Mathematical Modelling and Analysis
Volume 17 Number 5, November 2012, 696–714
http://dx.doi.org/10.3846/13926292.2012.736089
© Vilnius Gediminas Technical University, 2012

Publisher: Taylor&Francis and VGTU
http://www.tandfonline.com/TMMA
Print ISSN: 1392-6292
Online ISSN: 1648-3510

# Product Quasi-Interpolation in Logarithmically Singular Integral Equations*

**Eero Vainikko**[a] **and Gennadi Vainikko**[a,b]

[a] *University of Tartu, Faculty of Mathematics and Computer Science*
Liivi 2, 50409 Tartu, Estonia
[b] *Estonian Academy of Sciences*
Kohtu 6, 10130 Tallinn, Estonia
E-mail(*corresp.*): eero.vainikko@ut.ee
E-mail: gennadi.vainikko@ut.ee

Received February 20, 2012; revised August 18, 2012; published online November 1, 2012

**Abstract.** A discrete high order method is constructed and justified for a class of Fredholm integral equations of the second kind with kernels that may have boundary and logarithmic diagonal singularities. The method is based on the improving the boundary behaviour of the kernel with the help of a change of variables, and on the product integration using quasi-interpolation by smooth splines of order $m$. Properties of different proposed calculation schemes are compared through numerical experiments using, in particular, variable precision interval arithmetics.

**Keywords:** weakly singular integral equations, boundary singularities, spline quasi-interpolation, product integration, Nyström-type methods.

**AMS Subject Classification:** 65R20; 65D07; 65D30.

## 1 Introduction

In the present paper we treat a fully discrete method of accuracy $O(h^m)$ (see Section 4) for the integral equation

$$u(x) = \int_0^1 \big[a(x,y)\log|x-y| + b(x,y)\big]u(y)\,dy + f(x), \quad 0 \le x \le 1, \qquad (1.1)$$

with the logarithmic diagonal singularity in the kernel. The coefficient functions $a, b \in C^m([0,1]\times(0,1))$ and the free term $f \in C[0,1]\cap C^m(0,1)$ may have certain boundary singularities described below in detail. Due to diagonal and boundary singularities of the kernel, the derivatives of the solution to equation (1.1), as a rule, have certain boundary singularities. In Section 2

* The work was supported by the European Regional Development Fund through the Estonian Centre of Excellence in Computer Science and by the Estonian Science Foundation, grants 9019 and 9104.

we reduce (1.1) with the help of a smoothing change of variables to a similar problem

$$v(t) = \int_0^1 \big(\mathcal{A}(t,s)\log|t-s| + \mathcal{B}(t,s)\big)v(s)\,ds + g(t), \quad 0 \le t \le 1,$$

in which the coefficients $\mathcal{A}(t,s)$ and $\mathcal{B}(t,s)$ have no singularities and vanish for $s=0$ and $s=1$. Simultaneously the singularities of the solution will be milder or disappear at all for suitable parameters of the change of variables. On the contrast, the logarithmic diagonal singularity of the kernel of equation (1.1) still remains to be present. In Section 3 we recall some results about quasi-interpolation of functions by smooth splines of order $m$ (of degree $m-1$) on the uniform grid of the step size $h = 1/n$. In Section 4 we introduce and justify our method based on quasi-interpolation of the products $\mathcal{A}(t,s)v(s)$ and $\mathcal{B}(t,s)v(s)$ by smooth splines. The use of smooth splines of order $m$ enables an $m$-fold reduction of degrees of freedom, compared with the traditional use of discontinuous splines in the product integration methods [1, 3, 10]. The use of quasi-interpolation instead of real interpolation leads to some simplification of the numerical algorithm. Section 5 is devoted to the computation of the coefficients in the matrix form of the method. An important advantage of our method is that we can present simple exact formulae for the coefficients occurring in the discretized problem. Unfortunately, for big $n$, far from the main diagonal, these formulae become numerically unstable, so we complete them by traditional numerically stable quadrature approximation; the error of quadrature approximation rapidly decreases as we move away from the main diagonal. The use of only exact formulae in double precision arithmetics for all coefficients is usually quite acceptable in engineer computations. A numerical check and illustration of the method is performed in Section 6; also the interval arithmetics approach is used for the analysis of the numerical accuracy of exact formulae and the quadrature computation of the coefficients.

Linear integral equations with a logarithmic diagonal singularity often occur modelling physical processes. For instance, one of the main equations in radiative transfer theory, the Milne integral equation [2], has the form

$$u(x) = \frac{1}{2}\int_0^b a(y)E(x-y)u(y)\,dy + f(x),$$

where

$$E(x) = \int_x^\infty \frac{e^{-s}}{s}\,ds = -\log x + c + x - \frac{x^2}{2\cdot 2!} + \frac{x^3}{3\cdot 3!} - \frac{x^4}{4\cdot 4!} + \cdots, \quad x > 0,$$

is the integral exponent function, $c = \lim_{n\to\infty}\big(\sum_{k=1}^n \frac{1}{k} - \log n\big) \approx 0.5772$ is the Euler constant.

The present paper is a continuation of our work [13], where the convergence, error and numerical analysis of the same method has been presented for the integral equation

$$u(x) = \int_0^1 \big[a(x,y)|x-y|^{-\nu} + b(x,y)\big]u(y)\,dy + f(x), \quad 0 \le x \le 1, \qquad (1.2)$$

where $\nu \in (0,1)$. Since the convergence analysis for equation (1.1) differs from that for (1.2) presented in [13] only in small details, we omit detailed proofs and concentrate mainly on the specific numerical aspects of the method for equation (1.1).

Denote by $T$ the integral operator of equation (1.1),

$$(Tu)(x) = \int_0^1 \big(a(x,y)\log|x-y| + b(x,y)\big)u(y)\,dy.$$

The following lemma can be proved by standard argument, cf. [3, 5].

**Lemma 1.** *Let $a,b \in C([0,1]\times(0,1))$ satisfy for $(x,y)\in[0,1]\times(0,1)$ the inequality*

$$\big|a(x,y)\big| + \big|b(x,y)\big| \le cy^{-\lambda_0}(1-y)^{-\lambda_1},$$

*where $\lambda_0,\lambda_1\in\mathbb{R}$, $\lambda_0<1$, $\lambda_1<1$. Then $T$ maps $C[0,1]$ into $C[0,1]$, and $T: C[0,1]\to C[0,1]$ is compact.*

For $m\in\mathbb{N}$, $\lambda_0,\lambda_1<1$, denote by $C_\star^m(0,1)$ and $C_*^{m,\lambda_0,\lambda_1}(0,1)$ the weighted spaces of functions $u\in C[0,1]\cap C^m(0,1)$ such that, respectively,

$$\|u\|_{C_\star^m(0,1)} := \sum_{k=0}^m \sup_{0<x<1} x^k(1-x)^k\big|u^{(k)}(x)\big| < \infty,$$

$$\|u\|_{C_*^{m,\lambda_0,\lambda_1}(0,1)} := \sum_{k=0}^m \sup_{0<x<1} w_{k-1+\lambda_0}(x)w_{k-1+\lambda_1}(1-x)\big|u^{(k)}(x)\big| < \infty,$$

where

$$w_\rho(r) = \begin{cases}1, & \rho<0,\\ r^\rho/(1+|\log r|), & \varrho\ge 0,\end{cases} \qquad r,\rho\in\mathbb{R},\ r>0.$$

Clearly, $C^m[0,1]\subset C_*^{m,\lambda_0,\lambda_1}(0,1)\subset C_\star^m(0,1)$. Denote $\partial_x^k\partial_y^l = (\partial/\partial x)^k(\partial/\partial y)^l$.

**Lemma 2 [see [9]].** *Let $a,b\in C^m([0,1]\times(0,1))$ and let for $k+l\le m$, $(x,y)\in[0,1]\times(0,1)$,*

$$\big|\partial_x^k\partial_y^l a(x,y)\big| + \big|\partial_x^k\partial_y^l b(x,y)\big| \le cy^{-\lambda_0-l}(1-y)^{-\lambda_1-l},$$

*where $\lambda_0<1$, $\lambda_1<1$. Then $T$ maps the Banach spaces $C_\star^m(0,1)$ and $C_*^{m,\lambda_0,\lambda_1}(0,1)$ into themselves, $T: C_\star^m(0,1)\to C_\star^m(0,1)$ is bounded, $T^2: C_\star^m(0,1)\to C_\star^m(0,1)$ is compact, and $T: C_*^{m,\lambda_0,\lambda_1}(0,1)\to C_*^{m,\lambda_0,\lambda_1}(0,1)$ is compact.*

Denote $\mathcal{N}(I-T) = \{u\in C[0,1]\colon u=Tu\}$. The following theorem is a consequence of Lemmas 1 and 2.

**Theorem 1.** *Assume the conditions of Lemma 2 and $\mathcal{N}(I-T)=\{0\}$. Then for $f\in C_\star^m(0,1)$, equation (1.1) has a solution $u\in C_\star^m(0,1)$ which is unique in $C[0,1]$, and $\|u\|_{C_\star^m(0,1)}\le c\,\|f\|_{C_\star^m(0,1)}$. Further, if $f\in C_*^{m,\lambda_0,\lambda_1}(0,1)$ then also $u\in C_*^{m,\lambda_0,\lambda_1}(0,1)$, and $\|u\|_{C_*^{m,\lambda_0,\lambda_1}(0,1)}\le c\,\|f\|_{C_*^{m,\lambda_0,\lambda_1}(0,1)}$. The constant $c$ is independent of $f$.*

Our main results will be established under assumptions of Theorem 1.

## 2 The Smoothing Change of Variables

In the integral equation (1.1), we perform the change of variables

$$x = \varphi(t), \qquad y = \varphi(s), \quad 0 \le t \le 1,\ 0 \le s \le 1,$$

where $\varphi : [0,1] \to [0,1]$ is defined by the formula [7, 8, 13]

$$\varphi(t) = \frac{1}{c_\star}\int_0^t \sigma^{r_0-1}(1-\sigma)^{r_1-1}\, d\sigma,$$
$$c_\star = \int_0^1 \sigma^{r_0-1}(1-\sigma)^{r_1-1}\, d\sigma = \frac{\Gamma(r_0)\Gamma(r_1)}{\Gamma(r_0+r_1)},$$

$\Gamma$ is the Euler gamma function, $r_0, r_1 \in \mathbb{R}$, $r_0 \ge 1$, $r_1 \ge 1$; practicable algorithms correspond to $r_0, r_1 \in \mathbb{N}$. Clearly, $\varphi(0) = 0$, $\varphi(1) = 1$ and $\varphi(t)$ is strictly increasing in $[0,1]$. Hence $\frac{\varphi(t)-\varphi(s)}{t-s} > 0$, $|\varphi(t)-\varphi(s)| = \frac{\varphi(t)-\varphi(s)}{t-s}|t-s|$ for $s \ne t$, and equation (1.1) takes the form

$$v(t) = \int_0^1 \big(\mathcal{A}(t,s)\log|t-s| + \mathcal{B}(t,s)\big)v(s)\, ds + g(t), \quad 0 \le t \le 1, \tag{2.1}$$

where $v(t) = u(\varphi(t))$ is the new function we look for,

$$g(t) = f\big(\varphi(t)\big), \qquad \mathcal{A}(t,s) = a\big(\varphi(t),\varphi(s)\big)\varphi'(s),$$
$$\mathcal{B}(t,s) = \big[a\big(\varphi(t),\varphi(s)\big)\log\big(\Phi(t,s)\big) + b\big(\varphi(t),\varphi(s)\big)\big]\varphi'(s),$$
$$\Phi(t,s) = \left\{\begin{matrix} \frac{\varphi(t)-\varphi(s)}{t-s}, & t \ne s \\ \varphi'(s), & t = s \end{matrix}\right\}.$$

About the boundary behaviour of $\mathcal{A}(t,s)$, see [13]; together with estimates [13] for $\Phi(t,s)$, also the boundary behaviour of $\mathcal{B}(t,s)$ is clear. In particular, the following is true under conditions of Lemma 1: if $r_0, r_1 \ge 1$ satisfy

$$r_0 > 1/(1-\lambda_0), \qquad r_1 > 1/(1-\lambda_1), \tag{2.2}$$

then for $(t,s) \in [0,1]\times(0,1)$, with $\delta_0 = (1-\lambda_0)r_0 - 1 > 0$, $\delta_1 = (1-\lambda_1)r_1 - 1 > 0$, it holds

$$\big|\mathcal{A}(t,s)\big| \le cs^{\delta_0}(1-s)^{\delta_1}, \qquad \big|\mathcal{B}(t,s)\big| \le cs^{\delta_0}(1-s)^{\delta_1}\big|\log s(1-s)\big|. \tag{2.3}$$

The following is true under conditions of Lemma 2: if $r_0, r_1 \ge 1$ satisfy

$$r_0 > m/(1-\lambda_0), \qquad r_1 > m/(1-\lambda_1), \tag{2.4}$$

then for $(t,s) \in [0,1]\times(0,1)$, with $\delta_0 = (1-\lambda_0)r_0 - m > 0$, $\delta_1 = (1-\lambda_1)r_1 - m > 0$, it holds

$$\big|\mathcal{A}(t,s)\big| \le cs^{m-1+\delta_0}(1-s)^{m-1+\delta_1}, \tag{2.5}$$
$$\big|\mathcal{B}(t,s)\big| \le cs^{m-1+\delta_0}(1-s)^{m-1+\delta_1}\big|\log s(1-s)\big|;$$

moreover, for $u \in C_\star^m(0,1)$, $v(s) = u(\varphi(s))$, with a constant $c$ independent of $u$,

$$\left|\partial_s^m \left[\mathcal{A}(t,s)v(s)\right]\right| \le c s^{-1+\delta_0}(1-s)^{-1+\delta_1} \|u\|_{C_\star^m(0,1)}, \tag{2.6}$$

$$\left|\partial_s^m \left[\mathcal{B}(t,s)v(s)\right]\right| \le c s^{-1+\delta_0}(1-s)^{-1+\delta_1} \left|\log s(1-s)\right| \|u\|_{C_\star^m(0,1)}. \tag{2.7}$$

About the boundary behaviour of $v(t) = u(\varphi(t))$ we have on the basis of Lemma 3.1 of [13] the following result: if $u \in C_\star^m(0,1)$ then also $v \in C_\star^m(0,1)$; for $u \in C_\star^{m,\lambda_0,\lambda_1}(0,1)$, $j = 1,\dots,m$, $0 < t < 1$, it holds

$$\left|v^{(j)}(t)\right| \le c\|u\|_{C_\star^{m,\lambda_0,\lambda_1}(0,1)} \left\{\begin{matrix} t^{r_0-j}, & \lambda_0 < 0 \\ t^{(1-\lambda_0)r_0-j}|\log t|, & 0 \le \lambda_0 < 1 \end{matrix}\right\}$$
$$\times \left\{\begin{matrix} (1-t)^{r_1-j}, & \lambda_1 < 0 \\ (1-t)^{(1-\lambda_1)r_1-j}|\log t|, & 0 \le \lambda_1 < 1 \end{matrix}\right\}.$$

We see, in particular, that for $\lambda_0 < 0$, $\lambda_1 < 0$, $r_0 > m$, $r_1 > m$ we have the implication

$$\begin{aligned} u \in C_\star^{m,\lambda_0,\lambda_1}(0,1) \quad &\Longrightarrow \quad v \in C^m[0,1], \\ v^{(j)}(0) = v^{(j)}(1) = 0, &\quad j = 1,\dots,m. \end{aligned} \tag{2.8}$$

For $0 \le \lambda_0 < 1$, $0 \le \lambda_1 < 1$, (2.8) holds true under conditions (2.4).

We extend $\mathcal{A}(t,s)$ and $\mathcal{B}(t,s)$ with respect to $s$ outside $(0,1)$ by the zero value. Under conditions (2.2) we obtain continuous functions on $[0,1] \times \mathbb{R}$, see (2.3).

## 3 Interpolation and Quasi-Interpolation by Splines

### 3.1 The father B-spline

The *father* B-*spline* $B_m$ *of order* $m$ (or of degree $m-1$) can be defined by the formula

$$B_m(x) = \frac{1}{(m-1)!}\sum_{i=0}^{m}(-1)^i\binom{m}{i}(x-i)_+^{m-1}, \quad x \in \mathbb{R},\ m \in \mathbb{N}.$$

Here, as usual, $0! = 1$, $0^0 := \lim_{x\downarrow 0} x^x = 1$,

$$\binom{m}{i} = \frac{m!}{i!\,(m-i)!}, \qquad (x-i)_+^{m-1} = \begin{cases}(x-i)^{m-1}, & x-i \ge 0, \\ 0, & x-i < 0.\end{cases}$$

Let us recall some properties of $B_m$:

$$\operatorname{supp} B_m = [0,m], \qquad B_m(x) = B_m(m-x) > 0 \quad \text{for } 0 < x < m, \tag{3.1}$$
$$B_m \in C^{(m-2)}(\mathbb{R}),$$

$$B_m^{(m-1)}(x) = (-1)^l\binom{m-1}{l}, \quad \text{for } l < x < l+1,\ l = 0,\dots,m-1, \tag{3.2}$$

$$\int_{\mathbb{R}} B_m(x)\,dx = 1, \qquad \sum_{j\in\mathbb{Z}} B_m(x-j) = 1, \quad x \in \mathbb{R}. \tag{3.3}$$

## 3.2 Spline interpolation

Introduce in $\mathbb{R}$ the uniform grid $h\mathbb{Z} = \{ih\colon i \in \mathbb{Z}\}$ of the step size $h > 0$. Denote by $S_{h,m}$, $m \in \mathbb{N}$, the space of (maximally smooth) splines of order $m$ (of degree $m-1$) and defect 1 with the knot set $h\mathbb{Z}$. It consists of all functions $\sum_{j\in\mathbb{Z}} d_j B_m(h^{-1}x - j)$ where $d_j \in \mathbb{R}$ or $\mathbb{C}$. Further denote by $BC(\mathbb{R})$ the space of bounded continuous functions on $\mathbb{R}$ equipped with the norm $\|f\|_\infty = \sup_{x\in\mathbb{R}} |f(x)|$. For $v \in BC(\mathbb{R})$, the spline interpolant $Q_{h,m}v \in S_{h,m} \cap BC(\mathbb{R})$ is determined by the conditions

$$(Q_{h,m}v)(x) = \sum_{j\in\mathbb{Z}} d_j B_m\big(h^{-1}x - j\big),$$
$$(Q_{h,m}v)\big((k + m/2)h\big) = v\big((k + m/2)h\big), \quad k \in \mathbb{Z}. \tag{3.4}$$

For $m = 1$ and $m = 2$, $Q_{h,m}v$ is the usual piecewise constant, respectively, piecewise linear interpolant which can be determined on every subinterval $[ih, (i+1)h]$, $i \in \mathbb{Z}$, independently of other subintervals. For $m \geq 3$, the value of $Q_{h,m}v$ at a given point $x \in \mathbb{R}$ depends on the values of $f$ at all interpolation knots $(k + \frac{m}{2})h$, $k \in \mathbb{Z}$. It occurs (see [11]) that also for $m \geq 3$ conditions (3.4) uniquely determine $d_j$, $j \in \mathbb{Z}$, in the space of bounded bisequences, namely,

$$d_j = \sum_{k\in\mathbb{Z}} \alpha_{j-k,m} v\left(\left(k + \frac{m}{2}\right)h\right), \quad j \in \mathbb{Z}, \quad \alpha_{k,m} = \sum_{l=1}^{m_0} \frac{z_{l,m}^{m_0-1}}{P'_m(z_{l,m})} z_{l,m}^{|k|}, \quad k \in \mathbb{Z},$$

where

$$m_0 = \begin{cases} (m-2)/2 & \text{for even } m, \\ (m-1)/2 & \text{for odd } m, \end{cases}$$

and $z_{l,m} \in (-1, 0)$, $l = 1, \dots, m_0$, are the roots of the polynomial $P_m(z) = \sum_{|k|\leq m_0} B_m(k + \frac{m}{2}) z^{k+m_0}$ of degree $2m_0$ (the remaining $m_0$ roots are $z_{l+m_0,m} = 1/z_{l,m} \in (-\infty, -1)$, $l = 1, \dots, m_0$).

**Lemma 3 [see [4, 15]].** *If $v$ is $m$ times differentiable in $\mathbb{R}$ and $v^{(m)}$ is bounded then*

$$\|v - Q_{h,m}v\|_\infty \leq \Phi_{m+1} \pi^{-m} h^m \big\|v^{(m)}\big\|_\infty,$$

*where $\Phi_m = \frac{4}{\pi} \sum_{k=0}^\infty \frac{(-1)^{km}}{(2k+1)^m}$, $m \in \mathbb{N}$, is the Favard constant.*

## 3.3 Spline quasi-interpolation

For $m \geq 3$, $p \in \mathbb{N}$, we define the quasi-interpolant $Q^{(p)}_{h,m}v \in S_{h,m}$ of $v \in BC(\mathbb{R})$ by

$$\big(Q^{(p)}_{h,m}v\big)(x) = \sum_{j\in\mathbb{Z}} d^{(p)}_j B_m\big(h^{-1}x - j\big), \quad x \in \mathbb{R},$$
$$d^{(p)}_j = \sum_{|k|\leq p-1} \alpha^{(p)}_{k,m} v\left(\left(j - k + \frac{m}{2}\right)h\right),$$

where

$$\alpha_{k,m}^{(p)} = \sum_{q=|k|}^{p-1} (-1)^{k+q} \binom{2q}{k+q} \gamma_{q,m}, \quad |k| \le p-1, \tag{3.5}$$

$$\gamma_{0,m} = 1, \qquad \gamma_{q,m} = \sum_{l=1}^{m_0} \frac{(1+z_{l,m}) z_{l,m}^{m_0+q-1}}{(1-z_{l,m})^{2q+1} P'_m(z_{l,m})}, \quad q \ge 1.$$

It occurs [6] that for $v$ with uniformly continuous $m$th derivative $v^{(m)}$ and $2p > m$, the quasi-interpolant $Q_{h,m}^{(p)} v$ is asymptotically of the same accuracy as the interpolant $Q_{h,m} v$. It is reasonable to take the smallest $p \in \mathbb{N}$ for which $2p > m$; denote it by $m_1$,

$$m_1 = \left\{ \begin{matrix} m/2+1 & \text{for even } m \\ (m+1)/2 & \text{for odd } m \end{matrix} \right\} = m - m_0; \tag{3.6}$$

denote also

$$Q'_{h,m} := Q_{h,m}^{(m_1)}, \qquad \alpha'_{k,m} := \alpha_{k,m}^{(m_1)}, \quad |k| < m_1. \tag{3.7}$$

The quasi-interpolant $Q'_{h,m} v \in S_{h,m}$ can be constructed on every subinterval $[ih, (i+1)h]$, $i \in \mathbb{Z}$, independently of other subintervals provided that $v$ is given on $[(i-m+1)h, (i+m)h] \cap \{(j+\frac{m}{2})h \colon j \in \mathbb{Z}\}$.

We assumed that $m \ge 3$. For $m = 1$ and $m = 2$ we may put $Q'_{h,m} = Q_{h,m}$.

**Lemma 4 [see [6]].** *For $i \in \mathbb{Z}$, $v \in C^m[(i-m+1)h, (i+m)h]$, it holds*

$$\max_{ih \le x \le (i+1)h} \left| v(x) - (Q'_{h,m} v)(x) \right| \le c h^m \sup_{(i-m+1)h \le x \le (i+m)h} \left| v^{(m)}(x) \right|,$$

*where the constant $c$ is independent of $f$, $h$ and $i$ (more about $c$ see in [6]).*

## 4 Product Quasi-Interpolation Method

Let $h = 1/n$, $n \in \mathbb{N}$, $n \ge m$. We approximate equation (2.1) by its discretized version

$$v_n(t) = \int_0^1 \left[ \left(\log|t-s|\right) Q'_{h,m}\left(\mathcal{A}(t,s) v_n(s)\right) + Q'_{h,m}\left(\mathcal{B}(t,s) v_n(s)\right) \right] ds + g(t), \tag{4.1}$$

where the quasi-interpolation operator $Q'_{h,m}$ is applied to products $\mathcal{A}(t,s)v(s)$ and $\mathcal{B}(t,s)v(s)$ as functions of $s$. This can be done for $v$ given on $[0,1]$ since $\mathcal{A}(t,s) = \mathcal{B}(t,s) = 0$ for $s \le 0$ and for $s \ge 1$; recall that under conditions (2.2) it holds that $\mathcal{A}, \mathcal{B} \in C([0,1] \times \mathbb{R})$. Since $Q'_{h,m}(\mathcal{A}(t,s)v_n(s))$ and $Q'_{h,m}(\mathcal{B}(t,s)v_n(s))$ depend on the function $v_n$ through its knot values $v_n((k+\frac{m}{2})h)$, we obtain a closed system to determine $v_n((k+\frac{m}{2})h)$ by collocating equation (4.1) at the points $(i+\frac{m}{2})h$. In this way, the following system of equations can be derived (cf. [13]):

$$v_{i,n} = \sum_{k=-m_0}^{n-m_1} \tau_{i,k} v_{k,n} + g_i, \quad i = -m_0, \ldots, n-m_1, \tag{4.2}$$

where

$$v_{i,n} = v_n\big((i+m/2)h\big), \quad g_i = g\big((i+m/2)h\big), \ i = -m_0, \dots, n-m_1, \quad (4.3)$$

$$\tau_{i,k} = \sum_{j=k-m_1+1}^{k+m_1-1} \alpha'_{j-k,m}\big(\beta_{i,j} a_{i,k} + \beta_j^0 b_{i,k}\big), \quad i,k = -m_0, \dots, n-m_1, \quad (4.4)$$

$$\begin{aligned} a_{i,k} &= \mathcal{A}\big((i+m/2)h, (k+m/2)h\big), \\ b_{i,k} &= \mathcal{B}\big((i+m/2)h, (k+m/2)h\big), \end{aligned} \quad i,k = -m_0, \dots, n-m_1, \quad (4.5)$$

$$\beta_j^0 = \int_0^1 B_m(ns-j)\,ds, \quad j = -m+1, \dots, n-1, \quad (4.6)$$

$$\beta_{i,j} = \int_0^1 \log\big|(i+m/2)h - s\big| B_m(ns-j)\,ds,$$
$$i = -m_0, \dots, n-m_1, \ j = -m+1, \dots, n-1; \quad (4.7)$$

recall also formulae (3.5)–(3.7) for $\alpha'_{k,m}$, $m_0$ and $m_1$. The dimension of system (4.2) is $n-1$ for even and $n$ for odd $n$. Having the quantities $\alpha'_{k,m}$, $a_{i,k}$, $b_{i,k}$, $\beta_j^0$, $\beta_{i,j}$ in hand, the computation of the matrix elements $\tau_{i,k}$, $i,k = -m_0, \dots, n-m_1$ costs approximately $3mn^2$ flops. See [13] for a numerically stable evaluation of the function $\Phi(t,s)$ occurring in the definition of $\mathcal{B}(t,s)$ and for some other computational details, in particular, for the computation of $u_n(x) := v_n(\varphi^{-1}(x))$, $x \in [0,1]$, the approximate solution to (1.1). The evaluation of integrals (4.6) defining $\beta_j^0$ is elementary, in particular, $\beta_j^0 = h$ for $j = 0, \dots, n-m$, due to (3.3). It remains to explain how to evaluate the integrals (4.7) defining $\beta_{i,j}$. This is the most delicate part of the method. In Section 5 we present two exact and one approximate algorithm to compute $\beta_{i,j}$; each of algorithms has its advantages and disadvantages. Exploiting the symmetries, the number of different integrals reduces to $O(n)$, and they can be computed in $O(n)$ flops. System (4.2) can be solved in $O(n^2\log n)$ flops by GMRES ($n^2$ flops per a matrix to vector multiplication, $O(\log n)$ iterations, see [14]). Thus the complexity of the method is $O(n^2\log n)$ flops.

Having found the solution $v_{i,n}$, $i = -m_0, \dots, n-m_1$, of system (4.2) we can use (4.1) to define the Nystrom extension $v_n(t)$ of the grid solution for any $t \in [0,1]$; this is expensive. An essentially cheaper way is to construct the quasi-interpolant $\widetilde{v}_n = Q_{h,m}v_n$ of $v_n$ using the computed knot values $v_{i,n} = v_n((i+\frac{m}{2})h)$, $i = -m_0, \dots, n-m_1$, completed by $v_{i,n} = v_{-m_0,n}$ for $i < -m_0$ and $v_{i,n} = v_{n-m_1,n}$ for $i > n-m_1$; recall that the construction is independent for every subinterval $[ih, (i+1)h]$, $0 \le i \le n-1$. (A slightly more accurate way is to compute $v_n(0)$ and $v_n(1)$ from (4.1) and put $v_{i,n} = v_n(0)$ for $i < -m_0$ and $v_{i,n} = v_n(1)$ for $i > n-m_1$.)

**Theorem 2.** (i) *Let $a$ and $b$ satisfy the conditions of Lemma* 1, $\mathcal{N}(I-T) = \{0\}$, $f \in C[0,1]$, *and let $r_0$ and $r_1$ satisfy conditions* (2.2). *Then for sufficiently large $n$, system* (4.2) *is uniquely solvable, equation* (4.1) *has a unique solution* $v_n \in C[0,1]$, *and*

$$\|v - v_n\|_\infty := \max_{0\le t\le 1}\big|v(t) - v_n(t)\big| \to 0, \qquad \|v - \widetilde{v}_n\|_\infty \to 0 \quad \text{as } n \to \infty,$$

*where $v \in C[0,1]$ is the unique solution of equation* (2.1).

(ii) *Let $a$ and $b$ satisfy the conditions of Lemma* 2, $\mathcal{N}(I-T)=\{0\}$, $f \in C^m_\star(0,1)$, *and let $r_0, r_1 \geq 1$ satisfy conditions* (2.4). *Then*

$$\|v - v_n\|_\infty \leq ch^m \, \|f\|_{C^m_\star(0,1)} \, . \tag{4.8}$$

(iii) *Again, let $a$ and $b$ satisfy the conditions of Lemma* 2, $\mathcal{N}(I-T)=\{0\}$, *but $f \in C^{m,\lambda_0,\lambda_1}_*(0,1)$; let $r_0 > m$, $r_1 > m$ and let $r_0, r_1$ satisfy conditions* (2.4). *Then*

$$\|v - \widetilde{v}_n\|_\infty \leq ch^m \, \|f\|_{C^{m,\lambda_0,\lambda_1}_*(0,1)} \, , \tag{4.9}$$

*where $\widetilde{v}_n$ is the quasi-interpolant constructed onto the solution of system* (4.2). *Constant $c$ in* (4.8) *and* (4.9) *is independent of $n$ and $f$.*

*Proof.* We obtain the claims of the theorem repeating the arguments of the proofs of Theorems 4.1 and 4.2 in [13] but involving estimates (2.3) in the proof of (i), (2.5)–(2.7) in the proof of (ii), and (2.5)–(2.8) in the proof of (iii). □

While having in mind the implementation of the proposed methods using finite precision arithmetics, we observe that from the perspective on numerical stability the main difficulty is the computation of quadrature coefficients $\beta_{i,j}$ defined in (4.7). Therefore, we propose and analyse three different calculation schemes for $\beta_{i,j}$ in the next section with the numerical verification of the proposed algorithms in Section 6.

## 5 Computation of $\beta_{i,j}$

The condition number of system (4.2) is uniformly bounded in $n$, and small perturbations $\Delta\beta_{ij}$ of $\beta_{ij}$ cause the error $|\Delta v_h| \leq c \max_{i,j} |\Delta\beta_{ij}|$ in the spline collocation solution $v_n$ of (4.1), where the constant $c$ is independent of $n$.

We elaborate on three different computation schemes for $\beta_{i,j}$; the first two of them are based on exact arithmetics while the third one uses an approximate method for calculation of underlying integrals. We will see that the exact schemes compared with the inexact one behave differently for distinct values of $i$ and $j$, namely depending on the value of modulus $|i-j|$.

### 5.1 Exact algorithm 1

The integrals of type (4.7) can be evaluated exploiting the following result.

**Lemma 5 [see [13]].** *For $w \in L^1(0,1)$, it holds*

$$\int_0^1 w(s) B_m(ns-j)\, ds = h\big(D_h^m z_m\big)(jh), \quad h = 1/n, \ j \in \mathbb{Z},$$

*where the difference operator* $D_h^m = (D_h)^m$ *is defined by*

$$(D_h u)(x) = h^{-1}\big(u(x+h) - u(x)\big),$$
$$(D_h^m u)(x) = h^{-m}\sum_{k=0}^{m}(-1)^{m-k}\binom{m}{k}u(x+kh);$$
$$z_m(s) = \begin{cases} w^{(-m)}(s), & 0 \le s \le 1,\\ \sum_{k=0}^{m-1}\frac{w^{(k-m)}(0)}{k!}s^k, & s<0,\\ \sum_{k=0}^{m-1}\frac{w^{(k-m)}(1)}{k!}(s-1)^k, & s>1,\end{cases}$$

$w^{(-m)} \in C^{m-1}[0,1]$ *is an integral function of order* $m$ *for* $w$ *on* $[0,1]$, *i.e. in the sense of distributions,* $(d/ds)^m w^{(-m)}(s) = w(s)$, $0<s<1$, *and* $w^{(k-m)}(s) = (d/ds)^k w^{(-m)}(s)$, $0 \le s \le 1$, $k = 0,\dots,m-1$.

**Lemma 6.** *Let* $t \in \mathbb{R}$ *be fixed. An integral function* $w_t^{(-m)} \in C^{m-1}[0,1]$ *of order* $m$ *for* $w_t(s)=\log|t-s|$, $0<s<1$, *is given by*

$$w_t^{(-m)}(s) = \frac{1}{m!}\begin{cases}(-1)^m(t-s)^m\left[\log(t-s) - c_m\right], & 0 \le s \le \min\{1,t\},\\ (s-t)^m\left[\log(s-t) - c_m\right], & \max\{0,t\} \le s \le 1,\end{cases}$$

*where* $c_m = 1 + \frac{1}{2} + \cdots + \frac{1}{m}$.

*Proof.* This claim can be easily checked by induction in $m$. □

Applying Lemmas 5 and 6 we obtain the (exact) formula

$$\beta_{i,j} = \frac{1}{m!}h\triangle^m\gamma_{i,j}, \quad i = -m_0,\dots,n-m_1,\ j = -m+1,\dots,n-1, \tag{5.1}$$

where $\triangle\gamma_j = \gamma_{j+1} - \gamma_j$ is the forward difference and

$$\gamma_{i,j} = \begin{cases} \sum_{k=0}^{m-1}(-1)^{m-k}\binom{m}{k}(i+\frac{m}{2})^{m-k}[\log((i+\frac{m}{2})h) - c_{m-k}]j^k, & j<0,\\ (-1)^m(i+\frac{m}{2}-j)^m[\log((i+\frac{m}{2}-j)h) - c_m], & 0 \le j < i+\frac{m}{2},\\ 0, & j = i+\frac{m}{2},\\ (j-i-\frac{m}{2})^m[\log((j-i-\frac{m}{2})h) - c_m], & i+\frac{m}{2} < j \le n,\\ \sum_{k=0}^{m-1}\binom{m}{k}(n-i-\frac{m}{2})^{m-k}[\log((n-i-\frac{m}{2})h) - c_{m-k}](j-n)^k, & j>n \end{cases} \tag{5.2}$$

(the difference $\triangle^m\gamma_{i,j}$ is taken with respect to the argument $j$).

Exact formulae for $\beta_{i,j}$ are delicate. Formula (5.1)–(5.2) is suitable for moderate values of $|i-j+\frac{m}{2}|$. On the other hand, for great $|i-j|$, the quantities $\gamma_{i,j}$ are large in modulus (comparable with $|i-j|^m$) in contrast to their differences $\Delta^m\gamma_i$ (which are comparable with $m!$), causing a loss of accuracy in standard floating-point arithmetics such as IEEE single- or double-precision floating-point format. Nevertheless, for $|i-j|^m/m! \le 10^8$, using double-precision arithmetics, we obtain seven correct decimal digits of $\gamma_{i,j}$ by (5.1)–(5.2), which is usually sufficient in engineer calculations. For instance, in case $m = 4$ (cubic splines), seven correct digits of $\gamma_{i,j}$ are obtained for $|i-j| \le 220$; according to error estimates (4.8) and (4.9), $m = 4$, $n \approx 200$ usually yields a sufficient accuracy of the approximate solution.

If more accurate approximation is needed, formulae (5.1)–(5.2) can be applied for all $|i-j|$ only if a sufficiently high precision arithmetics is used; in Section 6.1 we illustrate the situation numerically. Alternatively, for great $|i-j|$, $\gamma_{i,j}$ can be computed in a numerically stable way by quadratures using standard arithmetics. The accuracy of quadratures rapidly grows as $|i-j|$ increases; see Section 5.3. The quadrature rules are based on another way for an exact computation of $\beta_{i,j}$ described in Section 5.2.

## 5.2 Exact algorithm 2

In case of equation (1.1), $\beta_{i,j}$ can be evaluated exactly integrating by parts. Since due to (3.1), $B_m(ns-j)$ is supported on the interval $[jh,(j+m)h]$, we have for $i = -m_0, \dots, n-m_1$, $j = -m+1, \dots, n-1$,

$$\begin{aligned}
\beta_{i,j} &= \int_0^1 \log\left|\left(i+\frac{m}{2}\right)h - s\right| B_m(ns-j)\,ds \\
&= \int\limits_{\max\{jh,0\}}^{\min\{(j+m)h,1\}} \log\left|\left(i+\frac{m}{2}\right)h - s\right| B_m(ns-j)\,ds \quad (\text{change } x = ns-j) \\
&= h \int\limits_{\max\{0,-j\}}^{\min\{m,n-j\}} \log\left|\left(i-j+\frac{m}{2}-x\right)h\right| B_m(x)\,dx \\
&= h \sum_{\ell=\max\{0,-j\}}^{\min\{m,n-j\}-1} \int_\ell^{\ell+1} \log\left|\left(i-j+\frac{m}{2}-x\right)h\right| B_m(x)\,dx. \qquad (5.3)
\end{aligned}$$

In the interval $[\ell,\ell+1]$, $0 \le \ell \le m-1$, $B_m(x) =: p_\ell(x)$ is a polynomial of degree $m-1$, $p_\ell(s) = \sum_{k=0}^{m-1} \xi_{k,\ell} s^k$ with some known coefficients $\xi_{k,\ell}$. Introduce the integral function

$$p_{\ell,i-j}^{(-1)}(x) = \int_{i-j+m/2}^{x} p_\ell(s)\,ds = \sum_{k=0}^{m-1} \frac{\xi_{k,\ell}}{k+1}\left(x^{k+1} - \left(i-j+\frac{m}{2}\right)^{k+1}\right),$$

it is a polynomial of degree $m$. Integration by parts yields

$$\begin{aligned}
\beta_{i,j} = h \sum_{\ell=\max\{0,-j\}}^{\min\{m,n-j\}-1} \Bigg\{ &\left[\log\left|\left(i-j+\frac{m}{2}-x\right)h\right| p_{\ell,i-j}^{(-1)}(x)\right]_{x=\ell}^{\ell+1} \\
&- \int_\ell^{\ell+1} \frac{p_{\ell,i-j}^{(-1)}(x)\,dx}{x-(i-j+\frac{m}{2})} \Bigg\}. \qquad (5.4)
\end{aligned}$$

The integrals in (5.4) can be computed exactly since the integrands are polynomials of degree $m-1$:

$$\begin{aligned}\frac{p^{(-1)}_{\ell,i-j}(x)}{x-(i-j+\frac{m}{2})} &= \sum_{k=0}^{m-1}\frac{\xi_{k,\ell}}{k+1}\sum_{l=0}^{k}\Big(i-j+\frac{m}{2}\Big)^{k-l}x^l\\ &= \sum_{l=0}^{m-1}\Bigg(\sum_{k=l}^{m-1}\frac{\xi_{k,\ell}}{k+1}(i-j+\frac{m}{2})^{k-l}\Bigg)x^l,\\ \int_{\ell}^{\ell+1}\frac{p^{(-1)}_{\ell,i-j}(x)\,dx}{x-(i-j+\frac{m}{2})} &= \sum_{l=0}^{m-1}\Bigg(\sum_{k=l}^{m-1}\frac{\xi_{k,\ell}}{k+1}(i-j+\frac{m}{2})^{k-l}\Bigg)\frac{(\ell+1)^{l+1}-\ell^{l+1}}{l+1}.\end{aligned} \tag{5.5}$$

Alternatively, omitting the division, the integrals in (5.4) can evaluated by an exact $\mu$-point Gauss quadrature formula, where $2\mu\ge m$, i.e.

$$\mu\ge\begin{cases}m/2 & \text{for even } m,\\ (m+1)/2 & \text{for odd } m.\end{cases}$$

For great $|i-j+\frac{m}{2}|$, the intermediate quantities $p^{(-1)}_{\ell,i-j}(\ell)$ and $p^{(-1)}_{\ell,i-j}(\ell+1)$ are great, and again a loss of accuracy takes place in standard arithmetics.

## 5.3 A numerically stable approximate algorithm

We obtain a numerically stable algorithm approximating the integrals in (see (5.3))

$$\beta_{i,j}=h\sum_{\ell=\max\{0,-j\}}^{\min\{m,n-j\}-1}\int_{\ell}^{\ell+1}\log\Big|\Big(i-j+\frac{m}{2}-x\Big)h\Big|B_m(x)\,dx \tag{5.6}$$

by the $\mu$-point Gauss quadrature, $2\mu\ge m$. Fortunately, as we see below, the accuracy of the quadrature increases as $|i-j|$ increases, so we obtain a suitable complement to the exact formulae for $\beta_{i,j}$ presented in Sections 5.1 and 5.2.

Recall that $B_m(x)\ge 0$. For even $m$, also $\log|(i-j+\frac{m}{2}-x)h|$ is sign-constant for $x\in(\ell,\ell+1)$, and the Gauss quadratures are stable numerically for the integrals in (5.6). For odd $m$, the integrand changes its sign when $x=i-j+\frac{m}{2}\pm n=\ell+\frac{1}{2}$ but there is still no danger for the numerical instability.

Assume that $|i-j|>m/2$, i.e. either $i-j+\frac{m}{2}<0$, or $i-j+\frac{m}{2}>m$. Denote by $\mathcal{I}^{(\mu)}_{\ell,i-j}$ the $\mu$-point Gauss quadrature of the integral

$$\mathcal{I}_{\ell,i-j}=\int_{\ell}^{\ell+1}\log\big|\big(i-j+m/2-x\big)h\big|B_m(x)\,dx.$$

As well known,

$$\mathcal{I}_{\ell,i-j}-\mathcal{I}^{(\mu)}_{\ell,i-j}=\frac{(\mu!)^4}{(2\mu+1)!\,((2\mu)!)^2}\left[\Big(\frac{d}{dx}\Big)^{2\mu}\Big\{\log\Big|\Big(i-j+\frac{m}{2}-x\Big)h\Big|B_m(x)\Big\}\right]_{x=\xi},$$

where $\xi \in (\ell, \ell+1)$. Since

$$\left|\left(\frac{d}{dx}\right)^k \log\left|\left(i-j+\frac{m}{2}-x\right)h\right|\right| = (k-1)!\left|i-j+\frac{m}{2}-x\right|^{-k}, \quad k \geq 1,$$

and $B_m$ is a polynomial of degree $m-1$ on $(\ell, \ell+1)$, we have

$$\begin{aligned}
&\left|\left(\frac{d}{dx}\right)^{2\mu}\left\{\log\left|\left(i-j+\frac{m}{2}-x\right)h\right|B_m(x)\right\}\right| \\
&\quad\leq \sum_{k=0}^{2\mu}\binom{2\mu}{k}\left|\left(\frac{d}{dx}\right)^k \log\left|\left(i-j+\frac{m}{2}-x\right)h\right|\right|\left|B_m^{(2\mu-k)}(x)\right| \\
&\quad\leq \sum_{k=2\mu-m+1}^{2\mu}\frac{(2\mu)!}{k!\,(2\mu-k)!}(k-1)!\left|i-j+\frac{m}{2}-x\right|^{-k} c_{\ell,2\mu-k} \\
&\quad\leq \sum_{k=2\mu-m+1}^{2\mu}\frac{(2\mu)!}{(2\mu-k)!\,k}\left|i-j+\frac{m}{2}-x_\ell\right|^{-k} c_{\ell,2\mu-k},
\end{aligned}$$

where $x_\ell = \ell$ for $i-j+\frac{m}{2} < 0$ and $x_\ell = \ell+1$ for $i-j+\frac{m}{2} > m$, whereas

$$c_{\ell,k} = \max_{\ell \leq x \leq \ell+1}\left|B_m^{(k)}(x)\right|.$$

This results to the estimates

$$\left|\mathcal{I}_{\ell,i-j} - \mathcal{I}_{\ell,i-j}^{(\mu)}\right| \leq c_\ell \frac{(\mu!)^4}{(2\mu+1)!(2\mu)!}\sum_{k=2\mu-m+1}^{2\mu}\frac{c_{\ell,2\mu-k}}{(2\mu-k)!\,k}\left|i-j+\frac{m}{2}-x_\ell\right|^{-k}$$

and

$$\begin{aligned}
\left|\beta_{i,j} - \beta_{i,j}^{(\mu)}\right| &\leq h\frac{(\mu!)^4}{(2\mu+1)!(2\mu)!}\sum_{\ell=0}^{m-1}\sum_{k=2\mu-m+1}^{2\mu}\frac{c_{\ell,2\mu-k}}{(2\mu-k)!k}\left|i-j+\frac{m}{2}-x_\ell\right|^{-k} \\
&= h\frac{(\mu!)^4}{(2\mu+1)!(2\mu)!}O\left(|i-j|^{-2\mu+m-1}\right) \quad \text{as } |i-j| \to \infty. \qquad (5.7)
\end{aligned}$$

We see that the accuracy rapidly increases as $|i-j+\frac{m}{2}|$ increases. Note also the smallness of the coefficient in (5.7). For instance, $\frac{(\mu!)^4}{(2\mu+1)!(2\mu)!} \approx 3.6 \cdot 10^{-10}$ for $\mu = 8$.

The accuracy can be even raised if we treat $B_m(x)$ in the integrals (5.6) as a weight function and apply corresponding Gauss type quadrature. Unfortunately, this way is labour consuming.

It is reasonable to compute $\beta_{i,j}$ by exact formulae (5.1)–(5.2) or (5.4)–(5.5) for $|i-j| \leq m$ or perhaps for $|i-j| \leq 2m$ and to apply the eight point Gauss quadrature in (5.6) for $|i-j| > m$, respectively, $|i-j| > 2m$, all in the framework of a standard arithmetics.

## 6 Numerical Experiments

Here we are testing numerically the properties of the methods described above. We start with investigating the calculation schemes for different ways of finding $\beta_{i,j}$ for the choice of the best strategy from the three different methods. For verifying the accuracy of the computed results of $\beta_{i,j}$ we use interval arithmetics with a specified number of bits of precision. We start with testing numerical properties of the exact calculation methods of $\beta_{i,j}$ (algorithms 1 and 2) in interval arithmetics with variable number of bits in precision. The third algorithm, namely the approximate integration method, is performed in standard double precision arithmetics.

We demonstrate that for best numerical accuracy in the case of double precision arithmetics different given algorithms need to be combined. Thereafter we set up a model problem with a known solution for which we observe, that the best way of combining the exact and the approximate scheme, does not depend only on $i$ and $j$ in calculation of $\beta_{i,j}$, but also on the size of discretization parameter $n$ and spline rank $m$. Nevertheless, the rules of thumb formulated in the end of Section 5 give quite acceptable accuracy.

### 6.1 Calculation of $\beta_{i,j}$ accuracy assessments using variable precision interval arithmetics

Here we compare the three different methods given above, namely, the exact method 1 (E1) for $\beta_{i,j}$ calculation described in Section 5.1, the exact method 2 (E2) given in Section 5.2 and the approximate integration method (AIM) outlined in Section 5.3. For testing the accuracy hypotheses given above, we perform cross-comparison tests with one of the methods in standard 53-bit interval arithmetics (or double precision) and the other method calculated using higher precision than normal, if needed, to achieve the maximal diameter of the result in interval arithmetics less than $10^{-15}$. To reach this goal we gradually increase the number of bits in the variable precision, starting from 53, until the goal is achieved. Standard double precision numbers are represented with 53 bits (one bit is reserved for the sign and 52 bits for the mantissa). While using interval field for calculation, the worst possible influence of round-off errors is given by the diameter of the calculated result. This makes it possible to numerically find out the algorithm behavior in case of real calculations with given input data. This technique is applied separately to all combinations of indexes $i$ and $j$ in $\beta_{i,j}$ calculation resulting in method accuracy across different values of $i$ and $j$ indicated with the minimum number of precision bits needed for the result to achieve the needed accuracy. (Due to the large number of computations needed for such calculations, we perform the tests only with moderate $n$.)

#### 6.1.1 Exact method $E1$ for calculation of $\beta_{i,j}$ accuracy

First we calculate $\beta_{i,j}^{E1}$ ($i \in [-m_0, \ldots, n-m_1]$, $j \in [-m+1, n-1]$), $n = 12$, $m = 6$, $m_0 = 2$ , $m_1 = 4$ in double precision arithmetics. In Fig. 1 we plot the error of the result, which is calculated as distance from $\beta_{i,j}^{E2(1e-15)}$. Here, we denote with E2($1e-15$) the algorithm E2, which has been calculated with the

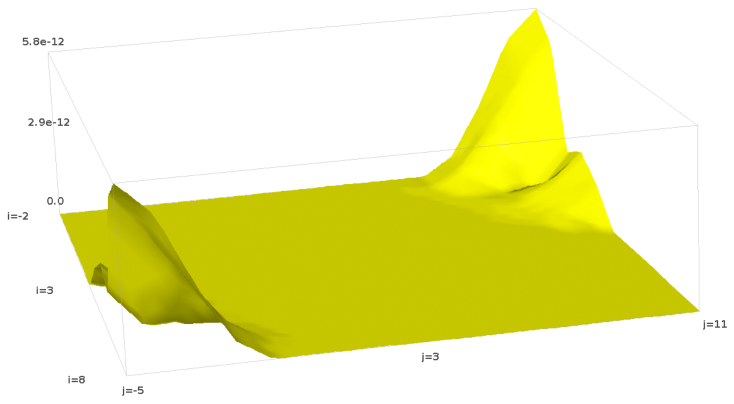

**Figure 1.** Accuracy of method E1 compared with precise E2: $|\beta_{i,j}^{E1} - \beta_{i,j}^{E2(1e-15)}|$, $i \in [-2, 8]$, $j \in [-5, 11]$.

**Table 1.** The number of bits needed to achieve the given accuracy $10^{-15}$ for method E1 ($E1(1e-15)$).

| $i \setminus j$ | $-5$ | $-4$ | $-3$ | $-2$ | $-1$ | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $-2$ | 71 | 70 | 71 | 72 | 72 | 72 | 72 | 73 | 73 | 74 | 75 | 76 | 73 | 72 | 72 | 74 | 75 |
| $-1$ | 73 | 71 | 70 | 71 | 72 | 72 | 72 | 72 | 73 | 73 | 74 | 75 | 72 | 70 | 72 | 72 | 74 |
| 0 | 74 | 73 | 71 | 70 | 71 | 72 | 72 | 72 | 72 | 73 | 73 | 74 | 70 | 69 | 70 | 72 | 72 |
| 1 | 75 | 74 | 73 | 71 | 70 | 71 | 72 | 72 | 72 | 72 | 73 | 73 | 70 | 67 | 68 | 70 | 72 |
| 2 | 76 | 75 | 74 | 73 | 71 | 70 | 71 | 72 | 72 | 72 | 72 | 73 | 69 | 66 | 66 | 68 | 70 |
| 3 | 76 | 76 | 75 | 74 | 73 | 71 | 70 | 71 | 72 | 72 | 72 | 72 | 68 | 65 | 64 | 66 | 68 |
| 4 | 77 | 76 | 76 | 75 | 74 | 73 | 71 | 70 | 71 | 72 | 72 | 72 | 67 | 65 | 63 | 64 | 66 |
| 5 | 78 | 77 | 76 | 76 | 75 | 74 | 73 | 71 | 70 | 71 | 72 | 72 | 68 | 64 | 62 | 61 | 64 |
| 6 | 79 | 78 | 77 | 76 | 76 | 75 | 74 | 73 | 71 | 70 | 71 | 72 | 67 | 64 | 61 | 59 | 60 |
| 7 | 79 | 79 | 78 | 77 | 76 | 76 | 75 | 74 | 73 | 71 | 70 | 71 | 67 | 64 | 61 | 57 | 54 |
| 8 | 80 | 79 | 79 | 78 | 77 | 76 | 76 | 75 | 74 | 73 | 71 | 70 | 66 | 64 | 60 | 58 | 53 |

warranty of $1e - 15$ maximal diameter in the result. This is achieved through gradually increasing the number of bits in precision of real interval field, that results in diameter of the result being at most $1e - 15$, starting over for each combination of indexes $i$ and $j$.

In addition, in Tab. 1 we give the minimum numbers of precision bits for reducing the interval diameter of the resulting values to $10^{-15}$. As we can see, the method is most accurate in the case $i = j$ and worst accurate with highly distinct values of $i$ and $j$.

### 6.1.2 Exact method E2 for calculation of $\beta_{i,j}$ accuracy

Similarly, in Fig. 2 we present the results of the accuracy calculation for the method E2. The corresponding numbers of precision bits needed for achieving $E2(1e - 15)$ are given in Tab. 2.

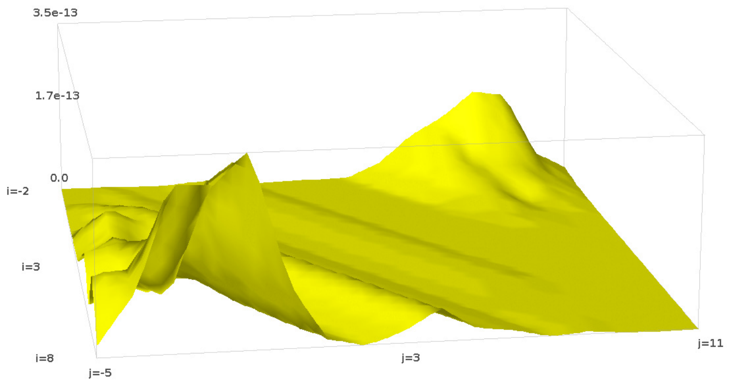

**Figure 2.** Accuracy of method E2 compared with precise E1: $|\beta_{i,j}^{E2} - \beta_{i,j}^{E1(1e-15)}|$, $i \in [-2, 8]$, $j \in [-5, 11]$.

**Table 2.** The number of bits needed to achieve the given accuracy $10^{-15}$ for method E2 ($E2(1e-15)$).

| $i \setminus j$ | −5 | −4 | −3 | −2 | −1 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| −2 | 71 | 63 | 58 | 56 | 57 | 60 | 63 | 65 | 66 | 68 | 69 | 71 | 73 | 74 | 77 | 80 | 87 |
| −1 | 73 | 66 | 62 | 58 | 56 | 57 | 60 | 63 | 65 | 66 | 68 | 69 | 71 | 73 | 75 | 79 | 85 |
| 0 | 75 | 68 | 64 | 61 | 58 | 55 | 57 | 60 | 63 | 65 | 66 | 68 | 70 | 72 | 74 | 77 | 84 |
| 1 | 77 | 70 | 66 | 63 | 61 | 57 | 55 | 57 | 60 | 63 | 65 | 66 | 68 | 70 | 72 | 76 | 83 |
| 2 | 79 | 72 | 68 | 65 | 63 | 60 | 57 | 55 | 57 | 60 | 63 | 65 | 67 | 69 | 71 | 75 | 82 |
| 3 | 80 | 73 | 69 | 67 | 65 | 63 | 60 | 57 | 55 | 57 | 60 | 63 | 65 | 67 | 69 | 73 | 80 |
| 4 | 82 | 75 | 71 | 69 | 67 | 65 | 63 | 60 | 57 | 55 | 57 | 60 | 63 | 65 | 68 | 72 | 79 |
| 5 | 83 | 76 | 72 | 70 | 68 | 66 | 65 | 63 | 60 | 57 | 55 | 57 | 61 | 63 | 66 | 70 | 77 |
| 6 | 84 | 77 | 74 | 72 | 70 | 68 | 66 | 65 | 63 | 60 | 57 | 55 | 58 | 61 | 64 | 68 | 75 |
| 7 | 85 | 79 | 75 | 73 | 71 | 69 | 68 | 66 | 65 | 63 | 60 | 57 | 56 | 58 | 62 | 66 | 73 |
| 8 | 87 | 80 | 77 | 74 | 73 | 71 | 69 | 68 | 66 | 65 | 63 | 60 | 57 | 56 | 58 | 63 | 71 |

### 6.1.3 Approximate integration method (AIM) accuracy

Here, we compare the result of AIM with E1($1e-15$). The result is shown in Fig. 3. We observe that AIM behaves numerically very much in an opposite manner to the exact methods given above.

While the methods E1 and E2 give the most accurate value for $\beta_{i,j}$ in cases when indexes $i$ and $j$ are not too distinct from each other, AIM works best in case of $|i-j|$ being not too small. Due to better performance, we will perform further experiments with E1 from the two exact methods.

Yet we do not know, what would be a reasonable size in $|i-j|$ for switching from one of the exact methods to AIM. This we are checking out together with the overall method verification with the following model problem.

## 6.2 Model problem solution

Consider the following model equation. Let in (1.1) $a(x,y) = y^{-\lambda_0}(1-y)^{-\lambda_1}$, $b \equiv 0$ and $f(x) = x^{\lambda_0}(1-x)^{\lambda_1} - x\log(x) - (1-x)\log(1-x) + 1$; i.e. we are solving the equation

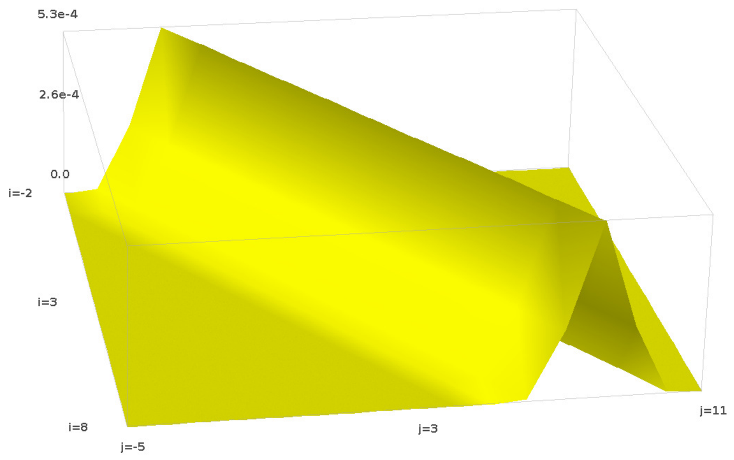

**Figure 3.** Accuracy of AIM compared with precise E1: $|\beta_{i,j}^{AIM} - \beta_{i,j}^{E1(1e-15)}|$, $i \in [-2, 8]$, $j \in [-5, 11]$.

**Table 3.** The best distance of $i$ and $j$ for switching from E1 to AIM in $\beta_{i,j}$ calculation when $m = 4$. The error is the distance of the calculated solution from known solution of the problem.

| $n$ | best $k$ | error (best $k$) | error $(k = 2)$ | error $(k = 4)$ |
|---|---|---|---|---|
| 16 | 1 | 7.71473354e–03 | 7.74274830e–03 | 8.01993108e–03 |
| 32 | 1 | 4.89300912e–04 | 5.06240070e–04 | 6.26558931e–04 |
| 64 | 3 | 2.32888330e–05 | 9.72858080e–05 | 3.56107280e–05 |
| 128 | 83 | 1.70448045e–06 | 4.88493046e–05 | 1.70450322e–06 |
| 256 | 151 | 7.29076350e–08 | 2.44263173e–05 | 7.31555995e–08 |
| 512 | 386 | 1.49411690e–09 | 1.22125301e–05 | 2.92666990e–09 |

$$\begin{aligned} u(x) = \int_0^1 y^{-\lambda_0}(1-y)^{-\lambda_1} \log|x-y|\,dy + x^{\lambda_0}(1-x)^{\lambda_1} \\ -x\log(x) - (1-x)\log(1-x) + 1, \quad 0 \le x \le 1. \end{aligned} \tag{6.1}$$

Integral equation (6.1) has a known solution $u(x) = x^{\lambda_0}(1-x)^{\lambda_1}$, which we use for verifying the method and testing the accuracy of the numerical schemes. In our experiments we have performed tests with different values for $\lambda_0$ and $\lambda_1$ from the range $[0.1, 0.9]$, but the properties of the resulting numerical solution have been quite similar in all cases. The presented results below were calculated with the case $\lambda_0 = \lambda_1 = 0.5$.

In the experiments we have calculated both $\beta_{i,j}^{(E1)}$ and $\beta_{i,j}^{(AIM)}$ and have found the model problem solution with our method with combining

$$\beta_{i,j} = \begin{cases} \beta_{i,j}^{(E1)}, & |i-j| \le k, \\ \beta_{i,j}^{(AIM)}, & |i-j| > k, \end{cases} \qquad k = 0, 1, \dots, n-1$$

for different values of $k = 0, 1, \dots, n-1$. We observe that, for instance, in case of $m = 4$ ($m_0 = 1$ and $m_1 = 3$) the best solution is found in case of $k$ as presented

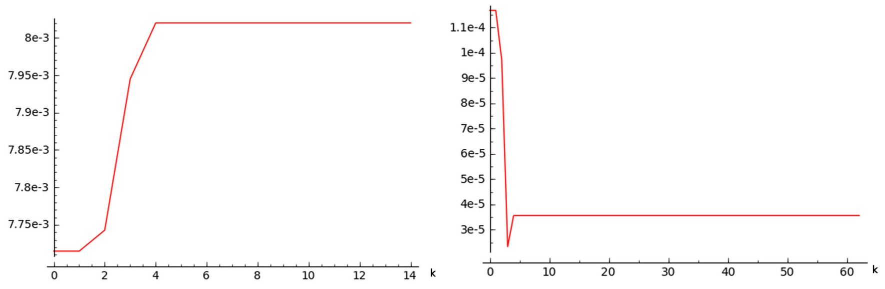

**Figure 4.** Difference from the known solution with different values of $k$ for switching from the exact method E1 to AIM in $\beta_{ij}$ calculations with $m = 4$: $n = 16$ (left), $n = 64$ (right).

in Tab. 3. As we can see, the strategy for choosing the switching value $k$ depends also on $n$. Two example plots of the error depending on $k$ are presented in Fig. 4. But if we use a higher order spline, say $m = 8$ ($m_0 = 3$, $m_1 = 5$), we have found out that with $n = 512$ the minimal error `3.49446166e-13` is achieved for $k = 8$. This means that if one really wants to achieve the absolutely best calculation scheme by combining the given $\beta_{i,j}$ algorithms, it depends on many parameters. Although, usually a general rule of choosing $k$ somewhere between $m$ and $2m$ should give quite good results, it follows from the results presented in Fig. 4, that the error is practically constant for $k > m$. Thus the selection of $k$ seems to be not a very sensitive step for controlling accuracy of the proposed numerical algorithm, at least for moderate $n$.

From the numerical examples we discovered that the exact method E1 worked actually better than we were originally expecting.

For all the numerical experiments we used the *sage* worksheet environment (see e.g. [12]) using *numpy* and *scipy* packages. The source code of the programs together with the test results with interactive graphics can be found at `http://www.ut.ee/~eero/LogSingIntEq/`.

## 7 Conclusion

We examined fully discrete methods for solving linear second kind integral equations containing boundary and logarithmic diagonal singularity in the kernel. The methods are based on the smoothing change of variable and the product quasi-interpolation by smooth splines of order $m$ on the uniform grid of step size $h = 1/n$. The methods are of optimal accuracy order $\mathcal{O}(h^m)$. The matrix form of the methods is given by (4.1)–(4.7). The main difficulty in the application of the methods is the computation of quadrature coefficients $\beta_{i,j}$ defined in (4.7). We presented a simple exact algorithm (5.1)–(5.2) for the computation of $\beta_{i,j}$, which due to some numerical instability can be recommended for small and moderate $|i-j|$, say for $|i-j| \le 2m$. For greater $|i-j|$ we we recommended to use quadratures described in Section 5.3, although our numerical example encourages us to recommend exact formula (5.1)–(5.2) also for much greater $|i-j|$ than $2m$; the preciseness of quadratures rapidly grows as $|i-j|$ increases. In engineer computations with $n^m/m! \le 10^8$, formulae (5.1)–(5.2) can be recommended for all $|i-j|$ using standard double precision arithmetics.

## References

[1] K.E. Atkinson. *The Numerical Solution of Integral Equations of the Second Kind.* Cambridge University Press, 1997.

[2] S. Chandrasekhar. *Radiative Transfer.* Courier Dover Publications, 1960.

[3] W. Hackbusch. *Integral Equations: Theory and Numerical Treatment.* Birkhäuser, 1995.

[4] N.P. Korneichuk. *Splines in Approximation Theory.* Mir, Moscow, 1984. In Russian

[5] R. Kress. *Linear Integral Equations.* Springer, 1999.

[6] E. Leetma and G. Vainikko. Quasi-interpolation by splines on the uniform knot sets. *Math. Model. Anal.*, **12**:107–120, 2007. http://dx.doi.org/10.3846/1392-6292.2007.12.107-120.

[7] G. Monegato and L. Scuderi. High order methods for weakly singular integral equations with nonsmooth input functions. *Math. Comp.*, **67**(224):1493–1515, 1998. http://dx.doi.org/10.1090/S0025-5718-98-01005-9.

[8] A. Pedas and G. Vainikko. Smoothing transformation and piecewise polynomial collocation for weakly singular Volterra integral equations. *Computing*, **73**(3):271–293, 2004. http://dx.doi.org/10.1007/s00607-004-0088-9.

[9] A. Pedas and G. Vainikko. Integral equations with diagonal and boundary singularities of the kernel. *J. Anal. Appl. (ZAA)*, **25**(4):487, 2006.

[10] C. Schneider. Product integration for weakly singular integral equations. *Math. Comp.*, **36**(153):207–213, 1981. http://dx.doi.org/10.1090/S0025-5718-1981-0595053-0.

[11] S.B. Stechkin and Y.N. Subbotin. *Splines in Numerical Mathematics.* Nauka, Moscow, 1976. In Russian

[12] W. Stein. *Sage: Open Source Mathematics Software, (Version 4.5.3).* The Sage Group, 2011. Available from Internet: `http://www.sagemath.org/`.

[13] E. Vainikko and G. Vainikko. A spline product quasi-interpolation method for weakly singular Fredholm integral equations. *SIAM J. Numer. Anal.*, **46**(4):1799–1820, 2008. http://dx.doi.org/10.1137/070693308.

[14] G. Vainikko. GMRES and discrete approximation of operators. *Proc. Estonian Acad. Sci. Phys. Math.*, **53**:124–131, 2004.

[15] G. Vainikko. Error estimates for the cardinal spline interpolation. *J. Anal. Appl. (ZAA)*, **28**(2):205–222, 2009.