© 2026 The Author(s). Published by Università degli studi di Torino. Open access under the CC BY license.

J. Approx. Softw., Vol. 3, No. 2 (2026) 1

Journal of Approximation Software

doi: 10.13135/3103-1935/12008 – ISSN: 3103–1935

A MatLab package for solving Fredholm integral equations arising from second-order boundary value problems on the real axis

M. C. De Bonis [1] and V. Sagaria [2,⋆]

[1] *Department of Basic and Applied Sciences, University of Basilicata, Italy*
[2] *Department of Engineering, University of Basilicata, Italy*

Received: 03/06/2025 – Published: 20/03/2026
Communicated by: F. Dell'Accio

Abstract

The goal of this paper is to introduce a MatLab package for the numerical solution of second-order Boundary Value Problems (BVPs) defined on the real line. This package implements the Nyström-type method proposed in [6] for solving equivalent Fredholm Integral Equations (FIEs) of the second kind, based on a product quadrature rule having the Hermite zeros as quadrature knots. The package is easy-to-use thanks to the MatLab Graphical User Interface (GUI) BVP_Real_Axis.

Keywords: Boundary value problem, Fredholm integral equation, Nyström method, Truncated Lagrange interpolation, Hermite polynomials (MSC2020: 65R20, 65L10)

1 Introduction

In this paper we propose a MatLab package designed to numerically solve second-order Boundary Value Problems (BVPs) defined on the real axis of the following type

$$\begin{cases} f''(x) - \mu a(x) f(x) = h(x), \\ f(\pm\infty) = 0, \end{cases} \tag{1}$$

where μ is a real parameter, and a,h are given functions satisfying suitable assumptions. Note that, if the boundary condition in (1) is replaced by $f(\pm\infty) = b, b \neq 0$, the corresponding BVP can be reduced to (1) simply taking $h(x) + \mu b a(x)$ in place of $h(x)$ and its solution is $f(x) + b$. Then, considering the boundary condition $f(\pm\infty) = 0$ does not produce a loss of generality.

⋆ Corresponding author: valeria.sagaria@unibas.it

Differential equations with such boundary conditions are of interest because are often used to model physical phenomena (see, for example, [1, 2, 3, 4, 7, 8, 9, 12, 13, 14, 16, 20] and the references there in). The numerical procedure implemented by the package has been introduced in [6] and, following an idea in [5], consists in approximating the solutions of the following Fredholm Integral Equations (FIEs)

$$f(t) - \int_{\mathbb{R}} G(x,t)\tilde{a}(x)f(x)\,dx = \int_{\mathbb{R}} G(x,t)h(x)\,dx, \tag{2}$$

where the kernel $G(x,t)$, referred to as Green's function, is expressed in the following form

$$G(x,t) = -\frac{1}{2}\begin{cases} e^{-t}e^{x}, & -\infty < x \le t, \\ e^{t}e^{-x}, & t \le x < \infty, \end{cases} \tag{3}$$

and $\tilde{a}(x) = \mu a(x) - 1$, that are equivalent to (1). A Nyström type method is employed and the approximate solution is constructed as a Nyström interpolant at the Hermite zeros. In [6], sufficient conditions for the stability and convergence of the method are proved.

The details of the implemented Nyström method, along with the theoretical results on its stability and convergence, are given in next section. The MatLab functions provided in the package are described in Section 3. In Section 4, five demos show the performance of the package. Finally, Section 5 is dedicated to the description of the MatLab Graphical User Interface (GUI) BVP_Real_Axis.

The package is licensed under the GNU General Public License v3.0 and is freely available through https://github.com/ValeriaSagaria/BVP_Real_Axis.

2 The implemented Nyström method

Starting from FIE (2) which is equivalent to BVP (1), as a first step we multiply and divide, under the integral sign, by the square root of the classical Hermite weight $w(x) = e^{-x^2}$. Then, setting $A(x) := \tilde{a}(x)e^{\frac{x^2}{2}}$ and denoting by $H(x)$ the right-side of (2), we write (2) as follows

$$f(t) - \int_{\mathbb{R}} G(x,t)A(x)f(x)\sqrt{w(x)}\,dx = H(t). \tag{4}$$

Subsequently, we approximate the integral operator

$$(Kf)(t) := \int_{\mathbb{R}} G(x,t)A(x)f(x)\sqrt{w(x)}\,dx$$

by the discrete operator

$$(K_m f)(t) := \int_{\mathbb{R}} G(x,t)L_m(w,Af,x)\sqrt{w(x)}\,dx, \tag{5}$$

where

$$L_m(w,Af,x) := \sum_{k=-j}^{j} A(x_k)f(x_k)l_{m,k}(x), \tag{6}$$

is the truncated version of the Lagrange polynomial interpolating the function Af at the zeros $\{x_k\}_{k=-j}^{j}$ of the orthonormal Hermite polynomial $p_m(w;x) =: p_m(x)$ (see, for example, [15, 19]). In (6), $l_{m,k}$ denotes the $k-$th fundamental Lagrange polynomial and

$$j := \min_{1\le k\le \lfloor m/2 \rfloor} \{k : x_k \ge \theta\sqrt{2m}\}, \tag{7}$$

with $0 < \theta < 1$ fixed, is the truncation parameter. Substituting (6) into (5), (5) becomes

$$(K_m f)(t) = \sum_{k=-j}^{j} A(x_k) M_k(t) f_m(x_k),$$

where

$$M_k(t) = \int_{\mathbb{R}} G(x,t) l_{m,k}(x) \sqrt{w(x)}\, dx, \quad k = -j, \ldots, j,$$

are the so-called modified moments, and replacing it into (4) we obtain the following finite-dimensional equation

$$f_m(t) - \sum_{k=-j}^{j} A(x_k) M_k(t) f_m(x_k) = H(t) \tag{8}$$

in the unknown f_m.

Finally, collocating equation (8) at the quadrature nodes leads to the following linear system of $2j$ equations

$$\sum_{k=-j}^{j} [\delta_{i,k} - A(x_k) M_k(x_i)] z_k = H(x_i), \quad i = -j, \ldots, j, \tag{9}$$

in the unknowns $\{z_k := f_m(z_k)\}_{k=-j}^{j}$. Once the above linear system is solved, its solution $\{\bar{z}_k\}_{k=-j}^{j}$ let us to construct the following so-called Nyström interpolating function

$$f_m(t) := \sum_{k=-j}^{j} A(x_k) M_k(t) \bar{z}_k + H(t), \tag{10}$$

solution of equation (8).

In [6], the analysis of the solvability of equation (2) (and then of equation (1)) and of the stability and convergence of the just described method, has been investigated within the following subspaces of the bounded continuous functions on $\mathbb{R}$

$$\mathscr{B} = \left\{ f \in C^0(\mathbb{R}) : \|f\|_\infty < +\infty, \lim_{x \to \pm\infty} f(x) = 0 \right\}$$

and

$$\mathscr{W}_r = \left\{ f \in \mathscr{B} : \|f^{(k)}\|_\infty < +\infty, 0 \le k \le r \right\},$$

endowed with the norm

$$\|f\|_{\mathscr{W}_r} := \max_{0 \le k \le r} \|f^{(k)}\|_\infty.$$

The main results in [6] are collected in the following theorem.

Theorem 2.1. *Let* $v_\delta(x) = (1+|x|)^\delta$, $\frac{2}{3} < \delta < 2$. *Let us suppose that* $\ker(I-K) = \{0\}$ *in* $\mathscr{B}$ *and that, for* $r \ge 1$, $h \in \mathscr{W}_{r-1}$ *and*

$$\sup_{x \in \mathbb{R}} |\tilde{a}^{(i)}(x)| < +\infty, \qquad i = 0, \ldots, r-1,$$

$$\sup_{x \in \mathbb{R}} |\tilde{a}^{(i-k)}(x)| x|^k v_\delta(x)| < \infty, \qquad \begin{matrix} i = 0, \ldots, r, \\ k = 0, \ldots, i, \end{matrix}$$

then

- *equation* (2) *(and then* (1)*) admits a unique solution* $f \in \mathscr{W}_r$*;*
- *for all sufficiently large* m *(say* $m \geq m_0$*), the method is stable, i.e., the approximate inverses* $(I-K_m)^{-1} : \mathscr{B} \to \mathscr{B}$ *exist and are uniformly bounded;*
- *the condition numbers in uniform norm of the coefficient matrices* $\mathscr{A}_m$ *of linear system* (9) *are bounded, i.e.,*

$$\sup_{m \geq m_0} \operatorname{cond}(\mathscr{A}_m) \leq \mathscr{C}, \quad \mathscr{C} \neq \mathscr{C}(m);$$

- *the Nyström interpolating function* (10) *converges to the exact solution* f *of integral equation* (2) *(and then of BVP* (1)*) and the following error estimate holds true*

$$\|f - f_m\|_\infty = \mathscr{O}\left(m^{-\frac{r}{2}}\right).$$

2.1 Computational aspects

In this subsection we explain some computational details useful to implement the method described above.

Concerning the truncation parameter in (7), numerical evidence shows that to have approximations of the solutions with the greatest number of exact decimal digits and with the lowest computational cost it is more convenient to analyze the effective decay of the coefficient matrix. Then, we have used the following dynamic criterion

$$j := \min_{1 \leq \rho \leq \lfloor \frac{m}{2} \rfloor} \{\rho \ : \ |A(x_\rho) M_\rho(x_1)| < eps\}, \tag{11}$$

where $eps = 2.22044e-16$ is the epsilon machine in double precision arithmetic.

Another key computational aspect is related to the computation of integral

$$H(t) = \int_{\mathbb{R}} G(x,t) h(x)\, dx. \tag{12}$$

For some choices of $h(t)$, a closed-form expression of $H(t)$ can be determined using Wolfram Mathematica Software or any other software capable of symbolic computation.

In all other cases, it is necessary to appropriately approximate $H(t)$. After splitting it using the Green's function (3) and doing a simple change of variable, we write

$$\begin{aligned} H(t) &= -\frac{1}{2}\left(e^{-t}\int_{-\infty}^{t} e^{x} h(x)\, dx + e^{t}\int_{t}^{+\infty} e^{-x} h(x)\, dx\right) \\ &= -\frac{1}{2}\left(\int_0^{+\infty} h(t-y) e^{-y}\, dy + \int_0^{+\infty} h(t+y) e^{-y}\, dy\right), \end{aligned}$$

and the two involved integrals are approximated by means of a truncated Gauss-Laguerre quadrature rule [18]. Then, $H(t)$ is approximated by

$$H_m(t) = -\frac{1}{2}\sum_{k=1}^{J} \lambda_{m,k}(v)\left[h(t-y_k) + h(t+y_k)\right], \tag{13}$$

with $v(x) = e^{-x}$ the Laguerre weight function, $\{y_k\}_{k=1}^{J}$ the zeros of the $m-$th Laguerre polynomial, $\{\lambda_{m,k}(v)\}_{k=1}^{J}$ the Christoffel numbers and J the truncation parameter defined as follows

$$J := \min_{1 \leq k \leq m} \{k \ : \ |\lambda_{m,k}[h(t-y_k) + h(t+y_k)]| < eps\}.$$

We remark that, in the case where it is necessary to approximate the integral $H(t)$, the Nyström method described above is still stable and convergent and, in particular, as stated in [5, Corollary 3.2], the convergence is guaranteed with the same order.

Finally, we provide the implementation details related to the computation of the modified moments $\{M_k\}_{k=-j}^{j}$. We can observe that, once the fundamental Lagrange polynomial $l_{m,k}$ is expressed by means of the Darboux kernel [17], we get

$$\begin{aligned} M_k(t) &= \int_{\mathbb{R}} G(x,t) l_{m,k}(x) \sqrt{w(x)}\, dx \\ &= \lambda_{m,k} \sum_{i=0}^{m-1} p_i(x_k) \int_{\mathbb{R}} G(x,t) p_i(x) \sqrt{w(x)}\, dx \\ &=: \lambda_{m,k} \sum_{i=0}^{m-1} p_i(x_k) c_i(t), \quad k = -j, \ldots, j, \end{aligned} \tag{14}$$

and the computation of integrals $\{M_k\}_{k=-j}^{j}$ relies on the evaluation of integrals $\{c_i\}_{i=0}^{m-1}$. We employ the following strategy. Using again the expression of Green's function (3), we first write

$$\begin{aligned} c_i(t) &= -\frac{1}{2} e^{-t} \int_{-\infty}^{t} e^{x} p_i(x) \sqrt{w(x)}\, dx - \frac{1}{2} e^{t} \int_{t}^{+\infty} e^{-x} p_i(x) \sqrt{w(x)}\, dx \\ &:= c_i^1(t) + c_i^2(t). \end{aligned}$$

Now, using the well-known three-term recurrence formula of the Hermite orthonormal polynomial (see, for instance [21, (5.5.1) and (5.5.4)]) and [21, (5.5.10)], we get

$$c_i(t) =: \sqrt{\frac{2}{i}} \left[c_{i-1}^1(t) - c_{i-1}^2(t) \right] + \sqrt{\frac{i-1}{i}} c_{i-2}(t), \quad i = 1, 2, \ldots, m-1, \tag{15}$$

where

$$\begin{aligned} c_i^1(t) &= \sqrt{\frac{2}{i}} \frac{p_{i-1}(t) \sqrt{w(t)}}{2} + \sqrt{\frac{2}{i}} c_{i-1}^1(t) + \sqrt{\frac{i-1}{i}} c_{i-2}^1(t), \\ c_i^2(t) &= -\sqrt{\frac{2}{i}} \frac{p_{i-1}(t) \sqrt{w(t)}}{2} - \sqrt{\frac{2}{i}} c_{i-1}^2(t) + \sqrt{\frac{i-1}{i}} c_{i-2}^2(t), \\ & i = 1, 2, \ldots, m-1, \end{aligned}$$

with the following starting moments

$$\begin{aligned} c_0^1(t) &= -\frac{1}{2} \sqrt{\frac{e\sqrt{\pi}}{2}} \left[1 + Erf\left(\frac{t-1}{\sqrt{2}} \right) \right] (\cosh t - \sinh t), \\ c_0^2(t) &= -\frac{1}{2} \sqrt{\frac{e\sqrt{\pi}}{2}} Erfc\left(\frac{t+1}{\sqrt{2}} \right), \end{aligned}$$

having denoted by $Erf(z)$ and $Erfc(z)$ the error function and the complementary error function, respectively.

3 The MatLab Package

The BVP Package we are going to propose is composed by seven novel MatLab functions. Moreover, the function gaussq.m, translation of a Fortran procedure [10, 11], is also used. In

Figure 1: The MatLab functions of the BVP Package.

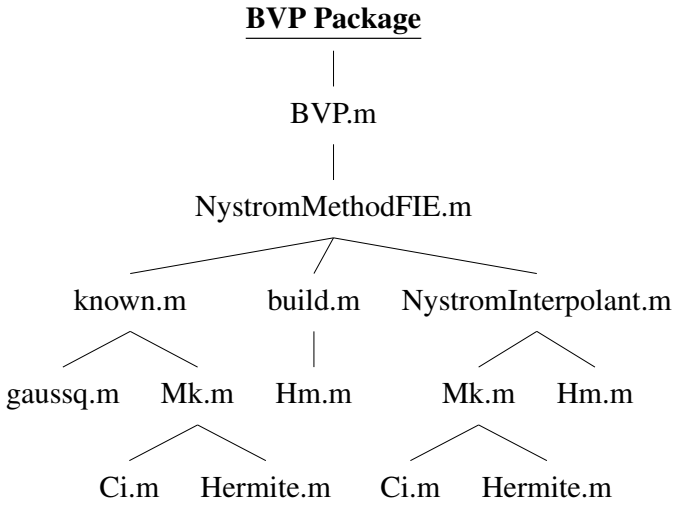


Figure 1 we have represented, starting from the top of the tree to the bottom, the core function `BVP.m` on the first level and the functions that are needed to run it on the underlying levels.

We start from the description of the main routine `BVP.m`:

Code 1: description of BVP.m MatLab function

```
function [fm,C,j,Merr,err] = BVP(h,AA,t,kind)
   %-------------------------------------------------------------------------
   % File: BVP.m
   %
   % Goal: Compute the solution of the BVP (Boundary Value Problem)
   %
   %           |f''(x)-mu a(x)f(x) = h(x)
   %           |f(-oo) = f(+oo) =0
   %
   % Use: [fm,C,j,Merr,err] = BVP(kind,h,A,t)
   %
   % Input:  kind - |1 if the function H(t) of FIE has an exact expression
   %                |2 if the function H(t) of FIE is approximated by the
   %                                      Gauss-Laguerre quadrature rule
   %         h    - |H(t) right-hand side of the FIE if kind == 1
   %                |h(x) right-hand side of the BVP if kind == 2
   %         A    - A(x)=(mu a(x)-1)*exp(x^2/2)
   %         t    - row array of evaluation points
   %
   % Output: fm   - solution of the BVP
   %         j    - size of the solved linear system
   %         C    - condition number of the solved linear system
   %         Merr - maximum of absolute errors
   %         err  - row of absolute errors
   %
   %
   % Recalls: NystromMethodFIE.m
   %-------------------------------------------------------------------------
```

Function `BVP.m`, recalling function `NystromMethodFIE.m`, computes the Nyström interpolant f_m in (10), approximate solution of the BVP. If the exact expression of the right-hand side $H(t)$ in (4) is known, it is possible to use such expression providing it in the input handle function h and choosing `kind=1`. Otherwise, choosing `kind=2` the numerical approximation in (13) of $H(t)$ is computed by function `Hm.m`, providing the function $h(x)$ of the BVP in the input handle function h. In order to further improve the accuracy of the computed approximation when the function $\tilde{a}(x) = \mu a(x) - 1$ contains an exponential part, the input function $A(x) = \tilde{a}(x)e^{-\frac{x^2}{2}}$ is

required in place of $a(x)$.

Below there is the description of function `NystromMethodFIE.m`, implementing the Nyström method described in Section 2:

Code 2: description of NystromMethodFIE.m MatLab function

```
function [fm,C,j] = NystromMethodFIE(m,h,AA,t,kind)
   %-------------------------------------------------------------------------
   % File: NystromMethodFIE.m
   %
   % Goal: Compute the solution of the Fredholm Integral Equation (FIE)
   %
   %         +oo                          +oo
   %   f(t) - |G(x,t)(mu a(x)-1)f(x)dx = |G(x,t)h(x)dx
   %        -oo                          -oo
   %
   %
   % Use: [fm,C,j,m] = NystromMethodFIE(m,kind,h,A,t)
   %
   % Input:  m    - number of quadrature knots
   %         kind - |1 if the function H(t) of FIE has an exact expression
   %                |2 if the function H(t) of FIE is approximated by
   %                                      Gauss-Laguerre quadrature rule
   %         h    - |H(t) right-hand side of the FIE if kind == 1
   %                |h(x) right-hand side of the BVP if kind == 2
   %         A    - A(x)=(mu a(x)-1)*exp(x^2/2)
   %         t    - row array of evaluation points
   %
   % Output: fm   - solution of the BVP
   %         C    - condition number of the solved linear system
   %         j    - size of the solved linear system
   %
   %
   % Recalls: known.m, build.m, NystromInterpolant.m
   %-------------------------------------------------------------------------
```

Function `NystromMethodFIE.m` recalls the functions `known.m`, `build.m` and `NystromInterpolant.m` reported below. The first one computes the truncation parameter j in (11) and, then, the function $A(x)$ and the modified moments $M_k(x)$ at the Hermite zeros; the second one constructs the coefficient matrix and the right-hand side term of the linear system (9); the third one computes the Nyström interpolant f_m in (10).

Code 3: description of known.m MatLab function

```
function [j,x,w,M,Ax,y,wy]=known(m,A,kind)
   %-------------------------------------------------------------------------
   % File: known.m
   %
   % Goal: identify the truncation parameter j, evaluate the known function
   %       A(x) and the matrix of the modified moments M_k(x) at the Hermite
   %       zeros
   %
   % Use: [j,x,w,M,Ax,y,wy]=known(m,A,kind)
   %
   % Input:  m - number of quadrature knots
   %         A - A(x)=(mu a(x)-1)*exp(x^2/2)
   %         kind - |1 if the function H(t) has an exact expression
   %                |2 if the function H(t) is approximated by Gauss-Laguerre
   %                                      quadrature rule
   %
   % Output: j  - truncation parameter
   %         x  - row array of the Hermite zeros x_k, k=-j,...,j
   %         w  - row array of the weights w_k, k=-j,...,j m-point Gauss-Hermite rule
   %         M  - matrix of the modifeid moments
   %         Ax - A(x_k)=(mu a(x_k)-1)*exp((x_k)^2/2), k=-j,...,j
   %
   % Recalls: gaussq.m, Mk.m
   %-------------------------------------------------------------------------
```

Code 4: description of build.m MatLab function

```
function[Am,b] = build(j,x,y,wy,M,Ax,h,kind)
    %-------------------------------------------------------------------------
    % File: build.m
    %
    % Goal: Build the matrix and the right-hand side term of the linear system
    %
    % Use: [Am,b] = build(j,x,y,wy,M,Ax,kind,h)
    %
    % Input:  j  - size of the linear system
    %         x  - row array of the Hermite zeros x_k, k=-j,...,j
    %         y  - row array of the Laguerre zeros y_k, k=1,...,N, N=1024
    %         wy - row array of the weights of the N-point Gauss-Laguerre rule with N=1024
    %         M  - matrix of the modified moments
    %         Ax - A(x_k)=(mu a(x_k)-1)*exp((x_k)^2/2), k=-j,...,j
    %         kind - |1 if the function H(t) of FIE has an exact expression
    %                |2 if the function H(t) of FIE is approximated by
    %                                    Gauss-Laguerre quadrature rule
    %         h    - |H(t) right-hand side of the FIE if kind == 1
    %                |h(x) right-hand side of the BVP if kind == 2
    %
    % Output: Am - matrix of the linear system
    %         b  - right-hand side term of the linear system
    %
    % Recalls: Hm.m
    %-------------------------------------------------------------------------
```

Code 5: description of NystromInterpolant.m MatLab function

```
function [fm] = NystromInterpolant(m,x,w,z,Ax,h,y,wy,t,kind)
    %-------------------------------------------------------------------------
    % File: NystromInterpolant.m
    %
    % Goal: Compute the Nystrom interpolant
    %
    % Use: [fm] = NystromInterpolant(m,x,w,z,Ax,kind,h,y,wy,t)
    %
    % Input:  m - number of quadrature knots
    %         x - row array of the Hermite zeros x_k, k=-j,...,j
    %         w - row array of the weights w_k, k=-j,...,j m-point Gauss-Hermite rule
    %         Ax - A(x_k)=(mu a(x_k)-1)*exp((x_k)^2/2), k=-j,...,j
    %         z  - solution of the linear system
    %         kind - |1 if the function H(t) of FIE has an exact expression
    %                |2 if the function H(t) of FIE is approximated by
    %                                    Gauss-Laguerre quadrature rule
    %         h    - |H(t) right-hand side of the FIE if kind == 1
    %                |h(x) right-hand side of the BVP if kind == 2
    %         y  - row array of the Laguerre zeros y_k, k=1,...,N, N=1024
    %         wy - row array of the weights of the N-point Gauss-Laguerre rule with N=1024
    %         t  - row array of evaluation points
    %
    % Output: f - Nystrom interpolant, solution of the BVP
    %
    % Recalls: Mk.m, Hm.m
    %-------------------------------------------------------------------------
```

Both functions `known.m` and `NystromInterpolant.m` recall function `Mk.m` to compute the modified moments $M_k(t)$ in (14) at the points provided in the row array t. Function `known.m` also recalls function `gaussq.m` which computes the knots and the weights of Gauss-Hermite and Gauss-Laguerre quadrature rules. The latter only when `kind=2`.

Code 6: description of Mk.m MatLab function

```
function [M] = Mk(m,x,w,t)
    %-------------------------------------------------------------------------
    % File: M.m
    %
    % Goal: Build the matrix of the modified moment
    %                    +oo
    %           M_k(t) = | G(x,t)lmk(x) exp(-x^2/2)dx
```

```
    %                   -oo
    %
    % Use: [M] = Mk(m,x,w,t)
    %
    % Input:  m - number of quadrature knots
    %         x - row array of the Hermite zeros x_k, k=-j,...,j
    %         w - row array of the weights w_k, k=-j,...,j m-point Gauss-Hermite rule
    %         t - row array of evaluation points
    %
    % Output: M - matrix of the modifeid moments evaluated at points t
    %             size(M) = length(t)xm
    %
    % Recalls: Ci.m, Hermite.m
    %-------------------------------------------------------------------------
```

Function `Mk.m` recalls functions `Ci.m` and `Hermite.m` reported below. `Ci.m` computes the integrals $c_i(t)$ in (14) using the recurrence scheme (15). `Hermite.m` computes the orthonormal Hermite polynomials using their three term recurrence relation. The latter is also recalled by `Ci.m`.

Code 7: description of Ci.m MatLab function

```
function [c]=Ci(m,t)
    %-------------------------------------------------------------------------
    % File: Ci.m
    %
    % Goal: Evaluate the integrals
    %             +oo
    %    c_i(t) = | G(x,t)p_i(x) exp(-x^2/2)dx,  k=0,...,m
    %            -oo
    %       using a recurrence scheme
    %
    % Use: [c] = Ci(m,t)
    %
    % Input:  m - number of Hermite zeros
    %         t - row array of evaluation points
    %
    % Output: c - matrix (c_i(t_j))
    %             size(c) = (m+1)xlength(t)
    %
    % Recalls: Hermite.m
    %-------------------------------------------------------------------------
```

Code 8: description of Hermite.m MatLab function

```
function [p]=Hermite(m,t)
    %-------------------------------------------------------------------------
    % File: Hermite.m
    %
    % Goal: Construct the orthonormal Hermite polynomials from degree 0 to
    %       degree m+1 evaluated at the points t
    %
    % Use: [p] = Hermite(m,x)
    %
    % Input:  m - degree of the polynomial
    %         t - row array of evaluation points
    %
    % Output: p - matrix of the orthonormal Hermite polynomials
    %             size(H) = length(t)x(m+1)
    %-------------------------------------------------------------------------
```

Finally, function `Hm.m` is described below. It is recalled by functions `build.m` and `NystromInterpolant.m` in order to compute the numerical approximation (13) of $H(t)$ in (12) when the choice `kind=2` is done.

Code 9: description of Hm.m MatLab function

```
function [H] = Hm(m,h,y,wy,t)
    %-------------------------------------------------------------------------
    % File: Hm.m
    %
    % Goal: Approximate the integral
    %               +oo
    %       H(t) = | G(x,t)h(x) dx,
    %              -oo
    %       using the N-point truncated Gauss-Laguerre quadrature rule
    %
    % Use: [H] = Hm(m,h,y,wy,t)
    %
    % Input:  m  - number of Laguerre knots
    %         h  - h(x) right-hand side of the BVP
    %         y  - row array of the Laguerre zeros y_k, k=1,...,N, N=1024
    %         wy - row array of the weights of the N-point Gauss-Laguerre rule with N=1024
    %         t  - row array of evaluation points
    %
    % Output: Hm -  column array of the approximations of H(t)
    %               length(Hm) = length(t)
    %-------------------------------------------------------------------------
```

4 Demos

The codes have been written using MATLAB R2021b and the simulations have been run on a personal computer with the following features: MacBook Pro 2,2 GHz Intel Core i7 quad-core, 16 GB Memory MHz DDR3, Intel Iris Pro 1536 MB. In the Demos folder of the package we have introduced eight different demos in order to show its performances:

`BVP_demoi.m, i=1,...,4,` return the graphs of the approximate solutions obtained with $m = 512$ together with the corresponding truncation parameters j, condition numbers of the involved linear systems and maximum absolute errors. In particular the graphs produced by `BVP_demo1.m` and `BVP_demo2.m` correspond to Figure 2 and Figure 3 in [6], respectively.

`BVP_demoi_1.m, i=1,...,4,` return both a table containing in each row the values of m, j, $\mathrm{cond}(A_m)$ and err_m, for increasing values of m, and the plot of the trend of the absolute error err_m. In particular, running `BVP_demo1_1.m` and `BVP_demo2_1.m` it is possible to display Table 3 and Table 4 in [6], respectively.

Note that, in `BVP_demo3.m` and `BVP_demo3_1.m` a BVP problem having the boundary condition $f(\pm\infty) = b$ with $b \neq 0$ is considered. Moreover, `BVP_demo4.m` and `BVP_demo4_1.m` deal with a BVP where, although the functions $a(x)$ and $h(x)$ do not satisfy the sufficient conditions given by Theorem 2.1, the implemented method is still stable and convergent.

For the sake of brevity here we will show only the results of `BVP_demoi.m`, $i = 1,\ldots,4$, and `BVP_demo1_1.m`.

4.1 BVP_demo1.m

We consider the following BVP

$$\begin{cases} f''(x) - \left(e^{-\frac{3}{7}x^2} + 1\right) f(x) = \dfrac{1}{25} e^{-\frac{37}{70}x^2} \left[e^{\frac{3}{7}x^2}(x^2 - 30) - 25\right] \\ f(\pm\infty) = 0. \end{cases} \tag{16}$$

Since $\tilde{a}(x) = e^{-\frac{3}{7}x^2}$ it results $A(x) = e^{\frac{x^2}{14}}$. The integral $H(t)$ in (12) has been approximated choosing `kind=2`.

Down below the outputs after running `BVP_demo1.m` are reported:

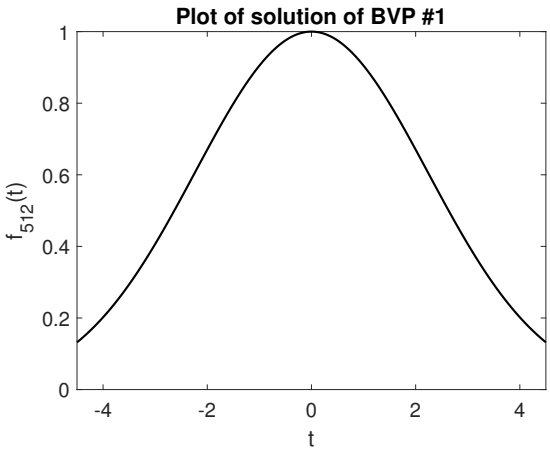
```
Welcome to BVP demo #1

m, truncation index j, condition number and maximum error:

512,  150,  2.2664e+00,    1.89e-15.

Plot of solution of BVP #1

Execution time for evaluating f_{512}(t): 3.0031 s
```



BVP_demo1_1.m

Here are reported the outputs obtained with `BVP_demo1_1.m` to solve (16):

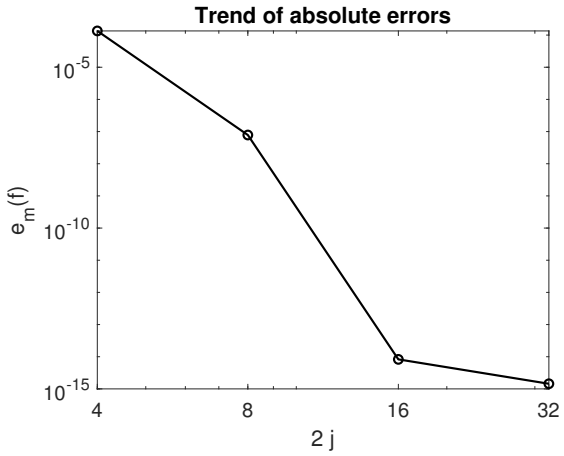
```
Welcome to BVP demo #1_1

Table:

m & j & cond(A_m) & err_m
4   & 4   & 1.6156e+00   & 1.35e-04
8   & 8   & 1.7917e+00   & 7.82e-08
16  & 16  & 1.8553e+00   & 8.27e-15
32  & 32  & 1.9532e+00   & 1.33e-15

Trend of absolute errors

Execution time: 4.1607 s
```



4.2 BVP_demo2.m

Now we take the following BVP

$$\begin{cases} f''(x) - \left(\dfrac{1}{x^8+5} + 1\right) f(x) = |x|^{\frac{13}{2}} e^{-\frac{3}{2}x^2} \\ f(\pm\infty) = 0. \end{cases}$$

We have $\tilde{a}(x) := \frac{1}{x^8+5}$ and, then, $A(x) = \frac{1}{x^8+5} e^{\frac{x^2}{2}}$. Moreover, we set `kind=2` to approximate the integral $H(t)$ in (12). The results obtained are the following:

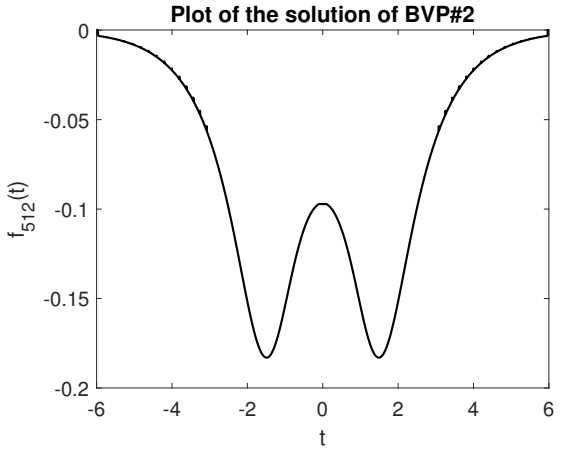
```
Welcome to BVP demo #2

m, truncation index j, condition number and maximum error:

512,  146,  1.2626e+00,    2.32e-10.

Plot of the solution of BVP #2

Execution time for evaluating f_{512}(t): 5.8409 s
```



4.3 BVP_demo3.m

As an example of a BVP with a non-zero boundary condition, let us consider the following

$$\begin{cases} f''(x) - 2\left(e^{-\frac{x^2}{10}} + 1\right) f(x) = -8 + e^{-\frac{11}{10}x^2}\left[4e^{\frac{x^2}{10}} - 8e^{x^2} - 2(x^2-1)\right] \\ f(\pm\infty) = 4, \end{cases}$$

where $\tilde{a}(x) := e^{-\frac{x^2}{10}}$, $A(x) = 2e^{\frac{2}{5}x^2} + e^{\frac{x^2}{2}}$. Choosing `kind=2`, we get the following outputs:

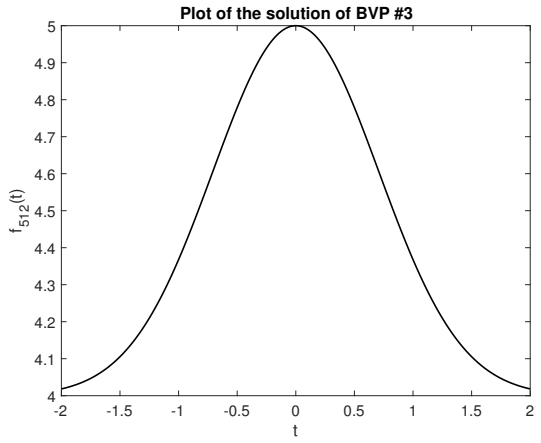
```
Welcome to BVP demo #3

m, truncation index j, condition number and maximum error:

512,  154,  8.9206e+00,    1.74e-07.

Plot of the solution of BVP #3

Execution time for evaluating f_{512}(t): 2.6860 s
```



4.4 BVP_demo4.m

Finally, we propose the following BVP

$$\begin{cases} f''(x) - \dfrac{1}{2}e^{-\frac{x^2}{15}} f(x) = -\dfrac{1}{5}e^{-\frac{17}{30}x^2} - e^{-\frac{x^2}{2}} + x^2 e^{-\frac{x^2}{2}} \\ f(\pm\infty) = 0, \end{cases}$$

whose involved functions do not respect the hypotheses of Theorem 2.1. This test was performed knowing that the exact solution is $f(x) = e^{-\frac{x^2}{2}}$ and taking in input $A(x) = \left(\frac{1}{2}e^{-\frac{x^2}{15}}\right)e^{\frac{x^2}{2}}$ and `kind=2`. The method is still convergent, as shown by the following outputs:

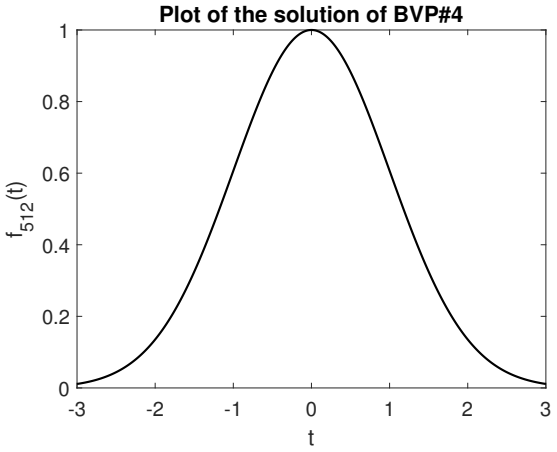
```
Welcome to BVP demo #4

m, truncation index j, condition number and maximum error:

512,  150,  8.7636e+00,    3.91e-15.

Plot of the solution of BVP #4

Execution time for evaluating f_{512}(t): 2.9480 s
```



5 The GUI MatLab BVP_Real_Axis

In order to make the proposed package easier to use, we have provided it with an user friendly GUI Matlab. To get started, users can type BVP_Real_Axis into the MatLab command line. The window in Figure 2 will appear.

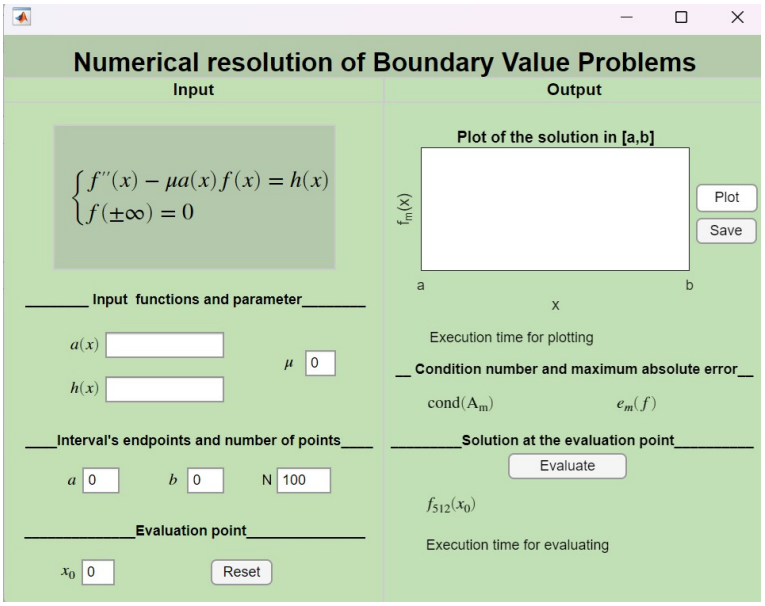
It is divided into two parts: the Input and the Output moduli. In the Input modulus the users have to enter the necessary input functions and parameters such as:

- the functions $a(x)$, $h(x)$ and the parameter μ defining the BVP problem;
- the endpoints a and b of the interval where the approximate solution of the BVP has to be plotted and the number N of the discretization points over $[a,b]$ (the default value is $N = 100$):
- the point x_0 where the approximate solution has to be evaluated.

By clicking the “Reset" button, all entered input values are removed. By clicking the “Plot” button on the Output mudulus, the graph of the approximate solution f_{512} in (10) of the BVP problem is plotted on the interval $[a,b]$ and displayed in the dedicated box. The following values are also displayed:

- the execution time required to draw the graph;
- the condition number $\mathrm{cond}(A_m)$ of the solved linear system;

Figure 2



- the maximum absolute error related to the provided approximation of the solution. Such error is computed assuming as exact the Nyström interpolant f_{600}.

Users can save the graph as a JPEG image by clicking on the "Save" button. Finally, by clicking on the "Evaluate" button, the value of $f_{512}(x)$ at $x = x_0$ is displayed together with the related execution time.

Several input data controls have been employed. Moreover, the following error messages will be displayed:

- '`The method is not applicable`' if the computed approximation is `NaN`;
- '`Results may be inaccurate`' if the condition number of the solved linear system is greater than `1/eps`;
- '`The method is not convergent`' if the maximum absolute error is greater than 1.

6 Conclusions

In this paper, we have introduced a new MatLab Package, called BVP_Real_Axis, with the aim of providing a user friendly package to solve numerically second-order boundary value problems on the real axis. The implemented numerical method consists in approximating the solution of an equivalent Fredholm integral equation of the second kind by a Nyström type method. The package is also equipped with an easy-to-use graphical interface.

The package is freely available on the GitHub repository https://github.com/ValeriaSagaria/BVP_Real_Axis.

Acknowledgements

The first author has been supported by University of Basilicata (local funds) and by PRIN 2022 PNRR project AQuAInt Grant P20229RMLB, CUP-C53D23008400001, financed by the

European Union - NextGeneration EU and by the Italian Ministry of University and Research (MUR).

The second author has been partially supported by the Next Generation EU-Italian MUR PNRR Tech4You project “Technologies for climate change adaptation and quality of life improvement", n. ECS 00000009 - CUP C43C22000400006.

The authors have been supported by GNCS Project 2025 - CUP E53C24001950001, “Metodi di approssimazione globale per operatori integrali e applicazioni alle equazioni funzionali". This research has been accomplished within RITA (Research ITalian network on Approximation) and the UMI Group TAA (Approximation Theory and Applications).

The authors are members of the Gruppo Nazionale per il Calcolo Scientifico-Istituto Nazionale di Alta Matematica (GNCS-INdAM).

Declaration of interests

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

[1] R. P. Agarwal and D. O’Regan, *Singular problems on the infinite interval modelling phenomena in draining flows*, IMA J. Appl. Math., vol. 66, no. 6 (2001) 621–635, URL http://dx.doi.org/10.1093/imamat/66.6.621

[2] R. P. Agarwal and D. O’Regan, *Infinite interval problems modelling the flow of a gas through a semi-infinite porous medium*, Stud. Appl. Math., vol. 108 (2002) 245–257, URL http://dx.doi.org/https://doi.org/10.1111/1467-9590.01411

[3] R. P. Agarwal and D. O’Regan, *Infinite interval problems arising in non-linear mechanics and non-Newtonian fluid flows*, Internat. J. Non-Linear Mech., vol. 38, no. 9 (2003) 1369–1376, URL http://dx.doi.org/https://doi.org/10.1016/S0020-7462(02)00076-8

[4] R. P. Agarwal and D. O’Regan, *Infinite interval problems modeling phenomena which arise in the theory of plasma and electrical potential theory*, Stud. Appl. Math., vol. 111, no. 3 (2003) 339–358, URL http://dx.doi.org/https://doi.org/10.1111/1467-9590.t01-1-00237

[5] M. C. De Bonis and V. Sagaria, *Numerical method for boundary value problems on the real line*, Appl. Numer. Math., vol. 200 (2024) 179–194, URL http://dx.doi.org/10.1016/j.apnum.2023.05.016

[6] M. C. De Bonis and V. Sagaria, *A new Nyström method for solving boundary value problems on the real axis*, J. Comput. Appl. Math., vol. 478 (2026) 117236, URL http://dx.doi.org/10.1016/j.cam.2025.117236

[7] R. W. Dickey, *Membrane caps under hydrostatic pressure*, Quart. Appl. Math., vol. 46, no. 1 (1988) 95–104, URL http://dx.doi.org/https://doi.org/10.1090/qam/934684

[8] R. W. Dickey, *Rotationally symmetric solutions for shallow membrane caps*, Quart. Appl. Math., vol. 47, no. 3 (1989) 571–581, URL http://dx.doi.org/https://doi.org/10.1090/qam/1012280

[9] L. Erbe and K. Schmitt, *On radial solutions of some semilinear elliptic equations*, Differ. Integral Equ., vol. 1, no. 1 (1988) 71–78, URL http://dx.doi.org/10.57262/die/1371747583

[10] G. Golub and J. Welsch, *Some modified matrix eigenvalue problems*, Siam Review, vol. 15 (1973) 318–334, URL http://dx.doi.org/10.1137/1015032

[11] G. H. Golub and J. H. Welsch, *Calculation of Gauss Quadrature Rules*, Mathematics of Computation, vol. 23, no. 106 (1969) 221–s10, URL http://www.jstor.org/stable/2004418

[12] A. Granas, R. B. Guenther, J. W. Lee, and D. O'Regan, *Boundary value problems on infinite intervals and semiconductor devices*, J. Math. Anal. Appl., vol. 116, no. 2 (1986) 335–348, URL http://dx.doi.org/https://doi.org/10.1016/S0022-247X(86)80002-6

[13] M. Greguš, *On a special boundary value problem*, Acta Math. Univ. Comenian., vol. 40/41 (1982) 161–168

[14] R. E. Kidder, *Unsteady flow of gas through a semi-infinite porous medium*, J. Appl. Mech., vol. 27 (1957) 329–332, URL http://dx.doi.org/https://doi.org/10.1115/1.4011542

[15] T. Laudadio, N. Mastronardi, and D. Occorsio, *Computing integrals with an exponential weight on the real axis in floating point arithmetic*, Appl. Math. Comput, vol. 200 (2023) 309–317, URL http://dx.doi.org/https://doi.org/10.1016/j.apnum.2023.05.025

[16] G. Maise and A. J. Sabadell, *Electrostatic probe measurements in solid-propellant rocket exhausts*, AIAA Journal, vol. 8, no. 5 (1970) 895–901, URL http://dx.doi.org/https://doi.org/10.2514/3.5784

[17] G. Mastroianni and G. V. Milovanović, *Interpolation processes*, Springer Monographs in Mathematics, Springer-Verlag, Berlin (2008), ISBN 978-3-540-68346-9, URL http://dx.doi.org/10.1007/978-3-540-68349-0

[18] G. Mastroianni and G. Monegato, *Truncated Gauss-Laguerre quadrature rules*, in *Recent trends in numerical analysis*, Nova Sci. Publ., Huntington, NY, vol. 3 of *Adv. Theory Comput. Math.*, 213–221 (2001), ISBN 1-56072-885-X

[19] G. Mastroianni and I. Notarangelo, *A Lagrange-type projector on the real line*, Math. Comp., vol. 79, no. 269 (2010) 327–352, URL http://dx.doi.org/10.1090/S0025-5718-09-02278-9

[20] T. Y. Na, *Computational methods in engineering boundary value problems*, vol. 145 of *Mathematics in Science and Engineering*, Academic Press Inc., New York-London (1979), ISBN 0-12-512650-6, URL https://www.sciencedirect.com/bookseries/mathematics-in-science-and-engineering/vol/145/suppl/C

[21] G. Szegö, *Orthogonal Polynomials*, vol. 23 of *American Mathematical Society Colloquium Publications*, American Mathematical Society, New York (1939)