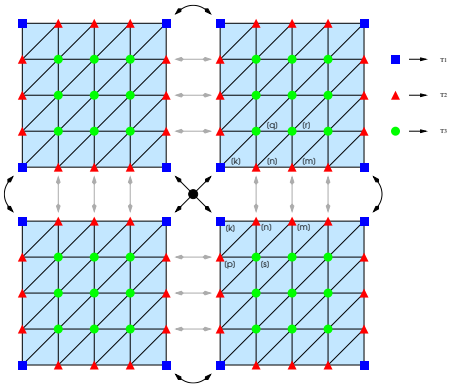
ulm university universität ulm

Final Projects

High Performance Computing II, Summer Term 2020



Prof. Dr. Stefan Funken
Prof. Dr. Karsten Urban
M.Sc. Lewin Ernst

Institute for Numerical Mathematics

Das vorliegende Manuskript beinhaltet die Ausarbeitungen der Projekte im Rahmen der Vorlesung High Performance Computing 2, welches im Sommersemester 2020 an der Fakultät für Mathematik und Wirtschaftswissenschaften der Universität Ulm stattgefunden hat.

Bedanken möchten wir uns an dieser Stelle nochmals bei allen Teilnehmern für ihre tollen Beiträge und gelungenen Vorträge, welche alle zusammen wesentlich zum Gelingen der Veranstaltung beigetragen haben.

Ulm, 2020 Stefan Funken, Karsten Urban, Constantin Greif

Incomplete Cholesky

Benjamin Bestler, Joachim Kröner, Sebastian Acerbi

18. September 2021

Inhaltsverzeichnis

1 Einleitung

Partielle Differentialgleichungen stellen eine der häufigsten Quellen für Probleme mit dünnbesetzten Matrizen dar. Eine typische Vorgehensweise ist die Diskrektrisierung der Gleichungen mit einer finiten Anzahl an Unbekannten. Die hierdurch entstehenden Gleichungssysteme sind meist sehr groß, dünnbesetzt und oftmals symmetrisch positiv definit.[2]

Als Lösungsmethoden für solche Probleme können sich klassische direkte Löser als nicht praktikabel erweisen. Diese nutzen die spezielle Struktur der Matrix nicht aus, und können sogar zu dicht besetzen Zerlegungen führen. Als geeigneter stellen sich iterative Methoden heraus, die sich schrittweise der optimalen Lösung annähern, heraus. Ein bewährtes Verfahren dieser Klasse ist das konjugierte Gradientenverfahren, welches fÃ¼r symmetrisch positiv definitite Matrizen anwendbar ist. Die anspruchvollste Operation stellt hier das Matrix-Vektor Produkt dar. Dieses kann für dünnbesetze Matrizen mit einem günstigen Aufwand berechnet werden. [3]

Allerdings weisen iterative Verfahren, wie das cg-Verfahren, eine geringere Robustheit als direkte Verfahren auf. Außerdem hängt die Konvergenzgeschwindigkeit von der Kondition der Matrix ab. Bei Problemen, die sich aus typischen Anwendungen wie der Strümungsdynamik oder der Festigkeits- und Verformungsanalyse ergeben, kann dies zu einer langsamen Konvergenz fÃ¼hren. Die Effizienz sowohl als die Robustheit können jedoch signifikant durch geeignete Vorkonditionierung signifikant verbessert werden. [2]

Die Wahl eines Vorkonditionierers hÃ¤ngt stark von den Eigenschaften der Matrix ab, weshalb die Wahl eines geeigneten Vorkonditionierers nicht leich ist. Während direkte Lösungsverfahren zur Lösung von groÃŸen und dünnbesetzten Gleichungssystemen oft nicht anwendbar sind, stellen sie in modifizierter Form geeignete Vorkonditionierer fÃ¼r iterative Verfahren dar. FÃ¼r

Im Rahmen der Arbeit werden eine Auswahl an Verfahren zur Vorkonditionierung des konjugierten Gradientenverfahrens vorgestellt und verglichen. Folgende Verfahren wurden als Vorkonditionierer angewendet: Jacob-Verfahren, Gauss-Seidel, multigrid, Incomplete Cholesky und INCE(0).

Hier noch ein Algorithmus:

Algorithm 1 Variable metric hybrid inexact proximal point method

function VMHIPPM(f, s, z^0)
  $k \leftarrow 0$
  while $k < k_{\max}$ **do**
    $(u^k, g^k, \epsilon_k) \coloneqq \text{BUNDLE}(f, s, M_k, z^k, \delta_k)$ ▷ Approximate the proximal point
    if $\frac{1}{2}\|g^k\|^2 \leq \rho \ \wedge \ \epsilon_k \leq \rho$ **then**
      return u^k ▷ Optimal or close to optimal
    end if
    $z^{k+1} \coloneqq u^k$ ▷ Step to the proximal point
    $k \leftarrow k + 1$
  end while
end function

Hier noch ein Code:

```
#include <mpi.h>
#include <stdio.h>
#include <string.h>

int main(int argc, char** argv) {
  int my_rank;              /* Rank of process */
  int p;                    /* Number of process */
  int source;               /* Rank of sender */
  int dest;                 /* Rank of revceiver */
  int tag = 50;             /* Tag for messages */
  char message[100];        /* Storage for the message */
  MPI_Status status;        /* Return status for receive */

  MPI_Init(&argc, &argv);
  MPI_Comm_rank(MPI_COMM_WORLD, &my_rank);
  MPI_Comm_size(MPI_COMM_WORLD, &p);

  if (my_rank != 0){
    sprintf(message, "Hello world from processor %d!",my_rank);
    dest = 0;
    /* Use strlen(message)+1 to include '\0' */
    MPI_Send(message,strlen(message)+1,MPI_CHAR, dest, tag,
       MPI_COMM_WORLD);
  } else {
```

```
24      for (source=1; source<p; source++){
25        MPI_Recv(message,100,MPI_CHAR,source,tag,MPI_COMM_WORLD,
              &status);
26        printf("%s\n",message);
27      }
28    }
29    MPI_Finalize();
30 }
```

./ProjectX/Code/HelloWorld.c

Literatur

[1] A. Ben-Tal, M. Kocvara, A. Nemirovski, J. Zowe: Free Material Design via Semidefinit Programming, SIAM Review, Vol. 42, No. 4 (2000), 695-715.

[2] Saad, Yousef : Iterative Methods for Sparse Linear Systems, SIAM, Second Edition, (2003).

[3] Golub, Gene H. and Van Loan, Charles F. : Matric Computations, The Johns Hopkins University Press, Third Edition, (1996).