Lösung von Sequenzen großer dünn-besetzter Hermitescher linearer Gleichungssysteme

Martin Neuenhofen *

20. Juli 2016

Zusammenfassung

In dieser Seminar-Arbeit werden drei verschiedene Krylov-Subspace-Recycling-Verfahren vorgestellt und miteinander verglichen. Dabei wird besonders der Fall betrachtet, dass die Systemmatrix Hermitesch ist.

Krylov-Subspace-Recycling-Verfahren sind eine wichtige Verfahrensklasse zur Lösung von Sequenzen linearer Gleichungssysteme, wie sie beispielsweise aus der Diskretisierung instationärer oder nicht-linearer partieller Differentialgleichungen resultieren.

Die im Aufsatz behandelten Verfahren sind im Einzelnen RGCR, R-MINRES und SR-PCR-ap. Der Aufsatz richtet sich an Studenten im Grund- und Aufbaustudium.

Inhaltsverzeichnis

*Martin.Peter.Neuenhofen@rwth-aachen.de

1 Einleitung

1.1 Problemstellung

In vielen numerischen Anwendungen treten nicht einzelne lineare Gleichungssysteme sondern Sequenzen von linearen Gleichungssystemen auf. Dies ist beispielsweise der Fall, wenn instationäre partielle Differentialgleichungen (PDE) gelöst werden müssen oder PDEs als Nebenbedingungen in Optimierungsproblemen auftreten. Anwendungen hierfür sind in der Topologie-Optimierung, Modell-Reduktion, Struktur-Dynamik, Schaltkreis-Analyse und Fluid-Dynamik zu finden, vgl. [D1]-[D6].

Diese Arbeit betrachtet Sequenzen der Form

$$\mathbf{A} \cdot \mathbf{x}^{(\iota)} = \mathbf{b}^{(\iota)} \quad \iota = 1, ..., M\,, \tag{1}$$

wobei $\mathbf{A} \in \mathbb{C}^{N \times N}$ regulär ist, und zu jedem Zeitpunkt nur eine der rechten Seiten $\mathbf{b}^{(\iota+1)} \in \mathbb{C}^N$ vorliegt; das heißt für jede rechte Seite muss nacheinander einzeln die Lösung berechnet werden. Krylov-Subspace-Recycling-Verfahren [B2.1, B2.2] sind eine Verfahrensklasse zur Lösung derartiger Sequenzen von linearen Gleichungssystemen.

1.2 Aufbau

Dieser Aufsatz vergleicht zwei verschiedene Krylov-Subspace-Recycling-Verfahren.

In Abschnitt 2 werden zunächst Krylov-Unterraum-Verfahren am Beispiel des GCR-Verfahrens und anschließend das Prinzip kurzer Rekursionen am Beispiel des CR-Verfahrens ausführlich rekapituliert.

Abschnitt 3 gibt eine Einführung in die Idee von Krylov-Unterraum-Recycling-Verfahren. Dazu wird RGCR als naives Krylov-Subspace-Recycling-Verfahren beschrieben und motiviert. Anschließend werden praktische Schwächen von RGCR im Hinblick auf den Speicherbedarf diskutiert. Daraus motiviert

wird das Recycling-Verfahren R-MINRES vorgestellt, durch dessen Anwendung sich der Speicheraufwand unter gewissem Informationsverlust limitieren lässt.

Abschnitt 4 stellt ein alternatives Krylov-Unterraum-Recycling-Verfahren namens SR-PCR-ap vor, mit dessen Hilfe sich trotz geringen Speicherbedarfs der Informationsverlust, der bei R-MINRES zwingenderweise auftritt, vermeiden lässt.

Im fünften Abschnitt werden die Verfahren R-MINRES und SR-PCR-ap anhand ihrer geometrischen Eigenschaften und ihres praktischen Nutzens bei der Lösung eines Testproblems (Fourier-Gleichung) verglichen.

2 Krylov-Unterraum-Verfahren

Krylov-Unterraum-Verfahren sind iterative Verfahren und Projektionsverfahren zur Lösung von einzelnen linearen Gleichungssystemen. [A1.1] bietet eine Übersicht.

2.1 Motivation

Da bei dünnbesetzten Gleichungssystemen der Fill-in und somit die Rechenzeit und der Speicherbedarf von direkten Verfahren stark von der Besetzungsstruktur der Systemmatrix abhängen, beschänkt man sich bei iterativen Verfahren lediglich darauf, Matrix-Vektor-Produkte mit der Systemmatrix zu evaluieren. Diese sind infolge der Dünnbesetztheit günstig berechenbar.

Gehen wir zum i-ten Schritt des iterativen Lösungsprozesses von einer numerischen Näherungslösung $\mathbf{x}_i$ mit zugehörigem Residuum $\mathbf{r}_i = \mathbf{b} - \mathbf{A} \cdot \mathbf{x}_i$ aus, und ist $\mathbf{M}^{-1}$ eine günstig für Matrix-Vektor-Produkte evaluierbare Approximation an die Inverse von $\mathbf{A}$, dann eignet sich der Korrektur-Ansatz

$$\mathbf{x}_{i+1} := \mathbf{x}_i + \alpha \cdot \mathbf{M}^{-1} \cdot \mathbf{r}_i \,.$$

Wird dieser Korrektur-Ansatz für die Näherungslösung wiederholt durchgeführt, so folgt die besondere Eigenschaft:

$$\mathbf{x}_m \in \mathbf{x}_0 + \mathbf{M}^{-1} \cdot \mathcal{K}_m(\mathbf{A} \cdot \mathbf{M}^{-1}; \mathbf{r}_0) := \mathbf{M}^{-1} \cdot \operatorname*{span}_{k=0,\ldots,m-1} \{(\mathbf{A} \cdot \mathbf{M}^{-1})^k \cdot \mathbf{b}\} \,.$$

$\mathcal{K}_m$ heißt Krylov-Unterraum der Stufe m. $\mathbf{x}_0$ und $\mathbf{r}_0$ sind dabei die Daten zum Zeitpunkt der Initialisierung des iterativen Verfahrens. Für $\mathbf{x}_0$ kann beispielsweise eine Schätzung der exakten Lösung oder $\mathbf{0}$ verwendet werden.

2.2 Projektionsprinzip

Da die Matrix-Vektor-Produkte (oft infolge der Präkonditionierung) den Rechenaufwand dominieren, möchte man im Fall von m Iterationen im Raum $\mathbf{x}_0 + \mathbf{M}^{-1} \cdot \mathcal{K}_m(\mathbf{A} \cdot \mathbf{M}^{-1}; \mathbf{r}_0)$ eine möglichst gute Lösung für $\mathbf{x}$ konstruieren. Krylov-Unterraum-Verfahren erfüllen diesen Anspruch, indem $\mathbf{x} \in \mathbf{x}_0 + \mathcal{K}_m(\mathbf{A}; \mathbf{r}_0)$ nach

gewissen Kriterien optimal gewählt wird. Dazu berechnen diese Verfahren Projektionslösungen.

Projektionslösungen lassen sich wie folgt konstruieren: Wählt man $\mathbf{x} \in \mathbf{x}_0 + \mathcal{U}$, wobei $\mathcal{U}$ ein m-dimensionaler Ansatzraum ist, dann kann $\mathbf{x}$ so gewählt werden, dass $\mathbf{r}$ senkrecht auf einem m-dimensionalen Testraum $\mathcal{W}$ liegt. Liegen Basismatrizen $\mathbf{U}, \mathbf{W} \in \mathbb{C}^{N \times m}$ für $\mathcal{U}, \mathcal{W}$ vor, so lässt sich $\mathbf{x}$ über

$$\mathbf{x} = \mathbf{x}_0 + \mathbf{U} \cdot (\mathbf{W}^H \cdot \mathbf{A} \cdot \mathbf{U})^{-1} \cdot \mathbf{W}^H \cdot \mathbf{r}_0 \tag{2}$$

berechnen. Für Krylov-Unterraum-Verfahren wählt man $\mathcal{U} = \mathbf{M}^{-1} \cdot \mathcal{K}_m(\mathbf{A} \cdot \mathbf{M}^{-1}; \mathbf{r}_0)$.

Der Ansatz wird als Projektion bezeichnet, weil der Urbildraum und der Bildraum von $\mathbf{A}$ jeweils durch eine Basis-Matrix auf kleinere Räume projiziert werden: Man löst effektiv das System

$$\mathbf{W}^H \cdot \mathbf{A} \cdot \mathbf{U} \cdot (\mathbf{x} - \mathbf{x}_0) = \mathbf{W}^H \cdot \mathbf{r}_0$$

über ein exaktes Verfahren.

Die Wahl von $\mathcal{W}$ ist abhängig vom jeweiligen Krylov-Unterraum-Verfahren und beeinflusst stark die Güte der Wahl von $\mathbf{x} \in \mathbf{x}_0 + \mathcal{U}$ sowie die Rechen- und Speicherkosten des Verfahrens.

Im Folgenden werden zwei wichtige Krylov-Unterraum-Verfahren vorgestellt. Anhand dieser Verfahren wird verdeutlicht, welchen Einfluss $\mathcal{W}$ auf die Eigenschaften der Verfahren hat.

2.3 Generalized Conjugate Residual (GCR)

GCR [A2.2] ist ein Krylov-Unterraum-Verfahren für allgemeine Systemmatrizen (d.h. auch im Fall $\mathbf{A}^H \neq \mathbf{A}$ geeignet), das in m Iterationen eine Näherungslösung $\mathbf{x} \in \mathbf{x}_0 + \mathcal{U}$ konstruiert, sodass $\|\mathbf{r}\|_2$ minimiert wird. Die Minimierung von $\|\mathbf{r}\|_2$ folgt durch die Wahl $\mathcal{W} = \mathbf{A} \cdot \mathcal{U}$.

Dazu baut GCR spaltenweise zwei Basis-Matizen $\mathbf{U}, \mathbf{V} \in \mathbb{C}^{N \times m}$ auf, sodass range($\mathbf{U}$) = $\mathcal{U}$ gilt und $\mathbf{V} = \mathbf{A} \cdot \mathbf{U}$ orthogonal ist, d.h. $\mathbf{V}^H \cdot \mathbf{V} = \mathbf{I}$. Durch $\mathbf{W} = \mathbf{V}$ gilt $(\mathbf{W}^H \cdot \mathbf{A} \cdot \mathbf{U}) = \mathbf{I}$ und die Projektionslösung (2) lässt sich leicht berechnen.

Zur Orthogonalisierung von $\mathbf{V}$ kann beispielsweise das Gram-Schmidt-Verfahren verwendet werden. Algo. 1 zeigt eine Implementierung.

Die Wahl von $\mathcal{W}$ bewirkt, dass GCR die Lösung mit kleinstmöglicher Residuum-Norm berechnet, sog. *Residuum-optimale Lösung*. Als Nachteil ergibt sich durch diese Wahl von $\mathcal{W}$ jedoch andererseits, dass GCR ist ein *Langterm-Rekursionsverfahren* ist, d.h. die Orthogonalisierung neuer Spalten $\mathbf{v}$ auf der Matrix $\mathbf{V}$ bzw. die $\mathbf{A}^H \cdot \mathbf{A}$-Konjugierung neuer Spalten $\mathbf{u}$ auf $\mathbf{U}$ kann nicht über einen konstanten Rechen- und Speicheraufwand realisiert werden. Stattdessen muss jede neue Spalte mit allen vorherigen Spalten verrechnet werden, siehe Zeile 7-8 in Algo. 1.

Algorithm 1 Generalized Conjugate Resdiual, Gram-Schmidt-Variante

1: **procedure** GCR($\mathbf{A}, \mathbf{x}, \mathbf{r}$)
2: $\mathbf{U} := []$
3: $\mathbf{V} := []$
4: **while** $\|\mathbf{r}\| > \text{tol}$ **do** *// m = Anzahl Schleifendurchläufe*
5: $\mathbf{u} := \mathbf{M}^{-1} \cdot \mathbf{r}$ *// neue Korrektur-Richtung*
6: $\mathbf{v} := \mathbf{A} \cdot \mathbf{u}$ *// Effekt der Korrektur-Richtung auf $\boldsymbol{r}$*
7: $\boldsymbol{\gamma} := \mathbf{V}^H \cdot \mathbf{v}$ *// Orthogonalisierung*
8: $\mathbf{u} := \mathbf{u} - \mathbf{U} \cdot \boldsymbol{\gamma}$, $\mathbf{v} := \mathbf{v} - \mathbf{V} \cdot \boldsymbol{\gamma}$
9: $\tau := \|\mathbf{v}\|_2$, $\mathbf{u} := 1/\tau \cdot \mathbf{u}$, $\mathbf{v} := 1/\tau \cdot \mathbf{v}$ *// Normierung*
10: $\mathbf{U} := [\mathbf{U}, \mathbf{u}]$, $\mathbf{V} := [\mathbf{V}, \mathbf{v}]$
11: $\omega := \mathbf{v}^H \cdot \mathbf{r}$, $\mathbf{x} := \mathbf{x} + \omega \cdot \mathbf{u}$, $\mathbf{r} := \mathbf{r} - \omega \cdot \mathbf{v}$
12: **end while**
13: **return** $\mathbf{x}, \mathbf{r}, \mathbf{U}, \mathbf{V}$
14: **end procedure**

Muss eine große Anzahl m von Iterationen durchgeführt werden, so sind Langterm-Rekursionsverfahren nicht praktikabel, da der Speicherbedarf für die Basismatrizen und der Rechenaufwand für die Orthogonalisierung anwachsen und die Speicher- und Rechenkapazitäten des Computers übersteigen. Daher ist GCR in der oben beschriebenen Form oft nicht in der Praxis anwendbar.

2.4 Kurze Rekursionen in Conjugate Residual (CR)

Wir zeigen im Folgenden, dass unter gewissen Eigenschaften von $\mathbf{M}$ und $\mathbf{A}$ die Einträge der Vektoren $\boldsymbol{\gamma} \in \mathbb{C}^i$ in Algo. 1 Zeile 7 mit Ausnahme des jeweils letzten Eintrags zu Null werden. Dazu betrachten wir zunächst den Fall $\mathbf{M} = \mathbf{I}$, d.h. der Präkonditionierer entspricht der Einheitsmatrix bzw. es wird kein Präkonditionierer benutzt.

Stellt man bzgl. $\mathbf{A}$ und eines Vektors $\mathbf{v}_1$ eine orthogonale Basismatrix $\mathbf{V}_m$ des Krylov-Unterraums $\mathcal{K}_m(\mathbf{A}; \mathbf{v}_1)$ auf, so können die Spalten $\mathbf{v}_1, ..., \mathbf{v}_m$ der Matrix $\mathbf{V}_m \in \mathbb{C}^{N \times m}$ wie folgt gewählt werden:

for $i = 1, 2, ..., m-1$ **do**

$$h_{i+1,i} \cdot \mathbf{v}_{i+1} = \mathbf{A} \cdot \mathbf{v}_i - \sum_{j=1}^{i} \mathbf{v}_j \cdot h_{j,i} \tag{3}$$

end for

Das heißt die jeweils nächste Spalte $\mathbf{v}_{i+1}$ kann aus den vorherigen Spalten $\mathbf{v}_1, ..., \mathbf{v}_i$ berechnet werden, indem der Vektor $\mathbf{A} \cdot \mathbf{v}_i$ als Matrix-Vektor-Produkt berechnet, dann zunächst auf den vorherigen Spalten über eine Linearkombination orthogonalisiert (die Summe im rechten Ausdruck, z.B. nach Gram-Schmidt), und schließlich auf die Länge 1 skaliert (geeignete Wahl von $h_{i+1,i}$ auf der linken Seite der Gleichung) wird.

Schreibt man die Hessenberg-Matrix

$$\overline{\mathbf{H}}_m = \left[\begin{array}{cccccc} h_{1,1} & h_{1,2} & \dots & \dots & h_{1,m-1} & h_{1,m} \\ h_{2,1} & h_{2,2} & \dots & \dots & h_{2,m-1} & h_{2,m} \\ 0 & h_{3,2} & \dots & \dots & h_{3,m-1} & h_{3,m} \\ 0 & 0 & \ddots & & \vdots & \vdots \\ \vdots & \vdots & \ddots & \ddots & \vdots & \vdots \\ 0 & 0 & \dots & 0 & h_{m,m-1} & h_{m,m} \\ \hline 0 & 0 & \dots & 0 & 0 & h_{m+1,m} \end{array}\right] \equiv \left[\begin{array}{c} \mathbf{H}_m \\ \hline \mathbf{0}\ h_{m+1,m} \end{array}\right] \in \mathbb{C}^{(m+1)\times m},$$

so lässt sich die folgende *Arnoldi-Zerlegung* [A1.1, Kap. 6.3]

$$\mathbf{A} \cdot \mathbf{V}_m = \mathbf{V}_{m+1} \cdot \overline{\mathbf{H}}_m \tag{4}$$

aufstellen, und daraus wiederum durch Multipliation mit $\mathbf{V}_m^H$ von links:

$$\mathbf{V}_m^H \cdot \mathbf{A} \cdot \mathbf{V}_m = \mathbf{V}_m^H \cdot \mathbf{V}_{m+1} \cdot \overline{\mathbf{H}}_m \equiv \mathbf{H}_m\,. \tag{5}$$

Aus (5) wird ersichtlich, dass $\mathbf{H}_m$ Hermitesch ist, genau dann wenn $\mathbf{A}$ Hermitesch ist. In diesem Fall wird $\mathbf{H}_m$ zu einer Tridiagonalmatrix. Die Summe in der Orthgonalisierung aus (3) verkürzt sich dann auf zwei Summanden.

Die resultierende kurze Rekursion wird *Lanczos-Rekursion* [A1.1, Kap. 6.6] genannt:

for $i = 1, 2, ..., m-1$ **do**
$\beta_{i+1} \cdot \mathbf{v}_{i+1} = \mathbf{A} \cdot \mathbf{v}_i - \mathbf{v}_i \cdot \alpha_i - \mathbf{v}_{i-1} \cdot \beta_i$
end for

Anstelle der Hessenbergmatrizen $\mathbf{H}_m, \overline{\mathbf{H}}_m$ notieren wir die Tridiagonalmatrizen $\mathbf{T}_m, \overline{\mathbf{T}}_m$ mit

$$\overline{\mathbf{T}}_m = \left[\begin{array}{ccccc} \alpha_1 & \beta_2 & 0 & \dots & 0 \\ \beta_2 & \alpha_2 & \beta_3 & \ddots & \vdots \\ 0 & \beta_3 & \ddots & \ddots & 0 \\ \vdots & \ddots & \ddots & \alpha_{m-1} & \beta_m \\ 0 & \dots & 0 & \beta_m & \alpha_m \\ \hline 0 & \dots & 0 & 0 & \beta_{m+1} \end{array}\right] \equiv \left[\begin{array}{c} \mathbf{T}_m \\ \hline \mathbf{0}\ \beta_{m+1} \end{array}\right] \in \mathbb{C}^{(m+1)\times m}.$$

Anstelle von (4) gilt

$$\mathbf{A} \cdot \mathbf{V}_m = \mathbf{V}_{m+1} \cdot \overline{\mathbf{T}}_m\,. \tag{6}$$

Lanczos-Rekursion mit Präkonditionierung Im Folgenden möchten wir anreißen, wie die Lanczos-Rekursion auf präkonditionierte Systeme angewendet werden kann. Das Problem hierbei ist, dass durch die Präkonditionierungsmatrix i.A. die Symmetrie von $\mathbf{A}$ aufgehoben wird.

Falls für $\mathbf{M}$ eine Cholesky-Zerlegung $\mathbf{M} = \mathbf{L} \cdot \mathbf{L}^H$ existiert, so kann $\mathbf{A}$ in (4) durch $\mathbf{L}^{-H} \cdot \mathbf{A} \cdot \mathbf{L}^{-1}$ ersetzt werden. Weil $\mathbf{L}^{-H} \cdot \mathbf{A} \cdot \mathbf{L}^{-1}$ weiterhin Hermitesch ist, lässt sich

$$\mathbf{L}^{-H} \cdot \mathbf{A} \cdot \mathbf{L}^{-1} \cdot \mathbf{Z}_m = \mathbf{Z}_{m+1} \cdot \overline{\mathbf{T}}_m$$

mit orthogonaler Matrix $\mathbf{Z}_{m+1}$ analog über kurze Rekursionen aufstellen. Üblicherweise liegt die Zerlegungsmatrix $\mathbf{L}$ jedoch nicht vor, beispielsweise wenn $\mathbf{M}^{-1}$ ein Mehrgitter-Verfahren ist. Sofern $\mathbf{M}$ jedoch positiv definit ist, exisitert $\mathbf{L}$. Zum Zwecke einer theoretischen Herleitung kann daher in obiger Gleichung entweder $\mathbf{L}^{-1} \cdot \mathbf{Z}_i =: \mathbf{V}_i \ \forall i \in \mathbb{N}$ substituiert werden, woraus

$$\mathbf{M}^{-1} \cdot \mathbf{A} \cdot \mathbf{V}_m = \mathbf{V}_{m+1} \cdot \overline{\mathbf{T}}_m \tag{7}$$

folgt; oder alternativ kann $\mathbf{Z}_i =: \mathbf{L}^{-H} \cdot \mathbf{V}_i \ \forall i \in \mathbb{N}$ substituiert werden, woraus

$$\mathbf{A} \cdot \mathbf{M}^{-1} \cdot \mathbf{V}_m = \mathbf{V}_{m+1} \cdot \overline{\mathbf{T}}_m \tag{8}$$

folgt. In beiden Fällen ist die Matrix $\mathbf{V}_m$ ***M**-orthogonal*, d.h. $\mathbf{V}_m^H \cdot \mathbf{M} \cdot \mathbf{V}_m = \mathbf{I}$. $\mathbf{L}$ wird zum Aufstellen der Zerlegung nicht mehr benötigt.

Ersetzt man das Residuum $\mathbf{r}$ in Algo. 1 entweder durch $\mathbf{M} \cdot \mathbf{r}$ bzw. durch $\mathbf{M}^{-1} \cdot \mathbf{r}$ (indem man das Gleichungssystem von links entsprechend multipliziert), so folgt aus (7) bzw. (8), dass die Einträge der Vektoren $\gamma \in \mathbb{C}^i$ in Algo. 1 Zeile 7 mit Ausnahme des jeweils letzten Eintrags zu Null werden. Darüber hinaus besteht die Möglichkeit, pro Schleifendurchlauf in $\mathcal{O}(1)$ die Koeffizienten α_i, β_{i+1} (für jeweiligen Schleifenindex i) von $\overline{\mathbf{T}}_m$ zu bestimmen. Algo. 2 zeigt dies durch eine entsprechende Implementierung aus unserer Arbeit [B3.2]. PCR steht für *preconditioned conjugate residual*. Für $\mathbf{M} = \mathbf{I}$ bezeichnen wir das Verfahren mit CR.

3 Krylov-Unterraum-Recycling

Krylov-Unterraum-Recycling-Verfahren sind iterative Verfahren zur Lösung von (1), die von Krylov-Unterraum-Verfahren abstammen.

3.1 Kernidee: Das RGCR-Verfahren

Die grundlegende Idee von Krylov-Unterraum-Recycling besteht darin, berechnete Informationen aus vorangegangenen Lösungsprozessen für nachfolgende Systeme wiederzuverwenden, vgl. [D3, D5, D6]. Wurde beispielsweise mit GCR ein System $\mathbf{A} \cdot \mathbf{x}^{(1)} = \mathbf{b}^{(1)}$ in m Iterationen näherungsweise gelöst, so liegen anschließend Basismatrizen $\mathbf{U}, \mathbf{V} \in \mathbb{C}^{N \times m}$ vor, die für ein nachfolgendes System $\mathbf{A} \cdot \mathbf{x}^{(2)} = \mathbf{b}^{(2)}$ wiederverwendet werden können. Konkret hat GCR (im Falle $\mathbf{M} = \mathbf{I}$) die Lösung

$$\mathbf{x}^{(1)} = \mathbf{U} \cdot \mathbf{V}^H \cdot \mathbf{b}^{(1)}$$

Algorithm 2 Preconditioned Conjugate Residual variant

1: **procedure** PCR($\mathbf{M}, \mathbf{A}, \mathbf{b}, \mathbf{x}_0, m$)
2: $\tau := 1, \quad \mathbf{u} := \mathbf{0}, \quad \mathbf{v} := \mathbf{0}$
3: $\mathbf{x} := \mathbf{x}_0, \quad \hat{\mathbf{r}} := \mathbf{M}^{-1}(\mathbf{b} - \mathbf{A}\mathbf{x}_0)$ // $\mathbf{r}_j = \mathbf{b} - \mathbf{A}\mathbf{x}_j$, $j = 0, \ldots, m+1$
4: **for** $j := 0, 1, 2, \ldots, m$ **do** // *use* $\|\hat{\mathbf{r}}\|_2$ *or* θ *as termination criterion*
5: $[\hat{\mathbf{v}}, \hat{\mathbf{d}}, \hat{\mathbf{u}}] :=$`A_func`$(\mathbf{M}, \mathbf{A}, \hat{\mathbf{r}})$
6: // `A_func` *is:* $\hat{\mathbf{u}} := \hat{\mathbf{r}}$, $\hat{\mathbf{d}} := \mathbf{A}\hat{\mathbf{r}}$, $\hat{\mathbf{v}} := \mathbf{M}^{-1}\hat{\mathbf{d}}$
7: $\xi := \langle \hat{\mathbf{d}}, \mathbf{v} \rangle$
8: $\mathbf{u} := \hat{\mathbf{u}} - \xi/\tau\, \mathbf{u}, \quad \mathbf{v} := \hat{\mathbf{v}} - \xi/\tau\, \mathbf{v}$
9: $\hat{\tau} := \langle \hat{\mathbf{d}}, \mathbf{v} \rangle$
10: // *columns of* $\mathbf{U}$*:* $\mathbf{u}_{j+1} := 1/\sqrt{\hat{\tau}}\, \mathbf{u}$
11: // *columns of* $\mathbf{V}$*:* $\mathbf{v}_{j+1} := 1/\sqrt{\hat{\tau}}\, \mathbf{v}$
12: // *but columns of* $\mathbf{D}$*:* $\mathbf{d}_{j+1} \neq 1/\sqrt{\hat{\tau}}\, \hat{\mathbf{d}}$ *- not required*
13: // *estimator:* $\theta := \hat{\tau} + \xi^2/\tau \equiv \|\mathbf{M}^{-1}\mathbf{A}\mathbf{r}_j\|^2_{\mathbf{M}}$
14: $\alpha_j := (\tau - \xi)/\eta, \quad \beta_{j+1} := -\sqrt{\tau\hat{\tau}}/\eta$ // *only if* $\overline{\mathbf{T}}$ *is of interest*
15: $\eta := \langle \hat{\mathbf{d}}, \hat{\mathbf{r}} \rangle, \quad \tau := \hat{\tau}$
16: $\mathbf{x} := \mathbf{x} + \eta/\tau\, \mathbf{u}, \quad \hat{\mathbf{r}} := \hat{\mathbf{r}} - \eta/\tau\, \mathbf{v}$ // $\hat{\mathbf{r}} \equiv \mathbf{M}^{-1}\mathbf{r}_{j+1}$
17: **end for**
18: **return x**
19: **end procedure**

berechnet und kann in gleicher Weise

$$\mathbf{x}^{(\iota)} = \mathbf{U} \cdot \mathbf{V}^H \cdot \mathbf{b}^{(\iota)}$$

für jede nachfolgende rechte Seite ($\iota = 2, 3, ...$) bestimmen. Anschließend lässt sich $\mathbf{x}^{(\iota)}$ durch weitere GCR-Schritte verbessern, indem die Basismatrizen $\mathbf{U}, \mathbf{V}$ iterativ weiter vergrößert werden. Dieser Ansatz führt auf das Verfahren RGCR [B2.1], s. Algo. 3. Dieser Algorithmus unterscheidet sich von GCR lediglich in den ersten beiden Zeilen: Die Initialisierung wird durch die Konstruktion einer Residuum-optimalen Näherungslösung zur recycelten Suchraum-Basis ersetzt. In der anschließenden while-Schleife wird die recycelte Basis iterativ vergrößert.

Wird eine längere Sequenz von Gleichungssystemen gelöst, so baut RGCR über die gesamte Sequenz (1) einen großen Suchraum auf, der sukzessive anwächst. Auf diese Weise bleiben alle Informationen aus zurückliegenden Rechenprozessen erhalten. Dadurch lässt sich die Anzahl benötigter Iterationen oft stark reduzieren, wie wir an folgendem Beispiel zeigen.

Algorithm 3 Recycled Generalized Conjugate Resdiual

procedure RGCR($\mathbf{A}, \mathbf{x}, \mathbf{r}, \mathbf{U}, \mathbf{V}$)
  $\boldsymbol{\gamma} := \mathbf{V}^H \cdot \mathbf{r}$ *// Residuum-optimale Projektionslösung*
  $\mathbf{x} := \mathbf{x} + \mathbf{U} \cdot \boldsymbol{\gamma}$, $\mathbf{r} := \mathbf{r} - \mathbf{V} \cdot \boldsymbol{\gamma}$
  while $\|\mathbf{r}\| >$ tol **do** *// Projektionsraum iterativ erweitern*
    $\mathbf{u} := \mathbf{M}^{-1} \cdot \mathbf{r}$
    $\mathbf{v} := \mathbf{A} \cdot \mathbf{u}$
    $\boldsymbol{\gamma} := \mathbf{V}^H \cdot \mathbf{v}$
    $\mathbf{u} := \mathbf{u} - \mathbf{U} \cdot \boldsymbol{\gamma}$, $\mathbf{v} := \mathbf{v} - \mathbf{V} \cdot \boldsymbol{\gamma}$
    $\tau := \|\mathbf{v}\|_2$, $\mathbf{u} := 1/\tau \cdot \mathbf{u}$, $\mathbf{v} := 1/\tau \cdot \mathbf{v}$
    $\mathbf{U} := [\mathbf{U}, \mathbf{u}]$, $\mathbf{V} := [\mathbf{V}, \mathbf{v}]$
    $\omega := \mathbf{v}^H \cdot \mathbf{r}$, $\mathbf{x} := \mathbf{x} + \omega \cdot \mathbf{u}$, $\mathbf{r} := \mathbf{r} - \omega \cdot \mathbf{v}$
  end while
return $\mathbf{x}, \mathbf{r}, \mathbf{U}, \mathbf{V}$
end procedure

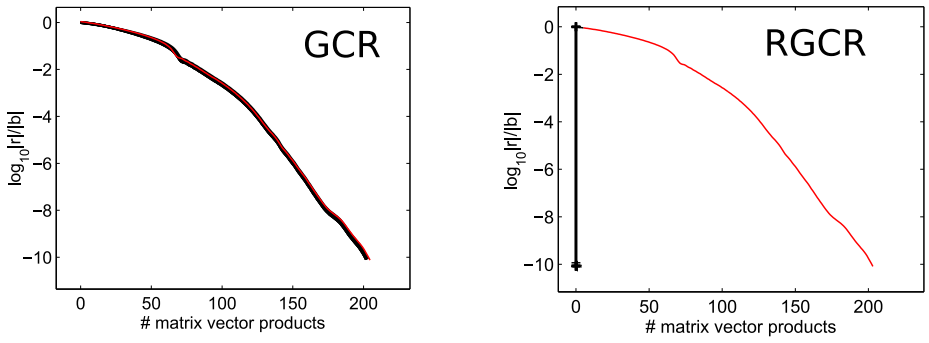

Abbildung 1: Vergleich von GCR und RGCR am Fourier-Testproblem.

Beispiel 1: Fourier-Testproblem

Wir lösen

$$\left\{ \begin{array}{rl} \partial_t u - \Delta u &= 1 \quad \text{in } \Omega_T := (0,1)^2 \times (0,1] \\ u &= 0 \quad \text{on } \partial\Omega_T \setminus \Omega_T \end{array} \right\}$$

mit finiten Differenzen ($\Delta x = 1/101$) und implizitem Euler-Verfahren ($\Delta t = 0.1$). Wir erhalten eine Sequenz der Form aus (1) mit $\mathbf{A} \in \mathbb{R}^{10000 \times 10000}$ aus dem (symmetrischen) Fünf-Punkt-Stencil (-1020.1, -1020.1, 4081.1, -1020.1, -1020.1) und den rechten Seiten $\mathbf{b}^{(1)} = \mathbf{1}$, $\mathbf{b}^{(\iota+1)} = \mathbf{A}^{-1} \cdot \mathbf{b}^{(\iota)} + 0.1 \cdot \mathbf{1}$ für $\iota = 2, ..., M = 10$.

Abb. 1 trägt die Konvergenzkurven für alle rechten Seiten von GCR und RGCR auf. Die rote Kurve zeigt dabei jeweils den Konvergenzverlauf für $\mathbf{b}^{(1)}$. Man sieht, dass sich die Kurven für GCR nicht großartig unterscheiden. Alle rechten Seiten scheinen ähnlich schwer lösbar. Man sieht außerdem, dass sich die rote Kurve von RGCR nicht zu der von GCR unterscheidet. Das ist logisch, denn zu Beginn liegen noch keine wiederverwertbaren Informationen vor, die RGCR nutzen könnte.

Für die nachfolgenden rechten Seiten jedoch konvergiert RGCR bereits nach wenigen (zwischen 1 und 3) Iterationen. Das liegt darin, dass mithilfe der recycelten Basismatrizen bereits sehr gute Näherungslösungen gefunden werden können.

Dieser Sachverhalt ist kein Zufall: Aus [A1.2, Kap. 5.2] ist bekannt, dass für eine Initialschätzung $\mathbf{x}_0 = \mathbf{0}$ für symmetrisch positiv definite Matrizen im Krylov-Unterraum $\mathcal{K}_m(\mathbf{A}; \mathbf{b}^{(\iota)})$ eine Näherungslösung existiert, für deren Residuum $\mathbf{r}^{(\iota)}$ die Schätzung

$$\|\mathbf{r}^{(\iota)}\|_2 \leq 2 \cdot \left(\frac{\sqrt{\kappa} - 1}{\sqrt{\kappa} + 1} \right)^m \cdot \|\mathbf{b}^{(\iota)}\|_2$$

gilt, wobei $\kappa = cond_2(\mathbf{A})$. Der Krylov-Unterraum $\mathcal{K}_m(\mathbf{A}; \mathbf{b}^{(\iota)})$ für $\iota > 1$ ist jedoch schon nach einer Iteration von RGCR in dessen Ansatzraum $\mathcal{U} \equiv \mathcal{K}_m(\mathbf{A}; \mathbf{b}^{(\iota-1)}) + span\{\mathbf{b}^{(\iota)}\}$ enthalten.

3.2 Das Speicherproblem

Im Folgenden wollen wir betrachten, wie sich das RGCR-Verfahren zur Lösung von Sequenzen Hermitescher linearer Gleichungssysteme in der Praxis anwenden lässt.

Ebenso wie bei GCR handelt es sich bei RGCR um ein Langterm-Rekursionsverfahren. Wie wir in Abschnitt 2.4 schon gesehen haben, können wir wegen $\mathbf{A} = \mathbf{A}^H$ den GCR-Part durch das Kurzterm-Rekursionsverfahren CR ersetzen. Infolge dessen erhalten wir pro Iteration jedoch nur einen konstanten Rechenaufwand. Die Speicherkomplexität zum Abspeichern der Matrizen

$\mathbf{U}, \mathbf{V}$ liegt jedoch weiterhin in $\mathcal{O}(N \times m)$ und steigt linear mit der Anzahl m der Iteration an.

In der Praxis werden zur Lösung eines Gleichungssystems oftmals $m = \mathcal{O}(100)$ Iterationen benötigt. Dabei wäre es für den praxisrelevanten Fall, d.h. wenn N sehr groß ist, nicht möglich, die Matrizen $\mathbf{U}, \mathbf{V} \in \mathbb{C}^{N \times m}$ vollständig im Speicher zu halten.

Wenn wir also ein System mit PCR lösen, dann können wir zwar jede Spalte von $\mathbf{U}, \mathbf{V}$ zu irgendeinem Zeitpunkt lesen (vgl. Algo. 2, Zeile 10-12); jedoch können wir nur einige, wenige Spalten speichern. Die Strategie jedes Recycling-Verfahrens besteht deshalb darin, mithilfe der gelesenen Spalten $\mathbf{u}_i, \mathbf{v}_i$, $i = 1, ..., m+1$ kleinere Basismatrizen $\tilde{\mathbf{U}}, \tilde{\mathbf{V}} \in \mathbb{C}^{N \times k}$ mit $k < m$ aufzustellen, die sich abspeichern und für nachfolgende Systeme wiederverwenden lassen.

Beim Aufstellen von $\tilde{\mathbf{U}}, \tilde{\mathbf{V}}$ lassen sich zwei Ansätze unterscheiden:

1. Man schätzt mithilfe der Spalten $\mathbf{u}_i, \mathbf{v}_i$ Eigenvektoren von Eigenwerten am Rande des Spektrums von $\mathbf{A}$. Man wählt $\tilde{\mathbf{V}}$ zu einem Projektor, der Richtungsanteile des Residuums in diesen Eigenvektoren eliminiert. Dadurch verbessert sich die Konvergenzrate.

2. Man wählt $\tilde{\mathbf{U}}, \tilde{\mathbf{V}}$ mithilfe einer Auswahl der Spalten $\mathbf{u}_i, \mathbf{v}_i$, sodass sich Matrix-Vektor-Produkte mit $\mathbf{U}$ und $\mathbf{V}^H$ kostengünstig berechnen lassen. Dadurch bleibt es möglich, Lösungen nachfolgender Systeme nach Formel (2) zu berechnen.

Im nachfolgenden Unterabschnitt wird das Verfahren R-MINRES vorgestellt. Dieses Verfahren stellt einen Vertreter des ersten Ansatzes dar. Im vierten Kapitel lernen wir mit SR-PCR-ap ein Verfahren kennen, dass den zweiten Ansatz realisiert.

3.3 Deflation: Das R-MINRES-Verfahren

R-MINRES [B1.4] ist eine MINRES-Variante mit Deflation durch Ritz-Vektoren. Im Folgenden wird erklärt, worum es sich dabei handelt. Zunächst wird das Verfahren beschrieben, anschließend folgt eine Diskussion dessen Nutzens.

Im Sinne einer vereinfachten Darstellung präsentieren wir eine algebraisch äquivalente Variante von R-MINRES, die anstelle von MINRES aus CR abgeleitet wurde. Wir nennen dieses Verfahren im Folgenden R-CR. Wir leiten dieses Verfahren aus CR her.

Wir nehmen vereinfachend $\mathbf{M} = \mathbf{I}$ an. Bezüglich der Vektoren $\mathbf{v}$, die iterativ in jedem Schleifendurchlauf von CR berechnet werden, können wir die Matrix $\mathbf{A}$ durch eine Matrix $(\mathbf{I} - \tilde{\mathbf{V}} \cdot \tilde{\mathbf{V}}^H) \cdot \mathbf{A} \cdot (\mathbf{I} - \tilde{\mathbf{V}} \cdot \tilde{\mathbf{V}}^H)$ ersetzen, wobei $\tilde{\mathbf{V}}$ eine beliebige orthogonale Matrix ist. Das begründen wir im Folgenden, indem wir Algo. 4 betrachten: Nach Zeile 6 wird $\mathbf{v}$ nicht nur auf $\tilde{\mathbf{v}}$ sondern auch auf $\tilde{\mathbf{V}}$ senkrecht gestellt. Daraus folgt dasselbe Ergebnis, als hätten wir $\mathbf{v}$ als Bild zur Matrix $(\mathbf{I} - \tilde{\mathbf{V}} \cdot \tilde{\mathbf{V}}^H) \cdot \mathbf{A}$ berechnet. Da $\mathbf{u} = \mathbf{r}$ in Zeile 6 senkrecht auf $\tilde{\mathbf{V}}$ ist, gilt $(\mathbf{I} - \tilde{\mathbf{V}} \cdot \tilde{\mathbf{V}}^H) \cdot \mathbf{A} \cdot \mathbf{u} \equiv (\mathbf{I} - \tilde{\mathbf{V}} \cdot \tilde{\mathbf{V}}^H) \cdot \mathbf{A} \cdot (\mathbf{I} - \tilde{\mathbf{V}} \cdot \tilde{\mathbf{V}}^H) \cdot \mathbf{u}$.

Die Matrix $\tilde{\mathbf{V}}$ führt dazu, dass das unterlagerte CR-Verfahren den Krylov-Unterraum der Matrix $(\mathbf{I} - \tilde{\mathbf{V}} \cdot \tilde{\mathbf{V}}^H) \cdot \mathbf{A} \cdot (\mathbf{I} - \tilde{\mathbf{V}} \cdot \tilde{\mathbf{V}}^H)$ aufbaut. Je nach Wahl von $\tilde{\mathbf{V}}$ ist die verallgemeinerte Kondition dieser Matrix deutlich besser als die von $\mathbf{A}$. Infolge dessen konvergiert das Verfahren schneller. Damit erklärt sich der Nutzen dieses Ansatzes.

Algorithm 4 R-CR-Verfahren

1: **procedure** R-CR($\mathbf{A}, \mathbf{r}_0, \mathbf{x}_0; \tilde{\mathbf{U}}, \tilde{\mathbf{V}}$)
2: $\hat{\mathbf{U}} := \tilde{\mathbf{U}}$, $\hat{\mathbf{V}} := \tilde{\mathbf{V}} \in \mathbb{C}^{N \times k}$
3: $\mathbf{x} := \mathbf{x}_0 + \tilde{\mathbf{U}} \cdot \tilde{\mathbf{V}}^T \cdot \mathbf{r}_0, \quad \mathbf{r} := \mathbf{r}_0 - \tilde{\mathbf{V}} \cdot \tilde{\mathbf{V}}^T \cdot \mathbf{r}_0$
4: $\tilde{\mathbf{u}} := \mathbf{0}$, $\tilde{\mathbf{v}} := \mathbf{0} \ \in \mathbb{C}^N$
5: **while** $\|\mathbf{r}\| >$ tol **do**
6: $\mathbf{u} := \mathbf{r}, \ \mathbf{v} := \mathbf{A} \cdot \mathbf{u}$
7: $\boldsymbol{\gamma} := [\tilde{\mathbf{V}}, \tilde{\mathbf{v}}]^T \cdot \mathbf{v}$
8: $\mathbf{u} := \mathbf{u} - [\tilde{\mathbf{U}}, \tilde{\mathbf{u}}] \cdot \boldsymbol{\gamma}, \ \mathbf{v} := \mathbf{v} - [\tilde{\mathbf{V}}, \tilde{\mathbf{v}}] \cdot \boldsymbol{\gamma}$
9: $\mathbf{u} := \mathbf{u}/\|\mathbf{v}\|, \ \mathbf{v} := \mathbf{v}/\|\mathbf{v}\|$
10: $\hat{\mathbf{U}} := [\hat{\mathbf{U}}, \mathbf{u}]$, $\hat{\mathbf{V}} := [\hat{\mathbf{V}}, \mathbf{v}]$, $\tilde{\mathbf{u}} := \mathbf{u}$, $\tilde{\mathbf{v}} := \mathbf{v}$
11: **if** `size`$(\hat{\mathbf{U}}, 2) \geq m$ **then**
12: $[\hat{\mathbf{U}}, \hat{\mathbf{V}}] :=$`Reduce`$(\hat{\mathbf{U}}, \hat{\mathbf{V}})$
13: **end if**
14: $\omega := \mathbf{v}^T \cdot \mathbf{r}$
15: $\mathbf{r} := \mathbf{r} - \omega \cdot \mathbf{v}, \ \mathbf{x} := \mathbf{x} + \omega \cdot \mathbf{u}$
16: **end while**
17: **return** $\mathbf{x}, \mathbf{r}, \hat{\mathbf{U}}, \hat{\mathbf{V}}$
18: **end procedure**

Nachdem wir besprochen haben, wie R-CR von einer geeigneten Matrix $\tilde{\mathbf{V}}$ profitieren kann, besprechen wir nun, wie geeignete Matrizen $\tilde{\mathbf{U}}, \tilde{\mathbf{V}}$ aufgestellt werden können.

In Algo. 4 werden zwei Matrizen $\hat{\mathbf{U}}, \hat{\mathbf{V}}$ aufgebaut und zum Abschluss der Prozedur ausgegeben. Diese Matrizen stellen die Eingabeargumente $\tilde{\mathbf{U}}, \tilde{\mathbf{V}}$ für den nachfolgenden Lösungsprozess bereit. $\hat{\mathbf{U}}, \hat{\mathbf{V}}$ werden gebildet, indem sukzessive die neuen Vektoren $\mathbf{u}, \mathbf{v}$ von rechts als neue Spalten angefügt werden. Wenn die Anzahl der Spalten von $\hat{\mathbf{U}}$ zu groß wird (Zeile 11), dann wird die Anzahl der Spalten von $\hat{\mathbf{U}}, \hat{\mathbf{V}}$ verkleinert.

Die dabei verwendete `Reduce`-Funktion versucht, die Matrizen $\hat{\mathbf{U}}, \hat{\mathbf{V}}$ so zu reduzieren, dass $\hat{\mathbf{U}}$ die Basis eines invarianten Unterraums approximiert. Dazu arbeitet es wie folgt: Aus $\hat{\mathbf{U}}$ lassen sich Eigenwerte und -vektoren von $\mathbf{A}$ schätzen. $\hat{\mathbf{U}}$ wird reduziert zu einer Basis der geschätzten Eigenvektoren zu den Eigenwerten mit kleinstem Betrag. $\hat{\mathbf{V}}$ wird nachgehalten, sodass stets $\hat{\mathbf{V}} = \mathbf{A} \cdot \hat{\mathbf{U}}$ gilt.

Aus $\hat{\mathbf{U}}$ lassen sich die Eigenwerte und -vektoren wie folgt schätzen: Es ist wohl-bekannt, dass die Eigenwerte von $\mathbf{H}_m$ der Arnoldi-Gleichung (4) extremale

Eigenwerte von $\mathbf{A}$ approximieren. Analog folgt aus der Gleichung

$$\mathbf{A} \cdot \mathbf{U} = \mathbf{V}\,,$$

dass die Eigenwerte von $\mathbf{G} := \mathbf{U}^H \cdot \mathbf{V}$ ebenfalls extremale Eigenwerte von $\mathbf{A}$ approximieren. Liegt die Schurzerlegung $\mathbf{G} = \mathbf{Q}^H \cdot \mathbf{R} \cdot \mathbf{Q}$ vor mit $\mathbf{R} = (r_{i,j})_{i,j}$, $r_{i,i} = \lambda_i$, $|\lambda_i| \leq |\lambda_{i+1}|$ $\forall i$, dann werden $\hat{\mathbf{U}} := \hat{\mathbf{U}} \cdot \mathbf{Q}(:, 1:k)$, $\hat{\mathbf{V}} := \hat{\mathbf{V}} \cdot \mathbf{Q}(:, 1:k)$ ersetzt, wobei $k < m$ vor Aufruf der Methode durch den Benutzer zu wählen ist. Man bemerke, dass die Orthogonalität von $\hat{\mathbf{V}}$ erhalten bleibt. Falls $\mathbf{A} \in \mathbb{R}^{N \times N}$, $\mathbf{b} \in \mathbb{R}^N$ und $\mathbf{R}$ anstelle der komplexen Eigenwerte nur diekonjugierten Paare bestimmt, so bleibt $\hat{\mathbf{V}}$ reell und das Verfahren vermeidet Rechnungen mit komplexen Zahlen.

In Abschnitt 5.1 zeigen wir anhand eines Beispiels den Effekt der Deflation auf die Konvergenzrate einer Matrix mit separierten Eigenwerten. Deflation bedeutet in der numerischen Mathematik u.A. die Elimination spektraler Anteile linearer Operatoren. Dies ist die hier zutreffende Bedeutung, denn die Eigenwerte, deren Vektoren in range($\mathbf{V}$) liegen, werden im projezierten Operator $(\mathbf{I} - \mathbf{V} \cdot \mathbf{V}^H) \cdot \mathbf{A} \cdot (\mathbf{I} - \mathbf{V} \cdot \mathbf{V}^H)$ zu Null. *Ritz-Vektor* ist die hier verwendete Approximation von Eigenvektoren über eine Projektion des Bildraums von $\hat{\mathbf{U}}$.

Es bleibt anzumerken, dass der Deflationsansatz die Konvergenzrate nur kaum verbessert, falls durch die Elimination einzelner, leicht berechenbarer Eigenwerte das Spektrum von $\mathbf{A}$ nicht stark verbessert werden kann. Der Nutzen von Deflation ist daher stark abhängig vom jeweils zu lösenden System.

4 Das Verfahren SR-PCR-ap

Im folgenden Abschnitt lassen wir zwecks vereinfachter Darstellung die Präkonditonierung außer Betracht, bzw. nehmen $\mathbf{M} = \mathbf{I}$ an. Zum Ablauf des Verfahrens mit Präkonditionierung siehe [B3.2].

4.1 Motivation des Verfahrens

Um aus RGCR ein praktisch anwendbares Verfahren zu gewinnen, wurde in R-CR ein Reduktionsansatz (Algo. 4, Zeile. 12) verwendet. Diese Reduktion führt zu Datenverlust. Wir wollen nun den Datenverlust in vereinfachter Form darstellen. Dazu betrachten wir die Situation, dass nacheinander zwei Systeme

$$\mathbf{A} \cdot \mathbf{x}^{(1)} = \mathbf{b}^{(1)}\,,$$
$$\mathbf{A} \cdot \mathbf{x}^{(2)} = \mathbf{b}^{(2)}$$

gelöst werden. Verwendet man zur Lösung dieser Sequenz das Verfahren R-CR, so werden - vereinfacht dargestellt - folgende Schritte durchgeführt:

1. Mit CR wird in m Iterationen für $\mathbf{b}^{(1)}$ gelöst.

1.1. CR stellt die Basismatrizen $\mathbf{U}, \mathbf{V} = \mathbf{A} \cdot \mathbf{U} \in \mathbb{C}^{N \times m}$ mit $\mathbf{V}^H \cdot \mathbf{V} = \mathbf{I}$ (virtuell) auf.

1.2. CR stellt $\mathbf{r}^{(1)}$ auf $\mathbf{V}$ senkrecht.

2. Die Basismatrizen $\mathbf{U}, \mathbf{V}$ werden zu kleineren Basismatrizen $\tilde{\mathbf{U}}, \tilde{\mathbf{V}}$ reduziert. (eigentlich geschieht dies während Schritt 1)

3. Mit R-CR wird für $\mathbf{b}^{(2)}$ gelöst.

   3.1. R-CR erweitert die Basismatrizen $\tilde{\mathbf{U}}, \tilde{\mathbf{V}}$.

   3.2. R-CR stellt $\mathbf{r}^{(2)}$ auf der erweiterten Basismatrix von $\tilde{\mathbf{V}}$ senkrecht.

Ein Verfahren, dass für $\tilde{\mathbf{U}}, \tilde{\mathbf{V}}$ die vollen Basismatrizen $\mathbf{U}, \mathbf{V}$ wiederverwenden *könnte* (unter geringem Speicherbedarf), wäre dem Verfahren R-CR grundsätzlich überlegen, da keine Informationen von $\mathbf{U}, \mathbf{V}$ nach $\tilde{\mathbf{U}}, \tilde{\mathbf{V}}$ verworfen werden müssten. Im Folgenden stellen wir ein Verfahren vor, das dies kann.

4.2 Herleitung von SR-PCR-ap

Das Verfahren SR-PCR-ap besteht aus zwei Schritten:

1. Projektionslösung: Zuerst wird für die neue rechte Seite eine Residuum-optimale Näherungslösung berechnet. Dies gelingt über die Formel

$$\mathbf{x} = \mathbf{x}_0 + \mathbf{U} \cdot \mathbf{V}^H \cdot \mathbf{r}_0^{(\iota)}, \tag{9}$$

   wobei $\mathbf{U}, \mathbf{V}$ die alten, wiederverwendeten Basismatrizen sind.

2. Nachiterationen: Anschließend soll analog wie bei R-CR das Residuum weiter iterativ durch eine CR-Iteration gemindert werden, bis die Toleranzschwelle tol erreicht wurde.

Um ein solches Verfahren zu realisieren, müssen zwei Probleme bewältigt werden: 1.) muss $\mathbf{x}$ nach Gleichung (9) berechnet werden, obwohl $\mathbf{U}, \mathbf{V}$ nicht vollständig im Speicher vorliegen können. Dies gelingt über sog. *kurze Repräsentationen* und wird in Abschnitt 4.2.1 beschrieben. 2.) muss für die Nachiterationen eine Möglichkeit gefunden werden, weitere Abstiegsrichtungen $\mathbf{v}$ günstig auf der recycelten, nicht im Speicher vorliegenden Matrix $\mathbf{V}$ senkrecht zu stellen. Dieses Problem wird in Abschnitt 4.2.2 behandelt.

4.2.1 Kurze Repräsentationen

Wir definieren die Block-Krylov-Matrix.

Definition 1: Block-Krylov-Matrix (BKM)

Seien $\mathbf{A} \in \mathbb{C}^{N \times N}$, $\tilde{\mathbf{U}} \in \mathbb{C}^{N \times k}$, und $m = J \cdot k$, $m, k, J \in \mathbb{N}$, dann heißt

$$K_J(\mathbf{A}; \tilde{\mathbf{U}}) := [\tilde{\mathbf{U}}, \mathbf{A} \cdot \tilde{\mathbf{U}}, ..., \mathbf{A}^{J-1} \cdot \tilde{\mathbf{U}}] \in \mathbb{C}^{N \times m}$$

Block-Krylov-Matrix von $\mathbf{A}, \tilde{\mathbf{U}}$ der Stufe J.

Die BKM hat folgende Eigenschaften.

Matrix-Vektor-Produkte mit der BKM Liegt lediglich die Matrix $\tilde{\mathbf{U}}$ im Speicher vor, so lassen sich dennoch günstig über folgendes Horner-Schema Produkte mit $K_J(\mathbf{A}; \tilde{\mathbf{U}})$ evaluieren. Soll $\mathbf{z} := K_J(\mathbf{A}; \tilde{\mathbf{U}}) \cdot \mathbf{y}$ berechnet werden, wobei $\mathbf{y} = (\mathbf{y}_1^T, \mathbf{y}_2^T, ..., \mathbf{y}_J^T) \in \mathbb{C}^m$, $\mathbf{y}_j \in \mathbb{C}^k$, so lässt sich rechnen:

1: $\mathbf{z} := \tilde{\mathbf{U}} \cdot \mathbf{y}_J$
2: **for** $j = J-1,\ J-2,\ ...,\ 1$ **do**
3: $\quad \mathbf{z} := \mathbf{A} \cdot \mathbf{z}$
4: $\quad \mathbf{z} := \mathbf{z} + \tilde{\mathbf{U}} \cdot \mathbf{y}_J$
5: **end for**
6: **return z**

Analog lassen sich Produkte mit $K_J(\mathbf{A}; \tilde{\mathbf{U}})^H$ evaluieren. Dies gelingt über das folgende Potenz-Schema. Soll $\mathbf{y} := K_J(\mathbf{A}; \tilde{\mathbf{U}})^H \cdot \mathbf{z}$ berechnet werden, wobei $\mathbf{y} = (\mathbf{y}_1^T, \mathbf{y}_2^T, ..., \mathbf{y}_J^T) \in \mathbb{C}^m$, $\mathbf{y}_j \in \mathbb{C}^k$, so lässt sich rechnen:

1: **for** $j = 1,\ 2,\ ...,\ J-1$ **do**
2: $\quad \mathbf{y}_J := \tilde{\mathbf{U}}^H \cdot \mathbf{z}$
3: $\quad \mathbf{z} := \mathbf{A} \cdot \mathbf{z}$
4: **end for**
5: $\mathbf{y}_J := \tilde{\mathbf{U}}^H \cdot \mathbf{z}$
6: **return y**

In beiden Fällen müssen nur $J-1$ Matrix-Vektor-Produkte mit $\mathbf{A}$ und J Matrix-Vektor-Produkte mit $\tilde{\mathbf{U}}$ bzw. ihrer Transponierten berechnet werden.

Transformation Abschließend können wir zeigen, dass eine BKM existiert, die ähnlich zu $\mathbf{U}$ ist. Dies heißt, dass die Berechnung von Matrix-Vektor-Produkten mit $\mathbf{U}$ und $\mathbf{V}^H = \mathbf{U}^H \cdot \mathbf{A}$ über Matrix-Vektor-Produkte mit der BKM ersetzt werden kann. Die dazu benötigte Transformation liefert das folgende Lemma.

Lemma 1: Kurze Repräsentation

Seien $\mathbf{A}, \mathbf{U}$ wie in (2) und erfülle das Bild $\mathbf{V} = \mathbf{A} \cdot \mathbf{U} \in \mathbb{C}^{N \times m}$ die Lanczos-Gleichung (6) mit regulärer Matrix $\mathbf{T} \in \mathbb{C}^{m \times m}$. Man wähle $\tilde{\mathbf{U}} = [\mathbf{u}_1, \mathbf{u}_{1+J}, \mathbf{u}_{1+2\cdot J}, ...] \in \mathbb{C}^{N \times k}$. Dann gilt mithilfe der nachfolgend definierten Dreiecksmatrix $\mathbf{R}$ und der Permutationsmatrix $\mathbf{\Pi}$, jeweils $\in \mathbb{C}^{m \times m}$:

$$\mathbf{U} = K_J(\mathbf{A}; \tilde{\mathbf{U}}) \cdot \mathbf{\Pi} \cdot \mathbf{R}^{-1}\,.$$

Beweis: Siehe [B3.2, S. 6] und [B3.3, S. 18]. Man stelle die Matrizen $\mathbf{\Pi}, \mathbf{R}$ wie folgt spaltenweise in Speicher- und Rechenaufwand $\mathcal{O}(m \cdot J)$ auf:

1: $\mathbf{R} := []$
2: **for** $i = 1, ..., k$ **do**
3: $\quad \mathbf{p} := \mathbf{e}_{1+(i-1)\cdot J} \in \mathbb{C}^m$

4: **for** $j = 1, ..., J$ **do**
5: $\mathbf{R} := [\mathbf{R}, \mathbf{p}]$
6: $\mathbf{p} := \mathbf{T} \cdot \mathbf{p}$ *// $\boldsymbol{p}$ hat maximal J Einträge; $\boldsymbol{T}$ tridiagonal*
7: **end for**
8: **end for**

Die Permutationsmatrix $\mathbf{\Pi}$ wird spaltenweise aufgestellt über

$$\mathbf{\Pi} \cdot \mathbf{e}_{1+j \cdot J+j} = \mathbf{e}_{1+j \cdot k+i}\,, \quad j = 0, ..., J-1, \quad i = 0, ..., k-1\,.$$

Einsetzen liefert die Behauptung. q.e.d.

Der Nutzen dieses Lemmas ist wie folgt: Wenn die Matrizen $\tilde{\mathbf{U}}, \mathbf{T}$ abgespeichert wurden, dann kann Ausdruck (9) für beliebige rechte Seiten $\mathbf{b} \in \mathbb{C}^N$ nun über folgende Schritte ausgerechnet werden:

1: $\mathbf{z} := \mathbf{A} \cdot (\mathbf{b} - \mathbf{A} \cdot \mathbf{x}_0)$
2: Berechne $\mathbf{y} := K_J(\mathbf{A}; \tilde{\mathbf{U}})^H \cdot \mathbf{z}$ über das Potenz-Schema.
3: Transformiere $\mathbf{y} := \mathbf{\Pi} \cdot \mathbf{R}^{-1} \cdot \mathbf{R}^{-H} \cdot \mathbf{\Pi}^H \cdot \mathbf{y}$
4: Berechne $\mathbf{z} := K_J(\mathbf{A}; \tilde{\mathbf{U}}) \cdot \mathbf{y}$ über das Horner-Schema.
5: **return** $\mathbf{x} := \mathbf{x}_0 + \mathbf{z}$

4.2.2 Kurze Orthogonalisierung auf V

Nachdem wir die Bestapproximation $\mathbf{x}$ bzgl. der alten Basismatrizen $\mathbf{U}, \mathbf{V}$ berechnet haben, müssen nun für nachfolgende CR-Iterationen die neuen Abstiegsrichtungen $\mathbf{v}$ über kurze Rekursionen auf $\mathbf{V}$ senkrecht gestellt werden. Da die Spalten von $\mathbf{V}$ der Lanczos-Rekursion genügen, lässt sich zeigen, dass jede Abstiegsrichtung $\mathbf{v}$ in CR lediglich auf der letzten Spalte von $\mathbf{V}$ orthogonalisiert werden muss. Für eine detailiertere Darstellung siehe [B3.2, Lemma 2].

Bezeichnen wir die letzte Spalte von $\mathbf{V}$ mit $\mathbf{v}_m$, so erhalten wir zusammenfassend das Verfahren SR-PCR-ap wie in Algo. 5:

Durch eine Sequenz von mehreren kurzen Repräsentationen lassen sich sowohl Suchräume mehrerer Lösungsprozesse miteinander kombinieren als auch die Stufe J der Block-Krylov-Matrix reduzieren (zwecks Begrenzung von Rundungsfehlern). Dies wird hier nicht betrachtet. Für Details verweisen wir auf [B3.2, Abs. 2.3.1]. In den Experimenten bezeichnen wir die Anzahl der verwendeten kurzen Repräsentationen mit ℓ.

4.2.3 Kosten von SR-PCR-ap

Mit SR-PCR-ap lässt sich ein Krylov-Unterraum der Dimension m residuumoptimal recyceln. Dabei ist jedoch nur ein Speicheraufwand in $\mathcal{O}(N \cdot k + m \cdot J)$ zum Abspeichern von $\mathbf{R}, \tilde{\mathbf{U}}$ erforderlich. Der Rechenaufwand setzt sich zusammen aus jeweils J Matrix-Vektor-Produkten mit den zwei Matrizen $\tilde{\mathbf{U}}, \tilde{\mathbf{U}}^H$ sowie $2 \cdot J - 1$ Matrix-Vektor-Produkten mit $\mathbf{A}$.

Durch n Nachiterationen kann in Rechenaufwand von (im Wesentlichen) n Matrix-Vektor-Produkten mit $\mathbf{A}$ der Suchraum von m auf $m + n$ Dimensio-

Algorithm 5 SR-PCR-ap-Verfahren

procedure R-CR($\mathbf{A}, \mathbf{r}_0, \mathbf{x}_0; \tilde{\mathbf{U}}, \mathbf{u}_m, \mathbf{v}_m$)
  Berechne mittels $\tilde{\mathbf{U}}$ das resdiuum-optimale $\mathbf{x}$ mithilfe der kurzen Repräsentation
  Berechne das zugehörige Residuum $\mathbf{r} := \mathbf{b} - \mathbf{A} \cdot \mathbf{x} \perp \mathit{range}(\mathbf{V})$.
  $\tilde{\mathbf{u}} = 0$, $\tilde{\mathbf{v}} := 0$ *// Initialisiere CR-Postiterationen*
  while $\|\mathbf{r}\| > \text{tol}$ **do**
    $\mathbf{u} := \mathbf{r}$, $\mathbf{v} := \mathbf{A} \cdot \mathbf{u}$
    $\boldsymbol{\gamma} := [\mathbf{v}_m, \tilde{\mathbf{v}}]^H \cdot \mathbf{v}$
    $\mathbf{u} := \mathbf{u} - [\mathbf{u}_m, \tilde{\mathbf{u}}] \cdot \boldsymbol{\gamma}$, $\mathbf{v} := \mathbf{v} - [\mathbf{v}_m, \tilde{\mathbf{v}}] \cdot \boldsymbol{\gamma}$
    $\mathbf{u} := \mathbf{u}/\|\mathbf{v}\|$, $\mathbf{v} := \mathbf{v}/\|\mathbf{v}\|$
    $\omega := \mathbf{v}^T \cdot \mathbf{r}$
    $\mathbf{r} := \mathbf{r} - \omega \cdot \mathbf{v}$, $\mathbf{x} := \mathbf{x} + \omega \cdot \mathbf{u}$
  end while
  return $\mathbf{x}, \mathbf{r}$
end procedure

nen vergrößert und im zusammengesetzten $m + n$-dimensionalen Suchraum die Residuum-optimale Lösung berechnet werden.

Zusammengefasst:

- Rechenaufwand: $\approx 2 \cdot J + n$ Matrix-Vektor-Produkte mit $\mathbf{A}$
- Speicheraufwand: $\approx m/J \cdot N$ Zahlen.

Muss J sehr groß gewählt werden, so fachen sich Rundungsfehler an (Horner- und Potenz-Schema sind numerisch schlecht gestellt). In diesem Fall sollte die Basis-Matrix $\mathbf{U} \in \mathbb{C}^{N \times m}$ über eine Anzahl von ℓ kurzen Repräsentationen dargestellt werden, wobei jede Repräsentation eine BKM der Stufe J benutzt. Es muss dann gelten: $m = \ell \cdot k \cdot J$. Die Verfahrenskosten ändern sich dann wie folgt:

- Rechenaufwand: $\approx \ell \cdot 2 \cdot J + n$ Matrix-Vektor-Produkte mit $\mathbf{A}$
- Speicheraufwand: $\approx \ell \cdot m/J \cdot N$ Zahlen.

Es zeigt sich, dass unter geeingeter Präkonditionierung die Rundungsfehler verringert werden können, was eine größere Wahl von J erlaubt, siehe [B3.2, Abs. 3.1, Fig. 3.2-3.3].

5 Numerischer Vergleich von R-CR und SR-PCR-ap

Abschließend vergleichen wir die beiden vorgestellten Recycling-Verfahren anhand ihrer theoretischen Eigenschaften und ihrer Effizienz an Testproblemen.

5.1 Deflation von separierten Eigenwerten

Wir wollen zunächst an einem klein-dimensionierten Testproblem die prinzipiell unterschiedlichen Konvergenzeigenschaften von R-CR und SR-PCR-ap hervorheben.

Wir betrachten das reell symmetrische Testproblem $\mathbf{A} = \text{diag}(-20, -19, ..., -10, 50, 51, ..., 250) \in \mathbb{R}^{212\times 212}$, $\mathbf{b}^{(1)} = \mathbf{1}$.

MINRES / CR Wir lösen nach $\mathbf{b}^{(1)}$ mit MINRES bzw. CR (algebraisch äquivalent $\Rightarrow$ identischer Konvergenzverlauf), siehe Fig. 2. Wir können beobachten, dass das Verfahren bis ca. zur 55 Iterationen langsam konvergiert. Danach hat die dem MINRES-Prozess unterliegende Lanczos-Rekursion in $\mathbf{T}$ eine sehr gute Näherung der Eigenwerte $-20, ..., -10$ entwickelt, sodass der Operator $\mathbf{A}$ auf den restlichen Raumdimensionen, in denen das Residuum verbleibt, besser konditioniert ist. Infolge der besseren Kondition wird die Konvergenzrate beschleunigt.

R-CR Wir lösen mittels R-CR für $m = 15, k = 10$ zwei mal nach $\mathbf{b}^{(1)}$. Beim ersten Lösungsprozess liegen noch keine Recycling-Daten vor, daher erhalten wir die blaue Konvergenzkurve. Beim zweiten Lösungsprozess können jedoch die online gewonnenen Basismatrizen $\hat{\mathbf{U}}, \hat{\mathbf{V}} \in \mathbb{C}^{N\times k}$ wiederverwendet werden, um die Konvergenzgeschwindigkeit zu beschleunigen (grüne Kurve). Wir sehen, dass diese Kurve ca. die gleiche Konvergenzrate wie die blaue Kurve zum Ende des Berechnungsprozesses besitzt. Der Grund dafür ist, dass durch die Deflation der Konvergenz-hemmende Einfluss der Eigenwerte $-20, ..., -10$ aus dem iterativen Lösungsprozess verschwindet. Die Anzahl der Matrix-Vektor-Produkte (und somit auch ca. die Rechenzeit) halbieren sich.

SR-PCR-ap SR-PCR-ap kann über eine kurze Repräsentation der Stufe $J = 6$ mithilfe von $k = 10$ Spaltenvektoren eine Residuum-optimale Lösung im Krylov-Unterraum $\mathcal{K}_{60}(\mathbf{A}; \mathbf{b}^{(1)})$ innerhalb von $2 \cdot J = 12$ Matrix-Vektor-Produkten bestimmen (siehe zweiter Stern in der roten Konvergenzkurve). Wir sehen, dass das zugehörige Residuum eine relative Norm von ca. 10^{-9} hat, wohingegen die blaue Kurve für $\#\mathsf{MV} = 60$ ein relatives Residuum von $< 10^{-10}$ besitzt. (Infolge von Rundungseinflüssen verlieren wir eine Nachkommastelle an Genauigkeit). Mithilfe einer anschließenden Nachiteration erreicht SR-PCR-ap die Toleranzschwelle.

SR-PCR-ap versucht nicht, gezielt einzelne Konvergenz-hemmende Eigenwerte zu deflationieren. Stattdessen recycelt es einfach sämtliche Informationen, die je zuvor gesammelt wurden; dies inkludiert per se die Deflation aller bisher approximierten Eigenwerte aus dem Spektrum von $\mathbf{A}$.

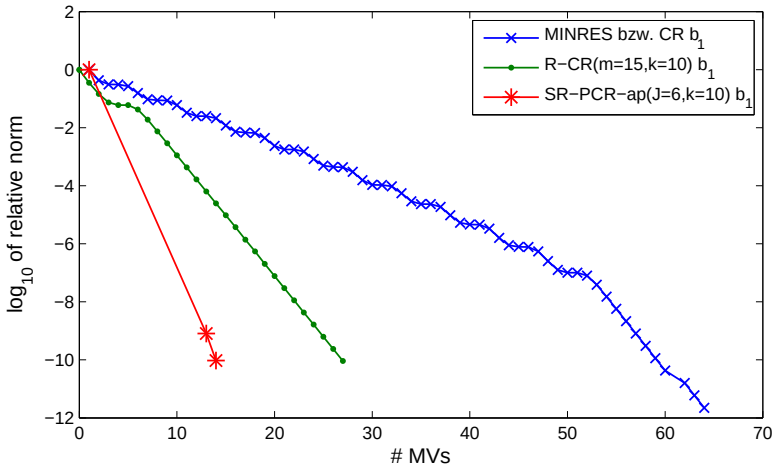

Abbildung 2: Konvergenzverlauf der Residuum-Norm für das Testproblem $\mathbf{A} = \text{diag}(-20, -19, ..., -10, 50, 51, ..., 250)$, $\mathbf{b}^{(1)} = \mathbf{1}$.

5.2 Konvergenzverhalten am Fourier-Testproblem

In [B3.2] wird SR-PCR-ap an einer Reihe von Testproblemen aus PDE-Diskretisierungen gegen R-MINRES verglichen. Dabei zeigt sich, dass im Durchschnitt der Rechenaufwand halbiert werden kann, während nur ein Viertel der Spalten der recycelten Basismatrix $\mathbf{U}$ gespeichert werden muss. Zur Demonstration der Robustheit von SR-PCR-ap sind die in [B3.2] betrachteten Systeme sind durchgehend schlecht konditioniert und es werden Präkonditionierer verwendet.

Da wir in diesem Aufsatz das Thema der Präkonditionierung jedoch nur am Rande behandelt haben, möchten wir hier stattdessen die Verfahren R-CR und SR-PCR-ap an einem gut-konditionierten Problem ohne Präkonditionierung vergleichen. Dazu betrachten wir das Fourier-Testproblem aus Beispiel 1.

Fig. 3 stellt die Konvergenzverläufe von R-CR und SR-PCR-ap mit verschiedenen Farben für die jeweiligen rechten Seiten gegenüber. Zur Lösung der ersten rechten Seite (blauer Graph) erzielen beide Verfahren infolge noch nicht vorhandener Recycling-Daten dieselbe Konvergenzkurve.

Im linken Diagramm ist erkennbar, dass R-CR bei der Lösung der nachfolgenden rechten Seiten die steilste Abstiegsrate der blauen Konvergenzkurve durchgehend aufrecht erhalten kann. Für R-CR wurden die Parameter $k = 20, m = 30$ gewählt.

SR-PCR-ap recycelt mittels über $\ell = 2$ Blöcke zu je $k = 10$ gespeicherten Spalten und kurzen Repräsentationen von jeweils der Stufe $J = 10$ den Suchraum $\mathcal{K}_{200}(\mathbf{A}; \mathbf{b}^{(1)})$. Zur Aufstellung der Residuum-optimalen Lösung werden pro rechte Seite $\ell \cdot 2 \cdot J = 40$ Matrix-Vektor-Produkte berechnet (dritte Markierung auf den Konvergenzgraphen in der rechten Teilabbildung).

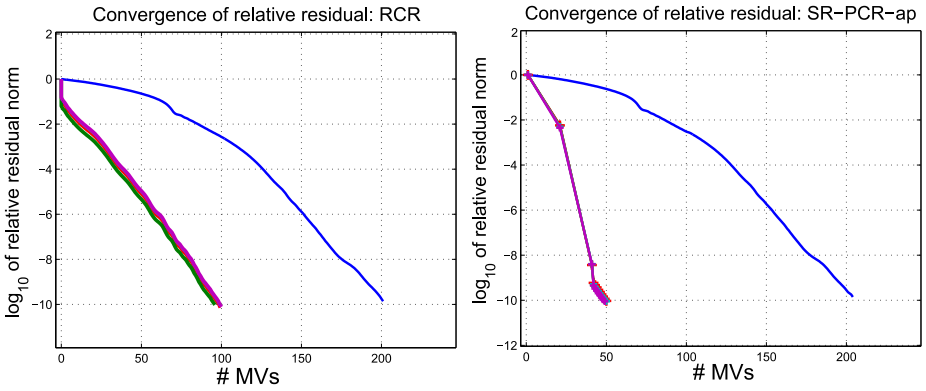

Abbildung 3: Konvergenzvergleich am Fourier-Testproblem für R-CR und SR-PCR-ap bei Speicherplatz für $k = 20$ Spaltenvektoren.

Im Vergleich benötigt SR-PCR-ap nur Speicherplatz für 20 Spaltenvektoren der Länge N, während R-CR 30 Spaltenvektoren speichern muss. Die Rechenkosten an AXPYs bzw. DOTs für SR-PCR-ap liegen pro Matrix-Vektor-Produkt bei maximal 10, bei R-CR hingegen zwischen 20 und 30.

Mithilfe der Abbildungen 1 und 3 können wir die Verfahren GCR, RGCR, R-CR und SR-PCR-ap bzgl. ihres Speed-Ups bei der Lösung nachfolgender rechter Seiten vergleichen:

GCR hat einen Seed-Up von 1, denn die Anzahl der Matrix-Vektor-Produkte nimmt von $\mathbf{b}^{(1)}$ zu den nachfolgenden rechten Seiten nicht sichtbar ab. RGCR hat einen Speed-Up von 1/200, da sich aus den Daten des ersten Lösungsprozesses alle nachfolgenden rechten Seiten mit jeweils maximal einem Matrix-Vektor-Produkt lösen lassen. SR-PCR-ap erzielt einen Speed-Up von 4 und R-CR halbiert immerhin die Iterationszahl (Speed-Up von 2).

6 Zusammenfassung

Wir haben die Grundidee von Krylov-Unterraum-Recycling vorgestellt und motiviert.

Bei der praktischen Anwendung von Recycling-Verfahren ist die Speicherintensität ein wichtiges Problem. Wir haben anhand zweier Verfahren gezeigt, wie sich das Speicherproblem durch gezieltes Abspeichern einzelner, weniger Spaltenvektoren beheben lässt.

Die Ansätze dieser präsentierten Verfahren sind grundverschieden: R-CR verwendet einen Deflationsansatz, um für nachfolgende rechte Seiten die Konvergenzrate zu verbessern. Dies ist mit einem Datenverlust von Basis-Informationen verbunden. SR-PCR-ap hingegen verwendet kurze Repräsentationen, um sämtliche zurückliegenden Basis-Informationenen verlustfrei für nachfolgende rechte Seiten wiederverwenden zu können.

Diese Arbeit legte den Schwerpunkt auf eine vereinfachte Darstellung der algorithmischen Strategien und Abläufe. Das Thema der Präkonditionierung wurde nur angeschnitten. Auch wurde nicht thematisiert, für welche Sequenzen von linearen Gleichungssystemen sich das Recycling überhaupt lohnt (siehe [B3.2, Beispiel 1]). Tatsächlich ist die Frage nach geeigneten Präkonditionierungsverfahren für *Sequenzen* von linearen Gleichungssystemen ein noch offenes Forschungsfeld.

7 Literatur

[A1.1] Y. Saad, *Iterative Methods for Sparse Linear Systems, 2nd edition*, SIAM, 2000.

[A1.2] S. Gross and A. Reusken, Numerical Methods for Two-phase Incompressible Flows. First edition 2011, Springer Series in Computational Mathematics, Vol. 40, ISBN 978-3-642-19685-0.

[A2.1] Y. Saad, M. H. Schultz, *GMRES: A generalized minimal residual algorithm for solving nonsymmetric linear systems*, SIAM J. Sci. Stat. Comput., Vol. 7(3), pp. 857-869, 1986.

[A2.2] S. C. Eisenstat and H. C. Elman and H. C. Schultz, *Variational iterative methods for non-symmetric systems of linear equations*. SIAM J. Numer. Anal. 20, pp. 345-357, 1983.

[B1.1] E. de Sturler, *Nested Krylov methods based on GCR*, Journal of Computational and Applied Mathematics, Vol. 67, pp. 15-41, 1996.

[B1.2] R. B. Morgan, *GMRES with Deflated Restarting*, SIAM J. Sci. Comput., 24(1), pp. 20-37, 2002.

[B1.3] E. de Sturler, *Truncation Strategies for optimal Krylov subspace methods*, SIAM J. Numer. Anal., Vol. 36(3), pp. 864-889, 1999.

[B1.4] S. Wang and E. de Sturler and G. H. Paulino, Large-scale topology optimization using preconditioned Krylov subspace methods with recycling, Int. J. for Num. Meth. in Engineering, Vol. 69(12), pp. 2441-2468, 2006.

[B2.1] P. Benner and L. Feng, *Recycling Krylov Subspaces for Solving Linear Systems with successively changing Right-Hand-Sides arising in Model Reduction*, Lecture Notes in Electrical Engineering, Vol. 74, pp. 125-140, Springer 2011.

[B2.2] M. Parks and E. de Sturler and G. Mackey and D.D. Johnson and S. Maiti, *Recycling Krylov subspaces for sequences of linear systems*, SIAM J. Sci. Comput. Vol. 28(5), pp. 1651-1674, 2006.

[B3.1] M. P. Neuenhofen, *Short-Recurrence and -Storage Recycling of large Krylov-Subspaces for Sequences of Linear Systems with changing Right-Hand-Sides*, Technical Report, arXiv: 1512.05101, 2015.

[B3.2] M. P. Neuenhofen and S. Groß, *Memory-efficient recycling of large Krylov-subspaces for sequences of Hermitian linear systems*, submitted manuscript, arXiv:1604.04052, 2016.

[B3.3] M. P. Neuenhofen, *Short-Recurrence and -Storage Recycling of large Krylov-Subspaces for Sequences of Linear Systems with changing Right-Hand-Sides*, Technical Report, available on arXiv: 1512.05101, 2015.

[D1] Z. Ye and Z. Zhu and J. R. Phillips, *Generalized Krylov Recycling Methods for Solution of Multiple Related Linear Equation Systems in Electromagnetic Analysis*, Design Automation Conference 2008, p. 682-687.

[D2] K. Mohamed and S. Nadarajah and M. Paraschivoiu, Krylov Recycling Techniques for Unsteady Simulation of Turbulent Aerodynamic Flows, 26th International Congress of the Aeronautical Sciences, 2008.

[D3] M. Kilmer and E. de Sturler, *Recycling Subspace Information for Diffuse Optical Tomography*, SIAM J. Sci. Comput., Vol. 27(6), pp. 2140-2166, 2006.

[D4] J. Bolten and N. Bozovic and A. Frommer, *Preconditioning of Krylov-subspace methods using recycling in Lattice QCD computations*, Proc. Appl. Math. Mech., Vol. 13, pp. 413-414, 2013.

[D5] K. Ahuja and E. de Sturler and S. Gugercin and E. R. Chang, *Recycling BiCG with an Application to Model Reduction*, SIAM J. Sci. Comput. Vol. 34, No. 4, pp. A1925-A1949, 2012.

[D6] K. Ahuja and E. de Sturler and P. Benner, *Recycling BiCGSTAB with an Application to Parametric Model Order Reduction*, MPI Magdeburg preprints, pp. 13-21, 2013.