CEE/MAE M20
Introduction to Computer Programming with MATLAB

Fall 2018

Final Project

Due: Friday, December 14, 2018, 11:59 pm

Contents

Instructions for report preparation

Prepare a technical report on solving equations of motion of type $m\ddot{x} = f(x)$ (or $m_i\ddot{x}_i = f_i(\mathbf{x})$ if $\mathbf{x}$ is a vector) both implicitly and explicitly for numerical simulation. Your report should also address the benefits and drawbacks of the implicit and explicit approach.

This project has four problems in the next four parts. Your report must use these four examples and address the specific questions asked within each problem. It should include the pseudocode or flowchart and an overview (or schematic) of your simulator, e.g. how the different functions and scripts work together to numerically simulate physical systems. A future MAE20/CEE20 student should be able to write a numerical simulator after reading your report.

Your report should be formatted as a journal article and divided into multiple sections and subsections (and, if necessary, a reference section). There is no page limit.

You are also encouraged to take this opportunity to learn LaTeX, given the abundance of scientific notations.

Submission instructions

Your submission should include (1) your .pdf report named as `project _ UID.pdf`, where `UID` is your university ID number, and (2) a .zip folder, named `project _ UID.zip`, that contains all your codes. Your .zip folder may *optionally* include four subfolders (named as `project _ UID_ p#` where # is the problem number). Your submission must include four *main* scripts for the four problems named as `project _ UID_ p# .m`.

Grading

There are 30 points in total, divided into four problems:

1. Problem 1	6 points
2. Problem 2	8 points
3. Problem 3	12 points
4. Problem 4	4 points

Part I

Rigid sphere falling in viscous flow

Consider the system shown in Fig. 1. A metal sphere with density $\rho_{\text{metal}} = 7000$ kg/m^3 and radius $R = 0.035$ m is inside an infinite bath of viscous fluid with density $\rho_{\text{fluid}} = 1000$ kg/m^3. The y-coordinate of the center of the sphere is $y_c(t)$ that can vary as a function of time. Initially, at time $t = 0$, the sphere is at rest (i.e. velocity is zero) located at $y_c = 0$. The sphere starts to fall starting at $t = 0$ under gravity and, at the same time, it experiences a drag force from the fluid. The viscosity of the fluid is $\mu = 1000$ Pa-s.

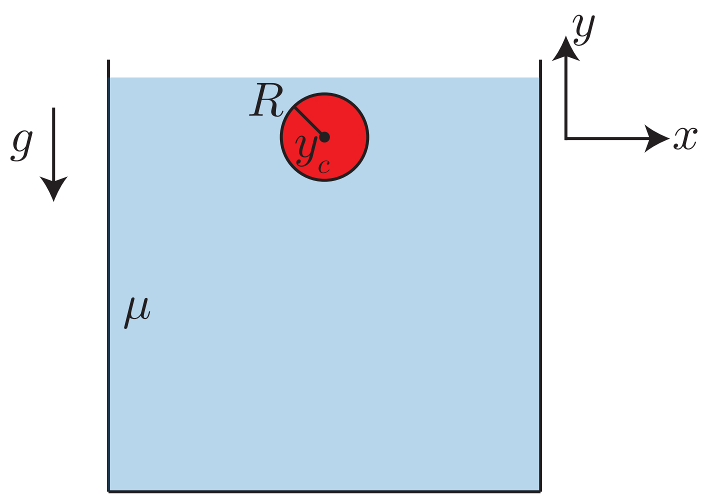

Figure 1: A rigid sphere falling in viscous fluid.

Following Newton's second law of motion ($F = ma$), we can write the equation of motion for $y_c(t)$:

$$m\ddot{y}_c = -W - (6\pi\mu R)\dot{y}_c, \tag{1}$$

where $W = \frac{4}{3}\pi R^3(\rho_{\text{metal}} - \rho_{\text{fluid}})g$ is the weight, $g = 9.8$ m/s^2 is the gravitational pull, and $m = \frac{4}{3}\pi R^3 \rho_{\text{metal}}$ is the mass of the sphere. Hereafter, dot ($\dot{()}$) represents derivative with respect to time. The second term on the right hand side is the drag force and can be obtained from Stokes's law. Eq. 1 is simply a statement of mass times acceleration is equal to the gravitational force and viscous drag.

Similar to homework 4, you have to develop a simulator that marches forward in time (with time step size Δt). Taking an *implicit* approach, the discretized governing equation to move from $t = t_k$ to $t = t_{k+1} = t_k + \Delta t$ is

$$\frac{m}{\Delta t}\left[\frac{y_c(t_{k+1}) - y_c(t_k)}{\Delta t} - \dot{y}_c(t_k)\right] = -W - (6\pi\mu R)\frac{y_c(t_{k+1}) - y_c(t_k)}{\Delta t}, \tag{2}$$

where the position ($y_c(t_k)$) and velocity ($\dot{y}_c(t_k)$) from the previous time step are known. After solving for the new position ($y_c(t_{k+1})$) using Eq. 2, the new velocity can be readily calculated:

$$\dot{y}_c(t_{k+1}) = \frac{y_c(t_{k+1}) - y_c(t_k)}{\Delta t}. \tag{3}$$

In an *explicit* approach, the discretized governing equation is

$$\frac{m}{\Delta t}\left[\frac{y_c(t_{k+1}) - y_c(t_k)}{\Delta t} - \dot{y}_c(t_k)\right] = -W - (6\pi\mu R)\dot{y}_c(t_k), \tag{4}$$

which can be readily solved for $y_c(t_{k+1})$.

Deliverable: Write solvers in MATLAB that simulates the position and the velocity of the sphere as a function of time (between $0 \leq t \leq 1$ seconds) *both* implicitly and explicitly. Use $\Delta t = 0.01$ s for the implicit simulation and $\Delta t = 0.0001$ s for the explicit simulation in your submitted code. Your code should generate plots showing the position and velocity as functions of time for both implicit and explicit simulation. Use appropriate titles and axis labels for all the plots.

In your report, include these plots. Verify the accuracy of your simulations using the following two criteria:

1. Compute the terminal velocity [1] from your simulation as well as from Eq. 1.
2. When the viscosity is zero ($\mu = 0$), the sphere is freely falling under gravity. The distance traversed by a freely falling object is $\frac{1}{2}gt^2$.

Try increasing the time step size (Δt) in your explicit and implicit simulations. It may help you address some of the drawbacks and benefits of the two approach.

[1]The velocity eventually reaches a stable value and does not change with time. This is the terminal velocity.

Part II

Rigid spheres and elastic beam falling in viscous flow

Referring to Fig. 2, let us consider three rigid spheres on an elastic beam that is falling under gravity in a viscous fluid. The viscosity of the fluid is $\mu = 1000$ Pa-s. The radii of the spheres are $R_1 = 0.005$ m, $R_2 = 0.025$ m, and $R_3 = 0.005$ m. The density of the spheres is $\rho_{\text{metal}} = 7000$ kg/m^3 whereas the fluid density is $\rho_{\text{fluid}} = 1000$ kg/m^3. The positions of the spheres are (x_1, y_1), (x_2, y_2), and (x_3, y_3). We can construct a *degrees of freedom* (dof) vector,

$$\mathbf{q} = \begin{bmatrix} x_1 \\ y_1 \\ x_2 \\ y_2 \\ x_3 \\ y_3 \end{bmatrix} \tag{5}$$

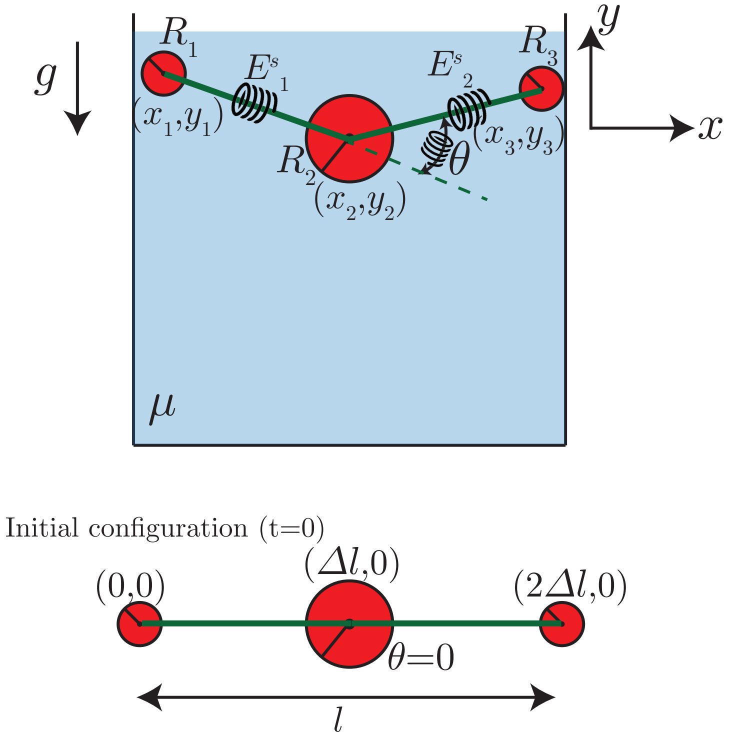

Figure 2: Rigid spheres attached to an elastic beam falling in viscous fluid.

Unlike the previous problem where we solved for a scalar, we have to solve for a vector with six elements. You may want to review Problem 3 of Homework 4 which involved solving a system of equations using Newton's method.

In this problem, in addition to gravity and viscous drag, we also have to include the elastic force of the beam. The elastic beam has a length, $l = 0.10$ m, cross-sectional radius, $r_0 = 0.001$ m, and Young's modulus, $E = 1.0 \times 10^9$ Pa. This corresponds to a stretching stiffness of $EA = E\pi r_0^2$ and bending stiffness of $EI = E\pi r_0^4/4$. We can consider the beam to be a combination of springs as shown in Fig. 2. The elastic energy of the beam is

$$E^{\text{elastic}} = E_1^s + E_2^s + E^b, \tag{6}$$

where superscript s refers to *stretching*, b corresponds to bending, and

$$E_1^s = \frac{1}{2}EA\left(1 - \frac{\sqrt{(x_2-x_1)^2+(y_2-y_1)^2}}{\Delta l}\right)^2 \Delta l, \tag{7}$$

$$E_2^s = \frac{1}{2}EA\left(1 - \frac{\sqrt{(x_3-x_2)^2+(y_3-y_2)^2}}{\Delta l}\right)^2 \Delta l, \tag{8}$$

$$E^b = \frac{1}{2}\frac{EI}{\Delta l}\left(2\tan\left(\frac{\theta}{2}\right)\right)^2, \tag{9}$$

where $\Delta l = \frac{l}{2}$ is the length of each segment between two nodes and θ is the *turning angle*, as shown in Fig. 2. Note that θ can be expressed in terms of $x_1, y_1, x_2, y_2, x_3, y_3$. Since elastic force is conservative, the elastic force on a degree of freedom (dof), e.g. x_1, is the negative of the gradient of elastic energy with respect to that dof, e.g. $-\frac{\partial E^{\text{elastic}}}{\partial x_1}$. You do not need to analytically evaluate the elastic force; see Appendix for MATLAB functions provided with this assignment that evaluate the gradient of the elastic energies.

Initially, at time $t = 0$, the entire system is at rest (i.e. velocity is zero) and the positions of the nodes are shown in Fig. 2. The entire structure starts to fall starting at $t = 0$ under gravity and, at the same time, it experiences a drag force from the fluid. First, write down the six equations of motion and their discretized version. For convenience, the first two equations of motion are

$$m_1\ddot{x}_1 = -(6\pi\mu R_1)\dot{x}_1 - \frac{\partial E^{\text{elastic}}}{\partial x_1}, \tag{10}$$

$$m_1\ddot{y}_1 = -W_1 - (6\pi\mu R_1)\dot{y}_1 - \frac{\partial E^{\text{elastic}}}{\partial y_1}, \tag{11}$$

where $W_1 = \frac{4}{3}\pi R_1^3(\rho_{\text{metal}} - \rho_{\text{fluid}})g$, $g = 9.8$ m/s^2, $m_1 = \frac{4}{3}\pi R_1^3 \rho_{\text{metal}}$. The discretized version is

$$\frac{m_1}{\Delta t}\left[\frac{x_1(t_{k+1}) - x_1(t_k)}{\Delta t} - \dot{x}_1(t_k)\right] = -(6\pi\mu R_1)\frac{x_1(t_{k+1}) - x_1(t_k)}{\Delta t} - \frac{\partial E^{\text{elastic}}}{\partial x_1}, \tag{12}$$

$$\frac{m_1}{\Delta t}\left[\frac{y_1(t_{k+1}) - y_1(t_k)}{\Delta t} - \dot{y}_1(t_k)\right] = -W_1 - (6\pi\mu R_1)\frac{y_1(t_{k+1}) - y_1(t_k)}{\Delta t} - \frac{\partial E^{\text{elastic}}}{\partial y_1}, \tag{13}$$

This can be re-written as

$$f_1 \equiv \frac{m_1}{\Delta t}\left[\frac{x_1(t_{k+1}) - x_1(t_k)}{\Delta t} - \dot{x}_1(t_k)\right] + (6\pi\mu R_1)\frac{x_1(t_{k+1}) - x_1(t_k)}{\Delta t} + \frac{\partial E^{\text{elastic}}}{\partial x_1} = 0, \tag{14}$$

$$f_2 \equiv \frac{m_1}{\Delta t}\left[\frac{y_1(t_{k+1}) - y_1(t_k)}{\Delta t} - \dot{y}_1(t_k)\right] + W_1 + (6\pi\mu R_1)\frac{y_1(t_{k+1}) - y_1(t_k)}{\Delta t} + \frac{\partial E^{\text{elastic}}}{\partial y_1} = 0. \tag{15}$$

For an *implicit* simulation, the elastic forces have to be evaluated using the positions at $t = t_{k+1}$. An implicit solution using Newton's method will require the Jacobian matrix (size 6×6). Write down the entries of the Jacobian matrix (or elaborate the steps to evaluate the Jacobian). It will involve terms such as $\frac{\partial^2 E^{\text{elastic}}}{\partial x_1 \partial y_1}$ (i.e. the *Hessian* of the energy). You do not need to analytically evaluate such terms; see Appendix for MATLAB functions provided with this assignment that evaluate the Hessian of elastic energies.

After solving the six discretized governing equations for the six degrees of freedom, i.e. $\mathbf{q}(t_{k+1})$, the velocity can simply be computed use $\dot{\mathbf{q}}(t_{k+1}) = \frac{\mathbf{q}(t_{k+1}) - \mathbf{q}(t_k)}{\Delta t}$.

Deliverable: Write a solver in MATLAB that simulates the position and the velocity of the sphere as a function of time (between $0 \leq t \leq 10$ seconds) both implicitly and explicitly. Use $\Delta t = 10^{-2}$ s for the implicit simulation and $\Delta t = 10^{-5}$ s for the explicit one. Your final submitted script, named `project _ UID _ p2.m`, should generate plots showing the position and velocity of the middle node and the turning angle θ as functions of time. Use appropriate titles and axis labels for all the plots.

For your report:

1. Include the aforementioned plots.
2. What happens to the turning angle if all the radii (R_1, R_2, R_3) are the same? Does your simulation agree with your intuition?
3. Try changing the time step size (Δt), particularly for your explicit simulation, and use the observation to elaborate the benefits and drawbacks of the explicit and implicit approach.

Part III

Generalized case of elastic beam falling in viscous flow

Generalize the implicit simulator such that the beam is composed of N nodes (N is an odd number). The sphere at the middle node has radius $R_{\text{mid}} = 0.025$ and all the other nodes have a sphere with radius $R = \Delta l/10$, where $\Delta l = l/(N-1)$ is the length of each discrete segment. Fig. 3 illustrates the system when $N = 5$. You are not required (but encouraged) to write an explicit solver for this problem.

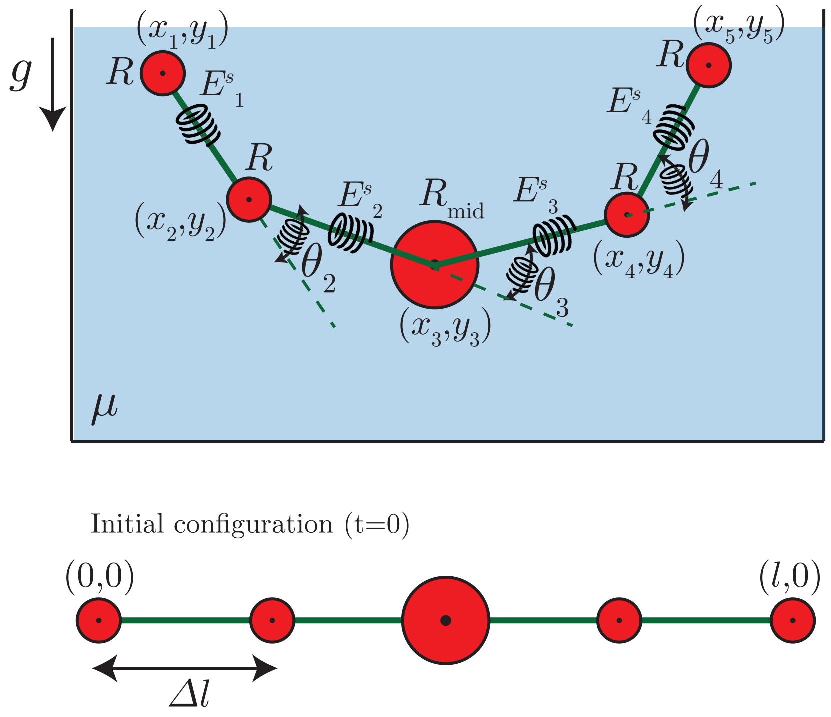

Figure 3: $N = 5$ rigid spheres attached to an elastic beam falling in viscous fluid.

Now we have $N-1$ stretching springs and $N-2$ bending springs. The elastic energy is

$$E^{\text{elastic}} = \sum_{j=1}^{N-1} E^s_j + \sum_{j=2}^{N-1} E^b_j, \tag{16}$$

where E^b_j is associated with the turning angle θ_j.

Deliverable: Write a solver in MATLAB that simulates the system of the sphere as a function of time (between $0 \leq t \leq 50$ seconds) implicitly (and, optionally, explicitly). Use $\Delta t = 10^{-2}$ s for the implicit simulation and $N = 21$. Your final submitted script, named `project _ UID _ p3.m`, should generate a **movie** showing the shape of the beam. Upload the movie to YouTube and include the URL in your report (preferred) or submit the movie file (please ensure that the file size is < 10 MB) along with your code.

For your report:

1. Include two plots showing the vertical position and velocity of the middle node with time. What is the terminal velocity?

2. Include the final deformed shape of the beam. You can also use screenshots from your movie to explain the simulation.

3. You can use this problem to discuss the significance of spatial discretization (i.e. the number of nodes, N) and temporal discretization (i.e. time step size, Δt). Any simulation should be sufficiently discretized such that the quantifiable metrics, e.g. terminal velocity, do not vary much if N is increased and Δt is decreased. You may include plots of terminal velocity vs. the number of nodes and terminal velocity vs. the time step size.

Part IV

Elastic beam bending

[This is a challenging (yet exciting) component of the project and, therefore, only four points have been allocated to this part.]

Recall the problem on Euler-Bernoulli beam bending from homework 7 where we solved a system of equations to study the displacement of a simply-supported aluminum beam subjected to a single point load. The bar has length $l = 1$ m and a constant, circular-tube cross section with outer radius $R = 0.013$ m and inner radius $r = 0.011$ m. A force, $P = 2000$ N, is applied 0.75 m away from the left-hand edge.The modulus of elasticity is $E = 70$ GPa for aluminum and I is the moment of inertia of the cross section, $I = \frac{\pi}{4}(R^4 - r^4)$.

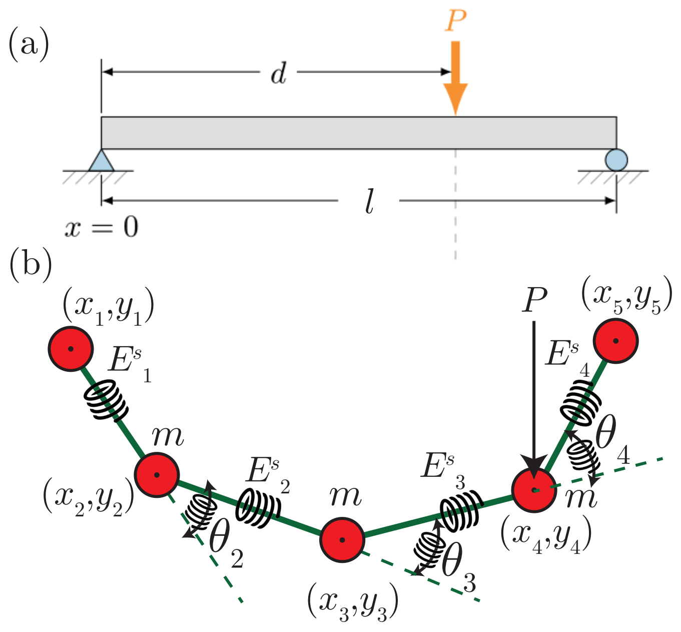

Figure 4: (a) Elastic beam and (b) its discrete representation.

We can represent the beam as a mass-spring system with a mass m located at each node where

$$m = \pi(R^2 - r^2)l\rho/(N - 1) \tag{17}$$

and density of aluminum is $\rho = 2700$ kg/m^3. The formulation of the elastic energy remains the same. Instead of gravity and viscous drag as external forces, the only external force is the force P applied at a node that is located 0.75m away from the first node. If no node is exactly 0.75 m away, take the node that is closest to the desired distance.

Also note that the first node is constrained along both x and y-axes and the last node is constrained along y-axis. For these three constrained degrees of freedom, the governing equations are simply

$$x_1(t_{k+1}) = 0, \tag{18}$$

$$y_1(t_{k+1}) = 0, \tag{19}$$

$$y_N(t_{k+1}) = 0. \tag{20}$$

Deliverable:

Write a solver in MATLAB that simulates the beam as a function of time (between $0 \leq t \leq 1$ seconds) implicitly. Use $\Delta t = 10^{-2}$ s for the implicit simulation and $N = 50$. Your final submitted script (`project _ UID _ p4.m`) should plot the maximum vertical displacement, y_{max}, of the beam as a function of time. Depending on your coordinate system, y_{max} may be negative.

For your report:

1. Include this plot of y_{max}. Does y_{max} eventually reach a steady value? Examine the accuracy of your simulation against the theoretical prediction from Euler beam theory:

$$y_{\text{max}} = \frac{Pc\,(L^2 - c^2)^{1.5}}{9\sqrt{3}EI\,l} \qquad \text{where} \qquad c = \min(d, l - d) \tag{21}$$

2. What is the benefit of your simulation over the predictions from beam theory? To address this, consider a higher load $P = 20000$ such that the beam undergoes large deformation. Compare the simulated result against the prediction from beam theory in Eq. 21. Euler beam theory is only valid for small deformation whereas your simulation, if done correctly, should be able to handle large deformation. Optionally, you can make a plot of P vs. y_{max} using data from both simulation and beam theory and quantify the value of P where the two solutions begin to diverge.

Appendices

A MATLAB code for force and Jacobian

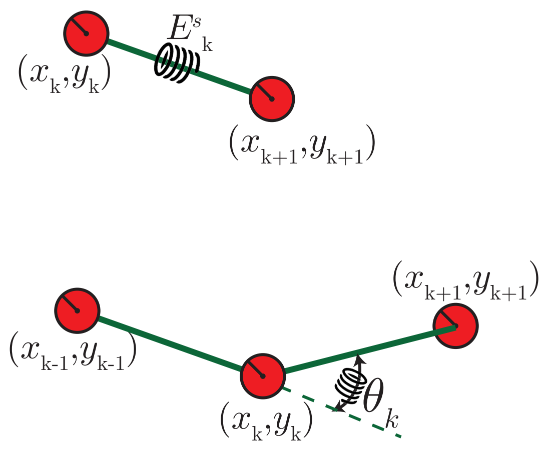

Figure 5: Stretching and bending springs.

Gradient and Hessian of stretching energy:

1. `F = gradEs(xk, yk, xkp1, ykp1, l_k, EA)` takes six inputs: $x_k, y_k, x_{k+1}, y_{k+1}, \Delta l, EA$; and returns a vector with 4 elements that contains the following gradient:

$$\begin{bmatrix} \frac{\partial E^s_k}{\partial x_k} \\ \frac{\partial E^s_k}{\partial y_k} \\ \frac{\partial E^s_k}{\partial x_{k+1}} \\ \frac{\partial E^s_k}{\partial y_{k+1}} \end{bmatrix} \tag{22}$$

   Derivative of E^s_k with respect to any other degree of freedom is zero.

2. `J = hessEs(xk, yk, xkp1, ykp1, l_k, EA)` takes six inputs: $x_k, y_k, x_{k+1}, y_{k+1}, \Delta l, EA$; and returns a matrix with 4×4 elements that contains the following Hessian:

$$\begin{bmatrix} \frac{\partial^2 E_k^s}{\partial x_k \partial x_k} & \frac{\partial^2 E_k^s}{\partial y_k \partial x_k} & \frac{\partial^2 E_k^s}{\partial x_{k+1} \partial x_k} & \frac{\partial^2 E_k^s}{\partial y_{k+1} \partial x_k} \\ \frac{\partial^2 E_k^s}{\partial x_k \partial y_k} & \frac{\partial^2 E_k^s}{\partial y_k \partial y_k} & \frac{\partial^2 E_k^s}{\partial x_{k+1} \partial y_k} & \frac{\partial^2 E_k^s}{\partial y_{k+1} \partial y_k} \\ \frac{\partial^2 E_k^s}{\partial x_k \partial x_{k+1}} & \frac{\partial^2 E_k^s}{\partial y_k \partial x_{k+1}} & \frac{\partial^2 E_k^s}{\partial x_{k+1} \partial x_{k+1}} & \frac{\partial^2 E_k^s}{\partial y_{k+1} \partial x_{k+1}} \\ \frac{\partial^2 E_k^s}{\partial x_k \partial y_{k+1}} & \frac{\partial^2 E_k^s}{\partial y_k \partial y_{k+1}} & \frac{\partial^2 E_k^s}{\partial x_{k+1} \partial y_{k+1}} & \frac{\partial^2 E_k^s}{\partial y_{k+1} \partial y_{k+1}} \end{bmatrix} \tag{23}$$

Gradient and Hessian of bending energy:

1. `F = gradEb(xkm1, ykm1, xk, yk, xkp1, ykp1, l_k, EI)` takes eight inputs: $x_{k-1}, y_{k-1}, x_k, y_k, x_{k+1}, y_{k+1}, \Delta l, EI$; and returns a vector with 6 elements that contains the following gradient:

$$\begin{bmatrix} \frac{\partial E_k^b}{\partial x_{k-1}} \\ \frac{\partial E_k^b}{\partial y_{k-1}} \\ \frac{\partial E_k^b}{\partial x_k} \\ \frac{\partial E_k^b}{\partial y_k} \\ \frac{\partial E_k^b}{\partial x_{k+1}} \\ \frac{\partial E_k^b}{\partial y_{k+1}} \end{bmatrix} \tag{24}$$

   Derivative of E_k^b with respect to any other degree of freedom is zero.

2. `J = hessEb(xkm1, ykm1, xk, yk, xkp1, ykp1, l_k, EI)` takes eight inputs: $x_{k-1}, y_{k-1}, x_k, y_k, x_{k+1}, y_{k+1}, \Delta l, EI$; and returns a matrix with 6×6 elements that contains the following Hessian:

$$\begin{bmatrix} \frac{\partial^2 E_k^b}{\partial x_{k-1} \partial x_{k-1}} & \frac{\partial^2 E_k^b}{\partial y_{k-1} \partial x_{k-1}} & \frac{\partial^2 E_k^b}{\partial x_k \partial x_{k-1}} & \frac{\partial^2 E_k^b}{\partial y_k \partial x_{k-1}} & \frac{\partial^2 E_k^b}{\partial y_{k+1} \partial x_{k-1}} & \frac{\partial^2 E_k^b}{\partial y_{k+1} \partial x_{k-1}} \\ \frac{\partial^2 E_k^b}{\partial x_{k-1} \partial y_{k-1}} & \frac{\partial^2 E_k^b}{\partial y_{k-1} \partial y_{k-1}} & \frac{\partial^2 E_k^b}{\partial x_k \partial y_{k-1}} & \frac{\partial^2 E_k^b}{\partial y_k \partial y_{k-1}} & \frac{\partial^2 E_k^b}{\partial y_{k+1} \partial y_{k-1}} & \frac{\partial^2 E_k^b}{\partial y_{k+1} \partial y_{k-1}} \\ \frac{\partial^2 E_k^b}{\partial x_{k-1} \partial x_k} & \frac{\partial^2 E_k^b}{\partial y_{k-1} \partial x_k} & \frac{\partial^2 E_k^b}{\partial x_k \partial x_k} & \frac{\partial^2 E_k^b}{\partial y_k \partial x_k} & \frac{\partial^2 E_k^b}{\partial y_{k+1} \partial x_k} & \frac{\partial^2 E_k^b}{\partial y_{k+1} \partial x_k} \\ \frac{\partial^2 E_k^b}{\partial x_{k-1} \partial y_k} & \frac{\partial^2 E_k^b}{\partial y_{k-1} \partial y_k} & \frac{\partial^2 E_k^b}{\partial x_k \partial y_k} & \frac{\partial^2 E_k^b}{\partial y_k \partial y_k} & \frac{\partial^2 E_k^b}{\partial y_{k+1} \partial y_k} & \frac{\partial^2 E_k^b}{\partial y_{k+1} \partial y_k} \\ \frac{\partial^2 E_k^b}{\partial x_{k-1} \partial x_{k+1}} & \frac{\partial^2 E_k^b}{\partial y_{k-1} \partial x_{k+1}} & \frac{\partial^2 E_k^b}{\partial x_k \partial x_{k+1}} & \frac{\partial^2 E_k^b}{\partial y_k \partial x_{k+1}} & \frac{\partial^2 E_k^b}{\partial y_{k+1} \partial x_{k+1}} & \frac{\partial^2 E_k^b}{\partial y_{k+1} \partial x_{k+1}} \\ \frac{\partial^2 E_k^b}{\partial x_{k-1} \partial y_{k+1}} & \frac{\partial^2 E_k^b}{\partial y_{k-1} \partial y_{k+1}} & \frac{\partial^2 E_k^b}{\partial x_k \partial y_{k+1}} & \frac{\partial^2 E_k^b}{\partial y_k \partial y_{k+1}} & \frac{\partial^2 E_k^b}{\partial y_{k+1} \partial y_{k+1}} & \frac{\partial^2 E_k^b}{\partial y_{k+1} \partial y_{k+1}} \end{bmatrix} \tag{25}$$