MAT-63506 Scientific Computing

Exercise Set 5 9.–15. 4. 2018

Before doing the exercises read the files "LinearEquations.pdf" and "OOP.mlx".

Exercise 1. Let

$$A = \begin{bmatrix} 1 & -2 & 3 & 1 \\ -2 & 1 & -2 & -1 \\ 3 & -2 & 1 & 5 \\ 1 & -1 & 5 & 3 \end{bmatrix}, \qquad b = \begin{bmatrix} 10 \\ -10 \\ 22 \\ 26 \end{bmatrix}.$$

Solve the equation $Ax = b$ with the Gaussian elimination solver \ and set the solution to the variable `x1`.

Next find the LUP decomposition of A with the command `[L, U, P] = lu(A)` and solve $Ax = b$ by solving the two triangular systems as described in Section 3.1 of the file "LinearEquations". Use the function `linsolve`, and remember to set the appropriate option for lower and upper triangular matrices. Set the solution to the variable `x2`. Do you get the same solutions?

Exercise 2. Check equation (2) in the file "LinearEquations" as follows. Generate a random $n \times n$ matrix A with integer elements in $[-10, 10]$ with `randi`, let the exact solution x_e be a vector of all ones, compute $b = Ax_e$, and then solve the equation $Ax = b$ with the backkslash operator \. Finally compute the ratio of the right and left hand sides of equation (2) and plot the results for `n = 100:10:1000`. Use the command `cond` with the second argument `Inf` to compute the condition number.

Exercise 3. Make a class `Circle` for representing circles. It should have the property values `Center` and `Radius`. `Center` is the center of the circle, which sholud be a 1×2 floating point vector `[x y]`. `Radius` is the radius of the circle and it must be a positive floating point number.

The constructor `Circle(c, r)` should take as arguments the center `c` and the radius `r`. If the radius is not given it should be set to 1, and if the center is not given it should be set to $(0, 0)$. All other cases should lead to an error. Do the error checking with MATLAB's property validators.

The class sholud have a `plot`-method for plotting the circle, and `plus`-method for adding two circles. Addition is defined by adding the centers and radii of the circles.

The `plot`-method should take the form `h = plot(obj, varargin)`, where `varargin` are arguments that are passed to the builtin `plot` command and `h` is a handle to the plot. See the file "Graphics2D.mlx" for how to make a circle with the `rectangle` command.

As a check construct and plot the circles `c1 = Circle`, `c2 = Circle([1 1])`, `c3 = Circle([5 4], 4)`, `c4 = Circle([5 7], 11)`, and `c5 = c3 + c4`.

Also check that the constructor calls `Circle([3 2 8], 3)` and `Circle([1 2], -1)` give an error.

Exercise 4. A discrete probability distribution is given by a vector $p = (p_1, \ldots, p_n)$, where

$$p_i \in [0, 1] \text{ for } i = 1, \ldots, n, \tag{1a}$$

$$\sum_{i=1}^{n} p_i = 1. \tag{1b}$$

Random integers in the interval $1 \le i \le n$ with probability distribution p can be generated as follows: generate a random variable u uniformly distributed in the interval $(0, 1)$ and return the smallest integer i satisfying

$$u < \sum_{j=1}^{i} p_j. \tag{2}$$

Make a class `DiscreteDistribution` with a property `P` and a constructor that takes the vector p as an argument. You can check that `P` is a double vector that satisfies conditions (1a) using MATLAB's builtin property validators. To check condition (1b) write your own validator function `mustSumTo1`. To avoid roundoff error giving wrong answers, you should actually check that the sum is within `n*eps` of 1, where `eps` is the machine epsilon.

The class should have a method `A = random(obj, varargin)`, which generates an array of random integers `A` with the dimensions given in `varargin`. The method should accept the dimensions in the same forms as the builtin `rand` does, i.e., all of the following should work: `random(obj, 5)`, `random(obj, 5, 11)`, `random(obj, [11 34])`. Use the command `cumsum` to compute the sums needeed in (2). You can store it as a (private) property to avoid computing it more than once.

Test your class with the distribution $p = (0.3, 0.1, 0.2, 0.1, 0.3)$. Generate a vector of a few hundred random integers and plot the frequencies with `histogram`.

HINT: Initiialize `A = zeros(varargin{:})` and loop `for k = 1:numel(A)`.