Zero-Copy Parallelized Empirical Gramians

Christian Himpe (christian.himpe@uni-muenster.de)
Mario Ohlberger (mario.ohlberger@uni-muenster.de)

WWU Münster
Institute for Computational and Applied Mathematics

PACO 2015
2015-07-06

What are empirical gramians?

$\rightarrow$ Empirical gramians are a system-theoretic tool for model reduction and system identification of input-ouput systems!

Motivation

Why **Model Order Reduction**[1] (MOR)?

- Enable or accelerate large-scale system evaluation,
- repeatedly in many-query settings,
- such as model-constrained optimization or control problems.

Why **Power-Aware** Reduced Order Model (ROM) Computation?

- Reduce waste of computationally resources.
- Realistically, offline time is not infinite.
- Save the Planet!

[1] Model Reduction is in itself PACO by using ROMs, i.e. on low-power devices.

Outline[2]

[2] Disclaimer: There are no experimental results yet due to unreleased hardware.

Mathematical Background

State-Space Systems

(Nonlinear) State-Space System:

$$\dot{x}(t) = f(x(t), u(t), \theta),$$
$$y(t) = g(x(t), u(t), \theta),$$
$$x(0) = x_0$$

State: $x(t)$, $\dim(x(t)) \gg 1$

Input: $u(t)$, $\dim(u(t)) \ll \dim(x(t))$

Output: $y(t)$, $\dim(y(t)) \ll \dim(x(t))$

Parameters: θ, $\dim(\theta) \gg 1$

Linear State-Space System:

$$\dot{x}(t) = Ax(t) + Bu(t),$$
$$y(t) = Cx(t),$$
$$x(0) = x_0$$

System Matrix: $A \in \mathbb{R}^{N \times N}$

Input Matrix: $B \in \mathbb{R}^{N \times J}$

Output Matrix: $C \in \mathbb{R}^{O \times N}$

Projection-Based Model Reduction

Reduced Order Model (ROM):

$$\dot{x}_r(t) = f_r(x_r(t), u(t), \theta), \qquad \dim(x_r(t)) \ll \dim(x(t))$$
$$y_r(t) = g_r(x_r(t), u(t), \theta), \qquad \|y(\theta) - y_r(\theta)\| \ll 1$$
$$x_r(0) = x_{r,0}$$

Projection-Based ROM:

$$\dot{x}_r(t) = Vf(Ux_r(t), u(t), \theta), \qquad U \in \mathbb{R}^{\dim(x(t)) \times \dim(x_r(t))}$$
$$y_r(t) = g(Ux_r(t), u(t), \theta), \qquad V \in \mathbb{R}^{\dim(x_r(t)) \times \dim(x(t))}$$
$$x_r(0) = Vx_0 \qquad VU = \mathbb{1}$$

Gramian-Based Model Reduction [Moore'81,Fernando & Nicholson'83]

Controllability Gramian:

$$W_C = \int_0^\infty e^{At} B B^T e^{A^T t} dt$$

$$\Rightarrow A W_C + W_C A^T = -B B^T$$

Observability Gramian:

$$W_O = \int_0^\infty e^{A^T t} C^T C e^{At} dt$$

$$\Rightarrow A^T W_O + W_O A = -C^T C$$

$\rightarrow$ Balanced Truncation

Cross Gramian:

$$W_X = \int_0^\infty e^{At} B C e^{At} dt$$

$$\Rightarrow A W_X + W_X A = -BC$$

$\rightarrow$ Direct Truncation

Empirical Gramians [Lall et al'99]

System Gramians,
from primal and adjoint impulse response trajectories:

$$W_C = \int_0^\infty (e^{At}B)(e^{At}B)^T dt = \int_0^\infty x_\delta(t)(x_\delta(t))^T dt$$

$$W_O = \int_0^\infty (e^{At}C)^T(Ce^{At})dt = \int_0^\infty x_\delta^*(t)(x_\delta^*(t))^T dt$$

Empirical Gramians for Nonlinear Systems [H. & Ohlberger'14]

Extend to nonlinear system[3], since only based on trajectories!

$$W_X = \sum_{\{\Delta u, \Delta x_0\}} \int_0^\infty x_\delta(t)(x_\delta^*(t))^T dt$$

Empirical Gramians Computation:

1. Simulate Trajectories
2. Gramian Assembly

[3] This is better than linearization!

Combined Reduction [H. & Ohlberger'14]

Combined State and Parameter Reduction:

$$\dot{x}_r(t) = Vf(Ux_r(t), u(t), P\theta_r), \qquad P \in \mathbb{R}^{\dim(\theta_r)\times\dim(\theta)},\ P^T P = \mathbb{1}$$
$$y_r(t) = g(Ux_r(t), u(t), P\theta_r), \qquad \dim(\theta_r) \ll \dim(\theta)$$
$$x_r(0) = Vx_0 \qquad \|y(\theta) - y_r(\theta_r)\| \ll 1$$
$$\theta_r = P^T\theta$$

Augmented System:

$$\begin{pmatrix}\dot{x}(t)\\ \dot{\theta}(t)\end{pmatrix} = \begin{pmatrix}f(x(t), u(t), \theta(t))\\ 0\end{pmatrix}$$
$$y(t) = g(x(t), u(t), \theta(t))$$
$$\begin{pmatrix}x(0)\\ \theta(0)\end{pmatrix} = \begin{pmatrix}x_0\\ \theta_0\end{pmatrix}$$

Empirical Gramian Flavors[4] [H. & Ohlberger'13]

- Empirical Controllability Gramian
- Empirical Observability Gramian
- Empirical Cross Gramian
- Empirical Linear Cross Gramian
- Empirical Non-Symmetric Cross Gramian
- Empirical Sensitivity Gramian (Parameter Controllability)
- Empirical Identifiability Gramian (Parameter Observability)
- Empirical Joint Gramian (Cross-Gramian-Based Combined Reduction)

[4] Compute with: `emgr` - Empirical Gramian Framework (`http://gramian.de`)

Computational Means

CPU vs GPU

	CPU	GPU
Cores	10^1	10^3
Clock	$\sim$ 4 Ghz	$\sim$ 1 Ghz
Flow-Control	☺	☹
GEMM	☹	☺
Parallelism	SMT, MT, SIMD	SIMT, SIMD

CPU + iGPU

Previously, on *Game of CPUs*:

1989 80486DX: CPU + iFPU

2004 GMA900: CPU + iGPU (Somewhat of a GPU)

2011 Llano: CPU + iGPU (=:APU)

2012 Intel HD: CPU + iGPU

2014 Kaveri: HSA-Ready APU (+hUMA)

Generally:

- CPU and GPU in one package
- CPU and GPU use shared memory

Zero-Copy

GPGPU with:

- discrete GPU (roughly Distributed Memory):
  $$\text{CPU} \overset{\textit{copies}}{\rightarrow} \text{GPU} \overset{\textit{computes}}{\rightarrow} \text{GPU} \overset{\textit{copies}}{\rightarrow} \text{CPU}$$
- integrated GPU (nUMA - Non-Uniform Memory Access):
  $$\text{CPU} \overset{\textit{copies}}{\rightarrow} \text{GPU} \overset{\textit{computes}}{\rightarrow} \text{GPU} \overset{\textit{copies}}{\rightarrow} \text{CPU}$$
- zero-copy GPU (Shared Virtual Memory):
  $$\text{CPU} \overset{\textit{copies}/\textit{transfers}}{\rightarrow} \text{GPU} \overset{\textit{computes}}{\rightarrow} \text{GPU} \overset{\textit{copies}/\textit{transfers}}{\rightarrow} \text{CPU}$$
- zero-copy iGPU (hUMA - Heterogenous Unified Memory Access):
  $$\text{CPU} \overset{\textit{notifies}}{\rightarrow} \text{GPU} \overset{\textit{computes}}{\rightarrow} \text{GPU} \overset{\textit{notifies}}{\rightarrow} \text{CPU}$$

Easy As BLAS

Requirements:

- capable of zero-copy
- should not affect high-level code (i.e. Matlab)
- on BLAS layer

Weapon of Choice: **Offloading** via ACML[5]

Pro No changes in HLL implementation required

Con LUA configuration requires knowledge of the problem

Pro Effectively uses OpenCL

Con Works only on supported systems (i.e. HSA/huma)

Pro ACML backend is the open-source clBLAS

Con ACML is free (of charge), but not open-source

[5] The MKL can offload to Xeon Phi, too.

iGPU Offloading from ACML

Where the magic happens:
`acmlbasedir/gfortran64_mp/lib/resource`

1. create device context: `context.lua`

   see: `createContexts( platforms )`

2. BLAS config: `DEVICE_NAME/BLAS_CALL.lua`

   see for example: `Spectre/gemm.lua`

   CPU / GPU heuristics in: `tableOfThresholds`

Zero-Copy Empirical Gramians

$$W_X = \sum_{\{\Delta u, \Delta x_0\}} \int_0^\infty x_\delta(t)(x_\delta^*(t))^T dt$$

1. Trajectory Simulation
   - General Linear Methods such as (Two-Step) Runge-Kutta
   - Rather Serial
   - Parallelized on CPU (i.e. multiple trajectories in parallel each SIMD'd)

2. Gramian Assembly
   - Dense Matrix Multiplication
   - Very Parallel
   - (to be) Parallelized on GPU

Now, how is this PACO?

Why APUs?

- Discrete GPU needs (more) power
- Discrete GPU has relatively little memory
- Virtual Shared Memory uses PCIe or copies

- Integrated GPU needs less power

  (i.e. A10-7850 APU (95W) vs Athlon X4 CPU (65W) + HD 7750 GPU (55W)
- Integrated GPU[6] can use full(!) system main memory
- Literally, 0 copies!

[6] with hUMA or VSM

Mathematical Setup

Hyperbolic Network Model:

$$\dot{x}(t) = A\tanh(K(\theta)x(t)) + Bu(t)$$
$$y(t) = Cx(t)$$

System Dimensions:

- $x(t) \in \mathbb{R}^{1024}$
- $u(t) \in \mathbb{R}$
- $y(t) \in \mathbb{R}$
- $\dim(\theta) = \dim(x(t))$

Numerical Experiment

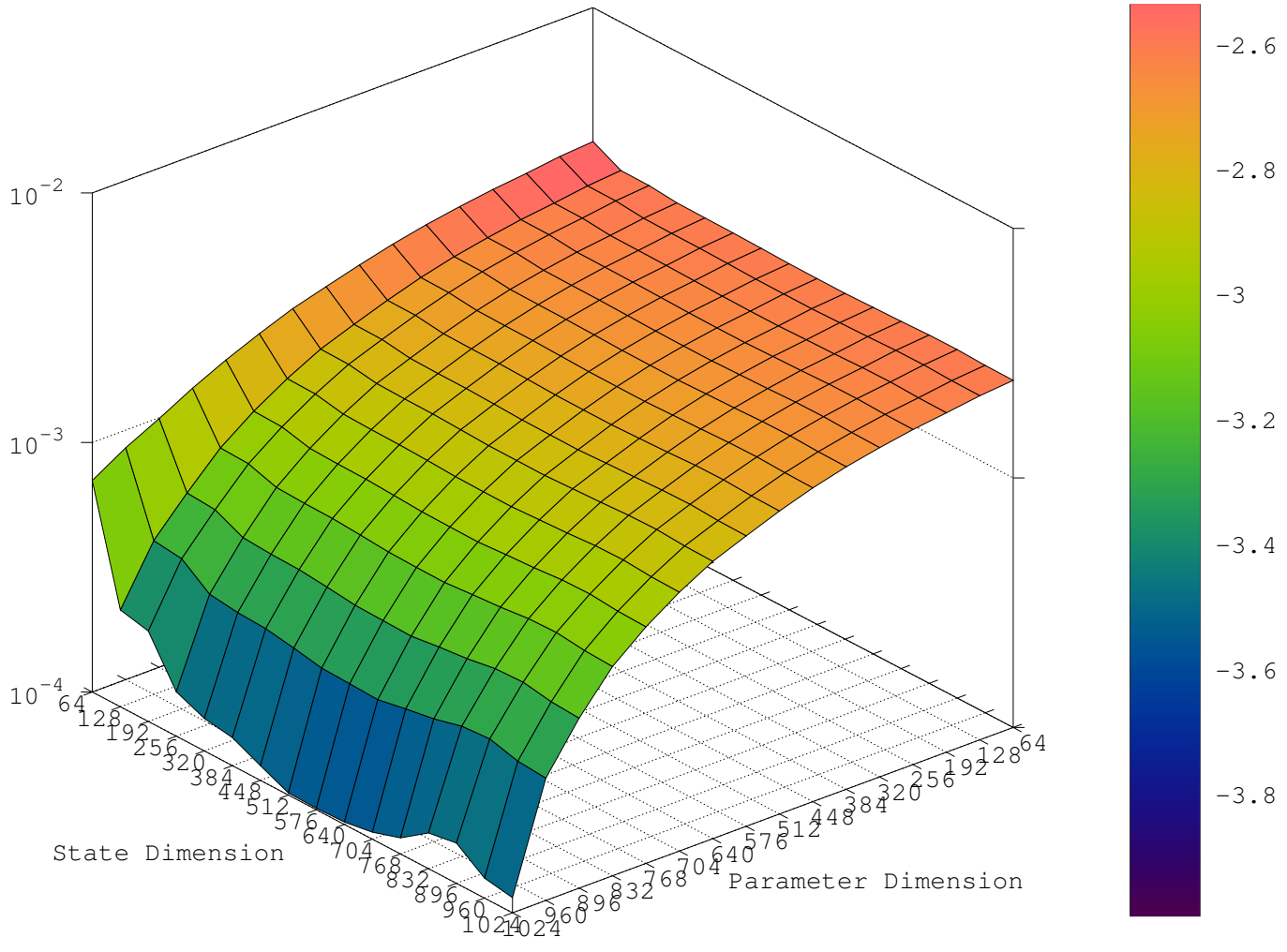

Figure : L_2-output-error for reduced state and parameter dimensions.

Performance

Offline Time Distribution:

- Trajectory Computation: 99%
- Gramian Assembly Time[7]: 1%

But:

- With $N = 1024$, this is still a small system.
- The trajectory computation parallelizes communication free.
- Can also be used for vector field evaluations[8].
- It is basically free.

[7] This has been reduced by using **Generalized Transpositions**.

[8] For $N > 512$ the default thresholds are met.

Outlook

Next, on *Game of CPUs* ("OpenCL 2.0 is coming!"):

2015 Carrizo: Full HSA 1.0 (+hUMA)

2015 Broadwell: OpenCL 2.0 support

hUMA / VSM is on the rise[9],

- attractive for scientific computing,
- accompanied by "smart" abstractions.

[9] This was one of the top developer requests for PS4 and XBone.

Rough Road Ahead!

So, hardware and opportunities are (about to be) available, yet...

- the software stack:
  1. Driver (Kernel, Propriety, HSA)
  2. OpenCL (ICD, Dispatch)
  3. Library (BLAS, LAPACK)

  interaction is not trivial;
- optimization of heuristics is tedious;
- distributed memory systems with APUs are not on the horizon;
- system memory is usually slower than video memory.

tl;dl

- GPGPU becomes more accessible,
- and potentially computes LA more energy-efficiently.
- PACO (planned) by exploiting iGPUs
- with zero-copy capabilities
- for empirical gramian computation[10].

`http://wwwmath.uni-muenster.de/u/himpe`

Thanks!

[10] Get the companion code: `http://j.mp/paco15`