,3] = stats.vals[13];
        err[i+1,3] = norm(x_ref[1:2] - x[1:2],Inf);
    end

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