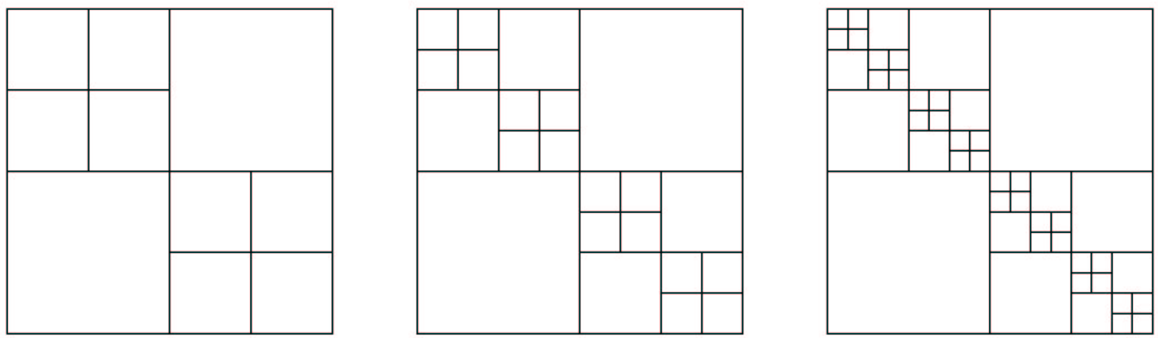
$\mathcal{H}$-matrix Inversion



Aniello Esposito **11.06.2004**

Überblick

Beispiel

- Inversion der Steifigkeitsmatrix einer elliptischen partiellen Differentialgleichung
- Die Frobenius Formel

Algorithmus

- Repetition: Implementierung `supermatrix`-Struktur
- Explizite Inversion einer 2×2 Block-Matrix
- (Komplexität)

Beispiel

Die Steifigkeitsmatrix

Man betrachtet die elliptische partielle Differentialgleichung ($d = 1$)

$$-\frac{\partial^2}{\partial x^2}\mathcal{U} = \mathcal{F}(x), \qquad x \in [0,1], \quad \mathcal{U} \in V$$

mit Dirichlet Randbedingungen

$$\mathcal{U}(0) = 0, \qquad \mathcal{U}(1) = 0$$

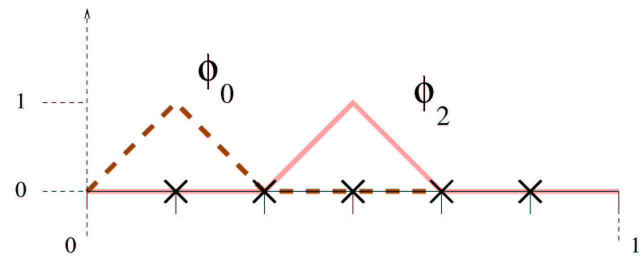
Die Galerkin Diskretisierung liefert nun das lineare Gleichungssystem

$$Gu = f, \quad G_{ij} \doteq \int_0^1 \frac{\partial}{\partial x}\varphi_i(x)\frac{\partial}{\partial x}\varphi_j(x)dx, \quad f_i \doteq \int_0^1 \mathcal{F}(x)\varphi_i(x)dx$$

wobei $\mathcal{U} = \sum_{j=0}^{n-1} u_j\varphi_j \quad (U \in V_n \subseteq V)$ und $V_n = span\{\varphi_0, \cdots, \varphi_{n-1}\}$.

Beispiel

Eine geeignete Wahl von Basisfunktionen auf $[0, 1]$



liefert die Steifigkeitsmatrix

$$G = (n+1) \begin{bmatrix} 2 & -1 & & & \\ -1 & 2 & -1 & & \\ & \ddots & \ddots & \ddots & \\ & & -1 & 2 & -1 \\ & & & -1 & 2 \end{bmatrix}$$

- Die Steifigkeitsmatrix ist dünn besetzt und somit leicht abzuspeichern.
- Die Inverse G^{-1} ist hingegen dicht besetzt und kompliziert auszurechnen.

Beispiel

Die Steifigkeitsmatrix im $\mathcal{H}$-matrix Format

Die Matrix G kann als 2×2 Block-Matrix geschrieben werden. ($n = 2^p$)

$$G = (n+1) \left[\begin{array}{c|c} G^* & \\ & -1 \\ \hline -1 & \\ & G^* \end{array} \right]$$

- G^* ist die Matrix für das reduzierte Problem ($n/2$).
- Die Block-Matrizen auf den Nebendiagonalen haben den Rang 1.

Betrachtet man den Block-Cluster Baum

$$\begin{aligned} root(T_{\mathcal{I}\times\mathcal{I}}) &\doteq \mathcal{I}\times\mathcal{I}, \qquad \mathcal{I} \doteq \{0,\cdots,n-1\} \\ sons(r\times s) &\doteq \begin{cases} \{r'\times s' | r' \in sons(r), s' \in sons(s)\} & \text{falls} \quad r = s \\ \emptyset & \text{sonst.} \end{cases} \end{aligned}$$

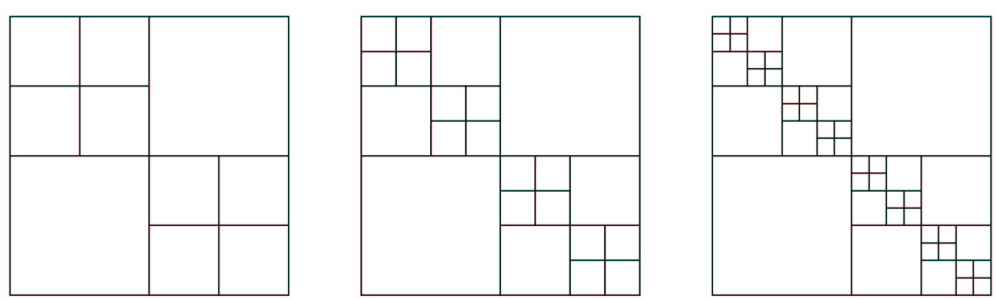
Beispiel

Die Steifigkeitsmatrix im $\mathcal{H}$-matrix Format

Die Matrix G kann als 2×2 Block-Matrix geschrieben werden. ($n = 2^p$)

$$G = (n+1) \left[\begin{array}{cc|cc} G^* & & & \\ & & -1 & \\ \hline & -1 & & \\ & & & G^* \end{array} \right]$$

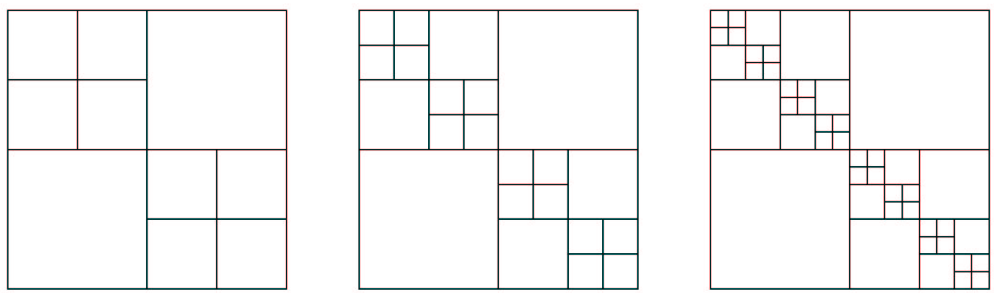
- G^* ist die Matrix für das reduzierte Problem ($n/2$).
- Die Block-Matrizen auf den Nebendiagonalen haben den Rang 1.



Beispiel

$$root(T_{\mathcal{I}\times\mathcal{I}}) \doteq \mathcal{I}\times\mathcal{I}, \qquad \mathcal{I} \doteq \{0,\cdots,n-1\}$$

$$sons(r\times s) \doteq \begin{cases} \{r'\times s' | r' \in sons(r), s' \in sons(s)\} & \text{falls} \quad r = s \\ \emptyset & \text{sonst.} \end{cases}$$



- Die Matrix G wird somit zu einer $\mathcal{H}$-Matrix basierend auf den Baum $T_{\mathcal{I}\times\mathcal{I}}$ mit blockweise Rang 1.

$$G \in \mathcal{H}(T_{\mathcal{I}\times\mathcal{I}}, 1)$$

Beispiel

Die Inverse zur Steifigkeitsmatrix im $\mathcal{H}$ Format

Die Inversion einer 2×2 Block-Matrix

Gegeben sei die 2×2 Block Matrix $M \in \mathbf{R}^{n \times n}$ in Blockform:

$$M = \begin{bmatrix} M_{11} & M_{12} \\ M_{21} & M_{22} \end{bmatrix}$$

mit Untermatrizen $M_{11} \in \mathbf{R}^{l \times l}$, $M_{22} \in \mathbf{R}^{m \times m}$, $M_{12} \in \mathbf{R}^{l \times m}$ und $M_{21} \in \mathbf{R}^{m \times l}$, wobei die Matrizen M und M_{11} regulär sind.

Nun betrachtet man eine Serie von regulären Operationen auf die Gleichung:

$$\boxed{M \cdot M^{-1} = I}$$

Beispiel

$$\boxed{M \cdot M^{-1} = I}$$

$$\begin{bmatrix} M_{11} & M_{12} \\ M_{21} & M_{22} \end{bmatrix} \cdot M^{-1} = \begin{bmatrix} I & 0 \\ 0 & I \end{bmatrix} \quad \Big| \quad \times \begin{bmatrix} M_{11}^{-1} & 0 \\ 0 & I \end{bmatrix}$$

$$\Rightarrow \begin{bmatrix} I & M_{11}^{-1} M_{12} \\ M_{21} & M_{22} \end{bmatrix} \cdot M^{-1} = \begin{bmatrix} M_{11}^{-1} & 0 \\ 0 & I \end{bmatrix} \quad \Big| \quad \times \begin{bmatrix} I & 0 \\ -M_{21} & I \end{bmatrix}$$

$$\Rightarrow \begin{bmatrix} I & M_{11}^{-1} M_{12} \\ 0 & M_{22} - M_{21} M_{11}^{-1} M_{12} \end{bmatrix} \cdot M^{-1} = \begin{bmatrix} M_{11}^{-1} & 0 \\ -M_{21} M_{11}^{-1} & I \end{bmatrix}$$

Beispiel

$$\begin{bmatrix} I & M_{11}^{-1}M_{12} \\ 0 & M_{22}-M_{21}M_{11}^{-1}M_{12} \end{bmatrix} \cdot M^{-1} = \begin{bmatrix} M_{11}^{-1} & 0 \\ -M_{21}M_{11}^{-1} & I \end{bmatrix}$$

$$\times \begin{bmatrix} I & 0 \\ 0 & (M_{22}-M_{21}M_{11}^{-1}M_{12})^{-1} \end{bmatrix}$$

$$\Rightarrow \begin{bmatrix} I & M_{11}^{-1}M_{12} \\ 0 & I \end{bmatrix} \cdot M^{-1}$$

$$= \begin{bmatrix} M_{11}^{-1} & 0 \\ -(M_{22}-M_{21}M_{11}^{-1}M_{12})^{-1}M_{21}M_{11}^{-1} & (M_{22}-M_{21}M_{11}^{-1}M_{12})^{-1} \end{bmatrix}$$

$$\times \begin{bmatrix} I & -M_{11}^{-1}M_{12} \\ 0 & I \end{bmatrix}$$

Beispiel

liefert das Schlussresultat

$$\begin{bmatrix} I & 0 \\ 0 & I \end{bmatrix} \cdot M^{-1} = \begin{bmatrix} M_{11}^{-1} + M_{11}^{-1} M_{12} S^{-1} M_{21} M_{11}^{-1} & -M_{11}^{-1} M_{12} S^{-1} \\ -S^{-1} M_{21} M_{11}^{-1} & S^{-1} \end{bmatrix}$$

$$M^{-1} = \begin{bmatrix} M_{11}^{-1} + M_{11}^{-1} M_{12} S^{-1} M_{21} M_{11}^{-1} & -M_{11}^{-1} M_{12} S^{-1} \\ -S^{-1} M_{21} M_{11}^{-1} & S^{-1} \end{bmatrix}$$

mit der Matrix $S \equiv M_{22} - M_{21} M_{11}^{-1} M_{12}$.

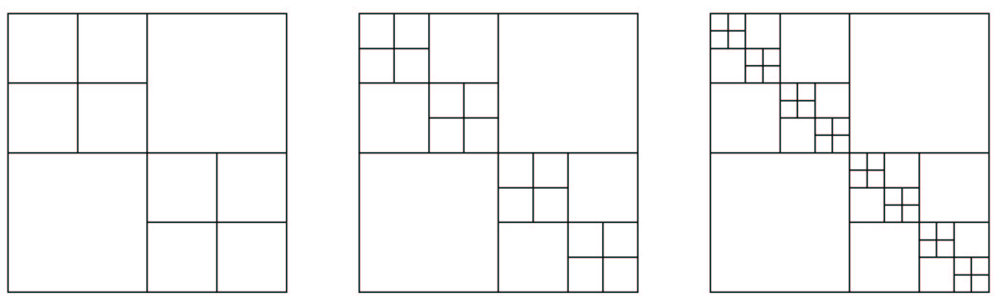
Beispiel

Die Frobenius Formel

$$M^{-1} = \left[\begin{array}{cc} M_{11}^{-1} + M_{11}^{-1} M_{12} S^{-1} M_{21} M_{11}^{-1} & -M_{11}^{-1} M_{12} S^{-1} \\ -S^{-1} M_{21} M_{11}^{-1} & S^{-1} \end{array} \right]$$

kann verallgemeinert werden auf beliebige $m \times m$ Block-Matrizen.

- Die Inversion wurde reduziert auf Addition und Multiplikation von Untermatrizen.
- Die Inverse zur Steifigkeitsmatrix erhält man über die sukzessive Anwendung der obigen Formel.



Algorithmus

Algorithmus

Implementierung der `supermatrix`-Struktur

```
typedef struct _supermatrix supermatrix;
typedef supermatrix *psupermatrix;

struct _supermatrix {
  int rows;
  int cols;
  int block_rows;
  int block_cols;
   prkmatrix r;
  pfullmatrix f;
  psupermatrix* s;
};
```

- Hier betrachtet man im Allgemeinen den Fall $s \neq 0$. Somit eine reine Supermatrix.

Algorithmus

Der array `s` aus `psupermatrix* s` enthält alle pointers zu den Untermatrizen $M_{i,j}$ der Matrix:

$$M = \begin{bmatrix} M_{1,1} & \cdots & M_{1,\mathtt{blockcols}} \\ \vdots & \ddots & \vdots \\ M_{\mathtt{blockrows},1} & \cdots & M_{\mathtt{blockrows},\mathtt{blockcols}} \end{bmatrix}$$

in der Reihenfolge:

$$M_{1,1}, \cdots, M_{\mathtt{blockrows},1}, \quad M_{1,2}, \cdots, M_{\mathtt{blockrows},2}, \quad M_{1,\mathtt{blockcols}}, \cdots$$
$$, M_{\mathtt{blockrows},\mathtt{blockcols}}$$

Algorithmus

Preliminary formatted Inversion

Sei ein Block Cluster Baum $T_{\mathcal{I}\times\mathcal{I}}$ und eine zahl $k \in \mathcal{N}$ gegeben. Der Preliminary formatted Inversion Operator ist definiert als:

$$Inv* : \{M \in \mathcal{R}^{n\times n} | \mathrm{Rang}(M) = n\} \to \mathcal{H}(T_{\mathcal{I}\times\mathcal{I}}, k), \qquad M \mapsto \mathcal{T}_k(M^{-1}).$$

Eine approximierte Inversion $Inv_{\mathcal{H}}(M)$ wird berechnet mit Hilfe von:

$$M^{-1} = \begin{bmatrix} M_{11}^{-1} + M_{11}^{-1} M_{12} S^{-1} M_{21} M_{11}^{-1} & -M_{11}^{-1} M_{12} S^{-1} \\ -S^{-1} M_{21} M_{11}^{-1} & S^{-1} \end{bmatrix}$$

wobei $S \equiv (M_{22} - M_{21} M_{11}^{-1} M_{12})^{-1}$.

Algorithmus

Preliminary formatted Inversion

Sei ein Block Cluster Baum $T_{\mathcal{I}\times\mathcal{I}}$ und eine zahl $k \in \mathcal{N}$ gegeben. Der Preliminary formatted Inversion Operator ist definiert als:

$$Inv* : \{M \in \mathcal{R}^{n\times n} | \mathrm{Rang}(M) = n\} \to \mathcal{H}(T_{\mathcal{I}\times\mathcal{I}}, k), \qquad M \mapsto \mathcal{T}_k(M^{-1}).$$

- Die Inversionen der Matrizen M_{11} und S seien schon gegeben.
- Die Additionen und Multiplikationen in der Formel werden durch die formattierten Operatoren $\oplus$ und $\odot$ erstetzt.
- Man erhält eine approximierte Inversion $Inv_{\mathcal{H}}(M)$ zu $Inv*(M)$ welche als formattierte Inverse bezeichnet wird.

Verallgemeinerung auf den Fall `blockrows` $\times$ `blockcols` über einen Algorithmus.

Algorithmus

Der Algorithmus für den Fall 2×2.

```
void
invert_supermatrix(psupermatrix si, psupermatrix s,
 psupermatrix swork){
  int block_rows,block_cols,i,j,l;
  psupermatrix *sw_e= swork->s, *s_e= s->s, *si_e = si->s;
  block_rows = s->block_rows;
  block_cols = s->block_cols;


if(si->s==0x0 || s->s==0x0 || swork->s==0x0){
    */ Inversion im Raum der fullmatrices */
    invert_fullmatrix(si->f,s->f,swork->f);
  }
```

- Ist die Matrix s eine reine Supermatrix ?

Algorithmus

```
else{
      for(l=0; l<block_rows; l++){

      /* Invertiere auf der Diagonalen */
      invert_supermatrix(si_e[l+block_rows*l],
      s_e[l+block_rows*l],sw_e[l+block_rows*l]);
      ...
```

$$Start \qquad s_e = \begin{bmatrix} M_{11} & M_{12} \\ M_{21} & M_{22} \end{bmatrix} \quad sw_e = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix} \quad si_e = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}$$

Algorithmus

```
else{
      for(l=0; l<block_rows; l++){

      /* Invertiere auf der Diagonalen */
      invert_supermatrix(si_e[l+block_rows*l],
      s_e[l+block_rows*l],sw_e[l+block_rows*l]);
      ...
```

$$l = 0 \qquad s_e = \begin{bmatrix} M_{11} & M_{12} \\ M_{21} & M_{22} \end{bmatrix} \qquad sw_e = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix} \qquad si_e = \begin{bmatrix} M_{11}^{-1} & 0 \\ 0 & 0 \end{bmatrix}$$

Algorithmus

```
for(j=l+1; j<bm; j++){
    mul_supermatrix(sw_e[l+bn*j],
            si_e[l+bn*l],s_e[l+bn*j]);
    copydata_supermatrix(sw_e[l+bn*j],s_e[l+bn*j]);
                      }
```

$$l = 0 \qquad s_e = \begin{bmatrix} M_{11} & M_{12} \\ M_{21} & M_{22} \end{bmatrix} \qquad sw_e = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix} \qquad si_e = \begin{bmatrix} M_{11}^{-1} & 0 \\ 0 & 0 \end{bmatrix}$$

Algorithmus

```
for(j=l+1; j<bm; j++){
    mul_supermatrix(sw_e[l+bn*j],
            si_e[l+bn*l],s_e[l+bn*j]);
    copydata_supermatrix(sw_e[l+bn*j],s_e[l+bn*j]);
                      }
```

$$l = 0 \qquad j = 1$$

$$s_e = \begin{bmatrix} M_{11} & M_{12} \\ M_{21} & M_{22} \end{bmatrix} \qquad sw_e = \begin{bmatrix} 0 & M_{11}^{-1} M_{12} \\ 0 & 0 \end{bmatrix}$$

$$si_e = \begin{bmatrix} M_{11}^{-1} & 0 \\ 0 & 0 \end{bmatrix}$$

Algorithmus

```
for(j=l+1; j<bm; j++){
    mul_supermatrix(sw_e[l+bn*j],
            si_e[l+bn*l],s_e[l+bn*j]);
    copydata_supermatrix(sw_e[l+bn*j],s_e[l+bn*j]);
                      }
```

$$l = 0 \qquad j = 1$$

$$s_e = \begin{bmatrix} M_{11} & M_{11}^{-1} M_{12} \\ M_{21} & M_{22} \end{bmatrix} \qquad sw_e = \begin{bmatrix} 0 & M_{11}^{-1} M_{12} \\ 0 & 0 \end{bmatrix}$$

$$si_e = \begin{bmatrix} M_{11}^{-1} & 0 \\ 0 & 0 \end{bmatrix}$$

Algorithmus

```
for(i=l+1; i<block_rows; i++){
/* Subtraktion von der Inversen */
for(j=0; j<=l; j++){
  mul_supermatrix(sw_e[i+bn*j],
  s_e[i+bn*l],si_e[l+bn*j]);
  scale_supermatrix(sw_e[i+bn*j],-1.0);
  addto_supermatrix(si_e[i+bn*j],sw_e[i+bn*j]);
}
```

$$s_e = \begin{bmatrix} M_{11} & M_{11}^{-1} M_{12} \\ M_{21} & M_{22} \end{bmatrix} \qquad sw_e = \begin{bmatrix} 0 & M_{11}^{-1} M_{12} \\ 0 & 0 \end{bmatrix}$$

$$si_e = \begin{bmatrix} M_{11}^{-1} & 0 \\ 0 & 0 \end{bmatrix}$$

Algorithmus

```
for(i=l+1; i<block_rows; i++){
/* Subtraktion von der Inversen */
for(j=0; j<=l; j++){
  mul_supermatrix(sw_e[i+bn*j],
  s_e[i+bn*l],si_e[l+bn*j]);
  scale_supermatrix(sw_e[i+bn*j],-1.0);
  addto_supermatrix(si_e[i+bn*j],sw_e[i+bn*j]);
}
```

$$l = 0$$
$$i = 1$$
$$j = 0$$

$$s_e = \begin{bmatrix} M_{11} & M_{11}^{-1} M_{12} \\ M_{21} & M_{22} \end{bmatrix} \qquad sw_e = \begin{bmatrix} 0 & M_{11}^{-1} M_{12} \\ M_{21} M_{11}^{-1} & 0 \end{bmatrix}$$

$$si_e = \begin{bmatrix} M_{11}^{-1} & 0 \\ 0 & 0 \end{bmatrix}$$

Algorithmus

```
for(i=l+1; i<block_rows; i++){
/* Subtraktion von der Inversen */
for(j=0; j<=l; j++){
  mul_supermatrix(sw_e[i+bn*j],
  s_e[i+bn*l],si_e[l+bn*j]);
  scale_supermatrix(sw_e[i+bn*j],-1.0);
  addto_supermatrix(si_e[i+bn*j],sw_e[i+bn*j]);
}
```

$$l = 0 \qquad i = 1 \qquad j = 0$$

$$s_e = \begin{bmatrix} M_{11} & M_{11}^{-1}M_{12} \\ M_{21} & M_{22} \end{bmatrix} \qquad sw_e = \begin{bmatrix} 0 & M_{11}^{-1}M_{12} \\ -M_{21}M_{11}^{-1} & 0 \end{bmatrix}$$

$$si_e = \begin{bmatrix} M_{11}^{-1} & 0 \\ -M_{21}M_{11}^{-1} & 0 \end{bmatrix}$$

Algorithmus

```
/* Subtraktion von der Matrix */
for(j=l+1; j<block_cols; j++){
  mul_supermatrix(sw_e[i+bn*j],
  s_e[i+bn*l],s_e[l+bn*j]);
  scale_supermatrix(sw_e[i+bn*j],-1.0);
  addto_supermatrix(s_e[i+bn*j],sw_e[i+bn*j]);
}
```

$$l = 0$$
$$i = 1$$
$$j = 0$$

$$s_e = \begin{bmatrix} M_{11} & M_{11}^{-1}M_{12} \\ M_{21} & M_{22} \end{bmatrix} \qquad sw_e = \begin{bmatrix} 0 & M_{11}^{-1}M_{12} \\ -M_{21}M_{11}^{-1} & 0 \end{bmatrix}$$

$$si_e = \begin{bmatrix} M_{11}^{-1} & 0 \\ -M_{21}M_{11}^{-1} & 0 \end{bmatrix}$$

Algorithmus

```
/* Subtraktion von der Matrix */
for(j=l+1; j<block_cols; j++){
  mul_supermatrix(sw_e[i+bn*j],
  s_e[i+bn*l],s_e[l+bn*j]);
  scale_supermatrix(sw_e[i+bn*j],-1.0);
  addto_supermatrix(s_e[i+bn*j],sw_e[i+bn*j]);
}
```

$$l = 0 \qquad i = 1 \qquad j = 1$$

$$s_e = \begin{bmatrix} M_{11} & M_{11}^{-1} M_{12} \\ M_{21} & M_{22} \end{bmatrix} \qquad sw_e = \begin{bmatrix} 0 & M_{11}^{-1} M_{12} \\ -M_{21} M_{11}^{-1} & M_{21} M_{11}^{-1} M_{12} \end{bmatrix}$$

$$si_e = \begin{bmatrix} M_{11}^{-1} & 0 \\ -M_{21} M_{11}^{-1} & 0 \end{bmatrix}$$

Algorithmus

```
/* Subtraktion von der Matrix */
for(j=l+1; j<block_cols; j++){
  mul_supermatrix(sw_e[i+bn*j],
  s_e[i+bn*l],s_e[l+bn*j]);
  scale_supermatrix(sw_e[i+bn*j],-1.0);
  addto_supermatrix(s_e[i+bn*j],sw_e[i+bn*j]);
}
```

$$l = 0 \qquad i = 1 \qquad j = 1$$

$$s_e = \begin{bmatrix} M_{11} & M_{11}^{-1}M_{12} \\ M_{21} & M_{22} - M_{21}M_{11}^{-1}M_{12} \end{bmatrix}$$

$$sw_e = \begin{bmatrix} 0 & M_{11}^{-1}M_{12} \\ -M_{21}M_{11}^{-1} & -M_{21}M_{11}^{-1}M_{12} \end{bmatrix}$$

$$si_e = \begin{bmatrix} M_{11}^{-1} & 0 \\ -M_{21}M_{11}^{-1} & 0 \end{bmatrix}$$

Algorithmus

```
for(l=0; l<block_rows; l++){

      /* Invertiere auf der Diagonalen */
      invert_supermatrix(si_e[l+block_rows*l],
      s_e[l+block_rows*l],sw_e[l+block_rows*l]);
      ...
```

$$s_e = \begin{bmatrix} M_{11} & M_{11}^{-1} M_{12} \\ M_{21} & M_{22} - M_{21} M_{11}^{-1} M_{12} \end{bmatrix}$$

$$sw_e = \begin{bmatrix} 0 & M_{11}^{-1} M_{12} \\ -M_{21} M_{11}^{-1} & -M_{21} M_{11}^{-1} M_{12} \end{bmatrix}$$

$$si_e = \begin{bmatrix} M_{11}^{-1} & 0 \\ -M_{21} M_{11}^{-1} & 0 \end{bmatrix}$$

Algorithmus

```
for(l=0; l<block_rows; l++){

      /* Invertiere auf der Diagonalen */
      invert_supermatrix(si_e[l+block_rows*l],
      s_e[l+block_rows*l],sw_e[l+block_rows*l]);
      ...
```

$$l = 1 \qquad s_e = \begin{bmatrix} M_{11} & M_{11}^{-1} M_{12} \\ M_{21} & M_{22} - M_{21} M_{11}^{-1} M_{12} \end{bmatrix}$$

$$sw_e = \begin{bmatrix} 0 & M_{11}^{-1} M_{12} \\ -M_{21} M_{11}^{-1} & -M_{21} M_{11}^{-1} M_{12} \end{bmatrix}$$

$$si_e = \begin{bmatrix} M_{11}^{-1} & 0 \\ -M_{21} M_{11}^{-1} & (M_{22} - M_{21} M_{11}^{-1} M_{12})^{-1} \end{bmatrix}$$

Algorithmus

```
for(j=0; j<l; j++){
    mul_supermatrix(sw_e[l+bn*j],
    si_e[l+bn*l],si_e[l+bn*j]);
    copydata_supermatrix(sw_e[l+bn*j],si_e[l+bn*j]);
      }
```

$$l = 1 \qquad s_e = \begin{bmatrix} M_{11} & M_{11}^{-1} M_{12} \\ M_{21} & M_{22} - M_{21} M_{11}^{-1} M_{12} \end{bmatrix}$$

$$sw_e = \begin{bmatrix} 0 & M_{11}^{-1} M_{12} \\ -M_{21} M_{11}^{-1} & -M_{21} M_{11}^{-1} M_{12} \end{bmatrix}$$

$$si_e = \begin{bmatrix} M_{11}^{-1} & 0 \\ -M_{21} M_{11}^{-1} & (M_{22} - M_{21} M_{11}^{-1} M_{12})^{-1} \end{bmatrix}$$

Algorithmus

```
for(j=0; j<l; j++){
    mul_supermatrix(sw_e[l+bn*j],
    si_e[l+bn*l],si_e[l+bn*j]);
    copydata_supermatrix(sw_e[l+bn*j],si_e[l+bn*j]);
      }
```

$$\begin{array}{l} l = 1 \\ j = 0 \end{array} \qquad s_e = \begin{bmatrix} M_{11} & M_{11}^{-1} M_{12} \\ M_{21} & M_{22} - M_{21} M_{11}^{-1} M_{12} \end{bmatrix}$$

$$sw_e = \begin{bmatrix} 0 & M_{11}^{-1} M_{12} \\ -(M_{22} - M_{21} M_{11}^{-1} M_{12})^{-1} M_{21} M_{11}^{-1} & -M_{21} M_{11}^{-1} M_{12} \end{bmatrix}$$

$$si_e = \begin{bmatrix} M_{11}^{-1} & 0 \\ -M_{21} M_{11}^{-1} & (M_{22} - M_{21} M_{11}^{-1} M_{12})^{-1} \end{bmatrix}$$

Algorithmus

```
for(j=0; j<l; j++){
    mul_supermatrix(sw_e[l+bn*j],
    si_e[l+bn*l],si_e[l+bn*j]);
    copydata_supermatrix(sw_e[l+bn*j],si_e[l+bn*j]);
      }
```

$$\begin{array}{c} l = 1 \\ j = 0 \end{array} \qquad s_e = \begin{bmatrix} M_{11} & M_{11}^{-1} M_{12} \\ M_{21} & M_{22} - M_{21} M_{11}^{-1} M_{12} \end{bmatrix}$$

$$sw_e = \begin{bmatrix} 0 & M_{11}^{-1} M_{12} \\ -(M_{22} - M_{21} M_{11}^{-1} M_{12})^{-1} M_{21} M_{11}^{-1} & -M_{21} M_{11}^{-1} M_{12} \end{bmatrix}$$

$$si_e = \begin{bmatrix} M_{11}^{-1} & 0 \\ -(M_{22} - M_{21} M_{11}^{-1} M_{12})^{-1} M_{21} M_{11}^{-1} & (M_{22} - M_{21} M_{11}^{-1} M_{12})^{-1} \end{bmatrix}$$

Algorithmus

```
    for(l=bn-1; l>=0; l--){
       for(i=l-1; i>=0; i--){
 for(j=0; j<bm; j++){
   mul_supermatrix(sw_e[i+bn*j],
   s_e[i+bn*l],si_e[l+bn*j]);
   scale_supermatrix(sw_e[i+bn*j],-1.0);
   addto_supermatrix(si_e[i+bn*j],sw_e[i+bn*j]);
                     }
                            }
                          }
```

Algorithmus

```
 for(j=0; j<bm; j++){
mul_supermatrix(sw_e[i+bn*j],
s_e[i+bn*l],si_e[l+bn*j]);
scale_supermatrix(sw_e[i+bn*j],-1.0);
addto_supermatrix(si_e[i+bn*j],sw_e[i+bn*j]);
                }
```

$$s_e = \begin{bmatrix} M_{11} & M_{11}^{-1}M_{12} \\ M_{21} & M_{22} - M_{21}M_{11}^{-1}M_{12} \end{bmatrix}$$

$$sw_e = \begin{bmatrix} 0 & M_{11}^{-1}M_{12} \\ -(M_{22} - M_{21}M_{11}^{-1}M_{12})^{-1}M_{21}M_{11}^{-1} & -M_{21}M_{11}^{-1}M_{12} \end{bmatrix}$$

$$si_e = \begin{bmatrix} M_{11}^{-1} & 0 \\ -(M_{22} - M_{21}M_{11}^{-1}M_{12})^{-1}M_{21}M_{11}^{-1} & (M_{22} - M_{21}M_{11}^{-1}M_{12})^{-1} \end{bmatrix}$$

Algorithmus

```
 for(j=0; j<bm; j++){
mul_supermatrix(sw_e[i+bn*j],
s_e[i+bn*l],si_e[l+bn*j]);
scale_supermatrix(sw_e[i+bn*j],-1.0);
addto_supermatrix(si_e[i+bn*j],sw_e[i+bn*j]);
                }
```

$$l = 1 \qquad i = 0 \qquad j = 0$$

$$s_e = \begin{bmatrix} M_{11} & M_{11}^{-1} M_{12} \\ M_{21} & S \end{bmatrix}$$

$$sw_e = \begin{bmatrix} -M_{11}^{-1} M_{12} S^{-1} M_{21} M_{11}^{-1} & M_{11}^{-1} M_{12} \\ -S^{-1} M_{21} M_{11}^{-1} & -M_{21} M_{11}^{-1} M_{12} \end{bmatrix}$$

$$si_e = \begin{bmatrix} M_{11}^{-1} & 0 \\ -S^{-1} M_{21} M_{11}^{-1} & S^{-1} \end{bmatrix}$$

mit $S = M_{22} - M_{21} M_{11}^{-1} M_{12}$

Algorithmus

```
 for(j=0; j<bm; j++){
mul_supermatrix(sw_e[i+bn*j],
s_e[i+bn*l],si_e[l+bn*j]);
scale_supermatrix(sw_e[i+bn*j],-1.0);
addto_supermatrix(si_e[i+bn*j],sw_e[i+bn*j]);
                 }
```

$$l = 1 \qquad i = 0 \qquad j = 0$$

$$s_e = \begin{bmatrix} M_{11} & M_{11}^{-1} M_{12} \\ M_{21} & S \end{bmatrix}$$

$$sw_e = \begin{bmatrix} M_{11}^{-1} M_{12} S^{-1} M_{21} M_{11}^{-1} & M_{11}^{-1} M_{12} \\ -S^{-1} M_{21} M_{11}^{-1} & -M_{21} M_{11}^{-1} M_{12} \end{bmatrix}$$

$$si_e = \begin{bmatrix} M_{11}^{-1} + M_{11}^{-1} M_{12} S^{-1} M_{21} M_{11}^{-1} & 0 \\ -S^{-1} M_{21} M_{11}^{-1} & S^{-1} \end{bmatrix}$$

Algorithmus

```
 for(j=0; j<bm; j++){
mul_supermatrix(sw_e[i+bn*j],
s_e[i+bn*l],si_e[l+bn*j]);
scale_supermatrix(sw_e[i+bn*j],-1.0);
addto_supermatrix(si_e[i+bn*j],sw_e[i+bn*j]);
                 }
```

$$l = 1 \qquad i = 0 \qquad j = 1$$

$$s_e = \begin{bmatrix} M_{11} & M_{11}^{-1} M_{12} \\ M_{21} & S \end{bmatrix}$$

$$sw_e = \begin{bmatrix} M_{11}^{-1} M_{12} S^{-1} M_{21} M_{11}^{-1} & M_{11}^{-1} M_{12} S^{-1} \\ -S^{-1} M_{21} M_{11}^{-1} & -M_{21} M_{11}^{-1} M_{12} \end{bmatrix}$$

$$si_e = \begin{bmatrix} M_{11}^{-1} + M_{11}^{-1} M_{12} S^{-1} M_{21} M_{11}^{-1} & 0 \\ -S^{-1} M_{21} M_{11}^{-1} & S^{-1} \end{bmatrix}$$

Algorithmus

```
 for(j=0; j<bm; j++){
mul_supermatrix(sw_e[i+bn*j],
s_e[i+bn*l],si_e[l+bn*j]);
scale_supermatrix(sw_e[i+bn*j],-1.0);
addto_supermatrix(si_e[i+bn*j],sw_e[i+bn*j]);
                 }
```

$$l = 1$$
$$i = 0$$
$$j = 1$$

$$s_e = \begin{bmatrix} M_{11} & M_{11}^{-1}M_{12} \\ M_{21} & S \end{bmatrix}$$

$$sw_e = \begin{bmatrix} M_{11}^{-1}M_{12}S^{-1}M_{21}M_{11}^{-1} & -M_{11}^{-1}M_{12}S^{-1} \\ -S^{-1}M_{21}M_{11}^{-1} & -M_{21}M_{11}^{-1}M_{12} \end{bmatrix}$$

$$si_e = \begin{bmatrix} M_{11}^{-1} + M_{11}^{-1}M_{12}S^{-1}M_{21}M_{11}^{-1} & -M_{11}^{-1}M_{12}S^{-1} \\ -S^{-1}M_{21}M_{11}^{-1} & S^{-1} \end{bmatrix}$$

Komplexität

Sei ein Block-Cluster-Baum $T_{\mathcal{I}\times\mathcal{I}}$ gegeben. Man nimmt an, dass für die Inversion der `fullmatrix`-Blocks die Komplexität über die der Multiplikation gebunden ist. Dann ist die Komplexität der formattierten Inversion $N_{\mathcal{H},Inv}(T,k)$ in die Menge $\mathcal{I}(T,k)$ gebunden über $N_{\odot}(T,k)$.