

## Assignment 5

2.086/2.090 Spring 2013

---

Released: *Friday, 19 April, at 5 PM.*

Due: *Friday, 3 May, at 5 PM.*

Upload your solution as a zip file “YOURNAME\_ASSIGNMENT\_5” which includes for each question the `AxQy` script *as well as* any MATLAB functions (of your own creation) which are called by your script. Note the MATLAB functions of your own creation should include both specific MATLAB functions requested in the question statement as well as any other MATLAB functions (called directly or indirectly by your script or by other functions) which you choose to develop as part of your answer. Both the scripts and (requested) functions must conform to the formats described in **Instructions** and **Questions** below. You should also include in your folder all the `grade_o_matic` `.p` files for Assignment 5.

---

### Instructions

Before embarking on this assignment you should

- (1) Complete the Textbook reading for Unit V and review the Lecture Notes for Unit V. You should also review the Lecture Notes for Unit IV on Eigen-problems (relevant to Question 1).
- (2) Execute (“cell-by-cell”) the MATLAB Tutorial(s) for Unit V: MATLAB Sparse Matrix Operations; MATLAB Sparse Backslash Operator. Note also that the Textbook Chapter 28 addresses MATLAB issues (such as “declared sparse”) relevant to Assignment 5.
- (3) Download the `Assignment_5_Materials` folder. This folder contains a template for the script associated with each question (`A5Qy_Template` for Question `y`), as well as a template for each function which we ask you to create (`func_Template` for a function `func`). The `Assignment_5_Materials` folder also contains the `grade_o_matic` codes needed for Assignment 5. (Please see Assignment 1 for a description of `grade_o_matic`.)

We indicate here several general format and performance requirements:

- (a.) Your script for Question `y` of Assignment `x` *must* be a proper MATLAB “.m” script file and *must* be named `AxQy.m` (or, if explicitly indicated as such for a particular question, `AxQy.p`). In some cases the script will be trivial and you may submit the template “as is” — just remove the `_Template` — in your “YOURNAME\_ASSIGNMENT\_5” folder. But note that you still must submit a proper `AxQy.m` script (or, if explicitly indicated as such for a particular question, `AxQy.p` script) or `grade_o_matic_A5` will not perform correctly.
- (b.) In this assignment, for each question `y`, we will specify inputs and outputs both for the script `A5Qy` and (as is more traditional) any requested MATLAB functions; we shall denote the former as script inputs and script outputs and the latter as function inputs and function outputs. For each question and hence each script, and also each function, we will identify *allowable instances* for the inputs — the parameter values or “parameter domains” for which the codes must work.

- (c.) Recall that for scripts, input variables must be assigned *outside* your script (of course before the script is executed) — *not* inside your script — in the workspace; all other variables required by the script must be defined *inside* the script. Hence you should test your scripts in the following fashion: `clear` the workspace; assign the input variables in the workspace; run your script. Note to test your MATLAB functions you need not take such precautions: all inputs and outputs are passed through the input and output argument lists; a function enjoys a private workspace.
- (d.) We ask that you not end any of your script names or function names with the suffix `_ref` in order to prevent conflicts with scripts or functions provided in the folder `Assignment_5_Materials` (and needed by `grade_o_matic_A5`).
- (e.) We ask that in the submitted version of your scripts and functions you suppress all display by placing a “;” at the end of each line of code. (Of course during debugging you will often choose to display many intermediate and final results.) We also require that **before** you upload your solution you should run `grade_o_matic_A5` (from your `YOURNAME_ASSIGNMENT_5` folder) for final confirmation that all is in order.

## Questions

- (40 points: 10 points for the correct entries for `B` but also `B` must be “declared sparse” (or no credit); 10 points for `omega_min`; 10 points for `chi`; 10 points for `num_iter`.)

We consider in this question a string of length  $L$  and mass per unit length  $m'$  under tension  $T$  which is furthermore attached to “ground” at  $x = L/2$  to a Hookean spring with spring constant  $k_{\text{mid}}$ . We discretize the string as  $n$  masses  $\text{mass}_i, 1 \leq i \leq n$ , each of mass  $m'h$ , located at respective  $x$  locations  $x_i = i \cdot h, 1 \leq i \leq n$ . Here  $h = L/(n + 1)$  is the length of the string segment between successive masses (and from the left wall at  $x = 0$  to the first mass, and from the last mass to the right wall at  $x = L$ ). The  $y$  (vertical) displacement of  $\text{mass}_i$  is given by  $u_i, 1 \leq i \leq n$ . We presume that  $n$  is chosen odd such that the Hookean spring is attached to  $\text{mass}_{n_{\text{mid}}}$  for  $n_{\text{mid}} = (n + 1)/2$ ; the force (in the  $y$  direction) exerted by the Hookean spring on  $\text{mass}_{n_{\text{mid}}}$  is given by  $-k_{\text{mid}} u_{n_{\text{mid}}}$ . We show in Figure 1 a string discretization for the particular case  $n = 5$ .<sup>1</sup>

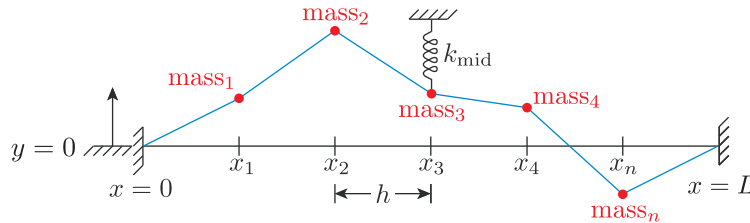


Figure 1: Discretization of the string for the particular case  $n = 5$ .

We shall consider here the case in which the string angles are small and furthermore there is *no* damping present in the system. It then follows from Newton’s second law applied in the  $y$

<sup>1</sup>You may assume that all quantities in this question are prescribed in kg-m-s units.

direction (and in the absence of applied forces) that the displacements of the masses satisfy<sup>2</sup>

$$\ddot{u}_i + \frac{\tau}{h^2}(-u_{i-1} + 2u_i - u_{i+1}) = 0 \quad \begin{cases} 2 \leq i \leq n_{\text{mid}} - 1 \\ n_{\text{mid}} + 1 \leq i \leq n \end{cases}, \quad (1)$$

$$\ddot{u}_i + \frac{\tau}{h^2}(2u_i - u_{i+1}) = 0 \quad i = 1, \quad (2)$$

$$\ddot{u}_i + \frac{\tau}{h^2}(-u_{i-1} + 2u_i) = 0 \quad i = n, \quad (3)$$

$$\ddot{u}_i + \alpha_1 u_{i-1} + \alpha_2 u_i + \alpha_3 u_{i+1} = 0 \quad i = n_{\text{mid}}, \quad (4)$$

where  $\tau \equiv T/m'$  and “dot” refers to differentiation in time (i.e.,  $\ddot{u}_i \equiv \frac{d^2 u_i}{dt^2}$ ); we also introduce  $\eta \equiv k_{\text{mid}}/m'$ . You will need to determine the constants  $\alpha_1, \alpha_2$ , and  $\alpha_3$  of (4) — which may depend on  $\tau, \eta, L$ , and  $h$  — in order to create the MATLAB function `string_frequency` we request below. (We recommend that you start with the equations in the form  $m'h\ddot{u}_i = \dots$  or you are likely to misplace a factor of  $h$ . Recall that  $m'$  is the mass *per unit length* of the string.) In practice we would also supplement these dynamical equations with appropriate initial conditions.

We next express in matrix form (with proper care for special case  $n = 1$ ) our system of  $n$  Equations (1)–(4) in the  $n$  unknowns  $u_i, 1 \leq i \leq n$ :

$$Bu = -\ddot{u}; \quad (5)$$

here  $B$  is an  $n \times n$  SPD matrix,  $u \equiv (u_1 \ u_2 \ \dots \ u_n)^T$  is the  $n \times 1$  vector of mass displacements, and  $\ddot{u} \equiv (\ddot{u}_1 \ \ddot{u}_2 \ \dots \ \ddot{u}_n)^T$  is the  $n \times 1$  vector of mass accelerations. You will need to determine the entries of  $B$  — which will depend on  $\tau, \eta, L$ , and  $h$  — from Equations (1)–(4) in order to create your MATLAB function `string_frequency`.

Finally, we now assume solutions of the form  $u = \chi e^{i\omega t}$  to arrive from (5) to an eigenproblem

$$B\chi = \lambda\chi, \quad (6)$$

for eigenvector  $\chi$  and eigenvalue  $\lambda \equiv \omega^2$ . (We recall that in the absence of dissipation  $u$  will oscillate indefinitely — without decay.) We may order the eigenvalues of (6) as  $0 < \lambda_{\min} \equiv \lambda_1 \leq \lambda_2 \leq \dots \leq \lambda_n$  with respective eigenvectors  $\chi_1, \chi_2, \dots, \chi_n$ . Our interest is in  $\omega_{\min} \equiv \sqrt{\lambda_{\min}}$ , the lowest natural frequency of our string system.

Towards that end we will ask you to write a function `string_frequency` with signature

```
function [B,omega_min,chi_1,num_iter] = ...
    string_frequency(tau,eta,L,n,epsilon,chi_init,lam_init)
```

which applies the inverse power iteration Algorithm 2 of Slide 37 of Lecture Notes 17 to (6) in order to find  $\lambda_{\min}$  and hence  $\omega_{\min}$ . Note that  $\omega_{\min}$  will depend on  $\tau, \eta, L$ , and  $h$ ;  $\omega$  will constitute a good approximation to the lowest natural frequency of

---

<sup>2</sup>Note for the case in which  $n = 1$  we consider only Equation (4) and we discard the other three equations; for the case in which  $n = 3$  we consider only Equations (2), (3), and (4) and we discard (1).

the (continuous) string system only for a sufficiently fine discretization —  $h$  sufficiently small and hence  $n$  (MATLAB  $n$ ) sufficiently large. Nevertheless, your code should perform correctly for any finite positive integer  $n$ .

Your MATLAB function must be named `string_frequency` and furthermore must be stored in a file named `string_frequency.m`. The function takes seven function inputs. The first input is the scalar `tau` which corresponds to  $\tau$ ; the set of allowable instances is  $1 \leq \tau \leq 100$ . The second input is the scalar `eta` which corresponds to  $\eta$ ; the set of allowable instances is  $0 \leq \eta \leq 1000$ . The third input is the scalar `L` which corresponds to  $L$ ; the set of allowable instances is  $.3 \leq L \leq 1.5$ . The fourth input is the scalar `n` which corresponds to  $n$ ; the set of allowable instances is  $1 \leq n \leq 10000$  for  $n$  odd. The fifth input is the scalar `epsilon` which corresponds to  $\varepsilon$  of the inverse power iteration Algorithm 2 and constitutes our error tolerance; the set of allowable instances is  $1 \times 10^{-6} \leq \varepsilon \leq .01$ . The sixth input is the  $n \times 1$  vector `chi_init` which is the initial guess for the eigenvector and corresponds to  $\hat{\chi}$  prescribed in Algorithm 2 prior to entering the `while` loop; there is no restriction on this input. Note, however, that, for testing purposes, setting `chi_init` to `zeros(n, 1)` will cause Algorithm 2 to return a trivial solution. The seventh and final input is the scalar `lam_init` which is the initial guess for the eigenvalue and corresponds to  $\hat{\lambda}$  prescribed in Algorithm 2 prior to entering the `while` loop. The function yields four function outputs. The first output is the  $n \times n$  matrix `B` which corresponds to  $B$ ; note the output `B` must be “declared sparse.” The second output is the real scalar `omega_min` which corresponds to  $\omega_{\min}$  (more precisely, the square root of  $\hat{\lambda}$  for  $\hat{\lambda}$  provided by Algorithm 2 upon exit from the `while` statement). The third output is `chi_1` which corresponds to the eigenvector  $\chi_1$  (more precisely,  $\hat{\chi}$  provided by Algorithm 2 upon exit from the `while` statement). The fourth output is `num_iter` which is the number of iterations — the number of times through the `while` loop of Algorithm 2, or equivalently the number of MATLAB backslash operations — required by your inverse power iteration to achieve the desired tolerance.

The script for this question is provided in `A5Q1.p`; you should not modify it in any way. We also provide a template for your inverse iteration function, `string_frequency_Template`. You should include both `A5Q1.p` and `string_frequency.m` in the `YOURNAME_ASSIGNMENT_5` folder you upload.

**Hints and Guidelines.** As always you should test your code thoroughly and in particular you should not rely on the few student instances of `grade_o_matic_A5` to confirm that your code is performing correctly in all allowable instances. You can inspect `B` “by hand” (for smaller  $n$ ). And you can compare your prediction for `omega_min` (and `chi_1`) to the results provided by MATLAB `eig` (for  $n$  not too large<sup>3</sup>). More ambitiously you can develop some simple closed-form expressions for  $\omega_{\min}$  and  $\chi_1$  in certain limits — see the Challenge below for several suggestions.

**Challenge.** Develop simple closed-form expressions for  $\omega_{\min}$  and  $\chi_1$  in the following cases:  $n = 1$  (not too difficult);  $\eta = 0$  and  $h \rightarrow 0$  (the first natural frequency of a “standard” (undamped) string in tension);  $\tau/(L\eta) \ll 1$  and  $h \rightarrow 0$  (a bit trickier — inspection of `chi_hat` from your code might provide a hint).

---

<sup>3</sup>For larger  $n$  the MATLAB function `eig` will be too slow because `eig` determines all  $n$  eigenvalues of `B`. The MATLAB function `eigs` is much more efficient for just a few (e.g., the smallest or largest) eigenvalues; do `help eigs` if you wish to (electively) add `eigs` to your MATLAB repertoire.

2. (15 points) *Preamble: You are not to use MATLAB for this question (except of course for the multiple-choice script file `AxQy.m` for `grade_o_matic_A5`) either to identify or confirm the correct result — you should develop your answer without recourse to a computer or even a calculator. The point of this question is to make sure that you understand the basic matrix operations.*

We consider in this problem the system of linear equations

$$Au = f \tag{7}$$

where  $A$  is a given  $3 \times 3$  matrix,  $f$  is a given  $3 \times 1$  vector, and  $u$  is the  $3 \times 1$  vector we wish to find.

We introduce two matrices

$$A^I = \begin{pmatrix} 1 & -1 & 0 \\ 1 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}, \tag{8}$$

and

$$A^{II} = \begin{pmatrix} 1 & -1 & 1 \\ 1 & 1 & -1 \\ 0 & 0 & 0 \end{pmatrix}, \tag{9}$$

which will be relevant in Parts (i),(ii) and Parts (iii),(iv) respectively.

In Parts (i),(ii),  $A$  of equation (7) is given by  $A^I$  of equation (8). In other words, we consider the system  $A^I u = f$  given by

$$\begin{matrix} \begin{pmatrix} 1 & -1 & 0 \\ 1 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} & \begin{pmatrix} u_1 \\ u_2 \\ u_3 \end{pmatrix} & = & \begin{pmatrix} f_1 \\ f_2 \\ f_3 \end{pmatrix}, \\ A^I & u & & f \end{matrix}$$

for  $f$  to be specified below.

- (i) (3.75 points) For  $f = (1 \ 1 \ 1)^T$  (recall  $T$  denotes transpose),

(a)  $A^I u = f$  has a unique solution

(b)  $A^I u = f$  has no solution

(c)  $A^I u = f$  has an infinity of solutions of the form

$$u = \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix} + \alpha \begin{pmatrix} 0 \\ 1 \\ 1 \end{pmatrix}$$

for any (real number)  $\alpha$

(d)  $A^I u = f$  has an infinity of solutions of the form

$$u = \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix} + \alpha \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}$$

for any (real number)  $\alpha$ .

(ii) (3.75 points) For  $f = (1 \ 1 \ 0)^T$ ,

(a)  $A^I u = f$  has a unique solution

(b)  $A^I u = f$  has no solution

(c)  $A^I u = f$  has an infinity of solutions of the form

$$u = \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix} + \alpha \begin{pmatrix} 0 \\ 1 \\ 1 \end{pmatrix}$$

for any (real number)  $\alpha$

(d)  $A^I u = f$  has an infinity of solutions of the form

$$u = \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix} + \alpha \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}$$

for any (real number)  $\alpha$ .

Now, in Parts (iii), (iv),  $A$  of equation (7) is given by  $A^{\text{II}}$  of equation (9). In other words, we consider the system  $A^{\text{II}} u = f$  given by

$$\begin{pmatrix} 1 & -1 & 1 \\ 1 & 1 & -1 \\ 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} u_1 \\ u_2 \\ u_3 \end{pmatrix} = \begin{pmatrix} f_1 \\ f_2 \\ f_3 \end{pmatrix},$$

$A^{\text{II}} \qquad u \qquad f$

for  $f$  to be specified below.

(iii) (3.75 points) For  $f = (1 \ 1 \ 1)^T$  (recall  $^T$  denotes transpose),

(a)  $A^{\text{II}} u = f$  has a unique solution

(b)  $A^{\text{II}} u = f$  has no solution

(c)  $A^{\text{II}}u = f$  has an infinity of solutions of the form

$$u = \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix} + \alpha \begin{pmatrix} 0 \\ 1 \\ 1 \end{pmatrix}$$

for any (real number)  $\alpha$

(d)  $A^{\text{II}}u = f$  has an infinity of solutions of the form

$$u = \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix} + \alpha \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}$$

for any (real number)  $\alpha$ .

(iv) (3.75 points) For  $f = (1 \ 1 \ 0)^{\text{T}}$ ,

(a)  $A^{\text{II}}u = f$  has a *unique solution*

(b)  $A^{\text{II}}u = f$  has *no solution*

(c)  $A^{\text{II}}u = f$  has an infinity of solutions of the form

$$u = \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix} + \alpha \begin{pmatrix} 0 \\ 1 \\ 1 \end{pmatrix}$$

for any (real number)  $\alpha$

(d)  $A^{\text{II}}u = f$  has an infinity of solutions of the form

$$u = \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix} + \alpha \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}$$

for any (real number)  $\alpha$ .

3. (15 points) *Preamble: You are not to use MATLAB for this question (except of course for the multiple-choice script file `AxQy.m` for `grade_o_matic_A5`) either to identify or confirm the correct result — you should develop your answer without recourse to a computer or even a calculator. The point of this question is to make sure that you understand the basic matrix operations.*

We consider the system of three springs and masses shown in Figure 2.

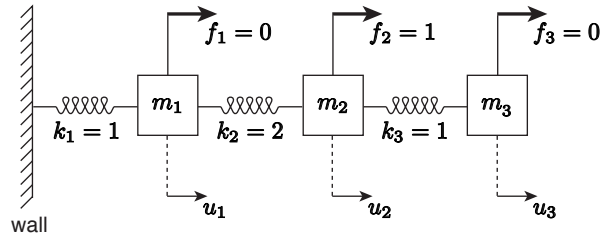


Figure 2: The spring-mass system for Question 3.

The equilibrium displacements satisfy the linear system of three equations in three unknowns,  $Au = f$ , given by

$$\begin{pmatrix} 3 & -2 & 0 \\ -2 & 3 & -1 \\ 0 & -1 & 1 \end{pmatrix} \begin{pmatrix} u_1 \\ u_2 \\ u_3 \end{pmatrix} = \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix}. \quad (10)$$

$A \qquad u \qquad f$

The matrix  $A$  is SPD (*Symmetric Positive Definite*). Note you should only consider the particular right-hand side  $f$  (forces) indicated.

We now reduce the system by Gaussian Elimination to form  $Uu = \hat{f}$ , where  $U$  is an upper triangular matrix. We may then find  $u$  by Back Substitution. Note that we do *not* perform any partial pivoting — reordering of the rows of  $A$  — for stability (since the matrix is SPD there is no need), and furthermore we do *not* perform any reordering of the columns of the matrix  $A$  for optimization: we work directly on the matrix  $A$  as given by equation (10).

(i) (3 points) The element  $U_{22}$  (i.e., the entry in the  $i =$  second row,  $j =$  second column) of  $U$  is given by

(a) 3

(b)  $13/3$

(c)  $5/3$

(d)  $2/3$

(ii) (3 points) The element  $U_{33}$  (i.e., the entry in the  $i =$  third row,  $j =$  third column) of  $U$  is given by

(a)  $-1$

(b)  $2/5$

(c) 1

(d)  $3/5$



(iii) (3 points) The element  $\hat{f}_2$  (i.e., the second entry in the  $\hat{f}$  vector) is given by

(a)  $2/3$

(b)  $1$

(c)  $0$

(d)  $5/3$

(iv) (3 points) The element  $\hat{f}_3$  (i.e., the third entry in the  $\hat{f}$  vector) is given by

(a)  $-3/5$

(b)  $0$

(c)  $3/5$

(d)  $8/5$

(v) (3 points) The displacement of the third mass,  $u_3$ , is given by

(a)  $5/2$

(b)  $-2/3$

(c)  $3/2$

(d)  $2/3$

4. (15 points) *Preamble: You are not to use MATLAB for this question (except of course for the multiple-choice script file `AxQy.m` for `grade_o_matic_A5`) either to identify or confirm the correct result — you should develop your answer without recourse to a computer or even a calculator. The point of this question is to make sure that you understand the basic matrix operations.*

We consider the system of springs and masses shown in Figure 3. Equilibrium — force balance on each mass and Hooke's law for the spring constitutive relation — leads to the system of

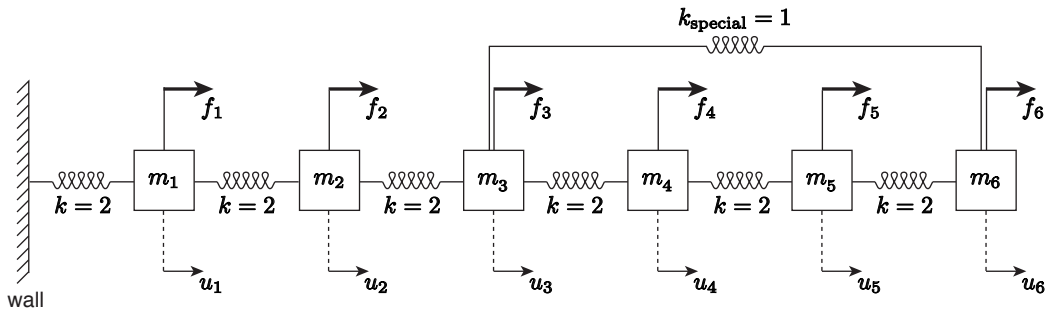


Figure 3: The spring-mass system for Question 4. All the spring constants are two ( $k = 2$ ), except for the “special” spring which links mass 3 and mass 6 ( $k_{\text{special}} = 1$ ); you may assume that all quantities are provided in consistent units. Note that all the springs are described by the linear Hooke relation.

six equations in six unknowns,  $Au = f$ ,

$$\begin{pmatrix} 4 & -2 & 0 & 0 & 0 & 0 \\ -2 & 4 & -2 & 0 & 0 & 0 \\ 0 & -2 & a & -2 & 0 & c \\ 0 & 0 & -2 & 4 & -2 & 0 \\ 0 & 0 & 0 & -2 & 4 & -2 \\ 0 & 0 & c & 0 & -2 & b \end{pmatrix} \begin{pmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \\ u_5 \\ u_6 \end{pmatrix} = \begin{pmatrix} f_1 \\ f_2 \\ f_3 \\ f_4 \\ f_5 \\ f_6 \end{pmatrix}. \quad (11)$$

$A$   $u$   $f$

where we will ask you to specify  $a$ ,  $b$ , and  $c$  in the questions below. Note for the correct choices of  $a$ ,  $b$ , and  $c$  the matrix  $A$  is SPD.

We now reduce the system  $Au = f$  by Gaussian Elimination to form  $Uu = \hat{f}$ , where  $U$  is an upper triangular matrix. Note that we do *not* perform any partial pivoting — reordering of the rows of  $A$  — for stability (since the matrix is SPD there is no need), and furthermore we do *not* perform any reordering of the columns of the matrix  $A$  for optimization: we work directly on the matrix  $A$  as given by equation (11). Recall that since  $A$  is SPD we are sure that we will *not* encounter a zero pivot.

(i) (3 points) The value of  $a$  is

(a) 1

(b) 2

(c) 3

(d) 4

(e) 5

(ii) (3 points) The value of  $b$  is

(a) 1

(b) 2

(c) 3

(d) 4

(e) 5

(iii) (3 points) The value of  $c$  is

(a) -1

(b) -2

(c) -3

(d) -4

(e) -5

(iv) (3 points) The number of *nonzero* elements in the (upper triangular) matrix  $U$  is

(a) 36

(b) 13

(c) 6

(d) 21

(e) 15

*Hint:* Consider Gaussian Elimination (and the fill-in process) to deduce the only possibly correct option from the available choices.

(v) (3 points) The entry  $(A^{-1})_{66}$ , (i.e., the entry in the  $i =$  sixth row,  $j =$  sixth column of the inverse matrix of  $A$ ) is

(a)  $1/A_{66}$

(b)  $1/U_{66}$

(c)  $A_{66}$

(d)  $U_{66}$

(e)  $(f_1 + f_2 + f_3 + f_4 + f_5 + f_6)/U_{66}$

where in each case subscript  $_{66}$  refers to the entry in the  $i =$  sixth row,  $j =$  sixth column.

*Hint:* Recall the physical interpretation of the sixth column of  $A^{-1}$ .

5. (15 points) *Preamble: You should develop your responses based on theoretical considerations. However, you may use MATLAB to motivate or confirm (or perhaps rectify) your theoretical predictions.*

We consider the system of  $n$  springs and masses shown in Figure 4. We consider the particular case in which  $k_i = 1, 1 \leq i \leq n$ , and  $f_i = 1, 1 \leq i \leq n$ ; you may assume that all quantities are

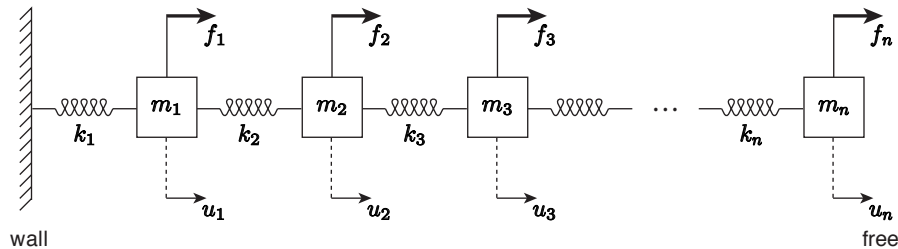


Figure 4: The spring-mass system for Question 5.

provided in consistent units. The displacements of the masses,  $u$ , satisfies a linear system of  $n$  equations in  $n$  unknowns,  $Au = f$ . The MATLAB script provided below forms the stiffness matrix  $A$  ( $= \mathbf{A}$  in MATLAB) and force vector  $f$  ( $= \mathbf{f}$  in MATLAB) and then solves for the displacements  $u$  ( $= \mathbf{u}$  in MATLAB) in three different fashions. You may assume that prior to execution of the script the workspace contains only  $n$  (MATLAB  $\mathbf{n}$ ) which is a positive integer scalar.

```
% begin script
```

```
% form A and f
```

```
A = spalloc(n,n,3*n);
```

```
A(1,1) = 2;
```

```
A(1,2) = -1;
```

```
for i = 2:n-1
```

```
    A(i,i) = 2;
```

```
    A(i,i-1) = -1;
```

```
    A(i,i+1) = -1;
```

```
end
```

```
A(n,n) = 1;
```

```
A(n,n-1) = -1;
```

```

numnonzero_of_A = nnz(A);

f = ones(n,1);

% solve A u = f in three different ways

numtimes_compute = 20;

tic
for itimes = 1:numtimes_compute
    u = A\f; % REFER TO THIS LINE in Question 5(ii)
end
avg_time_first_way = toc/numtimes_compute;

A_declare_full = full(A);
tic
for itimes = 1:numtimes_compute
    u = A_declare_full\f;
end
avg_time_second_way = toc/numtimes_compute;

Ainv_declare_sparse = sparse(inv(A));
% in fact if A is "declared sparse" then inv(A) will automatically be "declared sparse"
tic
for itimes = 1:numtimes_compute
    u = Ainv_declare_sparse*f;
end
avg_time_third_way = toc/numtimes_compute;
% note that we do not include the time to compute inv(A) in avg_time_third_way

% end script

```

The MATLAB backslash operator will not perform any partial pivoting — reordering of the rows of  $A$  — for stability (since the matrix is SPD there is no need); furthermore MATLAB will not perform any reordering of the columns of the matrix  $A$  for efficiency (since the matrix is tri-diagonal the structure is already optimal). In short, MATLAB backslash works directly on the matrix  $A$  as given — Gaussian Elimination to obtain  $U$  ( $= U$  in MATLAB) and  $\hat{f}$  followed by Back Substitution to obtain  $u$ .

We now run the script. In the questions below you should assume that the computational time to perform the operations is proportional to the number of FLOPs. (In actual practice, computational time and FLOPs is not synonymous since the former is affected by memory access, competition for cores, network speed, and other “real-life” considerations; furthermore, these “real-life” considerations become more important for the larger  $n$  of interest in this question. Inasmuch, your computational times should serve to guide, but not dictate, your answers.)

(i) (3 points) For  $n = 5000$  the script will set the value of `numnonzero_of_A` to

(a) 25,000,000

(b) 14,998

(c) 30,000

(d) 5,000

(ii) (3 points) For  $n=5000$  the number of nonzero elements of  $U$  will be

(a) 12,507,501

(b) 15,000

(c) 5,000

(d) 9,999

Note you do not see  $U$  explicitly in the script of the previous page. Here  $U$  is the upper triangular matrix  $U$  formed internally as part of the backslash operation `u = A\b` on the line of the script with comment `% REFER TO THIS LINE in Question 5(ii)` .

*Hint:* Recall Gaussian Elimination for tridiagonal matrices.

(iii) (3 points) The ratio

$$\frac{\text{avg\_time\_second\_way}}{\text{avg\_time\_first\_way}}$$

will behave asymptotically as  $Cn^\rho$  for  $n \rightarrow \infty$  for  $C$  a constant independent of  $n$  and  $\rho$  given by

(a) -3

(b) -2

(c) -1

(d) 0

(e) 1

(f) 2

(g) 3

(iv) (3 points) The ratio

$$\frac{\text{avg\_time\_third\_way}}{\text{avg\_time\_first\_way}}$$

will behave asymptotically as  $Cn^\rho$  for  $n \rightarrow \infty$  for  $C$  a constant independent of  $n$  and  $\rho$  given by

(a) -3

(b) -2

(c) -1

(d) 0

(e) 1

(f) 2

(g) 3

(v) (3 points) The ratio

$$\frac{\text{avg\_time\_third\_way}}{\text{avg\_time\_second\_way}}$$

will behave asymptotically as  $Cn^\rho$  for  $n \rightarrow \infty$  for  $C$  a constant independent of  $n$  and  $\rho$  given by

(a) -3

(b) -2

(c) -1

(d) 0

(e) 1

(f) 2

(g) 3

MIT OpenCourseWare  
<http://ocw.mit.edu>

2.086 Numerical Computation for Mechanical Engineers  
Spring 2013

For information about citing these materials or our Terms of Use, visit: <http://ocw.mit.edu/terms>.