```
% ReadMe File
%
% Name/Author : Joshua Hernandez
% Student #   : 6852561
%
% MATH 4310
% Final Examination
%
% The following is a comprehensive list of all functions used in the
% toolkit that was requested by assignment 2 and 3 and the final examination
% of MATH 4310. Following the list are block comments for each function of
% there purpose, required input and expected output. When appropriate,
% definitions will be explained and theorems will be mentioned.


% Function list :
bounds_spectral_radius
circle_for_plot
force_square_matrix
Gershgorin_disks
is_anti_hermitian_matrix
is_complex_matrix
is_diagonal_matrix
is_hermitian_matrix
is_Hermitian
is_irreducible_matrix
is_lower_triagular_matrix
is_matrix
is_nonnegative_matrix
is_normal
is_normal_matrix
is_positive_matrix
is_primitive_matrix
is_real_matrix
is_skewHermitian
is_square_matrix
is_symplectic_matrix
is_unitary
is_unitary_matrix
is_upper_block_triangular
is_upper_triagular_matrix
my_matrix_norm
my_norm
Perron_Frobenius
plot_all_eigenvalues
plot_Gershgorin_disks
plot_spectral_radius
properties_matrix
reduce_matrix
sample_worksheet
shortest_paths
spectral_abscissa
spectral_radius
tol
```

```
function bound = bounds_spectral_radius(M)
% Purpose:
%   Provides the bounds on the spectral radius of M given by row sums.
%
% Input :
%   M = Matrix of float (real or complex) values
%
% Returns :
%   bound = a float the bound of the spectral radius
%           returns NaN if matrix is not square


function [x,y] = circle_for_plot(x0,y0,radius)
% circle_for_plot(x0,y0,radius)
%
% Returns [x,y] coordinates for a circle centered at (x0,y0) and with radius
% radius.


function force_square_matrix(M)
% Purpose:
%   Checks to see if the passed matrix is square.
%    If not, an error is raised.
%
% Input :
%   M = Matrix of float (real or complex) values
%
% Returns :
%   Nothing

function [cx,cy,radius] = Gershgorin_disks(M)
% Purpose:
%   Calculates the center and the radius of gershgorin disks in
%   the complex plain.
%
% Input :
%   M = Matrix of float (real or complex) values
%
% Returns :
%   cx      = x-coordinate of the center
%   cy      = y-coordinate of the center
%   radius  = the radius of the disk

function isTrue = is_anti_hermitian_matrix(M)
% Purpose:
%   Checks to see is the matrix is anti-Hermitian (if complex matrix)
%   or symmetric ( if real matrix )
%
% Definition:
%   anti-Hermitian : M = - M^*  <-- conjugate transpose
%   anti-Symmetric : M = - M^T  <-- Transpose
%
% Input :
```

```
%   M = Matrix of float (real or complex) values
%
% Returns :
%   isTrue = boolean relating if matrix is anti-Hermitian

function isTrue = is_complex_matrix(M)
% Purpose:
%   Checks to see if passed matrix has at least one complex entry
%
% Input :
%   M = Matrix of float (real or complex) values
%
% Returns :
%   isTrue = boolean relating if matrix is complex

function isTrue = is_diagonal_matrix(M)
% Purpose:
%   Checks to see if passed matrix is diagonal
%
% Def:
%   A diagonal matrix is one where M(i,i)=0 for all i and
%   M(i,j) ~= 0  for i~=j
%
% Input :
%   M = Matrix of float (real or complex) values
%
% Returns :
%   isTrue = boolean relating if matrix is diagonal

function isTrue = is_Hermitian(M)
% Purpose:
%   Checks to see is the matrix is Hermitian (if complex matrix)
%   or symmetrix ( if real matrix )
%
% Note: This function just calls is_hermitian_matrix(M)
%
% Definition:
%   Hermitian : M = M*   <-- conjugate transpose
%   Symmetrix : M = M^T  <-- Transpose
%
% Input :
%   M = Matrix of float (real or complex) values
%
% Returns :
%   isTrue = boolean relating if matrix is Hermitian

function isTrue = is_hermitian_matrix(M)
% Purpose:
%   Checks to see is the matrix is Hermitian (if complex matrix)
%   or symmetric ( if real matrix )
%
% Definition:
%   Hermitian : M = M*   <-- conjugate transpose
%   Symmetric : M = M^T  <-- Transpose
%
```

```
% Input :
%   M = Matrix of float (real or complex) values
%
% Returns :
%   isTrue = boolean relating if matrix is Hermitian

function isTrue = is_irreducible_matrix(M)
% Purpose:
%   Checks to see if the matrix satifies the definition of an
%   irreducible matrix, using the theorem below. This is done by taking powers
%   of M. When a non-zero entry appears it is recorded using logical 'or'.
%   M^k represents the paths of length exactly k. So if there is a non zero
%   entry, then there is a path. Taking all power of M from 1 to n
%   (note that the matrix must be n by n) and keeping track of non-zero
%   entries in any if the matrices. If every entry had a non-zero entry at
%   some point then the matrix is irreducible.
%
% Def (Reducible Matrix):
%   A Reducible matrix is one where there exsist permutation matrix, P
%   such that P * M * P^T is upper block triangular
%
% Def (Irreducible Matrix):
%   A Irreducible matrix is one where no such permutation matrix exsist
%   to make the given matrix upper block triangular.
%
% Theorem:
%   A matrix is irreducible if and only if the assosiated di-graph is
%   strongly connected.
%
% Def (strongly connceted Di-Graph):
%   A di-graph is strongly connceted if there exist a path between any two
%   vertices in the graph.
%
% Input :
%   M = Matrix of float (real or complex) values
%
% Returns :
%   isTrue = Boolean represending if the matrix is irreducible
%

function isTrue = is_lower_triagular_matrix(M)
% Purpose:
%   Checks to see if the passed matrix is lower triangular.
%
% Def:
%   A matrix is upper triangular if M(i,j)=0 for all i>j
%
% Input :
%   M = Matrix of float (real or complex) values
%
% Returns :
%   isTrue = boolean representing if the matrix is lower triangular

function isTrue = is_matrix(M)
% Purpose:
```

```
%   Checks to see if the passed parameter is a matrix. This is done by
%   looking at the size vector of M. if the size vector is 1x2, then it
%   is a two dimensional array, which we interpret to be a matrix.
%
% Input :
%   M = Matrix of float (real or complex) values
%
% Returns :
%   isTrue = boolean representing if the parameter is a matrix

function isTrue = is_nonnegative_matrix(M)
% Purpose:
%   Checks to see if the passed matrix satisfies the definition of
%   a nonnegative matrix, with the following definition.
%
% Def:
%   M(i,j)>= 0 for any i,j
%
% Input :
%   M = Matrix of float (real or complex) values
%
% Returns :
%   isTrue = boolean relating if matrix is nonnegative

function isTrue = is_normal(M)
% Purpose:
%   Checks to see if the matrix passed is normal by the following
%   definition.
%
% Note : This function just calls is_normal_matrix(M)
%
% Def:
%   A square matrix is normal if M and it's congugate-transpose commute
%
% Input :
%   M = Matrix of float (real or complex) values
%
% Returns :
%   isTrue = boolean representing if the matrix is normal

function isTrue = is_normal_matrix(M)
% Purpose:
%   Checks to see if the matrix passed is normal by the following
%   definition.
%
% Def:
%   A square matrix is normal if M and it's conjugate-transpose commute
%
% Input :
%   M = Matrix of float (real or complex) values
%
% Returns :
%   isTrue = boolean representing if the matrix is normal

function isTrue = is_positive_matrix(M)
```

```
% Purpose:
%   Checks to see if the passed matrix satisfies the definition of
%   a positive matrix. There are two definitions implemented. Only
%   one should be used.
%
%   def1,def2 are embedded functions that are used to implement the
%   two definitions
%
% Input :
%   M = Matrix of float (real or complex) values
%
% Returns :
%   isTrue = boolean relating if matrix is positive

function isTrue = is_primitive_matrix(M)
% Purpose:
%   Checks to see if the passed matrix is primitive by the following
%   definition.
%
% Definition:
%     A nonnegative square matrix A=(aij) is said to be a primitive matrix
%     if there exists k such that A^k is a positive matrix.
%
% Note:
%   This function assume that def1 in the function is_positive_matrix is
%   being used.
%
% Input :
%   M = Matrix of float (real or complex) values
%
% Returns :
%   isTrue = a boolean representing if the matrix is primitive

function isTrue = is_real_matrix(M)
% Purpose:
%   Checks to see if passed matrix has at no complex entries
%
% Input :
%   M = Matrix of float (real or complex) values
%
% Returns :
%   isTrue = boolean relating if matrix is real and not complex

function isTrue = is_skewHermitian(M)
% Purpose:
%   Checks to see is the matrix is skew-Hermitian (if complex matrix)
%   or symmetrix ( if real matrix )
%
% Note : This function just calls is_anti_hermitian_matrix(M) as the
%   definition is exactly the same.
%
% Definition:
%   skew-Hermitian : M = - M^*  <-- conjugate transpose
%   skew-Symmetrix : M = - M^T  <-- Transpose
%
```

```
% Input :
%   M = Matrix of float (real or complex) values
%
% Returns :
%   isTrue = boolean relating if matrix is skew-Hermitian

function isTrue = is_square_matrix(M)
% Purpose:
%   Checks to see if the passed matrix is square. This is done by looking
%   a the size vector of M. If both entries are the same then each
%   dimension of the array is the same value. We take this to be a square
%   matrix.
%
% Input :
%   M = Matrix of float (real or complex) values
%
% Returns :
%   isTrue = boolean value representing if the matrix is square

function isTrue = is_symplectic_matrix(M)
% Purpose:
%   Checks to see if the passed matrix is symplectic by the following
%   definition.
%
% Def :
%   omega = [    0    I_n   ;
%              -I_n    0    ]
%
% Def (Symplectic):
%   A matrix is symplectic is : M' * omega * M = omega
%   where M' is the transpose of M if it is real
%   or the conjugate transpose if N is complex
%
% Note:
%   If M is real then the definition is well recognized, but if M is
%   complex then if M satisfies this definition then it is sometimes called
%   a conjugate symplectic.
%
% Input :
%   M = Matrix of float (real or complex) values
%
% Returns :
%   isTrue = boolean value representing if the matrix is symplectic

function isTrue = is_unitary(M)
% Purpose:
%   Checks to see is the matrix is unitary (if complex matrix)
%   or orthognal ( if real matrix )
%
% Note : This function just calls is_unitary_matrix(M)
%
% Definition:
%   Unitary    : M * (M^*) = I   <-- conjugate transpose
%   Orthogonal : M * (M^T) = I   <-- Transpose
%
```

```
% Input :
%   M = Matrix of float (real or complex) values
%
% Returns :
%   isTrue = boolean relating if matrix is unitary

function isTrue = is_unitary_matrix(M)
% Purpose:
%   Checks to see is the matrix is unitary (if complex matrix)
%   or orthogonal ( if real matrix )
%
% Definition:
%   Unitary    : M * (M^*) = I   <-- conjugate transpose
%   Orthogonal : M * (M^T) = I   <-- Transpose
%
% Input :
%   M = Matrix of float (real or complex) values
%
% Returns :
%   isTrue = boolean relating if matrix is unitary

function isTrue = is_upper_block_triangular(M)
% Purpose:
%   Checks to see if the passed matrix is upper block triangular.
%   This is done by checking the following conditions
%     - an upper left square block (if square)
%     - a lower right square block (enforced by algorithm)
%     - a lower left rectangular zero block (enforced/checked)
%
%   This is done by checking the lower left rectangular block
%   to see if it composed of zeros.
%
% Input :
%   M = Matrix of float (real or complex) values
%
% Returns :
%   isTrue = boolean representing if the matrix is upper block triangular

function isTrue = is_upper_triagular_matrix(M)
% Purpose:
%   Checks to see if the passed matrix is upper triangular by the
%   definition below
%
% Def:
%   A matrix is upper triangular if M(i,j)=0 for all i>j
%
% Input :
%   M = Matrix of float (real or complex) values
%
% Returns :
%   isTrue = boolean representing if the matrix is upper triangular

function norm = my_matrix_norm(A, p,varargin)
% Purpose:
%   Attempts to evaluate the matrix norm of A induced by the vector-p-norm.
```

```
% This is done with a maxamization of A*x where |x|=1
%
% Definition:
%   |A|_p = max( |A*x|_p such that |x|=1 )
%
% Input :
%   A           = Matrix of float (real or complex) values
%   p           = index of which vector norm to use
%   varargin{1} = boolean flag for test functions
%   varargin{2} = boolean flag for debug functions and options
%
% Returns :
%   norm = the aproximate value of the norm of A

function norm = my_norm(v, p)
% Purpose:
%   Attempts to evaluate the vector-p-norm of v
%
% Definition:
%   |v|_p = ( sum( v^p )  )^(1/p)
%
% Input :
%   v = vector to which we take the norm of
%   p = which power to use in the norm
%
% Returns :
%   norm = the value of the p-norm of v

function canApply = Perron_Frobenius(M)
% Purpose:
%   Checks to see if the Perron Frobenius theorem can be applied to the
%   passed matrix. If it is able to, then it will display the information
%   given by the theorem. It will also base the output based off the
%   specific case that the matrix calls into.
%
% Statment of Theorem ( 7.2.1.4 Keyfitz and Caswell ):
%
% The Perron?Frobenius theorem describes the eigenvalues and eigenvectors of
% a nonnegative matrix A. Its most important conclusion is that there generally
% exists one eigenvalue that is greater than or equal to any of the others in
% magnitude. Without loss of generality, we will call this eigenvalue ?1; it
% is called the dominant eigenvalue of A.
%
%     Primitive matrices: If A is primitive, then there exists a real, positive
%     eigenvalue ?1 that is a simple root of the characteristic equation. This
%     eigenvalue is strictly greater in magnitude than any other eigenvalue.
%     The right and left eigenvectors w1 and v1 corresponding to ?1 are real and
%     strictly positive. There may be other real eigenvalues besides ?1, but ?1
%     is the only eigenvalue with nonnegative eigenvectors.
%
%     Irreducible but imprimitive matrices:
%     If the matrix A is irreducible but  imprimitive, with index of
%     imprimitivity d, then there exists a real positive  eigenvalue ?1
%     which is a simple root of the characteristic equation. The associated
%     right and left eigenvectors w1 and v1 are positive. The dominant eigenvalue
```

```
%     ?1 is greater than or equal in magnitude to any of the other eigenvalues;
%     i.e., ?1 ? |?i| i > 1
%     but the spectrum of A contains d eigenvalues equal in magnitude to
%     ?1. One is ?1 itself; the others are the d ? 1 complex eigenvalues
%
%     Reducible matrices:
%     If A is reducible, there exists a real eigenvalue ?1 ? 0 with corresponding
%     right and left eigenvectors w1 ? 0 and v1 ? 0.
%     This eigenvalue ?1 ? |?i| for i > 1.
%
% Input :
%   M = Matrix of float (real or complex) values
%
% Returns :
%   canApply = boolean variable that represents if the Perron Frobenius
%                theorem can be applied.

function plot_all_eigenvalues(M)
% Purpose:
%   Takes the passed matrix and plots the eigenvalues, treating the graph
%   as the complex plane.
%
% Input :
%   M = Matrix of float (real or complex) values
%
% Returns :
%   Nothing

function plot_Gershgorin_disks(M)
% Purpose:
%   Plots the Gershporin Disks. The program has been modified so that it
%   runs for matrices that are bigger than 4x4
%
% Input :
%   M = Matrix of float (real or complex) values
%
% Returns :
%   Nothing

function plot_spectral_radius(M)
% Purpose:
%   To plot the circle that contains all the eigenvalues of the passed
%   matrix on the complex plane. This finds the spectral radius and plots a
%   circle of that radius centered at the origin. It is plotted it '.' for
%   a bolding effect.
%
% Input :
%   M = Matrix of float (real or complex) values
%
% Returns :
%   Nothing

function properties_matrix(M)
% Purpose:
%   Given the passed matrix, any information that can be concluded about
```

```
%   the matrix will be displayed. Also the eigenvalues, Gershporin Disks,
%   spectral circle will be plotted by treating the graph as the complex
%   plane.
%
% Input :
%   M = Matrix of float (real or complex) values
%
% Returns :
%   Nothing

function rMatrix = reduce_matrix(M)
% Purpose:
%   Given that the passed matrix is reducible, the function finds the
%   reduced matrix by trying all possible permutation matrices.
%
% Input :
%   M = Matrix of float (real or complex) values
%
% Returns :
%   rMatrix = the reduced matrix of M (if possible)

function sample_worksheet()
% Purpose:
%   The aim of this assignment is to start developing a MatLab/Octave
%   toolkit for dealing with matrix problems. This worksheet will test that
%   toolkit with various matrices.
%
% Input :
%   Nothing
%
% Returns :
%   Nothing

function paths = shortest_paths(M)
% Purpose:
%   If the passed matrix is interperted as an adjacency matrix then this
%   function will construct a new matrix where entry (i,j) is the number of
%   shortest paths from vertex i to j.
%
%   This is done by taking powers of M. When a non-zero entry appears it is
%   recorded. This is since any higher power of M results in a longer path.
%
% Note:
%   is_irreducible_matrix is not used as it would be inefficient. Instead it
%   is calculated with the calculation of path. Since the algorithms for
%   each are very similar, integrating the two is simple. If it is
%   determined that M is reducible, then the matrix is filled with zeros
%   before being returned
%
% Input :
%   M = Matrix of float (real or complex) values
%
% Returns :
%   paths = matrix of shortest paths from vertex i to j found at
%             entry (i,j)
```

```
function s_of_M = spectral_abscissa(M)
% Purpose:
%   Calculates the spectral abscissa of M and returns it. This is
%   determined by using the following definition.
%
% Def:
%   spectral abscissa of M denoted s(M) is
%   s(M) = max( real(lambda) | lambda in spectrum of M  )
%
% Input :
%   M = Matrix of float (real or complex) values
%
% Returns :
%   s_of_M = the spectral abcissa of M

function sRadius = spectral_radius(M)
% Purpose:
%   Calculates the spectral radius of M and returns it. This is
%   determined by using the following definition.
%
% Def:
%   spectral radius of M the maximum of the magnitudes of all the
%   eigenvalues of M.
%
% Input :
%   M = Matrix of float (real or complex) values
%
% Returns :
%   sRadius = the spectral radius of M


function tolernace = tol()
% Purpose:
%   To give global access to a constant used for the tolerance of float
%   calulations
%
% Input :
%   Nothing
%
% Returns :
%   tolernace = the tolerance and float calculation should use
```

Published with MATLAB® 7.12