Otto-von-Guericke-Universität Magdeburg
Max-Planck-Institut for Dynamics of Complex Technical Systeme
Computational Methods in Systems and Control Theory

Dr. Jens Saak, Dipl.-Math. Martin Köhler
Webseite: http://www.mpi-magdeburg.mpg.de/mpcsc/lehre/2013_SS_SC/

Sommer Semester 2013

Scientific Computing 2
Hausaufgabenblatt 4

Ausgabe: 21. Mai 2013 **Abgabe:** 28. Mai 2012

Aufgabe 1: (10 Punkte)

Wir betrachten das dünnbesetzte Matrix-Vektor Produkt $x = Ay$. Die Matrix A sei dabei im Compressed Row Storage Format gespeichert.

a.) Implementieren sie das Matrix-Vektor Produkt als C-Funktion

```
void sparse_mvp(struct sparse_matrix_st *A, double *y, double *y);
```

Parallelisieren Sie diese Funktion geeignet mittels OpenMP. Testen Sie mit Hilfe der auf der Vorlesungswebseite bereitgestellten Matrix diese Funktion. Evaluieren Sie die verschiedenen OpenMP-Scheduler Eigenschaften: `dynamic`, `static`, `guided` und `auto`. Variieren Sie die Chunk-Sizes zwischen 1 und 4096. Sind Änderungen der Laufzeit festzustellen?

b.) Wir nehmen an, dass das Matrix-Vektor Produkt auf folgenden Computer berechnet werden soll:

- 2 8-Kern CPUs mit 2,9 GHz. Peak Performance je CPU 180 GFlops/s,
- Jeder CPU besitzt einen eigenen Speicherkontroller, welcher den Arbeitsspeicher mit jeweils 21 GiB/s anbindet.
- 20 MB Level-3 Cache je CPU.

Bestimmen Sie die optimale Threadanzahl und die dazugehörige maximale Floprate, unter der Annahme das eine Matrix $A \in \mathbb{R}^{171657\times 171657}$ mit $5\ 144\ 734$ Nicht-Null-Einträgen verwendet wird. Es kann davon ausgegangen werden, dass beide Vektoren x und y im Cache des CPUs verbleiben.

Ein Grundgerüst steht unter http://www.mpi-magdeburg.mpg.de/mpcsc/lehre/2013_SS_SC/Data/sparse_skeleton.tar.gz zur Verfügung.
Die Testmatrix kann von http://www.mpi-magdeburg.mpg.de/mpcsc/lehre/2013_SS_SC/Data/sparse_matrix.mtx.gz herunter geladen werden. Diese muss vor der Verwendung mit `gunzip` entpackt werden.

Aufgabe 2: (10 Punkte)

In der Vorlesung wurde die sogenannte “Sparse-Approximate-Inverse” als alternativer, gut zu parallelisierender Vorkonditionierer vorgestellt. Leiten Sie einen Algorithmus zu Berechnung der “Sparse-Approximate-Inverse” einer Bandmatrix $A \in \mathbb{R}^{n\times n}$ her. Die Bandbreite dieser Matrix sei k. Geben Sie konkrete Vorschläge zur Parallelisierung an und nennen Sie mögliche Probleme.
Bonus(+5 Punkte): Implementieren Sie dieses Verfahren. Auftretende kleinste-Quadrate Probleme sind mit `DGELS` aus LAPACK zu lösen. Die Bandmatrix sei als Dense Matrix gespeichert.

Aufgabe 3: **(10 Punkte)**

Neben der LU-Zerlegung ist das Lösen von Dreieckssystemen eine der wichtigsten Grundoperationen im wissenschaftlichen Rechnen. Zum Lösen eines Gleichungssystem mit untrem Dreieck (L) wurde in der Vorlesung eine geblockte Version des Vorwärts-Einsetzen vorgestellt. Implementieren Sie diese mit Hilfe entsprechender BLAS Routinen. Kann das innere Update, d.h. die Summe über die bereits fertigen Komponenten der Lösung, so umgeschrieben werden, dass es nur mit einem BLAS Aufruf je Iteration berechnet werden kann? Wenn ja implementieren Sie auch diese Variante. Wenn nein, nennen sie Gründe, warum dies nicht möglich ist.

Gesamtpunktzahl: 30