COSC 462

Solving Linear Systems

Piotr Luszczek
Cannon’s (Systolic) Algorithm Recap

Distribution and partition in 2D has scalability advantages.
Applications Using Linear Solve

- Structural analysis (civil engineering)
- Heat transport (mechanical engineering)
- Analysis of power grids (electrical engineering)
- Production planning (economics)
- Regression analysis (statistics)
- Antenna/radar/stealth fighter design (electromagnetics)
- Plasma containment (physics)
- Benchmarking (TOP500, HPL)
### Gaussian Elimination Example

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th>=</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>+x</td>
<td>-3y</td>
<td>+z</td>
<td>=</td>
<td>+4</td>
</tr>
<tr>
<td>+2x</td>
<td>-8y</td>
<td>+8z</td>
<td>=</td>
<td>-2</td>
</tr>
<tr>
<td>-6x</td>
<td>+3y</td>
<td>-15z</td>
<td>=</td>
<td>9</td>
</tr>
</tbody>
</table>

Could divide by 2 to get row values closer to 1

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th>=</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>+x</td>
<td>-3y</td>
<td>+z</td>
<td>=</td>
<td>+4</td>
</tr>
<tr>
<td>0x</td>
<td>-2y</td>
<td>+6z</td>
<td>=</td>
<td>-10</td>
</tr>
<tr>
<td>0x</td>
<td>-15y</td>
<td>-9z</td>
<td>=</td>
<td>33</td>
</tr>
</tbody>
</table>

Could divide by 3 to get row values closer to 1

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th>=</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>+x</td>
<td>-3y</td>
<td>+z</td>
<td>=</td>
<td>+4</td>
</tr>
<tr>
<td>0x</td>
<td>-y</td>
<td>+3z</td>
<td>=</td>
<td>-5</td>
</tr>
<tr>
<td>0x</td>
<td>-5y</td>
<td>-3z</td>
<td>=</td>
<td>11</td>
</tr>
</tbody>
</table>

Could divide by 5 to get row values closer to 1

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th>=</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>+x</td>
<td>-3y</td>
<td>+z</td>
<td>=</td>
<td>+4</td>
</tr>
<tr>
<td>0x</td>
<td>-y</td>
<td>+3z</td>
<td>=</td>
<td>-5</td>
</tr>
<tr>
<td>0x</td>
<td>0y</td>
<td>-18z</td>
<td>=</td>
<td>36</td>
</tr>
</tbody>
</table>

Back-substitution:

- \( x = 3 \)
- \( y = -1 \)
- \( z = -2 \)
Reference Implementation

- \(Ax = b\)
  - \(A\) is \(N\) by \(N\) matrix
  - \(x, b\) are \(N\) by 1 vectors
- For (\(i = 0; i < N; ++i\))
  
  \[
  \text{pivot} = A[\text{max_loc}(\text{abs}(A[:][i]))][j]
  \]

  \[
  A[i+1:N][j] /= \text{pivot}
  \]

  For (\(j = i+1; j < N; ++j\))
    For (\(k = i+1; j < N; ++j\))
      \[
      A[j][k] -= A[k][i] \times A[i][j]
      \]

- Complexity
  - \(\frac{2}{3}N^3\)
Floating Point Arithmetic Primer

- Floating point numbers in a computer are stored in the IEEE 754 standard (1985, 2008)
  - A subset of rational numbers and infinities, NaN's, -0
  - Binary representation is sign, mantissa, exponent
  - Multiple sizes available
    - 16-bits (half precision, storage only)
    - 32-bits (single precision)
    - 64-bits (double precision)
    - 80-bits (extended precision)
    - 128-bits (quad precision)

The most amount of numbers per length

round up to: \((x_1 + x_2) + \varepsilon\)
Row Pivoting for Numerical Stability

Row pivoting:

\[
\begin{align*}
+x & -3y & +z & = & +4 \\
+2x & -8y & +8z & = & -2 \\
-6x & +3y & -15z & = & 9 \\
-6x & +3y & -15z & = & +9 \\
+2x & -8y & +8z & = & -2 \\
+x & -3y & +z & = & +4 \\
\end{align*}
\]

No pivoting:

\[
\begin{align*}
+x & -3y & +z & = & +4 \\
0x & -2y & +6z & = & -10 \\
0x & -15y & -9z & = & 33 \\
\end{align*}
\]

\[
\frac{\text{Without proper down-scaling, errors get multiplied and the magnitude of updated entries grows: phenomenon call pivot growth.}}{6}
\]

\[
\begin{align*}
-x & +\frac{1}{2}y & -\frac{5}{2}z & = & +\frac{3}{2} \\
+\frac{1}{3}x & -\frac{4}{3}y & +\frac{3}{2}z & = & +\frac{1}{2} \\
+\frac{1}{6}x & -\frac{1}{2}y & +\frac{z}{6} & = & +\frac{2}{3} \\
-x & +\frac{1}{2}y & -\frac{5}{2}z & = & 3/2 \\
0x & -\frac{7}{2}y & +\frac{8}{6}z & = & -\frac{5}{6} \\
0x & -\frac{5}{2}y & -\frac{3}{2}z & = & +\frac{11}{2} \\
\end{align*}
\]
Agglomeration: Blocking

1. Local pivot find-and-swap
2. Local solve
3. Global pivot swap
4. Global triangular update (DTRSM)
5. Global rectangular update (DGEMM)
Non-cyclic distribution sequentializes the computational steps:
1. Solve first block
2. Wait for pivot information and scaling factors.

Cyclic distribution in both dimensions minimizes communication and improves scalability.
Mapping: 2D Block Cyclic Distribution
Divisibility and Padding

- If $P$ is not a square of an integer
  - Use prime factors of $P$ to form as square of the process grid as possible
  - If $P$ is a prime then leave some of the processes out of the grid to get close-to-square grid
- If $N$ is not divisible by $P$
  - Consider implementing clean-up code
  - If extra operations are OK, pad the matrix with 0's and 1's
    - Example: extend to 8

```
| 0 0 0 1 0 0 |
| 0 0 0 0 1 0 |
| 0 0 0 0 0 1 |
```

identity matrix extension

padding

```
| aaaaa000 |
| aaaaa000 |
| aaaaa000 |
| aaaaa000 |
| aaaaa000 |
|
```

padding