Journal of Computational and Applied Mathematics 451 (2024) 116109

Contents lists available at ScienceDirect

Journal of Computational and Applied Mathematics

journal homepage: www.elsevier.com/locate/cam

A Matlab package computing simultaneous Gaussian quadrature rules for multiple orthogonal polynomials

Teresa Laudadio [a,*], Nicola Mastronardi [a], Walter Van Assche [b], Paul Van Dooren [c]

[a] Istituto per le Applicazioni del Calcolo "Mauro Picone", CNR, Via Amendola 122/D, Bari, 70126, Italy
[b] Department of Mathematics, KU Leuven, Celestijnenlaan 200B, Leuven, 3001, Belgium
[c] Department of Mathematical Engineering, Catholic University of Louvain, Batiment Euler (A.202), Avenue Georges Lemaitre 4, Louvain-la-Neuve, 1348, Belgium

ARTICLE INFO

MSC:
33C47
65D32
65F15

Keywords:
Multiple orthogonal polynomials
Simultaneous Gaussian quadrature rules
Banded Hessenberg eigenvalue problem

ABSTRACT

The aim of this paper is to describe a Matlab package for computing the simultaneous Gaussian quadrature rules associated with a variety of multiple orthogonal polynomials.

Multiple orthogonal polynomials can be considered as a generalization of classical orthogonal polynomials, satisfying orthogonality constraints with respect to r different measures, with $r \geq 1$. Moreover, they satisfy $(r+2)$-term recurrence relations. In this manuscript, without loss of generality, r is considered equal to 2. The so-called simultaneous Gaussian quadrature rules associated with multiple orthogonal polynomials can be computed by solving a banded lower Hessenberg eigenvalue problem. Unfortunately, computing the eigendecomposition of such a matrix turns out to be strongly ill-conditioned and the `Matlab` function `balance.m` does not improve the condition of the eigenvalue problem. Therefore, most procedures for computing simultaneous Gaussian quadrature rules are implemented with variable precision arithmetic. Here, we propose a `Matlab` package that allows to reliably compute the simultaneous Gaussian quadrature rules in floating point arithmetic. It makes use of a variant of a new balancing procedure, recently developed by the authors of the present manuscript, that drastically reduces the condition of the Hessenberg eigenvalue problem.

1. Introduction

In this paper, we consider the computation of simultaneous Gaussian quadrature rules associated with a variety of multiple orthogonal polynomials (MOPs). MOPs originally appeared in Hermite–Padé approximation (simultaneous rational approximation) and number theory. Recently, they turned out to be very useful in random matrix theory [1], combinatorics [2] and Markov chains [3]. Simultaneous Gaussian quadrature was introduced in [4] to model computer graphics illumination, where the computation of different weighted integrals with the same integrand function was needed. The aim was to minimize the evaluations of the integrand function and maximize the order of the quadrature rules based on the same set of nodes. Simultaneous Gaussian quadrature rules associated with MOPs related to the modified Bessel functions of the first and second kind were proposed in [5]. Gaussian quadrature with these special weight functions (and also with hypergeometric or confluent hypergeometric weights and the exponential integral) requires the computation of the recurrence coefficients of the corresponding orthogonal polynomials from the moments, which is an ill-conditioned numerical problem (see, *e.g.*, [6,7]). Surprisingly, the recurrence coefficients of MOPs for such weights are explicitly known, so that their numerical computation is avoided and the quadrature formula can be computed by

* Corresponding author.
E-mail address: teresa.laudadio@cnr.it (T. Laudadio).

https://doi.org/10.1016/j.cam.2024.116109
Received 20 March 2024; Received in revised form 18 June 2024
Available online 28 June 2024

solving an eigenvalue problem obtained by properly arranging these recurrence coefficients into an Hessenberg matrix, as proposed in [5,8].

MOPs are a generalization of orthogonal polynomials and can be divided into two classes: type I and type II [8]. In this paper we focus on MOPs of type II. Suppose r weight functions $w^{(i)}(x) \geq 0$, with support $\Delta^{(i)}$, $i = 1, \ldots, r$, on the real line are given. Then, the sequence of MOPs $\{p_n(x)\}_{n=0}^{\infty}$ of type II satisfy the following orthogonality conditions [8]:

$$\int_{\Delta^{(i)}} p_n(x) x^k w^{(i)}(x) dx = 0, \quad 0 \leq k \leq n_i - 1, \tag{1}$$

with $n = \sum_{i=1}^{r} n_i$.

Let $\Delta = \bigcup_{i=1}^{r} \Delta^{(i)}$. Two different systems of MOPs of type II can be considered [9,10]:

1. Angelesco system, where the open intervals $\Delta^{(i)}$, $i = 1, \ldots, r$, are disjoint, *i.e.*, $\Delta^{(i)} \bigcap \Delta^{(j)} = \emptyset$, for $i \neq j$, and the closed intervals $\Delta^{(i)}$ are allowed to touch.
2. algebraic Chebyshev system (AT system), where $\Delta^{(i)} = \Delta$, $i = 1, \ldots, r$.

Then, $p_n(x)$ of type II has exactly n zeros in Δ [9, Th. 2].

A set of MOPs satisfies an $(r + 2)$-term recurrence relation. Without loss of generality, in this paper we focus on the case $r = 2$. Therefore, the set of MOPs satisfies a 4-term recurrence relation[1]

$$x p_i(x) = a_i p_{i+1}(x) + b_i p_i(x) + c_i p_{i-1}(x) + d_i p_{i-2}(x), \quad i = 0, \ldots, n-1, \tag{2}$$

with $p_{-2}(x) = p_{-1}(x) = 0$. Writing (2) in matrix form, we obtain

$$H_n \begin{bmatrix} p_0(x) \\ p_1(x) \\ \vdots \\ p_{n-1}(x) \end{bmatrix} + a_{n-1} \begin{bmatrix} 0 \\ \vdots \\ 0 \\ p_n(x) \end{bmatrix} = x \begin{bmatrix} p_0(x) \\ p_1(x) \\ \vdots \\ p_{n-1}(x) \end{bmatrix},$$

where H_n is the $n \times n$ banded lower Hessenberg matrix with 2 sub-diagonals and one upper-diagonal :

$$H_n := \begin{bmatrix} b_0 & a_0 & 0 & 0 & 0 & \ldots & 0 \\ c_1 & b_1 & a_1 & 0 & 0 & \ldots & 0 \\ d_2 & c_2 & b_2 & a_2 & 0 & \ldots & 0 \\ 0 & d_3 & c_3 & b_3 & a_3 & \ddots & 0 \\ \vdots & \ddots & \ddots & \ddots & \ddots & \ddots & 0 \\ 0 & \ldots & 0 & d_{n-2} & c_{n-2} & b_{n-2} & a_{n-2} \\ 0 & \ldots & 0 & 0 & d_{n-1} & c_{n-1} & b_{n-1} \end{bmatrix}. \tag{3}$$

The Gaussian quadrature rule associated with classical orthogonal polynomials can be retrieved from the eigenvalue decomposition of a symmetric tridiagonal matrix, and it is exact for polynomials of degree $2n - 1$ [11]. The theory of simultaneous Gaussian quadrature rules for the general case $r > 2$ is described in [8].

For $r = 2$, the simultaneous Gaussian quadrature rule associated with MOPs [8],

$$\sum_{k=1}^{n} \omega_k^{(j)} f(x_j) = \int_{\Delta^{(i)}} f(x) w^{(j)}(x) dx + E_n^{(j)}(f), \quad j = 1, 2,$$

with $E_n^{(j)}(f) = 0$, if f is a polynomial of degree $n + n_j - 1$, can be retrieved from the eigenvalue decomposition of the matrix H_n [8, Th. 3.2],[5, Th. 2]. Therefore, the degree of exactness of the simultaneous Gaussian quadrature rule is maximal if all n_j, $j = 1, 2$, in (1) are equal [12]. The following theorem holds:

Theorem 1. *The nodes x_j, $j = 1, \ldots, n$, of the simultaneous Gaussian quadrature rule are given by the eigenvalues of the banded Hessenberg matrix H_n (3). Moreover, let us denote by $\boldsymbol{u}^{(j)} = [u_1^{(j)}, u_2^{(j)}, \ldots, u_n^{(j)}]^T$, and $\boldsymbol{v}^{(j)} = [v_1^{(j)}, v_2^{(j)}, \ldots, v_n^{(j)}]^T$ the left and right eigenvectors of H_n associated with x_j, respectively. Then,*

$$\omega_j^{(1)} = \frac{v_1^{(j)} f_{1,1} u_1^{(j)}}{\boldsymbol{u}^{(j)^T} \boldsymbol{v}^{(j)}}, \quad \omega_j^{(2)} = \frac{v_1^{(j)} \left(f_{2,1} u_1^{(j)} + f_{2,2} u_2^{(j)} \right)}{\boldsymbol{u}^{(j)^T} \boldsymbol{v}^{(j)}}, \quad j = 1, \ldots, n, \tag{4}$$

where

$$f_{1,1} = \int_{\Delta^{(1)}} p_0(x) w^{(1)}(x) dx,$$
$$f_{2,1} = \int_{\Delta^{(2)}} p_0(x) w^{(2)}(x) dx, \quad f_{2,2} = \int_{\Delta^{(2)}} p_1(x) w^{(2)}(x) dx. \tag{5}$$

[1] All the MOPS considered in the literature are monic, *i.e.*, $a_i = 1$, $i = 0, 1, \ldots, n-1$.

Remark 1. For $r > 2$, the simultaneous Gaussian quadrature rule

$$\sum_{k=1}^{n} \omega_k^{(j)} f(x_j) = \int_{\Delta^{(i)}} f(x) w^{(j)}(x) dx + E_n^{(j)}(f), \quad j = 1, \dots, r,$$

has degrees of exactness equal to $n + n_j - 1,\ j = 1, \dots, r$ [4, Th. 1], [8, Th. 3.1].

Hence, simultaneous Gaussian quadrature rules associated with MOPs reduces to the computation of the eigendecomposition of the Hessenberg matrix H_n, which, unfortunately, turns out to be strongly ill-conditioned [5]. Furthermore, the Matlab function balance.m applied to the Hessenberg matrix H_n does not improve the condition of the eigenvalue problem [13] and, then, the Matlab function eig.m yields unreliable results. Therefore, procedures for computing simultaneous Gaussian quadrature rules are implemented with variable precision arithmetic [5].

Recently, simultaneous Gaussian quadrature rules have been proposed for MOPs associated with modified Bessel functions of the first and second kind [5,13], where the banded lower Hessenberg matrix H_n is totally nonnegative. In particular, in [13], a new balancing procedure has been proposed that drastically reduces the condition of the aforementioned Hessenberg eigenvalue problem, thereby allowing to compute the associated simultaneous Gaussian quadrature rule in floating point arithmetic in a reliable way.

Based on the results described in [13], we develop here an algorithm for computing simultaneous Gaussian quadrature rules associated with different kinds of MOPs, for which the banded lower Hessenberg matrix was not totally nonnegative, and we describe the associated Matlab package, which requires only $\mathcal{O}(n^2)$ computational complexity and $\mathcal{O}(n)$ memory.

The paper is organized as follows. Notations are introduced in Section 2. The handled classes of MOPs are listed in Section 3. Moreover, the use of the Matlab function ClassMOP.m, generating the coefficients of the recurrence relations of the associated MOPs, is described in Section 4. The use of the function GaussMOP.m, computing the nodes and the weights of the chosen class of MOPs, is reported in Section 5, followed by the description of the proposed numerical method in Sections 6 and 7. Numerical tests are reported in Section 8, followed by the concluding remarks. Finally, the Matlab codes can be found in the Appendix.

2. Notations

Upper-case letters $A, B, \dots$, denote matrices and $A_{m,n}$, or simply A_m if $m = n$, denotes matrices of size (m, n). The entry (i, j) of a matrix A is denoted by $a_{i,j}$. Submatrices are denoted by the colon notation of Matlab, *i. e.*, $A(i : j, k : l)$ is the submatrix of A obtained by the intersection of rows i to j and columns k to l, and $A(i : j, :)$ and $A(:, k : l)$ are the rows of A from i to j and the columns of A from k to l, respectively.
Given $A \in \mathbb{R}^{n\times n}$ and $k \in \mathbb{Z}$, $-n + 1 \le k \le n - 1$, $\mathtt{triu}(A, k)$ denotes the matrix with elements on and above the kth diagonal of A.
Bold lower-case letters $\boldsymbol{x}, \boldsymbol{y}, \dots, \boldsymbol{\omega}, \dots$, denote vectors, and x_i denotes the ith element of the vector $\boldsymbol{x}$.
Lower-case letters $x, y, \dots, \lambda, \theta, \dots$, denote scalars.
The identity matrix of order n is denoted by I_n, and its ith column, $i = 1, \dots, n$, *i.e.*, the ith vector of the canonical basis of $\mathbb{R}^n$, is denoted by $\boldsymbol{e}_i$.
The zero vector of length n is denoted by $\boldsymbol{o}_n$.
The i-th subdiagonal of a matrix $H \in \mathbb{R}^{m\times n}$ is denoted by $\mathtt{diag}(H, -i)$.
The diagonal matrix with entries $d_1, \dots, d_n$ is denoted by $\mathtt{diag}(d_1, \dots, d_n)$.
The notation $\lfloor y \rfloor$ stands for the largest integer not exceeding $y \in \mathbb{R}_+$.
The notation $k \gg 0$ stands for $k \in \mathbb{N}$, with k very large.
If $x \in \mathbb{C}$, $\mathcal{R}(x)$ denotes the real part of x.
Numbers in scientific notation $a \times 10^b$, with $a, b \in \mathbb{R}$, are represented as $a(b)$ in Section 8.
A *flop* denotes a floating point operation (sum, subtraction, multiplication, division). The square root is considered a flop as well.

3. Classes of the multiple orthogonal polynomials

The considered classes of MOPs are listed below. For each of them, the corresponding weights $w^{(1)}(x)$ and $w^{(2)}(x)$, the integration intervals, the recurrence relations, and the coefficients (5), involved in the computation of the vectors of weights $\boldsymbol{\omega}^{(1)}$ and $\boldsymbol{\omega}^{(2)}$, are reported.

MOP$_1$: Multiple Jacobi–Piñeiro polynomials
Weights:

$$\left(w^{(1)}(x), w^{(2)}(x)\right) := (x^{\alpha_1}(1-x)^{\alpha_0}, x^{\alpha_2}(1-x)^{\alpha_0}),$$

$\alpha_j > -1,\ j =, 0, 1, 2,\ \ \alpha_1 - \alpha_2 \notin \mathbb{Z}$.
Interval: $\Delta^{(i)} = [0, 1],\ i = 1, 2.$

Recurrence relation coefficients:

$$
\begin{aligned}
b_0 \; &= (1+\alpha_1)/(2+\alpha_0+\alpha_1);\\
&\texttt{for}\, i = 1,2,\dots\\
b_{2i} \; &= \Big(36i^4 + (48\alpha_0 + 28\alpha_1 + 20\alpha_2 + 38)i^3 + (21\alpha_0^2 + 8\alpha_1^2 + 4\alpha_2^2 + 30\alpha_0\alpha_1\\
&+ 18\alpha_0\alpha_2 + 15\alpha_1\alpha_2 + 39\alpha_0 + 19\alpha_1 + 19\alpha_2 + 9)i^2 + (3\alpha_0^3 + 10\alpha_0^2\alpha_1\\
&+ 4\alpha_0^2\alpha_2 + 6\alpha_0\alpha_1^2 + 2\alpha_0\alpha_2^2 + 11\alpha_0\alpha_1\alpha_2 + 5\alpha_1^2\alpha_2 + 3\alpha_1\alpha_2^2 + 12\alpha_0^2 + 3\alpha_1^2\\
&+ 3\alpha_2^2 + 13\alpha_0\alpha_1 + 13\alpha_0\alpha_2 + 8\alpha_1\alpha_2 + 6\alpha_0 + 3\alpha_1 + 3\alpha_2)i + \alpha_0^2 + \alpha_0\alpha_1\\
&+ \alpha_2\alpha_1^2 + 2\alpha_2\alpha_1^2\alpha_0 + 2\alpha_0^2\alpha_1 + \alpha_1^2\alpha_0 + \alpha_2^2\alpha_0 + \alpha_2^2\alpha_1 + \alpha_0^3\alpha_1 + \alpha_0^2\alpha_1^2\\
&+ \alpha_2^2\alpha_0\alpha_1 + \alpha_2^2\alpha_1^2 + 2\alpha_2\alpha_0^2\alpha_1 + 3\alpha_2\alpha_1\alpha_0 + 2\alpha_2\alpha_0^2 + \alpha_1\alpha_2 + \alpha_0^3 + \alpha_0\alpha_2 \Big)\\
&\times\Big((3i+\alpha_0+\alpha_2)(3i+\alpha_0+\alpha_1)(3i+\alpha_0+\alpha_2+1)(3i+\alpha_0+\alpha_1+2)\Big)^{-1};
\end{aligned}
$$

$$
\begin{aligned}
&\texttt{for}\;\; i = 0,1,\dots\\
b_{2i+1} &= \Big(36i^4 + (48\alpha_0 + 20\alpha_1 + 28\alpha_2 + 106)i^3 + (21\alpha_0^2 + 4\alpha_1^2 + 8\alpha_2^2 + 18\alpha_0\alpha_1.\\
&+ 30\alpha_0\alpha_2 + 15\alpha_1\alpha_2 + 105\alpha_0 + 41\alpha_1 + 65\alpha_2 + 111)i^2 + (3\alpha_0^3 + 4\alpha_0^2\alpha_1\\
&+ 10\alpha_0^2\alpha_2 + 2\alpha_0\alpha_1^2 + 6\alpha_0\alpha_2^2 + 11\alpha_0\alpha_1\alpha_2 + 3\alpha_1^2\alpha_2 + 5\alpha_1\alpha_2^2 + 30\alpha_0^2 + 5\alpha_1^2\\
&+ 13\alpha_2^2 + 23\alpha_0\alpha_1 + 47\alpha_0\alpha_2 + 22\alpha_1\alpha_2 + 72\alpha_0 + 25\alpha_1 + 49\alpha_2 + 48)i\\
&+ 18\alpha_0\alpha_2 + 8\alpha_2\alpha_0^2 + 4\alpha_1 + 4\alpha_2^2\alpha_1 + 8\alpha_1\alpha_2 + 2\alpha_0^3 + 5\alpha_2^2\alpha_0 + 8\alpha_2\alpha_1\alpha_0\\
&+ 12\alpha_2 + 7 + 15\alpha_0 + \alpha_2^2\alpha_1^2 + 10\alpha_0^2 + 6\alpha_0\alpha_1 + 2\alpha_2\alpha_1^2 + 2\alpha_0^2\alpha_1 + \alpha_1^2\alpha_0\\
&+ 5\alpha_2^2 + \alpha_2\alpha_0^3 + \alpha_2^2\alpha_0^2 + \alpha_1^2 + \alpha_2\alpha_1^2\alpha_0 + 2\alpha_2\alpha_0^2\alpha_1 + 2\alpha_2^2\alpha_0\alpha_1 \Big)\\
&\times \Big((3i+\alpha_0+\alpha_2+1)(3i+\alpha_0+\alpha_1+2)(3i+\alpha_0+\alpha_2+3)(3i+\alpha_0+\alpha_1+3)\Big)^{-1};
\end{aligned}
$$

$$
\begin{aligned}
c_1 \; &= (1+\alpha_0)(1+\alpha_1)(3+\alpha_0+\alpha_1)^{-1}(2+\alpha_0+\alpha_1)^{-2};\\
&\texttt{for}\;\; i = 1,2,\dots\\
c_{2i} \; &= i(2i+\alpha_0)(2i+\alpha_0+\alpha_1)(2i+\alpha_0+\alpha_2)\,\Big(54i^4 + (63\alpha_0 + 45\alpha_1 + 45\alpha_2)i^3\\
&+ (24\alpha_0^2 + 8\alpha_1^2 + 8\alpha_2^2 + 42\alpha_0\alpha_1 + 42\alpha_0\alpha_2 + 44\alpha_1\alpha_2 - 8)i^2 + (3\alpha_0^3 + \alpha_1^3\\
&+ \alpha_2^3 + 12\alpha_0^2\alpha_1 + 12\alpha_0^2\alpha_2 + 3\alpha_0\alpha_1^2 + 3\alpha_0\alpha_2^2 + 33\alpha_0\alpha_1\alpha_2 + 8\alpha_1^2\alpha_2 + 8\alpha_1\alpha_2^2\\
&- 3\alpha_0 - 4\alpha_1 - 4\alpha_2)i + \alpha_0^3\alpha_1 + \alpha_0^3\alpha_2 + 6\alpha_0^2\alpha_1\alpha_2 + \alpha_1^3\alpha_2 + \alpha_1\alpha_2^3 + 3\alpha_0\alpha_1^2\alpha_2\\
&+ 3\alpha_0\alpha_1\alpha_2^2 - \alpha_0\alpha_1 - \alpha_0\alpha_2 - 2\alpha_1\alpha_2 \Big)\,\Big((3i+\alpha_0+\alpha_1+1)(3i+\alpha_0+\alpha_2+1)\Big)^{-1}\\
&\Big((3i+\alpha_0+\alpha_1)^2(3i+\alpha_0+\alpha_2)^2(3i+\alpha_0+\alpha_1-1)(3i+\alpha_0+\alpha_2-1)\Big)^{-1};\\
c_{2i+1} &= (2i+\alpha_0+1)(2i+\alpha_0+\alpha_1+1)(2i+\alpha_0+\alpha_2+1)\,\Big(54i^5 + (63\alpha_0 + 45\alpha_1\\
&+ 45\alpha_2 + 135)i^4 + (24\alpha_0^2 + 8\alpha_1^2 + 8\alpha_2^2 + 42\alpha_0\alpha_1 + 42\alpha_0\alpha_2 + 44\alpha_1\alpha_2 + 126\alpha_0\\
&+ 76\alpha_1 + 104\alpha_2 + 120)i^3 + (3\alpha_0^3 + \alpha_1^3 + \alpha_2^3 + 12\alpha_0^2\alpha_1 + 12\alpha_0^2\alpha_2 + 3\alpha_0\alpha_1^2\\
&+ 3\alpha_0\alpha_2^2 + 33\alpha_0\alpha_1\alpha_2 + 8\alpha_1^2\alpha_2 + 8\alpha_1\alpha_2^2 + 36\alpha_0^2 + 5\alpha_1^2 + 19\alpha_2^2 + 54\alpha_0\alpha_1\\
&+ 72\alpha_0\alpha_2 + 66\alpha_1\alpha_2 + 87\alpha_0 + 39\alpha_1 + 81\alpha_2 + 45)i^2 + (\alpha_0^3\alpha_1 + \alpha_0^3\alpha_2\\
&+ 6\alpha_0^2\alpha_1\alpha_2 + \alpha_1^3\alpha_2 + \alpha_1\alpha_2^3 + 3\alpha_0\alpha_1^2\alpha_2 + 3\alpha_0\alpha_1\alpha_2^2 + 3\alpha_0^3 + 2\alpha_2^3 + 12\alpha_0^2\alpha_1\\
&+ 12\alpha_0^2\alpha_2 + 6\alpha_0\alpha_2^2 + 33\alpha_0\alpha_1\alpha_2 + 5\alpha_1^2\alpha_2 + 11\alpha_1\alpha_2^2 + 18\alpha_0^2 + 20\alpha_0\alpha_1\\
&+ 38\alpha_0\alpha_2 + 14\alpha_2^2 + 26\alpha_1\alpha_2 + 24\alpha_0 + 6\alpha_1 + 24\alpha_2 + 6)i + \alpha_0^3\alpha_1 + 3\alpha_0^2\alpha_1\alpha_2\\
&+ 3\alpha_0\alpha_1\alpha_2^2 + \alpha_1\alpha_2^3 + \alpha_0^3 + \alpha_2^3 + 3\alpha_0^2\alpha_1 + 3\alpha_0^2\alpha_2 + 6\alpha_0\alpha_1\alpha_2 + 3\alpha_0\alpha_2^2\\
&+ 3\alpha_1\alpha_2^2 + 3\alpha_0^2 + 3\alpha_2^2 + 2\alpha_0\alpha_1 + 6\alpha_0\alpha_2 + 2\alpha_1\alpha_2 + 2\alpha_0 + 2\alpha_2 \Big)\\
&\times(3i+\alpha_0+\alpha_1+3)^{-1}(3i+\alpha_0+\alpha_2+2)^{-1}(3i+\alpha_0+\alpha_1+2)^{-2}\\
&\times(3i+\alpha_0+\alpha_2+1)^{-2}(3i+\alpha_0+\alpha_1+1)^{-1}(3i+\alpha_0+\alpha_2)^{-1};
\end{aligned}
$$

$$\begin{aligned}
&\texttt{for}\ \ i = 1,2,\dots\\
d_{2i} &= i(2i+\alpha_0)(2i+\alpha_0-1)(2i+\alpha_0+\alpha_1)(2i+\alpha_0+\alpha_1-1)(2i+\alpha_0+\alpha_2)\\
&\times(2i+\alpha_0+\alpha_2-1)(i+\alpha_1)(i+\alpha_1-\alpha_2)(3i+\alpha_0+\alpha_1+1)^{-1}\\
&\times(3i+\alpha_0+\alpha_1)^{-2}(3i+\alpha_0+\alpha_2)^{-1}(3i-1+\alpha_0+\alpha_1)^{-2}\\
&\times(3i+\alpha_0+\alpha_2-1)^{-1}(3i+\alpha_0+\alpha_1-2)^{-1}(3i+\alpha_0+\alpha_2-2)^{-1},\\
d_{2i+1} &= i(2i+\alpha_0+1)(2i+\alpha_0)(2i+\alpha_0+\alpha_1)(2i+\alpha_0+\alpha_1+1)\\
&\times(2i+\alpha_0+\alpha_2+1)(2i+\alpha_0+\alpha_2)(i+\alpha_2)(i+\alpha_2-\alpha_1)\\
&\times(3i+\alpha_0+\alpha_1+2)^{-1}(3i+\alpha_0+\alpha_2+2)^{-1}(3i+\alpha_0+\alpha_1+1)^{-1}\\
&\times(3i+1+\alpha_0+\alpha_2)^{-2}(3i+\alpha_0+\alpha_1)^{-1}(3i+\alpha_0+\alpha_2)^{-2}(3i+\alpha_0+\alpha_2-1)^{-1}.
\end{aligned}$$

Coefficients (5), involved in the computation of the weights:

$$\begin{aligned}
&f_{1,1} = \tfrac{\Gamma(1+\alpha_0)\Gamma(1+\alpha_1)}{\Gamma(2+\alpha_0+\alpha_1)},\\
&f_{2,1} = \tfrac{\Gamma(1+\alpha_0)\Gamma(1+\alpha_2)}{\Gamma(2+\alpha_0+\alpha_2)},\quad f_{2,2} = \big((1+\alpha_2)-(2+\alpha_0+\alpha_2)b_0\big)\,\tfrac{\Gamma(1+\alpha_0)\Gamma(1+\alpha_2)}{\Gamma(3+\alpha_0+\alpha_2)},
\end{aligned}$$

where Γ is the Gamma function [14, p. 255] defined as

$$\Gamma(z) = \int_0^\infty t^{z-1}e^{-t}dt, \qquad \mathcal{R}(z) > 0,$$

and computed by the `Matlab` function `gamma.m`.

References: [9,15].

MOP$_2$: Multiple Laguerre polynomials of first kind

Weights:

$$\big(w^{(1)}(x), w^{(2)}(x)\big) := (x^{\alpha_1}e^{-x}, x^{\alpha_2}e^{-x}),$$

$\alpha_j > -1,\ j = 1,2.$

Interval: $\Delta^{(i)} = [0,\infty),\ i = 1,2.$

Recurrence relation coefficients:

$$\begin{aligned}
&\texttt{for}\, i = 0,1,2,\dots\\
&\quad b_{2i} = 3i+\alpha_1+1,\\
&\quad b_{2i+1} = 3i+\alpha_2+2,\\
&\quad c_{2i} = i(3i+\alpha_1+\alpha_2),\\
&\quad c_{2i+1} = 3i^2+(\alpha_1+\alpha_2+3)i+\alpha_1+1,\\
&\quad d_{2i} = i(i+\alpha_1)(i+\alpha_1-\alpha_2),\\
&\quad d_{2i+1} = i(i+\alpha_2)(i+\alpha_2-\alpha_1).
\end{aligned}$$

Coefficients (5), involved in the computation of the weights:

$$\begin{aligned}
&f_{1,1} = \Gamma(1+\alpha_1),\\
&f_{2,1} = \Gamma(1+\alpha_2),\quad f_{2,2} = \Gamma(1+\alpha_2)(\alpha_2-\alpha_1).
\end{aligned}$$

References: [9,16,17].

MOP$_3$: Multiple Laguerre polynomials of second kind

Weights:

$$\big(w^{(1)}(x), w^{(2)}(x)\big) := (x^{\alpha_0}e^{-\alpha_1 x}, x^{\alpha_0}e^{-\alpha_2 x}),$$

$\alpha_0 > -1,\ \alpha_j > 0,\ j = 1,2,\ \alpha_1 \neq \alpha_2.$

Interval: $\Delta^{(i)} = [0,\infty),\ i = 1,2.$

Recurrence relation coefficients:

```
for $i = 0, 1, 2, \dots$
    $b_{2i} = \frac{i(\alpha_1 + 3\alpha_2) + (1 + \alpha_0)\alpha_2}{\alpha_1\alpha_2}$,
    $b_{2i+1} = \frac{i(3\alpha_1 + \alpha_2) + (2 + \alpha_0)\alpha_1 + \alpha_2}{\alpha_1\alpha_2}$,
    $c_{2i} = \frac{i(2i + \alpha_0)(\alpha_1^2 + \alpha_2^2)}{\alpha_1^2\alpha_2^2}$,
    $c_{2i+1} = \frac{2i^2(\alpha_1^2 + \alpha_2^2) + i\left(\alpha_1^2 + 3\alpha_2^2 + \alpha_0(\alpha_1^2 + \alpha_2^2)\right) + (1 + \alpha_0)\alpha_2^2}{\alpha_1^2\alpha_2^2}$,
    $d_{2i} = \frac{i(2i + \alpha_0)(2i + \alpha_0 - 1)(\alpha_2 - \alpha_1)}{\alpha_1^3\alpha_2}$,
    $d_{2i+1} = \frac{i(2i + \alpha_0)(2i + \alpha_0 + 1)(\alpha_1 - \alpha_2)}{\alpha_1\alpha_2^3}$.
```

Coefficients (5), involved in the computation of the weights:

$$f_{1,1} = \alpha_1^{-1-\alpha_0}\Gamma(1 + \alpha_0),$$
$$f_{2,1} = \alpha_2^{-1-\alpha_0}\Gamma(1 + \alpha_0), \quad f_{2,2} = \alpha_1^{-1}\alpha_2^{-2-\alpha 0}(\alpha_1 - \alpha_2)\Gamma(2 + \alpha_0).$$

References:[9,18].

MOP$_4$: Multiple Hermite polynomials

Weights:

$$\left(w^{(1)}(x), w^{(2)}(x)\right) := \left(e^{-x^2+\alpha_1 x}, e^{-x^2+\alpha_2 x}\right),$$

$\alpha_1, \alpha_2 \in \mathbb{R}, \;\; \alpha_1 \neq \alpha_2$.

Intervals: $\Delta^{(i)} = (-\infty, \infty), \; i = 1, 2$.

Recurrence relation coefficients:

```
for $i = 0, 1, 2, \dots$
    $b_{2i} = \alpha_1/2$,
    $b_{2i+1} = \alpha_2/2$,
    $c_i = i/2$,
    $d_{2i} = i(\alpha_1 - \alpha_2)/4$,
    $d_{2i+1} = i(\alpha_2 - \alpha_1)/4$.
```

Coefficients (5), involved in the computation of the weights:

$$f_{1,1} = e^{\frac{\alpha_1^2}{4}}\sqrt{\pi},$$
$$f_{2,1} = e^{\frac{\alpha_2^2}{4}}\sqrt{\pi}, \quad f_{2,2} = \tfrac{\alpha_2-\alpha_1}{2}e^{\frac{\alpha_2^2}{4}}\sqrt{\pi}.$$

References: [9].

MOP$_5$: Multiple Laguerre–Hermite polynomials

Weights:

$$\left(w^{(1)}(x), w^{(2)}(x)\right) := \left(e^{-x^2}|x|^{\beta}, e^{-x^2}x^{\beta}\right),$$

$\beta > -1$.

Intervals: $\Delta^{(1)} = (-\infty, 0], \; \Delta^{(2)} = [0, \infty)$.

Recurrence relation coefficients:

```
for $i = 0, 1, 2, \dots$
    $b_{2i} = X_i^{(\beta)}$,
    $b_{2i+1} = -X_i^{(\beta)}$,
    $c_{2i} = \frac{i}{2}$,
    $c_{2i+1} = \frac{2i+\beta+1}{2} - (X_i^{(\beta)})^2$
    $d_{2i} = \frac{i}{2} X_{i-1}^{(\beta)}$,
    $d_{2i+1} = -\frac{i}{2} X_i^{(\beta)}$
```

with $X_i^{(\beta)} = -\frac{\Gamma\left(\frac{i+\beta+2}{2}\right)}{\Gamma\left(\frac{i+\beta+1}{2}\right)}$. Moreover, for large i, $X_i^{(\beta)} = -\sqrt{\frac{\beta+i}{2}} + o(\sqrt{i})$ [9].

Coefficients (5), involved in the computation of the weights:

$$f_{1,1} = \frac{1}{2}\Gamma\left(\frac{1+\beta}{2}\right),$$

$$f_{2,1} = \frac{1}{2}\Gamma\left(\frac{1+\beta}{2}\right), \quad f_{2,2} = \frac{1}{2}\left(-b_0\Gamma\left(\frac{1+\beta}{2}\right) + \Gamma\left(\frac{2+\beta}{2}\right)\right).$$

References: [9,17].

MOP$_6$: MOPs Associated with the Modified Bessel function of the second kind (Macdonald function) $K_\nu(x)$

Weights:

$$\left(w^{(1)}(x), w^{(2)}(x)\right) := \left(2x^{\alpha+\nu/2}K_\nu(2\sqrt{x}), 2x^{\alpha+(\nu+1)/2}K_{\nu+1}(2\sqrt{x})\right),$$

$\alpha > -1,\ \nu \geq 0$, with $K_\nu(x)$ the modified Bessel function of the second kind given by

$$K_\nu(x) := \frac{1}{2}\left(\frac{x}{2}\right)^\nu \int_0^\infty exp\left(-t - \frac{x^2}{4t}\right) t^{-\nu-1} dt.$$

Interval: $\Delta^{(i)} = [0, \infty), i = 1, 2$.

Recurrence relation coefficients:

```
for $i = 0, 1, 2, \dots$
    $b_i = i(3i + \alpha + 2\nu) + (\alpha + 1)(3i + \alpha + \nu + 1)$,
    $c_i = i(i + \alpha)(i + \alpha + \nu)(3i + 2\alpha + \nu)$,
    $d_i = i(i - 1)(i + \alpha)(i + \alpha - 1)(i + \alpha + \nu)(i + \alpha + \nu - 1)$.
```

Coefficients (5), involved in the computation of the weights:

$$f_{1,1} = \Gamma(\alpha + 1)\Gamma(\alpha + \nu + 1),$$

$$f_{2,1} = \Gamma(\alpha + 1)\Gamma(\alpha + \nu + 2), \quad f_{2,2} = \Gamma(\alpha + 2)\Gamma(\alpha + \nu + 2).$$

References: [5,19,20].

MOP$_7$: MOPs associated with the Modified Bessel functions of the first kind $I_\nu(x)$

Weights:

$$\left(w^{(1)}(x), w^{(2)}(x)\right) := \left(x^{\nu/2}I_\nu(2\sqrt{x})e^{-\beta x}, x^{(\nu+1)/2}I_{\nu+1}(2\sqrt{x})e^{-\beta x}\right),$$

$\beta > 0,\ \nu > -1$ where $I_\nu(x)$ is the modified Bessel function of the first kind [21],

$$I_\nu(x) = \left(\frac{z}{2}\right)^\nu \sum_{k=0}^\infty \frac{\left(\frac{z}{2}\right)^{2k}}{k!\Gamma(\nu + k + 1)}, \quad \nu \in \mathbb{R}.$$

Interval: $\Delta^{(i)} = [0, \infty), i = 1, 2$.

Recurrence relation coefficients:

```
for $i = 0, 1, 2, \dots$
    $b_i = (1 + \beta(\nu + 2i + 1))/\beta^2$,
    $c_i = i(2 + \beta(\nu + i))/\beta^3$,
    $d_i = i(i - 1)/\beta^4$.
```

Coefficients (5), involved in the computation of the weights:

$$f_{1,1} = \beta^{-1-\nu} e^{\frac{1}{\beta}},$$
$$f_{2,1} = \beta^{-2-\nu} e^{\frac{1}{\beta}}, \quad f_{2,2} = \beta^{-3-\nu} e^{\frac{1}{\beta}}.$$

References: [5,21].

MOP$_8$: MOPs associated with the Gauss' hypergeometric function
Weights:

$$\begin{aligned} w^{(1)}(x) &:= \tfrac{\Gamma(c)\Gamma(d)}{\Gamma(a)\Gamma(b)\Gamma(\delta)} x^{a-1}(1-x)^{\delta-1}\ {}_2F_1\left(\begin{bmatrix} c-b \\ d-b \end{bmatrix}, \delta, 1-x\right), \\ w^{(2)}(x) &:= \tfrac{\Gamma(c+1)\Gamma(d)}{\Gamma(a)\Gamma(b+1)\Gamma(\delta)} x^{a-1}(1-x)^{\delta-1}\ {}_2F_1\left(\begin{bmatrix} c-b \\ d-b-1 \end{bmatrix}, \delta, 1-x\right), \end{aligned}$$

with $a, b, c, d \in \mathbb{R}^+,\ c+1, d > a,\ c, d > b,\ \delta = c+d-b-a > 0$, and

$${}_2F_1\left(\begin{bmatrix} \alpha \\ \beta \end{bmatrix}, \gamma, z\right) = \sum_{i=0}^{\infty} \frac{(\alpha)_i(\beta)_i}{(\gamma)_i} \frac{z^n}{n!}$$

is the Gauss' hypergeometric function, computed by the `Matlab` function `hypergeom.m`, and $(\mu)_i$ denotes the Pochhammer symbol defined by

$$\begin{aligned} (\mu)_0 &:= 0, \\ (\mu)_i &:= \mu(\mu+1)\cdots(\mu+i-1), \quad i \in \mathbb{Z}^+. \end{aligned}$$

Interval: $\Delta^{(i)} = [0,1], i = 1,2.$
Recurrence relation coefficients:
Let

$$\begin{aligned} \lambda_{3i} &= \frac{i(a+i-1)(c'_i-b-1)}{(c'_i+i-2)(c'_i+i-1)(c'_{i+1}+i-2)}, \\ \lambda_{3i+1} &= \frac{i(b+i)(c'_{i+1}-a-1)}{(c'_i+i-1)(c'_{i+1}+i-2)(c'_{i+1}+i-1)}, \\ \lambda_{3i+2} &= \frac{(a+i)(b+i)(c'_i-1)}{(c'_i+i-1)(c'_i+i)(c'_{i+1}+i-1)}, \end{aligned}$$

with

$$c'_i = \begin{cases} c+k, & \text{if } i = 2k-1, \\ d+k, & \text{if } i = 2k. \end{cases}$$

Then

```
for $i = 0, 1, 2, \dots$
    $b_i = \lambda_{3i}\lambda_{3i+1}\lambda_{3i+2}$,
    $c_i = \lambda_{3i+1}\lambda_{3i+3} + \lambda_{3i+2}\lambda_{3i+3} + \lambda_{3i+2}\lambda_{3i+4}$,
    $d_i = \lambda_{3i+2}\lambda_{3i+4}\lambda_{3i+6}$.
```

Coefficients (5), involved in the computation of the weights:

$$f_{1,1} = 1,$$
$$f_{2,1} = 1, \quad f_{2,2} = \tfrac{a(c-b)}{cd(c+1)}.$$

References: [22].

MOP$_9$: MOPs associated with the confluent hypergeometric function
Weights:

$$\begin{aligned} w^{(1)}(x) &:= \frac{\Gamma(c)}{\Gamma(a)\Gamma(b)} e^{-x} x^{a-1} U(c-b, a-b+1, x), \\ w^{(2)}(x) &:= \frac{\Gamma(c+1)}{\Gamma(a)\Gamma(b)} e^{-x} x^{a-1} U(c-b+1, a-b+1, x), \end{aligned}$$

where $U(a,b,z)$ is the confluent hypergeometric function [14, p.504–505], computed by the `Matlab` function `kummerU.m`,

$$U(a,b,z) = \frac{1}{\Gamma(a)} \int_0^{\infty} e^{-zt} t^{a-1}(1+t)^{b-a-1} dt, \qquad \mathcal{R}(z) > 0, \mathcal{R}(a) > 0,$$

$a, b, c \in \mathbb{R}^+,\ c > \max\{a, b\}.$

Interval: $\Delta^{(i)} = [0,\infty), i = 1,2$.
Recurrence relation coefficients:

$$\begin{aligned}
&\texttt{for}\, i = 0,1,2,\dots\\
b_{2i} &= \frac{(2i+1)(a+2i)(b+2i)}{c+3i} - \frac{2i(a+2i-1)(b+2i-1)}{c+3i-1},\\
b_{2i+1} &= \frac{(2i+2)(a+2i+1)(b+2i+1)}{c+3i+2} - \frac{(2i+1)(a+2i)(b+2i)}{c+3i},\\
c_{2i} &= \frac{2i(a+2i-1)(b+2i-1)}{c+3i-1}\left(\frac{(2i-1)(a+2i-2)(b+2i-2)}{2(c+3i-2)}\right.\\
&\left. - \frac{2i(a+2i-1)(b+2i-1)}{c+3i-1} + \frac{(2i+1)(a+2i)(b+2i)}{2(c+3i)}\right),\\
c_{2i+1} &= \frac{(2i+1)(a+2i)(b+2i)}{c+3i}\left(\frac{i(a+2i-1)(b+2i-1)}{(c+3i-1)}\right.\\
&\left. - \frac{(2i+1)(a+2i)(b+2i)}{(c+3i)} + \frac{(i+1)(a+2i+1)(b+2i+1)}{c+3i+1}\right),\\
d_{2i} &= \frac{2i(2i+1)(a+2i-1)(a+2i)(b+2i-1)(b+2i)(c+i-1)}{(c+3i-2)(c+3i-1)(c+3i)(c+3i-1)(c+3i)(c+3i+1)}\\
&\times (c-a+i)(c-b+i)\\
d_{2i+1} &= \frac{(2i+1)(2i+2)(a+2i)(a+2i+1)(b+2i)(b+2i+1)}{(c+3i)(c+3i+1)(c+3i+2)}.
\end{aligned}$$

Coefficients (5), involved in the computation of the weights:

$$\begin{aligned}
&f_{1,1} = 1,\\
&f_{2,1} = 1, \quad f_{2,2} = -\tfrac{ab}{c(c+1)}.
\end{aligned}$$

References: [23].

In the sequel, we denote the k-th MOP by $\mathbf{MOP}_k$, for $k \in \{1,2,\dots,9\}$.

4. Matlab **function** ClassMOP.m

Here, the Matlab function ClassMOP.m for computing the coefficients of the recurrence relation associated with different classes of MOPs is described.

The Matlab command is

$[\boldsymbol{b},\boldsymbol{c},\boldsymbol{d},F]$=ClassMOP(IC, n, $\boldsymbol{\alpha}$)

The input parameters of ClassMOP.m are:

- IC: kind of MOPs.
- n: number of nodes of the simultaneous Gaussian quadrature rule.
- $\boldsymbol{\alpha}$: vector of parameters characterizing the weights.

The output parameters of ClassMOP are:

- $\boldsymbol{b}$: the main diagonal of H_n.
- $\boldsymbol{c}$: the first subdiagonal of H_n.
- $\boldsymbol{d}$: the second subdiagonal of H_n.
- F : the 2×2 lower triangular matrix in (5).

The list of input parameters for the considered classes of MOPs is summarized in Table 1.

5. Matlab **function** GaussMOP.m

Here, the Matlab function GaussMOP.m for computing the simultaneous Gaussian quadrature rule associated with different kinds of weights, is reported.

The Matlab command for computing the simultaneous Gaussian quadrature rule is

$[\boldsymbol{x},\boldsymbol{\omega}^{(1)},\boldsymbol{\omega}^{(2)}$, ier]=GaussMOP($\boldsymbol{b},\boldsymbol{c},\boldsymbol{d},n,F$)

The input parameters of GaussMOP.m are:

- $\boldsymbol{b}$: the main diagonal of H_n.
- $\boldsymbol{c}$: the first subdiagonal of H_n.
- $\boldsymbol{d}$: the second subdiagonal of H_n.
- n: number of nodes of the simultaneous Gaussian quadrature rule.

Table 1
Input parameters of the Matlab function ClassMOP.m.

MOP	# nodes	α
1	n	$[\alpha_0,\ \alpha_1,\ \alpha_2]^T,\quad \alpha_j > -1,\ j = 0,1,2,\ \ \alpha_1 - \alpha_2 \notin \mathbb{Z}$
2	n	$[\alpha_1,\ \alpha_2]^T,\quad \alpha_j > -1,\ j = 1,2$
3	n	$[\alpha_0,\ \alpha_1,\ \alpha_2]^T,\ \alpha_0 > -1,\ \alpha_j > 0,\ j = 1,2,\ \alpha_1 \neq \alpha_2.$
4	n	$[\alpha_1,\ \alpha_2]^T,\ \alpha_1, \alpha_2 \in \mathbb{R},\ \alpha_1 \neq \alpha_2$
5	n	$[\beta]^T,\ \beta > -1$
6	n	$[\alpha,\ \nu]^T,\ \alpha > -1,\ \nu \geq 0$
7	n	$[\beta,\ \nu]^T,\ \beta > 0,\ \nu \geq -1$
8	n	$[a,\ b,\ c,\ d]^T,\ a,b,c,d \in \mathbb{R}^+,\ c+1, d > a,\ c,d > b.$
9	n	$[a,\ b,\ c]^T,\ a,b,c \in \mathbb{R}^+,\ c > \max\{a,b\}.$

- F : the 2×2 lower triangular matrix in (5).

The output parameters of GaussMOP.m are:

- $\boldsymbol{x}$: vector of nodes of the simultaneous Gauss quadrature rule.
- $\boldsymbol{\omega}^{(1)}$: vector of weights of the simultaneous Gaussian quadrature rule corresponding to $w^{(1)}(x)$.
- $\boldsymbol{\omega}^{(2)}$: vector of weights of the simultaneous Gaussian quadrature rule corresponding to $w^{(2)}(x)$.
- ier: is set to zero for normal return, otherwise ier is set to j if the j-th node has not been determined after 30 iterations.

The algorithm is summarized in the following steps, described in detail in Sections 6 and 7, respectively:

- Transformation of H_n to the similar matrix $\hat{H}_n := S_n^{-1} H_n S_n$.
- Computation of the simultaneous Gaussian quadrature rule.

6. Transformation of H_n to the similar matrix $\hat{H}_n := S_n^{-1} H_n S_n$

As described in Section 1, the computation of simultaneous Gaussian quadrature rules associated with MOPs relies on that of the eigenvalues x_j and corresponding left and right eigenvectors $\boldsymbol{u}^{(j)}$ and $\boldsymbol{v}^{(j)}$, $j = 1, \dots, n$, respectively, of H_n. As noticed in [5], the latter eigenvalue problem is very ill-conditioned for all classes of MOPs listed in Section 3. Furthermore, the Matlab function balance.m, commonly applied to reduce the eigenvalue condition number, does not improve the condition of the considered eigenvalue problem, and, therefore, the Matlab function eig.m does not yield reliable results [5,13]. Indeed, in most cases, the eigenvalues computed by eig.m are complex conjugate, while the MOPs have only real zeros.

To reduce the eigenvalue condition number of the Hessenberg matrix H_n, a new diagonal balancing procedure has been recently introduced in [13], whose main idea consists of transforming H_n into a similar matrix

$$\hat{H}_n := S_n^{-1} H_n S_n \tag{6}$$

having the same Hessenberg structure, with $S_n = \mathtt{diag}(s_1, \dots, s_n)$ a diagonal matrix, such that $\mathtt{triu}(\hat{H}_n, -1)$ is symmetric.

After this similarity transformation, the condition of the eigenvalue problem for $\hat{H}_n$ is drastically reduced with respect to that of H_n, as shown in Fig. 1, where the condition numbers of the eigenvalues of H_n and $\hat{H}_n$, with $n = 20$ and parameters listed in Table 2 for all the considered MOPs, are displayed. Observe that the entries of the diagonal balancing matrix $S_n = \mathtt{diag}(s_1, \dots, s_n)$ in (6), introduced in [13], grow as the factorial function, *i.e.*, $s_i \sim i!$ (see Table 3, fourth column), for most MOPs. Hence, for i large enough, *e.g.*, $i > 170$, $s_i > (2 - 2^{-52})2^{1023}$, the largest finite floating-point number in IEEE® double precision. Therefore, in such cases, the computation of s_i yields Inf in Matlab.

To overcome this drawback, we consider here the same balancing technique applied to the matrix H_n, but without explicitly computing the matrix S_n. Observe that we need to compute only s_1 and s_2, *i.e.*, the first two entries in the main diagonal of S_n, since they are involved in the computation of the weights of the simultaneous Gaussian quadrature rules (11). The Matlab function DscaleS2.m, implementing the new balancing technique, is displayed in Table 8 in the Appendix.

Since the considered MOPs are monic, *i.e.*, $a_i = 1, i = 0, 1, \dots, n-1$, then [13]

$$s_i = \sqrt{c_1 c_2 \cdots c_i}, \quad i = 1, \dots, n. \tag{7}$$

Furthermore, denoted by

$$\begin{aligned} \hat{c}_i &= \hat{h}_{i+1,i}, \quad i = 1, \dots, n-1, \\ \hat{d}_i &= \hat{h}_{i+1,i-1}, \quad i = 2, \dots, n-1, \end{aligned}$$

defined $s_0 \equiv 1$, it turns out [13],

$$\hat{c}_i = c_i \frac{s_{i-1}}{s_i}, \quad i = 1, \dots, n-1, \tag{8}$$

Table 2
MOPs parameters used for computing the eigenvalue condition number of H_{20} and $\hat{H}_{20}$, shown in Fig. 1.

IC	α
1	$[-0.5,\ -0.2,\ 0.4]^T$
2	$[-0.5,\ 0.5]^T$
3	$[-0.5,\ 0.2,\ 0.4]^T$
4	$[0.2,\ 0.5]^T$
5	$[0.5]^T$
6	$[-0.5,\ 0.5]^T$
7	$[0.5,\ -0.5]^T$
8	$[1,\ 1,\ 3,\ 2]^T$
9	$[3,\ 2.5,\ 7.5]^T$

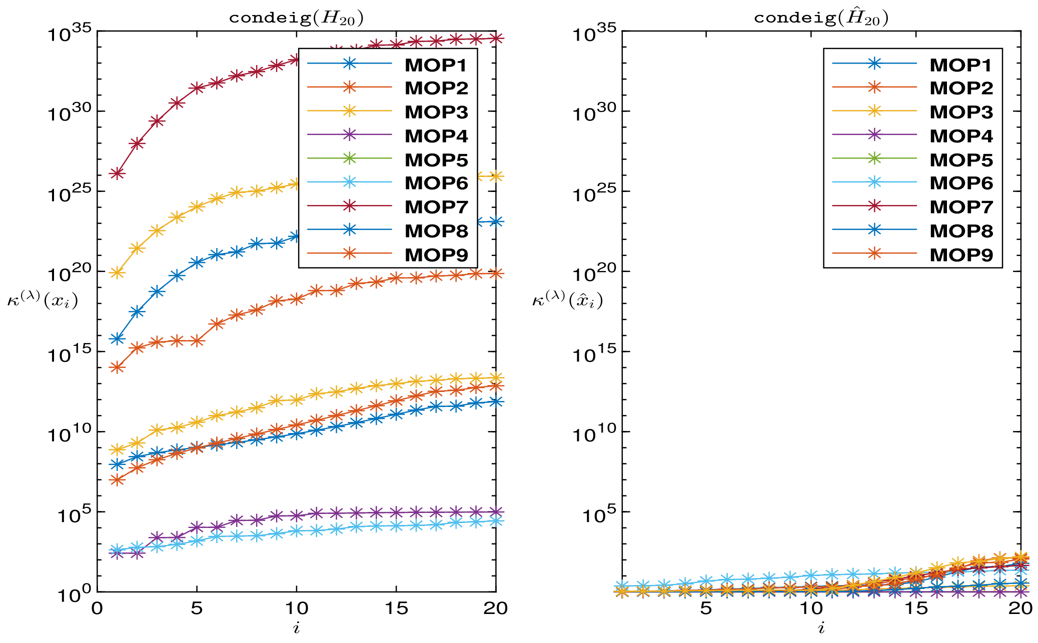

Fig. 1. Condition number of the eigenvalues of H_{20} (left) and of the eigenvalues of $\hat{H}_{20}$ (right) for all the considered MOPs.

$$\hat{d}_i = d_i \frac{s_{i-2}}{s_i}, \quad i = 2, \dots, n-1. \tag{9}$$

For each MOP in Section 3, the asymptotic behavior of c_i and d_i, respectively denoted by $c_{i\gg 0}$ and $d_{i\gg 0}$, can be represented as the product of a constant times a power of i, *i.e.*,

$$c_{i\gg 0} = \theta_c i^{\rho_c} \text{ and } d_{i\gg 0} = \theta_d i^{\rho_d}, \quad \theta_c, \theta_d, \rho_c, \rho_d \in \mathbb{R},$$

except for **MOP**$_3$, **MOP**$_4$, **MOP**$_5$, where the even and odd cases need to be distinguished. Moreover, $c_{i\gg 0}$ and $d_{i\gg 0}$ depend on α_1 and α_2 for **MOP**$_3$ and **MOP**$_4$, and on β for **MOP**$_7$.

The following Lemma holds.

Lemma 1. *Let c_i, d_i, $\hat{c}_i$, $\hat{d}_i$, $i = 0, 1, \dots, n-1$, be, respectively, the entries of the first and second subdiagonal of H_n and the entries of the first and second subdiagonal of $\hat{H}_n = S_n^{-1} H_n S_n$. Then*

$$\hat{c}_{i\gg 0} = \sqrt{c_i}, \qquad \hat{d}_{i\gg 0} = \frac{d_i}{c_i}. \tag{10}$$

Proof. By (7),

$$s_i = s_{i-1}\sqrt{c_i} = s_{i-1}\sqrt{\theta_c i^{\rho_c}}.$$

Table 3
Asymptotic behavior of the coefficients c_i, d_i, $\hat{c}_i$, $\hat{d}_i$, $i = 0, 1, \ldots, n-1$, and s_i, $i = 1, \ldots, n$..

MOP	$c_{i\gg 0}$	$d_{i\gg 0}$	$s_{i\gg 0}$	$\hat{c}_{i\gg 0}$	$\hat{d}_{i\gg 0}$
MOP$_1$	$3\left(\frac{4}{27}\right)^2$	$\left(\frac{4}{27}\right)^3$	$3^{i/2}\left(\frac{4}{27}\right)^i$	$\sqrt{3}\left(\frac{4}{27}\right)$	$\frac{1}{3}\,\frac{4}{27}$
MOP$_2$	$3\left(\frac{i}{2}\right)^2$	$\left(\frac{i}{2}\right)^3$	$\left(\frac{\sqrt{3}}{2}\right)^i i!$	$\frac{\sqrt{3}}{2}i$	$\frac{i}{6}$
MOP$_3$	$\frac{i^2}{2}\,\frac{\alpha_1^2+\alpha_2^2}{\alpha_1^2\alpha_2^2}$	$\frac{(-1)^i i^3(\alpha_2-\alpha_1)}{2\left(\alpha_1^{2+(-1)^i}\alpha_2^{2-(-1)^i}\right)}$	$\left(\frac{\alpha_1^2+\alpha_2^2}{2\alpha_1^2\alpha_2^2}\right)^{\frac{i}{2}} i!$	$\sqrt{\frac{\alpha_1^2+\alpha_2^2}{2(\alpha_1^2\alpha_2^2)}i}$	$\frac{(-1)^i i(\alpha_2-\alpha_1)}{\alpha_1^{(-1)^i}\alpha_2^{(-1)^{i+1}}(\alpha_1^2+\alpha_2^2)}$
MOP$_4$	$\frac{i}{2}$	$\frac{(-1)^i i(\alpha_1-\alpha_2)}{8}$	$\sqrt{\frac{i!}{2^i}}$	$\sqrt{\frac{i}{2}}$	$\frac{(-1)^i(\alpha_1-\alpha_2)}{4}$
MOP$_5$	$\frac{i}{4}$	$(-1)^{i+1}\frac{\sqrt{i^3}}{8}$	$\sqrt{\frac{i!}{4^i}}$	$\sqrt{\frac{i}{4}}$	$(-1)^{i+1}\sqrt{\frac{i}{4}}$
MOP$_6$	$3i^4$	i^6	$3^{\frac{i}{2}}(i!)^2$	$\sqrt{3}i^2$	$\frac{1}{3}i^2$
MOP$_7$	$\frac{i^2}{\beta^2}$	$\frac{i^2}{\beta^4}$	$\frac{i!}{\beta^i}$	i	$\frac{i}{\beta^2}$
MOP$_8$	$3\left(\frac{4}{27}\right)^2$	$\left(\frac{4}{27}\right)^3$	$\left(\sqrt{3}\,\frac{4}{27}\right)^i$	$\sqrt{3}\,\frac{4}{27}$	$\frac{1}{3}\,\frac{4}{27}$
MOP$_9$	$\frac{280}{81}i^2$	$3^{-3-3\,\frac{1+(-1)^i}{2}}\,2^6 i^3$	$\left(\frac{280}{81}\right)^{\frac{i}{2}} i!$	$\sqrt{\frac{280}{81}i}$	$\frac{1}{35}3^{-3\,\frac{1+(-1)^i}{2}}\,2^3 i$

Then

$$s_i = \theta_c^{\frac{i}{2}}\, i!^{\frac{\rho_c}{2}}.$$

Moreover, by (8),

$$\hat{c}_{i\gg 0} = c_i \frac{s_{i-1}}{s_i} = \theta_c i^{\rho_c} \frac{\theta_c^{\frac{i-1}{2}}(i-1)!^{\frac{\rho_c}{2}}}{\theta_c^{\frac{i}{2}}\, i!^{\frac{\rho_c}{2}}} = \sqrt{\theta_c i^{\rho_c}} = \sqrt{c_{i\gg 0}},$$

and, by (9),

$$\hat{d}_{i\gg 0} = d_i \frac{s_{i-2}}{s_i} = \theta_d i^{\rho_d} \frac{\theta_c^{\frac{i-2}{2}}(i-2)!^{\frac{\rho_c}{2}}}{\theta_c^{\frac{i}{2}}\, i!^{\frac{\rho_c}{2}}} = \frac{\theta_d}{\theta_c} i^{\rho_d-\rho_c} = \frac{d_{i\gg 0}}{c_{i\gg 0}}. \quad \square$$

The asymptotic behavior of the coefficients c_i, d_i, $\hat{c}_i$, $\hat{d}_i$, $i = 0, 1, \ldots, n-1$, and s_i, $i = 1, \ldots, n$, are displayed in Table 3 for all the considered MOPs. We observe that, except for **MOP**$_1$ and **MOP**$_8$, whose coefficients $c_{i\gg 0}, d_{i\gg 0}, \hat{c}_{i\gg 0}$ and $\hat{d}_{i\gg 0}$ are independent of i,

$$\lim_{i\to\infty} \frac{\hat{c}_{i\gg 0}}{c_{i\gg 0}} = 0, \quad \lim_{i\to\infty} \frac{\hat{d}_{i\gg 0}}{d_{i\gg 0}} = 0,$$

and

$$c_{i\gg 0} < |d_{i\gg 0}|, \quad \hat{c}_{i\gg 0} > |\hat{d}_{i\gg 0}|.$$

Therefore, the new balancing technique has the effect of reducing the size of the entries of the second lower subdiagonal of the Hessenberg matrix $\hat{H}_n$. As a consequence, the condition numbers of the eigenvalues of $\hat{H}_n$ are drastically reduced with respect to those of H_n (see Fig. 1).

7. Computation of the simultaneous Gaussian quadrature rule

By Theorem 1, the eigenvalues of H_n and the associated left and right eigenvectors are needed for computing the simultaneous Gaussian quadrature rule associated with MOPs. Let $\hat{x}_j$, $\hat{\boldsymbol{u}}^{(j)}$, and $\hat{\boldsymbol{v}}^{(j)}$, $j = 1, \ldots, n$, be the eigenvalues and corresponding left and right eigenvectors of $\hat{H}_n$, *i.e.*,

$$\hat{H}_n \hat{V}_n = \hat{V}_n \hat{X}_n, \quad \hat{U}_n^T \hat{H}_n = \hat{X}_n \hat{U}_n^T,$$

with

$$\hat{X}_n = \mathrm{diag}(\hat{x}_1, \ldots, \hat{x}_n),\ \hat{V}_n = [\hat{\boldsymbol{v}}^{(1)}, \ldots, \hat{\boldsymbol{v}}^{(n)}],\ \text{and}\ \hat{U}_n = [\hat{\boldsymbol{u}}^{(1)}, \ldots, \hat{\boldsymbol{u}}^{(n)}].$$

All the eigenvalues and corresponding left and right eigenvectors of H_n can be computed applying the `Matlab` function `eig.m` to the better conditioned matrix $\hat{H}_n$, with $\mathcal{O}(n^3)$ computational complexity and $\mathcal{O}(n^2)$ memory. Since $\hat{H}_n = S_n^{-1} H_n S_n$, then $U_n = S_n^{-1}\hat{U}_n$, and $V_n = S_n \hat{V}_n$.

By (4), for $j = 1, \dots, n$,

$$\begin{aligned}\omega_j^{(1)} &= \frac{v_1^{(j)} f_{1,1} u_1^{(j)}}{\boldsymbol{u}^{(j)^T} \boldsymbol{v}^{(j)}} = \frac{\hat{v}_1^{(j)} f_{1,1} \hat{u}_1^{(j)}}{\hat{\boldsymbol{u}}^{(j)^T} \hat{\boldsymbol{v}}^{(j)}},\\ \omega_j^{(2)} &= \frac{v_1^{(j)} \left(f_{2,1} u_1^{(j)} + f_{2,2} u_2^{(j)} \right)}{\boldsymbol{u}^{(j)^T} \boldsymbol{v}^{(j)}} = \frac{s_1 \hat{v}_1^{(j)} \left(f_{2,1} s_1^{-1} \hat{u}_1^{(j)} + f_{2,2} s_2^{-1} \hat{u}_2^{(j)} \right)}{\hat{\boldsymbol{u}}^{(j)^T} \hat{\boldsymbol{v}}^{(j)}}.\end{aligned} \tag{11}$$

Hence, the computation of the nodes and the set of weights of the simultaneous Gaussian quadrature rule relies only on the eigenvalues of $\hat{H}_n$ and corresponding left and right normalized eigenvectors, on $f_{1,1}, f_{2,1}$ and $f_{2,2}$ (5), and on the first two diagonal entries of S_n.

Here, we propose a more efficient approach to compute the eigenvalues and corresponding left and right eigenvectors of $\hat{H}_n$, with $\mathcal{O}(n^2)$ computational complexity and $\mathcal{O}(n)$ memory, based on the Ehrlich–Aberth method [24–26].

Given an initial guess $\hat{\boldsymbol{x}}^{(0)^T} = \left[\ \hat{x}_1^{(0)},\ \hat{x}_2^{(0)},\ \cdots, \hat{x}_{n-1}^{(0)},\ \hat{x}_n^{(0)}\ \right]^T$ of all the eigenvalues of $\hat{H}_n$, they can be simultaneously computed by the Ehrlich–Aberth iteration [24–26],

$$\hat{x}_j^{(\ell+1)} = \hat{x}_j^{(\ell)} - \frac{\dfrac{\hat{p}_n(\hat{x}_j^{(\ell)})}{\hat{p}_n'(\hat{x}_j^{(\ell)})}}{1 - \dfrac{\hat{p}_n(\hat{x}_j^{(\ell)})}{\hat{p}_n'(\hat{x}_j^{(\ell)})} \displaystyle\sum_{\substack{k=1\\k\neq j}}^{n} \frac{1}{\hat{x}_j^{(\ell)} - \hat{x}_k^{(\ell)}}}, \qquad j = 1, \dots, n,$$

where $\hat{p}_i(x),\ i = 0, 1 \dots, n$, are non-monic MOPs satisfying the recurrence relation whose coefficients are those of the balanced matrix $\hat{H}_{n,n+1}$,

$$\begin{cases} \hat{p}_{-2}(x) = 0, \\ \hat{p}_{-1}(x) = 0, & i = 0, \dots, n-1, \\ \hat{c}_{i-1}\hat{p}_{i+1}(x) = (x - \hat{b}_i)\hat{p}_i(x) - \hat{c}_i \hat{p}_{i-1}(x) - \hat{d}_i \hat{p}_{i-2}(x), \end{cases} \tag{12}$$

with $\hat{c}_i = \hat{a}_{i-1},\ i = 1, \dots, n-1$. Hence, $\hat{p}_n(\hat{x}_j^{(i)})$ and $\hat{p}_n'(\hat{x}_j^{(i)})$ need to be computed at iteration $\ell + 1$ of the Ehrlich–Aberth method.

To this end, (12) is written in matrix form, obtaining

$$\hat{F}_n(x)\hat{\boldsymbol{p}}_n(x) := \left(\hat{H}_{n,n+1} - x[I_n|\boldsymbol{o}_n]\right) \hat{\boldsymbol{p}}_n(x)$$
$$= \begin{bmatrix} \hat{b}_0 - x & \hat{a}_0 & 0 & 0 & 0 & \dots & 0 & 0 \\ \hat{c}_1 & \hat{b}_1 - x & \hat{a}_1 & 0 & 0 & \dots & 0 & 0 \\ \hat{d}_2 & \hat{c}_2 & \hat{b}_2 - x & \hat{a}_2 & 0 & \dots & 0 & 0 \\ 0 & \hat{d}_3 & \hat{c}_3 & \hat{b}_3 - x & \hat{a}_3 & \dots & 0 & 0 \\ \vdots & & \ddots & \ddots & \ddots & \ddots & \vdots & \vdots \\ 0 & \dots & 0 & \hat{d}_{n-2} & \hat{c}_{n-2} & \hat{b}_{n-2} - x & \hat{a}_{n-2} & 0 \\ 0 & \dots & 0 & 0 & \hat{d}_{n-1} & \hat{c}_{n-1} & \hat{b}_{n-1} - x & \hat{a}_{n-1} \end{bmatrix} \begin{bmatrix} \hat{p}_0(x) \\ \hat{p}_1(x) \\ \hat{p}_2(x) \\ \hat{p}_3(x) \\ \vdots \\ \hat{p}_{n-1}(x) \\ \hat{p}_n(x) \end{bmatrix} = \boldsymbol{o}_n.$$

Therefore, $\hat{\boldsymbol{p}}_n(x)$ belongs to the null-space of $\hat{F}_n(x)$, a full row rank matrix. Such a vector is computed by applying a sequence of n Givens rotations, $G_i \in \mathbb{R}^{(n+1)\times(n+1)},\ i = 1, \dots, n$, to the right of $\hat{F}_n(x)$ [13],

$$\hat{F}_n(x)Q_n = [L_n,\ \boldsymbol{o}_n],$$

with $L_n \in \mathbb{R}^{n\times n}$ lower triangular, and $Q_n = G_1^T G_2^T \cdots G_n^T$. Then, the last column of Q_n spans the null-space of $\hat{F}_n(x)$.

In order to compute $\hat{p}_n'(\tilde{x})$, we differentiate (13), obtaining

$$\hat{F}_n(x)\hat{\boldsymbol{p}}_n'(x) + \hat{F}'(x)\hat{\boldsymbol{p}}_n(x) = \boldsymbol{o}_n. \tag{13}$$

Since $\hat{F}'(x) = \left[-I_n\ \middle|\ \boldsymbol{o}_n\right]$ and $\hat{p}_0'(x) = 0$, then $[\hat{p}_1'(x), \hat{p}_2'(x), \dots, \hat{p}_{n-1}'(x), \hat{p}_n'(x)]^T$ is the solution of the lower triangular linear system

$$\begin{bmatrix} \hat{a}_0 & 0 & 0 & 0 & 0 & \dots & 0 \\ \hat{b}_1 - x & \hat{a}_1 & 0 & 0 & 0 & \dots & 0 \\ \hat{c}_2 & \hat{b}_2 - x & \hat{a}_2 & 0 & 0 & \dots & 0 \\ \hat{d}_3 & \hat{c}_3 & \hat{b}_3 - x & \hat{a}_3 & 0 & \dots & 0 \\ \vdots & \ddots & \ddots & \ddots & \ddots & \vdots & \vdots \\ 0 & \dots & \hat{d}_{n-2} & \hat{c}_{n-2} & \hat{b}_{n-2} - x & \hat{a}_{n-2} & 0 \\ 0 & \dots & 0 & \hat{d}_{n-1} & \hat{c}_{n-1} & \hat{b}_{n-1} - x & \hat{a}_{n-1} \end{bmatrix} \begin{bmatrix} \hat{p}_1'(x) \\ \hat{p}_2'(x) \\ \hat{p}_3'(x) \\ \hat{p}_4'(x) \\ \vdots \\ \hat{p}_{n-1}'(x) \\ \hat{p}_n'(x) \end{bmatrix} = \begin{bmatrix} \hat{p}_0(x) \\ \hat{p}_1(x) \\ \hat{p}_2(x) \\ \hat{p}_3(x) \\ \vdots \\ \hat{p}_{n-2}(x) \\ \hat{p}_{n-1}(x) \end{bmatrix}.$$

The `Matlab` function `one_step_Newton.m` computing $\hat{p}_n(x)$ and $\hat{p}_n'(x)$ with $35n$ flops, making use of the `Matlab` function `givens.m`, is displayed in Table 12 of the Appendix.

The sequence of approximations generated by the Ehrlich–Aberth method converges cubically to the eigenvalues of $\hat{H}_n$, or even faster if the implementation is in the Gauss–Seidel style, since the eigenvalues are simple. In practice, as noticed in [25], the Ehrlich–Aberth iteration exhibits good global convergence properties, though no theoretical results seem to be known about global convergence.

The main requirements for the success of the Ehrlich–Aberth method are a fast, robust, and stable computation of the Newton correction $\hat{p}_n(x)/\hat{p}'_n(x)$, and a good set of initial approximations for the zeros, $\hat{\boldsymbol{x}}^{(0)}$, so that the number of iterations needed for convergence is not too large.

Since the eigenvalues of $\hat{H}_n$ are real, different approaches can be taken into account to compute the initial guess of the vector $\hat{\boldsymbol{x}}^{(0)}$. Here, we consider two algorithms requiring $\mathcal{O}(n^2)$ floating point operations and $\mathcal{O}(n)$ memory:[2]

1. $\hat{\boldsymbol{x}}^{(0)}$=`gausq2`(`diag`($\hat{H}_n$), `diag`($\hat{H}_n, -1$)), *i.e.*, $\hat{\boldsymbol{x}}^{(0)}$ is the vector of the eigenvalues of the symmetric tridiagonal matrix obtained setting $\hat{d}_i = 0,\ i = 2, \dots, n-1$, in $\hat{H}_n$. Computational complexity: $\mathcal{O}(n^2)$, memory: $\mathcal{O}(n)$;
2. a. reduction of $\hat{H}_n$ to a similar nonsymmetric tridiagonal matrix $\hat{T}_n$, by using elementary transformations. Computational complexity: $\frac{7}{2}n^2$ flops;
   b. reduction of $\hat{T}_n$ to a similar symmetric tridiagonal matrix $T_n = \hat{D}_n^{-1}\hat{T}_n\hat{D}_n$, with $\hat{D}_n$ a diagonal matrix. Computational complexity: $5n$ flops;
   c. $\hat{\boldsymbol{x}}^{(0)}$=`gausq2`(`diag`($T_n$), `diag`($T_n, -1$)). Computational complexity: $\mathcal{O}(n^2)$, memory: $\mathcal{O}(n)$.

While the first approach yields an approximation of the eigenvalues of $\hat{H}_n$, and then of H_n, the second one provides the eigenvalues of $\hat{H}_n$ as $\hat{\boldsymbol{x}}^{(0)}$, if computed in exact arithmetic.

The analysis of the implementation of the second approach in floating point arithmetic has been described in [13]. In practice, this approach works in a stable way for all the MOPs listed in Section 3 and the Ehrlich–Aberth method converges in one iteration. Therefore, approach (2) is adopted in order to compute the initial vector $\hat{\boldsymbol{x}}^{(0)}$. We observed that, even though `diag`($\hat{H}_n, -2$) has negative entries, the entries of the subdiagonal and superdiagonal of $\hat{T}_n$ are always positive. Therefore, the similar symmetric matrix T_n can be computed without requiring complex arithmetic.

The `Matlab` function `tridEHbackwardV.m`, implementing the reduction of $\hat{H}_n$ into a similar nonsymmetric tridiagonal matrix $\hat{T}_n$ (step (2) (a)), is displayed in Table 10 of the Appendix.

The `Matlab` function `DScaleSV2.m`, implementing the reduction of $\hat{T}_n$ into a similar symmetric tridiagonal matrix T_n (step (2) (a)), is a simplification of the function `DScaleSV2.m` and it is displayed in the Appendix, Table 9.

At iteration $\ell + 1$, we consider

$$\hat{x}_j^{(\ell+1)} \quad \text{and} \quad \hat{\boldsymbol{v}}^{(j)} = \left[\hat{p}_0(x), \hat{p}_1(x), \hat{p}_2(x), \cdots, \hat{p}_{n-2}(x), \hat{p}_{n-1}(x)\right]^T,$$

respectively, as an eigenvalue and corresponding right eigenvector of $\hat{H}_n$ if

$$|\hat{p}_n(\hat{x}_j^{(\ell+1)})| \le tol_1.$$

The corresponding left eigenvector is computed applying a sequence of $n-1$ Givens rotations

$$G_i = \begin{bmatrix} I_{i-1} & & & \\ & \gamma_i & \sigma_i & \\ & -\sigma_i & \gamma_i & \\ & & & I_{n-i-1} \end{bmatrix}$$

to the left of $\hat{H}_n - \hat{x}_j^{(\ell+1)} I_n$, a numerically singular matrix, such that

$$G_1^T G_2^T \cdots G_{n-1}^T \left(\tilde{H}_n - \hat{x}_j^{(\ell+1)} I_n\right)$$

has the upper diagonal and the $(1,1)$ entry annihilated. Therefore,

$$\begin{bmatrix} \gamma_1 \\ -\sigma_1\gamma_2 \\ \sigma_1\sigma_2\gamma_3 \\ \vdots \\ (-1)^{n-2}\prod_{k=1}^{n-2}\sigma_k\gamma_{n-1} \\ (-1)^{n-1}\prod_{k=1}^{n-1}\sigma_k \end{bmatrix},$$

i.e., the first column of $G_{n-1}G_{n-2}\cdots G_2G_1$, is the normalized left eigenvector $\hat{\boldsymbol{v}}^{(j)}$. The `Matlab` function `left_eigvect.m`, computing the normalized left eigenvector of $\hat{H}_n$, is displayed in the Appendix, Table 13.

[2] We have written the `Matlab` function `gausq2.m`, a modified version of the `fortran` routine `imtql2.f` from `eispack` [27,28], to compute the eigenvalues of a symmetric tridiagonal matrix T_n, given only its main diagonal `diag`(T_n) and the subdiagonal `diag`($T_n, -1$): [x]=`gausq2`(`diag`(T_n), `diag`($T_n, -1$)).

The Matlab function EA_method.m, implementing the Ehrlich–Aberth method, is described in the Appendix, Table 11.

The Matlab implementation of the simultaneous Gaussian quadrature rules, called GaussMOP.m, is described in the Appendix, Table 7.

Remark 2. In case $r \geq 2$, the system of monic MOPs satisfies a $(r+2)$-term recurrence relation

$$xp_n(x) = p_{n+1}(x) + \sum_{j=0}^{r} a_{n,j} p_{n-j}(x), \quad n \geq 0. \tag{14}$$

Writing (14) in matrix form, we obtain

$$H_n \begin{bmatrix} p_0(x) \\ p_1(x) \\ \vdots \\ p_{n-1}(x) \end{bmatrix} + \begin{bmatrix} 0 \\ \vdots \\ 0 \\ p_n(x) \end{bmatrix} = x \begin{bmatrix} p_0(x) \\ p_1(x) \\ \vdots \\ p_{n-1}(x) \end{bmatrix},$$

with

$$H_n = \begin{bmatrix} a_{0,0} & 1 & & & & & & \\ a_{1,1} & a_{1,0} & 1 & & & & & \\ a_{2,2} & a_{2,1} & a_{2,0} & 1 & & & & \\ \vdots & \ddots & \ddots & \ddots & \ddots & & & \\ a_{r,r} & a_{r,r-1} & \cdots & a_{r,1} & a_{r,0} & 1 & & \\ & a_{r+1,r} & a_{r+1,r-1} & \cdots & a_{r+1,1} & a_{r+1,0} & \ddots & \\ & & \ddots & \ddots & \ddots & \ddots & \ddots & 1 \\ & & & a_{n-1,r} & a_{n-1,r-1} & \cdots & a_{n-1,1} & a_{n-1,0} \end{bmatrix}.$$

Hence, similarly to the case $r = 2$, the simultaneous Gaussian quadrature rule for $r > 2$ can be retrieved from the eigenvalues and corresponding left and right eigenvectors of H_n [8].

A sketch of an algorithm for computing the simultaneous Gaussian quadrature rule for $r > 2$ can be summarized in the following steps:

1. construct the Hessenberg matrix H_n;
2. compute $\hat{H}_n = S_n^{-1} H_n S_n$, with S_n a diagonal matrix such that $\mathtt{triu}(\hat{H}_n, -1)$ is symmetric;
3. compute the left and right eigendecompsition of $\hat{H}_n$: $\hat{U}_n^T \hat{H}_n = \hat{\Lambda}_n \hat{U}_n^T$ and $\hat{H}_n \hat{V}_n = \hat{V}_n \hat{\Lambda}_n$;
4. retrieve the nodes and the weights of the simultaneous Gaussian quadrature rule from $\hat{\Lambda}_n, \hat{U}_n, \hat{V}_n$ and S_n.

8. Numerical tests

In this section we report some numerical tests performed in Matlab R2022a, with machine precision $\varepsilon \approx 2.22 \times 10^{-16}$. For all considered MOPs, the results of GaussMOP.m with n nodes, for $n = 10, 20, \ldots, 90, 100$, are compared to those obtained by using the Matlab function integral.m applied in two ways: without setting the absolute error, denoted by I_1, and imposing the absolute error equal to 10^{-13}, denoted by I_2. The same integrals were also computed by the Mathematica 13.0 function Integrate, setting the precision to 80 digits. These values, rounded to floating point numbers by means of the matlab function double.m, are assumed to be the exact ones and compared with the results of GaussMOP.m, I_1, and I_2.

Example 1. In this example, GaussMOP.m is used to simultaneously compute the integrals

$$\int_{\Delta^{(i)}} f(x) w^{(1)}(x) dx, \quad \int_{\Delta^{(i)}} f(x) w^{(2)}(x) dx,$$

where the integrand function is $f(x) = xe^{-x}$, the same function considered in [5], and the input parameters for all MOPs are those displayed in Table 2.

The absolute errors of the integrals computed by GaussMOP.m are displayed in Table 4 and Table 5 for $w^{(1)}$ and $w^{(2)}$, respectively. Moreover, in the last two rows of these tables, the results obtained applying the Matlab function integral.m (I_1 and I_2), are reported.

Observe that, in some cases, the functions I_1 and I_2 yield NaN as results, displaying the message "Warning: Inf or NaN value encountered".

In Table 6 the execution times (in seconds) required by GaussMOP.m, for different values of n, and by the Matlab functions I_1 and I_2, applied for computing both integrals with weights $w^{(1)}$ and $w^{(2)}$, are reported. Although the code GaussMOP.m is interpreted by Matlab, GaussMOP.m, I_1 and I_2, exhibit comparable execution times, except for $\mathbf{MOP}_8$ and $\mathbf{MOP}_9$, for which GaussMOP.m is significantly faster than I_1 and I_2.

Table 4
Absolute errors of the integrals, with weight $w^{(1)}$, computed by `GaussMOP.m`, for $n = 10, 20, \ldots, 100$, and by I_1 and I_2.

	$\int_{\Delta^{(i)}} f(x)w^{(1)}(x)dx, \quad f(x) = xe^{-x}$								
n	**MOP$_1$**	**MOP$_2$**	**MOP$_3$**	**MOP$_4$**	**MOP$_5$**	**MOP$_6$**	**MOP$_7$**	**MOP$_8$**	**MOP$_9$**
10	0	3.23(−9)	7.17(−4)	5.23(−13)	1.23(−10)	3.88(−4)	3.75(−5)	4.16(−17)	5.79(−10)
20	1.99(−15)	2.10(−15)	4.59(−8)	1.34(−13)	2.22(−15)	6.86(−6)	1.47(−10)	1.94(−16)	4.99(−16)
30	1.77(−15)	2.05(−15)	2.18(−12)	5.45(−14)	5.61(−14)	7.47(−7)	3.33(−15)	1.80(−16)	5.55(−17)
40	5.10(−15)	1.24(−14)	2.54(−14)	2.92(−13)	3.78(−13)	5.97(−8)	1.99(−15)	1.80(−16)	3.88(−16)
50	4.32(−15)	2.58(−14)	6.98(−13)	3.35(−14)	2.49(−13)	6.07(−10)	4.21(−15)	6.93(−17)	0
60	7.32(−15)	1.66(−16)	4.18(−14)	5.99(−15)	4.19(−13)	5.86(−10)	7.32(−15)	4.16(−17)	9.21(−15)
70	7.21(−15)	1.08(−14)	1.01(−14)	6.70(−13)	9.43(−13)	3.50(−11)	7.32(−15)	3.46(−16)	3.49(−15)
80	1.11(−16)	9.38(−15)	8.10(−13)	2.25(−13)	9.39(−13)	1.12(−11)	4.88(−15)	5.13(−16)	5.55(−17)
90	1.22(−15)	3.36(−14)	2.44(−14)	4.53(−13)	9.39(−13)	8.46(−13)	2.37(−14)	1.38(−17)	3.74(−14)
100	2.33(−15)	3.68(−14)	1.27(−13)	4.10(−13)	1.27(−13)	1.18(−12)	2.66(−15)	0	2.83(−14)
I_2	2.17(−13)	1.11(−16)	0	NaN	NaN	0	NaN	5.96(−13)	5.55(−17)
I_1	8.74(−10)	0	1.11(−16)	NaN	NaN	2.77(−17)	NaN	1.38(−17)	0

Table 5
Absolute errors of the integrals, with weight $w^{(2)}$, computed by `GaussMOP.m`, for $n = 10, 20, \ldots, 100$, and by I_1 and I_2.

	$\int_{\Delta^{(i)}} f(x)w^{(2)}(x)dx, \quad f(x) = xe^{-x}$								
n	**MOP$_1$**	**MOP$_2$**	**MOP$_3$**	**MOP$_4$**	**MOP$_5$**	**MOP$_6$**	**MOP$_7$**	**MOP$_8$**	**MOP$_9$**
10	1.11(−16)	2.35(−8)	2.33(−3)	5.32(−15)	3.103(−11)	1.97(−3)	1.21(−3)	1.11(−16)	2.64(−10)
20	1.99(−15)	1.05(−15)	7.19(−7)	1.37(−13)	1.93(−14)	4.61(−5)	3.90(−9)	2.22(−16)	4.44(−16)
30	2.10(−15)	1.11(−15)	1.64(−10)	8.31(−14)	3.41(−14)	6.85(−7)	2.22(−16)	3.60(−16)	0
40	4.99(−15)	5.19(−15)	5.72(−14)	7.36(−13)	2.76(−14)	1.36(−7)	4.21(−15)	3.05(−16)	2.22(−16)
50	4.32(−15)	1.06(−14)	6.81(−13)	6.02(−13)	9.65(−15)	1.92(−8)	1.37(−14)	1.94(−16)	1.11(−16)
60	7.54(−15)	4.24(−15)	4.28(−14)	3.63(−13)	2.35(−15)	1.58(−10)	1.86(−14)	8.32(−17)	9.38(−15)
70	7.66(−15)	2.35(−15)	6.99(−15)	1.54(−13)	4.79(−14)	3.29(−10)	8.21(−15)	2.22(−16)	3.33(−15)
80	0	1.12(−14)	7.90(−13)	3.07(−13)	1.92(−14)	2.06(−12)	4.66(−15)	4.44(−16)	7.21(−16)
90	1.44(−15)	6.77(−15)	2.22(−14)	1.65(−13)	2.74(−14)	9.16(−12)	3.01(−14)	2.77(−17)	3.98(−14)
100	2.55(−15)	1.29(−14)	1.23(−13)	4.26(−13)	1.82(−13)	1.35(−12)	1.99(−15)	1.66(−16)	2.80(−14)
I_2	5.58(−14)	0	0	NaN	5.55(−17)	5.55(−17)	NaN	2.77(−17)	5.55(−17)
I_1	1.06(−13)	5.55(−17)	0	NaN	0	0	NaN	2.77(−17)	0

Table 6
Execution times in seconds required for computing both integrals, with weights $w^{(1)}$ and $w^{(2)}$, by `GaussMOP.m`, for $n = 10, 20, \ldots, 100$, and by I_1 and I_2.

	Execution time in seconds								
n	**MOP$_1$**	**MOP$_2$**	**MOP$_3$**	**MOP$_4$**	**MOP$_5$**	**MOP$_6$**	**MOP$_7$**	**MOP$_8$**	**MOP$_9$**
10	5.51(−2)	1.00(−2)	6.21(−3)	4.47(−3)	1.07(−2)	4.11(−3)	2.95(−3)	5.77(−3)	3.37(−2)
20	6.27(−3)	2.00(−3)	2.56(−3)	2.01(−3)	3.49(−3)	1.55(−3)	1.44(−3)	2.38(−3)	2.85(−3)
30	1.94(−3)	1.93(−3)	1.83(−3)	1.89(−3)	2.33(−3)	1.85(−3)	1.94(−3)	2.00(−3)	2.15(−3)
40	3.00(−3)	2.97(−3)	2.93(−3)	3.03(−3)	2.89(−3)	2.91(−3)	2.83(−3)	2.99(−3)	3.00(−3)
50	6.38(−3)	5.95(−3)	6.01(−3)	6.14(−3)	7.02(−3)	6.98(−3)	6.32(−3)	6.65(−3)	7.29(−3)
60	5.78(−3)	5.45(−3)	5.23(−3)	5.53(−3)	5.16(−3)	4.89(−3)	4.96(−3)	5.06(−3)	5.69(−3)
70	6.71(−3)	6.60(−3)	6.61(−3)	6.72(−3)	6.58(−3)	6.52(−3)	6.52(−3)	6.63(−3)	7.44(−3)
80	8.68(−3)	8.31(−3)	8.40(−3)	8.34(−3)	8.34(−3)	8.31(−3)	8.54(−3)	8.55(−3)	9.23(−3)
90	1.08(−2)	1.07(−2)	1.08(−2)	1.06(−2)	1.06(−2)	1.04(−2)	1.05(−2)	1.07(−2)	1.06(−2)
100	1.31(−2)	1.29(−2)	1.28(−2)	1.27(−2)	1.28(−2)	1.26(−2)	1.27(−2)	1.29(−2)	1.29(−2)
I_1	2.68(−2)	1.10(−2)	5.64(−3)	5.12(−3)	5.65(−3)	3.28(−3)	3.96(−3)	2.49(−1)	2.61(0)
I_2	3.11(−3)	2.31(−3)	9.91(−4)	2.26(−3)	1.79(−3)	7.18(−4)	1.60(−3)	9.82(−2)	1.39(0)

9. Conclusions

A `Matlab` package called `GaussMOP.m` is proposed for computing simultaneous Gaussian quadrature rules associated with different kinds of MOPs. The nodes and weights of such rules are retrieved from the eigendecomposition of a banded lower Hessenberg matrix, which turns out to be an ill-conditioned eigenvalue problem. Making use of a novel balancing procedure, the eigenvalue condition of the latter Hessenberg matrix is drastically reduced. Moreover, a variant of the Aberth–Ehrlich method is used to compute the eigenvalues and associated left and right eigenvectors with $\mathcal{O}(n)$ memory and $\mathcal{O}(n^2)$ computational complexity.

`GaussMOP.m` was applied for simultaneously computing integrals with two different weights associated with the considered MOPs, and its performance was compared, in terms of accuracy, robustness and execution time, to that of the `Matlab` intrinsic function `integral.m`. The values obtained by computing the latter integrals by the `Mathematica` function `Integrate`, requiring a precision of 80 digits, were considered as the exact ones.

The numerical tests show the reliability of the proposed numerical method.

Data availability

No data was used for the research described in the article.

Acknowledgments

Teresa Laudadio and Nicola Mastronardi are members of the Gruppo Nazionale Calcolo Scientifico-Istituto Nazionale di Alta Matematica (GNCS-INdAM). The work of Nicola Mastronardi was partly supported by MIUR, PROGETTO DI RICERCA DI RILEVANTE INTERESSE NAZIONALE (PRIN) 20227PCCKZ "Low–rank Structures and Numerical Methods in Matrix and Tensor Computations and their Application", Università degli Studi di BOLOGNA CUP J53D23003620006. The work of Walter Van Assche was supported by FWO, Belgium grant G0C9819N. The work of Paul Van Dooren was partly supported by Consiglio Nazionale delle Ricerche, Italy, under the Short Term Mobility program.

Appendix A. Supplementary data

Supplementary material related to this article can be found online at https://doi.org/10.1016/j.cam.2024.116109.
The `Matlab` functions implementing the simultaneus Gaussian quadrature rules for MOPs can be found in the Appendix at https://doi.org/10.1016/j.cam.2024.116109. The whole package can be downloaded from
https://users.ba.cnr.it/iac/irmanm21/MOP/
or from the webpage
https://users.ba.cnr.it/iac/irmanm21/MOP/MOP_W.html.

References

[1] A.B.J. Kuijlaars, Multiple orthogonal polynomial ensembles, in: J. Arvesú, F. Marcellán, A. Martínez-Finkelshtein (Eds.), Recent Trends in Orthogonal Polynomials and Approximation Theory, in: Contemp. Math., vol. 507, Springer, Berlin, 2010, pp. 155–176.
[2] A. Sokal, Multiple orthogonal polynomials, d-orthogonal polynomials, production matrices, and branched continued fractions, Trans. Amer. Math. Soc., Ser. B 11 (2024) 762–797.
[3] A. Branquinho, J.E.F. Díaz, A. Foulquié-Moreno, M. Mañas, Bidiagonal factorization of the recurrence matrix for the hahn multiple orthogonal polynomials, 2023, arXiv:2308.01288.
[4] C.F. Borges, On a class of Gauss–like quadrature rules, Numer. Math. 67 (1994) 271–288.
[5] W. Van Assche, A Golub–Welsch version for simultaneous Gaussian quadrature, Numer. Algorithms (2024) http://dx.doi.org/10.1007/s11075-024-01767-2.
[6] W. Gautschi, G.V. Milovanović, Orthogonal polynomials relative to weight functions of Prudnikov type, Numer. Algorithms 90 (1) (2022) 263–270.
[7] W. Gautschi, Another look at polynomials orthogonal relative to exponential integral weight functions, Numer. Algorithms 91 (4) (2022) 1547–1557.
[8] J. Coussement, W. Van Assche, Gaussian quadrature for multiple orthogonal polynomials, J. Comput. Appl. Math. 178 (2005) 131–145.
[9] W. Van Assche, E. Coussement, Some classical multiple orthogonal polynomials, J. Comput. Appl. Math. 127 (2001) 317–347.
[10] M.E.H. Ismail, Classical and Quantum Orthogonal Polynomials in One Variable, Fifth, in: Encyclopedia of Mathematics and its Applications, vol. 98, Cambridge University Press, London, 2005.
[11] G.H. Golub, J.H. Welsch, Calculation of Gauss quadrature rules, Math. Comp. 23 (1969) 221–230.
[12] D. Lubinsky, W. Van Assche, Simultaneous Gaussian quadrature for Angelesco systems, Jaen J. Approx. 8(2) (2016) 113–149 8 (2) (2016) 113–149.
[13] T. Laudadio, N. Mastronardi, P. Van Dooren, Computational aspects of simultaneous Gaussian quadrature, Numer. Algorithms (2024) http://dx.doi.org/10.1007/s11075-024-01785-0.
[14] M. Abramowitz, I.A. Stegun, Handbook of Mathematical Functions, fifth ed., Dover, New York, 1964.
[15] L.R. Piñeiro, On simultaneous approximations for a collection of Markov functions, Moscow Univ. Math. Bull. 42 (2) (1987) 52–55.
[16] V.N. Sorokin, Generalization of classical orthogonal polynomials and convergence of simultaneous Padé approximants, Trudy Sem. Im. I. G. Petrovsk. 11 (1986) 125–165, translated in J. Soviet Math. 45 (1989) 1461–1499.
[17] V.N. Sorokin, Simultaneous Padé approximation for functions of Stieltjes type, Siber. Mat. Zh. 31 (6) (1990) 128–137, translated in Siber. Math. J. 31(5) (1990) 809–817.
[18] E.M. Nikishin, V.N. Sorokin, Rational approximations and orthogonality, in: Translations of Mathematical Monographs, vol. 92, Amer. Math. Soc., Providence, RI, 1991.
[19] Y. Ben Cheikh, K. Douak, On two-orthogonal polynomials related to the Bateman's $J_n^{u,v}$–function, Methods Appl. Anal. 7 (4) (2000) 641–662.
[20] W. Van Assche, S. Yakubovich, Multiple orthogonal polynomials associated with macdonald functions, Integral Transforms Spec. Funct. 9 (3) (2000) 229–244.
[21] E. Coussement, W. Van Assche, Multiple orthogonal polynomials associated with the modified bessel functions of the first kind, Constr. Approx. 19 (2003) 237–263, http://dx.doi.org/10.1007/s00365-002-0499-9.
[22] H. Lima, A. Loureiro, Multiple orthogonal polynomials with respect to Gauss' hypergeometric function, Stud. Appl. Math. 148 (1) (2022) 154–185.
[23] H. Lima, A. Loureiro, Multiple orthogonal polynomials associated with confluent hypergeometric functions, J. Approx. Theory 260 (2020) 105484.
[24] O. Aberth, Iteration methods for finding all zeros of a polynomial simultaneously, Math. Comp. 27 (1973) 339–344.
[25] D.A. Bini, L. Gemignani, F. Tisseur, The Ehrlich–Aberth method for the nonsymmetric tridiagonal eigenvalue problem, SIAM J. Matrix Anal. Appl. 27 (2005) 153–175.
[26] L.W. Ehrlich, A modified Newton method for polynomials, Commun. ACM 10 (1967) 107–108.
[27] H. Bowdler, R. Martin, C. Reinsch, J. Wilkinson, The QR and QL algorithms for symmetric matrices: TQL1 and TQL2, Numer. Math. 11 (4) (1968) 293–306.
[28] B.S. Garbow, J.M. Boyle, J.J. Dongarra, C.B. Moler, Matrix eigensystem routines - EISPACK guide extension, in: Lecture Notes in Computer Science, vol. 51, Springer, 1977.