MATH2101 Exercise Generator

Functions specifically designed for creating exercises related to RREF, Eigenvalues and Eigenvectors, and Gram-Schmidt Orthogonalization

RREF

Function MatrixGen(m, n, rank, check0 = False)

Parameters:

- m: int, must be between 2 and 5 inclusive
- n: int, must be between 2 and 5 inclusive
- rank: int, must be between 0 and min{m,n} inclusive
- check0: boolean. If check0 is True, each row and column of the matrix returned must contain at least 2 nonzero entries. Otherwise, no restriction is imposed on the number of zero entries. Default is False

Output:

- A SymPy matrix (A)

Details

Assume we want to generate A, an integral $m \times n$ matrix of rank k. An approach could be:

1. Generate R, a reduced row echelon form of A with all integral entries
2. Perform EROs on R to form A

   - Type I: $r_i \leftrightarrow r_j$
   - Type II: $cr_i \rightarrow r_i$, where c is a non-zero integer
   - Type III: $r_i + ar_j \rightarrow r_i$, where a is an integer

To keep the problem neat and ensure sufficient level of difficulty, the following constraints are imposed in the following algorithm:

- Different ranges are set on the randomly generated values suggested
- Maximum magnitude of the entries of A is 20

Applications of the proposed algorithm:

- Generating any problems related to reduced row echelon form

Eigenvalues and Eigenvectors

Function EigGen(n, AM_RR, JB_RR=[ ], AM_RI=[ ], AM_CR=[ ], AM_CI=[ ], invertible=False, diagonalizable=False)

Parameters:

- n - int, dimension of matrix
- AM_RR - list of int, algebraic multiplicity of real rational eigenvalues
- JB_RR - list of sublist of int, each sublist contains the dimension(s) of the Jordan block(s) of the corresponding rational eigenvalue indicated in AM_RR, default is [ ], which induces the program to set all dimension of the Jordan blocks to be 1
- AM_RI - list of int, algebraic multiplicity of real irrational eigenvalue pairs, default is [ ]
- AM_CR - list of int, algebraic multiplicity of non-real rational eigenvalue pairs, default is [ ]
- AM_CI - list of int, algebraic multiplicity of non-real irrational eigenvalue pairs, default is [ ]
- invertible - boolean, if set to True the generated matrix is invertible (i.e. all eigenvalues are nonzero), otherwise it may or may not be invertible, default = False
- diagonalizable - boolean, if set to True the generated matrix is diagonalizable, otherwise it is non-diagonalizable, effective ONLY IF JB_RR=[ ], default = False

Ouput:

- A SymPy matrix (A)

Details

Assume we want to generate integral matrices A with self-defined algebraic multiplicty (AM) and geometric multiplicity (GM). Since the characteristic polynomial of this type of matrix has integral cofficients, the eigenvalues must exist in either of the following forms with AM and GM possibly larger than 1:

- Type RR: Real, rational
- Type RI: Real, irrational, in pair ($a+\sqrt{b}$ and $a-\sqrt{b}$)
- Type CR: Non-real, rational, in pair ($a+bi$ and $a-bi$)
- Type CI: Non-real, irrational, in pair ($a+\sqrt{b}i$ and $a-\sqrt{b}i$)

To simplify the exercise, all eigenvalues have integral real and imaginary parts.

Moreover, this program only supports the generation of matrices with AM = GM for every eigenvalue of type other than RR. For each eigenvalue of type RR, user can specify the dimension(s) of the Jordan block(s) in the Jordan Normal Form. For example, for an eigenvalue with AM = 5, specifying 3,1,1 will mean the Jordan Normal Form of the resultant matrix will contain one $\begin{bmatrix} x & 1 & 0 \\ 0 & x & 1 \\ 0 & 0 & x \end{bmatrix}$ and two $\begin{bmatrix} x \end{bmatrix}$ as its diagonal blocks.

Two different methods are employed to generate the matrices under different conditions:

Method 1

Condition: The matrix is diagonalizable, or JB_RR is specified

In this case, we first form a diagonal block matrix S that is similar to the desired matrix. To achieve this, eigenvalues are generated, then diagonal blocks are formed accordingly. Eventually, invertible matrix P is generated using the Neat Matrix Generator, and the desired matrix is $A=PSP^{-1}$.

The forms of the diagonal blocks are presented below:

- Type RR: Blocks described in the "Objective" section
- Type RI: $\begin{bmatrix} a & b \\ 1 & a \end{bmatrix}$ (which is similar to $\begin{bmatrix} a+\sqrt{b} & 0 \\ 0 & a-\sqrt{b} \end{bmatrix}$)
- Type CR: $\begin{bmatrix} a & b \\ -b & a \end{bmatrix}$ (which is similar to $\begin{bmatrix} a+bi & 0 \\ 0 & a-bi \end{bmatrix}$)
- Type CI: $\begin{bmatrix} a & -b \\ 1 & a \end{bmatrix}$ (which is similar to $\begin{bmatrix} a+\sqrt{b}i & 0 \\ 0 & a-\sqrt{b}i \end{bmatrix}$)

Method 2

Condition: The matrix is non-diagonalizable and JB_RR is not specified

In this case, we first form a characteristic polynomial by generating eigenvalues according to the given input. If the desired matrix A is of dimension $n \times n$, the polynomial is of degree n.

Suppose that the polynomial is of the form $p(t) = t^n + c_{n-1}t^{n-1} + \ldots + c_1 t + c_0$. Then since there are a total of n^2 entries in A, equating $p(t)$ to $det(A - tI)$, there are n equations with n^2 unknowns, so we can randomly assign $n(n-1)$ of the n^2 unknowns to some values, and solve for the remaining n unknowns to obtain a desired matrix.

Note that a random assignment of $n(n-1)$ of the n^2 unknowns can result in no solution or infinitely many solutions to the original equation. If this happens, then for simplicity the whole problem is discarded. A new problem will be generated.

Note that this generation method is quick, but it tends to return a non-diagonalizable matrix when the AM of one or more eigenvalues are larger than 1. Hence the condition imposed for this method.

Example

Calling EigGen(n=9, AM_RR=[3,2], JB_RR=[[2,1], [2]], AM_RI=[ ], AM_CR=[2], AM_CI=[ ], invertible=False) means:

You want to generate a 9×9 invertible matrix with

- 1 real rational eigenvalue of AM = 3 and GM = 2
- 1 real rational eigenvalue of AM = 2 and GM = 1
- 1 pair of non-real rational eigenvalues, each of which is of AM = GM = 2

The generated matrix is non-diagonalizable.

Note: This is just an example to illustrate how the parameters should be set. In reality, it takes too long to generate a 9×9 matrix. Therefore, DO NOT try to call the function with these input.

Gram-Schmidt Orthogonalization

Function GramSchmidtGen(n,k)

Parameters:

- n: int, dimension of vectors
- k: int, number of vectors

Output: A list of k SymPy $n \times 1$ matrices

Details

Let $U = \{u_1, u_2, \dots, u_k\}$, where $u_i \in \mathbf{R}^n$. Let $V = \{v_1, v_2, \dots, v_k\}$ be the orthogonal basis generated by the standard Gram-Schmidt process. According to the process, we have the following formulae:

$$v_1 = u_1$$

$$v_2 = u_2 - \frac{u_2^T v1}{\|v1\|^2} v_1$$

$$\dots$$

$$v_i = u_i - \frac{u_i^T v1}{\|v1\|^2} v_1 - \frac{u_i^T v2}{\|v2\|^2} v_2 - \dots - \frac{u_i^T v_{i-1}}{\|v_{i-1}\|^2} v_{i-1}$$

$$\dots$$

For the neatness of the problems generated, the following constraints are imposed:

$$\frac{u_i^T v_j}{\|v_j\|^2} \in \mathbf{N} - \{0\} \ \forall i > j$$

$$\Leftrightarrow \quad v_j^T u_i = c_j^i \|v_j\|^2 \ \exists c_j^i \in \mathbf{N} - \{0\} \ \forall i > j$$

$$\Leftrightarrow \quad \begin{bmatrix} v_1^T \\ v_2^T \\ \vdots \\ v_{i-1}^T \end{bmatrix} u_i = \begin{bmatrix} c_1^i \|v_1\|^2 \\ c_2^i \|v_2\|^2 \\ \vdots \\ c_{i-1}^i \|v_{i-1}\|^2 \end{bmatrix} \exists c_j^i \in \mathbf{N} - \{0\} \ \forall j = 1, 2, \dots, i-1 \ \forall i = 2, 3, \dots, k \quad ---- \ ($$

An algorithm is thus designed to generate U:

1. Generate $u_1 = v_1$
2. For $i = 2, 3, \dots, k$
   - Generate u_i according to system (1)
   - Generate v_i according to the formulae of the Gram Schmidt process.

How to generate u_i?

For any $i = 2, 3, \dots, k$, assume that c_j^i are given for any $j = 1, 2, \dots, i-1$.

Since V is defined to be a linearly independent set, $rank\left(\begin{bmatrix} v_1^T \\ v_2^T \\ \vdots \\ v_{i-1}^T \end{bmatrix}\right) = i-1$, i.e. the matrix is of full row rank.

Hence, system (1) is consistent and $nullity\left(\begin{bmatrix} v_1^T \\ v_2^T \\ \vdots \\ v_{i-1}^T \end{bmatrix}\right) = n-i+1$, meaning that the number of free variables is $n-i+1$.

To obtain a particular solution of system (1), the following assignment is performed:

- $u_j = s_j \; \exists s_j \in \mathbf{N} \; \forall j = i, i+1, \ldots, n$, i.e. to set $u_i, u_{i+1}, \ldots, u_n$ as the free variables of the equation.

Let $s = \begin{bmatrix} s_i \\ s_{i+1} \\ \vdots \\ s_n \end{bmatrix}$. Then, system (1) becomes:

$$\begin{bmatrix} v_1[1:i-1]^T & v_1[i:n]^T \\ v_2[1:i-1]^T & v_2[i:n]^T \\ \vdots & \\ v_{i-1}[1:i-1]^T & v_{i-1}[i:n]^T \end{bmatrix} \begin{bmatrix} u_i[1:i-1] \\ s \end{bmatrix} = \begin{bmatrix} c_1\|v_1\|^2 \\ c_2\|v_2\|^2 \\ \vdots \\ c_{i-1}\|v_{i-1}\|^2 \end{bmatrix} \; \exists c_j \in \mathbf{N} - \{0\} \; \forall j = 1, 2, .$$

or

$$\begin{bmatrix} v_1[1:i-1]^T \\ v_2[1:i-1]^T \\ \vdots \\ v_{i-1}[1:i-1]^T \end{bmatrix} u_i[1:i-1] = \begin{bmatrix} c_1\|v_1\|^2 - v_1[i:n]^T s \\ c_2\|v_2\|^2 - v_2[i:n]^T s \\ \vdots \\ c_{i-1}\|v_{i-1}\|^2 - v_{i-1}[i:n]^T s \end{bmatrix} \; \exists c_j \in \mathbf{N} - \{0\} \; \forall j = 1, 2, \ldots,$$

Note that a random assignment of $s_i, s_{i+1}, \ldots, s_n$ can result in no solution or infinitely many solutions to the original equation. If this happens, then for simplicity the whole problem is discarded. A new problem will be generated.