Selected Solutions for Exercises in

Numerical Methods with MATLAB: Implementations and Applications

Gerald W. Recktenwald

Chapter 3

MATLAB Programming

The latest version of this PDF file, along with other supplemental material for the book, can be found at `www.prenhall.com/recktenwald`.

3.3 Transform the `takeout` script in Listing 3.2 to a function. Add a single input argument called `kind` that allows you to select a type of food from your list of favorite take-out restaurants. Inside the body of the function, set up a multiple-choice `if ... elseif ... end` block to match the `kind` string with the predefined restaurant types. Make sure that there is a default choice—either a restaurant type or an error message—if `kind` does not match any of the available restaurant types. (*Hint*: use the `strcmp` function to compare to strings for equality.)

Solution: The `takeout2` function, listed below, is a solution to the assignment. Note that `disp` functions have been used to display the restaurant name and number to the screen. This is better at conveying the *intent* of the program, which is to print the name and telephone number. A reader of the original `takeout` function might conclude that the purpose of the function is to assign values that are not used in any subsequent statements — not an obvious purpose.

You should test the `takeout2` program. What happens if you misspell one of the kinds, for example, by entering `takeout2('burgur')`? An alternative implementation of the function would provide a default restaurant instead of an error message. While you are at it, you might as well replace the fictitious restaurants with real restaurants in your neighborhood.

```
function takeout2(kind)
% takeout2  Display restaurant telephone numbers.   Exercise 3-3
%
% Synopsis:  takeout2(kind)
%
% Input:     kind = (string) type of food.  Example: kind = 'pizza'
%
% Output:  Print out of telephone numbers for restaurants matching 'kind'
if strcmp(kind,'chinese')
  disp('Mandarin Cove  221-8343');
elseif strcmp(kind,'pizza')
  disp('Pizza Express   335-7616');
elseif strcmp(kind,'deli')
  disp('Bernstein Deli   239-4772');
elseif strcmp(kind,'burger')
  disp('Big Burger   629-4085');
elseif strcmp(kind,'mexican')
  disp('Burrito Barn   881-8844');
elseif strcmp(kind,'veggie')
  disp('Tofu Palace   310-7900');
else
  error(sprintf('"%s" does not match one of the restaurant types',kind));
end
```

3.5 Write a `newtsqrt` function to compute the square root of a number, using Newton's algorithm. Your function should encapsulate the code on page 115.

Solution: A version of `newtsqrt` is listed below. Does it work? What are the results of `newtsqrt(4)`, `newtsqrt(400)`, `newtsqrt(4e-16)`?

```
function r = newtsqrt(x,delta,maxit)
% newtsqrt  Newton's method for computing the square root of a number
%           Exercise 3-5
%
% Synopsis:  r = newtsqrt(x)
%            r = newtsqrt(x,delta)
%            r = newtsqrt(x,delta,maxit)
%
% Input:     x     = number for which the square root is desired
%            delta = (optional) convergence tolerance.  Default: delta = 5e-6
%            maxit = (optional) maximum number of iterations.  Default: maxit = 25
%
% Output:    r = square root of x to within relative tolerance of delta
%                The kth iterate for the square root of x is
%
%                    r(k) = 0.5*( r(k-1) + x/r(k-1))
%
%                Iterations are terminated after maxit iterations or
%                when abs(r(k-1)-r(k))/r(k-1) < delta

if x<0,  error('Negative input to newtsqrt not allowed');  end

if nargin<2,  delta = 5.0e-6;  end   %  Defaults for optional input arguments
if nargin<3,  maxit = 25;      end

r = x;         %  Initialize
rold = 2*r;    %  Make sure convergence test fails on first try
it = 0;
while abs(rold-r)/rold > delta & it<maxit
   rold = r;                   %  Save old value for convergence test
   r = 0.5*(rold + x/rold);    %  Update the guess at the root
   it = it + 1;                %  Increment the iteration counter
end
```

3.7 Elvis Presley was born January 8, 1935, and died on August 16, 1977. Write an `elvisAge` function that returns Elvis's age in years, assuming that he were alive today. (The fragment of code on page 103 may be of some help.)

Solution: The `elvisAge1` function is listed on the following page. Does it work?

```
function y = elvisAge1
% elvisAge  Compute age of Elvis Presley (assuming he's still alive)
%           Solutions to Exercise 3-7
%
% Synopsis:    y = elvisAge1
%
% Input:  none
%
% Output: y = years since Elvis was born on 8 January 1935

theDate = datevec(now);              %  measure age from today's date
birthdate = datevec(datenum(1935,1,8));
deltaTime = theDate - birthdate;     %  vector of time differences

y = deltaTime(1);
if ( birthdate(2) >= theDate(2) )  & ( birthdate(3) > theDate(3) )
  y = y - 1;   %  birthdate hasn't happened yet this month
end
```

3.10 Write a function called `mydiag` that mimics the behavior of the built-in `diag` function. At a minimum, the `mydiag` function should return a diagonal matrix if the input is a vector, and return the diagonal entries of the matrix if the input is a matrix. (*Hint*: The `size` function will be helpful.)

Partial Solution: An incomplete version of the `myDiag` function is listed below. Your job is to finish `myDiag` and verify that it works. Use the `demoMyDiag` function, listed on the next page, and any other tests you wish to devise, to validate your implementation of `myDiag`.

```
function D = myDiag(A)
% myDiag  Create a diagonal matrix, or extract the diagonal of a matrix
%
% Synopsis:  D = myDiag(A)  to extract a diagonal elements from a matrix A
%
% Input: A = a vector or a matrix
%
% Output:  If A is a vector, D is a diagonal matrix with the elements
%          of v on the diagonal.  If A is a matrix, D is a vector
%          containing the diagonal elements of A.

[m,n] = size(A);
if min(m,n)==1    %  input is a vector
  r = max(m,n);   %  m=1 or n=1, pick largest to dimension D
  D = zeros(r,r);

  ... your code here

else              %  input is a matrix

  ... your code here

end
```

```
function demoMyDiag
% demoMyDiag  Test the myDiag function.    Exercise 3-10

% --- Test creation of diagonal matrices
fprintf('\nVerify that myDiag correctly creates diagonal matrices\n');
fprintf('   m        error\n');
for m = [1 4 7 13 20]
  v = rand(m,1);
  D1 = diag(v);
  D2 = myDiag(v);
  fprintf(' %3d   %12.3e\n',m,norm(D1-D2,Inf));
end

% --- Test extraction of diagonal vector
fprintf('\nVerify that myDiag correctly extracts the diagonal of a matrix\n');
fprintf('   m     n        error\n');
for m = [1 4 20]
  for n = [1  5  17]
    A = rand(m,n);
    d1 = diag(A);
    d2 = myDiag(A);
    fprintf(' %3d   %3d   %12.3e\n',m,n,norm(d1-d2));
  end
end
```

Running `demoMyDiag` for a correct implementation of `myDiag` gives

```
>> demoMyDiag

Verify that myDiag correctly creates diagonal matrices
   m        error
   1      0.000e+00
   4      0.000e+00
   7      0.000e+00
  13      0.000e+00
  20      0.000e+00

Verify that myDiag correctly extracts the diagonal of a matrix
   m     n        error
   1     1      0.000e+00
   1     5     -0.000e+00
   1    17     -0.000e+00
   4     1     -0.000e+00
   4     5      0.000e+00
   4    17      0.000e+00
  20     1     -0.000e+00
  20     5      0.000e+00
  20    17      0.000e+00
```

3.16 Write the MATLAB statements that use a loop and the `fprintf` function to produce the following table (the format of the numerical values should agree exactly with those printed in the table).

```
theta    sin(theta)   cos(theta)
   0       0.0000       1.0000
  60       0.8660       0.5000
 120       0.8660      -0.5000
 180      -0.0000      -1.0000
 240      -0.8660      -0.5000
 300      -0.8660       0.5000
 360       0.0000       1.0000
```

Partial Solution: The body of the loop can be one line:

```
fprintf('%6d    %9.4f    %9.4f\n',thetad(k),sin(thetar(k)),cos(thetar(k)))
```

Many other loop blocks can be made to work.

3.22 Write a function to plot the displacement of a cantilevered beam under a uniform load. In addition to creating the plot, the function should return the maximum deflection and angle between the horizontal and the surface of the beam at its tip. The formulas for the displacement y and tip angle θ are

$$y = -\frac{wx^2}{24EI}\left(6L^2 - 4Lx + x^2\right) \qquad \text{and} \qquad \theta = \frac{1}{6}\,\frac{wL^3}{EI},$$

where w is the load per unit length, E is Young's modulus for the beam, I is the moment of inertia of the beam, and L is the length of the beam. Test your function with $E = 30$ Mpsi, $I = 0.163$ in^4, $L = 10$ in, $w = 100$ lb$_\text{f}$/in.

Partial Solution: The `cantunif` function, listed below, evaluates the formulas for the displacement y and tip angle θ for any set of E, I, L, and w values. The `cantunif` function also plots $y(x)$. To complete the Exercise, the `cantunif` function is called with the appropriate inputs. What is the value of y_max and θ for the given values of E, I, L, and w?

```
function [ymax,theta] = cantunif(E,I,L,w)
% cantunif  Plot the deflection of a uniformly loaded cantilevered beam.
%           Return the maximum deflection and the slope at the tip.  Exercise 3-22
%
% Synopsis:  [ymax,theta] = cantunif(E,I,L,w)
%
% Input:  E = Young's modulus
%         I = moment of inertia of the beam
%         L = length of the beam
%         w = uniform load on the beam (weight per unit length)
%
%  NOTE:  inputs must have internally consistent units
%
% Output: ymax = maximum deflection at the tip
%         theta = angle in radians between horizontal and the
%                 beam surface at the tip
x = linspace(0,L);
y = -w*x.^2 .* (x.^2 + 6*L^2 - 4*L*x)/(24*E*I);

ymax = max(abs(y));
if ymax~=abs(y(end))    %  displacement formula should gives max at x=L
  warning('Max displacement not at tip?');
  fprintf('y(end) = %g,  ymax = %g\n',y(end),ymax)
end
theta = w*L^3/(6*E*I);
plot(x,y);
xlabel('Position along the beam');
ylabel('Displacement under uniform load');
title(sprintf('E = %g, I = %5.3f, L = %g, w = %g',E,I,L,w));
```

3.28 The `corvRain.dat` file in the data directory of the NMM toolbox contains monthly precipitation data for Corvallis, Oregon, from 1890 to 1994. The data are organized into columns, where the first column lists the year and the next 12 columns give the precipitation, in hundredths of an inch for each month. Write a MATLAB function to compute and plot the *yearly total* precipitation in inches (not hundredths of an inch) from 1890 to 1994. The function should also print the average, the highest, and the lowest annual precipitation for that period. The `loadColData` function in the `utils` directory of the NMM toolbox will be helpful in reading the data without having to delete the column headings. Can you compute the yearly total precipitation without using any loops?

Partial Solution: The beginning of the `precip` function is listed below. This function takes a data file (`corvRain.dat` in this case) as input, and computes the precipitation totals. Your job is to finish the `precip` function.

```
function precip(fname)
% precip  Plot precipitation data from OSU Weather Service.
%         Uses loadColData() function to import data and labels.
%
% Synopsis:  precip(fname)
%
% Input:    fname = name of file containing the plot data
%
% Output:   Plot of annual rainfall.  Compute and print the average,
%           low and high values of yearly rainfall

%  Data file has 13 columns of data, 1 header of text, 1 row of column labels
[year,P,labels] = loadColData(fname,13,1);

% --- Sum over rows of P to get annual precipitation.
%  sum(P) adds entries in each *column* of P.   We need sum over
%  rows of the P matrix, so use sum(P').  After the sum is performed
%  convert from hundredths of an inch to inches, and transpose
%  again so that the yearly total is stored in a column vector.
pt = (sum(P')/100)';

... your code goes here
```

Running the completed `precip` function gives the following output. Note that the formatting of the output is not important. The numerical values will be helpful, however, as a check on your complete program.

```
>> precip('corvrain.dat')

Summary of precipitation data from file corvrain.dat
    Average annual precipitation =  40.3 inches
    Highest annual precipitation =  58.7 inches
    Lowest annual precipitation  =  23.0 inches
```

3.31 Write a function called `maxi` to return the index of the most positive element in a vector. In other words `imax = maxi(v)` returns `imax` such that `v(imax)` $\geq$ `v(i)` for all `i`.

Solution: The complete `maxi` function is listed below. To complete the Exercise, use the two parameter form of the `max` function to test `maxi` for a variety of vector inputs.

```
function imax = maxi(v)
% maxi  Given vector v, find imax such that v(imax) >= v(i) for all i
%       Use a for loop.  Exercise 3-31
imax = 1;
for i = 2:length(v)
  if v(i)>v(imax),  imax=i;  end
end
```

3.43 Write an m-file function to compute the heat transfer coefficient for a cylinder in cross flow. The heat transfer coefficient h is defined in terms of the dimensionless Nusselt number $\mathrm{Nu} = hd/k$, where d is the diameter of the cylinder and k is the thermal conductivity of the fluid. The Nusselt number is correlated with the Reynolds number Re and Prandtl number Pr by

$$\mathrm{Nu} = C\mathrm{Re}^m\mathrm{Pr}^{1/3}, \qquad \text{where} \qquad \mathrm{Re} = \frac{\rho V d}{\mu} \quad \text{and} \quad \mathrm{Pr} = \frac{\mu c_p}{k},$$

in which ρ, μ, and c_p are, respectively, the density, dynamic viscosity, and specific heat of the fluid and V is the velocity of the fluid approaching the cylinder. The parameters C and m depend on Re according to the following table:

Re range	c	m
0.4 — 4	0.989	0.330
4 — 40	0.911	0.385
40 — 4000	0.683	0.466
4000 — 40,000	0.193	0.618
40,000 — 400,000	0.027	0.805

Use the following for the first line of your m-file

```
function h = cylhtc(d,v)
```

where `d` is the cylinder diameter and `v` is the fluid velocity. Use the properties of air at one atmosphere and 20°C.

$$\rho = 1.204\ \mathrm{kg/m}^3, \quad \mu = 1.82 \times 10^{-5}\ \mathrm{N\cdot\ s/m}^2, \quad c_p = 1007\ \mathrm{J/(kg\ K)}, \quad k = 26.3 \times 10^{-3}\ \mathrm{W/(m\,^\circ C)}$$

Typographical Error: In the first printing of the text, the value of k is given as $26.3\,\mathrm{W/(m\,^\circ C)}$, which is off by a factor of 1000. The correct value of the thermal conductivity is $k = 26.3 \times 10^{-3}\,\mathrm{W/(m\,^\circ C)}$.

Partial Solution: The values of c and m are set with an extended `if...else` construct. The beginning of the construct looks like.

```
if Re < 0.4
  error(sprintf('Re = %e is too low for correlation; min Re is 0.4',Re));
elseif Re < 4
  c = 0.989;   m = 0.330;
...
```

A `demoCylhtc` function (not listed here) was written to test `cylhtc`. Running `demoCylhtc` gives the following output.

```
>> demoCylhtc

  diameter   velocity        Re          h
     m         m/s                    W/(m^2 C)
   0.001      0.008         0.53        18.693
   0.010      0.008         5.29         4.035
   0.200      0.015       198.46         0.937
   0.200      1.500     19846.15        10.190
   0.200     15.000    198461.54        57.894
```