BCEE 231 – Homework Set #4

P4.1 [30 marks]

Given the following two matrices (as sample data)

$$\mathbf{A} = \begin{bmatrix} 3 & 1 & -1 \\ 0 & 2 & 4 \\ -2 & 1 & 3 \end{bmatrix} \qquad \mathbf{B} = \begin{bmatrix} 2 & 0 & 1 \\ -1 & 2 & 2 \\ 2 & -1 & 2 \end{bmatrix}$$

we can form the matrix **C** = **A** – β **B** where β is a scalar value.

Except for a few special values of β, the solution vector **X** of **C X** = 0 is the trivial null vector. In many engineering problems[1], we want to find these special values of β that makes the solution vector **X** non-trivial.

There are three such special values of β (the so-called the eigenvalues) for the given sample matrices. They are found as the values that make the determinant of **C** zero as shown in following program segment:

```
main()
{
            defmat(A[N = 3,N], 3, 1, -1, 0, 2, 4, -2, 1, 3);
            defmat(B[N,N], 2, 0, 1, -1, 2, 2, 2, -1, 2);

            // Find b that makes the determinant of (C = A-b*B) equal 0

            // Explore range of roots b's
            clearplot(); plot(x, -0.5, 5, Det(x));

            zero(Bet[N]);  // Storage for the roots (eigenvalues b's)

            tol# = 1.e-8; // Reduce tolerance for more accurate root
            Bet[1] = root1(b, -0.5, 0.5, Det(b));
            Bet[2] = root1(b, 0.5, 2, Det(b));
            Bet[3] = root1(b, 2, 5, Det(b));
            print(Bet,         ^^,          "         Residuals          =
            ",Det(Bet[1]),Det(Bet[2]),Det(Bet[3]));
}


Det(float b)
{ // Return the determinant of C = A-b*B
            return det(!C = A-b*B);
}
```

[1] Such as the problems of natural vibration and stability of structures.

Your task is to extend the preceding program to find the non-trivial solution vector **X** (of the homogeneous equation **C X** = 0) for each of the three eigenvalues β and to verify that the residual vector **C X** is a null-vector.

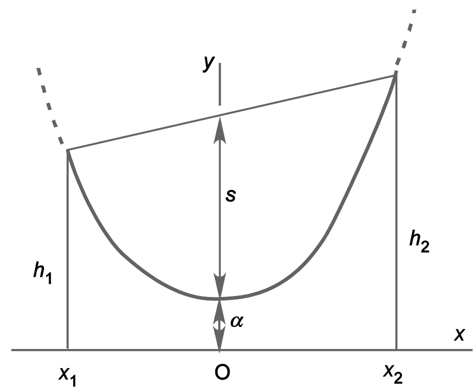
Your report should include the program, output, comments on the correctness of the program.

P4.2 [40 maks]

A uniform cable hanging under its own weight takes the shape of the catenary[2], whose equation is simplest when the *y*-axis is at the axis of symmetry, and the *x*-axis is placed at the distance α below the curve's vertex:

$$y = \alpha \cosh\left(\frac{x}{\alpha}\right)$$

i.e. $y(0) = \alpha$.



It can be shown that the constant α is:

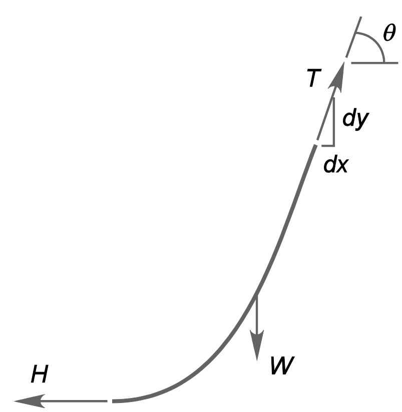
$$\alpha = \frac{H}{w} \equiv \left(\frac{\text{Horizontal component of cable tension}}{\text{Cable weight per unit length}}\right)$$

For any given α, the catenary can be plotted, but in order to fix a real cable on the curve we need two additional pieces of data[3]: x_1, x_2 or h_1, h_2, or x_1 and the cable length, etc .... The three unknowns can be solved from three equations as given subsequently.

Other useful info:

- H is constant throughout (as required for horizontal equilibrium)
- $\frac{dy}{dx} = \sinh\left(\frac{x}{\alpha}\right)$
- Cable length in the segment $a \le x \le b$:

$$L(a,b) = \int_a^b \sqrt{1+\left(\frac{dy}{dx}\right)^2}\, dx = \alpha\left[\sinh(\frac{b}{\alpha}) - \sinh(\frac{a}{\alpha})\right]$$



[2] https://en.wikipedia.org/wiki/Catenary

[3] Two cables symmetrical about the y-axis will have similar properties such as length, pole heights, distance between poles, difference in pole heights, tension, sag.

- $\theta = \tan^{-1}(\frac{dy}{dx})$ and $T = \frac{H}{\cos\theta}$
- $h_1 = y(x_1)$ & $h_2 = y(x_2)$ $\quad x_1 < 0 < x_2$
- Sag $s = \frac{1}{2}(h_1 + h_2) - \alpha$

Now, consider the following four cases of electric power cables (suspended between 2 posts) having its mass per unit length of w = 2.8 kg/m. Write ONE single program[4] to solve all four cases, and in each case, it determines α, x_1, x_2 (chosen as the primary unknowns), and from which computes the sag s, the cable length (if not given), as well as the cable tensions at the two posts of the cables. The program also plots the cable for each case--You may extend the given program which obtains the solution for Case (a) below:

Case (a): Cable length of 60m [Eq. $L(x_1, x_2) = 60$]; span of 56m [Eq. $(x_2 - x_1) = 56$]; and one support is higher than the other by 2.65m [Eq. $y(x_1) - y(x_2) = 2.65$ (take $h_1 > h_2$ to fix the cable)]. A similar problem is posted here[5].

Case (b): Cable touches the flat ground level with poles symmetrically placed [Eq. $(x_1 + x_2) = 0$], both of 50m high [Eq. $y(x_2) - \alpha = 50$]; and with cable length of 120m [Eq. $L(0, x_2) = 60$ (impose x_2>0 while x_1 = -x_2 by symmetry)]. Note: Numerical solution for this problem is in Case 1 of the paper: http://euclid.trentu.ca/aejm/V4N1/Chatterjee.V4N1.pdf

Case (c): Cable touches the flat ground level with the left pole of 70m high [Eq. $y(-|x_1|) - \alpha = 70$]; right pole of 50m high [Eq. $y(x_2) - \alpha = 50$]; and cable length of 140m [Eq. $L(x_1, x_2) = 140$]. Note: Numerical solution for this problem is in Case 2 of Chatterjee's paper.

Case (d): The supports are 70m apart [Eq. $(x_2 - x_1) = 70$], and one is 7m higher than the other [Eq. $y(x_1) - y(x_2) = 7$ (i.e. assume the right support lower)]; in addition, the bottom of the cable is 2m lower than the right support [Eq. $\alpha + 2 = y(x_2)$]. Note: This case is inspired by the question here: http://ask.metafilter.com/98191/Length-of-a-catenary-in-terms-of-sag-and-span

---

[4] For each case, solve the three given equations (shown shaded) using `roots()`. Observe that when the data allows 2 possible cable positions, the given equations lead to a unique solution. For convenience, frequently used math functions should be implemented by user-defined functions with suitable parameters. It's simplest to use the same set of important *variable names* for all cases. However, a different set of equations is to be implemented by a different user-defined function.

[5] Interestingly, the problem at http://mathhelpforum.com/calculus/96398-catenary-cable-different-heights.html requires another formulation with off-center y-axis. To use our current formulation, we modify the problem slightly: A cable of length 20.7m [Eq. $L(x_1, x_2) = 20.7$] is hung by posts 20m apart [Eq. $x_2 - x_1 = 20$] while the right post is 5m higher than the left post [Eq. $y(x_2) - y(x_1) = 5$]. These 3 equations give solution: α = 61.917m, x1 = 5.257m, x2 = 25.257m, L = 20.7m. Shifting the y-axis to the left post yields the desired equation for the cable as found by one of the responders.

The technique here shows the ease of getting numerical solutions to nonlinear problems, which otherwise require complex analytical treatment. It also shows how the given data is translated into equations for unique solution.

```
main() {
     tol# = 1.e-6;  // Root accuracy

     // **** Global data shared by all cases ****
     N = 3; // Number of equations
     w = 2.8 * 9.81; // N/m: cable weight per unit length
     // *****************************************

     clearplot(); useoption("RADIANS");
     setrange(-60, -10, 60, 100); // Data range for plotting
     zero(X[N], f[N]); // (Global) storage for solutions and residuals
     CaseA(); // Solve Case (a)
}

CaseA()
{ // Complete solution for Case (a)
     float a, x1, x2; // Local variables
     print(^^^" Case (a)");

     // **** Global data for use in FXA() ****
     d = 56;        // Span
     L = 60;        // Cable length
     h = 2.65; // Differential height
     // **************************************

     print(^, " Data: d =", d, "  L =", L, "     h   =", h);
     defmat(X, 10, -30, 30);  // Solution estimates
     roots(X,f, FXA(X, f));
     print(^, "   a =    ", a = X[1], "  x1 =  ", x1 = X[2],
          "  x2  = ", x2 = X[3], "   Residual =", hypot(!f));
     CableProperties(a, x1, x2);
     setop(C, 11); // Plot color
     Plot(a, x1, x2);
}


FXA(mat X, mat f)
{ // Equations for Case a
     // This function needs previously defined global data for d, L, h
     // Since this function will be called numerous times, the number of
     // parameters are kept to the minimum for faster execution.

     float a = X[1], x1 = X[2], x2 = X[3]; // For convenience

     f[1] = x2-x1 - d;                      // Span
     f[2] = y(x1,a) - y(x2,a) - h;          // Differential height
     f[3] = Length(x1, x2, a) - L;          // Cable length
}

CableProperties(float a, float x1, float x2)
{
     float h1, h2, Sag, CL, H, D1, D2, Theta1, Theta2, T1, T2;
     // Global variables
     h1 = y(x1, a); h2 = y(x2, a);
     Sag = 0.5*(h1+h2) - a;
     CL = Length(x1, x2, a);  // Cable length
     H = a * w; // Newtons;
     D1 = sinh(x1/a); D2 = sinh(x2/a); // dy/dx at x1, x2
     Theta1 = atan(D1);  Theta2 = atan(D2);
     T1 = H/cos(Theta1); T2 = H/cos(Theta2);
     print(^, "  h1 =    ", h1, "  h2 =  ", h2, "  Sag = ", Sag,
            ^, "  Length =", CL, "  T1 =", T1,   "    T2  =", T2);
}


y(float x, float a)
{ // Catenary equation
     return a*cosh(x/a);
}
```

```
Dy(float x, float a)
{ // First derivative of y wrt x
    return sinh(x/a);
}

Plot(float a, float x1, float x2)
{
    plot(x, x1, x2, y(x, a));
}

Length(float x1, float x2, float a)
{
    return a*(sinh(x2/a)-sinh(x1/a));
}
```

Your report should include the program, the output, the plots and comments on the correctness of your program & output.

P4.3 [30 maks]

Develop and test the program that finds the coefficients *a* and *b* of the function[6]

$$y = a\,(x-1)^b$$

in order to best fit the following set of data points.

x_i	1.1	2.2	3.2	4.8	7.6	10.
y_i	1.6	2.7	3.1	4.7	6.3	5.9

The program should:

- plot the fitted curve
- draw the data points
- draw the vertical lines from the data points to the fitted curve.
- compute the sum of the deviations squared: $D = \sum_{i=1}^{5} \left(y_i - y(x_i)\right)^2$

Your report should include the program, all output, and the plot.

[6] The fitting function is nonlinear in the unknowns *a*, *b*, and hence it should first be linearized by taking natural log of both sides: $\ln y = \ln a + b\ln(x-1)$ or $Y = A + bX$ where $Y = \ln y,\ \ X = \ln(x-1)$, and $A = \ln a$. The model equation $Y = A + bX$ is linear in the unknowns *A*, *b*. Thus, the problem becomes: find *A* and *b* that fits the data points { X_i, Y_i } instead of { x_i, y_i }. See Sample Program 7.2