Tutorial
Solutions of systems of ordinary differential equations using GNU-Octave or Scilab.

November 20, 2019

Some quick commands

1. `[P,D] = eig(A)` : `P` is the eigenvector matrix of `A`, diagonal matrix `D` contains eigen values of `A` as diagonal elements.
2. `[P,J] = jordan(sym(A))` : `J` is the Jordan canonical form of `A`, `P` is the generalised eigenvector matrix.
3. `A/B` $= AB^{-1}$, `A\B` $= A^{-1}B$, `A*B` $= AB$

Q.(1) Consider the following systems of first order ordinary differential equations (ODEs)

(a) $x_1' = 2x_1 + x_2,$
$x_2' = x_1 + 3x_2.$

(b) $x_1' = x_2,$
$x_2' = -x_1.$

(c) $x_1' = 2x_1,$
$x_2' = 2x_2 + 3x_3,$
$x_3' = -3x_2 + 2x_3.$

(d) $x_1' = 2x_1 + 3x_3,$
$x_2' = x_2,$
$x_3' = x_1 + 2x_3.$

(e) $x_1' = x_1 - x_2,$
$x_2' = x_1 + x_2,$
$x_3' = 3x_3 - 2x_4,$
$x_4' = x_3 + x_4.$

Write down the coefficient matrix for each of the systems Q.(1a)–Q.(1e). Create a GNU-Octave or Scilab file namely `coeff_matrix.m` or `coeff_matrix.sci`, respectively and store the coefficient matrices. Use the format `Mquestion_number` to name the matrices; for instance `Ma = [2, 1;1 3]`, `Mb = [0, 1;-1 0]`, etc. Write `coeff_matrix` in the preamble of a code to access these matrices.

Q.(2) Find eigenvalues and eigenvectors of coefficient matrices corresponding to Q.(1a)–Q.(1e). Use the command `[P,D] = eig(A)` (in Octave) or `[P,D] = spec(A)` in Scilab to obtain a diagonal matrix `D` with eigenvalues of A as diagonal entries, and a matrix `P` whose columns are the corresponding eigenvectors. Compute `P\A*P` ($P^{-1}AP$) and check whether it is same as `D`.

Q.(3) Choose an initial condition $\boldsymbol{x}_0$ and write a function code to compute the corresponding solution for each system Q.(1a)–Q.(1e). The code should take care of different cases of eigenvalues (real distinct, complex distinct, real and complex, real repeated and complex repeated). Name the function file as `ode_sys.m` with inputs A and $\boldsymbol{x}_0$.

Q.(4) Consider n^{th}-order ODE

$$a_0 x + a_1 x' + a_2 x'' + \cdots + a_n x^{(n)} = 0,$$

where $a_0, ..., a_n$ are constants. Convert it in the form $\boldsymbol{X}' = A\boldsymbol{X}$, where A is an $n \times n$ matrix and $\boldsymbol{X}$ is an $n \times 1$ vector. Store the entries a_0, a_1, $\ldots$, a_n as a vector `Va = [a_0,; a_1; a_2; ... ; a_n]` in a script namely `n_ode_coeff.m` or `n_ode_coeff.sci`. Use the following steps to create a function code, `coeff_gen.m`, with inputs `Va` and `n` that will convert the column vector `Va` to the coefficient matrix `coeff_A`.

- Create an $(n-1) \times (n-1)$ identity matrix using the command `eye(n-1)` with the name `eye_coef`.
- Create a scaled vector `Va_coef = -Va(1:n,1)/Va(n+1,1)` and zero vector `zero_coef = zeros(n-1,1)`.
- Create the coefficient matrix using the assembly command

  `coeff_A = [[zero_coef  eye_coef]; Va_coef'].`

Q.(5) Transform the equations

(a) $x'' + 4x' + 40x = 0$ (b) $x'''' = 3x' - x''$

into a system of equations and write down the coefficient matrix in each case. Compute the coefficient matrix using the function file `coeff_gen.m` and check whether the results are correct.

Q.(6) Use the initial conditions Q.(6a) and Q.(6b) for the questions Q.(5a) and Q.(5b) and the function `ode_sys.m` to solve them.

(a) $x(0) = 1, x'(0) = 0,$

(b) $x(1) = 2, x'(1) = 3, x''(1) = -6, x'''(1) = 4.$

Q.(7) **Jordan canonical form:** Consider the system of ODEs $\boldsymbol{X}' = A\boldsymbol{X}$, where

(a) $A = \begin{pmatrix} 1 & 0 & 0 \\ -1 & 2 & 0 \\ 1 & 1 & 2, \end{pmatrix}$ (b) $A = \begin{pmatrix} 0 & -1 & -2 & -1 \\ 1 & 2 & 1 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 \end{pmatrix}.$

Use the command `[P,D] = eig(A)` to obtain the eigenvector matrix and eigenvalues of A. Check the rank of `P` using the command `rank(P)`. What does `rank(P) < n` indicate? Are the matrices in Q.(7a)–Q.(7b) diagonalisable? Use the command `[P,J] = jordan(sym(A))` to find the Jordan canonical form of `A`.

Q.(8) Use the command `N = J - diag(diag(J))` to decompose `J` into a nilpotent matrix `N` and a diagonal matrix `D = diag(diag(J))`. Since `N` is nilpotent, use a `for` or `while` loop to obtain `Ne = exp(N*t)`. Then the exponential `exp(J*t)` can be computed using the command `EJ = diag(exp(diag(D)*t))*Ne`. Choose suitable initial conditions for the questions Q.(7a) and Q.(7b) and use the output `EJ` to obtain the solutions for $\boldsymbol{X}' = A\boldsymbol{X}$. Modify the code `ode_sys.m` such that these cases are also taken care of.

Note: The `jordan` command is contained in the package `symbolic` in GNU-Octave. So, when you need to use `jordan` in Octave, the argument must be of the form `sym(A)`. In MATLAB, you don't have to do this. A plain `jordan(A)` command will do the job.