you can either write your own (short) code, or use the Matlab command `polyval`. This is a bit tricky, since `polyval` expects the polynomial coefficients to be listed in the reverse order than we use here. To evaluate $A(s)$ in Matlab you can use the command `polyval(a(m+1:-1:1),s)`. To evaluate $b(s)$ you can use `polyval([b(m:-1:1);1],s)`.

Note: no credit will be given for implementing any algorithm other than the one described in this problem.

6.19 *Quadratic placement.* We consider an integrated circuit (IC) that contains N cells or modules that are connected by K wires. We model a cell as a single point in $\mathbf{R}^2$ (which gives its location on the IC) and ignore the requirement that the cells must not overlap. The positions of the cells are

$$(x_1, y_1),\ (x_2, y_2), \ldots, (x_N, y_N),$$

i.e., x_i gives the x-coordinate of cell i, and y_i gives the y-coordinate of cell i. We have two types of cells: *fixed cells*, whose positions are fixed and given, and *free cells*, whose positions are to be determined. We will take the first n cells, at positions

$$(x_1, y_1), \ldots, (x_n, y_n),$$

to be the free ones, and the remaining $N - n$ cells, at positions

$$(x_{n+1}, y_{n+1}), \ldots, (x_N, y_N),$$

to be the fixed ones. The task of finding good positions for the free cells is called *placement.* (The fixed cells correspond to cells that are already placed, or external pins on the IC.) There are K wires that connect pairs of the cells. We will assign an orientation to each wire (even though wires are physically symmetric). Specifically, wire k goes *from* cell $I(k)$ *to* cell $J(k)$. Here I and J are functions that map wire number (*i.e.*, k) into the origination cell number (*i.e.*, $I(k)$), and the destination cell number (*i.e.*, $J(k)$), respectively. To describe the wire/cell topology and the functions I and J, we'll use the *node incidence matrix* A for the associated directed graph. The node incidence matrix $A \in \mathbf{R}^{K \times N}$ is defined as

$$A_{kj} = \begin{cases} 1 & \text{wire } k \text{ goes to cell } j,\ i.e.,\ j = J(k) \\ -1 & \text{wire } k \text{ goes from cell } j,\ i.e.,\ j = I(k) \\ 0 & \text{otherwise.} \end{cases}$$

Note that the kth row of A is associated with the kth wire, and the jth column of A is associated with the jth cell. The goal in placing the free cells is to use the smallest amount of interconnect wire, assuming that the wires are run as straight lines between the cells. (In fact, the wires in an IC are *not* run on straight lines directly between the cells, but that's another story. Pretending that the wires do run on straight lines seems to give good placements.) One common method, called *quadratic placement*, is to place the free cells in order to minimize the the total square wire length, given by

$$J = \sum_{k=1}^{K} \left((x_{I(k)} - x_{J(k)})^2 + (y_{I(k)} - y_{J(k)})^2 \right).$$

(a) Explain how to find the positions of the free cells, *i.e.*,

$$(x_1, y_1), \ldots, (x_n, y_n),$$

that minimize the total square wire length. You may make an assumption about the rank of one or more matrices that arise.

(b) In this part you will determine the optimal quadratic placement for a specific set of cells and interconnect topology. The mfile `qplace_data.m` defines an instance of the quadratic placement

problem. Specifically, it defines the dimensions n, N, and K, and $N-n$ vectors `xfixed` and `yfixed`, which give the x- and y-coordinates of the fixed cells. The mfile also defines the node incidence matrix `A`, which is $K \times N$. Be sure to explain how you solve this problem, and to explain the matlab source code that solves it (which you must submit). Give the optimal locations of the free cells. Check your placement against various others, such as placing all free cells at the origin. You will also find an mfile that plots a proposed placement in a nice way:
`view_layout(xfree,yfree,xfixed,yfixed,A)`.
This mfile takes as argument the x- and y-coordinates of the free and fixed cells, as well as the node incidence matrix that describes the wires. It plots the proposed placement. Plot your optimal placement using `view_layout`.

6.20 *Least-squares state tracking.* Consider the system $x(t+1) = Ax(t) + Bu(t) \in \mathbf{R}^n$, with $x(0)=0$. We *do not* assume it is controllable. Suppose $x_{\rm des}(t) \in \mathbf{R}^n$ is given for $t = 1, \ldots, N$ (and is meant to be the desired or target state trajectory). For a given input u, we define the mean-square tracking error as

$$E(u) = \frac{1}{N} \sum_{t=1}^{N} \|x(t) - x_{\rm des}(t)\|^2.$$

(a) Explain how to find $u_{\rm opt}$ that minimizes E (in the general case). Your solution can involve a (general) pseudo-inverse.

(b) *True or false:* If the system is controllable, there is a unique $u_{\rm opt}$ that minimizes $E(u)$. Briefly justify your answer.

(c) Find $E(u_{\rm opt})$ for the specific system with

$$A = \begin{bmatrix} 0.8 & 0.1 & 0.1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}, \quad B = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix},$$

$x_{\rm des}(t) = [t\ 0\ 0]^T$, and $N = 10$.

6.21 *Time series prediction.* We consider an autonomous discrete-time linear system of the form

$$x(t+1) = Ax(t), \quad y(t) = Cx(t) + v(t),$$

where $x(t) \in \mathbf{R}^n$, $y(t) \in \mathbf{R}$ is the measured output signal, and $v(t) \in \mathbf{R}$ represents an output noise signal. In this problem, you *do not know* the matrices $A \in \mathbf{R}^{n \times n}$ or $C \in \mathbf{R}^{1 \times n}$, the state $x(t)$ (including the initial state $x(0)$), or even the system order n. You *do know* the measured output signal for $t = 1, \ldots, p$:

$$y(1), \ldots, y(p).$$

We give you two more pieces of information: the system order n is less than 20, and the RMS value of the noise, *i.e.*, $\left((1/p)\sum_{t=1}^{p} v(t)^2\right)^{1/2}$, is small (on the order of 0.001). The goal is to predict $y(t)$ for the next q time steps, *i.e.*, to predict what $y(p+1), \ldots, y(p+q)$ will be. Here is the problem: get the time series data from the class web site in the file `timeseriesdata.m`, which gives $y(1), \ldots, y(200)$. (We have $p = 200$ in this specific problem.) Then, predict what $y(201), \ldots, y(220)$ are. Plot your estimates $\hat{y}(201), \ldots, \hat{y}(220)$, and also, of course, give us the numbers. (You may also want to plot the whole set of data, from $t = 1$ to $t = 220$, just to make sure your prediction satisfies the 'eyeball test'.) It is extremely important that you explain very clearly how you come up with your prediction. What is your method? If you make choices during your procedure, how do you make them?

6.22 *Extracting RC values from delay data.* We consider a CMOS digital gate that drives a load consisting of interconnect wires that connect to the inputs of other gates. To find the delay of the gate plus its load, we have to solve a complex, nonlinear ordinary differential equation that takes into account