Matlab Exercises_Recitation 12 2.086 Spring 2012

Recitation 12: Wednesday, 2 May / Friday, 4 May
MATLAB Exercises_Recitation 12 due: *Monday, 7 May 2012, at 5 PM by upload to Stellar*

Format for upload: Students should upload to the course Stellar website a folder

`YOURNAME_MatlabExercises_Rec12`

which contains the completed scripts and functions for the assigned MATLAB Exercises_Recitation 12: all the scripts should be in a single file, with each script preceded by a comment line which indicates the exercise number; each function `.m` file should contain a comment line which indicates the exercise number.

---

1. In this question we ask you to time the solution of a sparse tri-diagonal system.

   (*a*) Create a function `timer_bslash_sparse` as a slight modification to your function `timer_matvec_sparse` of Recitation 11: replace the line `v = K*w` in the latter with `u = K\f` in the former.

   (*b*) Write a three-line script which invokes `timer_bslash_sparse` (three times) to display `avg_time/n` for `n` = 3,200, `n` = 6,400, and `n` = 12,800, and `numrepeats` = 100. (Note you will need to make sure you copy your function `generate_K` from Recitation 11 to the directory/folder from which you run `timer_bslash_sparse`.)

   You should observe that `avg_time/n` is roughly constant and thus conclude that the time required to perform a sparse tridiagonal solve increases only linearly with `n`.

2. In this question we ask you to demonstrate the advantage of sparse storage format by re-performing the timings of Question 1 but now for `K` converted to (and stored in) non-sparse storage format.

   (*a*) Create a function `timer_bslash_full` as a slight modification to your function `timer_matvec_full` of Recitation 11: replace the line `v = K*w` in the latter with `u = K\f` in the former.

   (*b*) Write a three-line script which invokes `timer_bslash_full` (three times) to display `avg_time/n^3` for `n` = 400, `n` = 800, and `n` = 1,600, and `numrepeats` = 100. (Note you will need to make sure you copy your function `generate_K` from Recitation 11 to the directory/folder from which you run `timer_bslash_full`.)

   You should observe that `avg_time/n^3` is roughly constant and thus conclude that *if your tridiagonal matrix is not stored in sparse format (i.e., recognized by* MATLAB *as sparse)*, the time required to perform a tridiagonal solve increases cubically with `n` — the same operation count we would expect if `K` was a fully populated (dense) matrix.

3. Consider the spring system shown in Figure 1 in which we take a standard "series" configuration of n springs (spring constants k_i, $1 \le i \le n$) but then add an additional spring (spring constant k_{special}) which connects the first and last mass. The equilibrium displacement u for given applied forces f is governed by the system of n equations in n unknowns $Ku = f$.

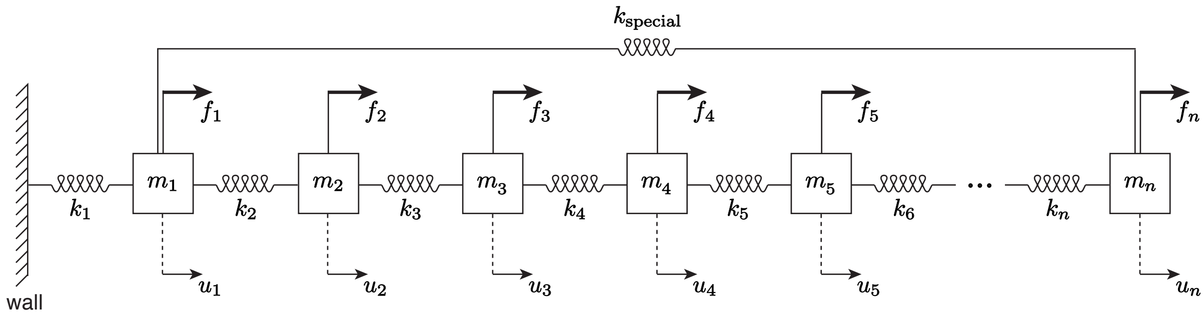

Figure 1: The spring-mass system for Question 3.

(*a*) Create a function

```
function [ K ] = generate_special_K(n,kvec,k_special)
```

which yields as output (in *sparse storage format*) the stiffness matrix K (MATLAB `K`) for given `n`, `kvec`$(i) = k_i$, $1 \le i \le n$, and `k_special` $= k_{\text{special}}$.

(*b*) Create a function `timer_bslash_special_K` as a slight modification to your function `timer_bslash_sparse` of Question 1: replace the call to `generate_K(n,kvec)` in the latter with a call to `generate_special_K(n,kvec,1)` in the former.

(*c*) Write a three-line script which invokes `timer_bslash_special_K` (three times) to display `avg_time/n` for `n` = 3,200, `n` = 6,400, and `n` = 12,800, and `numrepeats` = 100.

You should observe that `avg_time/n` is roughly constant and thus conclude that the time required to perform this sparse solve increases only linearly with `n`. This is an example of a sparse matrix for which Gaussian elimination creates relatively little "fill-in."