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

# Set precision to quadruple precision (QP)
# QP has 113 bits of precision
set_bigfloat_precision(113);

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

# Define the right-hand function
function rope(t,x)

    n2 = n*n; # n^2
    n3by4 = convert(Int64,3*n/4); # 3*n/4

    # Force in x-direction
    Fx = parse(BigFloat,"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 = [one(BigFloat);
        2*ones(BigFloat,n-2);
        3*one(BigFloat)];
    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
    return [x[n+1:2*n],C*v + D*u];
end

# Set up the initial conditions
t0 = zero(BigFloat);
T = parse(BigFloat,"3.723");
x0 = zeros(BigFloat,2*n);

# Set the tolerance
Tol = parse(BigFloat,"1e-33");

# Solve and get the solution at T = tEnd
(t_rope,x_tmp_rope) = ode78(rope,x0,[t0;T];
```

```
reltol=Tol,abstol=Tol,points=:specified);

x_ref = Array{BigFloat}(n);

x_ref[:] = x_tmp_rope[2,1][1:n];

save("refSolRope.jld","x_ref",x_ref);
```