Effective Multiplication by Wavelet Matrix

Václav Finěk, Martina Šimůnková

Departments of Mathematics, Technical University of Liberec

PANM, June 7, 2012

1. Problem and methods
2. 1D problem
3. Higher dimensional problem – design of the implementation
    - Approximate evaluation of the right-hand side
    - Data structures
    - Preconditioning
    - Approximate matrix multiplication

1. Problem and methods
2. 1D problem
3. Higher dimensional problem – design of the implementation
   - Approximate evaluation of the right-hand side
   - Data structures
   - Preconditioning
   - Approximate matrix multiplication

Dirichlet problem

$$-\sum_{i=1}^{d} \frac{\partial^2 u}{\partial x_i^2} + cu = f \qquad \text{on } \Omega = (0,1)^d$$

$$u = 0 \qquad \text{on } \partial\Omega$$

Galerkin method with a wavelet basis

$$\{\psi_i\}_{i=0}^{N-1}$$

Preconditioning

$$Au = f \rightarrow \quad (D^{-\frac{1}{2}} A D^{-\frac{1}{2}})(D^{\frac{1}{2}} u) = D^{-\frac{1}{2}} f$$

with D be a diagonal of A.

Iteration process

Dirichlet problem

$$-\sum_{i=1}^{d} \frac{\partial^2 u}{\partial x_i^2} + cu = f \qquad \text{on } \Omega = (0,1)^d$$

$$u = 0 \qquad \text{on } \partial\Omega$$

Galerkin method with a wavelet basis

$$\{\psi_i\}_{i=0}^{N-1}$$

Preconditioning

$$Au = f \rightarrow \quad (D^{-\frac{1}{2}} A D^{-\frac{1}{2}})(D^{\frac{1}{2}} u) = D^{-\frac{1}{2}} f$$

with D be a diagonal of A.

Iteration process

Dirichlet problem

$$-\sum_{i=1}^{d} \frac{\partial^2 u}{\partial x_i^2} + cu = f \qquad \text{on } \Omega = (0,1)^d$$

$$u = 0 \qquad \text{on } \partial\Omega$$

Galerkin method with a wavelet basis

$$\{\psi_i\}_{i=0}^{N-1}$$

Preconditioning

$$Au = f \rightarrow \quad (D^{-\frac{1}{2}} A D^{-\frac{1}{2}})(D^{\frac{1}{2}} u) = D^{-\frac{1}{2}} f$$

with D be a diagonal of A.

Iteration process

Dirichlet problem

$$-\sum_{i=1}^{d} \frac{\partial^2 u}{\partial x_i^2} + cu = f \qquad \text{on } \Omega = (0,1)^d$$

$$u = 0 \qquad \text{on } \partial\Omega$$

Galerkin method with a wavelet basis

$$\{\psi_i\}_{i=0}^{N-1}$$

Preconditioning

$$Au = f \rightarrow \quad (D^{-\frac{1}{2}} A D^{-\frac{1}{2}})(D^{\frac{1}{2}} u) = D^{-\frac{1}{2}} f$$

with D be a diagonal of A.

Iteration process

Dirichlet problem

$$-\sum_{i=1}^{d}\frac{\partial^2 u}{\partial x_i^2}+cu=f \qquad \text{on } \Omega=(0,1)^d$$

$$u=0 \qquad \text{on } \partial\Omega$$

Galerkin method with a wavelet basis

$$\{\psi_i\}_{i=0}^{N-1}$$

Preconditioning

$$Au=f \rightarrow \quad (D^{-\frac{1}{2}}AD^{-\frac{1}{2}})(D^{\frac{1}{2}}u)=D^{-\frac{1}{2}}f$$

with D be a diagonal of A.

Iteration process

Dirichlet problem

$$-\sum_{i=1}^{d} \frac{\partial^2 u}{\partial x_i^2} + cu = f \qquad \text{on } \Omega = (0,1)^d$$

$$u = 0 \qquad \text{on } \partial\Omega$$

Galerkin method with a wavelet basis

$$\{\psi_i\}_{i=0}^{N-1}$$

Preconditioning

$$Au = f \rightarrow \quad (D^{-\frac{1}{2}} A D^{-\frac{1}{2}})(D^{\frac{1}{2}} u) = D^{-\frac{1}{2}} f$$

with D be a diagonal of A.

Iteration process

$$D^{\frac{1}{2}} u^{n+1} = D^{\frac{1}{2}} u^{n} + \omega \left(D^{-\frac{1}{2}} f - (D^{-\frac{1}{2}} A D^{-\frac{1}{2}})(D^{\frac{1}{2}} u^{n}) \right)$$

Dirichlet problem

$$-\sum_{i=1}^{d} \frac{\partial^2 u}{\partial x_i^2} + cu = f \qquad \text{on } \Omega = (0,1)^d$$

$$u = 0 \qquad \text{on } \partial\Omega$$

Galerkin method with a wavelet basis

$$\{\psi_i\}_{i=0}^{N-1}$$

Preconditioning

$$Au = f \rightarrow \quad (D^{-\frac{1}{2}} A D^{-\frac{1}{2}})(D^{\frac{1}{2}} u) = D^{-\frac{1}{2}} f$$

with D be a diagonal of A.

Iteration process

$$u^{n+1} = u^n + \omega \left(D^{-1} f - D^{-1} A u^n \right)$$

1D wavelet basis of quadratic splines, $N = 2^l$, l is a number of levels

$$\{\psi_i\}_{i=0}^{N-1}$$

higher dimensional basis (anisotrophic wavelet basis)

$$\{\psi_{i_1} \times \psi_{i_2} \times \cdots \times \psi_{i_d}\}_{i_1,i_2,\ldots,i_d=0}^{N-1}$$

$\psi_0, \ldots, \psi_7$ are so called scaling functions

1D wavelet basis of quadratic splines, $N = 2^l$, l is a number of levels

$$\{\psi_i\}_{i=0}^{N-1}$$

higher dimensional basis (anisotrophic wavelet basis)

$$\{\psi_{i_1} \times \psi_{i_2} \times \cdots \times \psi_{i_d}\}_{i_1, i_2, \ldots, i_d = 0}^{N-1}$$

$\psi_0, \ldots, \psi_7$ are so called scaling functions

1D wavelet basis of quadratic splines, $N = 2^l$, l is a number of levels

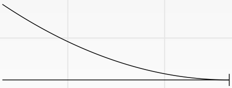
$$\{\psi_i\}_{i=0}^{N-1}$$

higher dimensional basis (anisotrophic wavelet basis)

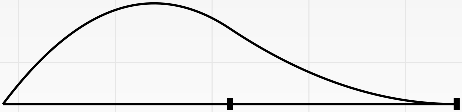
$$\{\psi_{i_1} \times \psi_{i_2} \times \cdots \times \psi_{i_d}\}_{i_1, i_2, \ldots, i_d = 0}^{N-1}$$

$\psi_0, \ldots, \psi_7$ are so called scaling functions

$$\psi_0(x) = \varphi_{sc\_bd}(8x), \quad \psi_7(x) = \varphi_{sc\_bd}(8(1-x))$$



$\frac{1}{2} - x + \frac{1}{2}x^2$



$2x - \frac{3}{2}x^2$ $\quad$ $\frac{1}{2} - (x-1) + \frac{1}{2}(x-1)^2$

1D wavelet basis of quadratic splines, $N = 2^l$, l is a number of levels

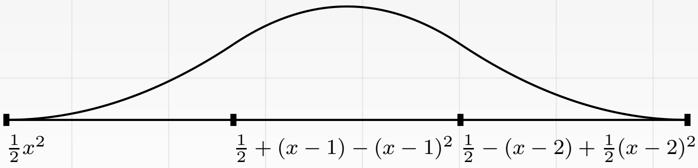
$$\{\psi_i\}_{i=0}^{N-1}$$

higher dimensional basis (anisotrophic wavelet basis)

$$\{\psi_{i_1} \times \psi_{i_2} \times \cdots \times \psi_{i_d}\}_{i_1, i_2, \ldots, i_d = 0}^{N-1}$$

$\psi_0, \ldots, \psi_7$ are so called scaling functions

$$\text{for } i = 1..6 \qquad \psi_i(x) = \varphi_{sc\_in}(8x - i + 1)$$



$\psi_8, \ldots$ are so called wavelets

we show graphs of so called 3/3 biorthogonal wavelets

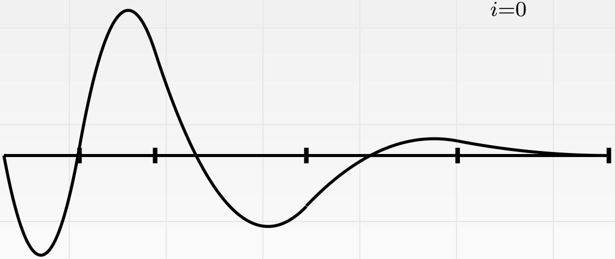
$\psi_8, \ldots$ are so called wavelets

we show graphs of so called 3/3 biorthogonal wavelets

$\psi_8, \ldots$ are so called wavelets

we show graphs of so called 3/3 biorthogonal wavelets

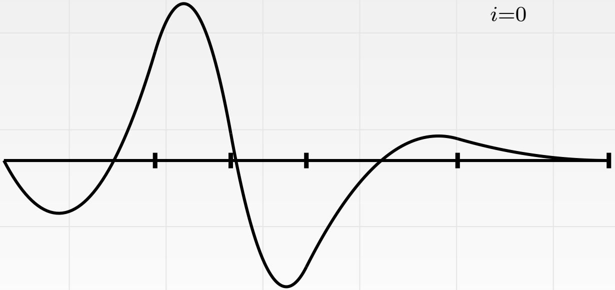
$$\psi_8(x) = a_0 \varphi_{sc\_bd}(16x) + \sum_{i=0}^{7} a_i \varphi_{sc\_in}(16x - i)$$



$\psi_8, \dots$ are so called wavelets

we show graphs of so called 3/3 biorthogonal wavelets

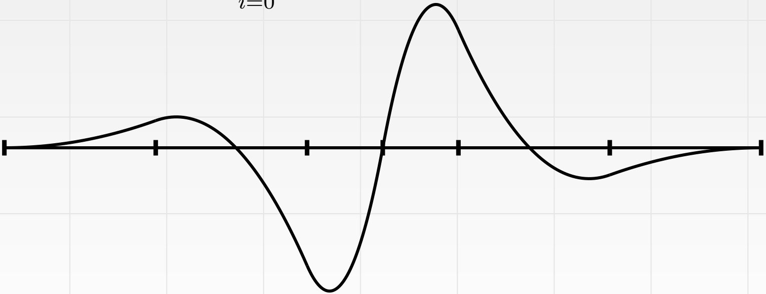
$$\psi_9(x) = b_0\varphi_{sc\_bd}(16x) + \sum_{i=0}^{7} b_i\varphi_{sc\_in}(16x - i)$$



$\psi_8, \ldots$ are so called wavelets

we show graphs of so called 3/3 biorthogonal wavelets

$$\psi_k(x) = \sum_{i=0}^{9} c_i \varphi_{sc\_in}(16x - i - 2k) \quad \text{for } k = 10..13$$



$$
\begin{aligned}
\psi_{14}(x) &= \psi_9(1-x) \\
\psi_{15}(x) &= \psi_8(1-x) \\
\psi_{16}(x) &= \psi_8(2x) \\
\psi_{17}(x) &= \psi_9(2x) \\
\psi_k(x) &= \psi_{10}(2x-k+18) \quad \text{for } k = 18..29 \\
\psi_{30}(x) &= \psi_{14}(2x) \\
\psi_{31}(x) &= \psi_{15}(2x) \\
&\vdots
\end{aligned}
$$

$$
\begin{aligned}
\psi_{14}(x) &= \psi_9(1-x) \\
\psi_{15}(x) &= \psi_8(1-x) \\
\psi_{16}(x) &= \psi_8(2x) \\
\psi_{17}(x) &= \psi_9(2x) \\
\psi_k(x) &= \psi_{10}(2x-k+18) \quad \text{for } k = 18..29 \\
\psi_{30}(x) &= \psi_{14}(2x) \\
\psi_{31}(x) &= \psi_{15}(2x) \\
&\vdots
\end{aligned}
$$

$$
\begin{aligned}
\psi_{14}(x) &= \psi_9(1-x) \\
\psi_{15}(x) &= \psi_8(1-x) \\
\psi_{16}(x) &= \psi_8(2x) \\
\psi_{17}(x) &= \psi_9(2x) \\
\psi_k(x) &= \psi_{10}(2x-k+18) \quad \text{for } k = 18..29 \\
\psi_{30}(x) &= \psi_{14}(2x) \\
\psi_{31}(x) &= \psi_{15}(2x) \\
&\vdots
\end{aligned}
$$

Moment of a function f of n-th order is

$$\int_{\mathbb{R}} x^n f(x)\,\mathrm{d}x$$

For $n = 0, 1, 2,\ i = 8, \ldots$

$$\int_{\mathbb{R}} x^n \psi_i(x)\,\mathrm{d}x = 0.$$

Corollary:

Moment of a function f of n-th order is

$$\int_{\mathbb{R}} x^n f(x)\,\mathrm{d}x$$

For $n = 0, 1, 2,$ $i = 8, \dots$

$$\int_{\mathbb{R}} x^n \psi_i(x)\,\mathrm{d}x = 0.$$

Corollary:

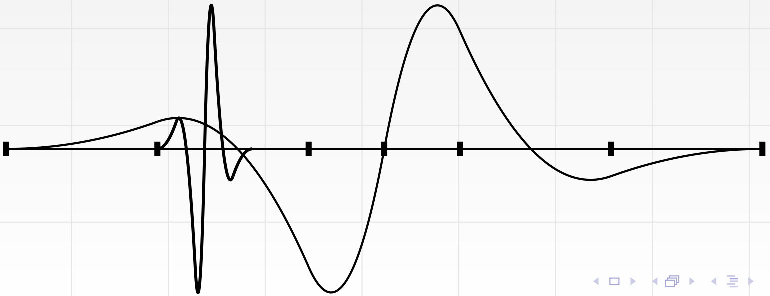
Moment of a function f of n-th order is

$$\int_{\mathbb{R}} x^n f(x)\,\mathrm{d}x$$

For $n = 0, 1, 2,\ i = 8, \ldots$

$$\int_{\mathbb{R}} x^n \psi_i(x)\,\mathrm{d}x = 0.$$

Corollary: $\int \psi_i(x)\psi_j(x)\,\mathrm{d}x = 0.$



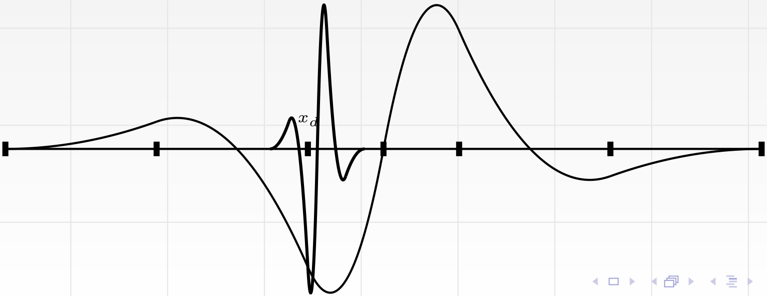
Moment of a function f of n-th order is

$$\int_{\mathbb{R}} x^n f(x)\,\mathrm{d}x$$

For $n = 0, 1, 2, i = 8, \ldots$

$$\int_{\mathbb{R}} x^n \psi_i(x)\,\mathrm{d}x = 0.$$

Corollary: $\int_{\mathbb{R}} \psi_i(x)\psi_j(x)\,\mathrm{d}x = a\int_{[x_d,+\infty)} (x - x_d)^2 \psi_j(x)\,\mathrm{d}x$.



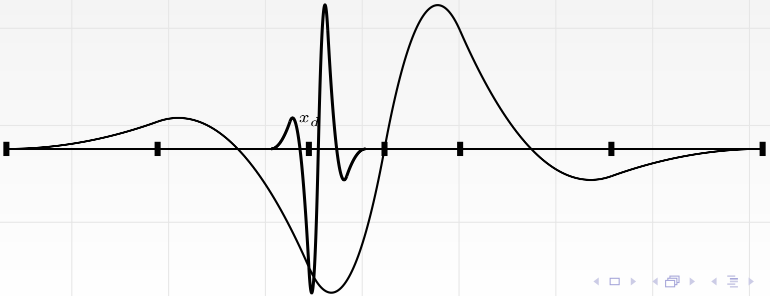
Moment of a function f of n-th order is

$$\int_{\mathbb{R}} x^n f(x)\,\mathrm{d}x$$

For $n = 0, 1, 2, i = 8, \dots$

$$\int_{\mathbb{R}} x^n \psi_i(x)\,\mathrm{d}x = 0.$$

Corollary: $\int_{\mathbb{R}} \psi_i'(x)\psi_j'(x)\,\mathrm{d}x = 2a \int_{[x_d,+\infty)} (x - x_d)\psi_j'(x)\,\mathrm{d}x.$



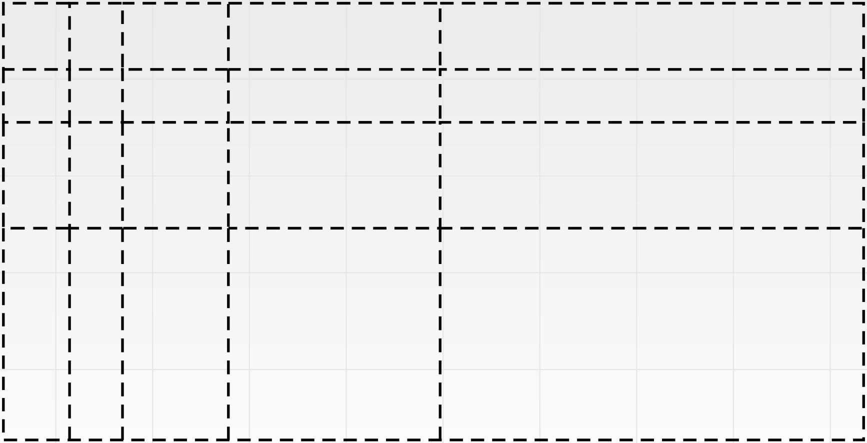
1. Problem and methods
2. 1D problem
3. Higher dimensional problem – design of the implementation
   - Approximate evaluation of the right-hand side
   - Data structures
   - Preconditioning
   - Approximate matrix multiplication

$$d_{ij} = \int_0^1 \psi_i'(x)\psi_j'(x)\,\mathrm{d}x \qquad g_{ij} = \int_0^1 \psi_i(x)\psi_j(x)\,\mathrm{d}x$$

k is the length of a wavelet,

n is the number of discontinuities of a wavelet

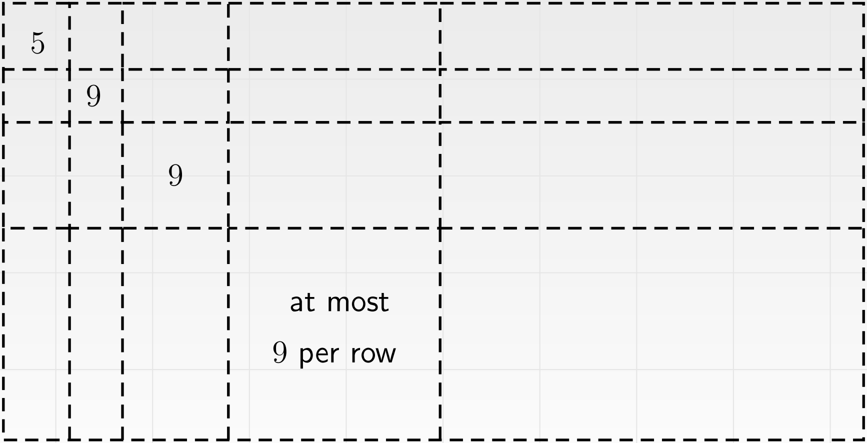
$$d_{ij} = \int_0^1 \psi_i'(x)\psi_j'(x)\,\mathrm{d}x \qquad g_{ij} = \int_0^1 \psi_i(x)\psi_j(x)\,\mathrm{d}x$$



k is the length of a wavelet,

n is the number of discontinuities of a wavelet

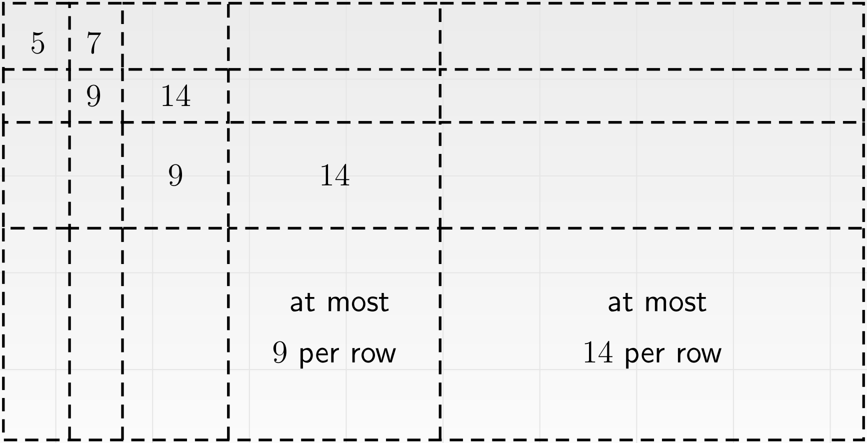
$$d_{ij} = \int_0^1 \psi_i'(x)\psi_j'(x)\,\mathrm{d}x \qquad g_{ij} = \int_0^1 \psi_i(x)\psi_j(x)\,\mathrm{d}x$$



k is the length of a wavelet,

n is the number of discontinuities of a wavelet

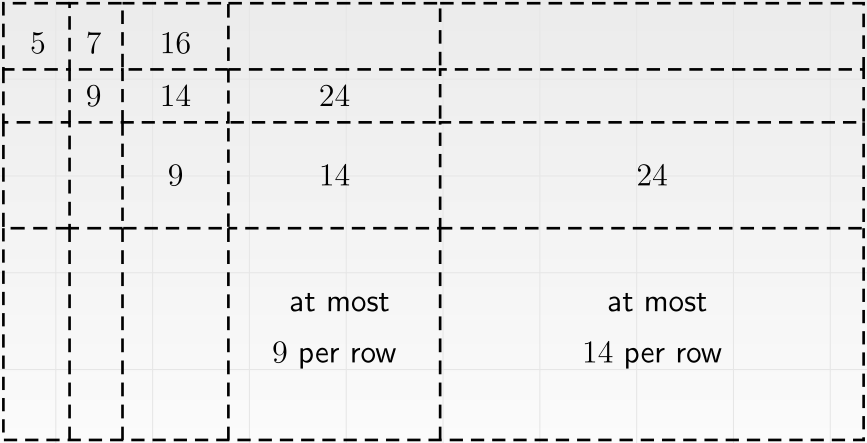
$$d_{ij} = \int_0^1 \psi_i'(x)\psi_j'(x)\,\mathrm{d}x \qquad g_{ij} = \int_0^1 \psi_i(x)\psi_j(x)\,\mathrm{d}x$$



k is the length of a wavelet,

n is the number of discontinuities of a wavelet

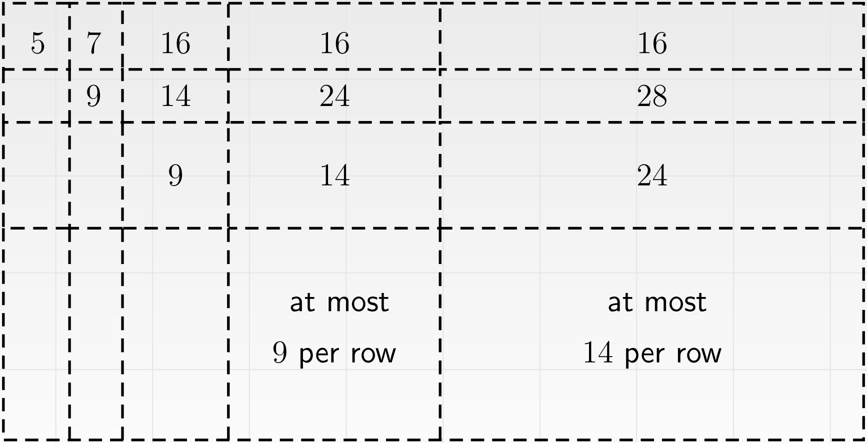
$$d_{ij} = \int_0^1 \psi_i'(x)\psi_j'(x)\,\mathrm{d}x \qquad g_{ij} = \int_0^1 \psi_i(x)\psi_j(x)\,\mathrm{d}x$$



k is the length of a wavelet,

n is the number of discontinuities of a wavelet

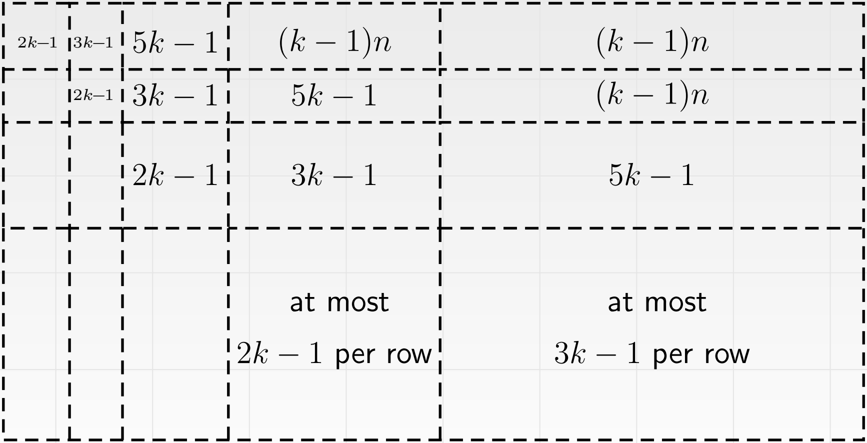
$$d_{ij} = \int_0^1 \psi_i'(x)\psi_j'(x)\,\mathrm{d}x \qquad g_{ij} = \int_0^1 \psi_i(x)\psi_j(x)\,\mathrm{d}x$$



k is the length of a wavelet,

n is the number of discontinuities of a wavelet

$$d_{ij} = \int_0^1 \psi_i'(x)\psi_j'(x)\,\mathrm{d}x \qquad g_{ij} = \int_0^1 \psi_i(x)\psi_j(x)\,\mathrm{d}x$$



k is the length of a wavelet,

n is the number of discontinuities of a wavelet

Theorem. Matrices D and G of the order $N = 2^n$ have at most $0.5(15k - 5 + kl - l)N$ number of nonzero coefficients.

1. Problem and methods
2. 1D problem
3. Higher dimensional problem – design of the implementation
   - Approximate evaluation of the right-hand side
   - Data structures
   - Preconditioning
   - Approximate matrix multiplication

Given the number of levels l and ε as an order of accuracy

for $i_1, \ldots, i_d = 0..(2^l - 1)$ we evaluate

$$f_{i_1 \ldots i_d} = \int_{\mathrm{supp}\psi_{i_1} \times \cdots \times \mathrm{supp}\psi_{i_d}} f(x_1, \ldots, x_d)\psi_{i_1}(x_1) \ldots \psi_{i_d}(x_d)\,\mathrm{d}x$$

by Simpson rule for two divisions

- $k = 2^5$ and $2k$ nodes at each of $\mathrm{supp}\psi_{i_k}$, $k = 1..d$
- estimate error
- repeat $k \to 2k$ until error $< \varepsilon$.

Given the number of levels l and ε as an order of accuracy for $i_1, \dots, i_d = 0..(2^l - 1)$ we evaluate

$$f_{i_1 \dots i_d} = \int_{\mathrm{supp}\psi_{i_1} \times \cdots \times \mathrm{supp}\psi_{i_d}} f(x_1, \dots, x_d) \psi_{i_1}(x_1) \dots \psi_{i_d}(x_d) \, \mathrm{d}x$$

by Simpson rule for two divisions

- $k = 2^5$ and $2k$ nodes at each of $\mathrm{supp}\psi_{i_k}$, $k = 1..d$
- estimate error
- repeat $k \to 2k$ until error $< \varepsilon$.

Given the number of levels l and ε as an order of accuracy for $i_1, \ldots, i_d = 0..(2^l - 1)$ we evaluate

$$f_{i_1 \ldots i_d} = \int_{\mathrm{supp}\psi_{i_1} \times \cdots \times \mathrm{supp}\psi_{i_d}} f(x_1, \ldots, x_d)\psi_{i_1}(x_1) \ldots \psi_{i_d}(x_d)\, \mathrm{d}x$$

by Simpson rule for two divisions

- $k = 2^5$ and $2k$ nodes at each of $\mathrm{supp}\psi_{i_k}$, $k = 1..d$
- estimate error
- repeat $k \to 2k$ until error $< \varepsilon$.

During evaluation of right-hand side we

- calculate $\|f\|_{l_2}$,
- sort f_i by heapsort (with limited heap size and merging them) sort them according their absolute value from smallest store them as a couple (value, index)

Smallest values we put zero

- $sum \leftarrow 0$, $i \leftarrow 0$
- $sum \leftarrow sum + v_i^2$
- while $sum < \varepsilon^2 \|f\|_{l_2}^2$ do $v_i \leftarrow 0$, $i \leftarrow i + 1$, $sum \leftarrow sum + v_i^2$
- while $i \leq maximal\_value$ put c_i to a linked list, $i \leftarrow i + 1$

During evaluation of right-hand side we

- calculate $\|f\|_{l_2}$,
- sort f_i by heapsort (with limited heap size and merging them)
  sort them according their absolute value from smallest
  store them as a couple (value, index)

Smallest values we put zero

- $sum \leftarrow 0$, $i \leftarrow 0$
- $sum \leftarrow sum + v_i^2$
- while $sum < \varepsilon^2 \|f\|_{l_2}^2$
  do $v_i \leftarrow 0$, $i \leftarrow i + 1$, $sum \leftarrow sum + v_i^2$
- while $i \leq maximal\_value$
  put c_i to a linked list, $i \leftarrow i + 1$

During evaluation of right-hand side we

- calculate $\|f\|_{l_2}$,
- sort f_i by heapsort (with limited heap size and merging them)
  sort them according their absolute value from smallest
  store them as a couple (value, index)

Smallest values we put zero

- $sum \leftarrow 0$, $i \leftarrow 0$
- $sum \leftarrow sum + v_i^2$
- while $sum < \varepsilon^2 \|f\|_{l_2}^2$
  do $v_i \leftarrow 0$, $i \leftarrow i + 1$, $sum \leftarrow sum + v_i^2$
- while $i \leq maximal\_value$
  put c_i to a linked list, $i \leftarrow i + 1$

During evaluation of right-hand side we

- calculate $\|f\|_{l_2}$,
- sort f_i by heapsort (with limited heap size and merging them)
  sort them according their absolute value from smallest
  store them as a couple (value, index)

Smallest values we put zero

- $sum \leftarrow 0$, $i \leftarrow 0$
- $sum \leftarrow sum + v_i^2$
- while $sum < \varepsilon^2 \|f\|_{l_2}^2$
  do $v_i \leftarrow 0$, $i \leftarrow i + 1$, $sum \leftarrow sum + v_i^2$
- while $i \leq maximal\_value$
  put c_i to a linked list, $i \leftarrow i + 1$

We use 1D stifness matrix for multiplication. We store it in a relatively small structure (approximately 1000 elements).

Possibilities how to store right-hand side and iterations:

1. To store them in an associative C++ container.
2. To store them in blocks in d-dimensional arays.
   To store pointers to blocks in an aray of size l^d.
   To store coefficients of nonzero elements of an iteration in a linked list.
3. To store couples (index, value) in a smaller structure with index hashed.

We use 1D stifness matrix for multiplication. We store it in a relatively small structure (approximately 1000 elements).

Possibilities how to store right-hand side and iterations:

1. To store them in an associative C++ container.
2. To store them in blocks in d-dimensional arays.
   To store pointers to blocks in an aray of size l^d.
   To store coefficients of nonzero elements of an iteration in a linked list.
3. To store couples (index, value) in a smaller structure with index hashed.

We use 1D stifness matrix for multiplication. We store it in a relatively small structure (approximately 1000 elements).

Possibilities how to store right-hand side and iterations:

1. To store them in an associative C++ container.
2. To store them in blocks in d-dimensional arays.
   To store pointers to blocks in an aray of size l^d.
   To store coefficients of nonzero elements of an iteration in a linked list.
3. To store couples (index, value) in a smaller structure with index hashed.

We use 1D stifness matrix for multiplication. We store it in a relatively small structure (approximately 1000 elements).

Possibilities how to store right-hand side and iterations:

1. To store them in an associative C++ container.
2. To store them in blocks in d-dimensional arays.
   To store pointers to blocks in an aray of size l^d.
   To store coefficients of nonzero elements of an iteration in a linked list.
3. To store couples (index, value) in a smaller structure with index hashed.

We use 1D stifness matrix for multiplication. We store it in a relatively small structure (approximately 1000 elements).

Possibilities how to store right-hand side and iterations:

1. To store them in an associative C++ container.
2. To store them in blocks in d-dimensional arays.
   To store pointers to blocks in an aray of size l^d.
   To store coefficients of nonzero elements of an iteration in a linked list.
3. To store couples (index, value) in a smaller structure with index hashed.

We normalize basis functions

$$\int_0^1 (\psi_i(x))^2 \, \mathrm{d}x = 1$$

Diagonal element of the matrix (we display it for the dimension $d = 3$)

$$D \times G \times G \; + \; G \times D \times G \; + \; G \times G \times D \; + \; c \, G \times G \times G$$

is then

$$a_{i_1 i_1 \ldots i_d i_d} = c + \sum_{k=1}^{d} d_{i_d i_d}$$

We store d_{ii} for first two levels – scaling functions and first level of wavelets (16 elements). Others can be easily calculated – they grows 4 times from one to finer level.

We normalize basis functions

$$\int_0^1 (\psi_i(x))^2 \,\mathrm{d}x = 1$$

Diagonal element of the matrix (we display it for the dimension $d = 3$)

$$D \times G \times G \;+\; G \times D \times G \;+\; G \times G \times D \;+\; c\,G \times G \times G$$

is then

$$a_{i_1 i_1 \ldots i_d i_d} = c + \sum_{k=1}^{d} d_{i_d i_d}$$

We store d_{ii} for first two levels – scaling functions and first level of wavelets (16 elements). Others can be easily calculated – they grows 4 times from one to finer level.

We normalize basis functions

$$\int_0^1 (\psi_i(x))^2 \, \mathrm{d}x = 1$$

Diagonal element of the matrix (we display it for the dimension $d = 3$)

$$D \times G \times G \; + \; G \times D \times G \; + \; G \times G \times D \; + \; c\, G \times G \times G$$

is then

$$a_{i_1 i_1 \ldots i_d i_d} = c + \sum_{k=1}^{d} d_{i_d i_d}$$

We store d_{ii} for first two levels – scaling functions and first level of wavelets (16 elements). Others can be easily calculated – they grows 4 times from one to finer level.

We normalize basis functions

$$\int_0^1 (\psi_i(x))^2 \, \mathrm{d}x = 1$$

Diagonal element of the matrix (we display it for the dimension $d = 3$)

$$D \times G \times G \; + \; G \times D \times G \; + \; G \times G \times D \; + \; c\, G \times G \times G$$

is then

$$a_{i_1 i_1 \ldots i_d i_d} = c + \sum_{k=1}^{d} d_{i_d i_d}$$

We store d_{ii} for first two levels – scaling functions and first level of wavelets (16 elements). Others can be easily calculated – they grows 4 times from one to finer level.

Strategy: given ε as an order of accuracy
multiply an element of a value v with blocks with elements $\geq \frac{\varepsilon}{|v|}$.

We evaluate max to every block (maximal absolute value of its elements) of matrices D and G before the iteration process starts. From them we calculate max for tenzor products of blocks.

Blocks B_{ij} we for now index by one-dimensional indeces $i, j = 0..(l^d - 1)$ and we store $max(B_{ij})$ in two-dimensional $array[l^d][l^d]$ – every row is sorted – $array[i][0]$ is $\max\{max(B_{ij}) : j = 0..(l^d - 1)\}$.

Strategy: given ε as an order of accuracy
multiply an element of a value v with blocks with elements $\geq \frac{\varepsilon}{|v|}$.

We evaluate max to every block (maximal absolute value of its elements) of matrices D and G before the iteration process starts. From them we calculate max for tenzor products of blocks.

Blocks B_{ij} we for now index by one-dimensional indeces $i, j = 0..(l^d - 1)$ and we store $max(B_{ij})$ in two-dimensional $array[l^d][l^d]$ – every row is sorted – $array[i][0]$ is $\max\{max(B_{ij}) : j = 0..(l^d - 1)\}$.

Strategy: given ε as an order of accuracy
multiply an element of a value v with blocks with elements $\geq \frac{\varepsilon}{|v|}$.

We evaluate max to every block (maximal absolute value of its elements) of matrices D and G before the iteration process starts. From them we calculate max for tenzor products of blocks.

Blocks B_{ij} we for now index by one-dimensional indeces $i, j = 0..(l^d - 1)$ and we store $max(B_{ij})$ in two-dimensional $array[l^d][l^d]$ – every row is sorted – $array[i][0]$ is $\max\{max(B_{ij}) : j = 0..(l^d - 1)\}$.