Adv. Radio Sci., 7, 101–105, 2009
www.adv-radio-sci.net/7/101/2009/

Advances in Radio Science

Effiziente Verfahren zur Parallelisierung von EMV-Simulationen für den Entwurfsprozess integrierter Schaltungen

M. Barke, T. Preisner, and W. Mathis

Leibniz Universität Hannover, Institut für Theoretische Elektrotechnik, Appelstr. 9A, 30167 Hannover, Germany

Zusammenfassung. In dieser Arbeit wird mit einer parallelisierten Version des PCG-Verfahrens (*preconditioned conjugate gradient method*) eine effiziente Technik zur numerischen Lösung großer Gleichungssysteme vorgestellt. Der Anwendungsbereich ist dabei die Simulation numerischer Feldberechnungen (beispielsweise durch die Finite-Elemente-Methode). Es wird ferner auf eine Datenstruktur eingegangen, die für die spezielle Struktur der auftretenden Koeffizientenmatrizen geeignet ist.

1 Einführung

Infolge der fortschreitenden Miniaturisierung integrierter Schaltungen müssen sowohl die Wechselwirkungen einzelner Bauelemente untereinander, als auch der Einfluss auf ihre schaltungstechnische Umgebung schon in der Entwurfsphase berücksichtigt werden. So spielt die elektromagnetische Verträglichkeit (EMV) von Halbleiterbauelementen z.B. in der Automobilelektronik eine große Rolle. Werden die dabei auftretenden elektromagnetischen Felder erst nach der Fertigung mit Messsonden bestimmt, können bei ungünstigen EMV-Eigenschaften kostspielige und zeitintensive Änderungen des Designs notwendig sein. Es bietet sich deshalb an, die benötigten Informationen für besonders kritische Gebiete der integrierten Schaltung schon während des Entwurfsprozesses durch geeignete Simulationsverfahren zu analysieren. Auch wenn dieses Vorgehen durch die Verlagerung der EMV-Analyse vor die Produktion zu einer erheblichen Zeitersparniss führen kann, handelt es sich dabei, infolge der hohen Anzahl der Strukturen auf einem Chip, um ein sehr rechenintensives Verfahren. Es werden deshalb in dieser Arbeit effiziente Verfahren zur Parallelisierung der numerischen Feldberechung, welche beispielweise mittels der Finiten-Elemente-Methode (FEM) für diese Thematik in Preisner und Mathis (2009) angewendet wird, vorgestellt. Durch die immer stärkere Verbreitung von Multicore-Prozessoren sowohl im Server- als auch im Desktop-Bereich und der damit verbundenen Kostensenkung für die Anschaffung eines solchen parallelen Systems, ist dieses Vergehen von aktueller Bedeutung. Als Plattform soll ein heterogenes Workstation-Cluster dienen. Durch diese hybride Architektur kann eine Parallelisierung auf verschiedenen Ebenen durchgeführt werden. So können Teilprobleme mit einer starken Kopplung auf der Shared-Memory-Architektur einzelner Server berechnet und die Lösungen anschließend zusammengeführt werden.

Correspondence to: M. Barke
(barke@tet.uni-hannover.de)

2 Verfahren der konjugierten Gradienten

Das im Jahre 1952 von Hestenes und Stiefel (Hestenes und Stiefel, 1952) entwickelte Verfahren der konjugierten Gradienten (kurz: CG-Verfahren) dient zur numerischen Lösung des linearen Gleichungssystems (Kanzow, 2005)

$$\mathbf{A} \cdot \mathbf{x} = \mathbf{b}. \tag{1}$$

Das Verfahren macht sich dabei zu Nutze, dass die Minimierung des quadratischen Funktionals

$$Q(\mathbf{x}) = \frac{1}{2}\mathbf{x}^T\mathbf{A}\mathbf{x} - \mathbf{b}^T\mathbf{x} \tag{2}$$

gleich dem Lösen des in Gl. (1) gezeigten Gleichungssystems ist. Dabei sei die Matrix $\mathbf{A}$ symmetrisch und positiv definit (SPD).

Die Zuordnung des CG-Verfahrens zu der Gruppe der *direkten* bzw. *iterativen* Verfahren zur Lösung von linearen Gleichungssystemen ist nicht eindeutig (Kanzow, 2005). Die CG-Methode kann durch ihre Eigenschaft, nach spätestens n

Published by Copernicus Publications on behalf of the URSI Landesausschuss in der Bundesrepublik Deutschland e.V.

Tabelle 1. Pseudo-Code: CG-Verfahren

```
Wähle Startlösung x_0
r_0 := b − a x_0
p_0 := r_0
for k := 1 to k_max do
    α_k := (r_{k−1} * r_{k−1}) / (p_{k−1} * A p_{k−1})
    x_k := x_{k−1} + α_k p_{k−1}
    if 'ausreichend genau' then
        x_{k−1} ist die gesuchte Lösung.
        Beende Schleife
    else
        r_k := r_{k−1} − α A p_{k−1}
        β_k := (r_k * r_k) / (r_{k−1} * r_{k−1})
        p_{k+1} := r_k + β_k p_{k−1}
end for
```

Schritten die exakte Lösung eines LGS zu finden, zu den *direkten* Verfahren gezählt werden (Kanzow, 2005). Für besonders große Gleichungssysteme kann diese Eigenschaft allerdings nicht mehr als erstrebenswert angesehen werden. Man kann sich aber in diesen Fällen zu Nutze machen, dass das CG-Verfahren oft schon nach sehr wenigen Schritten zu einer akzeptablen Näherungslösung führt. Aus diesem Grund wird das CG-Verfahren meistens als ein *iteratives* Verfahren bezeichnet. Im Folgenden ist der prinzipielle Aufbau der CG-Methode dargestellt.

3 Konvergenzeigenschaften

Ziel des CG-Verfahrens ist es, den anfänglichen Fehler

$$\|\mathbf{x} - \mathbf{x}_0\|_A \tag{3}$$

um das ϵ-fache zu verringern ($0 < \epsilon < 1$) (Jung und Langer, 2001). Es gilt:

$$\|\mathbf{x} - \mathbf{x}_k\|_A \leq \epsilon \|\mathbf{x} - \mathbf{x}_0\|_A. \tag{4}$$

Ferner lässt sich dieser Approximationsfehler folgendermaßen abschätzen:

$$\frac{\|\mathbf{x} - \mathbf{x}_k\|_A}{\|\mathbf{x} - \mathbf{x}_0\|_A} \leq 2 \left(\frac{\sqrt{\kappa(\mathbf{A})} - 1}{\sqrt{\kappa(\mathbf{A})} + 1} \right)^k. \tag{5}$$

Dabei steht k für die Anzahl der Iterationen und $\kappa(\mathbf{A})$ für die Kondition der Matrix $\mathbf{A}$. Um unter Einhaltung des Faktors ϵ die Anzahl der zur Lösung benötigten Schritte abschätzen zu können, kann nach Deuflhard und Hohmann (2000) und Schwarz und Köckler (2007) die aus Gl. (4) und Gl. (5) folgende Ungleichung

$$\left(\frac{\sqrt{\kappa(\mathbf{A})} + 1}{\sqrt{\kappa(\mathbf{A})} - 1} \right)^k \leq \frac{2}{\epsilon} \tag{6}$$

mit $\theta = \left(\frac{\sqrt{\kappa(\mathbf{A})} - 1}{\sqrt{\kappa(\mathbf{A})} + 1} \right)$ nach k aufgelöst werden.

$$k \leq \log_\theta \left(\frac{2}{\epsilon} \right) = \frac{\ln\left(\frac{2}{\epsilon}\right)}{\ln \theta} \tag{7}$$

Mit Hilfe der für den natürlichen Logarithmus geltenden Beziehung

$$\ln\left(\frac{a+1}{a-1} \right) > \frac{2}{a} \quad \text{für} \quad a > 1, \tag{8}$$

kann Gl. (7) zu

$$k \leq \frac{1}{2} \sqrt{\kappa(\mathbf{A})} \ln\left(\frac{2}{\epsilon} \right) \tag{9}$$

vereinfacht werden. Dies stellt eine obere Schranke für die Anzahl der benötigten Iterationen dar und lässt erkennen, dass bei gleichbleibender Genauigkeit (ϵ) eine weitere Verringerung der Schrittanzahl nur durch Senkung der Kondition $\kappa(\mathbf{A})$ möglich ist. Dies kann durch eine geeignete Vorkonditionierung erreicht werden, die im nächsten Abschnitt genauer vorgestellt werden soll.

4 Vorkonditionierung

Um die Konvergenzeigenschaften des CG-Verfahrens zu verbessern, wird bei einer Vorkonditionierung das lineare Gleichungssystem $\mathbf{A}\cdot\mathbf{x}=\mathbf{b}$ mit Hilfe einer regulären Matrix $\mathbf{C} \in \mathbb{R}^{n \times n}$ in eine äquivalente Form gebracht.

$$\mathbf{C}^{-1}\mathbf{A}\mathbf{C}^{-T}\mathbf{C}^T\mathbf{x} = \mathbf{C}^{-1}\mathbf{b} \tag{10}$$

Fasst man die einzelnen Komponenten von Gl. (10) zu folgenden Ersatzgrößen zusammen,

$$\tilde{\mathbf{A}} := \mathbf{C}^{-1}\mathbf{A}\mathbf{C}^{-T}, \quad \tilde{\mathbf{x}} := \mathbf{C}^T\mathbf{x}, \quad \tilde{\mathbf{b}} := \mathbf{C}^{-1}\mathbf{b} \tag{11}$$

ergibt sich mit Gl. (12) ein neues transformiertes Gleichungssystem. Die neue Systemmatrix $\tilde{\mathbf{A}}$ ist durch die Anwendung einer Kongruenztransformation wie das Ausgangssystem symmetrisch und positiv definit, so dass das CG-Verfahren weiterhin anwendbar ist.

$$\tilde{\mathbf{A}}\tilde{\mathbf{x}} = \tilde{\mathbf{b}} \tag{12}$$

Für die Auswahl einer geeigneten Matrix $\mathbf{C}$ soll folgende Beziehung beachtet werden:

$$\kappa_2(\tilde{\mathbf{A}}) = \kappa_2(\mathbf{C}^{-1}\mathbf{A}\mathbf{C}^{-T}) \ll \kappa_2(\mathbf{A}) \tag{13}$$

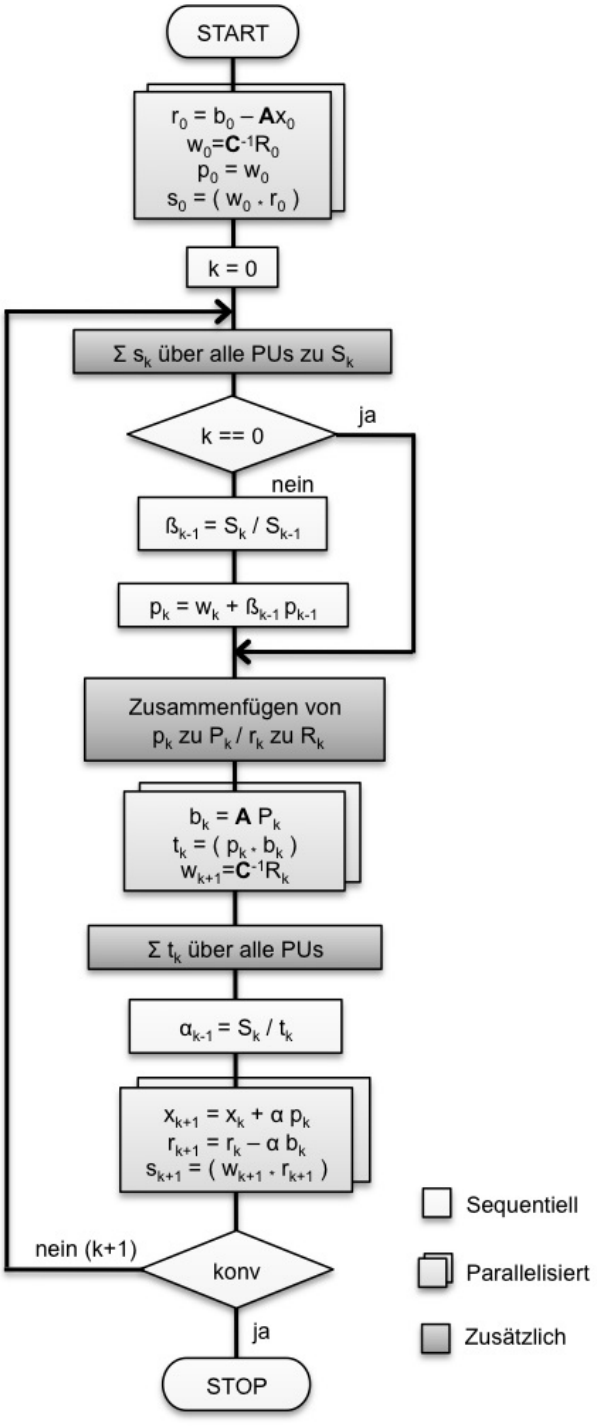

Abb. 1. Flussdiagramm PCG-Verfahren.

Nach Schwarz und Köckler (2007) kann die zu $\tilde{\mathbf{A}}$ ähnliche Matrix $\mathbf{K}$ eingeführt werden. Mit

$$\begin{aligned}\mathbf{K} &:= \mathbf{C}^{-T}\tilde{\mathbf{A}}\mathbf{C}^{T} = \mathbf{C}^{-T}\mathbf{C}^{-1}\mathbf{A} \\ &= (\mathbf{C}\mathbf{C}^{T})^{-1}\mathbf{A} =: \mathbf{M}^{-1}\mathbf{A},\end{aligned} \tag{14}$$

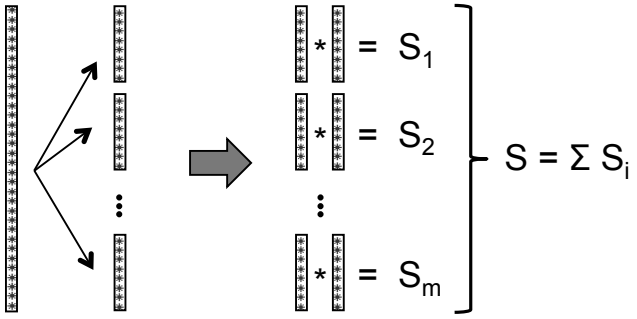

Abb. 2. Parallelisierung des Skalarproduktes.

wird deutlich, dass die beste Vorkonditioniermatrix $\mathbf{M}^{-1} := \left(\mathbf{C}\mathbf{C}^{T}\right)^{-1}$ die Inverse der ursprünglichen Systemmatrix $\mathbf{A}$ wäre, da für die Kondition der Matrix $\tilde{\mathbf{A}}$ gelten würde

$$\kappa(\tilde{\mathbf{A}}) = \kappa(\mathbf{A}^{-1}\mathbf{A}) = \kappa(\mathbf{I}) = 1 \tag{15}$$

und somit der in Gl. (13) geforderte Unterschied zwischen $\kappa_2(\tilde{\mathbf{A}})$ und $\kappa_2(\mathbf{A})$ maximal wäre.

Da die Berechnung der Inversen der Matrix $\mathbf{A}$ für sehr große Gleichungssysteme aber zu aufwendig wäre, wird eine der Inversen angenäherte Vorkonditioniermatrix gesucht. Für einen Überblick über die dafür geeigneten Verfahren sei auf Jung und Langer (2001) verwiesen.

5 Parallelisierungsansätze

Für die im Folgenden genauer erläuterten Parallelisierungsansätze soll von einem Gleichungssystem mit *n* Unbekannten sowie *m* zur Verfügung stehenden Recheneinheiten (PUs) ausgegangen werden. Bei einer Parallelisierung des CG-Verfahrens mit Vorkonditionierung (PCG), dessen Flussdiagramm in Abb. 1 dargestellt ist, muss den rechenintensivsten Operationen besondere Beachtung geschenkt werden. Hierzu zählen in erster Linie das Matrix-Vektor-Produkt $\mathbf{b}_k = \mathbf{A} \cdot \mathbf{p}_k$ sowie mehrere Skalarprodukte. Die der Parallelisierung zugrunde liegenden Konzepte werden in der Literatur an verschiedenen Stellen wie z.B. in Barrett et al. (1994), Takahashi et al. (1998) erläutert, so dass es vielmehr um eine Anpassung an das jeweilige System gehen muss.

Zur parallelen Berechnung eines Skalarproduktes werden die beteiligten Vektoren in *m* Abschnitte aufgeteilt und die Teilvektoren auf die verschiedenen PUs verteilt. Im Anschluss werden die jeweiligen Teilsummen in den einzelnen PUs berechnet und die Gesamtsumme anschließend in einem sequentiellen Abschnitt des Programms gebildet (siehe Abb. 2).

Bei der Parallelisierung des Matrix-Vektor-Produktes $\mathbf{b}_k = \mathbf{A} \cdot \mathbf{p}_k$ wird die Matrix $\mathbf{A}$ gemäß Abb. 3 in *m* (hier: 4) Abschnitte aufgeteilt und an die verschieden PUs verteilt.

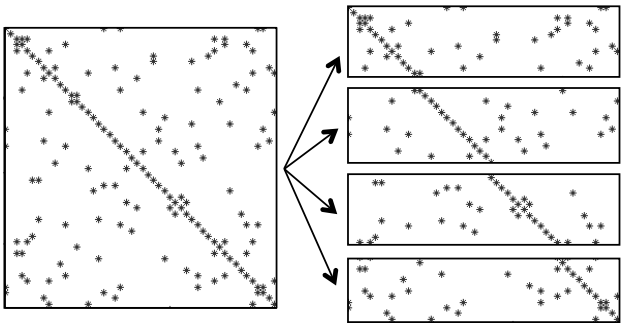

Abb. 3. Aufteilung in Matrix-Abschnitte.

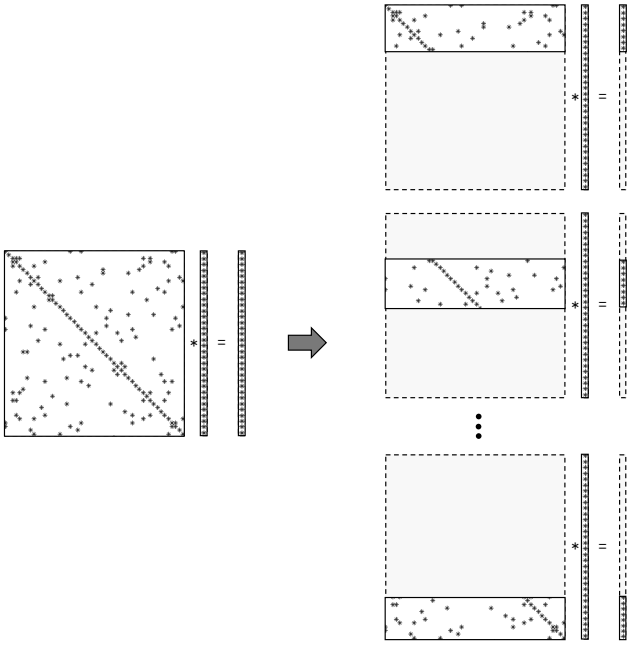

Abb. 4. Berechnung der Teilprodukte.

Um die Teilprodukte, wie in Abb. 4 gezeigt, bilden zu können, muss jede PU zusätzlich über eine lokale Kopie des Vektors $\mathbf{p}_k$ verfügen.

Im Anschluss werden die n/m langen Teilprodukte gemäß Abb. 5 zum Vektor $\mathbf{b}_k$ zusammengefügt. Das Bilden des Gesamtvektors aus den einzelnen Teilvektoren wird, wie schon die Summierung der Teil-Skalarprodukte, sequentiell durchgeführt. Dies verdeutlicht, dass für die Parallelisierung einer Operation und somit für die Beschleunigung des Programms immer auch ein zusätzlicher sequentieller Anteil hinzugefügt werden muss.

6 Speicherung schwachbesetzter Matrizen

Mit zunehmender Größe des zu lösenden Gleichungssystems $\mathbf{A}\cdot\mathbf{x}=\mathbf{b}$ wird die Wahl des zur Speicherung der Matrix $\mathbf{A}$ verwendeten Verfahrens von immer größerer Bedeutung. Die gewählte Speichermethode hat nicht nur starke Auswirkungen auf die maximale Größe des Systems, sondern auch auf die Performance des gesamten Lösungsverfahrens.

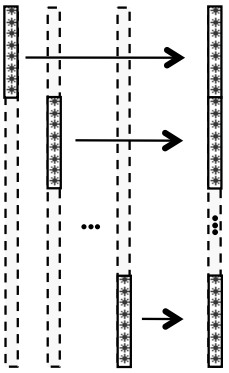

Abb. 5. Zusammenfügen der Teilvektoren.

Die einfachste Methode zum Speichern einer $(n\times n)$-Matrix mit n^2 Elementen ist, jedes Element der Matrix an einer direkt adressierbaren Position in einem n^2 Elemente umfassenden Array abzuspeichern. Für dicht besetzte Matrizen spricht nichts gegen eine solche Speicherung, zumal Operationen mit dieser Datenstruktur sehr einfach sind.

Da die Anzahl der Nicht-Null-Elemente einer Zeile der Koeffizientenmatrix bei Verfahren für die numerische Feldberechnung, wie z.B. der Finiten Elemente Methode (FEM), nicht im gleichen Maße ansteigt wie die Größe des Gleichungssystems, sind die auftretenden Matrizen sehr schwach besetzt. Man spricht in diesem Fall auch von sog. *Sparse*-Matrizen. Für solche Matrizen bietet es sich daher an, nur die Nicht-Null-Elemente zu speichern. Neben dem in dieser Arbeit verwendeten CRS-Verfahren (compressed row storage) gibt es noch einige weitere Techniken, wie das CCS- (compressed column storage), das JDS- (jagged diagonal storage) oder das SKS-Verfahren (skyline storage), auf die im Folgenden nicht weiter eingegangen werden soll (Barrett et al., 1994; Langer, 2003).

Zum Speichern einer $(n\times n)$-Matrix $\mathbf{A}$ werden beim CRS-Verfahren drei Arrays benötigt (siehe Abb. 6), deren Länge sich nach den Matrix-Eigenschaften richtet. Die Anzahl der Nicht-Null-Einträge soll im Folgenden mit *nnz* (engl. *number of non-zeros*) bezeichnet werden. Dabei speichert das *nnz*-lange Double-Array *value* die Nicht-Null-Elemente (Zeile für Zeile). Das ebenfalls *nnz* Werte umfassende Integer-Array *index* speichert die Spaltenindizes der in *value* gespeicherten Nicht-Null-Elemente. Das $(n+1)$ lange Integer-Array *ptr* zeigt jeweils an den Anfang einer neuen Zeile in *value* und *index*.

Für ein besseres Verständnis der CRS-Technik ist in Abb. 6 die Belegung der Arrays *value*, *index* und *ptr* für die Matrix $\mathbf{A}$ dargestellt.

$$\mathbf{A} = \begin{pmatrix} \mathbf{3\,9}\,0\;\;0 \\ 0\,\mathbf{1\,2}\;\;0 \\ 0\,0\,\mathbf{1}\;\;\mathbf{7} \\ \mathbf{8}\,0\,\mathbf{2\,10} \end{pmatrix} \tag{16}$$

Durch die Einführung einer derartigen Datenstruktur kann nicht nur der benötigte Speicherplatz drastisch gesenkt werden, sondern auch die Performance erhöht werden. Diese Steigerung wird dadurch erreicht, dass bei einer Matrix-Vektor- bzw. Matrix-Matrix-Multiplikation nur noch die Nicht-Null-Einträge für die Multiplikation verwendet werden. Eine mittels OpenMP parallelisierte Umsetzung des Matrix-Vektor-Produktes **b**=**A**·**p** unter Verwendung des CRS-Speicherformates wird in folgendem Codeausschnitt gezeigt.

```
1  #pragma omp parallel for private(i,j,sum)
2  for(i = 0; i < A->height; i++)
3  {
4    sum = 0;
5    for(j = A->ptr[i]; j < A->ptr[i+1]; j++)
6          sum += A->value[j] * p[A->index[j]];
7
8        b[i] = sum;
9  }
```

Listing 1. CRS-Format: Matrix-Vektor-Produkt

Es ist dabei zu beachten, dass die indirekte Adressierung des CRS-Formates starken Einfluss auf die innere Schleife des Produktes hat (vgl. Listing 1 Zeilen 5 und 6).

7 Zusammenfassung

In der vorliegenden Arbeit wurde eine Parallelisierung des vorkonditionierten konjugierten Gradienten-Verfahrens (PCG) vorgestellt. Diese Arbeit wurde dabei durch die äußerst rechenintensive numerische Feldsimulation (Preisner und Mathis, 2008), beispielsweise mit der Finiten-Elemente-Methode, motiviert. Nach einer Einführung in das CG-Verfahren wurde die Notwendigkeit einer Vorkonditionierung näher erläutert. Weiterhin wurden Parallelisierungsstrategien für das PCG-Verfahren vorgestellt. Im Anschluss wurde mit der *compressed row storage* (CRS) eine effiziente Datenstruktur zur Speicherung von *sparse* Matrizen eingeführt, bevor auf erzielte Ergebnisse eingegangen wurde.

In weiterführenden Arbeiten kann – aufbauend auf den in dieser Arbeit gesammelten Erfahrungen – mit der sogenannten *Domain-Dekomposition* (DD) eine weitere Methode zur Parallelisierung numerischer Lösungsverfahren untersucht werden (Langer, 2003). Durch das Konzept der Domain-Dekomposition kann die Koeffizienten-Matrix **A**, durch Ausnutzung der geometrischen Beziehungen der einzelnen Elemente, bereits beim Aufstellen des Gleichungssystems in eine gut zu parallelisierende Form gebracht werden.

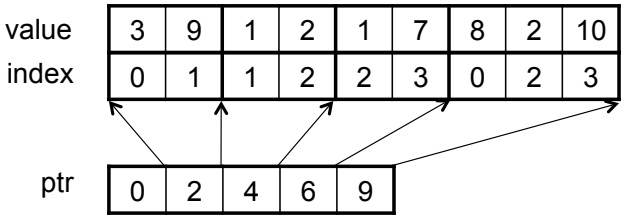

Abb. 6. Beispiel zum CRS-Format.

Literatur

Barrett, R., Berry, M., Chan, T., et al.: Templates for the Solution of Linear Systems: Building Blocks for Iterative Methods, SIAM, Philadelphia, 1994.

Deuflhard, P. und Hohmann, A.: Numerische Mathematik I: Eine algorithmisch orientierte Einführung, de Gruyter, 2000.

Hestenes, M. R. und Stiefel, E.: Methods of conjugate gradients for solving linear systems, J. Res. Nat. Bur. Stand., 49(6), 409–436, 1952.

Jung, M. und Langer, U.: Methode der finiten Elemente für Ingenieure, Teubner, 2001.

Kanzow, C.: Numerik linearer Gleichungssysteme: Direkte und iterative Verfahren, Springer-Verlag, 2005.

Douglas, C. C., Haase, G., und Langer, U.: A Tutorial on Elliptic PDE Solvers and Their Parallelization, SIAM, Philadelphia, 2003.

Preisner, T. und Mathis, W.: Magnetic Force Calculations Applied to Magnetic Force Microscopy, Proc. SCEE, Helsinki, accepted, 2008.

Preisner, T. und Mathis, W.: A Three Dimensional FEM-BEM Approach for the Simulation of Magnetic Force Microscopes, Proc. PIERS Beijing, accepted, 2009.

Schwarz, H. R. und Köäckler, N.: Numerische Mathematik, 6., überarbeitete Auflage, Teubner, 2007.

Takahashi, N., Nakano, T., Fujiwara, K., et al.: Investigation of parallel computation in 3D magnetic field analysis on distributed memory type of multiprocessors containing 296 PUs, COMPEL, 1998.