3 Structured algorithms for stuctured matrices

Reading material FFT: Björck Numerical methods in matrix computations, 2014. chapter 1.8.5

3.1 Fast fourier transform

The discrete Fourier transform is defined as follows. Let $f(x)$ be a function, with known function values at $f_k = f(x_k)$, $k = 0, \ldots, N-1$. Then, we can view the transform as an interpolation in the points $x_0, \ldots, x_{N-1}$

$$x_k = \frac{2\pi k}{N} \tag{3.1}$$

Note that we assume that the grid is (equispaced) uniform in (3.1). The techniques for the main algorithms are not trivially transferred to the non-uniform case. Non-uniform FFT is an active research area.

with the exponential as basis functions:

$$f^*(x) = \sum_{j=0}^{N-1} c_j e^{ijx}$$

The coefficients $c_0, \ldots, c_{N-1}$ are given by

$$c_j = \frac{1}{N} \sum_{k=0}^{N-1} f_k e^{-ijx_k}.$$

We now let ω_N be the Nth root of unity

$$\omega_N = e^{-2\pi i/N}. \tag{3.2}$$

The coefficients can also be expressed as

$$c_j = \frac{1}{N} \sum_{k=0}^{N-1} f_k \omega_N^{jk}. \tag{3.3}$$

The naive approach to compute the coefficients c_j would require $\mathcal{O}(N^2)$ operations. The FFT is a procedure which achieves this in $\mathcal{O}(N \log(N))$ operations. To ease the notation we define $y_i = Nc_i$, such that the factor $1/N$ in (3.3) can be removed.

The discrete Fourier transform (DFT) is defined from the DFT matrix.

Definition 3.1.1 (DFT and DFT matrix)**.** *The application of DFT to the vector $f \in \mathbb{C}^N$ is*

$$y = F_N f$$

where

$$F_N = \begin{bmatrix} \omega_N^{00} & \cdots & \omega_N^{0(N-1)} \\ \vdots & & \vdots \\ \omega_N^{(N-1)0} & \cdots & \omega_N^{(N-1)(N-1)} \end{bmatrix} \tag{3.4}$$

and ω_N *is given by* (3.2).

It is easy to verify that

$$\frac{1}{N} F_N^H F_N = I$$

which implies that the inverse DFT transform satisfies

$$f = \frac{1}{N} F_N^H y$$

The hermitian transpose of the matrix is the same as the matrix (3.4), if we replace ω_N with $\bar{\omega}_N = e^{2\pi i/N}$. Therefore, the algorithm developed below can analogously be applied to compute the inverse DFT.

The approach that follows assumes that the size of the matrix is $N = 2^p$. This is not a dramatic assumption in practice, since the matrix size can be increased by adding trivial equations. Let $m = N/2$. We now reorder the equations, by considering odd and even parts separately. For $j = 0, \ldots, N-1$,

$$y_j = \sum_{k_1=0}^{m-1} (\omega_N^2)^{jk_1} f_{2k_1} + \omega_N^j \sum_{k_1=0}^{m-1} (\omega^2)^{jk_1} f_{2k_1+1}. \tag{3.5}$$

Now let $j = \beta m + j_1$, where j_1 is the remainder in the division j/m, such that

$$(\omega_N^2)^{jk_1} = \omega_m^{j_1 k_1}.$$

We can now define

$$\phi_{j_1} := \sum_{k_1=0}^{m-1} \omega_m^{j_1 k_1} f_{2k_1} \quad \text{and} \quad \psi_{j_1} := \sum_{k_1=0}^{m-1} \omega_m^{j_1 k_1} f_{2k_1+1} \tag{3.6}$$

The equations (3.6) is again an FFT-problem of half the size, and the solution can be constructed from (3.7).

and (3.5) can be reduced to $y_j = \phi_{j_1} + \omega_N^j \psi_{j_1}$.

The relations (3.7) are called the butterfly relations.

$$y_{j_1} = \phi_{j_1} + \omega_N^j \psi_{j_1} \tag{3.7a}$$

$$y_{j_1+N/2} = \phi_{j_1} - \omega_N^j \psi_{j_1} \tag{3.7b}$$

Note that the equations (3.6) are two fourier series which allows us to repeat the procedure on a smaller series. The recursion can be repeated until we obtain a small fourier series which can be computed directly.

A general implementation for matrices of size $N = 2^p$ is given in Algorithm 1.

function `fftx(x);`
Input: x
Output: y
Input: The matrix $A \in \mathbb{R}^{n\times n}$ and vector b.

```
n =length (x)
omega = exp(-2i*pi/n);
```

if *rem(n,2)=0* **then**

```
k=(0:n/2-1)'; w=omega.^k;
u=fftx(x(1:2:n-1));
v=w.*fftx(x(2:2:n));
y=[u+v;u-v];
```

else

```
j=0:n-1; k=j';
F=omega.^(k*j);
y=F*x;
```

end

Algorithm 1: Recursive formulation of FFT. Algorithm 1.8.1 in Björck

Reading material: Cooley-Tukey FFT: Björck: pages 194-196.

3.2 *Definitions of Toeplitz like matrices*

The DFT matrix in the previous section belongs to a more general class of structured matrices.

Definition 3.2.1 (Toeplitz matrix)**.** *A square matrix A is a Toeplitz matrix if the diagonals are constant.*

$$\begin{bmatrix} r_0 & r_1 & r_2 & r_3 \\ r_{-1} & r_0 & r_1 & r_2 \\ r_{-2} & r_{-1} & r_0 & r_1 \\ r_{-3} & r_{-2} & r_{-1} & r_0 \end{bmatrix}$$

If additionally the diagonals wrap around, the matrix is called circulant.

Definition 3.2.2 (Circulant matrix)**.** *A square Toeplitz matrix $C(z) \in \mathbb{C}^{n\times n}$ is called circulant if the upper off-diagonal j is equal to diagonal $n-j+1$. The vector $z \in \mathbb{C}^n$ is the elements in first column in the circulant matrix.*

The numbering of the elements in a Toeplitz and a circulant matrix are slightly different. The first column in a circulant matrix has index $0,1,\ldots n-1$.

$$\begin{bmatrix} z_0 & z_3 & z_2 & z_1 \\ z_1 & z_0 & z_3 & z_2 \\ z_2 & z_1 & z_0 & z_3 \\ z_3 & z_2 & z_1 & z_0 \end{bmatrix}$$

A matrix which is a flipped (upside down) Toeplitz matrix is called a Hankel matrix.

Definition 3.2.3 (Hankel matrix)**.** *A square matrix A is a Toeplitz matrix if the anti-diagonals are constant.*

$$\begin{bmatrix} r_{-3} & r_{-2} & r_{-1} & r_0 \\ r_{-2} & r_{-1} & r_0 & r_1 \\ r_{-1} & r_0 & r_1 & r_2 \\ r_0 & r_1 & r_2 & r_3 \end{bmatrix}$$

Note that if T is a Toeplitz matrix, and H is a Hankel matrix, we have the relation

$$R = TJ$$

where J is the flipped identity matrix

$$J = \begin{bmatrix} & & 1 \\ & \iddots & \\ 1 & & \end{bmatrix}.$$

3.3 *Algorithms of Levinson, Durbin and Trench*

Reading material: Golub & Van Loan (GVL), chapter 4.7

The following sequence of algorithms concerns linear system and inverses of Toeplitz matrices. We focus on symmetric matrices and positive definite matrices. Moreover, we assume that the main diagonal consists of ones:

$$T = \begin{bmatrix} 1 & r_1 & \cdots & r_{n-1} \\ r_1 & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & r_1 \\ r_{n-1} & \cdots & r_1 & 1 \end{bmatrix} \tag{3.8}$$

The assumption in (3.8) that the T matrix has ones on the diagonal is not a restriction of generality. If T has $r_0 \neq 1$ on the diagonal, we can define a new Toeplitz matrix $\tilde{T} = T/r_0$ with ones on the diagonal. The linear system $Ty = b$ is transformed to $\tilde{T}\tilde{y} = b$ where $\tilde{y} = r_0 y$. If $r_0 = 0$ the matrix is not positive definite.

and let

$$r = \begin{bmatrix} r_1 \\ \vdots \\ r_{n-1} \\ r_n \end{bmatrix} \in \mathbb{R}^n \tag{3.9}$$

where r_n is an arbitrary number not coupled with the first column of the T-matrix. More precisely, for a Toeplitz matrix T we separate the algorithms into

The equation (3.10) is called the Yule-Walker equations.

- Durbins algorithm: Compute y such that

$$Ty = -r \tag{3.10}$$

  Note that the right-hand side vector in (3.10) is coupled with the first column in T as in (3.8).

- Levinson-Durbin algorithm: Compute y such that

$$Ty = b$$

  for an arbitrary b

- Trench algorithm: Compute T^{-1}.

Persymmetry

The derivations that follow use several times the concept called persymmetry. It is essentially the property that the transpose of the matrix can be obtain by flipping the matrix both upside down and left and right.

Definition 3.3.1 (Persymmetry)**.** *A matrix $B \in \mathbb{R}^{n\times n}$ is called persymmetric if*

$$B^T = E_n B E_n$$

where $E_n \in \mathbb{R}^{n\times n}$ is the flipped identity matrix

$$E_n = \begin{bmatrix} & & 1 \\ & \iddots & \\ 1 & & \end{bmatrix}.$$

Inverse of Toeplitz is persymmetric

The matrix

$$B = \begin{bmatrix} 2 & 1 & 0 \\ 1 & 2 & 1 \\ 0 & 1 & 2 \end{bmatrix}$$

is persymmetric since it is symmetric and if we flip the matrix both in the middle column and the middle row, we obtain B again. The inverse of B is

$$B^{-1} = \frac{1}{4}\begin{bmatrix} 3 & -2 & 1 \\ -2 & 4 & -2 \\ 1 & -2 & 3 \end{bmatrix}.$$

The matrix B^{-1} is also persymmetric, but note that it is not a Toeplitz matrix.

3.3.1 *Durbins algorithm*

The algorithm derivation is based on truncating the Toeplitz matrix and the right-hand side vector in (3.10) and carrying out recursive reasoning. We therefore introduce the notation: T_k is the leading $k \times k$ submatrix of $T = T_n$ and $\bar{r}_k$ is the leading k elements of $r = \bar{r}_n$ as defined in (3.9).

The algorithms is derived by a recursion:

- Given a solution to $T_k \bar{y}_k = -\bar{r}_k$
- compute a solution to $T_{k+1}\bar{y}_{k+1} = -\bar{r}_{k+1}$

Note that the bar in $\bar{r}_k$ denotes that it is a vector, not complex conjugate. This subsection does not contain complex numbers.

The algorithm can be initiated by noting that solution to $T_1\bar{y}_1 = -\bar{r}_1$ is given by

$$\bar{y}_1 = -r_1/T_1 = -r_1.$$

In the recursive reasoning, we use the vectors $\bar{y}_k$. Note that in contrast to $\bar{r}_k$ the vector $\bar{y}_k$ is not a subvector of $\bar{y}_{k+1}$.

In order to relate T_k with T_{k+1} we use the E_k operator from the persymmetry definition Definition 3.3.1. Due to the fact that T_k is the leading submatrix of T_{k+1}, we see that

$$T_{k+1} = \begin{bmatrix} T_k & E_k\bar{r}_k \\ \bar{r}_k^T E_k & 1 \end{bmatrix}.$$

Property of E_k:

$$E_k\bar{r}_k = \begin{bmatrix} & & 1 \\ & \iddots & \\ 1 & & \end{bmatrix}\begin{bmatrix} r_1 \\ \vdots \\ r_k \end{bmatrix} = \begin{bmatrix} r_k \\ \vdots \\ r_1 \end{bmatrix}$$

Consider now the Yule-Walker system in this notation

$$-\begin{bmatrix} \bar{r}_k \\ r_{k+1} \end{bmatrix} = T_{k+1}\bar{y}_{k+1} = \begin{bmatrix} T_k & E_k\bar{r}_k \\ \bar{r}_k^T E_k & 1 \end{bmatrix}\begin{bmatrix} z \\ \alpha \end{bmatrix}. \tag{3.11}$$

Temporary notation in (3.11):

$$\tilde{y}_{k+1} = \begin{bmatrix} z \\ \alpha \end{bmatrix}.$$

The first block row gives us

$$T_k z + \alpha E_k \bar{r}_k = -\bar{r}_k$$

which we can solve for T_k

$$\begin{aligned} z &= T_k^{-1}(-\alpha E_k\bar{r}_k - \bar{r}_k) \\ &= \bar{y}_k - \alpha T_k^{-1} E_k \bar{r}_k. \end{aligned}$$

If we also use that T_k^{-1} is persymmetric we obtain that

$$z = \bar{y}_k + \alpha E_k \bar{r}_k. \tag{3.12}$$

The last row in (3.11) is

$$\bar{r}_k^T E_k z + \alpha = -r_{k+1}.$$

We can solve this by combining with the formula for z and obtain the following relation

$$\alpha(1 + \bar{r}_k^T \bar{y}_k) = -r_{k+1} - \bar{r}_k^T E_k \bar{y}_k$$

such that we have an explicit formula for α:

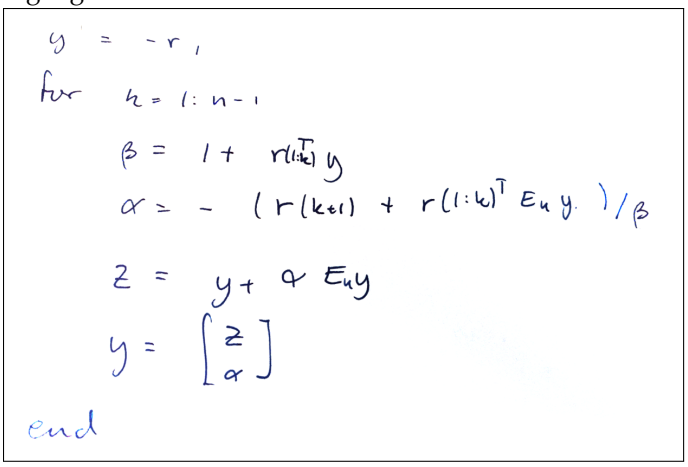
$$\alpha = -\frac{r_{k+1} + \bar{r}_k^T E_k \bar{y}_k}{1 + \bar{r}_k^T \bar{y}_k} \tag{3.13}$$

The denominator in (3.13) cannot be zero. This is due to the assumption that T is positive definite.

The formulas (3.12) and (3.13) can be combined into a constructive formula for $\bar{y}_{k+1}$.

By repeating the procedure, for $k = 1, \ldots, n-1$ we obtain the following algorithm:

The corresponding algorithm in GVL: Algorithm 4.7.1 which is reduced from $3n^2$ to $2n^2$ operations.



If the operations with E_k are optimized with $E_k q = q(k:-1:1)$, all operations are with vectors of length at most n. Since we carry out $n-1$ iterations, it has complexity $\mathcal{O}(n^2)$.

3.3.2 *Levinson-Durbin*

See GV section 4.7.4

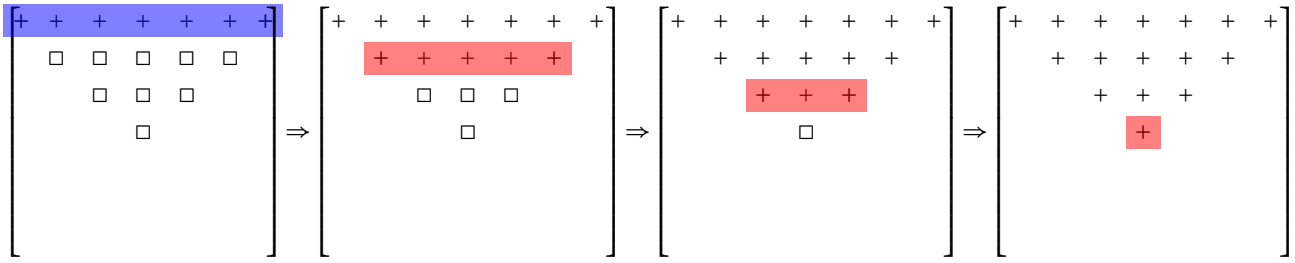
3.3.3 *Trench*

See derivation in GVL section 4.7.5

The derivation leads up to the formula

$$b_{i,j} = b_{i-1,j-1} + \frac{1}{\gamma}(v_{n+1-j}v_{n+1-i} - v_{i-1}v_{j-1}) \tag{3.14}$$

The procedure the algorithm computes elements can be visualizes as follows, where + denotes a computed element and □ a not yet computed element. Note that if we know the elements in the wedge shape, we know the entire matrix due to symmetry and persymmetry. The blue elements are computed directly from the Yule-Walker solution and the red elements are computed from the relation (3.14)



3.4 *Circulant matrices via FFT*

Reading material: Golub & Van Loan, chapter 4.8

Similar to the previous section, we will now see procedures how to compute Cb and $C^{-1}b$ where C is a circulant matrix. It turns out that this can be done via a transformation which allows us to use FFT. More precisely, we now show that one can explicitly diagonalize C with the DFT matrix as eigenvector matrix.

First consider the downshift matrix, which in itself is a circulant matrix

$$D_n = \begin{bmatrix} 0 & & & 1 \\ 1 & & & \\ & \ddots & & \\ & & 1 & 0 \end{bmatrix}.$$

The powers of this matrix can be easily be computed explicitly. For example for $n = 4$ we have

$$D_n^2 = \begin{bmatrix} 0 & & & 1 \\ 1 & 0 & & \\ & 1 & 0 & \\ & & 1 & 0 \end{bmatrix} \begin{bmatrix} 0 & & & 1 \\ 1 & 0 & & \\ & 1 & 0 & \\ & & 1 & 0 \end{bmatrix} = \begin{bmatrix} 0 & & 1 & 0 \\ 0 & 0 & & 1 \\ 1 & 0 & 0 & \\ & 1 & 0 & 0 \end{bmatrix}.$$

and

$$D_n^3 = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \end{bmatrix}.$$

The powers of D_n correspond to one diagonal in a circulant matrix. Therefore we can now view the powers of D_n as a basis of the space

of circulant matrices, where the coefficients are the diagonal elements. More precisely, we note that

$$\begin{bmatrix} z_0 & z_3 & z_2 & z_1 \\ z_1 & z_0 & z_3 & z_2 \\ z_2 & z_1 & z_0 & z_3 \\ z_3 & z_2 & z_1 & z_0 \end{bmatrix} = z_0 I + z_1 D_n + z_2 D_n^2 + \cdots z_{n-1} D_n^{n-1}. \tag{3.15}$$

Our approach is based on diagonalization. Any circulant matrix $C(z)$ can be expressed as

$$C(z) = V \Lambda V^{-1} \tag{3.16}$$

In equation (3.16) we assume that $C(z)$ is diagonalizable. Not all matrices are diagonalizable, for instance not the matrix

$$\begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix}$$

but it turns out all circulant matrices are diagonalizable as a consequence of Theorem 3.4.2.

where V is a matrix consisting of eigenvectors (not necesserily normalized). Due to (3.15), this implies in turn that

$$\Lambda = V^{-1} \left(\sum_{k=0}^{n-1} z_k D_n^k \right) V = \sum_{k=0}^{n-1} (V D_n V^{-1})^k.$$

We will now show that:

- the V matrix that diagonalizes $C(z)$ also diagonalizes D_n; and
- $V = F_n$ where F_n is the DFT-matrix.

The following result characterizes the eigenvalues and eigenvectors of the D_n matrix.

The lemma is given in GVL: Lemma 4.8.1.

Lemma 3.4.1. *If $V = F_n$, then*

(a) $V^{-1} D_n V = G$ is a diagonal matrix, and

(b) the diagonal elements of G are

$$\lambda_{j+1} = \bar{\omega}_n^j.$$

for $j = 0, \ldots, n-1$.

Recall the definition of $\omega_n := \exp(-2\pi i/n)$ in the description of FFT.

Proof. First consider the product, where we explicitly write out column $j+1$ in the F_n matrix,

$$D_n F_n = \begin{bmatrix} 0 & & & 1 \\ 1 & & & \\ & \ddots & & \\ & & 1 & 0 \end{bmatrix} \begin{bmatrix} \omega_n^{0\cdot 0} & \cdots & \omega_n^{0\cdot j} & \cdots & \omega_n^{0\cdots(n-1)} \\ \vdots & & \vdots & & \\ \omega_n^{(n-1)\cdot 0} & \cdots & \omega_n^{(n-1)\cdot j} & \cdots & \omega_n^{(n-1)(n-1)} \end{bmatrix}$$

Therefore column $j+1$ can be expressed as:

$$D_n F_n(:, j+1) = D_n \begin{bmatrix} \omega_n^{0\cdot j} \\ \omega_n^{1\cdot j} \\ \vdots \\ \omega_n^{(n-1)\cdot j} \end{bmatrix} = \begin{bmatrix} \omega_n^{(n-1)\cdot j} \\ \omega_n^{0\cdot j} \\ \vdots \\ \omega_n^{(n-2)\cdot j} \end{bmatrix} = \bar{\omega}_n^j \begin{bmatrix} 1 \\ \omega_n^{1\cdot j} \\ \vdots \\ \omega_n^{(n-1)\cdot j} \end{bmatrix}. \tag{3.17}$$

In the last step we factorized $\bar{\omega}_n^j = \omega_n^{-j}$ and used that ω_n is an nth root of unity, such that $\omega_n^{n\cdot j} = 1$. In order to show that $D_n F_n = F_n G$ we now consider column $j+1$ of $F_n G$. A direct computation yelds

$$F_n G(:, j+1) = \bar{\omega}_n^j F_n(:, j+1) = \bar{\omega}_n^j \begin{bmatrix} 1 \\ \omega_n^{1\cdot j} \\ \vdots \\ \omega_n^{(n-1)\cdot j} \end{bmatrix}. \tag{3.18}$$

We note that (3.18) and (3.17) are equal. Since the column $j+1$ was arbitrary, the proof holds for all columns. □

The lemma can now be used to explicitly determine the diagonalization of $C(z)$.

Theorem 3.4.2. *Suppose $C(z)$ is a circulant matrix. Let $V = F_n$ and the vector $\lambda = \bar{F}_n z \in \mathbb{C}^n$. Then,*

$$V^{-1} C(z) V = \Lambda := \operatorname{diag}(\lambda_1, \ldots, \lambda_n).$$

Proof. Let f be the second column of $\bar{F}_n$:

$$\begin{bmatrix} 1 \\ \bar{\omega}_n^1 \\ \vdots \\ \bar{\omega}_n^{n-1} \end{bmatrix}.$$

Column $k+1$ of $\bar{F}_n$ is now the componentwise power k of f, in formulas easily expressed in terms of powers of diagonal matrices

$$\operatorname{diag}(\bar{F}_n(:, k+1)) = \operatorname{diag}(f)^k.$$

We define $G = \operatorname{diag}(f)$. From the lemma that diagonalizes D_n (Lemma 3.4.1), we conclude that

$$V^{-1} C(z) V = \sum_{k=0}^{n-1} z_k G^k = \operatorname{diag}(\bar{F} z)$$

since

$$\sum_{k=0}^{n-1} z_k \operatorname{diag}(f)^k = \operatorname{diag}\left(\sum_{k=0}^{n-1} z_k \bar{F}(:, k+1) \right).$$

□

The above theorem implies that

$$C(z) = F_n \Lambda F_n^{-1}$$

and the action of $C(z)$ on a vector b is

$$g = C(z) b = F_n \Lambda F_n^{-1} b.$$

Remark 3.4.3 (Action of circulant matrix via FFT)**.** We can compute the action of a circulant matrix by

- Computing $c = F_n^{-1} b$ with inverse FFT applied to b: $\mathcal{O}(n \log(n))$
- Computing $\lambda = \bar{F}_n z \in \mathbb{C}^n$ with FFT applied to z: $\mathcal{O}(n \log(n))$
- Multiplying c with a diagonal matrix: $d = \Lambda c$: $\mathcal{O}(n)$
- Computing $g = F_n c$ with FFT applied to c: $\mathcal{O}(n \log(n))$

The inverse action $C(z)^{-1} b$ can be done completely analogously by changing Λc to $\Lambda^{-1} c$.

Action of Toeplitz matrices via circulant matrices

The above technique explicitly uses the circulant structure of the matrix. The same technique can not be directly applied to a general Toeplitz matrix. However, the action corresponding to a Toeplitz matrix can be extended to a linear system which instead involves a circulant matrix.

Consider a toeplitz matrix T and a given vector b, we want to compute

$$c = Tb.$$

We extend this linear system by adding n rows to the b vector and we also extend the matrix. We define Q such that we obtain the following structure

$$\begin{bmatrix} r_0 & r_1 & r_2 & r_3 & r_0 & r_{-3} & r_{-2} & r_{-1} \\ r_{-1} & r_0 & r_1 & r_2 & r_3 & r_0 & r_{-3} & r_{-2} \\ r_{-2} & r_{-1} & r_0 & r_1 & r_2 & r_3 & r_0 & r_{-3} \\ r_{-3} & r_{-2} & r_{-1} & r_0 & r_1 & r_2 & r_3 & r_0 \\ r_0 & r_{-3} & r_{-2} & r_{-1} & r_0 & r_1 & r_2 & r_3 \\ r_3 & r_0 & r_{-3} & r_{-2} & r_{-1} & r_0 & r_1 & r_2 \\ r_2 & r_3 & r_0 & r_{-3} & r_{-2} & r_{-1} & r_0 & r_1 \\ r_1 & r_2 & r_3 & r_0 & r_{-3} & r_{-2} & r_{-1} & r_0 \end{bmatrix} = \begin{bmatrix} T & Q \\ Q & T \end{bmatrix}$$

The matrix Q is selected such that

- Q is a Toeplitz matrix;
- the diagonals of Q match the shifted diagonals of T such that

$$C(z) := \begin{bmatrix} T & Q \\ Q & T \end{bmatrix} \tag{3.19}$$

is a circulant matrix.

The result of matrix vector product Tb is now the first n rows in the product with the bigger matrix, due to the identity

$$\begin{bmatrix} T & Q \\ Q & T \end{bmatrix} \begin{bmatrix} b \\ 0 \end{bmatrix} = \begin{bmatrix} Tb \\ Qb \end{bmatrix}$$

We use the FFT procedure for circulant matrices, by invoking it on the vector

$$\begin{bmatrix} b \\ 0 \end{bmatrix}$$

and extracting only the first n rows. The last n rows are ignored.

The extended matrix (3.19) can be explicitly expressed with a z vector built from the r-elements,

$$z = \begin{bmatrix} r_0 \\ \vdots \\ r_{-(n-1)} \\ r_0 \\ r_{n-1} \\ \vdots \\ r_1 \end{bmatrix}.$$

3.5 *Hierarchical and semi-separable matrices*

3.5.1 *Semi-separable matrices*

Reading material: Chapter 5 in the book of PGM "Fast direct solvers for elliptic PDEs".

Many matrices stemming from data have some form of low-rank structure. The semi-separable is a low-rank structure defined only in the lower triangular part.

Definition 3.5.1. *A symmetric matrix is called semi-separable (of order p), if all submatrices taken out of the low triangular part have rank $\leq p$.*

Example: Banded matrices is semi-separable with order same as the band

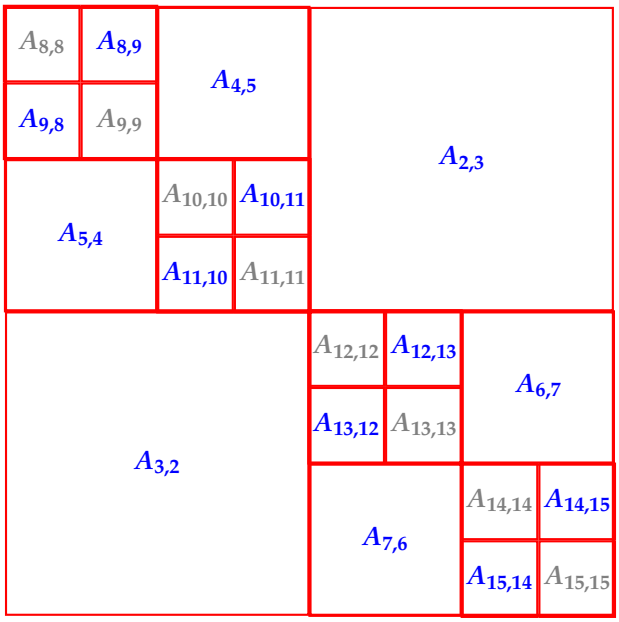
Example: The inverse of a tridiagonal matrix is semi-separable

Example: Electrostatics

Properties: Inverse of a semi-separable matrix is again semi-separable.

3.5.2 *Hierarchical semi-separable matrices*

We will now the exploit the properties of semi-separable matrices, considering a particular type of partitioning of the matrix.



The system for the numbering of the blocks is explained by the recursive subdivision formulation in Section 3.5.3.

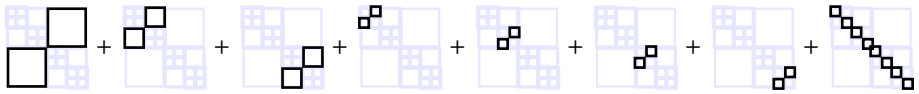
What follows will be based on two observations.

1. Due to the property that the matrix is semi-separable, all the matrices below the diagonal will be of low-rank. Therefore, the blocks in the the nodes marked in blue above are of low-rank. We assume those blocks are given in terms of SVD-factorizations:

$$A_{\sigma,\tau} = U_\sigma \tilde{A}_{\sigma,\tau} V_\tau^T.$$

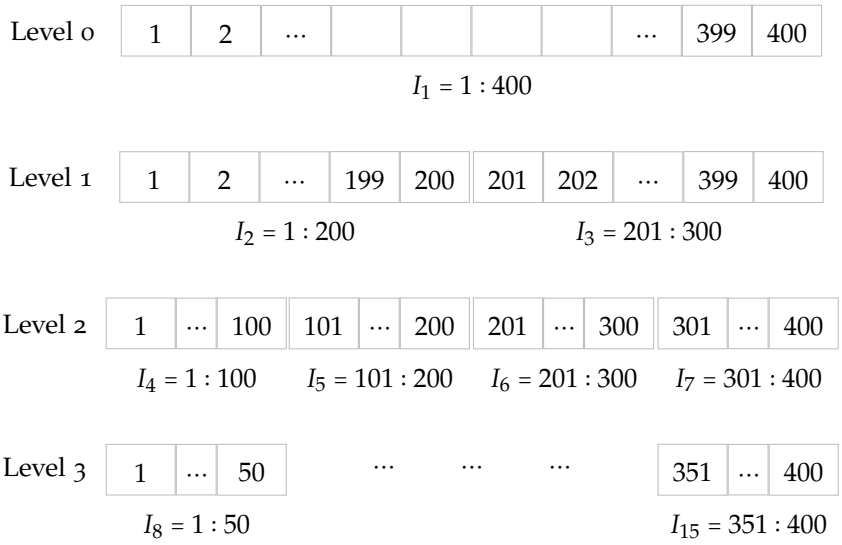
The matrix $\tilde{A}_{\sigma,\tau}$ in the middle of the factorization can be (but does not have to be) a diagonal matrix and is called the **core** of the block.

2. In order to implement operations with this matrix, we separate the matrix into a sum of blocks. We will loop through all the blocks of the matrix in the following way. Note that the final matrix is just a diagonal matrix.



3.5.3 *Recursive index vector formulation*

In order to exploit the second observation, we will formulate it in a recursive way in terms of index vectors. First, define the index vectors $I_1, I_2, \ldots, I_{15}$ as follows:

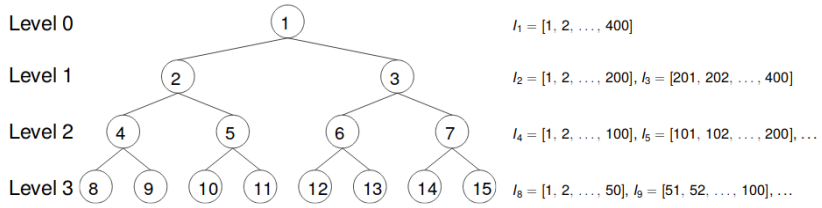


With this notation, we can denote the blocks of the matrix as

$$A_{\sigma,\tau} = A(I_\sigma, I_\tau), \quad \sigma, \tau = 1, \ldots, 15.$$

Algorithms for this matrix are naturally formulated in a recursive way, so we draw the corresponding tree:

Credit for the image: PGM. Will be replaced.



3.5.4 *A recursive algorithm for the matrix vector product Ac*

The above tools can be used to carry out several operations associated with A. Most importantly, we now show how it can be combined into an algorithm to compute the matrix times a vector c.

** Matrix-vector multiply using tree and index vectors **

loop τ is a node in the tree
  if τ is a leaf
    $b(I_\tau) = b(I_\tau) + A_{\tau,\tau} c(I_\tau)$
  else
    Let σ_1 and σ_2 be children of τ
    Compute the following by a recursive call

$$b(I_\tau) = b(I_\tau) + \begin{bmatrix} 0 & A_{\sigma_1,\sigma_2} \\ A_{\sigma_2,\sigma_1} & 0 \end{bmatrix} \begin{bmatrix} c(I_{\sigma_1}) \\ c(I_{\sigma_2}) \end{bmatrix} \tag{3.20}$$

  end
end

Note that the matrix product in (3.20) can be computed with computation of $A_{\sigma_1,\sigma_2} c(\sigma_2)$ and $A_{\sigma_2,\sigma_1} c(\sigma_1)$, which can be computed by two recursive calls.

3.6 *Other structures*

