Numerical Analysis and Computational Mathematics

Fall Semester 2019 - Section CSE

Dr. Rafael Vázquez Hernández

Assistant: Ondine Chanon

Session 8 - 5 November 2019

Numerical Integration & Linear Systems: Direct Methods

Exercise I (MATLAB)
Let us consider a function $f : [a, b] \to \mathbb{R}$, with $f \in C^0([a, b])$, whose integral is $I(f) = \int_a^b f(x)\, dx$. The approximation of $I(f)$ by means of a simple interpolatory formula reads:

$$I_{approx}(f) = \sum_{j=0}^{n} \alpha_j\, f(y_j),$$

where α_j are the quadrature weights and y_j the quadrature nodes, with $j = 0, \ldots, n$. The type of polynomial approximation of the function $f(x)$ in $[a, b]$ determines the specific quadrature formula. We observe that such formulas are typically defined for the reference interval $[\overline{a}, \overline{b}] = [-1, 1]$ where the quadrature nodes $\overline{y}_j$ and the weights $\overline{\alpha}_j$ are referred. Then, the quadrature nodes and weights corresponding to the generic interval $[a, b]$ are obtained as:

$$y_j = \frac{a+b}{2} + \frac{b-a}{2}\overline{y}_j, \qquad \alpha_j = \frac{b-a}{2}\overline{\alpha}_j, \qquad \text{for } j = 0, \ldots, n.$$

The (simple) *Gauss–Legendre quadrature formulas* constitute a family of interpolatory formulas, each one specified by n, where $n+1$ is the number of quadrature nodes and weights; the Gauss–Legendre quadrature formula corresponding to $n \geq 0$ has degree of exactness equal to $2n+1$. The quadrature nodes and weights for some of the Gauss–Legendre quadrature formulas in the reference interval $[\overline{a}, \overline{b}] = [-1, 1]$ are:

n	$\{\overline{y}_j\}$	$\{\overline{\alpha}_j\}$
0	$\{0\}$	$\{2\}$
1	$\left\{-\frac{1}{\sqrt{3}},\, +\frac{1}{\sqrt{3}}\right\}$	$\{1,\, 1\}$
2	$\left\{-\frac{\sqrt{15}}{5},\, 0,\, +\frac{\sqrt{15}}{5}\right\}$	$\left\{\frac{5}{9},\, \frac{8}{9},\, \frac{5}{9}\right\}$

a) Write the MATLAB function `gauss_legendre_simple_quadrature.m` that implements the approximation of $I(f)$ by means of the simple Gauss–Legendre quadrature formula for $n = 0, 1, 2$. Use the following template `gauss_legendre_simple_quadrature_template.m`.


```
function [ Ih ] = gauss_legendre_simple_quadrature( fun, a, b, n )
% GAUSS_LEGENDRE_SIMPLE_QUADRATURE approximate the integral of a function in
% the interval [a,b] by means of the simple Gauss-Legendre quadrature formula
%  [ Ih ] = gauss_legendre_simple_quadrature( fun, a, b, n )
%  Inputs: fun = function handle,
%          a,b = extrema of the interval [a,b]
%          n + 1 = number of quadrature nodes and weights
%  Output: Ih = approximate value of the integral
%


return
```

The inputs of the functions are: `fun` (the function handle of $f(x)$), the extrema of the interval `a`, `b`, and `n`, where $n+1$ is the number of quadrature nodes and weights.

b) Let us consider the function $f(x) = \sin((7/2)\,x) + e^x - 1$ with $a = 0$ and $b = 1$ ($f \in C^\infty([a,b])$) for which $I(f) = 2/7\,(1 - \cos(7/2)) + e - 2$. Use the MATLAB function implemented at point a) to approximate the integral $I(f)$ by means of the simple Gauss–Legendre formulas for $n = 0, 1, 2$. Report the values of the approximated integrals in comparison with $I(f)$.

c) We set $f(x) = x^d$, $a = 0$, and $b = 1$, with $d \in \mathbb{N}$. We obtain that $I(f) = 1/(d+1)$. By using the MATLAB function implemented at point a), verify the degrees of exactness of the Gauss–Legendre formulas (in the simple case) for $n = 0, 1, 2$ by approximating the integral $I(f)$ for different values of $d = 0, 1, 2, \ldots, 6$. Motivate the results obtained; then, compare and discuss the results with the approximated values of the integrals obtained by means of the simple midpoint, trapezoidal, and Simpson quadrature formulas.

Exercise II (Theoretical)
Consider the interval $I = [-1, 1]$ and verify that the Gauss-Legendre formula for $n = 1$ has degree of exactness $r = 3$. i.e. $I_{GL,1}(f) = I(f)$ for all $f \in \mathbb{P}_3$.

Exercise III (MATLAB)
Let us consider the linear system $A\mathbf{x} = \mathbf{b}$ with $A \in \mathbb{R}^{n\times n}$, $\mathbf{x}$, $\mathbf{b} \in \mathbb{R}^n$, with $n \geq 1$. We are interested in computing the solution vector $\mathbf{x}$ by means of the LU factorization method.

a) Write the MATLAB functions `forward_substitutions.m` and `backward_substitutions.m` which implement the forward and backward substitutions algorithms, respectively. The forward substitution algorithm should solve the linear system $L\mathbf{y} = \mathbf{b}$, with $L \in \mathbb{R}^{n\times n}$ a lower triangular matrix and $\mathbf{y} \in \mathbb{R}^n$; the backward substitution algorithm should solve the linear system $U\mathbf{x} = \mathbf{y}$, with $U \in \mathbb{R}^{n\times n}$ an upper triangular matrix. The functions should take as inputs the matrix L or U and the vectors $\mathbf{b}$ or $\mathbf{y}$, respectively; the outputs are the corresponding solutions vectors $\mathbf{y}$ or $\mathbf{x}$. Use the templates `forward_substitutions_template.m` and `backward_substitutions_template.m` reported in the following.

```
function [ y ] = forward_substitutions( L, b )
% FORWARD_SUBSTITUTIONS solve the linear system L y = b by means of the
% forward subsitutions algorithm; L must be a lower triangular matrix
%  [ y ] = forward_substitutions( L, b )
```

```
%  Inputs: L = lower triangular matrix (square matrix)
%          b = vector (right hand side of the linear system)
%  Output: y = solution vector (column vector)
%


return
```

```
function [ x ] = backward_substitutions( U, y )
% BACKWARD_SUBSTITUTIONS solve the linear system U x = y by means of the
% backward subsitutions algorithm; U must be an upper triangular matrix
%  [ x ] = backward_substitutions( U, y )
%  Inputs: U = upper triangular matrix (square matrix)
%          y = vector (right hand side of the linear system)
%  Output: x = solution vector (column vector)
%

return
```

b) By assigning a priori the exact solution $\mathbf{x}_{ex} \in \mathbf{R}^n$ of the linear system $A\mathbf{x} = \mathbf{b}$ we set for $n = 3$:

$$A = \begin{bmatrix} 4 & -2 & -1 \\ -1 & 3 & -1 \\ -1 & -3 & 5 \end{bmatrix}, \qquad \mathbf{x}_{ex} = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}, \qquad \mathbf{b} = A\mathbf{x}_{ex}.$$

Solve the linear system by means of the LU factorization method using the MATLAB functions implemented at point a). In order to obtain the LU factorization of the matrix A, use the MATLAB function `lu` with the following syntax: `[L,U,P]=lu(A)` (see `help lu`); indeed, we observe that the MATLAB function `lu` returns by default the LU factorization with pivoting even when not strictly required. Compare the result $\mathbf{x}$ with the exact solution $\mathbf{x}_{ex}$ by computing $\mathbf{x} - \mathbf{x}_{ex2}$. Verify if the pivoting technique has been applied by the MATLAB function `lu` by displaying the permutation matrix P.

c) We consider the following matrix $A \in \mathbb{R}^{n \times n}$ and vectors $\mathbf{x}_{ex}$, $\mathbf{b} \in \mathbb{R}^n$:

$$A = \begin{bmatrix} 4 & -1 & 0 & & & & & \cdots & 0 & 1 \\ -2 & 4 & -1 & 0 & & & & \cdots & 0 & 0 \\ -1 & -2 & 4 & -1 & 0 & & & \cdots & & 0 \\ 0 & -1 & -2 & 4 & -1 & 0 & & \cdots & & 0 \\ \vdots & & & \ddots & \ddots & \ddots & \ddots & & & \vdots \\ 0 & & \cdots & & 0 & -1 & -2 & 4 & -1 & 0 \\ 0 & 0 & \cdots & & & 0 & -1 & -2 & 4 & -1 \\ -1 & 0 & \cdots & & & & 0 & -1 & -2 & 4 \end{bmatrix}, \quad \mathbf{x}_{ex} = \begin{bmatrix} 1 \\ \vdots \\ 1 \end{bmatrix}, \quad \mathbf{b} = A\mathbf{x}_{ex}.$$

Set $n = 20$ and use the MATLAB function `diag` to assign the matrix A. Then, repeat point b) by visualizing the matrices A, L, and U by means of the MATLAB function `spy`; report a sketch of the pattern of the nonzero elements of the matrices.

d) Consider the matrix A introduced at point c) with $n = 1000$. Assign the matrix A in MATLAB in the *full* format (as done at point c)) and in *sparse* format; for the latter case suitably use the MATLAB function `sparse`. Compare the memory required to store in MATLAB the matrix A in the full and sparse formats by using the command `whos`; comment the results obtained.

e) With the notation of point b), let us consider the linear system $A\mathbf{x} = \mathbf{b}$ with:

$$A = \begin{bmatrix} 4 & -2 & -1 \\ -2 & 7 & -4 \\ -1 & -4 & 6 \end{bmatrix}, \qquad \mathbf{x}_{ex} = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}, \qquad \mathbf{b} = A\mathbf{x}_{ex}.$$

The matrix A is symmetric and positive definite. Solve the linear system by means of the *Cholesky* factorization method using the MATLAB functions implemented at point a); in order to obtain the *Cholesky* factorization of the matrix A, use the MATLAB function `chol` with the syntax `R=chol(A)` (see `help chol`). Compare the result $\mathbf{x}$ with the exact solution $\mathbf{x}_{ex}$ by computing $\mathbf{x} - \mathbf{x}_{ex2}$.

Exercise IV (Theoretical)

Let us consider the following matrix $A \in \mathbb{R}^{3\times 3}$ depending on the parameter $\alpha \in \mathbb{R}$:

$$A = \begin{bmatrix} 1 & \alpha & -1 \\ \alpha & \frac{35}{3} & 1 \\ -1 & \alpha & 2 \end{bmatrix}.$$

a) Calculate the values of the parameter $\alpha \in \mathbb{R}$ for which the matrix A is invertible (non singular).

b) Calculate the Gauss factorization LU of the matrix A (when non singular) for a generic value of the parameter $\alpha \in \mathbb{R}$.

c) Calculate the values of the parameter $\alpha \in \mathbb{R}$ for which the Gauss factorization LU of the matrix A (when non singular) exists and is unique.

d) Set $\alpha = \sqrt{\frac{35}{3}}$ and use the pivoting technique to calculate the Gauss factorization LU of the matrix A.

e) For $\alpha = 1$, the matrix A is symmetric and positive definite. Calculate the corresponding Cholesky factorization of the matrix A, i.e. the upper triangular matrix with positive elements on the diagonal, say R, for which $A = R^T R$.