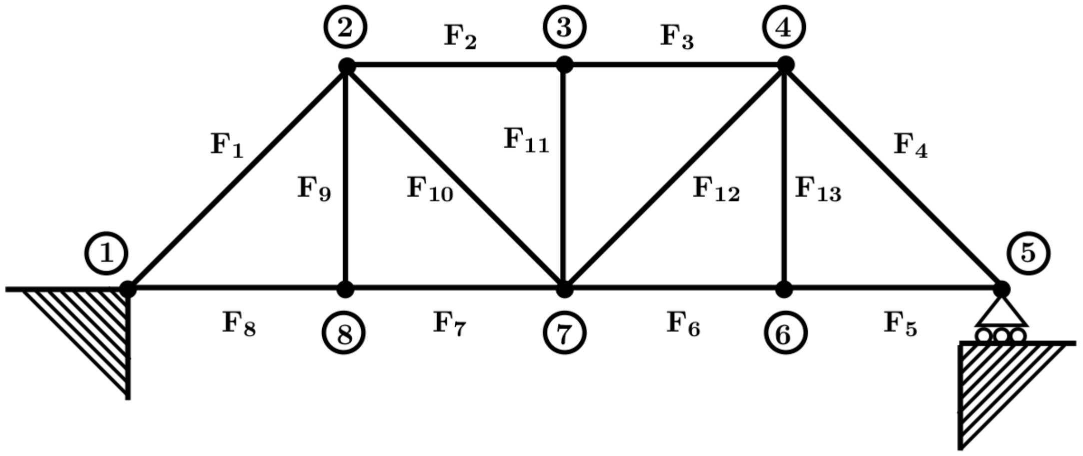
Due Thursday January 27th at 11:59pm

## Problem 1

Consider the bridge truss diagrammed below:



Given a vector of external forces (what we call the "data" or the "right hand side") $\mathbf{b}$ on the bridge, we can compute the forces $F_1$, $F_2$, $\ldots$, $F_{13}$ (the forces on each girder) by solving the system $A\mathbf{x} = \mathbf{b}$, where $\mathbf{x} = (F_1, F_2, \ldots F_{13})$ is the column vector of unknown forces and $A$ is the $13 \times 13$ matrix, which will be called in for you in the solution template.

We will solve for the unknown forces assuming that there are three vehicles at positions 13, 11 and 9 with weights $\pi$ tons, $e^2$ tons and 3 tons, respectively. This means that there is a weight of $\pi$ tons pulling down on girder 13, a weight of $e^2$ tons pulling down on girder 11 and a weight of 3 tons pulling down on girder 9, so we need to use $\mathbf{b} = (0, 0, 0, 0, 0, 0, 0, 0, 3, 0, e^2, 0, \pi)$ (remember this is a column vector; be very careful about the shapes).

(1) Solve for $\mathbf{x}$ and save your answer as a $13 \times 1$ column vector named `A1`.

(2) Compute the $PA = LU$ factorization of $A$. Save a copy of the resulting matrix $L$ in a variable named `A2`.

(3) From the $PA = LU$ factorization above, find the vector $\mathbf{c}$ in the $U\mathbf{x} = c$ problem (don't forget to permute – for MATLAB and Python this will work differently). Save the intermediate vector $\mathbf{c}$ in a variable named `A3`.

(4) Now suppose that we add weight to the truck in position 8 (which corresponds to the 9th entry of $\mathbf{b}$) in increments of 0.001 tons until the bridge collapses. Each girder can withstand 20 tons of compression or tension (i.e., positive or negative forces) before breaking. This means that the bridge will collapse when the largest (in absolute value; i.e. max) force in $\mathbf{x}$ is larger than 20. Find the smallest weight of the truck at position 8 for which the bridge collapses and save this weight in a variable named `A4`. Find the index of the girder that breaks first and save it in a variable named `A5`. (**Note: In the problem description we started our indices at 1, but python starts counting indices at 0. You will therefore need to add 1 to the index from python.**)

Hint: The function `max` in MATLAB and `np.argmax` in python will be useful. Also, we are only changing the right hand side, and we already found the $PA = LU$ factorization, so can we use the $PA = LU$ factorization instead of redoing the Gaussian Elimination.

## Problem 2

In this problem, we will explore a simple way to model oscillatory motion. The model in this problem could be used to approximate the motion of a sky scraper, a swinging pendulum, or a start orbiting a supermassive black hole. (We will fully derive this method later in the course. For those of you interested in looking ahead, this problem implements the backward Euler method to solve a linear differential equation.)

Let the position of an object at time $t$ be given by $\mathbf{x}(t) = (x(t), y(t))$; a column vector where the first entry is the $x$-coordinate (abscissa) and the second entry is the $y$-coordinate (ordinate). The object's motion is (approximately) governed by the following numerical discretization:

$$A\mathbf{x}(t+1) = \mathbf{x}(t),$$

where

$$A = \begin{pmatrix} 1-\alpha & -\omega \\ \omega & 1-\alpha \end{pmatrix}.$$

For instance, if you knew the position of the object at time 0 (i.e., if you knew $\mathbf{x}(0)$) then you could calculate the position at time 1 by solving the linear system $A\mathbf{x}(1) = \mathbf{x}(0)$. You could then calculate the position of the object at time 2 by solving the system $A\mathbf{x}(2) = \mathbf{x}(1)$.

For this problem, assume that $\alpha = -0.002$, $\omega = 0.06$, and $\mathbf{x}(0) = (1, -1)$ (a column vector).

(1) Find the location of the object at times $0, 1, 2, ..., 1000$. Save the $x$-coordinates of the object at each time in a $1 \times 1001$ row vector named `A6`. Save the $y$-coordinates of the object at each time in a $1 \times 1001$ row vector named `A7`.

(2) The distance from the object to the origin is given by $d(t) = \sqrt{x(t)^2 + y(t)^2}$. Find $d(t)$ at times $0, 1, 2, ..., 1000$ and save these values in a $1 \times 1001$ row vector named `A8`.

(3) Find the first time when the distance from the object to the origin is less than 0.06. Save the resulting time in a variable named `A9` and the exact distance in a variable named `A10`.

## Problem 3

There are many mathematical applications where we want to rotate objects about an axis in $\mathbb{R}^3$ (three-dimensional space). For example, when controlling an aircraft, rotations about different axes are known as *yaw*, *pitch* and *roll*. Imagine that an object is at position $\mathbf{x} = (x, y, z)$ (a column vector representing the $x$, $y$, and $z$ coordinates). To rotate this vector counterclockwise by an angle $\theta$ about the $y$-axis, you can multiply $\mathbf{x}$ by the matrix

$$R(\theta) = \begin{pmatrix} \cos\theta & 0 & \sin\theta \\ 0 & 1 & 0 \\ -\sin\theta & 0 & \cos\theta \end{pmatrix}.$$

In other words, $\mathbf{b} = R(\theta)\mathbf{x}$ is the rotated vector of $\mathbf{x}$.

(1) Write a function that takes $\theta$ as an argument and returns the matrix $R(\theta)$. Use this function to calculate $R(\pi/8)$ and save this matrix in a variable named `A11`.

(2) Suppose we have the vector $\mathbf{x} = (\pi/10, 2.1, -e)$. Rotate this vector about the $y$-axis by an angle of $1\pi/3$. Save the resulting $3 \times 1$ vector in a variable named `A12`.

(3) Suppose that we have a vector $\mathbf{b} = (1.4, -\pi/10, 2.8)$ (a column vector), which was obtained by rotating another vector $\mathbf{x}$ about the $y$-axis by an angle of $\pi/6$. Find the vector $\mathbf{x}$ by solving the appropriate $3 \times 3$ matrix equation. Save the resulting $3 \times 1$ vector in a variable named `A13`.

(4) Find the inverse matrix of $R(3\pi/4)$ (using `inv` in MATLAB or `scipy.linalg.inv` in python). Save your answer in a variable named `A14`. (While we don't want to use inverse to solve matrix equations, there are times when we will need to just calculate the inverse.)

(5) This is an application where inverse matrices are used quite often, but it is still a bad idea to actually use the inverse command. But can we figure out an easier way to do it? The inverse of a rotation is just another rotation. That is, $R(\theta)^{-1} = R(\phi)$, where $\phi$ is a different angle. Find the angle $\phi$ such that $R(3\pi/4)^{-1} = R(\phi)$. Save this answer in a variable named `A15`. (This does not require any code, just some geometric reasoning. If you rotate a vector by an angle $\theta$, what would you have to do to rotate the vector back to where it started? The answer is not unique, because adding any multiple of $2\pi$ to an angle gives the same rotation matrix. Your answer should be between $-\pi$ and $\pi$.)