Technische Universität Chemnitz-Zwickau

Mathias Meisel   Arnd Meyer

Implementierung eines parallelen vorkonditionierten Schur-Komplement CG-Verfahrens in das Programmpaket FEAP

Fakultät für Mathematik
TU Chemnitz-Zwickau
PSF 09107
D-09107 Chemnitz, FRG
(0371)-531-2657 (fax)

mathias.meisel@imech.tu-chemnitz.de
(0371)-531-4686

arnd.meyer@mathematik.tu-chemnitz.de
(0371)-531-2659

Preprint-Reihe der Chemnitzer DFG-Forschergruppe
"Scientific Parallel Computing"

SPC 95_2   **Januar 1995**

Diese Arbeit wurde gefördert durch die Deutsche Forschungsgemeinschaft (Vertrag Nr. Me1224).

Implementierung eines parallelen vorkonditionierten Schur-Komplement CG-Verfahrens in das Programmpaket FEAP

Mathias Meisel　　Arnd Meyer

17. Februar 1997

Abstract

A parallel realisation of the Conjugate Gradient Method with Schur–Complement preconditioning, based on a domain decomposition approach, is described in detail. Special kinds of solvers for the resulting interiour and coupling systems are presented. A large range of numerical results is used to demonstrate the properties and behaviour of this solvers in practical situations.

Inhaltsverzeichnis

1 Einführung

Das Programmpaket FEAP (**F**inite **E**lement **A**nalysis **P**rogram) wurde für die Durchführung von Simulationsrechnungen zu Problemen der Festkörpermechanik auf UNIX-Rechnern oder leistungsfähigen PC's entwickelt.

Mit seinen etwa 40.000 Quelltextzeilen bietet es vielfältige Möglichkeiten zur Problemformulierung, zur interaktiven Gestaltung des Lösungsprozesses und zur Ergebnispräsentation ([3]).

Da die Simulation realer Bauteilstrukturen nicht selten Berechnungsgitter mit relativ großen Gitterpunktzahlen erfordert, sind bei der Behandlung derartiger Problemstellungen typischerweise großdimensionierte Gleichungssysteme mit schwach besetzter Systemmatrix zu lösen, wofür neben anderen die Klasse der vorkonditionierten Gradientenverfahren i.allg. besser geeignet erscheint als der ursprünglich in FEAP implementierte direkte Löser. Dennoch bleibt die Lösung dieser Gleichungssysteme der rechenzeitintensivste Programmteil mit dem größten Speicherbedarf.

Um der interaktiven Arbeitsweise von FEAP gerecht zu werden ist es wünschenswert, die im Löser verbrauchte Rechenzeit so gering wie möglich zu halten. Hier bietet sich der Einsatz von Mehrprozessorsystemen mit verteiltem Speicher an.

Als Parallelisierungszugang wird wie in [2] die Methode der Gebietszerlegung (Domain Decomposition) gewählt. Dies gestattet die vollständige Parallelisierung des gesamten Arithmetikaufwandes von der Netzgenerierung in den Teilgebieten über die Assemblierung der lokalen Teilsteifigkeitsmatrizen bis hin zur Lösung der Gleichungssysteme.

Ziel dieses Artikels ist es, die theoretische Beschreibung des Lösungsverfahrens mit einigen Bemerkungen zu seiner Implementierung in eine derzeit entstehende parallele Version des Programmpakets FEAP zu verbinden. Zur Abhebung vom allgemeintheoretischen Teil werden die speziell auf FEAP bezogenen Textteile im weiteren als Fußnoten gesetzt.

2 Gittergenerierung und Assemblierung

Für die Simulationsrechnung auf $p = 2^m$ Prozessoren (m-dimensionaler Hypercube) sei das Berechnungsgebiet Ω nichtüberlappend in $r \leq p$ Teilgebiete Ω_s zerlegt ($\Omega_s \cap \Omega_t = \emptyset$ für $s \neq t$; $0 \leq s, t \leq r-1$) und jedem der Prozessoren $\mathcal{P}_s$ sei höchstens eines dieser Teilgebiete zugeordnet. Der Einfachheit halber sei im folgenden $r = p$. Auf den Teilgebietsrändern $\Gamma_s := \partial\Omega_s$ seien Randgitterpunkte mit der für die Berechnungen gewünschten Dichte vorgegeben. Stimmen diese Vorgaben auf den Koppelrändern $\Gamma_{st} := \overline{\Omega}_s \cap \overline{\Omega}_t$ in jeweils beiden Prozessoren $\mathcal{P}_s$ und $\mathcal{P}_t$ in Anzahl und Koordinatenwerten überein, dann kann kommunikationsfrei das globale FEM-Netz erzeugt werden, indem in allen Prozessoren ein lokales FEM-Netz generiert wird. [1]

[1] Hierzu konnten weitgehend die entsprechenden Subroutinen der sequentiellen FEAP-Version benutzt werden, nachdem das Eingabekonzept geeignet modifiziert und die Verteilung der Teilgebiete Ω_s auf die Prozessoren $\mathcal{P}_s$ programmtechnisch realisiert waren ([4]).

Die dabei in $\overline{\Omega}_s$ entstandenen Gitterpunkte seien ohne Beschränkung der Allgemeinheit lokal so numeriert, daß die auf Γ_s vorgegebenen Punkte die Nummern 1 bis m_s^C und die m_s^I im Inneren von Ω_s liegenden Punkte die Nummern m_s^C+1 bis $m_s^C+m_s^I =: m_s$ erhalten. Dieser Numerierung folgend sei $n_s := n_s^I + n_s^C$ die Zahl der Freiheitsgrade in $\overline{\Omega}_s$. Ferner seien n die globale Anzahl der Freiheitsgrade im zu lösenden diskreten Problem, $n^I := \sum_{s=0}^{p-1} n_s^I$ die Gesamtzahl aller im Inneren der Teilgebiete Ω_s liegenden Freiheitsgrade , $n^C := n - n^I$ die Anzahl aller Koppel- bzw. Randfreiheitsgrade und n_e die Zahl der Freiheitsgrade im finiten Element $e \subset \overline{\Omega}$.

Da jeder Prozessor über alle Geometriedaten seines Subdomains verfügt, können die Elementsteifigkeitsmatritzen $K_e \in R^{n_e,n_e}$ sowie die Elementlasten $b_e \in R^{n_e}$ aller finiten Elemente $e \subset \overline{\Omega}_s$ und daraus gemäß

$$K_s = \sum_{e\subset\overline{\Omega}_s} A_e^T K_e A_e \qquad\qquad b_s = \sum_{e\subset\overline{\Omega}_s} A_e^T b_e \tag{2.1}$$

die lokalen Steifigkeitsmatritzen $K_s \in R^{n_s,n_s}$ sowie die lokalen Lastvektoren $b_s \in R^{n_s}$ parallel und ohne Kommunikation zwischen den Prozessoren[2] berechnet werden. Die Boolschen Matrizen $A_e \in R^{n_e,n_s}$ in (2.1) stellen den Zusammenhang zwischen der jeweiligen Numerierung der Freiheitsgrade innerhalb der Elemente e und der Numerierung in Ω_s her.

Zwischen K_s , b_s , der globalen Steifigkeitsmatrix $K \in R^{n,n}$ und der globalen rechten Seite $b \in R^n$ besteht analog zu (2.1) der Zusammenhang

$$K = \sum_{s=0}^{p-1} A_s^T K_s A_s \qquad\qquad b = \sum_{s=0}^{p-1} A_s^T b_s \tag{2.2}$$

mit der Boolschen Matrix $A_s \in R^{n_s,n}$, jedoch würde die Bildung von K und b einen globalen Datenaustausch über die Koppelknoten erfordern. In [2] wurde bereits gezeigt, daß es für die Realisierung des parallelen CG-Verfahrens ausreicht, K und b in summarisch verteilter Form abzuspeichern und (2.2) nicht explizit auszuführen.

Neben der durch (2.2) definierten Speicherform für Vektoren (und Matrizen), bei der sich der Gesamtvektor - theoretisch - durch die FEM-Assemblierung seiner lokalen Anteile ergibt, wird u.a. für die zu berechnende Lösung x eine zweite Speicherform, definiert durch

$$x_s = A_s x \ , \tag{2.3}$$

benötigt, bei der in $\mathcal{P}_s$ derjenige Teil von x gespeichert ist, der in $\overline{\Omega}_s$ liegenden Freiheitsgraden entspricht.

Der wesentliche Unterschied zwischen beiden Speicherformen besteht darin, daß die einem Koppelknoten Q zuzuordnenden Komponenten der Vektoren bei (2.2) summarisch aus allen in $\mathcal{P}_t$ mit $Q \in \overline{\Omega}_t$ gespeicherten Anteilen zu bilden sind, während bei (2.3) jeder der Prozessoren $\mathcal{P}_t$ eine Kopie dieser Komponenten besitzt.

[2]unter direkter Verwendung der Assemblierungsroutinen aus der sequentiellen FEAP-Version, aber eine grundlegend andere Datenstruktur zur Speicherung der Matritzen K_s benutzend

2.1 Einige weitere Bezeichnungen

Einige Bezeichnungen und Zusammenhänge sollen hier exemplarisch für den Lösungsvektor x und die Steifigkeitsmatrix K dargestellt werden, wenngleich sie im weiteren auch für andere Vektoren und Matrizen Verwendung finden.

Die angenommene lokale Numerierungsweise der Knoten und Freiheitsgrade impliziert für Vektoren x_s folgende Strukturierung:

$$x_s = \begin{pmatrix} x_s^C \\ x_s^I \end{pmatrix} \quad . \tag{2.4}$$

Dabei enthält $x_s^C \in R^{n_s^C}$ die zu Knoten $Q \in \Gamma_s$ gehörenden Freiheitsgrade und $x_s^I \in R^{n_s^I}$ sind die inneren Freiheitsgrade, die zu Knoten $Q \in \Omega_s$ gehören.
Für die Matrizen K_s ergibt sich daraus die Blockstruktur

$$K_s = \begin{pmatrix} K_s^C & K_s^{CI} \\ K_s^{IC} & K_s^I \end{pmatrix} \tag{2.5}$$

mit $K_s^C \in R^{n_s^C, n_s^C}$, $K_s^I \in R^{n_s^I, n_s^I}$, $K_s^{CI} \in R^{n_s^I, n_s^C}$ und $K_s^{IC} \in R^{n_s^C, n_s^I}$.

Eine zu (2.4) und (2.5) analoge Blockstruktur ergibt sich für Vektoren $x \in R^n$ und die Matrix $K \in R^{n,n}$, wenn global erst alle n^C Koppel- oder Randfreiheitsgrade und dann alle bezüglich der Teilgebiete Ω_s inneren Freiheitsgrade numeriert werden:

$$\begin{aligned} K &= \begin{pmatrix} K^C & K^{CI} \\ K^{IC} & K^I \end{pmatrix} \\ K^I &= diag\,(K_0^I \quad \ldots \quad K_{p-1}^I) . \end{aligned} \qquad x = \begin{pmatrix} x^C \\ x^I \end{pmatrix} \qquad x^I = \begin{pmatrix} x_0^I \\ \vdots \\ x_{p-1}^I \end{pmatrix} \tag{2.6}$$

Diese Struktur überträgt sich ebenfalls auf die Matrizen A_s aus (2.2), die den Zusammenhang zwischen der jeweiligen lokalen Numerierung der Freiheitsgrade in $\overline{\Omega}_s$ und deren globaler Numerierung herstellen:

$$A_s = \begin{pmatrix} A_s^C & \mathbf{0} \\ \mathbf{0} & A_s^I \end{pmatrix} \tag{2.7}$$

mit $A_s^C \in R^{n_s^C, n^C}$ und $A_s^I \in R^{n_s^I, n^I}$. Durch Kombination von (2.5),(2.6) und (2.7) mit (2.2) werden zu (2.2) analoge Darstellungsformeln für die Teilblöcke von K erhalten:

$$\begin{aligned} K^C &= \sum_{s=0}^{p-1} (A_s^C)^T K_s^C A_s^C \qquad K^{CI} = \sum_{s=0}^{p-1} (A_s^C)^T K_s^{CI} A_s^I \\ K^{IC} &= \sum_{s=0}^{p-1} (A_s^I)^T K_s^{IC} A_s^C = \left((K_s^{IC} A_s^C) \right)_{s=0}^{p-1} . \end{aligned} \tag{2.8}$$

Die Block–Spaltenvektorstruktur der Matrix K^{IC} in (2.8) ergibt sich daraus, daß $((A_s^I)^T)_{ij} = 1$ genau dann, wenn der im Sinne der globalen Numerierung i-te Freiheitsgrad bei der lokalen Numerierung in Ω_s die Nummer j hat, und $((A_s^I)^T)_{ij} = 0$ in allen anderen Fällen.

Die Notation (2.6) benutzend ist

$$K = \begin{pmatrix} I & K^{CI}K^{-I} \\ \mathbf{0} & I \end{pmatrix} \begin{pmatrix} S & \mathbf{0} \\ \mathbf{0} & K^I \end{pmatrix} \begin{pmatrix} I & \mathbf{0} \\ K^{-I}K^{IC} & I \end{pmatrix} \tag{2.9}$$

mit der Schur-Komplement-Matrix

$$S = K^C - K^{CI}K^{-I}K^{IC} \tag{2.10}$$

und $K^{-I} := (K^I)^{-1}$ die bereits in [2] zur Beschreibung des Lösers benutzte Blockfaktorisierung der Matrix K.

Sind geeignete Vorkonditionierer V^C und $V^I = diag(V_0^I, ..., V_{p-1}^I)$ für die Teilblöcke S und K^I in (2.9) gefunden, so ist (vgl. [2]) die symmetrische Matrix

$$C = \begin{pmatrix} I & K^{CI}V^{-I} \\ \mathbf{0} & I \end{pmatrix} \begin{pmatrix} V^C & \mathbf{0} \\ \mathbf{0} & V^I \end{pmatrix} \begin{pmatrix} I & \mathbf{0} \\ V^{-I}K^{IC} & I \end{pmatrix} \tag{2.11}$$

ein symmetrischer, positiv definiter Vorkonditionierer für die Steifigkeitsmatrix K.

3 Realisierung des parallelen CG-Verfahrens

Sind ausgehend von einer Startlösung $x^{(0)}$ (oft: $x^{(0)} = \vec{0}$) die Anfangswerte

$$\begin{array}{lrcl} (1) & r^{(0)} & := & Kx^{(0)} - b \\ (2) & w^{(0)} & := & C^{-1}r^{(0)} \\ (3) & q^{(0)} & := & w^{(0)} \\ (4) & \gamma^{(0)} & := & (w^{(0)}, r^{(0)}) \end{array} \tag{3.1}$$

berechnet, ist der Iterationszyklus (3.2) für $i = 0, 1, \ldots$

$$\begin{array}{lrcl} (5) & u^{(i+1)} & := & Kq^{(i)} \\ (6) & \delta^{(i+1)} & := & (q^{(i)}, u^{(i+1)}) \\ (7) & \alpha^{(i+1)} & := & -\gamma^{(i)}/\delta^{(i+1)} \\ (8) & x^{(i+1)} & := & x^{(i)} + \alpha^{(i+1)}q^{(i)} \\ (9) & r^{(i+1)} & := & r^{(i)} + \alpha^{(i+1)}u^{(i+1)} \\ (10) & w^{(i+1)} & := & C^{-1}r^{(i+1)} \\ (11) & \gamma^{(i+1)} & := & (w^{(i+1)}, r^{(i+1)}) \\ (12) & \beta^{(i+1)} & := & \gamma^{(i+1)}/\gamma^{(i)} \\ (13) & q^{(i+1)} & := & w^{(i+1)} + \beta^{(i+1)}q^{(i)} \end{array} \tag{3.2}$$

so lange zu durchlaufen, bis die gewünschte Genauigkeit erreicht ist.

Bemerkung 3.1 *Werden die Zeilen (5) und (13) in (6) eingesetzt, ergibt sich die Beziehung* $\delta^{(i+1)} = (w^{(i)}, Kq^{(i)}) + \beta^{(i)}\,(q^{(i-1)}, Kq^{(i)}) = (w^{(i)}, Kq^{(i)})$ *, denn es ist* $(Kq^{(i)}, q^{(j)}) = 0 \quad \forall \quad i \neq j$ *(vgl.* [6]*). Nochmals (13) sowie (5) benutzend wird* *daraus* $\delta^{(i+1)} = (Kw^{(i)}, w^{(i)}) + \beta^{(i)}(u^{(i)}, w^{(i)})$ *erhalten. Wegen (9), (10) und (11) ist* $\alpha^{(i)}(u^{(i)}, w^{(i)}) = (r^{(i)}, w^{(i)}) - (r^{(i-1)}, w^{(i)}) = \gamma^{(i)} - (r^{(i-1)}, C^{-1}r^{(i)})$ *mit* $C^{-1}r^{(i)} = C^{-1}(r^{(i-1)} + \alpha^{(i)}u^{(i)}) = w^{(i-1)} + \alpha^{(i)}C^{-1}u^{(i)}$. *Hieraus folgt unter Verwendung von (5), (6), (7), (10), (11) und (13), daß* $(\, r^{(i-1)}\, ,\, C^{-1}r^{(i)}\,) = (\, r^{(i-1)}\, ,\, w^{(i-1)}\,) + \alpha^{(i)}\,(\, r^{(i-1)}\, ,\, C^{-1}u^{(i)}\,) = \gamma^{(i-1)} + \alpha^{(i)}\,(\, C^{-1}r^{(i-1)}\, ,\, u^{(i)}\,) = \gamma^{(i-1)}\left[1 - \frac{(\, w^{(i-1)}\, ,\, u^{(i)}\,)}{\delta^{(i)}}\right] = \alpha^{(i)}\left[\,(\, w^{(i-1)}\, ,\, u^{(i)}\,) - \delta^{(i)}\,\right] = \alpha^{(i)}\left[\,(\, w^{(i-1)}\, ,\, u^{(i)}\,) - (\, q^{(i-1)}\, ,\, u^{(i)}\,)\,\right] = \alpha^{(i)}\,(\, w^{(i-1)} - q^{(i-1)}\, ,\, Kq^{(i-1)}\,) = -\alpha^{(i)}\,\beta^{(i-1)}\,(\, q^{(i-2)}\, ,\, Kq^{(i-1)}\,) = 0$ *gilt. Werden diese Beziehungen ineinander eingesetzt, ergibt sich die Gleichung* $\delta^{(i+1)} = (Kw^{(i)}, w^{(i)}) + \frac{\beta^{(i)}\gamma^{(i)}}{\alpha^{(i)}}$. *Folglich kann die Größe* $\alpha^{(i+1)}$ *von (7) auch nach der Formel*

$$\alpha^{(i+1)} := \frac{-1}{\dfrac{(Kw^{(i)}, w^{(i)})}{\gamma^{(i)}} + \dfrac{\beta^{(i)}}{\alpha^{(i)}}} \tag{3.3}$$

berechnet werden. Die zweimalige Berechnung eines Matrix–Vektor–Produkts bei (5) und (3.3) kann dabei vermieden werden, wenn (5) durch

$$u^{(i+1)} := Kw^{(i)} + \beta^{(i)}u^{(i)} \tag{3.4}$$

ersetzt wird, denn mit $u^{(1)} := Kw^{(0)}$ *gilt* $u^{(i+1)} = Kq^{(i)} = K(w^{(i)} + \beta^{(i)}q^{(i-1)}) = Kw^{(i)} + \beta^{(i)}Kq^{(i-1)} = Kw^{(i)} + \beta^{(i)}u^{(i)}$. *Hiervon wird in den Abschnitten 3.2 und 4.1 Gebrauch gemacht werden.*

Wie in [2] werden das Residuum r sowie dessen Korrektur u wie die rechte Seite b summarisch verteilt abgespeichert (vgl.(2.2)), während das vorkonditionierte Residuum w und die Suchrichtung q wie die Lösung x nach (2.3) gespeichert sein sollen. Dann ergibt sich (die Iterationsindizees weglassend) folgendes für die Teilschritte (1) bis (13) aus (3.1) und (3.2):

Wegen

$$Kq = \sum_{s=0}^{p-1} A_s^T K_s A_s q = \sum_{s=0}^{p-1} A_s^T K_s q_s = \sum_{s=0}^{p-1} A_s^T u_s = u$$

und

$$Kx - b = \left(\sum_{s=0}^{p-1} A_s^T K_s A_s\right) x - \sum_{s=0}^{p-1} A_s^T b_s = \sum_{s=0}^{p-1} A_s^T\,(K_s x_s - b_s) = \sum_{s=0}^{p-1} A_s^T r_s = r$$

sind die Teilschritte (1) und (5) vollständig parallel und ohne Kommunikation ausführbar. Indem in jedem Prozessor $\mathcal{P}_s$ die Teil-Matrixprodukte $K_s q_s$ und $K_s x_s$ sowie die Vektordifferenz $(K_s x_s) - b_s$ berechnet werden, entsteht das Ergebnis in der gewollten Speicherform (2.2). Dies gilt entsprechend auch für Kw bei (3.4), da w und q vom gleichen Speichertyp sind. In Teilschritt (3) sind lediglich gleichartig gespeicherte Vektoren q und w lokal zu kopieren. Bei der Berechnung der globalen Skalarprodukte (w, r) und (q, u) in den Teilschritten (4),(6) und (11) sind Vektoren unterschiedlicher Speichertypen miteinander zu verknüpfen, jedoch ist

$$(w, r) = \left(w, \sum_{s=0}^{p-1} A_s^T r_s\right) = \sum_{s=0}^{p-1} (w, A_s^T r_s) = \sum_{s=0}^{p-1} (A_s w, r_s) = \sum_{s=0}^{p-1} (w_s, r_s) = \sum_{s=0}^{p-1} \gamma_s = \gamma$$

und analog dazu auch $(q,u) = \sum_{s=0}^{p-1} \delta_s = \delta$. Zur Berechnung dieser Zahlen sind also zunächst parallel in allen Prozessoren die lokalen Skalarprodukte $\gamma_s := (w_s, r_s)$ bzw. $\delta_s := (q_s, u_s)$ zu bestimmen, und diese sind dann global über alle Prozessoren aufzusummieren. Da γ und δ in allen Prozessoren benötigt werden, sind diese Ergebnisse allen $\mathcal{P}_s$ mitzuteilen [3]. In einem m-dimensionalen Hypercube ($p = 2^m$) sind zur Berechnung eines dieser globalen Skalarprodukte in allen p Prozessoren insgesamt nur $2m$ send- und receive–Operationen ausreichend ([2]). Damit können (7) und (12) in jedem Prozessor ausgeführt werden. In den Teilschritten (8),(9) und (13) sind Linearkombinationen von Vektoren jeweils gleichen Speichertyps zu bilden. Dies geschieht, indem in allen $\mathcal{P}_s$ die entprechenden Operationen mit den lokalen Teilvektoren ausgeführt werden. Damit ist - mit Ausnahme des Vorkonditionierungsschrittes (2) bzw. (10) - die Realisierung aller Teilschritte in (3.1) und (3.2) beschrieben.

3.1 Parallelisierte Vorkonditionierung

Selbst ohne jede Vorkonditionierung (statt C von (2.11) wird in (3.1) und (3.2) $C = I$ benutzt) ist die Berechnung des in der Form (2.3) gespeicherten Vektors w aus dem nach (2.2) gespeicherten Residuum r derjenige Teil des Iterationsverfahrens (3.1), (3.2) , der den größten Kommunikationsaufwand beinhaltet, denn zur Berechnung der Komponenten w_s^C (vgl.(2.4)) sind Komponenten von r_t^C aus allen Prozessoren $\mathcal{P}_t$, deren Subdomain $\overline{\Omega}_t$ die entsprechenden Koppelknoten enthält, aufzuakkumulieren:

$$w_s = A_s w = A_s r = \sum_{t=0}^{p-1} A_s A_t^T r_t = A_s A_s^T r_s + \sum_{t \neq s} A_s A_t^T r_t = r_s + \sum_{t \neq s} A_s A_t^T r_t \qquad (3.5)$$

In (3.5) ist $A_s A_t^T = \mathbf{0}$ wenn $\overline{\Omega}_s \cap \overline{\Omega}_t = \emptyset$, denn wegen $A_s^{ij} = 1$, wenn der in der globalen Numerierung j-te Freiheitsgrad bei der lokalen Numerierung in $\overline{\Omega}_s$ die Nummer i trägt und $A_s^{ij} = 0$ in allen anderen Fällen, ist $(A_s A_t^T)^{kl} = 1$, wenn es einen Freiheitgrad gibt, der bei der lokalen Numerierung in $\overline{\Omega}_s$ die Nummer k und in $\overline{\Omega}_t$ die Nummer l erhält, und sonst ist $(A_s A_t^T)^{kl} = 0$. Jeder Prozessor $\mathcal{P}_t$ muß folglich Teile seines Vektors r_t^C (praktischer Weise zusammen mit einigen Integer-Informationen über die Knotennumerierung) an all diejenigen Prozessoren $\mathcal{P}_s$ senden, deren Subdomains $\overline{\Omega}_s$ an $\overline{\Omega}_t$ angrenzen, und er muß von genau diesen Prozessoren $\mathcal{P}_s$ deren r_s^C-Anteile empfangen und anschließend (unter Verwendung dieser Integer-Informationen) zu seinen addieren. Die Größe der dabei auszutauschenden Datenpakete ist von der Ordnung $O(h^{-1})$ und die Anzahl der Adressaten $\mathcal{P}_s$ ist auch für große Prozessorzahlen eine nur von der vorgenommenen Gebietszerlegung abhängende kleine Zahl.

Mit den bei (2.8) bis (2.11) definierten Teilblöcken der Vorkonditionierungsmatrix C sind zur Lösung des Gleichungssystems $Cw = r$ in den Schritten (2) und (10)

[3] Hierzu eignen sich beispielsweise die in [1] dokumentierten Routinen tree_dod /tree_down oder cube_dod.

bei (3.1) bzw. (3.2) folgende drei Teilsysteme zu lösen:

$$\begin{array}{lrcl} (10\text{-}1) & V^I h^I & = & r^I \\ (10\text{-}2) & V^C w^C & = & h^C \\ \quad\text{mit} & h^C & := & r^C - K^{CI} h^I \\ (10\text{-}3) & V^I w^I & = & r^I - K^{IC} w^C \,. \end{array} \tag{3.6}$$

Da wie bereits dargelegt eine Interprozessorkommunikation nur für die Koppelfreiheitsgrade erforderlich ist ($V^I = diag(V_0^I, ..., V_{p-1}^I)$ und $r^I = (r_o^I, ..., r_{p-1}^I)^T$), kann (10-1) in allen Prozessoren gleichzeitig und voneinander unabhängig ausgeführt werden, indem in jedem Prozessor $\mathcal{P}_s$ das lokale Gleichungssystem $V_s^I h_s^I = r_s^I$ gelöst wird. Bei den Vektorsegmenten r^I und h^I muß zwischen den Darstellungsweisen (2.2) und (2.3) nicht unterschieden werden, weil sich diese Speicherformen nur in den Komponenten unterscheiden, die Koppelfreiheitsgraden entsprechen.

Liegt die Lösung w^C von (10-2) in der Speicherform (2.3) vor, so ist nach (2.8) $K^{IC} w^C = ((K_s^{IC} A_s^C w^C))_{s=0}^{p-1} = ((K_s^{IC} w_s^C))_{s=0}^{p-1}$. Deshalb kann unter dieser Voraussetzung auch (10-3) parallel und ohne Kommunikation ausgeführt werden, indem in jedem Prozessor $\mathcal{P}_s$ zunächst die Hilfsgröße $d_s^I := r_s^I - K_s^{IC} w_s^C$ berechnet und mit dieser dann das lokale Gleichungssystem $V_s^I w_s^I = d_s^I$ gelöst wird [4].

Nach (2.2), (2.3) und (2.6)–(2.8) ist

$$r^C = \sum_{s=0}^{p-1} (A_s^C)^T r_s^C \quad , \quad K^{CI} = \sum_{s=0}^{p-1} (A_s^C)^T K_s^{CI} A_s^I \quad , \quad A_s^I h^I = h_s^I$$

und daraus folgend

$$h^C := r^C - K^{CI} h^I = \sum_{s=0}^{p-1} (A_s^C)^T (r_s^C - K_s^{CI} h_s^I) = \sum_{s=0}^{p-1} (A_s^C)^T h_s^C \,.$$

Daher kann auch die rechte Seite von (10-2) parallel und kommunikationsfrei berechnet werden. Indem in jedem Prozessor $\mathcal{P}_s$ die lokale Größe

$$h_s^C := r_s^C - K_s^{CI} h_s^I \tag{3.7}$$

berechnet wird, entsteht der globale Vektor h^C in nach (2.2) summarisch verteilter Speicherform.

Bezeichnet V^{-C} die (nicht notwendigerweise explizit berechnete) Inverse zu V^C und soll, wie oben für den Teilschritt (10-3) vorausgesetzt, w^C in der Speicherform (2.3) erhalten werden, dann ist $w_s^C = A_s^C w^C = A_s^C V^{-C} h^C = A_s^C V^{-C} \sum_{t=0}^{p-1} (A_t^C)^T h_t^C = A_s^C \sum_{t=0}^{p-1} V^{-C} (A_t^C)^T h_t^C$ auszuwerten. Dazu ist in jedem Prozessor $\mathcal{P}_t$ die Lösung $\tilde{w}_t^C \in R^{n^C}$ des Gleichungssystems $V^C \tilde{w}_t^C = (A_t^C)^T h_t^C$ zu bestimmen, und diese Lösungen sind dann gemäß $w_s^C := A_s^C \sum_{t=0}^{p-1} \tilde{w}_t^C$ global zu assemblieren. Der dahinter stehende Aufwand, in jedem Prozessor ein Gleichungssystem der Dimension n^C zu lösen, die Lösung an alle (!) anderen Prozessoren zu senden und die entsprechenden

[4] Hierzu ist in FEAP momentan die Choleskyzerlegung der Matrizen $V_s^I := K_s^I$ mit anschließendem Rückwärts- und Vorwärtseinsetzen implementiert.

Teile aller empfangenen Lösungen zur in diesem Prozessor ermittelten Lösung zu addieren, kann etwas verringert werden, wenn V^{-C} explizit berechnet ist. Bezeichnet $\tilde{V}_t^{-C} := V^{-C}(A_t^C)^T \in R^{n^C, n_t^C}$ diejenige Matrix, die durch Streichen all derjenigen Spalten von V^{-C} entsteht, deren Nummer gleich der globalen Nummer eines nicht in $\overline{\Omega}_t$ liegenden Koppelfreiheitsgrades ist, dann kann $\tilde{w}_t^C$ gemäß $\tilde{w}_t^C = \tilde{V}_t^{-C} h_t^C$ berechnet werden. Beiden Varianten ist gemeinsam, daß in jedem Prozessor Größen von globaler Dimension zu behandeln sind: Entweder sind Gleichungssysteme der Dimension n^C zu lösen oder es sind Matrix–Vektor–Produkte mit $n^C \times n_t^C$–Matrizen zu bilden. Wie sich zeigt, kann dies vermieden werden, wenn die zunächst summarisch verteilt gespeicherte rechte Seite h^C von (10-2) vor der Lösung des Gleichungssystems $V^C w^C = h^C$ zu einem nach (2.3) gespeicherten Vektor global assembliert wird und der Vorkonditionierer V^C folgende Eigenschaft hat:

$$\boxed{\text{Mit der Matrix } B_s^C := (A_s^C)^T A_s^C \text{ gelte } A_s^C V^{-C} = A_s^C V^{-C} B_s^C \quad \forall s\, .} \tag{3.8}$$

Mit der Größe h_s^C von (3.7) sei durch

$$g_s^C := h_s^C + \sum_{t \neq s} A_s^C (A_t^C)^T h_t^C \tag{3.9}$$

die global Assemblierte zu h^C definiert, deren Berechnung den bei (3.5) beschriebenen Kommunikationsaufwand beinhaltet und für die $A_s^C h^C = g_s^C$ gilt. Dann ist $w_s^C = A_s^C w^C = A_s^C V^{-C} h^C = A_s^C V^{-C} B_s^C h^C = A_s^C V^{-C} (A_s^C)^T A_s^C h^C = V_s^{-C} g_s^C$ mit

$$V_s^{-C} := A_s^C V^{-C} (A_s^C)^T \, . \tag{3.10}$$

Wegen $(A_s^C)^{ij} = 1$ genau dann, wenn der Koppelfreiheitsgrad mit der globalen Nummer j bei der lokalen Numerierung in $\overline{\Omega}_s$ die Nummer i erhält und $(A_s^C)^{ij} = 0$ in allen anderen Fällen, ist B_s^C eine Diagonalmatrix mit $(B_s^C)^{ii} = 1$, wenn der global i-te Freiheitsgrad in $\overline{\Omega}_s$ liegt, und $(B_s^C)^{ii} = 0$ für alle anderen Indizees i. Folglich werden durch $A_s^C V^{-C} B_s^C$ in der Matrix $A_s^C V^{-C}$ genau die Spalten Null gesetzt, deren Nummer gleich der globalen Nummer eines nicht zu $\overline{\Omega}_s$ gehörenden Freiheitsgrads ist. Die Multiplikation der Matrix V^{-C} von links mit A_s^C bewirkt, daß in V^{-C} alle Zeilen gestrichen werden, deren Nummer gleich der globalen Nummer eines nicht zu $\overline{\Omega}_s$ gehörenden Freiheitsgrads ist und daß die verbleibenden Zeilen der lokalen Numerierung in $\overline{\Omega}_s$ entsprechend umnumeriert werden. Die Bedingung (3.8) ist also genau dann erfüllt, wenn $A_s^C V^{-C}$ keine Nichtnullelemente besitzt, die bei der Multiplikation mit h^C auf Freiheitsgrade zugreifen, die nicht zum Prozessor $\mathcal{P}_s$ gehören. Das bedeutet, daß die Komponenten von w_s^C im Gleichungssystem $V^C w^C = h^C$ nur von den Komponenten g_s^C abhängen dürfen. Also muß V^{-C} bei geeigneter Numerierung der Freiheitsgrade eine Blockdiagonalmatrix sein, deren Teilblöcke durch die jeweils im Inneren der Randstücke von Γ_s liegenden Koppelfreiheitsgrade bestimmt sind, und die Crosspointfreiheitsgrade dürfen in V^{-C} nur Diagonaleinträge besitzen. Diese Struktur muß dann natürlich auch die Matrix V^C besitzen.

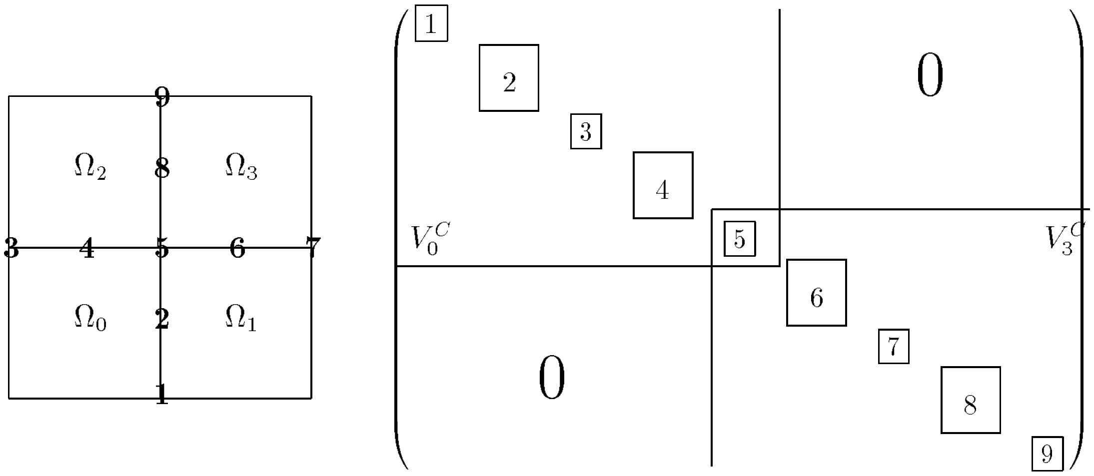

Abbildung 1: Blockstruktur von V^C bei 4 Teilgebieten

In dem Beispiel aus Abbildung 1 werden die Teilmatrizen

$$V_s^C := A_s^C V^C (A_s^C)^T \tag{3.11}$$

durch den rechten Teil dieser Abbildung und durch (3.12) beschrieben:

$$V_1^C = \begin{pmatrix} \boxed{1} & & & & \\ & \boxed{2} & & & \\ & & \boxed{5} & & \\ & & & \boxed{6} & \\ & & & & \boxed{7} \end{pmatrix} \qquad V_2^C = \begin{pmatrix} \boxed{3} & & & & \\ & \boxed{4} & & & \\ & & \boxed{5} & & \\ & & & \boxed{8} & \\ & & & & \boxed{9} \end{pmatrix} \tag{3.12}$$

Bei der Formulierung der Bedingung (3.8) und in Formel (3.10) wird implizit davon ausgegangen, daß die Matrix V^{-C} explizit berechnet worden ist. Zur praktischen Realisierung ist es jedoch ausreichend, die lokalen Matrizen V_s^{-C} zu bilden.

Ist die Bedingung (3.8) erfüllt, dann ist $V_s^C V_s^{-C} = A_s^C V^C (A_s^C)^T A_s^C V^{-C} (A_s^C)^T = A_s^C V^C B_s^C V^{-C} (A_s^C)^T = A_s^C V^C V^{-C} (A_s^C)^T = A_s^C I^C (A_s^C)^T = I_s^C$, d.h., die Matrizen V_s^C von (3.11) und V_s^{-C} von (3.10) sind tatsächlich zueinander invers. Folglich ist die Invertierung von V^C ideal parallelisierbar. Indem in jedem Prozessor $\mathcal{P}_s$ das Gleichungssystem $V_s^C w_s^C = g_s^C$ gelöst wird, entsteht w^C in der Speicherform (2.3). Die Matrix V_s^C entsteht aus V^C, indem in V^C all diejenigen Zeilen und Spalten gestrichen werden, deren Nummer gleich der globalen Nummer eines Freiheitsgrades ist, der nicht zu $\overline{\Omega}_s$ gehört, und die Reihenfolge der verbliebenen Zeilen und Spalten mit der lokalen Numerierung in $\overline{\Omega}_s$ in Einklang gebracht wird. Bei geeigneter lokaler Numerierung in $\overline{\Omega}_s$ sind die Matrizen V_s^C selbst wieder Blockdiagonalmatrizen. Siehe dazu auch Abschnitt 4.2.

Durch die Struktur von V^C ist gesichert, daß in jedem Koppelpunkt von allen Prozessoren $\mathcal{P}_s$, in deren Teilgebiet $\overline{\Omega}_s$ der Koppelpunkt liegt, die gleiche Lösung berechnet wird, weil jeweils das gleiche Teilgleichungssystem gelöst wird (z.B. in den Prozessoren $\mathcal{P}_0$ und $\mathcal{P}_2$ das Gleichungssystem mit der Matrix $\boxed{4}$ aus Abbildung 1 zur Bestimmung der auf der mit "4" markierten Linie liegenden Freiheitgrade).

3.2 Prozessororientierte Betrachtung des CG-Verfahrens

Zur Realisierung der Schritte (1)–(13) in (3.1) und (3.2) sind lokal im Prozessor $\mathcal{P}_s$ folgende Operationen auszuführen:

$$
\begin{array}{lrcl}
(1) & r_s^{(0)} & := & K_s x_s^{(0)} - b_s = \begin{pmatrix} K_s^C x_s^{(0)C} + K_s^{CI} x_s^{(0)I} \\ K_s^{IC} x_s^{(0)C} + K_s^I x_s^{(0)I} \end{pmatrix} - \begin{pmatrix} b_s^C \\ b_s^I \end{pmatrix} =: \begin{pmatrix} r_s^{(0)C} \\ r_s^{(0)I} \end{pmatrix} \\
(2) & w_s^{(0)} & : & \text{wie in Schritt (10) bestimmen} \\
(3) & q_s^{(0)} & := & w_s^{(0)} = \begin{pmatrix} w_s^{(0)C} \\ w_s^{(0)I} \end{pmatrix} =: \begin{pmatrix} q_s^{(0)C} \\ q_s^{(0)I} \end{pmatrix} \\
(4) & \gamma_s^{(0)} & := & (w_s^{(0)}, r_s^{(0)}) = (w_s^{(0)C}, r_s^{(0)C}) + (w_s^{(0)I}, r_s^{(0)I}) \\
 & \gamma^{(0)} & := & \sum\limits_{s=0}^{p-1} \gamma_s^{(0)} \qquad \Longleftarrow \underline{\textsf{globale Kommunikation}} \\
 & & & \qquad i := 0
\end{array}
$$

$$
\begin{array}{lrcl}
(5) & u_s^{(i+1)} & := & K_s q_s^{(i)} = \begin{pmatrix} K_s^C q_s^{(i)C} + K_s^{CI} q_s^{(i)I} \\ K_s^{IC} q_s^{(i)C} + K_s^I q_s^{(i)I} \end{pmatrix} =: \begin{pmatrix} u_s^{(i+1)C} \\ u_s^{(i+1)I} \end{pmatrix} \\
(6) & \delta_s^{(i+1)} & := & (q_s^{(i)}, u_s^{(i+1)}) = (q_s^{(i)C}, u_s^{(i+1)C}) + (q_s^{(i)I}, u_s^{(i+1)I}) \\
 & \delta^{(i+1)} & := & \sum\limits_{s=0}^{p-1} \delta_s^{(i+1)} \qquad \Longleftarrow \underline{\textsf{globale Kommunikation}} \\
(7) & \alpha^{(i+1)} & := & -\,\gamma^{(i)}/\delta^{(i+1)} \\
(8) & x_s^{(i+1)} & := & x_s^{(i)} + \alpha^{(i+1)} q_s^{(i)} = \begin{pmatrix} x_s^{(i)C} \\ x_s^{(i)I} \end{pmatrix} + \alpha^{(i+1)} \begin{pmatrix} q_s^{(i)C} \\ q_s^{(i)I} \end{pmatrix} =: \begin{pmatrix} x_s^{(i+1)C} \\ x_s^{(i+1)I} \end{pmatrix} \\
(9) & r_s^{(i+1)} & := & r_s^{(i)} + \alpha^{(i+1)} u_s^{(i+1)} = \begin{pmatrix} r_s^{(i)C} \\ r_s^{(i)I} \end{pmatrix} + \alpha^{(i+1)} \begin{pmatrix} u_s^{(i+1)C} \\ u_s^{(i+1)I} \end{pmatrix} =: \begin{pmatrix} r_s^{(i+1)C} \\ r_s^{(i+1)I} \end{pmatrix} \\
(10) & \text{löse} & & V_s^I h_s^{(i+1)I} = r_s^{(i+1)I} \\
 & h_s^{(i+1)C} & := & r_s^{(i+1)C} - K_s^{CI} h_s^{(i+1)I} \\
 & g_s^{(i+1)C} & := & h_s^{(i+1)C} + \sum\limits_{t \neq s} A_s^C (A_t^C)^T h_t^{(i+1)C} \qquad \Longleftarrow \underline{\textsf{globale Kommunikation}} \\
 & \text{löse} & & V_s^C w_s^{(i+1)C} = g_s^{(i+1)C} \\
 & d_s^{(i+1)I} & := & r_s^{(i+1)I} - K_s^{IC} w_s^{(i+1)C} \\
 & \text{löse} & & V_s^I w_s^{(i+1)I} = d_s^{(i+1)I} \\
(11) & \gamma_s^{(i+1)} & := & (w_s^{(i+1)}, r_s^{(i+1)}) = (w_s^{(i+1)C}, r_s^{(i+1)C}) + (w_s^{(i+1)I}, r_s^{(i+1)I}) \\
 & \gamma^{(i+1)} & := & \sum\limits_{s=0}^{p-1} \gamma_s^{(i+1)} \qquad \Longleftarrow \underline{\textsf{globale Kommunikation}} \\
(12) & \beta^{(i+1)} & := & \gamma^{(i+1)}/\gamma^{(i)} \\
(13) & q_s^{(i+1)} & := & w_s^{(i+1)} + \beta^{(i+1)} q_s^{(i)} = \begin{pmatrix} w_s^{(i+1)C} \\ w_s^{(i+1)I} \end{pmatrix} + \beta^{(i+1)} \begin{pmatrix} q_s^{(i)C} \\ q_s^{(i)I} \end{pmatrix} =: \begin{pmatrix} q_s^{(i+1)C} \\ q_s^{(i+1)I} \end{pmatrix} \\
 & & & \qquad i := i+1 \\
 & & & \quad \Longrightarrow (5) \quad \text{oder} \quad \text{STOP}
\end{array}
\tag{3.13}
$$

Neben der globalen Assemblierung in Schritt (10) sind in jedem Iterationsschritt von (3.13) noch zwei weitere globale Kommunikationen zur Berechnung der Zahlen $\delta^{(i+1)}$ und $\gamma^{(i+1)}$ erforderlich. Unter Verwendung von (3.3) und (3.4) aus Bemerkung 3.1 ist es möglich, diese zu einer Kommunikation zusammenzufassen:

$$
\begin{array}{lrcl}
(0) & u_s^{(0)} & := & 0 = \begin{pmatrix} 0 \\ 0 \end{pmatrix} =: \begin{pmatrix} u_s^{(0)C} \\ u_s^{(0)I} \end{pmatrix} \\
 & \beta^{(0)} & := & 0 \qquad \alpha^{(0)} := 1 \\
(1) & r_s^{(0)} & := & K_s x_s^{(0)} - b_s \qquad \text{(vgl. (3.13))} \\
(2) & w_s^{(0)} & : & \text{wie in Schritt (10) bestimmen} \\
(3) & q_s^{(0)} & := & w_s^{(0)} \qquad \text{(vgl. (3.13))} \\
 & h_s^{(0)} & := & K_s w_s^{(0)} = \begin{pmatrix} K_s^C w_s^{(0)C} + K_s^{CI} w_s^{(0)I} \\ K_s^{IC} w_s^{(0)C} + K_s^I w_s^{(0)I} \end{pmatrix} =: \begin{pmatrix} h_s^{(0)C} \\ h_s^{(0)I} \end{pmatrix} \\
(4) & \gamma_s^{(0)} & := & (w_s^{(0)}, r_s^{(0)}) = (w_s^{(0)C}, r_s^{(0)C}) + (w_s^{(0)I}, r_s^{(0)I}) \\
 & \tau_s^{(0)} & := & (w_s^{(0)}, h_s^{(0)}) = (w_s^{(0)C}, h_s^{(0)C}) + (w_s^{(0)I}, h_s^{(0)I}) \\
 & \gamma^{(0)} & := & \sum_{s=0}^{p-1} \gamma_s^{(0)} \qquad \tau^{(0)} := \sum_{s=0}^{p-1} \tau_s^{(0)} \qquad \Longleftarrow \underline{\textsf{globale Kommunikation}} \\
 & & & i := 0 \\
\hline
(5) & u_s^{(i+1)} & := & h_s^{(i)} + \beta^{(i)} u_s^{(i)} = \begin{pmatrix} h_s^{(i)C} \\ h_s^{(i)I} \end{pmatrix} + \beta^{(i)} \begin{pmatrix} u_s^{(i)C} \\ u_s^{(i)I} \end{pmatrix} =: \begin{pmatrix} u_s^{(i+1)C} \\ u_s^{(i+1)I} \end{pmatrix} \\
(7) & \alpha^{(i+1)} & := & - \left[\frac{\tau^{(i)}}{\gamma^{(i)}} + \frac{\beta^{(i)}}{\alpha^{(i)}} \right]^{-1} \\
(8) & x_s^{(i+1)} & := & x_s^{(i)} + \alpha^{(i+1)} q_s^{(i)} \qquad \text{(vgl. (3.13))} \\
(9) & r_s^{(i+1)} & := & r_s^{(i)} + \alpha^{(i+1)} u_s^{(i+1)} \qquad \text{(vgl. (3.13))} \\
(10) & \multicolumn{3}{l}{\text{identisch mit (10) von (3.13)} \qquad \Longleftarrow \underline{\textsf{globale Kommunikation}}} \\
(11) & h_s^{(i+1)} & := & K_s w_s^{(i+1)} = \begin{pmatrix} K_s^C w_s^{(i+1)C} + K_s^{CI} w_s^{(i+1)I} \\ K_s^{IC} w_s^{(i+1)C} + K_s^I w_s^{(i+1)I} \end{pmatrix} =: \begin{pmatrix} h_s^{(i+1)C} \\ h_s^{(i+1)I} \end{pmatrix} \\
 & \gamma_s^{(i+1)} & := & (w_s^{(i+1)}, r_s^{(i+1)}) = (w_s^{(i+1)C}, r_s^{(i+1)C}) + (w_s^{(i+1)I}, r_s^{(i+1)I}) \\
 & \tau_s^{(i+1)} & := & (w_s^{(i+1)}, h_s^{(i+1)}) = (w_s^{(i+1)C}, h_s^{(i+1)C}) + (w_s^{(i+1)I}, h_s^{(i+1)I}) \\
 & \gamma^{(i+1)} & := & \sum_{s=0}^{p-1} \gamma_s^{(i+1)} \qquad \tau^{(i+1)} := \sum_{s=0}^{p-1} \tau_s^{(i+1)} \quad \Longleftarrow \underline{\textsf{globale Kommunikation}} \\
(12) & \beta^{(i+1)} & := & \gamma^{(i+1)} / \gamma^{(i)} \\
(13) & q_s^{(i+1)} & := & w_s^{(i+1)} + \beta^{(i+1)} q_s^{(i)} \qquad \text{(vgl. (3.13))} \\
 & & & i := i+1 \\
 & & & \Longrightarrow (5) \quad \text{oder} \quad \text{STOP}
\end{array}
\tag{3.14}
$$

Statt zwei globale Kommunikationen mit jeweils einer reellen Zahl, wie bei (3.13) in den Schritten(6) und (11) erforderlich, muß im Algorithmus (3.14) nur eine globale Kommunikation mit zwei rellen Zahlen (Schritt (11)) ausgeführt werden[5]. Der Kommunikationsaufwand in Schritt (10) ist in beiden Varianten gleich.

[5]Indem die Zahlen γ und τ als Komponenten eines Vektors der Länge 2 gespeichert werden.

4 Eine spezielle Vorkonditionierungsmethode

In diesem Abschnitt soll ein spezielles Verfahren zur Lösung des Gleichungssystems $Cw = r$ von (3.1) bzw. (3.2) vorgestellt werden, welches derzeit in FEAP implementiert ist. Dazu werden die Vorkonditionierungsmatrizen V^I und V^C von (2.11) beschrieben.

4.1 Vorkonditionierung im Inneren der Teilgebiete

In den Teilschritten (2) und (10) des Algorithmus (3.13) sind in jedem Prozessor $\mathcal{P}_s$ jeweils zwei Gleichungssysteme mit der Systemmatrix V_s^I von (2.11) zu lösen. Durch geeignete Modifikation des Startvektors x kann dieser Berechnungsaufwand mehr als halbiert werden, wenn $V_s^I = K_s^I$ gewählt wird. Zugleich vermindert sich auch der übrige Berechnungsaufwand je Iterationsschritt beträchtlich und der Speicherbedarf für die Hilfsvektoren innerhalb der CG-Iteration sinkt um wenigstens 2 $(n_s^I - n_s^C)$ reelle Zahlen:

Sei $\overline{x}_s = (\,\overline{x}_s^C \quad \overline{x}_s^I\,)^T$ der Startvektor für die CG–Iteration, $\overline{r}_s^I := (b_s - K_s\overline{x}_s)^I = b_s^I - K_s^{IC}\overline{x}_s^C - K_s^I\overline{x}_s^I$, $\quad V_s^I := K_s^I$ mit K_s^I von (2.5) und $h_s^I := K_s^{-I}\overline{r}_s^I$ die Lösung des Gleichungssystems $K_s^I h_s^I = \overline{r}_s^I$. Wird das Iterationsverfahren (3.13) mit dem neuen Startvektor

$$x_s^{(0)} = \begin{pmatrix} x_s^{(0)C} \\ x_s^{(0)I} \end{pmatrix} := \begin{pmatrix} \overline{x}_s^C \\ \overline{x}_s^I + h_s^I \end{pmatrix} \tag{4.1}$$

betrachtet, so sind wegen $K_s^I x_s^{(0)I} = K_s^I(\overline{x}_s^I + h_s^I) = K_s^I(\overline{x}_s^I + K_s^{-I}\overline{r}_s^I) =$ $K_s^I\overline{x}_s^I + b_s^I - K_s^{IC}\overline{x}_s^C - K_s^I\overline{x}_s^I = b_s^I - K_s^{IC}\overline{x}_s^C = b_s^I - K_s^{IC}x_s^{(0)C}$ in (3.13 (1)) die Vektorkomponenten $r_s^{(0)I} = K_s^{IC}x_s^{(0)C} + K_s^I x_s^{(0)I} - b_s^I = 0$ sämtlich gleich Null. Damit entfällt innerhalb des Teilschritts (3.13 (2)) der Schritt " löse $V_s^I h_s^{(\cdot)I} = r_s^{(\cdot)I}$ ", denn es ist $h_s^{(\cdot)I} = 0$. Dies vereinfacht den zweiten und den vorletzten Teilschritt von (3.13 (2)) zu $h_s^{(\cdot)C} := r_s^{(\cdot)C}$ bzw. $d_s^{(\cdot)I} := -K_s^{IC}w_s^{(\cdot)C}$. Alle anderen Teilschritte von (3.13 (2)) bleiben unverändert. Die Operation $\gamma_s^{(0)} := (w_s^{(0)C}, r_s^{(0)C}) + (w_s^{(0)I}, r_s^{(0)I})$ in (3.13 (4)) reduziert sich auf $\gamma_s^{(0)} := (w_s^{(0)C}, r_s^{(0)C})$, da $r_s^{(0)I} = 0$ erhalten wurde. Im Teilschritt (5) von (3.13) muß lediglich $u_s^{(\cdot)C}$ berechnet werden, denn für $i = 0$ ist $u_s^{(i+1)I} =$ $K_s^{IC}q_s^{(i)C} + K_s^I q_s^{(i)I} = K_s^{IC}w_s^{(0)C} + K_s^I w_s^{(0)I} = 0$, weil $K_s^I w_s^{(0)I} = K_s^I V_s^{-I} d_s^{(0)I} = d_s^{(0)I} =$ $-K_s^{IC}w_s^{(0)C}$ gilt. In Folge dessen verschwindet der Summand $(q_s^{(\cdot)I}, u_s^{(\cdot)I})$ in (3.13 (6)), und in (3.13 (9)) ist $r_s^{(i+1)I} := r_s^{(i)I} + \alpha^{(i+1)}u_s^{(i+1)I} = \alpha^{(i+1)}u_s^{(i+1)I} = 0$, so daß zum einen die Eigenschaft $r_s^{(\cdot)I} = 0$, die durch den Vorbereitungsschritt (4.1) erzeugt werden konnte, erhalten bleibt und zum anderen ist auch in Teilschritt (3.13 (11)) $\gamma_s^{(\cdot)} = (w_s^{(\cdot)C}, r_s^{(\cdot)C})$. Für die in (3.13 (10)) auszuführenden Operationen gilt das zu Teilschritt (3.13 (2)) bereits gesagte sinngemäß weiter. Um alle bisher gewonnenen Vereinfachungen für alle Iterationsschritte zu sichern muß noch untersucht werden, ob die Eigenschaft $u_s^{(\cdot)I} = 0$ auch mit dem in Schritt (3.13 (13)) berechneten Vektor $q_s^{(\cdot)}$ erhalten wird, denn nur dann wird die Eigenschaft $r_s^{(\cdot)I} = 0$ durch die Iteration in Schritt (3.13 (9)) nicht zerstört. Nach (3.13 (5),(10),(13)) ist $q_s^{(i)} = w_s^{(i)} + \beta^{(i)}q_s^{(i-1)}$, $u_s^{(i)I} = K_s^{IC}q_s^{(i-1)C} + K_s^I q_s^{(i-1)I}$, $q_s^{(i)C} - w_s^{(i)C} = \beta^{(i)}q_s^{(i-1)C}$, $w_s^{(i)I} = -V_s^{-I}K_s^{IC}w_s^{(i)C}$ und $u_s^{(i+1)I} = K_s^{IC}q_s^{(i)C} + K_s^I q_s^{(i)I} = K_s^{IC}q_s^{(i)C} + K_s^I(w_s^{(i)I} + \beta^{(i)}q_s^{(i-1)I}) = K_s^{IC}q_s^{(i)C} +$ $K_s^I(\beta^{(i)}q_s^{(i-1)I} - V_s^{-I}K_s^{IC}w_s^{(i)C}) = K_s^{IC}(q_s^{(i)C} - w_s^{(i)C}) + \beta^{(i)}K_s^I q_s^{(i-1)I} = \beta^{(i)}(K_s^{IC}q_s^{(i-1)C} +$

$K_s^I q_s^{(i-1)I}) = \beta^{(i)} u_s^{(i)I}$. Da für das erstmalige Abarbeiten der Operation (3.13 (5)) bereits $u_s^{(\cdot)I} = 0$ erhalten wurde, ist folglich $u_s^{(\cdot)I} = 0$ in jedem Iterationsschritt gesichert. Schließlich können auch in den Schritten (8) und (13) von (3.13) die sich auf die inneren Freiheitsgrade beziehenden Operationen eingespart werden:
Aus der Rekursion $q_s^{(i)} := w_s^{(i)} + \beta^{(i)} q_s^{(i-1)}$ mit $q_s^{(0)} := w_s^{(0)}$ und $\beta^{(i)} := \gamma^{(i)}/\gamma^{(i-1)}$ folgt die Beziehung $q_s^{(m)} = \sum_{i=0}^{m} \frac{\gamma^{(m)}}{\gamma^{(i)}} w_s^{(i)}$. Aus der letzten Zeile von (3.13 (10)) ergibt sich (wegen $r_s^{(\cdot)I} = 0$) $w_s^{(i)I} = -K_s^{-I} K_s^{IC} w_s^{(i)C}$, woraus $q_s^{(m)I} = -K_s^{-I} K_s^{IC} \sum_{i=0}^{m} \frac{\gamma^{(m)}}{\gamma^{(i)}} w_s^{(i)C} = -K_s^{-I} K_s^{IC} q_s^{(m)C}$ für jeden Iterationsschritt m folgt. Werden die im Verlauf der Iteration berechneten $\alpha^{(i+1)} q_s^{(i)C}$ nicht sofort zu $x_s^{(\cdot)C}$ addiert sondern zunächst auf einem Hilfsvektor $\xi_s^{(\cdot)C}$ akkumuliert, kann daraus wegen $\sum_i \alpha^{(i+1)} q_s^{(i)I} = -K_s^{-I} K_s^{IC} \sum_i \alpha^{(i+1)} q_s^{(i)C} = -K_s^{-I} K_s^{IC} \xi_s^{(\cdot)C}$ nach Abschluß der Iteration die Korrektur von $x_s^{(\cdot)I}$ berechnet werden, womit die Akkumulation der $x_s^{(\cdot)I}$-Werte bei (3.13 (8)) überflüssig geworden ist. Aus (3.13 (5)) und (3.13 (13)) folgt ferner, daß $u_s^{(i+1)C} = K_s^C (w_s^{(i)C} + \beta^{(i)} q_s^{(i-1)C}) + K_s^{CI} (w_s^{(i)I} + \beta^{(i)} q_s^{(i-1)I}) = \beta^{(i)} (K_s^C q_s^{(i-1)C} + K_s^{CI} q_s^{(i-1)I}) + K_s^C w_s^{(i)C} + K_s^{CI} w_s^{(i)I} = \beta^{(i)} u_s^{(i)C} + K_s^C w_s^{(i)C} + K_s^{CI} w_s^{(i)I}$ gilt (vgl. auch (3.4)). Demnach müssen auch die $q_s^{(\cdot)I}$–Werte bei (3.13 (13)) nicht explizit berechnet werden und statt der Initialisierung von $q_s^{(0)I}$ bei (3.13 (3)) sind die Ausgangswerte $\beta^{(0)} = 0$ und $u_s^{(0)C} = 0$ zu setzen. Insgesamt reduziert sich das Verfahren (3.13) zu

$$
\begin{array}{llcl}
(0) & u_s^{(0)C} & := & 0 \qquad \xi_s^{(0)C} := 0 \qquad \beta^{(0)} := 0 \\
 & r_s^{(0)I} & := & b_s^I - K_s^{IC} x_s^{(0)C} - K_s^I x_s^{(0)I} \\
 & \text{löse} & & K_s^I h_s^{(0)I} = r_s^{(0)I} \\
 & x_s^{(1)I} & := & x_s^{(0)I} + h_s^{(0)I} \\
(1) & r_s^{(0)C} & := & K_s^C x_s^{(0)C} + K_s^{CI} x_s^{(1)I} - b_s^C \\
(2) & w_s^{(0)} & : & \text{wie in Schritt (10) bestimmen} \\
(3) & q_s^{(0)C} & := & w_s^{(0)C} \\
 & h_s^{(0)C} & := & K_s^C w_s^{(0)C} + K_s^{CI} w_s^{(0)I} \\
(4) & \gamma_s^{(0)} & := & (w_s^{(0)C}, r_s^{(0)C}) \\
 & \gamma^{(0)} & := & \sum_{s=0}^{p-1} \gamma_s^{(0)} \qquad \Longleftarrow \underline{\text{globale Kommunikation}} \\
 & & & \qquad i := 0 \\
\hline
(5) & u_s^{(i+1)C} & := & h_s^{(i)C} + \beta^{(i)} u_s^{(i)C} \\
(6) & \delta_s^{(i+1)} & := & (q_s^{(i)C}, u_s^{(i+1)C}) \\
 & \delta^{(i+1)} & := & \sum_{s=0}^{p-1} \delta_s^{(i+1)} \qquad \Longleftarrow \underline{\text{globale Kommunikation}} \\
(7) & \alpha^{(i+1)} & := & -\gamma^{(i)}/\delta^{(i+1)} \\
(8) & \xi_s^{(i+1)C} & := & \xi_s^{(i)C} + \alpha^{(i+1)} q_s^{(i)C} \\
(9) & r_s^{(i+1)C} & := & r_s^{(i)C} + \alpha^{(i+1)} u_s^{(i+1)C}
\end{array}
\tag{4.2}
$$

$$
\begin{array}{lrcll}
(10) & g_s^{(i+1)C} & := & r_s^{(i+1)C} + \sum\limits_{t\neq s} A_s^C (A_t^C)^T r_t^{(i+1)C} & \Longleftarrow \underline{\textsf{globale Kommunikation}} \\
 & \text{löse} & & V_s^C w_s^{(i+1)C} = g_s^{(i+1)C} & \\
 & d_s^{(i+1)I} & := & -K_s^{IC} w_s^{(i+1)C} & \\
 & \text{löse} & & K_s^I w_s^{(i+1)I} = d_s^{(i+1)I} & \\
(11) & h_s^{(i+1)C} & := & K_s^C w_s^{(i+1)C} + K_s^{CI} w_s^{(i+1)I} & \\
 & \gamma_s^{(i+1)} & := & (w_s^{(i+1)C}, r_s^{(i+1)C}) & \\
 & \gamma^{(i+1)} & := & \sum\limits_{s=0}^{p-1} \gamma_s^{(i+1)} & \Longleftarrow \underline{\textsf{globale Kommunikation}} \\
(12) & \beta^{(i+1)} & := & \gamma^{(i+1)} / \gamma^{(i)} & \\
(13) & q_s^{(i+1)C} & := & w_s^{(i+1)C} + \beta^{(i+1)} q_s^{(i)C} & \\
 & & & \qquad i := i+1 & \\
 & & & \Longrightarrow (5) \quad \text{oder} \quad \Longrightarrow (14) & \\
\hline
(14) & x_s^{(i)C} & := & x_s^{(0)C} + \xi_s^{(i)C} & \\
(15) & d_s^{(i)I} & := & K_s^{IC} \xi_s^{(i)C} & \\
(16) & \text{löse} & & K_s^I \xi_s^{(i)I} = d_s^{(i)I} & \\
(17) & x_s^{(i)I} & := & x_s^{(1)I} + \xi_s^{(i)I} & \qquad \text{STOP}
\end{array}
\tag{4.2}
$$

Bei der Realisierung des Algorithmus (4.2) sind, wie auch bei (3.13), neben der Kommunikation im Teilschritt (10) noch zwei weitere Kommunikationen zur Berechnung der globalen Skalarprodukte erforderlich. Wird, wie bereits bei (3.14) geschehen, Formel (3.3) zur Berechnung der $\alpha^{(i)}$ benutzt, kann der Algorithmus (4.2) auch folgendermaßen notiert werden:

$$
\begin{array}{llcll}
(0) & \multicolumn{3}{l}{\text{identisch mit (0) von (4.2) und } \alpha^{(0)} := 1} & \\
\multicolumn{4}{l}{(1) - (3) \text{ identisch mit } (1) - (3) \text{ von } (4.2)} & \\
(4) & \gamma_s^{(0)} & := & (w_s^{(0)C}, r_s^{(0)C}) & \\
 & \tau_s^{(0)} & := & (w_s^{(0)C}, h_s^{(0)C}) & \\
 & \gamma^{(0)} & := & \sum\limits_{s=0}^{p-1} \gamma_s^{(0)} \qquad \tau^{(0)} := \sum\limits_{s=0}^{p-1} \tau_s^{(0)} & \Longleftarrow \underline{\textsf{globale Kommunikation}} \\
 & & & \qquad i := 0 & \\
\hline
(5) & \multicolumn{3}{l}{\text{identisch mit (5) von (4.2)}} & \\
(7) & \multicolumn{3}{l}{\text{identisch mit (7) von (3.14)}} & \\
\multicolumn{4}{l}{(8) - (9) \text{ identisch mit } (8) - (9) \text{ von } (4.2)} & \\
(10) & \multicolumn{3}{l}{\text{identisch mit (10) von (4.2)}} & \Longleftarrow \underline{\textsf{globale Kommunikation}}
\end{array}
\tag{4.3}
$$

$$
\begin{array}{lrcl}
(11) & h_s^{(i+1)C} & := & K_s^C w_s^{(i+1)C} + K_s^{CI} w_s^{(i+1)I} \\
& \gamma_s^{(i+1)} & := & (w_s^{(i+1)C}, r_s^{(i+1)C}) \\
& \tau_s^{(i+1)} & := & (w_s^{(i+1)C}, h_s^{(i+1)C}) \\
& \gamma^{(i+1)} & := & \sum_{s=0}^{p-1} \gamma_s^{(i+1)} \quad \tau^{(i+1)} := \sum_{s=0}^{p-1} \tau_s^{(i+1)} \quad \Longleftarrow \underline{\textsf{globale Kommunikation}}
\end{array}
$$

(12) – (13) identisch mit (12) – (13) von (4.2)

$$i := i + 1$$

$$\Longrightarrow (5) \quad \text{oder} \quad \Longrightarrow (14)$$

---

(14) – (17) identisch mit (14) – (17) von (4.2)

$$\text{STOP} \tag{4.3}$$

Bei der Ausführung der durch (4.2) und (4.3) beschriebenen Algorithmen kann für die Vektoren h_s^I und d_s^I der gleiche Speichervektor a_s^I benutzt werden. Ebenso kann für die Vektoren ξ_s^I , w_s^I und r_s^I ein gemeinsames Speicherfeld $\underline{a}_s^I$ Verwendung finden.

Die Matrix K^I von (2.6) ist eine Blockdiagonalmatrix: $K^I = diag(K_s^I)$. Deshalb erzeugt die Choleskyzerlegung der Matrizen K_s^I zugleich eine Choleskyzerlegung von K^I. Ist vor Eintritt in den Iterationszyklus (4.2) bzw. (4.3) im Prozessor $\mathcal{P}_s$ die Zerlegung $K_s^I = L_s L_s^T$ berechnet worden, so können die lokalen Gleichungssysteme $K_s^I w_s^I = d_s^I$ in (4.2 (10)) sowie die Systeme $K_s^I \xi_s^I = d_s^I$ in Schritt (16) von (4.2) durch einfaches Vorwärts- und anschließendes Rückwärtseinsetzen gelöst werden. In diesem Fall kann speichertechnisch auch $a_s^I \equiv \underline{a}_s^I$ gesetzt werden. Zur Realisierung des direkten Solvers im Inneren der Teilgebiete ist es erforderlich, die Matrizen L_s im Prozessor $\mathcal{P}_s$ abzuspeichern. Dieser hohe Speicheraufwand begrenzt die maximal mögliche Anzahl von Freiheitsgraden je Prozessorknoten. Deshalb wäre zu prüfen, welche iterativen Löser ohne großen Effektivitätsverlust das Choleskyverfahren im Inneren ersetzen können.

4.2 Vorkonditionierung auf dem Rand der Teilgebiete

Im folgenden wird ein Vorkonditionierer V^C des Typs (3.11) zur Lösung des Gleichungssystems $V^C w^C = h^C$ von (3.6) betrachtet. Wegen (3.11) genügt es, die Gleichung $V_s^C w_s^C = g_s^C$ von (4.2 (10)) zu betrachten und V_s^C bzw. V_s^{-C}, vgl. (3.10), zu beschreiben. Dazu werden einige weitere Bezeichnungen benötigt:

Mit ndf sei die in $\overline{\Omega}_s$ konstante Zahl der Freiheitsgrade je Gitterpunkt bezeichnet. Das Subdomain Ω_s sei ein (möglicherweise auch krummlinig berandetes) Polygon mit dem Rand $\Gamma_s = \cup_{i=1}^{k_s} \Gamma_s^i$. Für $1 \le i \le k_s$ sei l_s^i die Anzahl der im Inneren des Randstücks Γ_s^i liegenden Randgitterpunkte und M_s^i die kleinste Zahl, die sich mit einem ganzzahligen ν_s^i in der Form $M_s^i = 2^{\nu_s^i} - 1$ darstellen läßt und nicht kleiner als l_s^i ist.

Weiterhin wird folgender Operator $P^{(N \to M)} : R^N \to R^M$ benötigt:
Werden die Komponenten $x(i)$ eines Vektors $x \in R^N$ als Werte einer Funktion f

über einem äquidistanten Gitter betrachtet, d.h. $x(i) = f(\frac{i}{(N+1)})$, $i = 1, \ldots, N$, wobei $f(0) = f(1) = 0$ gelten möge, und ist $\overline{f}$ die stückweise linear Interpolierte von f über diesem Gitter, so sei $y := P^{(N \to M)} x$ der Vektor $y \in R^M$, dessen Komponenten durch $y(j) := \overline{f}(\frac{j}{(M+1)})$, $j = 1, \ldots, M$ definiert seien.

Ohne Beschränkung der Allgemeinheit seien die Randgitterpunkte auf Γ_s lokal so numeriert, daß die Eckpunkte von Ω_s die Nummern 1 bis k_s erhalten. Die übrigen Randgitterpunkte sollen polygonkantenweise so numeriert sein, daß geometrisch benachbarte Punkte auch benachbarte Nummern erhalten. Dem folgend seien die zu Γ_s gehörenden Koppelfreiheitsgrade punktweise numeriert, so daß die den Eckpunkten von Ω_s zuzuordnenden Freiheitsgrade die Nummern 1 bis $q_s := k_s \star ndf$ und die zu Gitterpunkten auf Γ_s^i gehörenden Freiheitsgrade Nummern zwischen $\underline{q}_s^i$ und $\overline{q}_s^i$ tragen ($\underline{q}_s^1 = q_s + 1$, $\underline{q}_s^i = \overline{q}_s^{i-1} + 1$ für $2 \leq i \leq k_s$, $\overline{q}_s^{k_s} = n_s^C$, $\overline{q}_s^i - \underline{q}_s^i + 1 = l_s^i \star ndf$) .

Diese Konventionen benutzend werden für jedes Randstück Γ_s^i und für $1 \leq j \leq ndf$ die Matrizen $\mathcal{M}_{sij} \in R^{l_s^i, n_s^C}$, $\mathcal{S}_{sij} \in R^{M_s^i, M_s^I}$ und $P_{sij} \in R^{M_s^i, l_s^i}$ sowie die Diagonalmatrizen $E_{sij} \in R^{M_s^i, M_s^i}$ durch

$$
\begin{aligned}
\mathcal{M}_{sij}^{kl} &:= \begin{cases} 1, & \text{wenn } k = j + (l-1)ndf + \underline{q}_s^i - 1 \\ 0, & \text{sonst} \end{cases} \\
&\qquad\qquad (1 \leq k \leq l_s^i \, , \; 1 \leq l \leq n_s^C) \; , \\
\mathcal{S}_{sij}^{kl} &:= \sin\left(\frac{kl\pi}{M_s^i+1}\right) \qquad (1 \leq k \leq M_s^i \, , \; 1 \leq l \leq M_s^i) \; , \\
P_{sij} &:= P^{(l_s^i \to M_s^i)} \qquad \text{und} \\
E_{sij}^{kk} &:= \left[(M_s^i + 1) \, \sin\left(\frac{k\pi}{2M_s^i+2}\right) \sqrt{1 + \sin^2\left(\frac{k\pi}{2M_s^i+2}\right)} \; \right]^{-1} \\
&\qquad\qquad (1 \leq k \leq M_s^i)
\end{aligned}
\tag{4.4}
$$

definiert und es sei

$$
\begin{aligned}
\mathcal{M}_{si} &:= \begin{pmatrix} \mathcal{M}_{si1} \\ \vdots \\ \mathcal{M}_{si\,ndf} \end{pmatrix} \in R^{ndf \star l_s^i, n_s^C} \; , \\
\mathcal{S}_{si} &:= diag(\mathcal{S}_{sij}) \in R^{ndf \star M_s^i, ndf \star M_s^i} \; , \\
P_{si} &:= diag(P_{sij}) \in R^{ndf \star M_s^i, ndf \star l_s^i} \qquad \text{und} \\
E_{si} &:= diag(E_{sij}) \in R^{ndf \star M_s^i, ndf \star M_s^i}
\end{aligned}
\tag{4.5}
$$

sowie

$$
\begin{aligned}
\mathcal{M}_s &:= \begin{pmatrix} \mathcal{M}_{s0} \\ \mathcal{M}_{s1} \\ \vdots \\ \mathcal{M}_{sk_s} \end{pmatrix} \in R^{n_s^C, n_s^C} \; , \\
\mathcal{S}_s &:= diag(I_{q_s}, \mathcal{S}_{s1}, \ldots, \mathcal{S}_{sk_s}) \in R^{\overline{n}_s^C, \overline{n}_s^C} \; , \\
P_s &:= diag(I_{q_s}, P_{s1}, \ldots, P_{sk_s}) \in R^{\overline{n}_s^C, n_s^C} \qquad \text{und} \\
E_s &:= diag(I_{q_s}, E_{s1}, \ldots, E_{sk_s}) \in R^{\overline{n}_s^C, \overline{n}_s^C} \; ,
\end{aligned}
\tag{4.6}
$$

wobei I_{q_s} die q_s–dimensionale Einheitsmatrix bezeichnen soll, $\overline{n}_s^C := q_s + ndf \sum_{i=1}^{k_s} M_s^i$ und $\mathcal{M}_{s0}^{kl} := \begin{cases} 1, \text{ wenn } k = l \\ 0, \text{ sonst} \end{cases} \quad 1 \le k \le q_s \;,\; 1 \le l \le n_s^C$.

Aus der Hauptdiagonale der Submatrix K_s^C von (2.5) wird folgende Diagonalmatrix $D_s \in R^{n_s^C, n_s^C}$ gebildet:

$$
\begin{aligned}
\tilde{d}_s^j &:= (K_s^C)^{jj} \quad \text{für} \quad 1 \le j \le n_s^C \\
\overline{d}_s &:= \tilde{d}_s + \textstyle\sum_{t \neq s} A_s^C (A_t^C)^T \tilde{d}_t \\
\hat{d}_s^j &:= \begin{cases} 0 & \text{, wenn } \overline{d}_s^j = 0 \\ \dfrac{1}{\sqrt{\overline{d}_s^j}} & \text{, wenn } \overline{d}_s^j \neq 0 \end{cases} \quad \text{für} \quad 1 \le j \le n_s^C \\
d_s^j &:= \begin{cases} \hat{d}_s^j & \text{, wenn } \quad 1 \le j \le k_s \\ \dfrac{\hat{d}_s^j l_s^i}{M_s^i} & \text{, wenn } \quad \underline{q}_s^i \le j \le \overline{q}_s^i \end{cases} \\
D_s &:= \operatorname{diag}(d_s^j) \; .
\end{aligned}
\tag{4.7}
$$

Die Hauptdiagonale $\tilde{d}_s$ wird zu $\overline{d}_s$ global assembliert, dann wird komponentenweise $\hat{d}_s := (\overline{d}_s)^{-1/2}$ berechnet und schließlich wird der den Inneren Randstücken Γ_s^i entsprechende Teil mit dem Verhältnis von l_s^i zu M_s^I skaliert [6].

Mit den bei (4.6) und (4.7) angegebenen Matrizen kann der verwendete Vorkonditionierer V_s^C durch

$$
V_s^{-C} := D_s \mathcal{M}_s^T P_s^T \mathcal{S}_s E_s \mathcal{S}_s P_s \mathcal{M}_s D_s \tag{4.8}
$$

beschrieben werden, d.h., zur Lösung von $V_s^C w_s^C = g_s^C$ sind die Operationen

$$
w_s^C := D_s \mathcal{M}_s^T P_s^T \mathcal{S}_s E_s \mathcal{S}_s P_s \mathcal{M}_s D_s g_s^C \tag{4.9}
$$

auszuführen.

Aufgrund der Blockstruktur der Matrizen E_s, P_s und $\mathcal{S}_s$ kann (4.9) mit $\mathcal{Z}_s := diag(I_{q_s}, \mathcal{Z}_{s1} \ldots \mathcal{Z}_{sk_s})$, $\mathcal{Z}_{si} := diag(\mathcal{Z}_{sij})_{j=1}^{ndf}$ und $\mathcal{Z}_{sij} := P_{sij}^T \mathcal{S}_{sij} E_{sij} \mathcal{S}_{sij} P_{sij}$ auch in der Form

$$
\begin{array}{lrcl}
(1) & \check{g}_s^C & := & D_s g_s^C \\
(2) & \hat{g}_s^C & := & \mathcal{M}_s \check{g}_s^C \\
(3) & \hat{w}_s^C & := & \mathcal{Z}_s \hat{g}_s^C \\
(4) & \check{w}_s^C & := & \mathcal{M}_s^T \hat{w}_s^C \\
(5) & w_s^C & := & D_s \check{w}_s^C
\end{array}
\tag{4.10}
$$

aufgeschrieben werden.

In den Schritten (1) und (5) wird der jeweilige Vektor skaliert.
Die Vektoren $\check{g}_s^C$ und $\hat{g}_s^C$ bzw. $\check{w}_s^C$ und $\hat{w}_s^C$ unterscheiden sich lediglich hinsichtlich der Numerierung der innerhalb der Randsegmente Γ_s^i lokalisierten Freiheitsgrade. Diese Umnumerierung wird durch die Schritte (2) und (4) beschrieben.

---

[6] Die Größen D_s und E_{si} sind für den gesamten Iterationszyklus konstant und daher vorab zu berechnen und abzuspeichern.

Die Komponenten der mit einem ˇ gekennzeichneten Vektoren sind entsprechend der eingangs dieses Abschnitts postulierten Numerierung angeordnet und die mit einem ˆ gekennzeichneten Vektoren haben die Struktur

$$\hat{g}_s^C = (\hat{g}_{s0}^C, \ldots, \hat{g}_{sk_s}^C)^T \quad , \quad \hat{w}_s^C = (\hat{w}_{s0}^C, \ldots, \hat{w}_{sk_s}^C)^T \ , \tag{4.11}$$

wobei $\hat{g}_{s0}^C$ und $\hat{w}_{s0}^C$ die zu den Eckpunkten von Ω_s gehörenden Freiheitsgrade symbolisieren. Die Teilvektoren $\hat{g}_{si}^C$ und $\hat{w}_{si}^C$ $(1 \leq i \leq k_s)$ enthalten die zum Inneren von Γ_s^i gehörenden Freiheitsgrade, sortiert nach gleichnamigen Freiheitsgraden: $\hat{g}_{si}^C = (\hat{g}_{si1}^C, \ldots, \hat{g}_{si\,ndf}^C)^T$ und $\hat{w}_{si}^C = (\hat{w}_{si1}^C, \ldots, \hat{w}_{si\,ndf}^C)^T$. Der Schritt (3) in (4.10) kann daher als $\hat{w}_{s0}^C := \hat{g}_{s0}^C$ und $\hat{w}_{si}^C := \mathcal{Z}_{si}\hat{g}_{si}^C$ für $1 \leq i \leq k_s$ bzw. auch in der Form

$$\begin{array}{ll}
(3a) & \hat{w}_{s0}^C \ := \ \hat{g}_{s0}^C \\
(3b) & \left\{ \begin{array}{lcl} \hat{h}_{sij}^C & := & P_{sij}\hat{g}_{sij}^C \\ \hat{d}_{sij}^C & := & \mathcal{S}_{sij}E_{sij}\mathcal{S}_{sij}\hat{h}_{sij}^C \\ \hat{w}_{sij}^C & := & P_{sij}^T\hat{d}_{sij}^C \end{array} \right\} \ , \ 1 \leq j \leq ndf \ \ , \ 1 \leq i \leq k_s
\end{array} \tag{4.12}$$

geschrieben werden. In der ersten Zeile von (3b) werden die l_s^i Werte des j-ten Freiheitsgrades auf der Kante Γ_i zu M_s^i Werten extrapoliert, während in der dritten Zeile von (3b) die M_s^i Werte aus $\hat{d}_{sij}^C$ wieder zu l_s^i Werten restringiert werden. Hinter der zweiten Zeile von (3b) verbirgt sich die schnelle Fouriertransformation (FFT), vgl.[5].

5 Einige numerische Resultate

Alle in diesem Abschnitt genannten Iterationszahlen, Rechenzeiten u.a. beziehen sich auf Rechnungen zu dem in Abbildung 2 dargestellten technischen Problem:

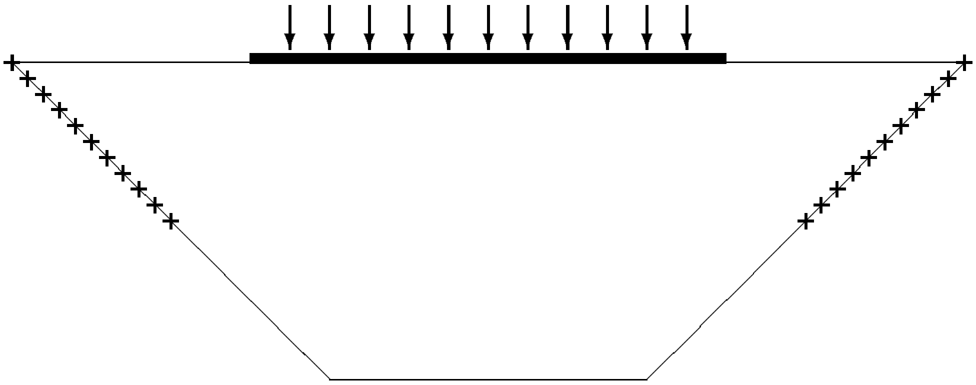

Abbildung 2: Belasteter Keil

Ein keilförmiger, beidseitig längs der Hälfte seiner schrägen Kanten fest eingespannter und ansonsten frei bewewglicher Körper wird mit einer flächenverteilten Last nach unten gedrückt.

In Abbildung 3 ist dargestellt, wie das Gebiet auf 2, 4, 8, 16, 32 bzw. 64 Prozessoren aufgeteilt wurde.

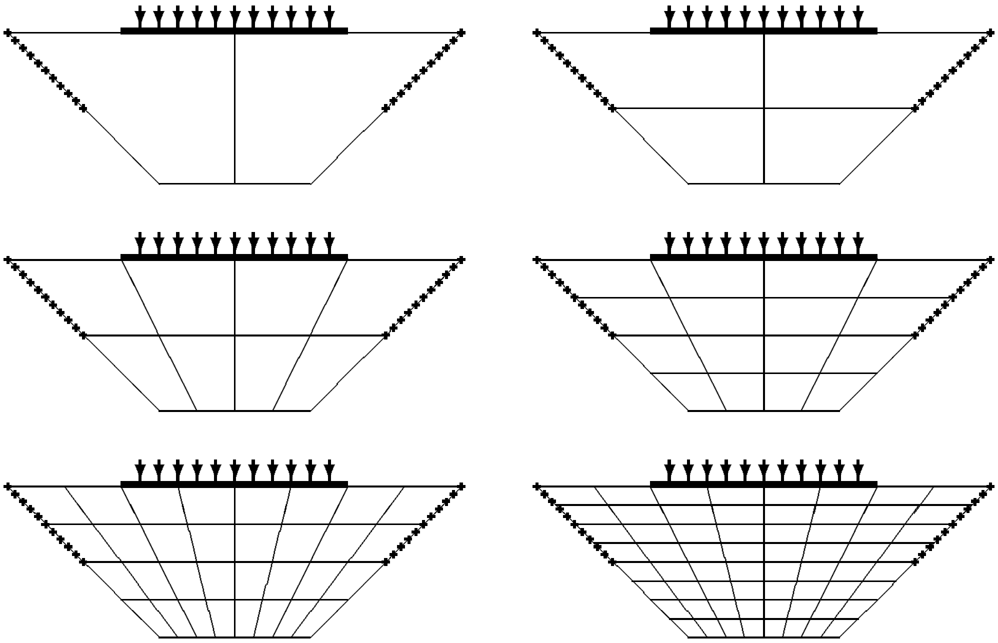

Abbildung 3: Teilgebiete Ω_i bei 2 bis 64 Prozessoren

Die Rechnungen wurden auf einem Vierecksnetz ausgeführt. Dieses wurde durch Zerlegung der Subdomains Ω_s in jeweils N_s^x mal N_s^y Vierecke erzeugt.

Die CG–Iteration wurde beendet, wenn die Zahl $\hat{\gamma}$ von Schritt (11) in (4.2) die Ungleichung $\hat{\gamma} \leq 10^{-14}\hat{\gamma}^o$ erfüllt, wobei $\hat{\gamma}^o$ der beim erstmaligen Abarbeiten dieser Zeile ermittelte Wert sein soll.

Die angegebenen Rechenzeiten für die Lösung der Gleichungssysteme wurden auf Transputersystemen der Firma PARSYTEC (Prozessortyp T800 bei bis zu 8 und T805 bei 16 bis 128 Prozessoren) ermittelt.

5.1 Abhängigkeit der Iterationszahlen von der Geometrie der Vernetzung

Zunächst soll untersucht werden, wie die zum Erreichen der geforderten Genauigkeit notwendige Zahl der CG–Iterationsschritte von den geometrischen Verhältnissen im Berechnungsgitter abhängt. Dazu wurden für $N_s^x \in \{2, 5, 10, 15, 20, 25\}$ jeweils eine Schar von Vernetzungen mit $2 \leq N_s^y \leq 80$ durchgerechnet und dabei die Dimension des zu lösenden Gleichungssystems und die Zahl der ausgeführten Iterationsschritte ermittelt. In Tabelle 1 sind einige dieser auf 16 Prozessoren ermittelten Daten zusammengestellt. Mit $\boxed{}$ sind die maximale und die minimale Iterationszahl jeder Serie markiert:

Tabelle 1: Iterationszahlen bei 16 Prozessoren

N_s^x	N_s^y	n	# Iter.	N_s^x	N_s^y	n	# Iter.	N_s^x	N_s^y	n	# Iter.
2	2	162	$\boxed{44}$		2	378	$\boxed{55}$		2	738	74
	3	234	51		3	546	58		3	1066	72
	4	306	57		4	714	58		4	1394	$\boxed{68}$
	5	378	62	5	5	882	63		5	1722	69
	10	738	89		10	1722	81	10	10	3362	75
	20	1458	136		20	3402	106		20	6642	96
	30	2178	167		30	5082	117		30	9922	104
	40	2898	198		40	6762	148		40	13202	123
	50	3618	228		50	8442	155		50	16482	131
	60	4338	256		60	10122	165		60	19762	138
	70	5058	263		70	11802	188		70	23042	149
	80	5778	$\boxed{315}$		80	13482	$\boxed{211}$		80	26322	$\boxed{170}$
	2	1098	85		2	1458	$\boxed{100}$		2	1818	$\boxed{111}$
	3	1586	83		3	2106	97		3	2626	106
	4	2074	73		4	2754	87		4	3434	93
	5	2562	73		5	3402	87		5	4242	93
	6	3050	72		6	4050	86		6	5050	90
	7	3538	73		7	4698	88		7	5858	93
	8	4026	$\boxed{70}$		8	5346	80		8	6666	84
	9	4514	$\boxed{70}$		9	5995	$\boxed{79}$		9	7474	83
	10	5002	78		10	6642	80		10	8282	83
15	15	7442	83		15	9882	80		15	12322	$\boxed{80}$
	20	9882	100		18	11826	83		20	16362	85
	30	14762	107	20	20	13122	87		21	17170	88
	35	17202	124		22	14418	91		23	18786	88
	40	19642	129		25	16362	90		24	19594	89
	45	22082	$\boxed{135}$		30	19602	94	25	25	20402	88

Von der Theorie her ist bekannt, daß das Verhältnis von maximaler zu minimaler

Kantenlänge der Elemente multiplikativ sowohl in die Abschätzungen für den Diskretisierungsfehler als auch in die Konditionszahl der Matrix K eingeht.

Bei der Interpretation der Daten aus Tabelle 1 ist zu beachten, daß wegen der Gestalt der Ω_s (vgl. auch Abbildung 3) dieses Verhältnis nicht für $N_s^x = N_s^y$ sondern für $N_s^x = \nu N_s^y$ mit einem $\nu > 1$ minimal wird. Es kann festgestellt werden, daß in allen 6 aufgelisteten Serien die maximale Iterationszahl stets dort beobachtet wird, wo $|N_s^x - N_s^y|$ maximal wird, während die kleinste Iterationszahl bei $N_s^x \approx 2N_s^y$ erreicht wird.

Die Abhängigkeit der Iterationszahlen von der Netzgeometrie wird noch von einem zweiten Effekt überlagert: Weichen die Zahlen N_s^x und N_s^y stark voneinander ab, so ist das Verhältnis der Zahl der Koppelfreiheitsgrade zur Zahl der inneren Freiheitsgrade (n^C/n^I) wesentlich größer als bei $N_s^x \approx N_s^y$. Stehen sehr vielen Koppelfreiheitsgraden nur relativ wenige innere Freiheitsgrade gegenüber, so ist zwar die Kopplung zwischen den Prozessoren sehr intensiv (und der Kommunikationsaufwand entsprechend groß), aber der im Inneren wirkende direkte Löser (Cholesky) arbeitet nur auf einem sehr kleinen Teil des Gesamtgleichungssystems. Diese Disbalance wirkt sich negativ auf die erforderlichen Iterationszahlen aus.

Die in Tabelle 1 angegebenen Iterationszahlen unterstreichen, wie wichtig die sorgfältige Wahl der Subdomains und des Netzes für die Effizienz des Lösers ist.

In Abbildung 4 sind die Daten aus Tabelle 1 grafisch dargestellt:

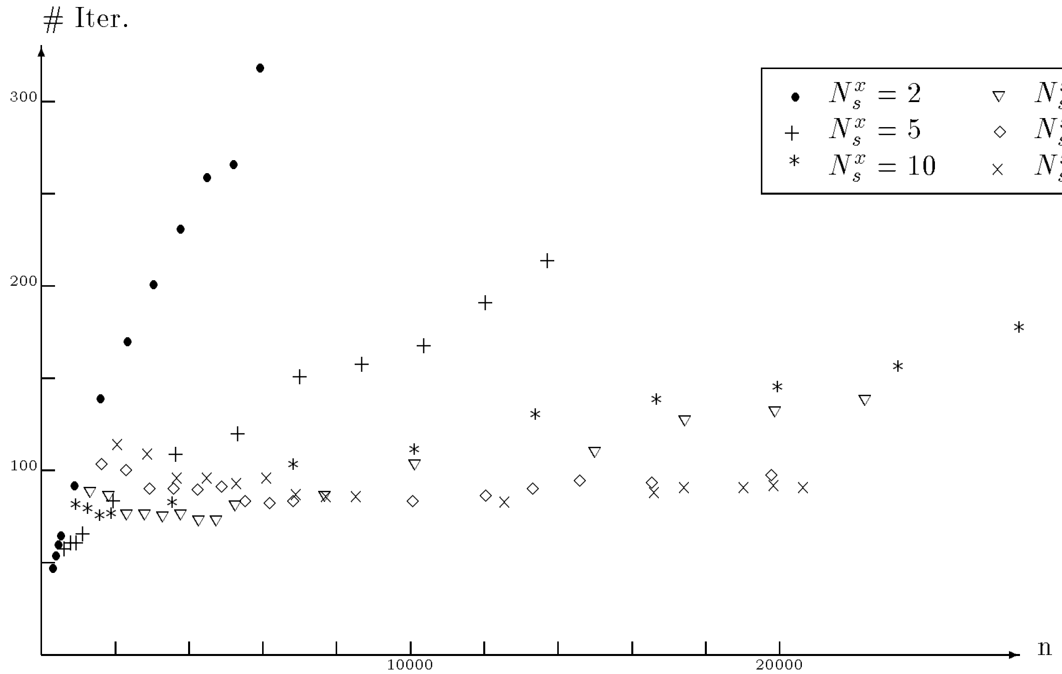

Abbildung 4: Iterationszahlen bei verschiedenen Netzgeometrien

Um den Einfluß der Netzgeometrie auf die Iterationszahlen in den folgenden Betrachtungen auszuschließen, werden in den Abschnitten 5.2, 5.3 und 5.4 nur noch Berechnungsgitter mit $N_s^x = N_s^y =: N_s$ verwendet.

5.2 Abhängigkeit der Iterationszahlen von der Zahl der Prozessoren

In diesem Abschnitt soll die Abhängigkeit der Iterationszahlen von der Zahl der zur Lösung benutzten Prozessoren bei konstanter Dimension n des Gleichungssystems untersucht werden. Dazu wurde das Problem aus Abbildung 2 für verschiedene $n \in [\,542\,, 74110\,]$ jeweils auf 1,2,4,8,16,32,64 oder 128 Prozessoren gerechnet, wobei die Iterationszahlen und die zur Lösung benötigte Gesamtzeit ermittelt wurden. Einige dieser Daten sind in Tabelle 2 zusammengestellt, wobei mit $\boxed{}$ das Minimum der jeweiligen Zeile hervorgehoben wurde:

Tabelle 2: Zusammenhang von Iterationszahlen und Prozessoranzahl

n	1	2	4	8	16	32	64	128	# Proc.
542	$\boxed{32}$	38	35	44	55	68	88		# Iter.
	4.9	3.6	2.0	$\boxed{1.6}$	1.9	2.8	6.2		Sekunden
1198	55	60	$\boxed{49}$	57	64	82	96		# Iter.
	46.5	15.0	6.6	3.9	$\boxed{3.8}$	5.2	9.9		Sekunden
2110		54	$\boxed{41}$	53	66	83	105	133	# Iter.
		28.3	11.5	5.7	$\boxed{5.6}$	6.9	14.1	44.7	Sekunden
3278			$\boxed{55}$	65	75	94	114		# Iter.
			28.7	11.9	$\boxed{10.3}$	10.8	19.8		Sekunden
4702			$\boxed{56}$	68	78	97	117	160	# Iter.
			48.3	18.2	14.8	$\boxed{13.7}$	24.3	79.9	Sekunden
6382				$\boxed{68}$	80	99	117		# Iter.
				26.2	20.7	$\boxed{17.0}$	28.5		Sekunden
8318				$\boxed{62}$	77	96	120	160	# Iter.
				33.8	27.1	$\boxed{19.8}$	34.0	106.2	Sekunden
10510				$\boxed{72}$	81	103	128		# Iter.
				56.3	40.6	$\boxed{28.1}$	43.4		Sekunden
12958					$\boxed{87}$	110	136	185	# Iter.
					55.8	$\boxed{35.1}$	52.2	154.7	Sekunden
18622					$\boxed{90}$	113	139	189	# Iter.
					91.9	$\boxed{49.2}$	67.1	190.3	Sekunden
25310						$\boxed{110}$	137	186	# Iter.
						$\boxed{64.7}$	82.2	220.0	Sekunden
33022						$\boxed{108}$	137	185	# Iter.
						$\boxed{84.6}$	100.8	251.2	Sekunden
41758						$\boxed{118}$	148	202	# Iter.
						$\boxed{127.3}$	137.3	316.8	Sekunden
74110							$\boxed{162}$	219	# Iter.
							$\boxed{253.9}$	472.7	Sekunden

Eine Auswahl der Iterationszahlen und Rechenzeiten aus Tabelle 2 ist in Abbildung 5 grafisch dargestellt. Dabei zeigt sich, daß die Iterationszahlen drastisch anwachsen, wenn die Zahl der Freiheitsgrade je Prozessor zu klein und damit das Verhältnis von n^C zu n^I zu groß wird. Dies gilt auch für die Rechenzeiten, weil bei immer weniger Freiheitsgraden je Prozessor der Kommunikationsaufwand gegenüber dem Rechenaufwand immer dominanter wird.

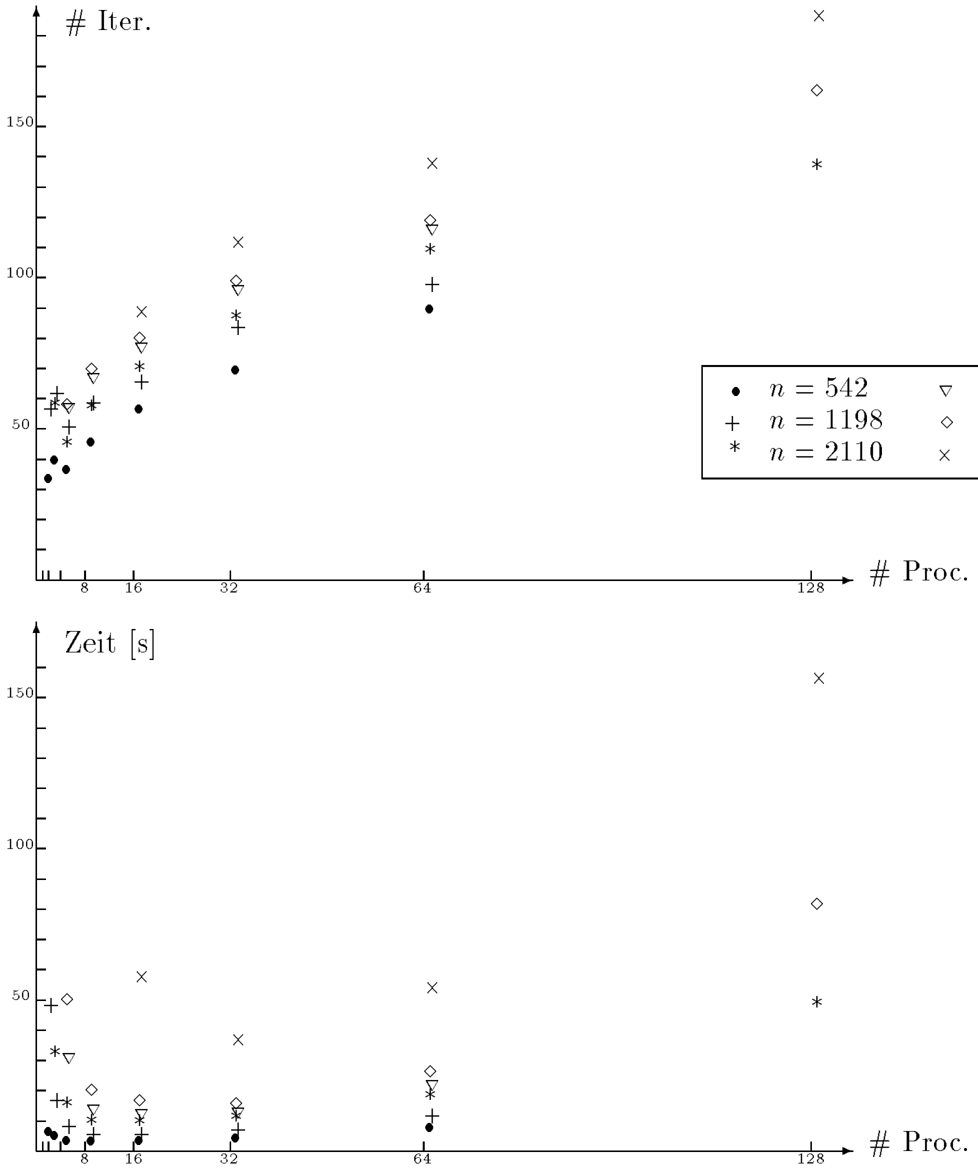

Abbildung 5: Iterationszahlen und Rechenzeiten bei verschiedenen Prozessorzahlen

Aus Tabelle 2 ist aber auch ersichtlich, daß die Prozessoranzahl, bei der die minimale Itarationszahl beobachtet wird, im allgemeinen kleiner ist als diejenige, bei der die kleinste Gesamtrechenzeit erreicht wurde. Dies resultiert aus der Verteilung des insgesamt zu bewältigenden Arithmetikaufwandes auf eine größere Prozessorzahl, was dem Anwachsen der Rechenzeiten wegen des anwachsenden Kommunikationsaufwandes entgegenwirkt.

Den gemessenen Zeiten für die Lösung der Gleichungssysteme liegt noch eine Kommunikationstechnologie zugrunde, bei der zur globalen Assemblierung eines Vektors r in jedem Prozessor $\mathcal{P}_s$ die lokalen Anteile r_t aller Prozessoren $\mathcal{P}_t$ angesammelt werden. Da aber nur von höchstens 8 geometrisch benachbarten Prozessoren diese Anteile wirklich benötigt werden, sind im betrachteten Beispiel bei 8 Prozessoren mehr als 28%, bei 16 Prozessoren mehr als 47%, bei 32 Prozessoren mehr als 74%, bei 64 Prozessoren mehr als 87% und bei 128 Prozessoren mehr als 93% der in jedem Prozessor zusammengefaßten Daten anderer Prozessoren und damit auch der entprechende Kommunikationsaufwand im Grunde überflüssig. Hinzu kommt, daß der als virtuelle Topologie benutzte Hypercube in dem zweidimensionalen Prozessorgitter physisch gar nicht existiert und daher die Prozessoren noch zusätzlich mit dem Durchrouten fremder Daten belastet sind, was sich umso mehr auswirkt, je mehr Prozessoren kommunizieren und je größer die auszutauschenden (und durchzuroutenden) Datenpakete sind. Durch Verwendung eines in diesem Sinne wesentlich verbesserten Prinzips zur globalen Assemblierung lassen sich hier beträchtliche Einsparungen erreichen[7].

5.3 Abhängigkeit der Iterationszahlen von der Zahl der Unbekannten

In diesem Abschnitt soll untersucht werden, wie stark die Iterationszahlen anwachsen, wenn die Dimension des Gleichungssystems zunimmt. Dazu wurden auf 1,2,4,8,16,32,64 und 128 Prozessoren jeweils eine Schar von Aufgaben mit gleicher Netzgeometrie gerechnet. Die ermittelten Iterationszahlen und Rechenzeiten sind in Tabelle 3 zusammengestellt:

Tabelle 3: Zusammenhang von Iterationszahlen und Systemdimension

# Proc.	n	# Iter.	T	# Proc.	n	# Iter.	T
	142	25	1.2		142	28	0.8
	310	38	4.9		310	43	2.4
	542	32	8.9		542	38	3.6
1	838	53	26.9	2	838	56	9.3
	1198	55	46.5		1198	60	14.9
					2110	54	28.3
					2590	76	56.7
	142	50	0.6		142	36	0.7
	310	39	1.5		310	45	1.3
	542	35	2.0		542	44	1.6
	838	47	4.4		838	54	3.0
	1198	49	6.6		1198	57	3.9
	2110	41	11.5		2110	53	5.7
4	2590	53	21.1	8	2590	62	9.4
	3278	55	28.7		3278	65	11.9
	3870	58	38.8		3870	68	15.1
	4702	56	48.3		4702	68	18.2
					5510	66	21.3
					6382	68	26.2
					7318	72	32.8
					8318	62	33.8
					10510	72	56.3

[7]Ein solches Prinzip ist bereits entwickelt worden und wird demnächst in FEAP integriert. Vgl. auch die Ausführungen im Anschluß an (3.5)

# Proc.	n	# Iter.	T	# Proc.	n	# Iter.	T
	142	44	0.8		542	68	2.8
	310	53	1.4		1198	82	5.2
	542	55	1.9		2110	83	6.9
	838	63	3.1		3278	94	10.8
	1198	64	3.8		4702	97	13.7
	2110	66	5.6		6382	99	17.0
	2590	71	8.3		8318	96	19.8
	3278	75	10.3		10510	103	28.1
	3870	78	12.6		12958	110	35.1
	4702	78	14.8		15662	115	43.0
16	5510	77	17.1	32	18662	113	49.2
	6382	80	20.7		25310	110	64.7
	7318	83	25.0		33022	108	84.6
	8318	77	27.1		41758	118	127.3
	10510	81	40.6				
	12958	87	55.8				
	15662	93	75.4				
	17110	90	82.1				
	18622	90	91.9				
	20198	88	100.6				
	542	88	6.2		2110	133	44.7
	1198	96	9.9		4702	160	79.9
	2110	105	14.1		8318	160	106.2
	3278	114	19.8		12958	185	154.7
	4702	117	24.3		18622	189	190.3
	6382	117	28.5		25310	186	220.0
	8318	120	34.0		33022	185	251.2
	10510	128	43.4		41758	202	316.8
64	12958	136	52.2	128	51518	214	375.0
	15662	143	61.6		62302	223	435.1
	18622	139	67.1		74110	219	472.7
	25310	137	82.0		86942	211	499.9
	33022	137	100.8		100798	209	541.9
	41758	148	137.3		115678	206	581.2
	46776	150	152.5		131582	205	628.6
	51518	158	175.4		148510	221	746.6
	62302	166	219.3				
	74110	162	253.9				

Den Datenlisten aller Prozessorgruppierungen ist gemeinsam, daß sich die Iterationszahlen von der kleinsten Gleichungssystemdimension ($\underline{n}$) zur größten ($\overline{n}$) in etwa verdoppeln, obwohl $\overline{n}/\underline{n} \approx 8$ bei 1 Prozessor und $\overline{n}/\underline{n} \approx 142$ bei 16 Prozessoren erreicht wurde. Dies deutet auf die auch aus den einzelnen Zahlenreihen ablesbare Tendenz hin, daß die Iterationszahlen nur noch sehr langsam anwachsen, wenn eine gewisse Mindestdimension des Gleichungssystems, die offenbar auch von der Prozessoranzahl (und damit von der lokalen Anzahl der Freiheitsgrade) abhängt, erreicht ist.

Das Verhältnis der Wachstumsgeschwindigkeit der Rechenzeiten zu der der Zahl der Freiheitsgrade ist monoton fallend mit der Zahl der Prozessoren.

Zur besseren Illustration sind die Iterationszahlen in Abbildung 6 grafisch dargestellt. Dabei ist zu beachten daß die Skalierungen beider Achsen in jedem der Diagramme verändert wurde:

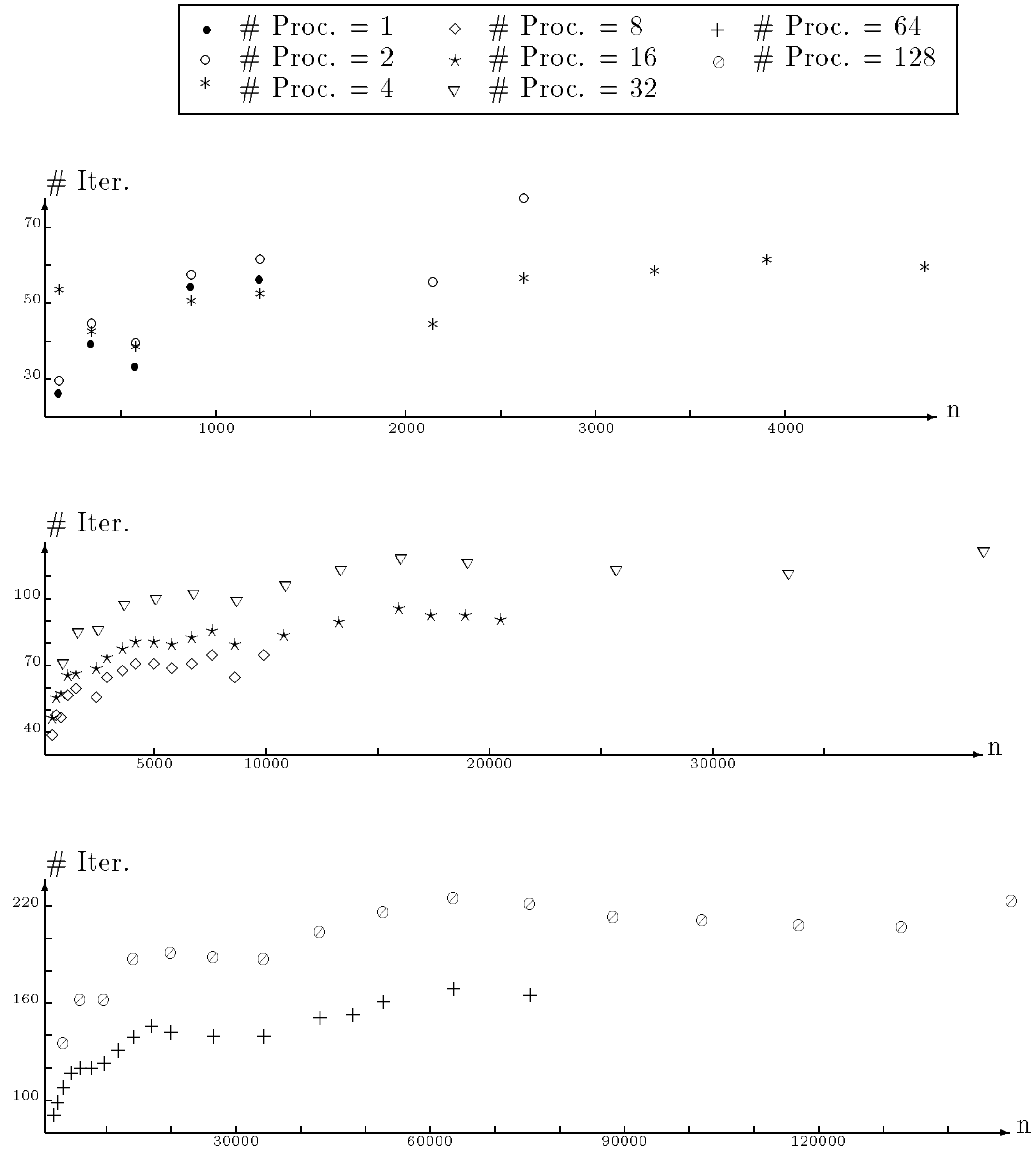

Abbildung 6: Iterationszahlen bei wachsender Systemdimension

In Abbildung 7 ist das Verhalten der Rechenzeit in Abhängigkeit von der Zahl der Freiheitsgrade dargestellt. Auch hier ändern sich die Skalierungen der Achsen von Diagramm zu Diagramm:

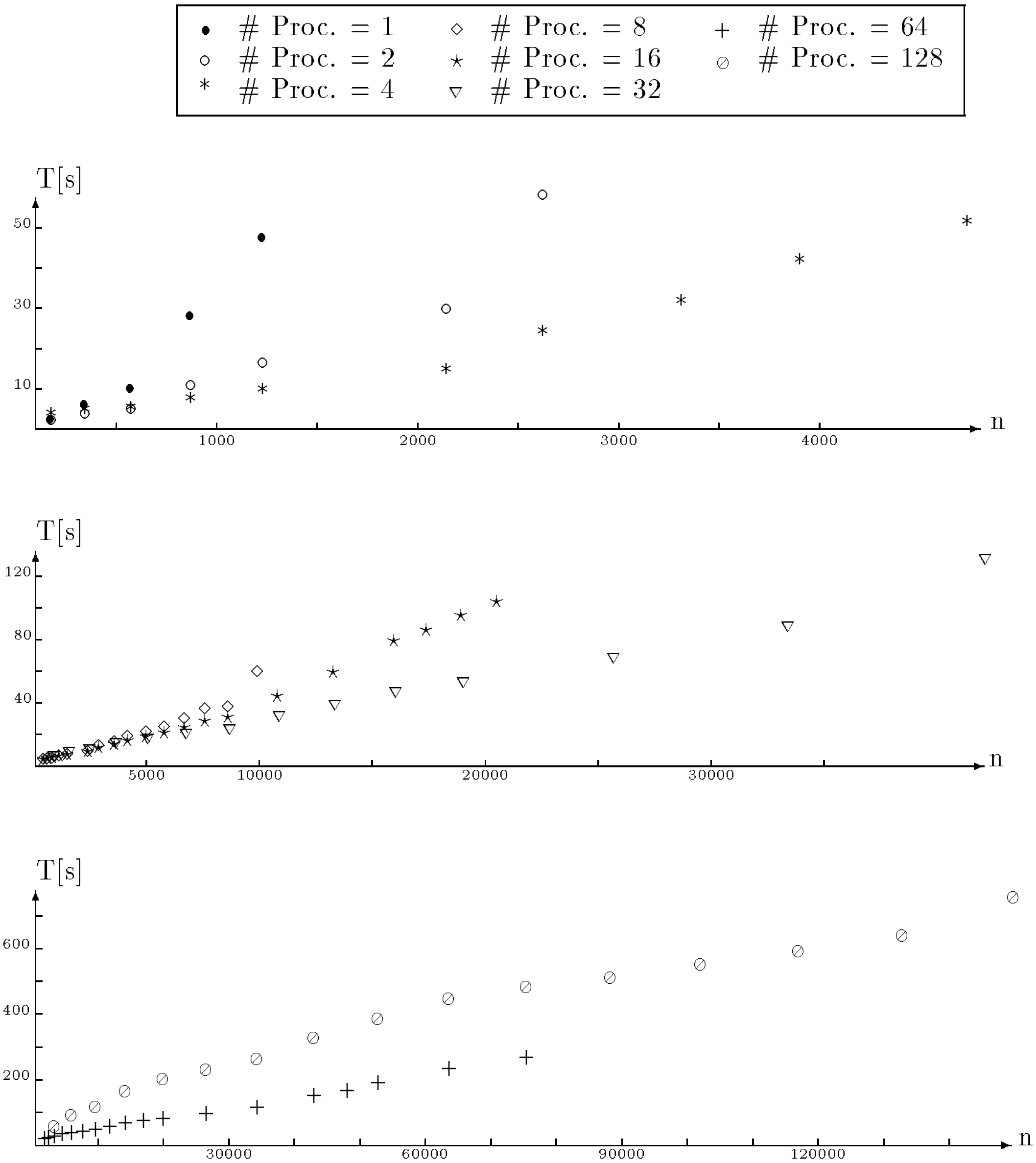

Abbildung 7: Rechenzeiten bei wachsender Systemdimension

5.4 Speed Up

Zur Bestimmung von Speed–Up–Werten wurde die Aufgabe aus Abbildung 2 mit $n \in \{142,\ 542,\ 2110,\ 4702,\ 8318,\ 18622,\ 33022\}$ jeweils auf allen Prozessorgruppierungen gerechnet, auf denen die entsprechende Vernetzung realisierbar war. Dabei wurden sowohl die Gesamtrechenzeiten (T_1) als auch die reinen Arithmetikzeiten (T_2) für den Lösungsprozeß gemessen. Die Differenz $T_1 - T_2$ entspricht dem Kommuniukationsaufwand. Aus den Zeiten T_1 und T_2 wurden die Speed–Up–Werte S_1 und S_2 berechnet, indem das k–Fache der auf k Prozessoren ermittelten Zeiten durch die entsprechenden auf mehr als k Prozessoren ermittelten Zeiten dividiert wurden, wobei k die kleinste Prozessoranzahl ist, auf der die Aufgabe mit der entsprechenden Anzahl von Freiheitsgraden gelöst werden konnte.

Werden diese Speed–Up–Werte noch durch die Zahl der benutzten Prozessoren dividiert, ergeben sich die Effizienzen E_1 und E_2. Die so ermittelten Zahlen sind in Tabelle 4 zusammengestellt:

Tabelle 4: Speed Up

n	# Proc.	# Iter.	T_1	T_2	S_1	S_2	E_1	E_2
	1	25	1.25	1.25	-	-	-	-
	2	28	0.84	0.84	1.49	1.49	0.74	0.74
142	4	50	0.64	0.57	1.95	2.19	0.49	0.55
	8	36	0.68	0.51	1.84	2.45	0.23	0.31
	16	44	0.84	0.44	1.49	2.48	0.09	0.16
	1	32	8.86	8.86	-	-	-	-
	2	38	3.56	3.49	2.49	2.54	1.24	1.27
	4	35	2.04	1.91	4.34	4.64	1.09	1.16
542	8	44	1.59	1.26	5.57	7.03	0.70	0.88
	16	55	1.87	1.05	4.74	8.43	0.30	0.53
	32	68	2.81	0.92	3.15	9.63	0.10	0.30
	64	88	6.16	1.02	1.44	8.69	0.02	0.14
	2	54	28.31	28.15	-	-	-	-
	4	41	11.51	11.25	4.92	5.00	1.23	1.25
	8	53	5.68	4.99	9.97	11.28	1.25	1.41
2110	16	66	5.62	3.87	10.07	14.54	0.63	0.91
	32	83	6.89	2.71	8.22	20.77	0.26	0.65
	64	105	14.13	2.75	4.01	20.47	0.06	0.32
	128	133	44.67	2.92	1.26	19.28	0.01	0.15
	4	56	48.26	47.73	-	-	-	-
	8	68	18.21	16.93	10.60	11.28	1.33	1.41
4702	16	78	14.79	11.81	13.05	16.17	0.82	1.01
	32	97	13.73	6.64	14.06	28.75	0.44	0.90
	64	117	24.28	5.82	7.95	32.80	0.12	0.51
	128	160	79.94	5.95	2.41	32.09	0.02	0.25
	8	62	33.79	32.25	-	-	-	-
	16	77	27.07	23.22	9.99	11.11	0.62	0.69
8318	32	96	19.84	10.64	13.63	24.25	0.43	0.76
	64	120	33.96	9.03	7.96	28.57	0.12	0.45
	128	160	106.19	7.94	2.54	32.49	0.02	0.25
	16	90	91.90	85.33	-	-	-	-
18622	32	113	49.24	33.29	29.86	41.01	0.93	1.28
	64	139	67.08	24.85	21.92	54.94	0.34	0.86
	128	189	190.32	18.22	7.72	74.93	0.06	0.59
	32	108	84.59	64.41	-	-	-	-
33022	64	137	100.80	45.49	26.85	45.31	0.42	0.71
	128	185	251.16	27.77	10.78	74.22	0.08	0.58

Die Daten zeigen, daß die Effizienz E_1 umso größer ist, je mehr Freiheitsgrade auf jedem Prozessor liegen. Die sehr schlechten Werte für E_1 bei mehr als 32 Prozessoren sind durch den (unnötig großen, vgl. die Ausführungen am Ende von Abschnitt 5.2) Kommunikationsaufwand bedingt. Die angegebenen Werte von E_2, bei denen nur der reine Arithmetikaufwand zu grunde gelegt ist, zeigen die durch Effektivierung der Kommunikation erschließbaren Reserven. Ist die Zahl der Freiheitsgrade je Prozessor hinreichend groß, werden sogar Effizienzen $E_2 > 1$ und $E_1 > 1$, letzteres nur auf Prozessorzahlen, bei denen die Kommunikationsverluste nur wenig ins Gewicht fallen, beobachtet. In Abbildung 8 sind die Effizienzen grafisch dargestellt:

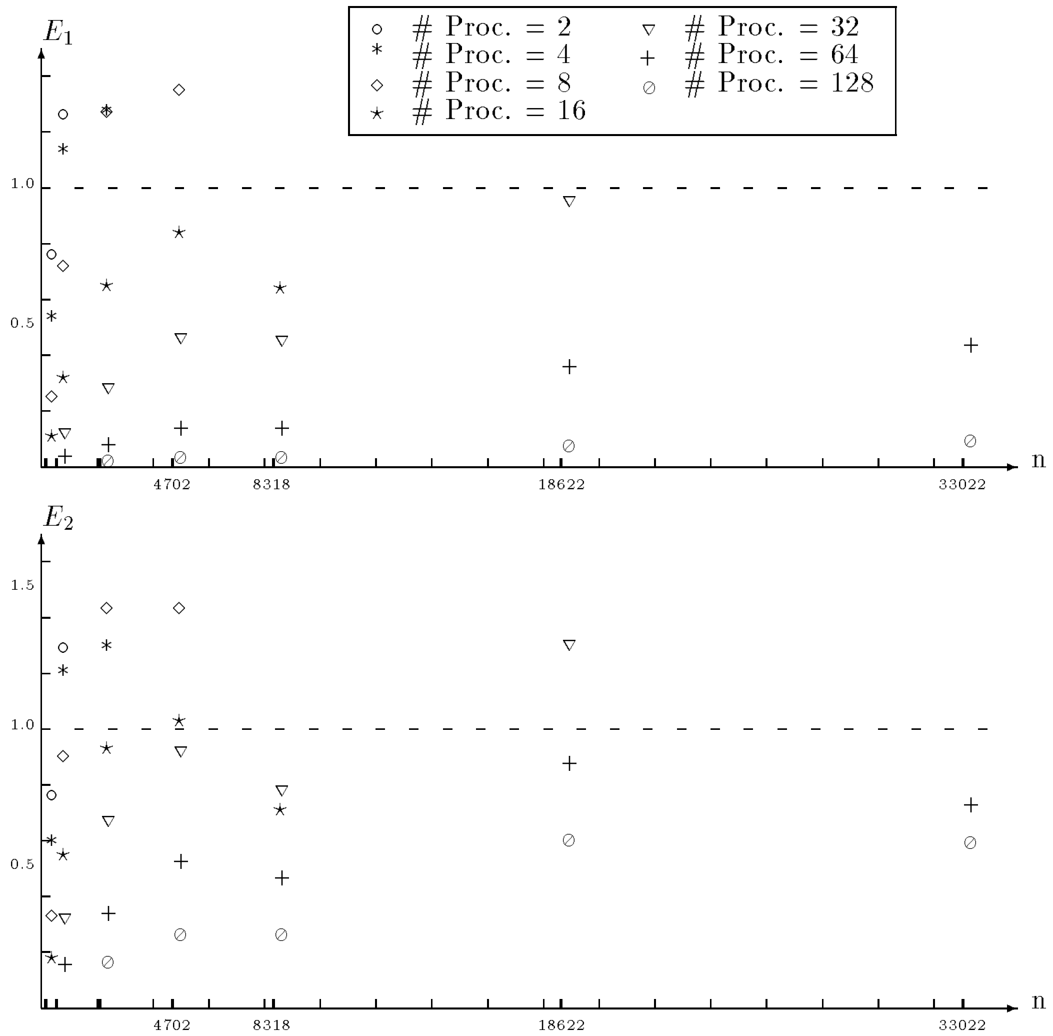

Abbildung 8: Effizienzen E_1 und E_2

6 Appendix

Es ist vorgesehen, die in den Abschnitten 5.2, 5.3 und 5.4 angegebenen Rechenzeiten nochmals zu ermitten, wenn die effektivierte Kommunikationstechnologie in das Programm FEAP integriert worden ist. Diese Daten sollen in einem Folgeartikel zusammengestellt werden.

Literatur

[1] G.Haase,T.Hommel,A.Meyer und M.Pester,
Bibliotheken zur Entwicklung paralleler Algorithmen,
Preprint Nr. SPC 94_4, Fakultät f. Mathematik, TU Chemnitz-Zwickau

[2] A.Meyer,
A Parallel Preconditioned Conjugate Gradient Method Using Domain Decomposition and Inexact Solvers on Each Subdomain,
Computing 45, 1990.

[3] R.L.Taylor,
FEAP - A finite element analysis program, Description and Users-Manual,
University of California, Berkeley,1990

[4] S.Meynen und P.Wriggers,
Tätigkeitsbericht Darmstadt zum Forschungsvorhaben Wr19/5-1
Parallele Iterationsverfahren,
Technische Hochschule Darmstadt, Institut für Mechanik, Juli 1994

[5] M.Dryja,
A finite element–capacitance method for elliptic problems on regions patitioned into subregions,
Numer. Math. 44, 153–168 (1984)

[6] A.Chronopoulos,
A class of parallel iterative methods implemented on multiprocessors
University of Illinois at Urbana, November 1986