Lesson 104: Matrix diagonalization

Singular Value Decomposition

- The Singular Value Decomposition (SVD) of a $n \times n$ matrix M is given by:

$$M = U\Sigma V^T$$

- U and V are orthogonal matrices and Σ is a diagonal matrix of singular values
- There are many methods to calculate SVD, Jacobi method is one of them
- The **Jacobi method** seeks to systematically reduce the off-diagonal elements to zero. This is done by applying a sequence of plane rotations to M which transforms M into Σ.
- Several sweeps over the entire matrix M may be necessary to complete the SVD.
- Within each sweep, the matrix elements need to be paired and appropriate rotations needs to be calculated. The $n \times n$ matrix is partitioned in $n/2 \times n/2$ blocks, each block being a 2×2 matrix.

Singular Value Decomposition – Jacobi method

- Assume the following matrix M:

$$M = \begin{pmatrix} m_{00} & \dots & m_{0i} & \dots & m_{0j} & \dots & m_{0n} \\ \vdots & & \vdots & & \vdots & & \vdots \\ m_{i0} & \dots & m_{ii} & \dots & m_{ij} & \dots & m_{in} \\ \vdots & & \vdots & & \vdots & & \vdots \\ m_{j0} & \dots & m_{ji} & \dots & m_{jj} & \dots & m_{jn} \\ \vdots & & \vdots & & \vdots & & \vdots \\ m_{n0} & \dots & m_{ni} & \dots & m_{nj} & \dots & m_{nn} \end{pmatrix}$$

- Choose (i, j) such that $|m_{ij}|$ is the maximum non-diagonal element
- For the following matrix, force m_{ij} and m_{ji} to vanish

$$\begin{pmatrix} m_{ii} & m_{ij} \\ m_{ji} & m_{jj} \end{pmatrix}$$

- Propagate the computation effects along the rows and columns

Singular Value Decomposition – Jacobi method

- Major drawback: Jacobi method requires at each step the scanning of $n(n-1)/2$ numbers for one of maximum modulus
  - This can be time consuming for large matrices
- **Cyclic Jacobi method**: select the pairs (i, j) in some cyclic order
- Try the following order (cyclic-by-rows):

  $1-2, 1-3, \ldots, 1-n, 2-3, \ldots, 2-n, 3-4, \ldots (n-1)-n$
- More than one sweep may be needed!
- Although some on-diagonal energy may go off-diagonal at some iterations, the process is known to converge in a small number of sweeps
- *It is not needed to vanish a non-diagonal element completely!*
  - Think in terms of off-diagonal energy going on-diagonal

Singular Value Decomposition – the core operation

- The basic operation is the two-sided rotation of each 2×2 matrix.

$$R(\theta_l)^T \begin{pmatrix} a & b \\ c & d \end{pmatrix} R(\theta_r) = \begin{pmatrix} \Psi_1 & 0 \\ 0 & \Psi_2 \end{pmatrix}$$

  where θ_l and θ_r are the left and right rotation angles, respectively.

- The input 2×2 matrix subject to diagonalization is:

$$\begin{pmatrix} a & b \\ c & d \end{pmatrix}$$

- A rotation matrix has the following form:

$$\begin{pmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{pmatrix}$$

- Two issues need to be addressed:
  - Calculation of the rotation angles
  - Performing the rotations

University of Victoria
British Columbia • Canada

Singular Value Decomposition – operation budget

- **Calculation of the rotation angles requires**:
  - The evaluation of $\arctan$
- $\arctan$ is a transcendental function
  - Is series expansion appropriate to evaluate $\arctan$?
- **Performing the rotations requires**:
  - The evaluation of $\cos$ and $\sin$
  - Matrix multiplication
- $\cos$ and $\sin$ are transcendental functions
  - Is series expansion appropriate to evaluate $\cos$ and $\sin$?
- Matrix multiplication can be carried out within the standard instruction set

Singular Value Decomposition (cont'd)

- The efficient computation of the rotation parameters is essential.
- The direct two-angle method calculates θ_l and θ_r by computing the inverse tangents of the data elements of M:

$$\theta_{\text{SUM}} = \theta_r + \theta_l = \arctan\left(\frac{c+b}{d-a}\right)$$

$$\theta_{\text{DIFF}} = \theta_r - \theta_l = \arctan\left(\frac{c-b}{d+a}\right)$$

- The two angles, θ_l and θ_r, can be separated from the sum and difference results and applied to the two-sided rotation module to diagonalize M.
- In a typical serial computer, the **calculation of the rotation angles** and **performing the rotations** are both expensive tasks.
- Provide architectural support (define a new instruction and deploy the associated computing unit) for $\arctan$, $\cos$, $\sin$

How to calculate $\arctan(x)$?

- The function $\arctan : (-\infty, \infty) \longrightarrow \left(-\frac{\pi}{2}, \frac{\pi}{2}\right)$
  - Integer representation: we clearly don't like a domain like $(-\infty, \infty)$
  - **An idea!**
    - Calculate $\arctan(x)$ when $|x| \leq 1$
    - Calculate $\mathrm{arccot}(x)$ when $|x| > 1$ and adjust the angle accordingly

- In C using floating point:

```
#include <math.h>
int x, y, angle;

if (x > y)
  angle = arctan( y/x);         /* arctan() returns a float */
else
  angle = PI/2 - arctan( x/y);  /* arctan() returns a float */
```

University of Victoria
British Columbia • Canada

How to calculate $\arctan(x)$?

- Integer arithmetic required!
  - C standard library (**math.h**): $\arctan()$ is a floating-point function
  - "/" is not a good option to divide integers
  - π is a fractional number
- Implement our own $\arctan()$ routine – what algorithm shall we use?
  - Taylor series expansion about a point – approximation good for 1 point

$$\arctan(x) = x - \frac{1}{3}x^3 + \frac{1}{5}x^5 - \frac{1}{7}x^7 + \ldots$$

  - Tchebishev polynomial – approximation good for an interval (homework)
  - Piecewise linear approximation with three middle points:

$$\arctan(x) = \begin{cases} 0.644\,x + 0.142 & \text{if } 0.5 < x \leq 1.0, \\ 0.928\,x & \text{if } -0.5 \leq x \leq 0.5, \\ 0.644\,x - 0.142 & \text{if } -1.0 \leq x < -0.5. \end{cases}$$

University of Victoria
British Columbia • Canada

$\arctan(x)$ – piecewise linear approximation

- The formula using fractional numbers

$$\arctan(x) = \begin{cases} 0.644\,x + 0.142 & \text{if } 0.5 < x \leq 1.0, \\ 0.928\,x & \text{if } -0.5 \leq x \leq 0.5, \\ 0.644\,x - 0.142 & \text{if } -1.0 \leq x < -0.5. \end{cases}$$

- From fractional to integer (assume 12-bit signed integer representation):
  - 1.0 is *represented* as 2^{11} (in fact, as $2^{11} - 1$)
  - 0.928 is represented as $1900 = 76\mathrm{C_h}$
  - 0.644 is represented as $1319 = 527_\mathrm{h}$
  - 0.142 is represented as $291 = 123_\mathrm{h}$
  - 0.5 is represented as $1024 = 400_\mathrm{h}$
  - x is represented as $X = 2^{11}\,x$

$\arctan(x)$ – piecewise linear approximation

- Piecewise linear approximation using integer arithmetic

$$\arctan(X) = \begin{cases} 1319\,X + 291 & \text{if } 1024 < X \leq 2048, \\ 1900\,X & \text{if } -1024 \leq X \leq 1024, \\ 1319\,X - 291 & \text{if } -2048 \leq X < -1024. \end{cases}$$

- $\arctan(X)$ is a signed integer ranging $-1,350,947 \cdots + 1,350,947$, which in hex is $-149\mathrm{D}23_{\mathrm{h}} \cdots + 149\mathrm{D}23_{\mathrm{h}}$
  - Homework: how many bits are needed to represent $\arctan(X)$?

- Questions that can be posed:
  - Computing time: software implementation versus hardware implementation
  - Precision of the piecewise linear approximation using integer arithmetic

- Same problem for $\sin(x)$ and $\cos(x)$

University of Victoria
British Columbia • Canada

Jacobi method – side effects

- It works fine with rectangular matrices, too.
- If the matrix is symmetric, the algorithm finds the eigenvalues.
- Matrix triangularization can be achieved with one-side rotations
  - Upper triangularization with left-side rotations
  - Lower triangularization with right-side rotations

Jacobi method – bibliography

- Professor Richard P. Brent:

  `http://web.comlab.ox.ac.uk/oucl/work/richard.brent/`

- Any textbook on linear algebra

Matrix diagonalization – project requirements

- Build the testbench: the input is a square matrix of integers
- Assume the piecewise linear approximation for $\arctan$, $\sin(x)$, and $\cos(x)$, and determine the maximum error for an approximation with three middle points.
- Implement piecewise linear approximation using integer arithmetic for $\sin$, $\cos$, and $\arctan$ in:
  - software (write C routines)
  - horizontal firmware with two issue slots
  - custom hardware (write VHDL/Verilog)
- Define a new instruction that will return the trigonometric function
  - You must comply with the ARM architecture (you can have at most two arguments and one result per instruction call)

Matrix diagonalization – project requirements

- Rewrite the high-level code and instantiate the new instruction
  - Use assembly inlining
- Diagonalize a square matrix using piecewise linear approximation of trigonometric functions and estimate:
  - the performance improvement of hardware-based solution versus software-based solution
  - the performance improvement of a 2-issue slot firmware-based solution versus software-based solution
- Estimate the penalty in terms of number of gates for the hardware solution

Questions, feedback

?