A Bootstrap Method for Sum-of-Poles Approximations

Kuan Xu & Shidong Jiang

Journal of Scientific Computing

ISSN 0885-7474

J Sci Comput
DOI 10.1007/s10915-012-9620-9

Springer

J Sci Comput
DOI 10.1007/s10915-012-9620-9

A Bootstrap Method for Sum-of-Poles Approximations

Kuan Xu · Shidong Jiang

Received: 8 October 2011 / Revised: 1 May 2012 / Accepted: 8 June 2012

Abstract A bootstrap method is presented for finding efficient sum-of-poles approximations of causal functions. The method is based on a recursive application of the nonlinear least squares optimization scheme developed in (Alpert et al. in SIAM J. Numer. Anal. 37:1138–1164, 2000), followed by the balanced truncation method for model reduction in computational control theory as a final optimization step. The method is expected to be useful for a fairly large class of causal functions encountered in engineering and applied physics. The performance of the method and its application to computational physics are illustrated via several numerical examples.

Keywords Rational approximation · Sum-of-poles approximation · Model reduction · Balanced truncation method · Square root method

1 Introduction

Suppose that $f \in H_+^\infty$, where H_+^∞ is the Hardy space containing all functions that are analytic in the open right half of the complex plane and bounded in the closed right half-plane (see, for example, [12, 13, 19, 40]). Such a function is called causal function in the literature of electrical engineering and physics. Indeed, when a causal function is used as the transfer function of a control system, the output and internal states of the system depend only on the current and previous input values (see, for example, [13, 43]). While in electromagnetics the complex permittivity function belongs to H_+^∞ indicating the fact that the induced polarization field can depend only on the current and previous electric field (see, for example,

K. Xu (✉)
One Research Circle, GE Global Research, Niskayuna, NY 12309, USA
e-mail: kuan.xu@njit.edu

S. Jiang
Department of Mathematical Sciences, New Jersey Institute of Technology, University Heights, Newark, NJ 07102, USA
e-mail: shidong.jiang@njit.edu

Published online: 22 June 2012

[11, 20]). Causal functions also arise in the construction of the exact nonreflecting boundary conditions of many time-dependent partial differential equations where they appear as Laplace transforms of the convolution kernels, due to the fact that the nonreflecting boundary conditions are often nonlocal in both space and time but still causal (see, for example, [5–7, 17, 21, 22]). We also assume that f decays to zero at infinity. However, very often f is not smooth on the imaginary axis and may have, say, branch points on the imaginary axis.

In this paper, we consider an efficient approximation of f by the sum of a small number of poles. That is, we try to find two sets of complex numbers $\{w_k\}$ and $\{z_k\}$ with $k = 1, \ldots, N$ such that

$$\left| f(iy) - \sum_{k=1}^{N} \frac{w_k}{iy - z_k} \right| < \epsilon, \quad y \in \mathbb{R}, \tag{1}$$

where i is the unit imaginary number and ϵ is the prescribed precision. We also require that all the poles $\{z_k\}$ lie on the open left half of the complex plane and thus (1) implies that

$$\left| f(z) - \sum_{k=1}^{N} \frac{w_k}{z - z_k} \right| < \epsilon, \quad z \in \mathbb{C}^+,$$

This approximation is useful in a number of applications. For example, in control theory it can be used to obtain a reduced system with a much smaller number of parameters which is very close to the original system (see, for example, [13, 18, 23, 34, 37, 38, 41, 43]). In computational electromagnetics, it can be used to compute the fractional derivatives in the Havriliak-Negami dielectric model of induced polarization (see, for example, [11]). The sum-of-poles approximation is also used to speed up the computation of nonreflecting boundary conditions for the wave and the Schrödinger equations (see, for example, [5, 6, 21, 22]). For some of other recent applications of sum-of-poles approximation, please see, for example, [26, 27].

This problem has been investigated theoretically by Adamjan, Arov, and Kreĭn using infinite Hankel matrices [2–4, 36]. When the function can be expressed as a contour integral and thus the poles are known to lie on some contour in the complex plane, such approximation can be obtained via some special quadratures [47] or generalized Gaussian quadrature [10, 33, 48]. However, there seems to be very few algorithms for directly constructing such an approximation when the only information about the poles is that they must lie in the open left half-plane. Indeed, the well-known exchange algorithm deals with the real rational approximation of a real function on an interval (see, for example, Chap. 10 in [39]). In a series of papers [14–16, 44–46], Gutknecht and Trefethen et al. introduced the so-called CF method for computing complex rational approximations for smooth functions based on the finite Hankel matrices resulting from the discretization with equispaced points. Their method can be applied to nonsmooth functions as well, but may result in either a very large number of poles or low accuracy near the singular points of the given function. In [5] (see also [6] for its applications on the wave equation), Alpert et al. developed a nonlinear least squares algorithm for such problem when the pole locations are roughly known asymptotically, upon which our bootstrap method is built.

Since the Laplace transform of an exponential function is a pole, a closely related problem is approximation by sum-of-exponentials. Indeed, in many applications including the nonreflecting boundary conditions for some time-dependent partial differential equations, one often needs to evaluate a convolution integral of the form $I(t) = \int_0^t K(t-\tau)\sigma(\tau)\,d\tau$, where K is the convolution kernel and σ is a given or unknown function. Furthermore,

the kernel K is not given explicitly, but rather its Laplace transform is known. That is, $K = \mathcal{L}^{-1}(f)$ for some known function f. Clearly, the direct method of evaluating the convolution integral at N_T time steps will take $O(N_T^2)$ operations, rendering the long term simulations unfeasible. But if we can find an efficient and accurate sum-of-poles approximation for the Laplace transform of the kernel, i.e., $f \approxeq \sum_{j=1}^{N_p} \frac{w_j}{z-p_j}$, then $K(t-\tau) \approxeq \sum_{j=1}^{N_p} w_j e^{p_j(t-\tau)}$ and we readily have

$$I(t) \approxeq \sum_{j=1}^{N_p} w_j \int_0^t e^{p_j(t-\tau)}\sigma(\tau)\,d\tau := \sum_{j=1}^{N_p} w_j C_j(t).$$

Now, it is well known that the convolution with exponential functions can be computed using a recurrence relation

$$C_j(t) = \int_0^t e^{p_j(t-\tau)}\sigma(\tau)d\tau = e^{p_j\Delta t}C_j(t-\Delta t) + \int_{t-\Delta t}^t e^{p_j(t-\tau)}\sigma(\tau)\,d\tau.$$

With the help of the above recurrence relation, each mode C_j can be computed in $O(1)$ operations at each time step and the total computational cost is reduced from $O(N_T^2)$ to $O(N_pN_T)$, where N_p is the number of poles in the approximation of f. Very often, one can show that N_p is of order $\log N_T$ and the computational cost is reduced to near optimal.

We would like to remark here that when the function f is sectorial, López-Fernández et al. have developed a spectral order method for inverting the Laplace transform and thus obtained an efficient sum-of-exponentials approximation for the convolution kernel K directly (see, for example, [28, 29] for this method and [30–32, 42] for its applications on fast and oblivious convolution quadrature). Trefethen et al. have presented a detailed comparison and discussion about inverting sectorial Laplace transforms using the trapezoidal or midpoint rule for discretizing various contour integrals in [47]. Finally, we would like to point out that sum-of-exponentials approximation has many other applications by itself and may be obtained via other methods (for a detailed discussion please see Beylkin et al. [8, 9]).

In this paper, we propose a bootstrap method for computing an efficient sum-of-poles approximation for functions which are not very smooth on the imaginary axis. We assume that the only known information on f is its value along the imaginary axis which is supplied by a black-box subroutine. Our algorithm is roughly as follows. We first construct a binary tree structure containing successively larger intervals centered around the singular point of the function. We then apply the nonlinear least squares procedure in [5] on each subinterval in the binary tree to extract out the so-called near poles recursively. Finally, we apply the balanced truncation method in model reduction (see, for example, [13, 18, 25, 34]) to reduce the number of poles.

The paper is organized as follows. The nonlinear least squares method in [5] and the balanced truncation method are reviewed in Sect. 2. The bootstrap method is presented in Sect. 3. Section 4 contains numerical experiments on an *ad hoc* example, the kernels of the Havriliak-Negami dielectric model, and the convolution kernels in the nonreflecting boundary conditions for the Schrödinger equation. Finally, we give a short discussion in Sect. 5.

2 Numerical Preliminaries

In this section, we review the nonlinear least squares method in [5] and the balanced truncation method. We only present the tailored version which can be readily incorporated into our bootstrap algorithm.

2.1 Nonlinear Least Squares Method

Given a complex-valued function $f \in L^2[a,b]$ and a positive integer d, the nonlinear least squares method in [5] tries to solve the following minimization problem

$$\min_{P,Q} \int_a^b \left| \frac{P(z)}{Q(z)} - f(z) \right|^2 dz, \tag{2}$$

where P and Q are polynomials with $\deg(P)+1 = \deg(Q) = d$ and the leading coefficient of Q is normalized to 1. Assuming that a good initial guess of Q is available, [5] tries to find the solution to (2) iteratively by solving the following linear least squares problem at each iteration step:

$$\min_{P^{(i+1)},Q^{(i+1)}} \int_a^b \left| \frac{P^{(i+1)}(z)}{Q^{(i)}(z)} - \frac{Q^{(i+1)}(z)}{Q^{(i)}(z)} f(z) \right|^2 dz. \tag{3}$$

Straightforward computation shows that problem (3) is equivalent to the following linear system:

$$\left\langle -P^{(i+1)}(z) + Q^{(i+1)}(z) f(z),\, f_i(z) \right\rangle = 0 \quad \text{for } i = 1, \ldots, 2d, \tag{4}$$

where

$$f_i(z) = \begin{cases} z^{(i-1)/2} f(z), & \text{for odd } i \\ z^{i/2-1}, & \text{for even } i \end{cases}$$

and the inner product is defined by

$$\langle f, g \rangle = \int_a^b \frac{f(z)\bar{g}(z)}{|Q^{(i)}(z)|^2} dz. \tag{5}$$

However, we do not solve (3) by representing P, Q in terms of their monomial coefficients and forming the corresponding matrix according to (4) for two reasons. First, the condition number of the resulting matrix is extremely large. Second, the rootfinding and evaluation of a polynomial in terms of its monomial coefficients are also very ill-conditioned. Instead, [5] solves (3) by Gram-Schmidt orthogonalization. The $2d+1$ functions

$$f, 1, zf, z, \ldots, z^{d-1} f, z^{d-1}, z^d f$$

are orthogonalized with respect to the inner product (5) to obtain the orthogonal functions

$$g_n(z) = \begin{cases} f(z), & n = 1 \\ 1 - c_{21} g_1(z), & n = 2 \\ z g_{n-2}(z) - \sum_{j=2}^{\min\{4,n-1\}} c_{nj} g_{n-j}(z), & n = 3, \ldots, 2d+1 \end{cases} \tag{6}$$

Algorithm 1 Nonlinear least squares method

Comment: Given $f \in L^2[a,b]$, a positive integer d, and an initial guess Q_0, find polynomials P and Q with $\deg(P)+1 = \deg(Q) = d$ so that $\int_a^b |\frac{P(z)}{Q(z)} - f(z)|^2 dz$ is minimized.

1: **while** iter $<$ iter$_{\max}$ **do**
2: Use Gram-Schmidt orthogonalization to construct P and Q such that $\int_a^b |\frac{P(z)}{Q_0(z)} - \frac{Q(z)}{Q_0(z)} f(z)|^2 dz$ is minimized;
3: Set $Q_0(z) = Q(z)$, where $Q(z)$ is obtained from the previous step;
4: iter $=$ iter $+ 1$.
5: **end while**

where

$$c_{nj} = \frac{\langle z g_{n-2}, g_{n-j}\rangle}{\langle g_{n-j}, g_{n-j}\rangle}, \quad \text{for } n = 3, \ldots, 2d+1, \ j = 1, \ldots, \min\{4, n-1\}.$$

Remark 1 It is easy to see that the orthogonal functions g_n satisfy the five-term recurrence relation (6) by noting that $\langle z g_{n-2}, g_j\rangle = \langle g_{n-2}, z g_j\rangle$ on the imaginary axis, $z g_j$ are linear combinations of $g_1, g_2, \ldots, g_{j+2}$, and thus $\langle z g_{n-2}, g_j\rangle = 0$ for $j < n-4$.

We now observe that

$$g_{2d+1} = Q^{(i+1)} f - P^{(i+1)},$$

so $P^{(i+1)}$ and $Q^{(i+1)}$ are computed from the recurrence coefficients c_{nj} by splitting it into even- and odd-numbered parts.

2.2 The Balanced Truncation Method

Given a rational function of the form

$$R(z) = \sum_{i=1}^{n} \frac{w_i}{z - p_i}$$

with all the poles p_i lie on the open left half of complex plane, Algorithm 2 tries to find another rational function of the same form

$$\hat{R}(z) = \sum_{i=1}^{k} \frac{\hat{w}_i}{z - \hat{p}_i}$$

but with much fewer poles (i.e., $k \ll n$) such that $|R(z) - \hat{R}(z)| < \epsilon$ for all $z \in \mathbb{C}^+$ for some prescribed precision ϵ. The algorithm is a special case of the general balanced truncation method for model reduction in computational control theory (for a comprehensive review on the balanced truncation method, see, for example, [13]). Here we would like to remark that Algorithm 2 has been applied to find an efficient sum-of-poles approximation for the spherical nonreflecting boundary kernel for the wave equation in [25].

Algorithm 2 Balanced truncation method

Comment: Given $R(z) = \sum_{i=1}^{n} \frac{w_i}{z-p_i}$ and a prescribed precision ϵ, find $\hat{R}(z) = \sum_{i=1}^{k} \frac{\hat{w}_i}{z-\hat{p}_i}$ so that $|R(z) - \hat{R}(z)| < \epsilon$ for $z \in \mathbb{C}^+$.

1: Form a diagonal matrix $A = \text{diag}(p_1, p_2, \ldots, p_n)$, a row vector $C = (\sqrt{w_1}, \sqrt{w_2}, \ldots, \sqrt{w_n})$, and a column vector $B = C^T$;
2: Compute the Cholesky factor S of the solution to the Lyapunov equation $AP + PA^* = -BB^*$;
3: Compute the Cholesky factor L of the solution to the Lyapunov equation $AQ + QA^* = -CC^*$;
4: Compute the singular value decomposition of $LS^* = U\Sigma V^*$, where the singular values σ_i $(i = 1, \ldots, n)$ are also the Hankel singular values of the dynamic system with R as the transfer function;
5: Find k such that $2\sum_{i=k+1}^{n} \sigma_i \le \epsilon$;
6: Form an $n \times k$ matrix J whose nonzero entries are $J_{ii} = \sigma_i^{-1/2}$, then form two matrices $T_l = L^*UJ$ and $T_r = S^*VJ$;
7: Form a $k \times k$ matrix $\hat{A} = T_l^* A T_r$, a column vector of length k $\hat{B} = T_l^* B$, and a row vector of length k $\hat{C} = CT_r$;
8: Compute the eigenvalue decomposition of $\hat{A} = X\Lambda X^{-1}$ and set $\hat{p}_i = \Lambda_{ii}$, $i = 1, \ldots, k$;
9: Compute $\tilde{B} = X^{-1}\hat{B}$, $\tilde{C} = \hat{C}X$, and set $\hat{w}_i = \tilde{B}_i\tilde{C}_i$ $(i = 1, \ldots, k)$.

3 The Bootstrap Algorithm

3.1 An Informal Description

We now consider our main problem, that is, given a function $f \in H_+^\infty$ and a prescribed precision ϵ, find a sum-of-poles approximation $r(z) = \sum_{i=1}^{n} \frac{w_i}{z-z_i}$ such that $|f(z) - r(z)| < \epsilon$ for all $z \in C^+$. We observe that Algorithm 1 is fairly robust and can produce very accurate sum-of-poles approximations for a function on a finite interval $[a, b]$ if the following conditions hold. First, the function being approximated is smooth on the interval $[a, b]$. Second, the number of poles needed in the approximation is small, say, less than 20. Third, a good initial guess for poles can be obtained by some means. However, none of the above three conditions hold for a general function in H_+^∞. And Algorithm 1 fails to converge due to the ill-conditioning of the problem when we try to apply it to the given function on a very large interval on the imaginary axis.

Our bootstrap algorithm tries to find an accurate and efficient sum-of-poles approximation to f in the following steps. First, we choose a large interval $[A, B] = [a_0, b_0]$ on the imaginary axis so that if $|f(iy) - r(iy)| < \epsilon$ for any $y \in [A, B]$ and some sum-of-poles approximation r, then $|f(z) - r(z)| < \epsilon$ for all $z \in C^+$. This is possible since both f and r decay to zero as $z \to \infty$. Second, we construct a sequence of nested intervals $[a_i, b_i]$ $(i = 0, 1, \ldots, L)$ with $[a_i, b_i] \subset [a_{i-1}, b_{i-1}]$ $(i = 1, \ldots, L)$ so that the smallest interval $[a_L, b_L]$ is centered around the singular point of f if any. Third, starting from the smallest interval $[a_L, b_L]$, we apply Algorithm 1 to find an accurate sum-of-poles approximation r_i to f_i on $[a_i, b_i]$ successively for $i = L, L-1, \ldots, 0$; we initialize $f_L = f$ (f is the original given function) and set $f_i = f_{i+1} - r_{i+1}^n$, where r_{i+1}^n is the near-pole part of r_{i+1}. The near-pole part of r_i contains the contribution of poles that are close to the interval $[a_i, b_i]$; this will be defined more rigorously later. Finally, we form $r = \sum_{i=1}^{L} r_i^n + r_0$ and r is an accurate (but not necessarily efficient) sum-of-poles approximation to f; we use Algorithm 2 to reduce r to a more efficient approximation R without losing accuracy.

Remark 2 When the function has multiple singular points on the imaginary axis, one needs to refine the interval around each singular point.

The third step in the above algorithm is the bootstrapping step, where we apply Algorithm 1 iteratively to find sum-of-poles approximation of f_i on larger intervals $[a_i, b_i]$. The function f_i is formed by subtracting the near-pole contribution from f. The idea here is that we could either shrink the interval or modify the function to make it less ill-conditioned so that Algorithm 1 can be applied successfully. And the function f_i has similar structure with respect to the interval $[a_i, b_i]$ for all $i = L, \ldots, 0$ in the sense that the integral $\int_{a_i}^{b_i} f_i(iy)\,dy$ can be computed accurately using about the same number of nodes. We would like to provide an intuitive explanation about why bootstrapping works. In the Fourier domain, both the near poles and the far poles have contributions at low frequencies, but the high frequency contribution comes mainly from the near poles. So it is clear that if we want to extract out the poles successively, we should extract the near poles first. We now observe that the near poles are localized in the physical space, so the restriction to a smaller interval in the physical space will retain all the essential information about the near poles. Therefore, if we confine the approximation problem to a smaller interval, then only the poles near that interval matter and hence we need many fewer poles to approximate the given function. Hence, by restricting our attention to a smaller interval, we effectively mitigate the ill-conditioning of the problem and can extract the near poles accurately.

We now fill in some details of the algorithm.

3.2 Binary Tree Structure

To obtain a sequence of nested intervals $[a_i, b_i]$ $(i = 0, \ldots, L)$, we will first build a binary tree of intervals. We start from the root interval $[a_0, b_0]$, and recursively subdivide each interval into two smaller intervals until some splitting criterion is no longer satisfied. We use standard binary tree notation so that each node has a parent, left, and right child pointers, and a tree depth. Each node also contains the locations of its two end points a and b. The root interval is defined to have depth 0. In the construction, we record the maximal depth of the tree for later use. Our algorithm is as follows (Algorithm 3).

Once this binary tree is constructed, the sequence of nested subintervals is easily obtained by starting from the leaf of maximal depth and traversing up to the root interval. We now explain our splitting criterion in detail.

3.3 Splitting Criterion

We first compute the K-term Chebyshev interpolant $\bar{f}(y)$ of $f(iy)$ on the interval $[a, b]$ as follows. We translate and scale $[a, b]$ to $[-1, 1]$ by

$$x = \frac{2}{b-a}\left(y - \frac{b+a}{2}\right).$$

Then

$$\bar{f}(x) = \sum_{k=0}^{K-1} \alpha_k T_k(x).$$

Algorithm 3 *Recursive* INSERT(t, p, a, b, f, maxdepth)

Comment: Construction of a binary tree of subintervals. t denotes the current node, p denotes its parent, a and b are the end points of the interval, f is the function being approximated, $maxdepth$ is the maximal depth of the tree. Subroutine split(a,b,f) returns true if f satisfies the splitting criterion, false otherwise.

```
t.a = a
t.b = b
t.parent = p
t.depth = t.parent.depth + 1
if maxdepth < t.depth then
    maxdepth = t.depth
end if
if split(a,b,f) then
    a1 = a
    b1 = (a + b)/2.0
    a2 = b1
    b2 = b
    Call INSERT(t.left,t,a1,b1,f,maxdepth)
    Call INSERT(t.right,t,a2,b2,f,maxdepth)
end if
End INSERT
```

Here, the Chebyshev coefficients α_k are given by

$$\alpha_0 = \frac{1}{K}\sum_{j=1}^{K} f(i\tau_j), \qquad \alpha_k = \frac{2}{K}\sum_{j=1}^{K} f(i\tau_j)T_k(\tau_j) \quad \text{for } k \geq 1,$$

where the classical Chebyshev nodes τ_j are given by

$$\tau_j = \cos\frac{(2K-2j+1)\pi}{2K} \quad \text{for } j = 1, \ldots, K.$$

We then compute the following quantity

$$S = \frac{|\alpha_{K-1}| + |\alpha_{K-2}|}{\sum_{k=0}^{K-2} |\alpha_k|}.$$

We subdivide the interval $[a, b]$ into two smaller intervals if $S > \delta$ where the constant $\delta \ll 1$ is provided by the user. This splitting mechanism guarantees that $f(iy)$ can be approximated by a low-degree polynomial in each subinterval, since α_{K-1} and α_{K-2} are the coefficients corresponding to the high-frequency components and the magnitude of them are not relatively negligible when $S > \delta$. This splitting criterion is already used in [24] to construct an adaptive mesh in developing a fast adaptive method for stiff two-point boundary value problems.

Remark 3 In practice, we set $K = 20$ and $\delta = 10^{-8}$. This works well for our testing examples.

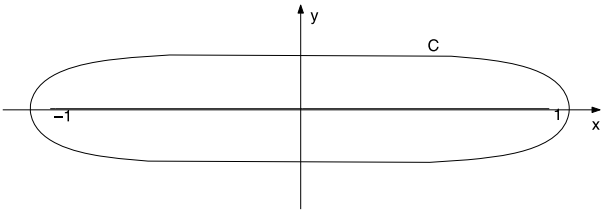


Fig. 1 The curve C within which the near poles lie

Remark 4 The above splitting criterion will sometimes cause excessive refinement around the singular point such as a branch point when the function is too singular. In our actual implementation, we have added a parameter which specifies the maximal number of refinement, or the maximal depth of the tree to prevent such excessive refinement around the singular point.

Remark 5 The use of Chebyshev polynomials is not essential. We may use Legendre polynomials or other orthogonal polynomials as the basis.

3.4 Near Pole Criterion

Suppose that in the iterative procedure we have computed a sum-of-poles approximation $r_i(z) = \sum_{j=1}^{k_i} \frac{w_j}{z-p_j}$ for $f_i(z)$ on the interval $[a_i, b_i]$. We then need to form a new function f_{i-1} by subtracting the near pole contribution $r_i^n(z)$ from f and find a sum-of-poles approximation r_{i-1} for f_{i-1} on some larger interval $[a_{i-1}, b_{i-1}]$. Without loss of generality, we assume that $[a, b] = [-1, 1]$ (the general case can be treated by a simple translation and scaling). We define a near pole as follows. A pole p_j is a near pole with respect to the interval $[-1, 1]$ if it lies within the closed curve C where the curve C is defined by $C = \{z \mid \int_{-1}^{1} \frac{1}{|x-z|^2}\, dx = c, z \in \mathbb{C}\}$. We then have

$$r_i^n(z) = \sum_{p_j \in NP} \frac{w_j}{z - p_j},$$

where $NP = \{p_j \mid p_j \text{ is a near pole}, \ j = 1, \ldots, k_i\}$.

Remark 6 We set $c = 12$ in our numerical experiments. The curve C is very close to an ellipse. For $c = 12$, it is close to the ellipse $E = \{(x, y) \mid \frac{x^2}{a^2} + \frac{y^2}{b^2} = 1, x, y \in \mathbb{R}\}$ where $a = 1.0801$ and $b = 0.2249$. Also, since we carry out the approximation on the imaginary axis, the curve needs to be rotated by 90 degrees (see Fig. 1).

Remark 7 We have chosen the above curve C to define the near poles since any pole function $1/(x - z)$ (here z is the pole location) with the pole located on the curve C has exactly the same L^2 norm with respect to the interval $[-1, 1]$. This is quite natural since we are trying to use the least squares minimization to find sum-of-poles approximations. Presumably, one could replace the curve C by some other simpler ones such as an ellipse.

3.5 Some Issues Related with Algorithm 1

We need a good initial guess for Q (i.e., a good initial guess for the location of the poles) and the number of poles d in order for Algorithm 1 to work robustly. We revise Algorithm 1

Algorithm 4 Revised nonlinear least squares method

Comment: Given $f(iy) \in L^2[a,b]$, return polynomials P, Q and a positive integer d with $\deg(P)+1 = \deg(Q) = d$ such that $\int_a^b |\frac{P(iy)}{Q(iy)} - f(iy)|^2\,dy$ is less than the desired precision ϵ.

1: Set $d = 1$, TOL $= \epsilon \int_a^b |f(iy)|^2\,dy$, error $= 10$TOL.
2: Set the initial guess $Q_0(z) = z - (\frac{b+a}{2} - i\frac{b-a}{5})$.
3: **while** error $\geq$ TOL **do**
4: Apply Algorithm 1 with the initial guess $Q = Q_0$ to find P, Q such that error $= \int_a^b |\frac{P(iy)}{Q(iy)} - f(iy)|^2\,dy$ is minimized, where P, Q are polynomials with $\deg(P) + 1 = \deg(Q) = d$.
5: $d = d + 1$
6: Set the initial guess $Q_0(z) = (z - 2z_{d-1})Q(z)$ where z_{d-1} is the farthest zero of $Q(z)$ to the interval $[ia, ib]$.
7: **end while**

by embedding it in an iterative procedure (see Algorithm 4) so that it can work without a good initial guess for Q and d.

Remark 8 We discretize the integral in each subinterval with the same number of quadrature nodes no matter how large the subinterval is. This is consistent with the assumption that the function after the bootstrap step on any subinterval can be well-approximated by a low-degree polynomial, as discussed in Sect. 3.3.

Remark 9 We need to represent P/Q as a sum of poles to find $Q_0(z)$ and to suit our application. Since we can compute $P(z)$, $Q(z)$, and $Q'(z)$ accurately by the recurrence coefficients resulting from Gram-Schmidt orthogonalization (6), the problem boils down to finding the complex zeros of a given polynomial ($Q(z)$). We use Muller's method [35] to compute the zeros of $Q(z)$. We choose Muller's method because it is very robust for complex root finding and very insensitive to initial guesses.

We now summarize the entire method in Algorithm 5.

4 Numerical Experiments

We have implemented Algorithm 5 in Fortran 95 and some numerical experiments are shown below.

4.1 An *Ad Hoc* Example

Our first numerical example is an *ad hoc* one, namely, we try to approximate

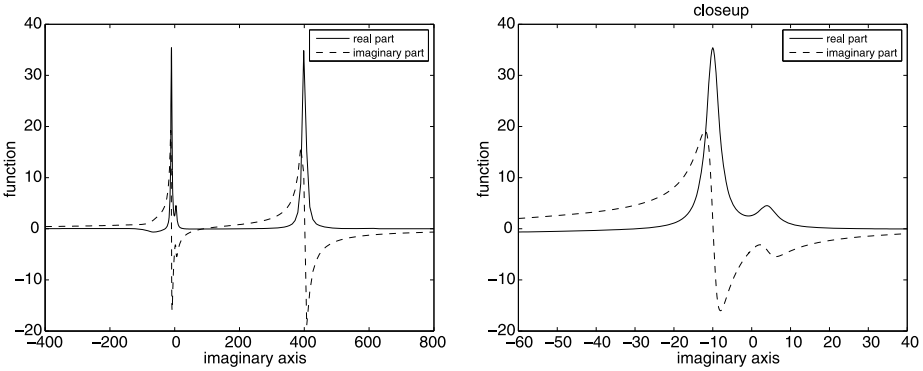
$$f(z) = \frac{71}{z+2+10i} + \frac{12}{z+3-4i} + \frac{230}{z+6-400i} + \frac{-20-10i}{z+30+70i} + \frac{1-2i}{z+200+500i} + \frac{10}{z+1000-4000i}, \tag{7}$$

which is already a sum of six poles. However, we would like to emphasize that in our bootstrap code we have not used any information about f other than the value of the function

Algorithm 5 Bootstrap method for sum-of-poles approximation

Comment: Given a function $f \in H_+^\infty$, find an efficient sum-of-poles approximation $R(z) = \sum \frac{w_k}{z-z_k}$ within the desired precision ϵ.

1: Choose a large interval $[ia_0, ib_0]$ on the imaginary axis and set it as the root interval.
2: Use Algorithm 3 to construct nested subintervals $[ia_j, ib_j]$ $(j = 0, 1, \ldots,$ maxdepth) such that $[ia_j, ib_j] \subset [ia_{j-1}, ib_{j-1}]$ $(j = 1, \ldots,$ maxdepth).
3: Set $f_{\text{maxdepth}} = f(z)$ and $r(z) = 0$.
4: **for** $j =$ maxdepth $: -1 : 1$ **do**
5: Apply Algorithm 4 to obtain a rational approximation $r_j(z)$ for $f_j(z)$ on the interval $[ia_j, ib_j]$.
6: Use near-pole criterion (Sect. 3.4) to extract the near pole part of r_j and denote it by r_j^n.
7: Set $f_{j-1}(z) = f_j(z) - r_j^n(z)$.
8: Set $r(z) = r(z) + r_j^n(z)$.
9: **end for**
10: Apply Algorithm 4 to obtain a rational approximation $r_0(z)$ for $f_0(z)$ on the root interval $[ia_0, ib_0]$.
11: Set $r(z) = r(z) + r_0(z)$. $r(z)$ is an accurate but inefficient rational approximation to $f(z)$.
12: Apply Algorithm 2 to reduce $r(z)$ to a more efficient rational approximation $R(z)$.



(a) (b)

Fig. 2 (**a**) The plot of the original function in the vicinity of two "singular points". (**b**) A closeup of the "singular point" near the origin

on the imaginary axis. We plot the function in Fig. 2, which clearly shows that there are two clusters of peaks and troughs around $-10i$ and $400i$ along the imaginary axis, respectively.

In this example, the computational domain $[A, B]$ along the imaginary axis is chosen $[-10^8 i, 10^8 i]$ and the bound for tree depth *maxdepth* is set to 30. Obviously, the binary tree structure should be built around those two "singular" points for $f(z)$ to be well-resolved. We plot the binary tree structure in the histogram-like graph in Fig. 3, where the x-axis represents the imaginary axis and the height indicates the depth of the subinterval. Figure 3

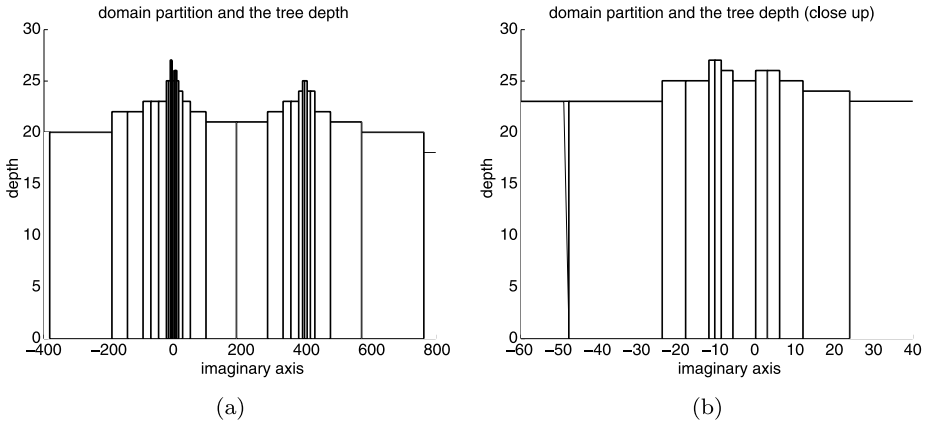


Fig. 3 (**a**) Domain partition and subinterval depths of the tree in the vicinity of two "singular points" which are shown in Fig. 2(a). (**b**) A closeup at the "singular point" near the origin which is shown in Fig. 2(b)

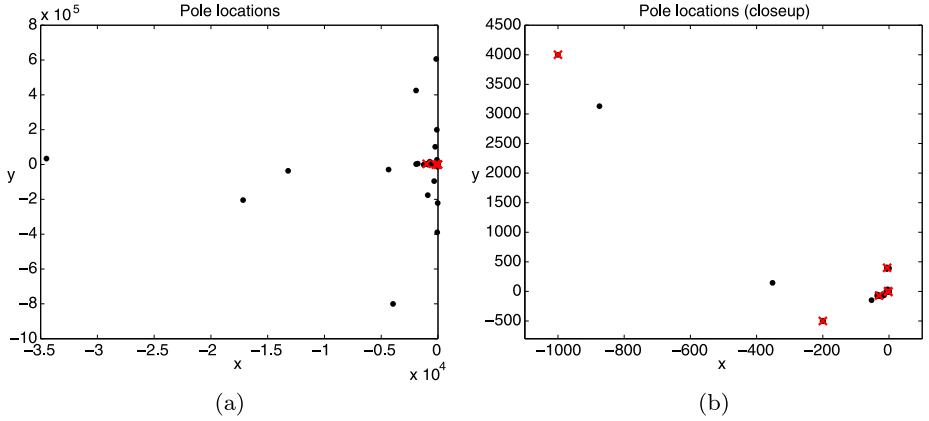


Fig. 4 Pole locations where the *dots* represent poles found in the bootstrap step and the *crosses* represent poles after the balanced truncation step

shows that dense partitions occur around the two "singular" points as expected and the depth of the entire tree is actually 27.

Next we plot the pole locations found by the bootstrap method in Fig. 4, where the dots represent poles found in the bootstrap step and the crosses represent poles after the balanced truncation step.

The bootstrap step found 58 poles in order to achieve the desired precision 10^{-12}. Among these 58 poles, six of them are very close to the original poles in (7) and the rest 52 poles all have very small weights. Setting the reduction error bound as 10^{-8}, the balanced truncation step successively reduced the number of poles to six and moved those six poles to their original positions within the prescribed precision. We list the pole locations and weights found by the bootstrap method, and their absolute errors as compared with the original values in Table 1.

Table 1 Pole locations and weights recovered by Algorithm 5

	Pole location	Absolute error of pole location
Pole 1	−1.9999999999998306 − 9.9999999999997016i	3.432D−13
Pole 2	−3.0000000000013589 + 4.0000000000008784i	5.993D+12
Pole 3	−6.0000000000005373 + 399.99999999999943i	7.822D−13
Pole 4	−30.000000000069846 − 69.999999999846338i	1.688D−10
Pole 5	−200.00000001195605 − 500.00000000509129i	1.299D−8
Pole 6	−999.99999999300542 + 4000.0000000054019i	8.838D−9

	Pole weight	Absolute error of pole weight
Pole 1	70.999999999993960 − 1.40690410710408997D−11i	1.531D−11
Pole 2	12.000000000013918 − 1.41815963269550926D−11i	1.987D−11
Pole 3	230.00000000001975 + 1.59899038454369702D−11i	2.541D−11
Pole 4	−20.000000000134609 − 9.9999999998676810i	1.888D−10
Pole 5	1.0000000001671803 − 2.0000000001507807i	2.251D−10
Pole 6	9.9999999999363958 + 1.19021320577061829D−11i	6.471D−11

4.2 Havriliak-Negami Dielectric Model

The Havriliak-Negami model is the most general material model that arises from considerations of the multi-scale nature of the spatial microstructure of a broad class of dielectrics. When a dielectric material is modeled by the Havriliak-Negami model, the induced polarization P is given by the convolution

$$P(x,t) = \int_0^t \chi(t-\tau)E(x,\tau)\,d\tau,$$

where the susceptibility $\chi(t)$ is defined by the formula

$$\chi(t) = \Delta\epsilon\mathcal{L}^{-1}\big(f(s\tau_r)\big)$$

with the function f defined by the formula

$$f(z) = \frac{1}{(1+z^\alpha)^\beta} \tag{8}$$

and τ_r the central relaxation time of the material. Note that the well-known Debye model is obtained by setting $\alpha = \beta = 1$, while the Cole-Cole and Cole-Davidson models are obtained by setting $\beta = 1$ and $\alpha = 1$, respectively. We have used our bootstrap method to find the sum-of-poles approximation to the function f defined in (8). We present three cases below. For all these cases the computational domain is chosen $[-10^8 i, 10^8 i]$ and the desired precision for bootstrap step and the error bound for reduction are set 10^{-12} and 10^{-8} respectively.

4.2.1 Cole-Cole Kernel

We set $\alpha = 0.6$ and $\beta = 1$, which is a special case of the Cole-Cole model. This function has a branch point at the origin and its graph is plotted in Fig. 5. The splitting procedure creates

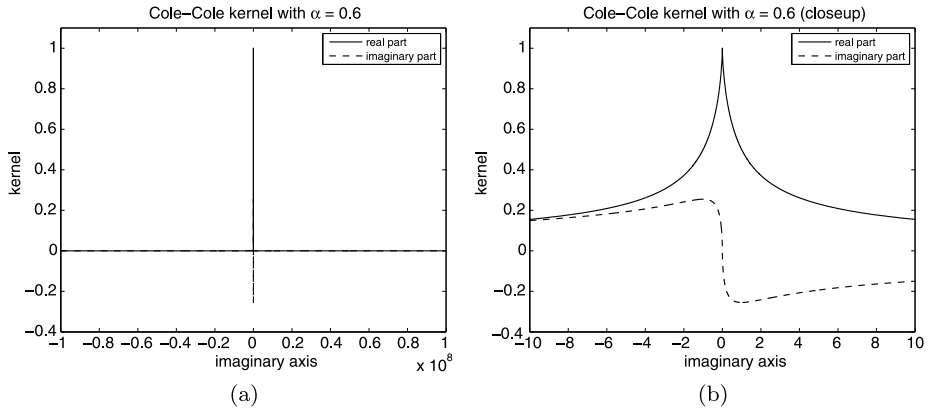


Fig. 5 (**a**) The real and imaginary parts of the Cole-Cole kernel with $\alpha = 0.6$. (**b**) A closeup around the "singular point" at the origin

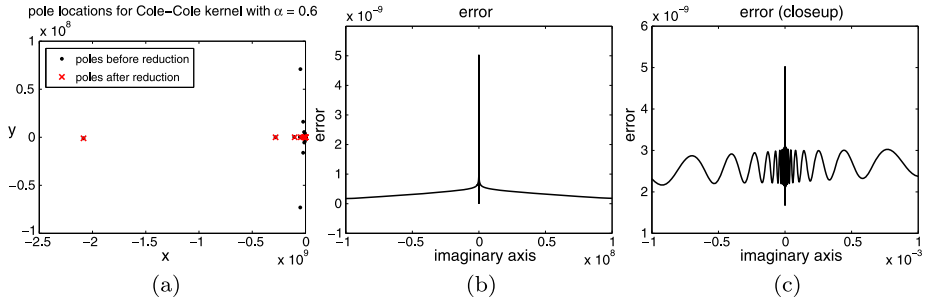


Fig. 6 (**a**) The pole locations of the Cole-Cole kernel with $\alpha = 0.6$. (**b**) Error of the sum-of-pole representation. (**c**) A closeup of the error curve near the origin

a tree structure with the depth of 50 to resolve the "singular point". As shown in Fig. 6(a), a total of 268 poles are located by the algorithm before the reduction is implemented and the maximum error 3.803×10^{-12} occurs near the origin. As the reduction error bound is set to 10^{-8}, we are left with 72 poles yielding an approximation of the original Cole-Cole kernel with the maximum absolute error about 5.016×10^{-9}. The error curve is plotted in Figs. 6(b) and 6(c).

4.2.2 Cole-Davidson Kernel

We set $\alpha = 1$ and $\beta = 0.7$. The corresponding Cole-Davidson kernel is plotted in Fig. 7. For this example, a tree with depth of 22 is constructed in partitioning the computational domain $[-10^8 i, 10^8 i]$. As shown in Fig. 8(a), a total of 145 poles are located by the algorithm without the reduction and the maximum absolute error is approximately 2.373×10^{-13} in $[-10^8 i, 10^8 i]$. After reduction with the reduction error bound set as 10^{-8}, 31 poles can give an approximation of the original Cole-Davidson kernel with the maximum absolute error about 9.832×10^{-9}. The error along the imaginary axis is shown in Figs. 8(b) and 8(c).

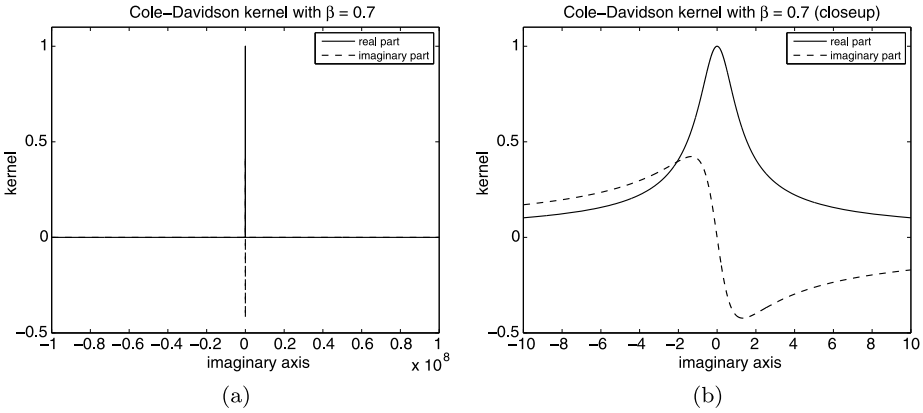


Fig. 7 (**a**) The real and imaginary parts of the Cole-Davidson kernel with $\beta = 0.7$. (**b**) A closeup around the "singular point" at the origin

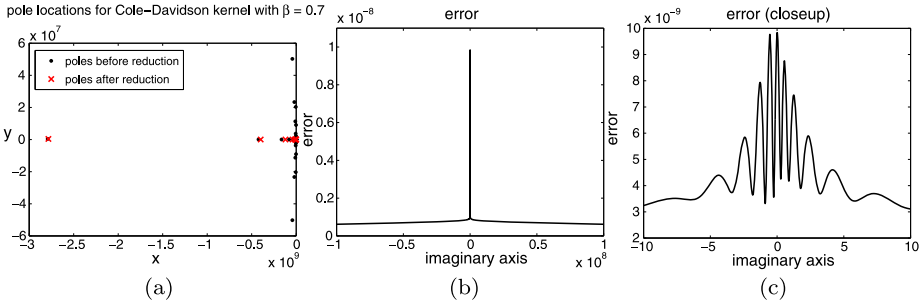


Fig. 8 (**a**) The pole locations of the Cole-Davidson kernel with $\beta = 0.7$. (**b**) Error of the sum-of-pole representation. (**c**) A closeup of the error curve near the origin

4.2.3 A General Havriliak-Negami Kernel

We choose $\alpha = 0.85$ and $\beta = 0.5$, the corresponding Havriliak-Negami kernel is plotted in Fig. 9. The domain $[-10^8 i, 10^8 i]$ is partitioned using a 45-level tree. As shown in Fig. 10(a), a total of 244 poles are located by the algorithm before the reduction step giving the maximum absolute error along the whole imaginary axis no larger than 1.16×10^{-10} and when the reduction error bound is set 10^{-8}, 63 poles are left to approximate the original Havriliak-Negamin kernel with a maximum absolute error about 8.359×10^{-9}. The locations and weights of these 63 poles are listed in Table 2. The error along the imaginary axis is shown in Figs. 10(b) and 10(c).

4.3 Schrödinger Kernels

The initial-value problem for the Schrödinger equation in two and three dimensions unbounded domains is as follow:

$$\begin{cases} iu_t(x,t) = \Delta u(x,t) + V(x,t)u(x,t), \quad x \in \mathbb{R}^n \ (n = 2, 3), \ t > 0, \\ u(x,0) = u_0(x). \end{cases}$$

Table 2 Pole locations and weights for the general Havriliak-Negami kernel with $\alpha = 0.85$ and $\beta = 0.5$

Pole location	Pole weight
−1.6261696972532313D−08 − 1.8999256951789423D−11i	6.6897067289762098D−16 + 1.3880111508598038D−18i
−1.1300889068780398D−07 − 1.1743784163462279D−10i	1.6629695258928394D−14 + 3.1199928216274363D−17i
−4.8864261594253980D−07 − 4.8165604974655016D−10i	2.0227144201858987D−13 + 3.6299623936237331D−16i
−1.6700138175527535D−06 − 1.5945780951456406D−09i	1.6981830793549590D−12 + 2.9622242797074512D−15i
−4.9261016585204196D−06 − 4.5949140638535438D−09i	1.1235668529125787D−11 + 1.9172680507792585D−14i
−1.3103076398079006D−05 − 1.1984186164005556D−08i	6.2712275421445702D−11 + 1.0498730190503655D−13i
−3.2243418668661069D−05 − 2.8976232096915848D−08i	3.0751739175283097D−10 + 5.0593154224145867D−13i
−7.4621357063370884D−05 − 6.5972042346866275D−08i	1.3604489190060119D−09 + 2.2020105040397959D−12i
−1.6427440760095413D−04 − 1.4300137241618604D−07i	5.5315240555666790D−09 + 8.8166350643359232D−12i
−3.4683929123846449D−04 − 2.9751367877438528D−07i	2.0954120899751043D−08 + 3.2923204791108645D−11i
−7.0667110806289140D−04 − 5.9799873924975297D−07i	7.4727993470186633D−08 + 1.1599295440196937D−10i
−1.3960758158578317D−03 − 1.1684476781905981D−06i	2.5298286752467625D−07 + 3.8966704457564965D−10i
−2.6843691508191248D−03 − 2.2296024991304160D−06i	8.1862837407861228D−07 + 1.2564690820989025D−09i
−5.0389241914238631D−03 − 4.1659568894186327D−06i	2.5473766223713008D−06 + 3.9128476855840082D−09i
−9.2571140266795963D−03 − 7.6459123566934399D−06i	7.6659841297897001D−06 + 1.1863738245257384D−08i
−1.6678147028053463D−02 − 1.3833059911250092D−05i	2.2439724511428661D−05 + 3.5334708633653130D−08i
−2.9518537038991904D−02 − 2.4761435262298923D−05i	6.4306610138417278D−05 + 1.0447205980645325D−07i
−5.1395825331164713D−02 − 4.4022684048374245D−05i	1.8187094330244155D−04 + 3.1080252968294463D−07i
−8.8131588157783783D−02 − 7.8072317295714138D−05i	5.1311693059247405D−04 + 9.4783238621681345D−07i
−1.4894341600686228D−01 − 1.3875903804765939D−04i	1.4664117051281927D−03 + 3.0382339685143450D−06i
−2.4806504361279491D−01 − 2.4781171716649629D−04i	4.3383526519226419D−03 + 1.0548242864146138D−05i
−4.0623115638930218D−01 − 4.4205811627483704D−04i	1.3617490851647717D−02 + 4.0380548269918381D−05i
−6.4855626386495790D−01 − 7.6478791866562486D−04i	4.4394886299826915D−02 + 1.5240981957156681D−04i
−9.9665982576319812D−01 − 1.2818916821852489D−03i	1.1861736336526271D−01 + 3.4523109251534515D−04i

Table 2 (*Continued*)

Pole location	Pole weight
−1.508632095093221 0D+00 − 2.394104856365115 3D−03i	2.011793552524037 7D−01 + 4.232052408197917 5D−04i
−2.316606481552831 1D+00 − 4.783335570151987 8D−03i	2.631576975882655 7D−01 + 5.869046815231085 3D−04i
−3.593447256040878 6D+00 − 9.735192553361908 5D−03i	3.261688140641405 6D−01 + 9.586037600606297 1D−04i
−5.590738146133136 7D+00 − 1.982082602047188 1D−02i	4.040673286721996 9D−01 + 1.551701580728840 4D−03i
−8.701145988373181 0D+00 − 3.913821400084323 3D−02i	5.074678841516746 5D−01 + 2.263367153800226 1D−03i
−1.364158869789625 0D+01 − 7.293360344289794 0D−02i	6.682031360038186 4D−01 + 3.010990243942155 0D−03i
−1.613368618555301 4D+01 + 1.068575364961880 7D+01i	−3.622223528594973 3D−05 − 5.845400867498289 3D−04i
−1.617802848950175 9D+01 − 1.091366504227084 3D+01i	−1.044694150190389 4D−04 + 6.174335825273936 4D−04i
−2.182263194132270 8D+01 − 1.264290830609198 8D−01i	8.759380183705375 3D−01 + 2.366484983651256 7D−03i
−3.435940727285341 0D+01 − 1.674858771861442 7D−01i	1.088277559427035 7D+00 − 2.809391852060709 3D−04i
−5.483538280205041 1D+01 − 1.884591964503684 6D−01i	1.526470682552531 2D+00 − 1.477476715835990 9D−03i
−9.073345003375762 0D+01 − 1.988229359687569 7D−01i	2.157145612424999 5D+00 − 1.696629118273704 3D−03i
−1.535092073368840 9D+02 − 2.044731024282355 5D−01i	2.994263461521105 9D+00 − 1.573384149558077 0D−03i
−2.624653653010987 0D+02 − 2.060823904905801 4D−01i	4.122699847986389 4D+00 − 1.415304765086084 4D−03i
−4.512181759232693 7D+02 − 2.010347759848941 4D−01i	5.667380081233885 0D+00 − 1.310207928339071 9D−03i
−7.786752632115423 0D+02 − 1.830366940536708 5D−01i	7.800107729160857 6D+00 − 1.293641351073995 1D−03i
−1.348496932547687 0D+03 − 1.390996167897527 2D−01i	1.076043457527893 2D+01 − 1.400437750079002 0D−03i
−2.343970996711703 3D+03 − 4.280602317383019 9D−02i	1.488643239672481 4D+01 − 1.684441821503914 4D−03i
−4.090919465038133 7D+03 + 1.597266048054397 9D−01i	2.065889447333893 3D+01 − 2.235796810401913 9D−03i
−7.171909183768930 2D+03 + 5.792012469204592 7D−01i	2.876552582817158 3D+01 − 3.207534315061835 7D−03i
−1.263503122652328 4D+04 + 1.445475927987625 0D+00i	4.019502130368813 7D+01 − 4.861202348052762 2D−03i
−2.237831241320992 6D+04 + 3.241567133542516 0D+00i	5.637591838815568 2D+01 − 7.646603328106615 7D−03i
−3.986288451337161 7D+04 + 6.997731445100994 2D+00i	7.938282535148854 9D+01 − 1.235755386757882 7D−02i
−7.144662202022802 4D+04 + 1.495238848910227 D+01i	1.122445343380214 5D+02 − 2.042268215184564 5D−02i

Table 2 (*Continued*)

Pole location	Pole weight
−1.2889888810532542D+05 + 3.2070883040622348D+01i	1.5940710397380926D+02 − 3.4468446948481724D−02i
−2.3418379716940189D+05 + 6.9632241404067031D+01i	2.2743404049757117D+02 − 5.9458350912149120D−02i
−4.2864115815454989D+05 + 1.5395585563404657D+02i	3.2607132638105196D+02 − 1.0503420990791842D−01i
−7.9077484694106120D+05 + 3.4830644138568164D+02i	4.6987721587318026D+02 − 1.9046334754858499D−01i
−1.4710562794020737D+06 + 8.0964922824999849D+02i	6.8073279163653638D+02 − 3.5522659070703444D−01i
−2.7607182703318312D+06 + 1.9402712762749891D+03i	9.9174736524809589D+02 − 6.8231126076493864D−01i
−5.2292938561918428D+06 + 4.8069084236547669D+03i	1.4534818781469660D+03 − 1.3519440316568112D+00i
−1.0004010389507411D+07 + 1.2354907336550530D+04i	2.1447275106461311D+03 − 2.7815363296935689D+00i
−1.9358301668143917D+07 + 3.3353700312647830D+04i	3.1972948003635311D+03 − 6.1634063642689991D+00i
−3.8130806644445576D+07 + 1.0148578669153416D+05i	4.9082221597574644D+03 − 1.8059203309233254D+01i
−8.1151529179336846D+07 + 7.6590891482274735D+05i	1.0428539709965567D+04 − 4.5405313294697277D+02i
−1.2571099094916432D+08 − 2.2373715784476224D+07i	3.3486487276984317D+03 − 7.7529302508448054D+03i
−1.2654796676183879D+08 + 2.0133554815602448D+07i	4.7521646933361490D+03 + 8.2435732619744540D+03i
−4.1056275254191780D+08 − 5.1492945687695428D+04i	3.7810934453175905D+04 − 7.4343541700874400D+00i
−4.1323131510750036D+09 + 4.6987832689540635D+05i	4.6500398527705751D+05 − 4.2110626579124187D+01i

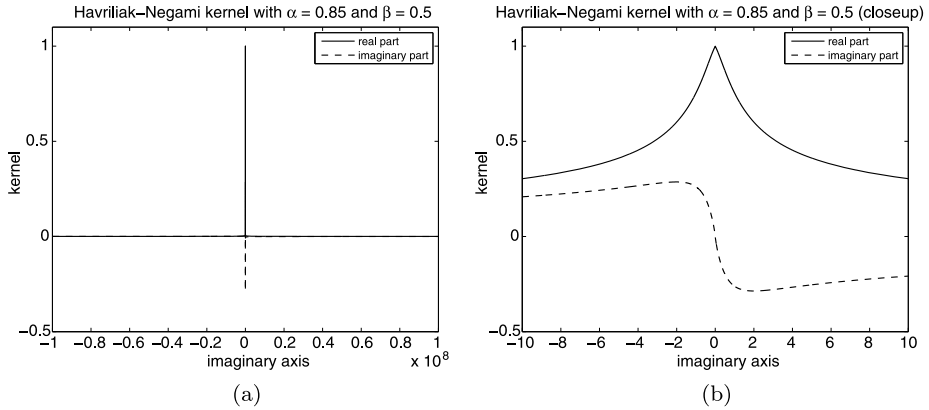


Fig. 9 (**a**) The real and imaginary parts of the Havriliak-Negami kernel with $\alpha = 0.85$ and $\beta = 0.5$. (**b**) A closeup around the "singular point" at the origin

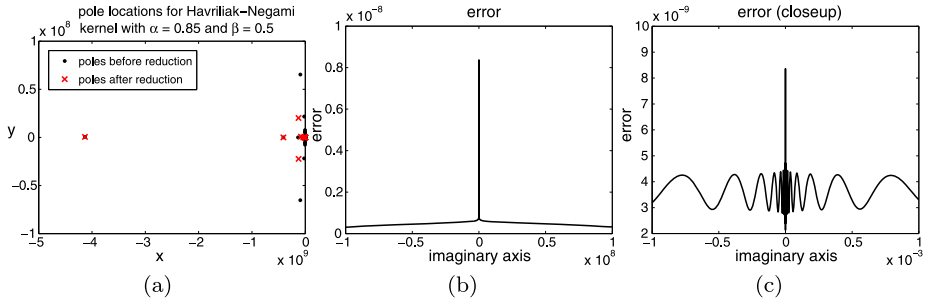


Fig. 10 (**a**) The pole locations of the Havriliak-Negami kernel with $\alpha = 0.85$ and $\beta = 0.5$. (**b**) Error of the sum-of-pole representation. (**c**) A closeup of the error curve near the origin

If we represent the solution of the above problem by a Fourier series in two dimensions and a spherical harmonic series in three dimensions, then the nonreflecting boundary condition (for details please see, for example, [21, 22]) for each mode contains a convolution integral whose kernel is given by

$$K(t) = \mathcal{L}^{-1}\left[\frac{\sqrt{is}K_\nu'(\sqrt{is})}{(s - s_\nu)K_\nu(\sqrt{is})}\right](t),$$

where K_ν is the modified Bessel function [1] and ν of integer values and ν of half integer values correspond to two dimensions and three dimensions, respectively. The exact value of s_ν is not essential and will not affect the performance of the nonreflecting boundary conditions, provided that it does not coincide with the zeros of $K_\nu(\sqrt{is})$ and does not introduce new "singular" point rather than the existing turning point $i\nu^2$. In our numerical experiments, we have used the following value of s_ν

$$s_\nu = i\nu^2 - \nu^{\frac{4}{3}}.$$

As discussed in the introduction, in order to be able to implement the nonreflecting boundary conditions for the Schrödinger equations efficiently and accurately, one needs to

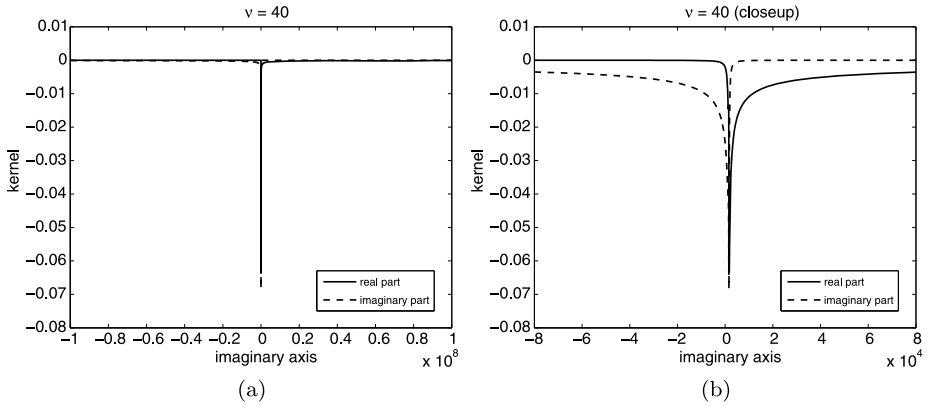


Fig. 11 (**a**) The real and imaginary parts of the kernel $f_{40}(s)$ are plotted respectively. (**b**) A closeup at the "singular" point

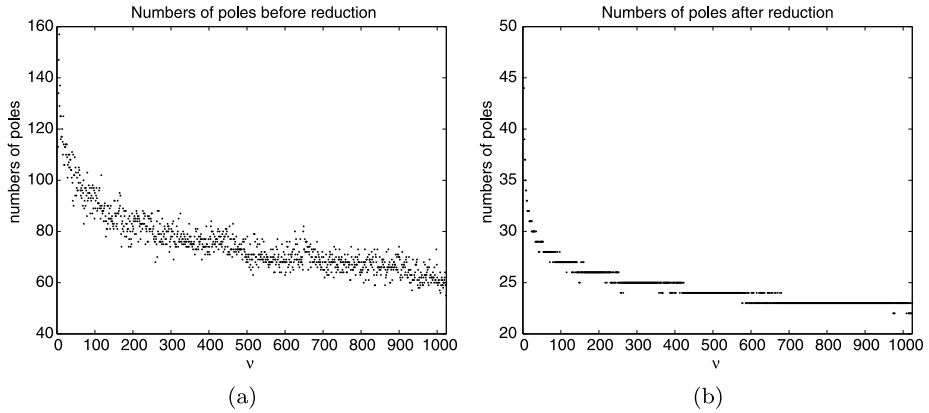


Fig. 12 Numbers of poles for approximating $f_\nu(s)$ on the interval $[-10^8 i, 10^8 i]$ with the error $\epsilon = 10^{-9}$. (**a**) Numbers of poles for $\nu = 1, 2, \ldots, 1024$ before pole reduction. (**b**) Numbers of poles for $\nu = 1, 2, \ldots, 1024$ after reduction using the balanced truncation method of Algorithm 2

find an efficient and accurate sum-of-poles approximation for the following functions

$$f_\nu(s) = \frac{\sqrt{is}K'_\nu(\sqrt{is})}{(s - s_\nu)K_\nu(\sqrt{is})}, \qquad \nu = \begin{cases} 0, 1, \ldots, 1024 & \text{for 2D} \\ \frac{1}{2}, 1 + \frac{1}{2}, \ldots, 1024 + \frac{1}{2} & \text{for 3D.} \end{cases}$$

These functions $f_\nu(s)$ have similar graphs on the imaginary axis and the graph of f_{40} on the imaginary axis is shown in Fig. 11.

We have used our bootstrap method to find the sum-of-poles approximations for these functions. Since these functions decay rather slowly (like $1/\sqrt{s}$), we set the root interval to a very large interval, i.e., $[-10^8 i, 10^8 i]$. The desired precision for the bootstrap step and the error bound for reduction are 10^{-12} and 10^{-9}, respectively. The maximal depth of the binary tree is set to 27 for all ν. Figure 12 plots the number of poles versus the kernel index ν before and after the pole reduction using the balanced truncation method of Algorithm 2. In summary, the average number of poles needed to achieve 9-digit accuracy for all kernels

Table 3 Pole locations and weights for $\nu = 40$

Pole location	Pole weight
−1.3679806100099026D+02 + 1.5999990386504376D+03i	−3.1849616557713867D−01 − 7.169846038239819 0D+00i
−3.8446755962630084D+02 + 1.3169809502189084D+03i	−3.9150782519657925D+00 − 6.2952078325734986D+00i
−5.6428136502340499D+02 + 1.1357455584334505D+03i	−2.3113843565381416D+00 − 1.0948559612462301D+00i
−9.5585667999565669D+02 + 1.1602072986967814D+03i	−5.6179935085539432D+00 − 3.0974169617535630D+00i
−1.7574920112533832D+03 + 1.1567903606233506D+03i	−6.9228247905308820D+00 − 5.2558213606601027D+00i
−3.2557811321442528D+03 + 1.1472119663653730D+03i	−8.8150867488287812D+00 − 7.6659822135545665D+00i
−6.0698384248048515D+03 + 1.1354788044793968D+03i	−1.1619261503921365D+01 − 1.0796473887385295D+01i
−1.1393185918857180D+04 + 1.1223915580994021D+03i	−1.5691042769964152D+01 − 1.5089435373087708D+01i
−2.1547888270464267D+04 + 1.1073038789419918D+03i	−2.1542757915010366D+01 − 2.1100187383702856D+01i
−4.1101074946101922D+04 + 1.0880427831104066D+03i	−2.9933919176431957D+01 − 2.9610981849295339D+01i
−7.9142456464328105D+04 + 1.0593906649773116D+03i	−4.1994536822572165D+01 − 4.1765902316715824D+01i
−1.5399513855957030D+05 + 1.0088487068166596D+03i	−5.9414872873524779D+01 − 5.9266113226215062D+01i
−3.0310605513654469D+05 + 9.0544870734455640D+02i	−8.4742190799772914D+01 − 8.4670255658649083D+01i
−6.0414394408523501D+05 + 6.6896942434458185D+02i	−1.2184853139895576D+02 − 1.2186686670456919D+02i
−1.2207671048801406D+06 + 7.9717799525686075D+01i	−1.7667263681860186D+02 − 1.7682605247346936D+02i
−2.5036495053363717D+06 − 1.5066843222518185D+03i	−2.5839749253571256D+02 − 2.5880219386962762D+02i
−5.2173390288080219D+06 − 6.1397067280630727D+03i	−3.8129084150107445D+02 − 3.8224549021827539D+02i
−1.1057097850383533D+07 − 2.0942155042915379D+04i	−5.6740066191373694D+02 − 5.6970858762503258D+02i
−2.3831792672948387D+07 − 7.2909604404998143D+04i	−8.4956547243372938D+02 − 8.5549852807102786D+02i
−5.2099200386209391D+07 − 2.7135128767604678D+05i	−1.2703616229464419D+03 − 1.2864456917163832D+03i
−1.1449768316870768D+08 − 1.0520492150020560D+06i	−1.8729600957004234D+03 − 1.9148297253953972D+03i
−2.5468596260458374D+08 − 3.7025324363216721D+06i	−2.9791099302296293D+03 − 3.0565398404843359D+03i
−6.9997568941479504D+08 − 1.2154758238207720D+07i	−7.6454464257735144D+03 − 7.7942521703595030D+03i
−6.1049966832967319D+09 − 1.0992207269816059D+08i	−6.9386301848368865D+04 − 7.0656682349776151D+04i

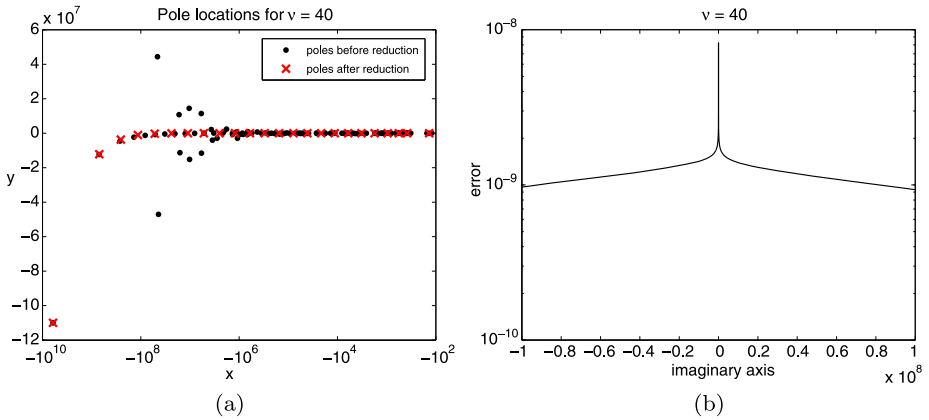


Fig. 13 (**a**) The pole locations for approximating the kernel $f_{40}(s)$ before and after reduction and a logarithmic scale (base 10) is used for the X-axis. (**b**) The error when approximates the kernel $f_{40}(s)$ with 24 poles and logarithmic scale (base 10) is used for the Y-axis

is 25 with the standard deviation roughly equal to 2 after the balanced truncation reduction, while they are 76 and 12 respectively before the reduction. The pole locations generally follow a nice trajectory. Figure 13(a) shows the locations of the 109 poles obtained by bootstrap step and the 24 poles after reduction for approximating the kernel $f_{40}(s)$. The exact locations and weights of the 24 poles obtained after reduction are listed in Table 3. Finally, Fig. 13(b) shows the absolute error of the sum-of-poles approximation for f_{40} on the imaginary axis. We observe that the largest error occurs around the turning point $i\nu^2$, where the function changes most rapidly. We would like to remark here that these figures are fairly representative for this class of functions.

5 Conclusions

We have presented a bootstrap method for finding efficient sum-of-poles approximations for causal functions. Our method is based on the nonlinear least squares method in [5]. Due to its bootstrapping nature, this method can handle such approximation problems on a much larger interval and those with multiple "singular" points. It is believed that the algorithm reported in this paper is more robust since it removes the dependence of a good initial guess on pole locations. Our method also generates very efficient sum-of-poles approximations since we have applied the balanced truncation method to reduce the number of poles in the final optimization step. We expect that the method can be applied to obtain efficient sum-of-poles approximations for a broad class of causal functions in various areas, including control theory, computational electromagnetics, and nonreflecting boundary conditions for other PDEs.

Acknowledgements S. Jiang was supported in part by National Science Foundation under grant CCF-0905395 and would like to thank Dr. Bradley Alpert at National Institute of Standards and Technology for many useful discussions on this project. Both authors would like to thank the anonymous referees for their careful reading and very useful suggestions which have greatly enhanced the presentation of the work.

References

1. Abramowitz, M., Stegun, I.: Handbook of Mathematical Functions. Dover, New York (1965)
2. Adamyan, V.M., Arov, D.Z., Krein, M.G.: Infinite Hankel matrices and generalized Carathéodory-Fejér and I. Schur problems. Funct. Anal. Appl. **2**, 269–281 (1968)
3. Adamyan, V.M., Arov, D.Z., Krein, M.G.: Infinite Hankel matrices and generalized problems of Carathéodory-Fejér and Riesz problems. Funct. Anal. Appl. **2**(1), 1–18 (1968)
4. Adamyan, V.M., Arov, D.Z., Krein, M.G.: Analytic properties of the Schmidt pairs of a Hankel operator and the generalized Schur-Takagi problem. Mat. Sb. **86**, 34–75 (1971)
5. Alpert, B., Greengard, L., Hagstrom, T.: Rapid evaluation of nonreflecting boundary kernels for time-domain wave propagation. SIAM J. Numer. Anal. **37**, 1138–1164 (2000)
6. Alpert, B., Greengard, L., Hagstrom, T.: Nonreflecting boundary conditions for the time-dependent wave equation. J. Comput. Phys. **180**, 270–296 (2002)
7. Antoine, X., Arnold, A., Besse, C., Ehrhardt, M., Schädle, A.: A review of transparent and artificial boundary conditions techniques for linear and nonlinear Schrödinger equations. Commun. Comput. Phys. **4**, 729–796 (2008)
8. Beylkin, G., Monzón, L.: On approximation of functions by exponential sums. Appl. Comput. Harmon. Anal. **19**, 17–48 (2005)
9. Beylkin, G., Monzón, L.: On generalized Gaussian quadratures for exponentials and their applications. Appl. Comput. Harmon. Anal. **12**, 332–373 (2002)
10. Bremer, J., Gimbutas, Z., Rokhlin, V.: A nonlinear optimization procedure for generalized Gaussian quadratures. SIAM J. Sci. Comput. **32**, 1761–1788 (2010)
11. Causley, M., Petropolous, P., Jiang, S.: Incorporating the Havriliak-Negami dielectric model in numerical solutions of the time-domain Maxwell equations. J. Comput. Phys. **230**, 3884–3899 (2011)
12. Dym, H., McKean, H.P.: Fourier Series and Integrals. Academic Press, San Diego (1972)
13. Glover, K.: All optimal Hankel-norm approximations of linear multivariable systems and their L^∞-error bounds. Int. J. Control **39**, 1115–1193 (1984)
14. Gutknecht, M.H., Trefethen, L.N.: Real and complex Chebyshev approximation on the unit disk and interval. Bull., New Ser., Am. Math. Soc. **8**, 455–458 (1983)
15. Gutknecht, M.H., Smith, J.O., Trefethen, L.N.: The Carathéodory-Fejér (CF) method for recursive digital filter design. IEEE Trans. Acoust. Speech Signal Process. **31**, 1417–1426 (1983)
16. Gutknecht, M.H.: Rational Carathéodory-Fejér approximation on a disk, a circle, and an interval. J. Approx. Theory **41**, 257–278 (1984)
17. Hagstrom, T.: Radiation boundary conditions for the numerical simulation of waves. Acta Numer. **8**, 47–106 (1999)
18. Hammarling, S.: Numerical solution of the stable, non-negative definite Lyapunov equation. IMA J. Numer. Anal. **2**, 303–323 (1982)
19. Hardy, G.H.: On the mean value of the modulus of an analytic function. Proc. Lond. Math. Soc. **s2_14**, 269–277 (1915)
20. Jackson, J.D.: Classical Electrodynamics. Wiley, New York (1998)
21. Jiang, S.: Fast evaluation of the nonreflecting boundary conditions for the Schrödinger equation. Ph.D. thesis, Courant Institute of Mathematical Sciences, New York University, New York (2001)
22. Jiang, S., Greengard, L.: Efficient representation of nonreflecting boundary conditions for the time-dependent Schrödinger equation in two dimensions. Commun. Pure Appl. Math. **61**, 261–288 (2008)
23. Laub, A., Heath, M., Paige, C., Ward, R.: Computation of system balancing transformations and other applications of simultaneous diagonalization algorithms. IEEE Trans. Autom. Control **32**, 115–122 (1987)
24. Lee, J., Greengard, L.: A fast adaptive numerical method for stiff two-point boundary value problems. SIAM J. Sci. Comput. **18**, 403–429 (1997)
25. Li, J.R.: Low order approximation of the spherical nonreflecting boundary kernel for the wave equation. Linear Algebra Appl. **415**, 455–468 (2006)
26. Li, J.R.: A fast time stepping method for evaluating fractional integrals. SIAM J. Sci. Comput. **31**, 4696–4714 (2010)
27. Lin, L., Lu, J., Ying, L., E, W.: Pole-based approximation of the Fermi-Dirac function. Chin. Ann. Math., Ser. B **30**, 729–742 (2009)
28. López-Fernández, M., Palencia, C.: On the numerical inversion of the Laplace transform of certain holomorphic mappings. Appl. Numer. Math. **51**, 289–303 (2004)
29. López-Fernández, M., Palencia, C., Schädle, A.: A spectral order method for inverting sectorial Laplace transforms. SIAM J. Numer. Anal. **44**, 1332–1350 (2006)
30. López-Fernández, M., Lubich, C., Schädle, A.: Adaptive, fast, and oblivious convolution in evolution equations with memory. SIAM J. Sci. Comput. **30**, 1015–1037 (2008)
31. Lubich, C.: Convolution quadrature revisited. BIT Numer. Math. **44**, 503–514 (2004)

32. Lubich, C., Schädle, A.: Fast convolution for nonreflecting boundary conditions. SIAM J. Sci. Comput. **24**, 161–182 (2002)
33. Ma, J., Rokhlin, V., Wandzura, S.: Generalized Gaussian quadrature rules for systems of arbitrary functions. SIAM J. Numer. Anal. **33**, 971–996 (1996)
34. Moore, B.: Principal component analysis in linear systems: controllability, observability, and model reduction. IEEE Trans. Autom. Control **26**, 17–32 (1981)
35. Muller, D.: A method for solving algebraic equations using an automatic computer. Math. Tables Other Aids Comput. **10**, 208–215 (1956)
36. Peller, V.V.: Hankel Operators and Their Applications. Springer Monographs in Mathematics. Springer, New York (2003)
37. Penzl, T.: Algorithms for model reduction of large dynamical systems. Linear Algebra Appl. **415**, 322–343 (2006)
38. Penzl, T.: Numerical solution of generalized Lyapunov equations. Adv. Comput. Math. **8**, 33–48 (1998)
39. Powell, M.J.D.: Approximation Theory and Methods. Cambridge University Press, Cambridge (1981)
40. Rudin, W.: Real and Complex Analysis. McGraw-Hill, New York (1987)
41. Safonov, M.G., Chiang, R.Y.: A Schur method for balanced-truncation model reduction. IEEE Trans. Autom. Control **34**, 729–733 (1989)
42. Schädle, A., López-Fernández, M., Lubich, C.: Fast and oblivious convolution quadrature. SIAM J. Sci. Comput. **28**, 421–438 (2006)
43. Sontag, E.D.: Mathematical Control Theory: Deterministic Finite Dimensional Systems. Springer, New York (1998)
44. Trefethen, L.N.: Rational Chebyshev approximation on the unit disk. Numer. Math. **37**, 297–320 (1981)
45. Trefethen, L.N.: Chebyshev approximation on the unit disk. In: Werner, K.E., Wuytack, L., Ng, E. (eds.) Computational Aspects of Complex Analysis, pp. 309–323. D. Reidel Publishing, Dordrecht (1983)
46. Trefethen, L.N., Gutknecht, M.H.: The Carathéodory-Fejér method for real rational approximation. SIAM J. Numer. Anal. **20**, 420–436 (1983)
47. Trefethen, L.N., Weideman, J., Schmelzer, T.: Talbot quadratures and rational approximations. BIT Numer. Math. **46**, 653–670 (2006)
48. Yarvin, N., Rokhlin, V.: Generalized Gaussian quadratures and singular value decompositions of integral operators. SIAM J. Sci. Comput. **20**, 699–718 (1998)