Assignment 5 — 2.086 Fall 2014

Due: *Wednesday, 10 December at 5 PM.*

Upload your solution to course website as a zip file "`YOURNAME_ASSIGNMENT_5`" which includes the script for each question *as well as* all MATLAB® functions and scripts (of your own creation) called by the scripts; both scripts and functions must conform to the formats described in **Instructions** and **Questions** below.

Instructions

Download (from the course website Assignment 5 page) the `Assignment_5_Templates` 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_Templates` folder also contains the `grade_o_matic` files for Assignment 5 (please see Assignment 1 for a description of `grade_o_matic`[1]) as well as all auxiliary files which you will need for Assignment 5.

We indicate here several general format and performance requirements:

(a.) Your script for Question `y` of Assignment `5` *must* be a proper MATLAB ".m" script file and *must* be named `A5Qy.m`. 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 `A5Qy.m` script or `grade_o_matic` 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 for 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 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 to course website you run `grade_o_matic` (from your YOURNAME_ASSIGNMENT_5 folder) for final confirmation that all of your scripts and functions are in the proper format.

Note that, in Assignment 5, four of the five questions, `A5Qy` for `y` = 1,2,3,4, are **multiple choice**: you should enter your answers in the respective scripts `A5Qy.m` per standard `grade_o_matic` pro-

[1] Note that, for display in verbose mode, `grade_o_matic` will "unroll" arrays and present as a row vector.

tocol; please double-check the "letters" you have chosen for each question before uploading your `YOURNAME_ASSIGNMENT_5` folder. Note that the last question, `A5Q5`, is not multiple choice.

Questions

1. (*20 points*) *Preamble: You are not to use* MATLAB *for this question (except of course for the multiple-choice script file* `A5Q1.m` *for* `grade_o_matic`*) 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 linear algebra.*

   We consider in this problem the system of linear equations

$$Au = f \tag{1}$$

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

   We introduce two matrices

$$A^{\mathrm{I}} = \begin{pmatrix} 1 & -1 & 0 \\ 1 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}, \tag{2}$$

   and

$$A^{\mathrm{II}} = \begin{pmatrix} 1 & -1 & -1 \\ 1 & 1 & -1 \\ 0 & 0 & 0 \end{pmatrix}, \tag{3}$$

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

   In Parts (i),(ii), A of equation (1) is given by A^{I} of equation (2). In other words, we consider the system $A^{\mathrm{I}}u = f$ given by

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

   for f to be specified below.

   (i) (5 points) For $f = (1 \quad 1 \quad 1)^{\mathrm{T}}$ (recall $^{\mathrm{T}}$ denotes transpose),

      (a) $A^{\mathrm{I}}u = f$ has *a unique solution*.

      (b) $A^{\mathrm{I}}u = f$ has *no solution*.

(*c*) $A^{\mathrm{I}}u = f$ has an infinity of solutions of the form

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

for any (real number) α.

(*d*) $A^{\mathrm{I}}u = f$ has an infinity of solutions of the form

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

for any (real number) α.

(*ii*) (5 points) For $f = (1 \quad 1 \quad 0)^{\mathrm{T}}$,

(*a*) $A^{\mathrm{I}}u = f$ has *a unique solution*.

(*b*) $A^{\mathrm{I}}u = f$ has *no solution*.

(*c*) $A^{\mathrm{I}}u = f$ has an infinity of solutions of the form

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

for any (real number) α.

(*d*) $A^{\mathrm{I}}u = f$ has an infinity of solutions of the form

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

for any (real number) α.

Now, in Parts (*iii*), (*iv*), A of equation (1) is given by A^{II} of equation (3). In other words, we consider the system $A^{\mathrm{II}}u = f$ given by

$$\underbrace{\begin{pmatrix} 1 & -1 & -1 \\ 1 & 1 & -1 \\ 0 & 0 & 0 \end{pmatrix}}_{A^{\mathrm{II}}} \underbrace{\begin{pmatrix} u_1 \\ u_2 \\ u_3 \end{pmatrix}}_{u} = \underbrace{\begin{pmatrix} f_1 \\ f_2 \\ f_3 \end{pmatrix}}_{f},$$

for f to be specified below.

(*iii*) (5 points) For $f = (1 \quad 1 \quad 1)^{\mathrm{T}}$ (recall $^{\mathrm{T}}$ denotes transpose),

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

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

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

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

for any (real number) α.

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

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

for any (real number) α.

(*iv*) (5 points) For $f = (1 \quad 1 \quad 0)^{\mathrm{T}}$,

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

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

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

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

for any (real number) α.

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

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

for any (real number) α.

There are no inputs or outputs for this question. Your answer should be a one-line script, with your multiple-choice answers, as indicated in `A5Q1_Template.m` (as always, remove the `_Template` before uploading to course website in your folder `YOURNAME_ASSIGNMENT_5`).

2. (*20 points*) *Preamble: You are not to use* MATLAB *for this question (except of course for the multiple-choice script file* `A5Q2.m` *for* `grade_o_matic`*) 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 linear algebra.*

We consider the system of three springs and masses shown in Figure 1.

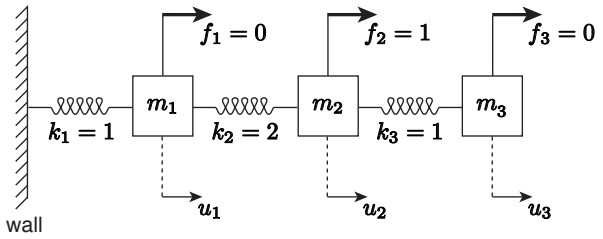

Figure 1: The spring-mass system for Question 2.

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

$$\underbrace{\begin{pmatrix} 3 & -2 & 0 \\ -2 & 3 & -1 \\ 0 & -1 & 1 \end{pmatrix}}_{A} \underbrace{\begin{pmatrix} u_1 \\ u_2 \\ u_3 \end{pmatrix}}_{u} = \underbrace{\begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix}}_{f}. \tag{4}$$

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 the 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 (4).

(i) (4 points) The element $U_{2\,2}$ (i.e., the entry in the i = second row, j = second column) of U is given by

(a) 2/3

(b) 3

(c) 13/3

(d) 5/3

(*ii*) (4 points) The element $U_{2\,3}$ (i.e., the entry in the i = second row, j = third column) of U is given by

(a) -1

(b) 1

(c) 2/5

(d) 0

(*iii*) (4 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*) (4 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*) (4 points) The displacement of the third mass, u_3, is given by

(a) 3/2

(b) 2/3

(c) 5/2

(d) $-2/3$

There are no inputs or outputs for this question. Your answer should be a one-line script, with your multiple-choice answers, as indicated in `A5Q2_Template.m` (as always, remove the `_Template` before uploading to course website in your folder `YOURNAME_ASSIGNMENT_5`).

3. (*20 points*) *Preamble: You are not to use* MATLAB *for this question (except of course for the multiple-choice script file* `A5Q3.m` *for* `grade_o_matic`*) 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 linear algebra.

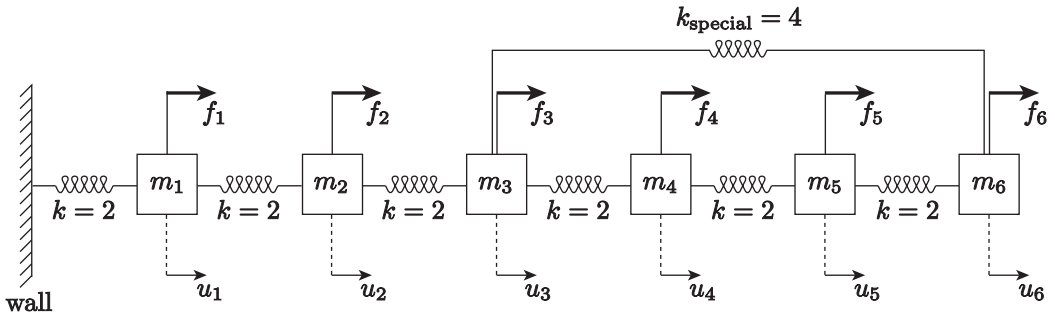

Figure 2: The spring-mass system for Question 3. For the six springs in series the spring constants are $k = 2$ whereas for the parallel "special" spring which directly links mass 3 and mass 6 the spring constant is $k_{\text{special}} = 4$; you may assume that all quantities are provided in consistent units. Note that all the springs are described by the linear Hooke relation.

We consider the system of springs and masses shown in Figure 2. Equilibrium — force balance on each mass and Hooke's law for the spring constitutive relation — leads to the system of six equations in six unknowns, $Au = f$,

$$\underbrace{\begin{pmatrix} 4 & -2 & 0 & 0 & 0 & 0 \\ -2 & 4 & -2 & 0 & 0 & 0 \\ 0 & -2 & a & -2 & 0 & c \\ 0 & 0 & -2 & 4 & -2 & 0 \\ 0 & 0 & 0 & -2 & 4 & -2 \\ 0 & 0 & c & 0 & -2 & b \end{pmatrix}}_{A} \underbrace{\begin{pmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \\ u_5 \\ u_6 \end{pmatrix}}_{u} = \underbrace{\begin{pmatrix} f_1 \\ f_2 \\ f_3 \\ f_4 \\ f_5 \\ f_6 \end{pmatrix}}_{f}; \tag{5}$$

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 (5). Recall that since A is SPD we are sure that we will *not* encounter a zero pivot.

(i) (4 points) The value of a is

(a) 2

(b) 4

(*c*) 6

(*d*) 8

(*e*) 12

(*ii*) (4 points) The value of b is

(*a*) 2

(*b*) 4

(*c*) 6

(*d*) 8

(*e*) 12

(*iii*) (4 points) The value of c is

(*a*) -2

(*b*) -4

(*c*) -6

(*d*) -8

(*e*) -12

(*iv*) (4 points) The number of *nonzero* elements in the (upper triangular) matrix U is

(*a*) 21

(*b*) 6

(*c*) 13

(*d*) 15

(*e*) 36

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

(*v*) (4 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*) $1/U_{66}$

(*b*) $1/A_{66}$

(*c*) U_{66}

(*d*) A_{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}.

There are no inputs or outputs for this question. Your answer should be a one-line script, with your multiple-choice answers, as indicated in `A5Q3_Template.m` (as always, remove the `_Template` before uploading to course website in your folder `YOURNAME_ASSIGNMENT_5`).

4. (*20 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 3. We consider the particular case in which $k_i = 1, 1 \le i \le n$, and $f_i = 1, 1 \le i \le n$; you may assume that all quantities are

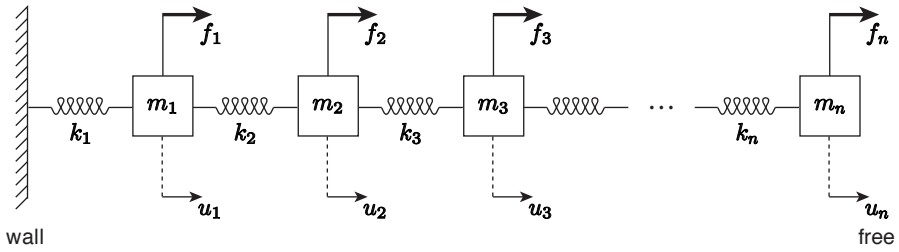

Figure 3: The spring-mass system for Question 4.

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 (= `A` in MATLAB) and force vector f (= `f` in MATLAB) and then solves for the displacements u (= `u` in MATLAB) in three different fashions. You may assume that prior to execution of the script the workspace contains only n (MATLAB `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 4(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) (5 points) For `n = 5000` the script will set the value of `numnonzero_of_A` to

(a) `14,998`

(b) `30,000`

(c) `5,000`

(d) `25,000,000`

(ii) (5 points) For `n = 5000` the number of nonzero elements of `U` will be

(a) `15,000`

(b) `12,507,501`

(c) `9,999`

(d) `5,000`

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\f` on the line of the script with comment `% REFER TO THIS LINE in Question 4(ii)`.

Hint: Recall Gaussian Elimination for tridiagonal matrices.

(iii) (5 points) The ratio

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

will behave asymptotically as Cn^{ρ} for $n \to \infty$, where C is a constant independent of n, and ρ is given by

(a) 3

(b) 2

(c) 1

(*d*) 0

(*e*) -1

(*f*) -2

(*g*) -3

Note we ask here for the value of ρ, not for the value of C.

(*iv*) (5 points) The ratio

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

will behave asymptotically as Cn^ρ for $n \to \infty$, where C is a constant independent of n, and ρ is given by

(*a*) 3

(*b*) 2

(*c*) 1

(*d*) 0

(*e*) -1

(*f*) -2

(*g*) -3

Note we ask here for the value of ρ, not for the value of C.

There are no inputs or outputs for this question. Your answer should be a one-line script, with your multiple-choice answers, as indicated in `A5Q4_Template.m` (as always, remove the `_Template` before uploading to course website in your folder `YOURNAME_ASSIGNMENT_5`).

5. (*20 points*) We would like you to write a MATLAB function with signature

```
function [root] = root_finder(alpha)
```

which, given a parameter α (MATLAB `alpha`) such that $12 \leq \alpha \leq 20$, finds a root z^* (MATLAB `root`) of a function $f_{\text{wiggly}}(z, \alpha)$ such that

$$f_{\text{wiggly}}(z^*; \alpha) = 0 \quad \textit{and} \quad 0 \leq z^* \leq 1 \; ; \tag{6}$$

the value of z^* will of course depend on the value of the parameter α. Two comments: we have chosen f_{wiggly} and the allowable instances of α to ensure that there will always be at

least one z^* which satisfies (6); we ask you only to find *a* root in the interval $[0, 1]$, not all roots in the interval $[0, 1]$.

Your function must be named `root_finder` and furthermore must be stored in a file named `root_finder.m`. Your function takes a single input: the value of the parameter α (MATLAB `alpha`); allowable instances must satisfy $12 \leq \alpha \leq 20$. Your function yields a single output: a z^* (MATLAB `root`) which satisfies a "numerical" version of (6),

$$|f_{\text{wiggly}}(z^*; \alpha)| \leq \text{1e-2} \quad \textit{and} \quad 0 \leq z^* \leq 1 \; . \tag{7}$$

We provide you with the MATLAB function `f_wiggly.p` with signature

```
function [f_value] = f_wiggly(z,alpha)
```

such that `f_wiggly(z,alpha)` returns the output `f_value` $= f_{\text{wiggly}}(\texttt{z}; \alpha)$. The MATLAB function `f_wiggly` will accept an input $1 \times M$ array `z` to yield output $1 \times M$ array `f_value` such that `f_value(i)` $= f_{\text{wiggly}}(\texttt{z(i)}, \alpha), i = 1, \ldots, M$; this could prove convenient if you wish (electively) to plot the z-dependence of f_{wiggly}. Note however, that input `alpha` must be a scalar.

We ask that your function `root_finder` call the MATLAB built-in nonlinear equation solver `fsolve`. Please read the Appendix to familiarize yourself with this MATLAB routine. Three comments: although `fsolve` will take care of all the details related to the solution procedure, you (in your `root_finder` function) must check and ultimately ensure (7); for reasons explained in the Appendix, your function `root_finder` will need to consider a sequence of initial guesses for `fsolve` to ensure that success — a z^* which satisfies (7) — is ultimately realized[2]; you should use the *default values* for the various `fsolve` tolerances.

We provide you with a script `A5Q5_Template.m` which you should not modify (but as always, please remove the `_Template` before uploading to course website). The deliverables for this question are the script `A5Q5.m` and most importantly your function `root_finder` (and any other scripts or functions of your own creation which are called by `root_finder`); please upload these deliverables to course website in your folder `YOURNAME_ASSIGNMENT_5`.

Appendix: MATLAB `fsolve`

We consider the problem of finding a real root (or "zero") of a univariate function: given a function $f(z), a \leq z \leq b$, we wish to find a real number z^* such that $f(z^*) = 0$; there could be no (real) roots, one root, or many roots. (The ostensibly more general problem of solving a nonlinear equation is in fact equivalent to the problem of finding a root.) Although in this assignment we consider only univariate functions, `fsolve` can readily treat the general multivariate case.

In the textbook we discuss Newton iteration for root finding. The MATLAB function `fsolve` implements an alternative, though often related, route: application of optimization procedures to a nonlinear least-squares re-formulation of the root-finding problem. It is simple to describe the framework. If $f(z^*) = 0$, then clearly $f^2(z^*) = 0$, and furthermore — since $f^2(z) \geq 0$ for all z — z^* is a minimizer of $f(z)$. Hence to find a root of $f(z)$ we can instead search for a minimizer of $f^2(z)$;

[2]For each input instance of the parameter α, `grade_o_matic` shall test your `root_finder` code 10 times (for this same value of α), to make sure that success is not a matter of good luck.

the latter is an easier problem than the former, as we can "simply" proceed downhill until we arrive at the minimum. There is one subtlety, however: though a root of $f(z)$ is necessarily a minimizer of $f^2(z)$, a minimizer of $f^2(z)$ is not necessarily a root of $f(z)$. A simple example of the latter is $f(z) = 1 + z^2$: $z = 0$ is clearly a minimizer of $f^2(z)$, but $f(0) = 1$, not zero, and hence $z = 0$ is not a root of $f(z)$. The approach must thus be formulated in two steps: in the first step we look for a minimizer z^{**} of $f^2(z)$; in the second step we evaluate $f^2(z^{**})$ and we accept z^{**} as a root — in other words, we identify $z^* = z^{**}$ — only if $f^2(z^{**}) = 0$ (or, in actual practice, sufficiently small).

The syntax for `fsolve` is very simple:

```
[zstarstar,fval,exitflag] = fsolve(@func, z_0)
```

where `zstarstar` is the alleged root, `fval` is the value of `func` at `zstarstar`, `exitflag` is an exit flag which provides information on the minimization process and alleged root, `func` is the MATLAB embodiment of the function for which we seek a root, and `z_0` is the initial guess. In more detail, `func` is a user-defined function with signature

```
function [func_value] = func(z)
```

which returns `func_value` $= f($`z`$)$ for given `z`.

We note that, as is typical in the (iterative) solution of nonlinear equations, `fsolve` — which effectively goes "downhill" to find a minimizer of $f^2(z)$ (or the norm squared of f in the case of a system of nonlinear equations) — will typically find the minimizer at the bottom of the valley to which the initial guess z_0 belongs. Thus, first, if the minimum of this valley does not constitute a root, then `fsolve` will not find a root, and second, in any event `fsolve` will only find the one root associated with the particular valley implicitly selected by z_0. In actual practice, the initial guess — *often several initial guesses will be required* — plays an important role in the solution of nonlinear equations. You can visualize this result with the small script `fsolve_demo_2014.m` included in the `Assignment_5_Templates` folder.

As already indicated, the output `zstarstar` is z^{**}, a minimizer of $f^2(z)$. In addition, `fsolve` indicates whether z^{**} corresponds to "Equation solved" in the sense that $f^2(z^{**})$ is zero or very small, or "No solution found" in the sense that $f^2(z^{**})$ is far from zero. You can and should also obtain this information yourself directly from `fval` (or `func(zstarzstar)`). In the "Equation solved" case — `fval` sufficiently close to zero — we have found a minimizer of $f^2(z)$ which is indeed a root of $f(z)$; in the "No solution found" case we have found an extremum of $f^2(z)$, typically though not always a minimum of $f^2(z)$, which is not a zero of $f(z)$ — `fval` is not close to zero.

MIT OpenCourseWare
http://ocw.mit.edu

2.086 Numerical Computation for Mechanical Engineers
Fall 2014

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