Faculty of Mathematical Sciences

University of Twente

University for Technical and Social Sciences

P.O. Box 217
7500 AE Enschede
The Netherlands
Phone: +31-53-4893400
Fax: +31-53-4893114
Email: memo@math.utwente.nl

Memorandum No. 1502

The efficacy of estimates for influence coefficients in wavelet basis

A.A.R. Metselaar and C.R. Traas

September 1999

ISSN 0169-2690

The efficacy of estimates for influence coefficients in wavelet basis

Arnold Metselaar Cees Traas

September 24, 1999

Abstract

We use wavelets for the discretisation of an integral equation. Upper bounds are derived for elements of the resulting matrix. These upper bounds are used to compute only those elements that may exceed a certain threshold. Numerical experiments are presented in which this manner of computing a sparse matrix is compared with computing the matrix in nodal basis, folllowed by a transformation and, for most comparisons, thresholding.

Key words: wavelets – integral equations – sparsening
Mathematics subject classification (1991): 65R20 – 45Exx

1 Introduction

One of the fields in numerical analysis where wavelets are a promising tool is that of boundary integral equations. Choosing wavelets with compact support and vanishing moments to discretise these equations will result in an influence coefficient matrix that is nearly sparse. For a general overview we refer to [4].

Cohen et al. consider a wider range of elliptic equations in [3]. An adaptive wavelet method leads to a sparse coefficient vector. To apply a matrix to such a vector involves only a part of the columns and only elements from these columns are computed but the preprint does not discuss how the large elements within such a column should be selected.

In this paper we reconsider a boundary integral equation. We will show how this equation can be discretised using wavelets and derive upper bounds for most of the influence coefficients. In these bounds the distance between the supports plays a crucial role. We use these upper bounds to compute just those elements of the influence coefficient matrix that may be large. We can also find the idea of using the distance between the 'supports' of the wavelet to make a matrix sparse in [6]. The main goal is to reduce the memory usage such that larger problems can be addressed. Furthermore as we avoid computing matrix elements that we do not need, we expect to save time while filling the matrix, at least for large systems.

We performed some numerical experiments to find out whether these upper bounds give a good indication of the sparsity pattern. We have also looked how much sparsening we get when trying to keep the convergence rate of the discretisation method.

The boundary integral equation we consider originates from a linearised periodic

water-wave problem, with a depth h and a wavelength L.

$$\theta(\vec{x})u(\vec{x}) = \oint_{\xi\in\partial\Omega}\left(u(\vec{\xi})\left(\hat{n}_{\vec{\xi}}\cdot\nabla\right)G(\vec{x},\vec{\xi}) - G(\vec{x},\vec{\xi})\left(\hat{n}_{\vec{\xi}}\cdot\nabla\right)u(\vec{\xi})\right)\mathrm{d}\xi$$
$$\text{for } \vec{x}\in\partial\Omega, \quad (1)$$

where $\Omega = [0, L] \times [-h, 0]$ is a rectangle and

$$\theta(\vec{x}) = \lim_{r\downarrow 0}\frac{\text{area}\left(\{\vec{\xi}\in\Omega|\left\|\vec{\xi}-\vec{x}\right\| < r\}\right)}{2\pi r^2} \quad (2)$$

$$G(\vec{x},\vec{\xi}) = \ln\left\|\vec{x}-\vec{\xi}\right\| \quad (3)$$

with periodic boundary conditions on the sides, $\partial_n u = 0$ for $y = -h$ and prescribed $u(x, 0)$. Due to the mixed boundary conditions, this boundary integral equation is mixed of first and second kind. This boundary integral equation is equivalent to

$$\nabla\cdot\nabla u = 0 \text{ on } \Omega = [0, L] \times [-h, 0], \quad (4)$$

(with the same boundary conditions).

2 Choice of scaling functions and wavelets

2.1 Discretisation

For simplicity we haven chosen a first-degree B-spline as the scaling function $\phi(x)$ for the unknown.

$$\phi(x) = \max(0, 1 - |x|) \quad (5)$$

Its two-scale relation is:

$$\phi(x) = \sum_{n=-1}^{+1} h_n\phi(2x-n) \text{ with } (h_{-1}, h_0, h_1) = (\tfrac{1}{2}, 1, \tfrac{1}{2}) \quad (6)$$

This results in a piecewise affine approximation of the unknown function. The values at the knots are equal to the coefficients with respect to the nodal basis. For our wavelet we take a semi-orthogonal spline (pseudo-)wavelet with two vanishing moments. Semi-orthogonality means that wavelets from different scales are orthogonal but translates need not be. Vanishing moments are desirable because they help making the matrix sparse, see e.g. [2, section 4.4], but they also lead to larger supports and thus to more complexity. The requirement that the wavelet is semi-orthogonal implies that it has at least two vanishing moments. The requirements of semi-orthogonality and minimal support determine ψ up to a multiplicative constant.

We define this wavelet by its two-scale relation:

$$\psi(x) = \sum_{n=-1}^{3} g_n\phi(2x-n),$$
$$\text{with } (g_{-1}, g_0, g_1, g_2, g_3) = \tfrac{1}{10}(+1, -6, +10, -6, +1) \quad (7)$$

These functions can be translated and dilated as usual:

$$\begin{aligned}\phi_{jk}(x) &= 2^{j/2}\phi(2^j x - k)\\ \psi_{jk}(x) &= 2^{j/2}\psi(2^j x - k)\end{aligned} \tag{8}$$

Since the boundary consists of finite edges we have to make a multi-scale basis on a finite interval. By scaling arguments we only need to consider the interval $I = [0,1]$. We simply omit the scaling functions that vanish on I and restrict the remaining ones to the interval. At resolution level J we find the scaling-function basis $\{\phi_{Jk}|k = 0,\ldots,2^J\}$. To get a multi-scale basis we should only retain those wavelets for which the centre of the support is in $[0,1]$. At the endpoints we introduce endpoint-wavelets ψ^L and ψ^R to preserve semi-orthogonality.

The left wavelet, ψ^L, is defined in a way similar to (7), but with $(g_0^L, g_1^L, g_2^L, g_3^L) = \frac{1}{10}(-12,+11,-6,+1)$, ψ^R is the mirror image of ψ^L. We can combine the scaling functions at some coarse scale $1 \le j_0 \le J$ with wavelets to form a multi-scale basis at the same resolution.

$$\{\phi_{j_0 k}|0 \le k \le 2^{j_0}\} \cup \bigcup_{j=j_0}^{J-1} \left(\{\psi_{j,0}^L, \psi_{j,2^j-1}^R\} \cup \{\psi_{jk}|0 < k < 2^j - 1\}\right). \tag{9}$$

Figure 1 shows two scaling function and two wavelets on the interval.

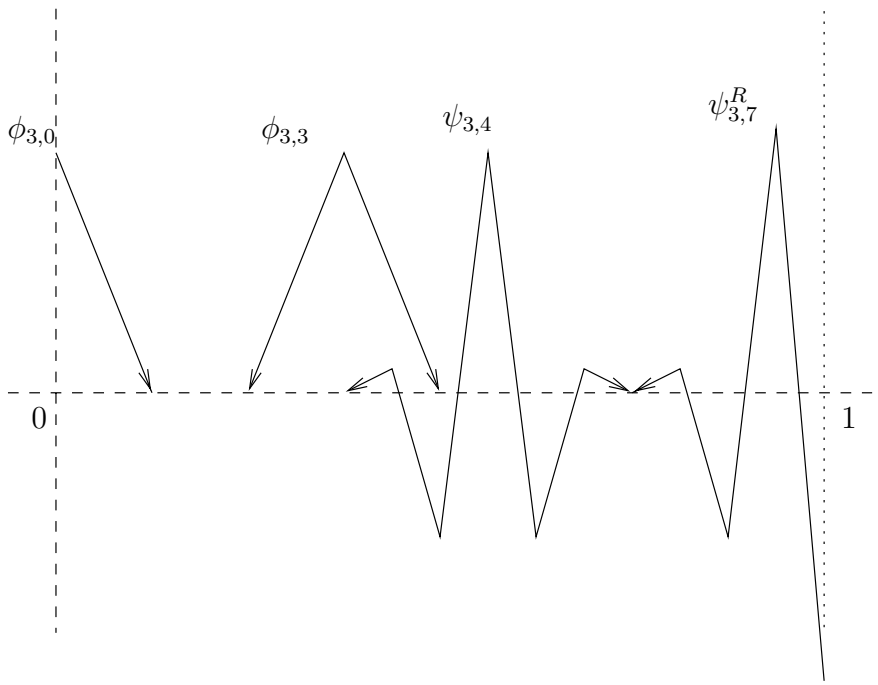

Figure 1: Some scaling functions and wavelets on the interval

2.2 collocation of the integral equation

Our collocation method is based on point collocation. We can reformulate this by saying that we use the δ-function of Dirac as a scaling distribution:

$$\check{\phi}(x) = \delta(x) \tag{10}$$

We define a collocation pseudo-wavelet by a two scale relation that is somewhat simpler than (7):

$$\check{\psi}(x) = \sum_{n=0}^{2} g_n \check{\phi}(2x - n) \text{ with } (g_0, g_1, g_2) = (-1, 2, -1) \tag{11}$$

This $\check{\psi}$ has two vanishing moments but there is no orthogonality between levels. The distributions $\check{\phi}_{jk}$ and $\check{\psi}_{jk}$ are defined in the same way as ϕ_{jk} and ψ_{jk} are defined in (8).

We need to restrict the scaling distributions and collocation wavelets to the interval $I = [0, 1]$. Like before we simply omit those scaling distributions that vanish on I and retain only those collocation wavelets for which the centre of the support is in I. This time there is no need to restrict distributions to the interval or to introduce special endpoint-wavelets. There is, however, a subtlety to the scaling distributions in the endpoints that we will discuss later.

3 Transformation of the matrix-vector equation

One approach to obtaining the matrix of the integral equation in multiscale basis is to transform the matrix in the nodal basis to a matrix in multiscale basis. The advantage of this approach is that the matrix elements are relatively easy to compute in the multiscale basis. The disadvantage is that, generally, the matrix is not sparse in the nodal basis. We denote the system of equations in nodal basis as $Ax = b$.

The vector of unknowns x is divided in chunks, each corresponding to one unknown function u or $\partial_n u$ on one edge of the boundary. These chunks contain the coefficients of the unknown function, here denoted as $f(s)$, in the scaling-function basis:

$$f(s) = \sum_{k=0}^{2^J} c_{Jk} \phi_{Jk} \tag{12}$$

We can also expand the unknown function in the multi-scale basis (9):

$$f(s) = \sum_{k=0}^{2^{j_0}} c_{j_0 k} \phi_{j_0 k} + \sum_{j=1}^{J-1} \sum_{k=0}^{2^j - 1} d_{jk} \psi_{jk}, \tag{13}$$

where we dropped the superscripts for the boundary wavelets. We can also make a vector $\hat{x}$, that is divided in chunks like x, but contains the coefficients of the expansion (13) with $j_0 = 1$. We can reconstruct x from $\hat{x}$ chunk by chunk using the pyramidal algorithm, cf. [1]. If we go from $j_0 = j$ to $j_0 = j + 1$ in expansion (13), the following formulae give the new coefficients:

$$c_{j+1,0} = \tfrac{1}{2}\sqrt{2}\ \left(c_{j0} + g_0^L d_{j0}\right) \tag{14a}$$

$$c_{j+1,1} = \tfrac{1}{2}\sqrt{2}\ \left(\tfrac{1}{2}(c_{j0} + c_{j1}) + g_1^L d_{j0} + g_{-1} d_{j1}\right) \tag{14b}$$

$$c_{j+1,2k} = \tfrac{1}{2}\sqrt{2}\ \left(c_{jk} + g_0 d_{jk} + g_2 d_{j,k-1}\right), \quad \text{for } k = 1 \ldots 2^j - 1 \tag{14c}$$

$$c_{j+1,2k+1} = \tfrac{1}{2}\sqrt{2}\ \Big(\tfrac{1}{2}(c_{jk} + c_{j,k+1}) + g_3 d_{j,k-1} + g_1 d_{jk} + g_{-1} d_{j,k+1}\Big), \quad \text{for } k = 1 \ldots 2^j - 2 \tag{14d}$$

$$c_{j+1,2^{j+1}-1} = \tfrac{1}{2}\sqrt{2}\ \left(\tfrac{1}{2}(c_{j,2^j-1} + c_{j,2^j}) + g_3 d_{j,2^j-2} + g_1^L d_{j,2^j-1}\right) \tag{14e}$$

$$c_{j+1,2^{j+1}} = \tfrac{1}{2}\sqrt{2}\ \left(c_{j,2^j} + g_0^L d_{j,2^j-1}\right) \tag{14f}$$

Let S denote the matrix transforming $\hat{x}$ to x. We can pre-multiply with S by applying the formulae (14) repeatedly on each chunk. A similar scheme for post-multiplying with S can be derived by using transposed matrices.

A row in the original matrix equation corresponds to the integral equation in a certain point. Only the bottom four rows have another origin. The division of the boundary in edges naturally leads to a division of the matrix in blocks. We have to treat each block separately transform our matrix into the collocation wavelet basis.

There are collocation points in the corners and these lie on two edges of the boundary but their integral equation occurs only once in the matrix (otherwise the matrix would be singular). Therefore we need to divide our matrix into overlapping blocks. This is the subtlety we mentioned before.

One column in such a block in the matrix contains the values of a certain function, say f, in the collocation points along the edge to which the block corresponds. We use some scaling to make f a function from $[0, 1]$ to $\mathbb{R}$. If the block has $2^J + 1$ rows then these values are:

$$f(2^{-J}k) = \int \delta(x - 2^J k) f(x) \mathrm{d}x = \int 2^{J/2} \check{\phi}_{Jk} f(x) \mathrm{d}x \text{ with } k = 0..2^J \tag{15}$$

We define the coarse scale and the detail coefficients as:

$$\check{c}_{jk} = \int \check{\phi}_{jk}(x) f(x) \mathrm{d}x \tag{16a}$$

$$\check{d}_{jk} = \int \check{\psi}_{jk}(x) f(x) \mathrm{d}x \tag{16b}$$

The column we have now is (up to a multiplicative constant) a special case of this more general family of representations:

$$\left((\check{c}_{j_0 k})_{k=0\ldots 2^{j_0}}\ ((\check{d}_{jk})_{k=0\ldots 2^j-1})_{j=j_0\ldots J-1}\right) \tag{17}$$

We can go from $j_0 = j + 1$ to $j_0 = j$ in this representation by using the following formulae:

$$\check{d}_{jk} = \sqrt{2}\ \left(-\tfrac{1}{2}\check{c}_{j+1,2k} + \check{c}_{j+1,2k+1} - \tfrac{1}{2}\check{c}_{j+1,2k+2}\right) \quad \text{for } k = 0 \ldots 2^j - 1 \tag{18a}$$

$$\check{c}_{jk} = \sqrt{2}\ \check{c}_{j+1,2k} \quad \text{for } k = 0 \ldots 2^j \tag{18b}$$

We use these formulae repeatedly to go from $j_0 = J$ to $j_0 = 1$ in representation (17). Let T denote the matrix describing this block-wise decomposition. We use the formulae (18) to pre-multiply with T.

The new program uses the following algorithm to compute x, starting from A and b:

- $\hat{A} = TAS$
- $\hat{b} = Tb$
- $\hat{x} = \hat{A}^{-1}\hat{b}$ (with LU-factorisation)
- $x = S\hat{x}$

Note that this algorithm is mathematically equivalent to solving the equation $Ax = b$.

In both A and $\hat{A}$ we can set small elements to zero, where we consider an element a_{ij} to be small if $|a_{ij}| < \delta \max_{k,l}(|a_{kl}|)$. The thresholding parameter δ must be chosen.

4 Direct sparse computation of the matrix

Another approach for obtaining the matrix in multiscale basis is to compute the elements directly in the multiscale basis. We will derive upper bounds for most elements. We can set those elements for which we have a small upper bound to zero without affecting the accuracy of the solution too much. As we don't need to compute these elements this saves computational effort.

The matrix $\hat{A}$ has contributions from three sources. We impose some matching conditions at the vertices, where the edges meet another, this is the first source. These conditions make up only four rows of the matrix and these rows are not computed directly but via the nodal basis and the transformation.

The integral kernels in the integral equation form the most important source of contributions to $\hat{A}$. A general element from this source can be written in the following form:

$$\int_0^1 \int_0^1 \check{g}(t) K(t,s) f(s) ds dt, \tag{19}$$

where $\check{g}$ is a collocation distribution, f is a basis function for the discretisation of the unknown and K is the integral kernel. We handle the periodic boundary conditions on the sides by using certain parts of the vector of unknowns for both sides simultaneously. This means that the elements in some blocks of $\hat{A}$ get two contributions of the form (19). In these cases the two contributions are treated separately and half the threshold is used for them. If f or $\check{g}$ is a scaling function (distribution) then we just compute the element.

We introduce the following notation for the supports of the wavelets:

$$\begin{aligned} I_{jk} &= \operatorname{supp} \psi_{jk} = 2^{-j}[k - \tfrac{1}{2}, k + \tfrac{3}{2}] \\ \check{I}_{jk} &= \operatorname{hull}(\operatorname{supp} \check{\psi}_{jk}) = 2^{-j}[k, k+1] \end{aligned} \tag{20}$$

Wherever we write $\int_{\check{I}_{jk}} (\dots) \check{\psi}_{jk}$, we do not mean to slice the outer δ-functions in the collocation wavelet in half.

We present estimates for the common case that $\check{g}$ is a collocation wavelet and f is an interior (non-endpoint) wavelet. With support-considerations and substitution we can

rewrite such an element as follows:

$$a_{j'k',jk} = \int_0^1 \int_0^1 \check{\psi}_{j'k'}(t) K(t,s) \psi_{jk}(s) ds dt$$
$$= \int_{\check{I}_{j'k'}} \int_{I_{jk}} \check{\psi}_{j'k'}(t) K(t,s) \psi_{jk}(s) \mathrm{d}s \mathrm{d}t$$
$$= \int_{\check{I}_{00}} \int_{I_{00}} 2^{-(j+j')/2} \check{\psi}(t') K\Big(2^{-j'}(k'+t'), 2^{-j}(k+s')\Big) \psi(s') \mathrm{d}s' \mathrm{d}t' \quad (21)$$

To use the vanishing moments of the wavelets, we make a two-dimensional Taylor expansion of the kernel function:

$$K(t,s) = K(t_0,s) + \frac{\partial K}{\partial t}(t_0,s)(t-t_0) + \int_{t_0}^t (t-\tau) \frac{\partial^2 K}{\partial t^2}(\tau,s) \mathrm{d}\tau$$
$$= K(t_0,s_0) + \frac{\partial K}{\partial t}(t_0,s_0)(t-t_0) + \int_{t_0}^t (t-\tau) \frac{\partial^2 K}{\partial t^2}(\tau,s) \mathrm{d}\tau$$
$$+ \frac{\partial}{\partial s} \left(K(t_0,s_0) + \frac{\partial K}{\partial t}(t_0,s_0)(t-t_0) + \int_{t_0}^t (t-\tau) \frac{\partial^2 K}{\partial t^2}(\tau,s) \mathrm{d}\tau \right) s$$
$$+ \int_{s_0}^s (s-\sigma) \frac{\partial^2}{\partial s^2} \left(K(t_0,\sigma) + \frac{\partial K}{\partial t}(t_0,\sigma) t \right) \mathrm{d}\sigma$$
$$+ \int_{s_0}^s (s-\sigma) \frac{\partial^2}{\partial s^2} \left(\int_{t_0}^t (t-\tau) \frac{\partial^2 K}{\partial t^2}(\tau,\sigma) \mathrm{d}\tau \right) \mathrm{d}\sigma \quad (22)$$

This is only valid where K is sufficiently smooth, we discuss how we recognize and treat singularities later.

In principle this gives nine terms but eight of them are cancelled by the vanishing moments. We proceed the computation with the surviving term:

$$|a_{j'k',jk}| = 2^{-(j+j')/2} \Bigg| \int_{\check{I}_{00}} \int_{I_{00}} \check{\psi}(t) \psi(s) \int_{\frac{1}{2}}^s (s-\sigma) \int_{\frac{1}{2}}^t (t-\tau)$$
$$\cdot \frac{\partial^2}{\partial \sigma^2} \left(\frac{\partial^2}{\partial \tau^2} \left(K(2^{-j'}(k+\tau), 2^{-j}(k+\sigma)) \right) \right) \mathrm{d}\tau \mathrm{d}\sigma \mathrm{d}s \mathrm{d}t \Bigg|$$
$$\leq 2^{-(j+j')/2} \int_{\check{I}_{00}} \int_{I_{00}} \left| \check{\psi}(t) \right| |\psi(s)| \int_{\frac{1}{2}}^{2^{-j'}s} (s-\sigma) \int_{\frac{1}{2}}^{2^{-j}t} (t-\tau)$$
$$\cdot \sup_{(x,y) \in \check{I}_{j'k'} \times I_{jk}} \left| 2^{2(j+j')} \partial_y^2 \partial_x^2 K(x,y) \right| \mathrm{d}\tau \mathrm{d}\sigma \mathrm{d}s \mathrm{d}t$$
$$= \int_{\check{I}_{00}} \left| \check{\psi}(t) \right| \int_{\frac{1}{2}}^t (t-\tau) \mathrm{d}\tau \mathrm{d}t \cdot \int_{I_{00}} |\psi(s)| \int_{\frac{1}{2}}^s (s-\sigma) \mathrm{d}\sigma \mathrm{d}s$$
$$\cdot 2^{-5(j+j')/2} \sup_{(t,s) \in \check{I}_{j'k'} \times I_{jk}} \left| \partial_s^2 \partial_t^2 K(t,s) \right| \quad (23)$$

The integrals in this estimate depend only on the choice of wavelets. For our choice

they can be evaluated analytically:

$$\int_{\check{I}_{00}} \left|\check{\psi}(t)\right| \int_{\frac{1}{2}}^{t} (t-\tau)\mathrm{d}\tau\mathrm{d}t \cdot \int_{I_{00}} |\psi(s)| \int_{\frac{1}{2}}^{s} (s-\sigma)\mathrm{d}\sigma\mathrm{d}s = \frac{1}{8} \cdot \frac{1484237}{7024640} \quad (24)$$

When we do similar computations for f being an endpoint-wavelet we arrive at a smaller prefactor in (23). We use the prefactor above whenever f are both wavelets, though it is unsharp if f is an endpoint-wavelet. If f or $\check{g}$ is a scaling function (distribution) then the derivation above becomes invalid by lack of vanishing moments, and we compute the element.

Now we need an estimate for $\left|\partial_t^2 \partial_s^2 K(t,s)\right|$. The variables s and t map to points on the boundary. As mentioned in the previous section, the matrix can be divided in blocks corresponding to combinations of edges. Different blocks have different kernel functions but all of them can be written in one of the following forms:

$$K(t,s) = G(\vec{x}(t), \vec{\xi}(s) = \ln \left\| (\vec{\xi}_0 + s\vec{w}) - (\vec{x}_0 + t\vec{v}) \right\| \quad (25a)$$

or

$$K(t,s) = \partial_{\hat{n}} G(\vec{x}(t), \vec{\xi}(s) = \frac{\partial}{\partial r} \left(\ln \left\| (\vec{\xi}_0 + s\vec{w} + r\hat{n}) - (\vec{x}_0 + t\vec{v}) \right\| \right) \quad (25b)$$

The vectors $(\vec{x}_0, \vec{v}, \vec{\xi_0}$ and $\vec{w}$ describe the edges and $\hat{n}$ is the inward unit normal on the edge $\{\vec{\xi_0} + s\vec{w} | s \in [0,1]\}$. These vectors can differ from block to block.

In the case (25a) we introduce an auxiliary function f

$$f(t,s) = \left\| (\vec{\xi}_0 + s\vec{w}) - (\vec{x}_0 + t\vec{v}) \right\|^2 \quad (26)$$

Now $K(t,s) = \ln(f(t,s))/2$. For convenience we leave out the arguments (t,s) and we denote derivatives with the index notation ($f_u = \frac{\mathrm{d}f}{\mathrm{d}u}$, etc.), this allows us to write:

$$\begin{aligned} \partial_s^2 \partial_t^2 K(t,s) &= G_{ttss} \\ &= \frac{2ff_t^2 f_{ss} - f^2 f_{tt} f_{ss} + 8ff_s f_t f_{st} - 2f^2 (f_{st})^2 + 2ff_s^2 f_{tt} - 6f_s^2 f_t^2}{2f^4} \end{aligned} \quad (27)$$

For this result we used the fact that any third order derivative of f vanishes. Cauchy-Schwartz' theorem provides estimates for some of the derivatives.

$$\begin{aligned} f_s^2 &= \left(2\left((\vec{\xi}_0 + s\vec{w}) - (\vec{x}_0 + t\vec{v})\right) \cdot \vec{w}\right)^2 \leq 2f * 2\left\|\vec{w}\right\|^2 = 2ff_{ss} \\ f_t^2 &= \left(2\left((\vec{\xi}_0 + s\vec{w}) - (\vec{x}_0 + t\vec{v})\right) \cdot -\vec{v}\right)^2 \leq 2f * 2\left\|\vec{v}\right\|^2 = 2ff_{tt} \\ (f_{st})^2 &= (2\vec{v} \cdot \vec{w})^2 \leq 2\left\|\vec{v}\right\|^2 * 2\left\|\vec{w}\right\|^2 = f_{ss} f_{tt} \end{aligned} \quad (28)$$

This leads to the following estimate:

$$G_{ttss} \leq 70 f^{-2} \left\|\vec{v}\right\|^2 \left\|\vec{w}\right\|^2 \quad (29)$$

For the case (25b) we change the function f to a function of three variables:

$$f(t,s,r) = \left\| (\vec{\xi}_0 + s\vec{w} + r\hat{n}) - (\vec{x}_0 + t\vec{v}) \right\|^2 \tag{30}$$

This time we have $K(t,s) = \frac{1}{2}\frac{\mathrm{d}\ln(f(t,s,r))}{\mathrm{d}r}|_{r=0}$. If the lines $\{\vec{\xi}_0 + s\vec{w}\}$ and $\{\vec{x}_0 + t\vec{v}\}$ coincide then $\hat{n} \perp (\vec{\xi}_0 + s\vec{w} - (\vec{x}_0 + t\vec{v})$ and $K(t,s) = 0$. This is the case when we look at the influence coefficients from an edge to itself. Otherwise we have to make a similar derivation to find an upper bound. To make the upper bound somewhat sharper we used the angles between $(\vec{\xi}_0 + s\vec{w}) - (\vec{x}_0 + t\vec{v})$ and $\hat{n}$ and between $\vec{w}$ and $\vec{v}$ to express the derivative. Term-wise estimates lead to the following:

$$G_{nttss} \leq 152 f^{-5/2} \left\|\vec{v}\right\|^2 \left\|\vec{w}\right\|^2 \tag{31}$$

For each matrix element we use the distance d between the corresponding line segments to estimate $f^{-5/2}$ in (31) or f^{-2} in (29) respectively . If this distance is zero then we have no estimate due to the singularity in K and we compute the element.

Fortunately we need not compute the estimate for each element separately. For a fixed combination of edges and levels the estimate only depends on the distance d between the corresponding line segments. Given a threshold for the element we can compute an f_{cr} such that the estimate is smaller than the threshold if and only if $d^2 > f_{cr}$. This is consistent with our treatment of the singularity and allows to skip multiple elements in one step.

Our integral equation is mixed with first and second kind parts. The second kind parts give a important contributions to the matrix. For most elements these contributions are zero because of non-overlapping supports or vanishing moments. The other elements are computed analytically, which is rather simple because the integral of a collocation distribution is equivalent to a finite sum.

5 Numerical experiments

We have compared the two approaches on getting the matrix in multiscale basis. The methods in section 3 produces what we simply call the *full matrix* (after the transform, but before applying the threshold) and the *truncated matrix* (thereafter). Section 4 describes the method to compute what we call the *sparsely computed matrix*.

The first thing we consider here is whether the upper bounds we use in the second approach are adequate for getting the sparsity pattern.

We have tested this in the situation where $L = 2$ and $h = 1$ with 128 boundary elements on the surface and on the bottom and 64 on both sides. This gives a uniform grid along the boundary. To give an idea what the sparsity pattern means we discuss the structure of the matrix. The vector of unknowns has the following division in chunks, the number between brackets is the size of the chunk.

$$x = \begin{pmatrix} \partial_n u(x,0)\ [129] \\ \partial_n u(L,y)\ [65] \\ u(x,-h)\ [129] \\ u(0,y)\ [65] \end{pmatrix} \tag{32}$$

The chunks and the data therein are stored in clockwise direction.

This division in chunks also imposes a division on the columns of the matrix A. The upper 384 rows of A correspond to the integral equation collocated at points along the

boundary, these points coincide with the maxima of the nodal basis functions but we do not count the vertices twice in this case. The collocation points are also ordered in clockwise direction. The wavelet transform orders the the coefficients from coarse to fine, within each level they have their natural ordering The last four rows correspond to equations expressing periodicity and continuity in the vertices.

Thresholding the matrix A with a parameter $\delta = 1.25\ 10^{-5}$ leaves 87% of the matrix elements unaffected, most of the small elements are influence coefficients from the bottom to itself. As we can see in figure 2 the other truncated matrix is sparse. Using the same δ as for A we find that the matrix $\hat{A}$ has only 6.5% non-small elements.

Figure 3 shows the sparsity pattern of the sparsely computed matrix. To compute the value for the threshold we first computed the contributions of the second kind parts of the integral equation and then we multiplied δ with the maximum absolute value of those matrix elements. This gives the same threshold as computing the whole matrix $\hat{A}$. The horizontal and vertical lines come from the elements that involve scaling functions or distributions, for which we did not derive upper bounds. Some of the elements without a small upper bound are actually small, in total 17.2% of the elements is computed.

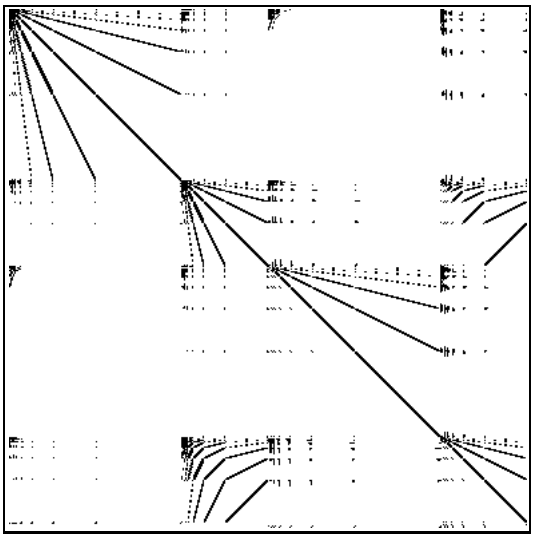

Figure 2: Sparsity pattern of the truncated matrix

Comparing the figures 2 and 3 we see that the sparsely computed matrix has thicker 'fingers' than the truncated one and that not all elements involving scaling functions (distributions) survive in the truncated matrix. Otherwise the matrices are rather similar. The small elements in the sparsely computed matrix, lead to a better approximation of the full matrix than guaranteed by the upper bounds. In this case the maximum absolute difference between an element of full and the sparsely computed matrix is about six times smaller than the threshold.

We have further compared the methods in terms of accuracy of the resulting solution, sparseness of the matrix and the CPU-time needed to compute the matrix.

We have measured the accuracy by taking the boundary conditions from an analytic solution and comparing the results with that solution. The analytic solutions used were $u_a(x,y) = \cos(kx)\cosh(k(y+h))$ and $u_b(x,y) = \sin(kx)\cosh(k(y+h))$, with $k = 2\pi/L$. The geometry of the domain was the same as above.

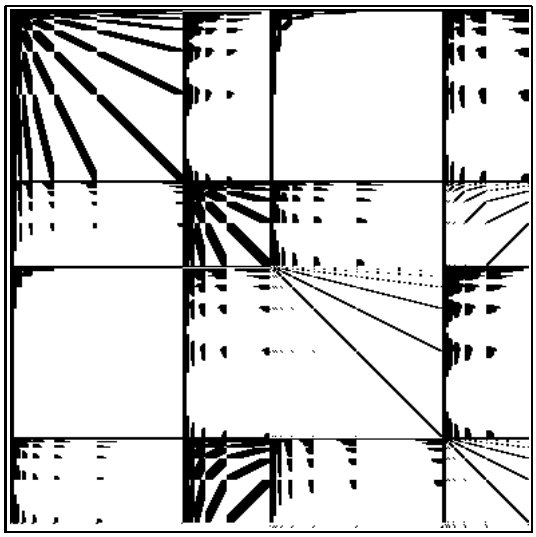

Figure 3: Sparsity pattern of the sparsely computed matrix

To get an impression of how these things scale with the size of the matrix we have considered six different numbers of boundary elements. We started with 48 and then doubled the number each time.

We have tried to choose the threshold parameters such that the error introduced by the thresholding is of the same order as the discretisation error. To do this we have first solved the problem without thresholding to compute the discretisation error. The effect of thresholding can be estimated as follows:

We denote the part of the matrix that is kept with M and the rest with E, $\hat{x}^*$ is the solution without thresholding and δx is the error induced by the thresholding. We have:

$$\begin{aligned} \hat{A}\hat{x}^* = (M+E)\hat{x}^* &= b \\ M(\hat{x}^* + \delta x) &= b \end{aligned}$$

By combining these equations we get:

$$\begin{aligned} \delta x &= M^{-1}E\hat{x}^* \\ \|\delta x\| &\leq \left\|M^{-1}\right\| \|E\| \|\hat{x}^*\| \\ &= \kappa(M) \frac{\|E\|}{\|M\|} \|\hat{x}^*\| \\ \frac{\|E\|}{\|M\|} &\geq \frac{\|\delta x\|}{\|\hat{x}^*\|} \big(\kappa(M)\big)^{-1} \end{aligned}$$

In the process of solving without truncation we also got estimates for the condition number $\kappa(M+E) \approx \kappa(M)$, together with $\|\hat{x}^*\|$ and $\|\delta x\|$ this gives lower bounds for for $\frac{\|E\|}{\|M\|}$.

The threshold parameters were chosen approximately equal to these values, $4 \cdot 10^{-4}$ in the first case and then divided by 8 each time. We came to this rapid decrease of the threshold parameter by combining the $N\sqrt{N}$ -like convergence in the case of u_a and the $N\sqrt{N}$ -like growth of the condition numbers. The condition number of the

untransformed matrix has an $\mathcal{O}(N)$-behaviour, which is probably due to the mixed nature of the boundary integral equation. Von Petersdorff [7] has shown that is is possible to transform problems like (4) to integral equations of the first kind. Furthermore it is known that a Laplace equation with Dirichlet boundary conditions can be transformed in an integral equation of first or second kind, at wish, see e.g. [5]. This makes it plausible that the equation (4) can also be transformed into an integral equation of the second kind, which normally leads to better conditioned matrices, and in this case thus to higher thresholds and sparser matrices.

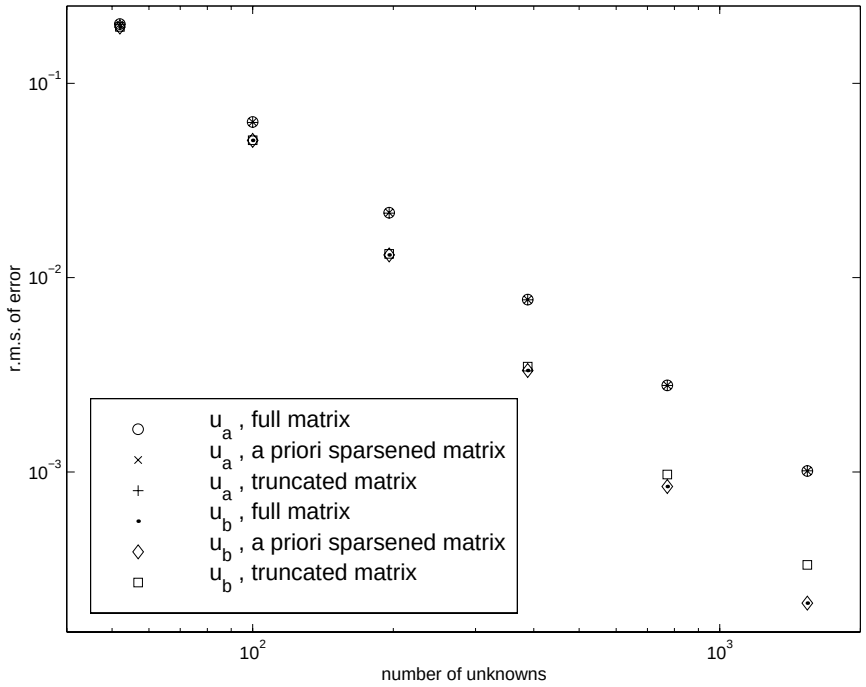

Figure 4: The effect of thresholding on accuracy

Figure 4 shows the errors, measured as the root mean square of the the nodal basis coefficients of the difference between the numerical approximation and the analytic solution. With the thresholds chosen this way the effect of the thresholding is very small. In the case of u_a the symbols representing data points even coincide. The convergence is faster in the case of u_b, but now the truncated matrices give results that are somewhat less accurate. The symbols for the full matrix and the sparsely computed matrix still coincide so apparently the 'extra' elements in the matrix computed based on the estimates improve the accuracy.

Figure 5 shows that when the number of boundary elements is increasing, the fraction of non-zero matrix-elements is indeed going down both in the truncated matrix and the sparsely computed matrix. The ratio of the fractions of nonzero elements in these matrices gives an indication of the sharpness of the upper bounds. This ratio is going down but seems to stabilise at circa 10^{-1}.

We have also measured the times needed to compute the matrices in multiscale basis. These measurements were performed using a Pentium-II processor at 266 MHz. The program we used was compiled with the Gnu `C++`-compiler and ran under Linux.

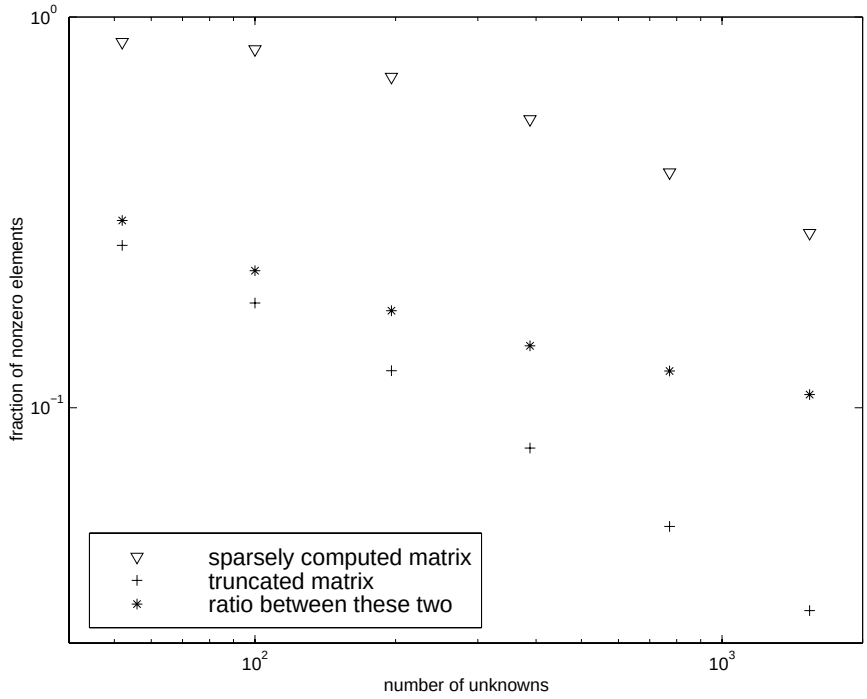

Figure 5: The sparseness in multiscale basis

The results are plotted in figure 6. As we can see using the upper bounds and direct computation of the matrix elements is not faster, for the matrix sizes we considered, but it clearly scales better with an increasing number of unknowns. We would expect that the direct sparse computation becomes faster when the number of unknowns becomes high enough, but we haven't checked that as the full matrix used in the other method would become too large for the memory.

6 Conclusion

We have shown that multiscale bases are indeed useful to sparsen the matrices involved in the boundary integral problem considered. We also showed that the upper bounds we derived using a Taylor expansion and the vanishing moments of the wavelets give a good indication on which elements are going to be small.

This sparsening reduces memory usage and therefore allows to address finer discretisation with the multiscale basis than with the nodal basis. It is more work to compute one matrix element directly in the multiscale than to compute one in the nodal basis, but we expect that for sufficiently large systems sparsely computing the matrix is faster than computing and transforming the matrix in nodal basis.

References

[1] Charles K. Chui. *An Introduction to Wavelets.* Academic Press, 1992.

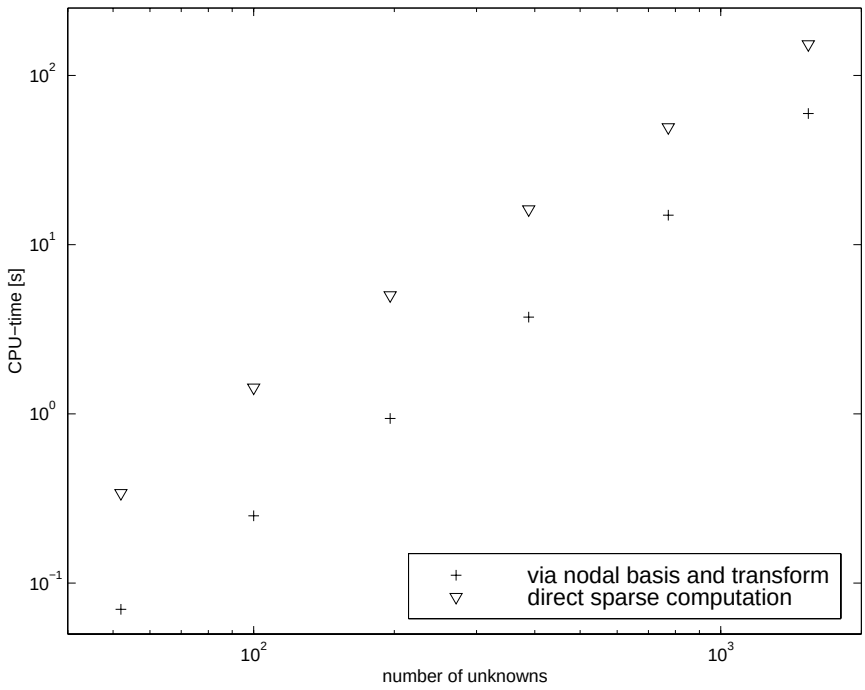

Figure 6: The time needed to compute the matrix in multiscale basis

[2] Albert Cohen. Wavelet methods for numerical analysis. Elaborated lecture notes, distributed after the summerschool on *Multiscale Methods and Wavelets in Numerical Simulation*, organised by CEA - EDF - INRIA, 1997.

[3] Albert Cohen, Wolfgang Dahmen, and Ronald DeVore. Adaptive wavelet methods for elliptic operator equations — convergence rates. preprint, 1998.

[4] Wolfgang Dahmen. Wavelet and multiscale methods for operator equations. In *Acta numerica, 1997*, volume 6 of *Acta Numer.*, pages 55–228. Cambridge Univ. Press, Cambridge, 1997.

[5] P.R. Garabedian. *partial differential equations*. Chelsea Publishing company, New York, N.Y., 1986.

[6] Reinhold Schneider. *Multiskalen- und Wavelet-Matrixkompression*. B. G. Teubner, Stuttgart, 1998. Analysisbasierte Methoden zur effizienten Lösung großer vollbesetzter Gleichungssysteme. [Analysis-based methods for the efficient solution of large nonsparse systems of equations].

[7] T. von Petersdorff. Boundary integral equations for mixed dirichlet, neumann and transmission problems. *Mathematical Methods in the Applied Sciences*, 11:185–213, 1989.