A Parallel Method for the Computation of Matrix Exponential based on Truncated Neumann Series

V. S. Dimitrov[1,2], V. Ariyarathna[3], D. F. G. Coelho[1], L. Rakai[1], A. Madanayake[3], R. J. Cintra[4]

[1]ECE Department, University of Calgary, Canada
[2]Computer Modelling Group, Canada [3]ECE Department, University of Akron, USA
[4]Statistics Department, Universidade Federal de Pernambuco, Brazil

July 20, 2017

UNIVERSITY OF CALGARY

Problems in many areas require the solution of sets of linear, constant coefficient differential equations in the form:

$$\dot{\mathbf{x}}(t) = \mathbf{A}\mathbf{x}(t) \implies \mathbf{x}(t) = \exp(t\mathbf{A})\mathbf{x}_0$$

When multiple inputs are used for the same system, it might be advantageous compute the matrix exponential.

- Series expansion:
  - Taylor;
  - Padé;
  - Scaling & Squaring.
- Newton Interpolation;
- Cayley-Hamilton method;
- Eigenvectors decomposition.

- The problem can be treated as the evaluation of a polynomial.

- Existing methods:
  - Horner rule;
  - Estrin method;
  - Binary tree.

Conventions

- Let $\mathbf{A}$ be a square matrix of size $n \times n$.
- Let $p_N(\cdot)$ be a polynomial of degree $N-1$ over the real numbers.
- Let also $g_N(\cdot)$ be a geometric series with N terms.

The critical path associated with the computation of a matrix polynomial $p_N(\mathbf{A})$ is the largest chain of matrix multiplications (MM) in order to evaluate $p_N(\mathbf{A})$.

Definition (Critical Path for Matrix Polynomial)

- Horner rule: $N - 1$ MM;
- Estrin method: $2\log_2(N-1)$ MM;
- Binary tree: $2\log_2(N-1)$ MM.

Geometric series of matrix arguments can be computed efficiently with the use of different polynomial factorizations.

$$g_N(\mathbf{A}) = \begin{cases} (\mathbf{I} + \mathbf{A}^2) \cdot g_{N/2}(\mathbf{A}^2), & \text{if } N \equiv 0 \bmod 2 \\ \mathbf{I} + (\mathbf{A} + \mathbf{A}^2) \cdot g_{(N-1)/2}(\mathbf{A}^2), & \text{if } N \equiv 1 \bmod 2. \end{cases}$$

$$g_N(\mathbf{A}) = \begin{cases} (\mathbf{I} + \mathbf{A} + \mathbf{A}^2) \cdot g_{N/3}(\mathbf{A}^3), & \text{if } N \equiv 0 \bmod 3, \\ \mathbf{I} + (\mathbf{A} + \mathbf{A}^2 + \mathbf{A}^3) \cdot g_{(N-1)/3}(\mathbf{A}^3), & \text{if } N \equiv 1 \bmod 3, \\ \mathbf{I} + \mathbf{A} + (\mathbf{A}^2 + \mathbf{A}^3 + \mathbf{A}^4) \cdot g_{(N-2)/3}(\mathbf{A}^3), & \text{if } N \equiv 2 \bmod 3. \end{cases}$$

In general, the use of basis P demands $P' \log_2(N) - 2 < 2 \log_2(N) - 2$.

In general, the use of basis P demands $P' \log_2(N) - 2 < 2\log_2(N) - 2$.

Examples:

- Basis 2: $2\log_2(N) - 2$;
- Basis 3: $1.89\ldots\log_2(N) - 2$;
- Basis 5: $1.72\ldots\log_2(N) - 2$;
- Basis 6: $1.92\ldots\log_2(N) - 2$;
- Basis 26: $1.70\ldots\log_2(N) - 2$.

We write the matrix exponential truncated series expansion $p_N(\mathbf{A})$ as a linear combination of different geometric series on $\alpha_k \mathbf{A}$, $k = 0, 1, \ldots, N-1$:

$$\begin{aligned} p_N(\mathbf{A}) &= \sum_{n=0}^{N-1} g_{n+1}(\alpha_n \mathbf{A}) \\ &= \sum_{n=0}^{N-1} \left(\sum_{k=0}^{n-1} \alpha_n^k \mathbf{A}^k \right) \\ &= \sum_{n=0}^{N-1} \left(\sum_{k=n}^{N-1} \alpha_k^n \right) \cdot \mathbf{A}^n. \end{aligned}$$

If the coefficients of $p_N(\cdot)$ are $p_0, p_1, \ldots, p_{N-1}$, we have he system

$$\begin{aligned} \alpha_0 + \alpha_1 + \alpha_2 + \ldots + \alpha_{N-1} &= p_0 \\ \alpha_1^2 + \alpha_2^2 + \ldots + \alpha_{N-1}^2 &= p_1 \\ \alpha_2^3 + \ldots + a_{N-1}^3 &= p_2 \\ &\vdots \\ \alpha_{N-1}^N &= p_{N-1}. \end{aligned}$$

This system has several complex solutions that can be found by back substitution.

A Numerical Example

Small degree polynomials does not require complex solutions.
Considering $N = 4$, we have

$$\alpha_0 = 0.451801$$
$$\alpha_1 = 0.420627$$
$$\alpha_2 = 0.344837$$
$$\alpha_3 = -0.217308$$

Table: Calculated coefficients for $N = 9$.

Coefficient	Value
α_8	0.265650069930254
α_7	0.270164634258582
α_6	0.294213807881822
α_5	0.320211897615876
α_4	0.339931939777992
α_3	0.312847569943447
β_1	−0.803019919407973
β_0	−0.046083639081852

If we modify the formulation to

$$p_N(\mathbf{A}) = \sum_{n=0}^{N-1} g_N(\alpha_n \mathbf{A}) = \sum_{n=0}^{N-1} \left(\sum_{k=0}^{N-1} \alpha_k^n \right) \cdot \mathbf{A}^n.$$

we obtain

$$\begin{aligned} \alpha_0 + \alpha_1 + \alpha_2 + \ldots + \alpha_{N-1} &= 1 \\ \alpha_0^2 + \alpha_1^2 + \alpha_2^2 + \ldots + \alpha_{N-1}^2 &= \frac{1}{2} \\ &\vdots \\ \alpha_0^{N-1} + \alpha_1^{N-1} + \alpha_2^{N-1} + \ldots + \alpha_{N-1}^{N-1} &= \frac{1}{(N-1)!} \end{aligned}$$

Algorithmic Example for $N = 9$

Pre Computation: $\mathbf{B} = \mathbf{A}^2$, $\mathbf{C} = \mathbf{B}^2$, broadcast $\mathbf{A}$, $\mathbf{B}$, $\mathbf{C}$, β_0, and β_1

Processor 0 computes $H_9(\mathbf{A}) \leftarrow \beta_0\mathbf{A} + \beta_1\mathbf{B} - (N-4)\mathbf{I}$
Processor 1 computes $g_4(\alpha_3\mathbf{A}) \leftarrow (\mathbf{I} + \alpha_3\mathbf{A})(\mathbf{I} + \alpha_3^2\mathbf{B})$
Processor 2 computes $g_5(\alpha_4\mathbf{A}) \leftarrow \mathbf{I} + (\alpha_4\mathbf{A} + \alpha_4^2\mathbf{B})(\mathbf{I} + \alpha_4^2\mathbf{B})$
Processor 3 computes $g_6(\alpha_5\mathbf{A}) \leftarrow (\mathbf{I} + \alpha_5\mathbf{A})(\mathbf{I} + \alpha_5^2\mathbf{B} + \alpha_5^4\mathbf{C})$
Processor 4 computes $g_7(\alpha_6\mathbf{A}) \leftarrow \mathbf{I} + (\alpha_6\mathbf{A} + \alpha_6^2\mathbf{B})(\mathbf{I} + \alpha_6^2\mathbf{B} + \alpha_6^4\mathbf{C})$
Processor 5 computes $g_8(\alpha_7\mathbf{A}) \leftarrow (\mathbf{I} + \alpha_7\mathbf{A})(\mathbf{I} + \alpha_7^2\mathbf{B})(\mathbf{I} + \alpha_7^4\mathbf{C})$
Processor 6 computes $g_9(\alpha_8\mathbf{A}) \leftarrow \mathbf{I} + (\alpha_8\mathbf{A} + \alpha_8^2\mathbf{B})(\mathbf{I} + \alpha_8^2\mathbf{B})(\mathbf{I} + \alpha_8^4\mathbf{C})$

Return $E_9(\mathbf{A}) = \sum_{n=3}^{9} g_{n+1}(\alpha_n\mathbf{A}) + H_9(\mathbf{A})$

Figure: Fragment of the algorithm for computing $E_9(\mathbf{A})$.

UNIVERSITY OF CALGARY

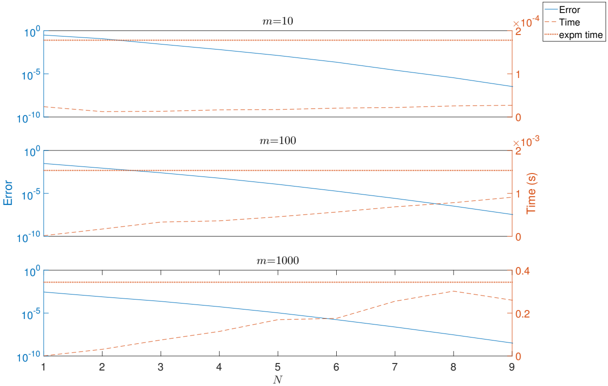

Figure: Illustration of the accuracy versus computing time trade-off for different values of N and m.

Hardware Realization

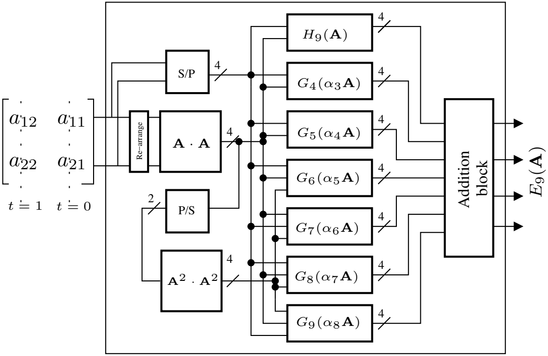

Figure: Top level view of the implementation of the proposed algorithm.

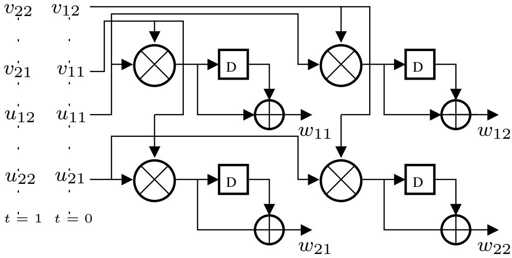

Figure: Multiplication block for 2×2 matrices.

Table: Timing and resource consumption comparison for Xilinx xc6vlx240t-ff1156 FPGA

Figure of merit	Horner's Rule	New Algorithm
Latency (clock cycles)	16	6
Critical path delay (ns)	14.392	11.160
Slice LUTs used	24537	90993
No. of adders	48	118
No. of multipliers	32	112

Table: ASIC synthesis results

Figure of merit	Horner's Rule	New Algorithm	Percentage Change
T (ns)	3.556	1.350	$\downarrow$ 62.04 %
Occupied area (A, mm^2)	1.355	4.080	$\uparrow$ 201.10%
Dynamic power ($\mathrm{mW/GHz}$)	3199.088	2403.661	$\downarrow$ 24.86%
AT ($\mathrm{mm}^2 \cdot \mathrm{ns}$)	4.8175	5.5083	$\uparrow$ 14.34%
AT^2 ($\mathrm{mm}^2 \cdot \mathrm{ns}^2$)	17.1313	7.4363	$\downarrow$ 56.59%
Max frequency (GHz)	0.2812	0.7407	$\uparrow$ 163.40%
Latency (Clock cycles)	16	6	$\downarrow$ 62.5 %
Total gate count	61009	131874	$\uparrow$ 116.15%

- Advantages:
  - the proposed method reduce critical path;
- Disadvantages:
  - requires more processors and memory (software);
  - requires more hardware resources such as LUT and gates (hardware).
- Future works:
  - consider different combinations of Neumann series for different solutions (real solution possible?);
  - consider more matrix functions and general polynomials;
  - provide accurate error analysis.

Questions?