Comprehensive Memory-Bound Simulations on Single Board Computers

Christian Himpe[1] Tobias Leibner[2] Stephan Rave[3]

Numerical simulations of increasingly complex models, demand growing amounts of (main) memory. Typically, large quantities of memory are provided by workstation- and server-type computers, but in turn consume massive amounts of power. Model order reduction can reduce the memory requirements of simulations by constructing reduced order models, yet the assembly of these surrogate models itself often requires memory-rich compute environments. We resolve this deadlock by careful algorithmic design of the model reduction technique. The presented empirical-cross-Gramian-based model reduction comprises two phases; in a first phase the empirical cross Gramian matrix is computed, secondly, a singular value decomposition of this system Gramian matrix reveals a low-rank projection, which can be applied to the original full order model. This model reduction approach can be realized economically memory-wise using the HAPOD algorithm, and we demonstrate its applicability on a low-end single board computer device.

1 Introduction

Numerical simulations of models based on parametric differential equations are an important tool in science and engineering. A common scenario is the repeated (multi-query, many-query) simulation for different parameters. Using high fidelity resolutions or more comprehensive models usually results in large-scale systems, which may even need to be processed on distributed memory systems due to memory or computational constraints. Such multi-node compute clusters consume wast amounts of power.

Model reduction can overcome computational complexity constraints by constructing algorithmically reduced order models. Yet, the assembly of the reduced model may require significant memory resources. Especially, data-driven model reduction techniques for the reduction of the time-domain representation of nonlinear systems need to simulate the full order model multiple times.

This work demonstrates that dense model reduction algorithms can be adapted to memory-constraints environments, not only by using reduced order models for the application simulation, the "online-phase"; but also for the assembly of the reduced order model,

[1]Computational Methods in Systems and Control Theory, Max Planck Institute for Dynamics of Complex Technical Systems, Sandtorstrasse 1, 39106 Magdeburg,
himpe@mpi-magdeburg.mpg.de

[2]Institute for Computational and Applied Mathematics, Westfälische Wilhelms Universtät, Einsteinstrasse 62, 48149 Münster,
tobias.leibner@uni-muenster.de

[3]Institute for Computational and Applied Mathematics, Westfälische Wilhelms Universtät, Einsteinstrasse 62, 48149 Münster,
stephan.rave@uni-muenster.de

the so-called "offline phase". Hence, a full-cycle memory economic simulation environment is provided. Due to the memory-resource independence, low-power or power-aware platforms become applicable for complex simulations.

In the scope of this work, input-output system models of the following form are considered:

$$\begin{aligned} \dot{x}(t) &= f(x(t), u(t)), \\ y(t) &= g(x(t), u(t)), \\ x(0) &= x_0, \end{aligned} \tag{1}$$

which consist of a dynamical system given by an ordinary differential equation (ODE) and an output function. This class of models maps an input function $u : \mathbb{R}_{\geq 0} \to \mathbb{R}^M$ via the state trajectory $x : \mathbb{R}_{\geq 0} \to \mathbb{R}^N$, that is the solution to the ODE with the vector field $f : \mathbb{R}^N \times \mathbb{R}^M \to \mathbb{R}^N$, to the output trajectory $y : \mathbb{R}_{\geq 0} \to \mathbb{R}^Q$ resulting from the output functional $g : \mathbb{R}^N \times \mathbb{R}^M \to \mathbb{R}^Q$.

2 Model Order Reduction (MOR)

Given an input-output system (1), an associated reduced order model has the form:

$$\begin{aligned} \dot{x}_r(t) &= f_r(x_r(t), u(t)), \\ y_r(t) &= g_r(x_r(t), u(t)), \\ x_r(0) &= x_{r,0}, \end{aligned}$$

with a reduced state $x_r : \mathbb{R}_{\geq 0} \to \mathbb{R}^n$, $n \ll N$, a reduced vector field $f_r : \mathbb{R}^n \times \mathbb{R}^M \to \mathbb{R}^n$, a reduced output functional $g_r : \mathbb{R}^n \times \mathbb{R}^M \to \mathbb{R}^Q$ and an approximate output $y_r : \mathbb{R}_{\geq 0} \to \mathbb{R}^Q$, such that $\|y - y_r\| \ll 1$. Various methods exist to obtain such a reduced order model. In the scope of this work we will focus on a data-driven approach from the class of projection-based model order reduction methods.

2.1 Projection-Based MOR

A popular class of model reduction methods uses truncated projections to construct reduced order models. The aim in projection-based model order reduction is to find a set of projections to, and back from, a coordinate system in which its base vectors are ordered by importance in some sense, so that lesser relevant directions can be truncated. Practically, a projection-based reduced order model to (1) is given by:

$$\begin{aligned} \dot{x}_r(t) &= V_1 f(U_1 x_r(t), u(t)), \\ y_r(t) &= \quad g(U_1 x_r(t), u(t)), \\ x_r(0) &= V_1 x_0, \end{aligned}$$

with the truncated reconstructing projection $U_1 \in \mathbb{R}^{N \times n}$, and the truncated reducing projection $V_1 \in \mathbb{R}^{n \times N}$, which are bi-orthogonal: $V_1 U_1 = I_n$.

2.2 Cross-Gramian-Based MOR

To delineate the subsequent nonlinear model reduction approach, the underlying linear model reduction technique is briefly summarized. Given a square ($M = Q$), linear system:

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

with a linear vector field consisting of $A \in \mathbb{R}^{N\times N}$, $B \in \mathbb{R}^{N\times M}$, and a linear output functional, $C \in \mathbb{R}^{Q\times N}$, the cross Gramian matrix [2] is defined as:

$$W_X := \int_0^\infty \mathrm{e}^{At}\, BC\, \mathrm{e}^{At}\, \mathrm{d}t \in \mathbb{R}^{N\times N}.$$

Following the approximate balancing technique in [9], a singular value decomposition (SVD) of this cross Gramian,

$$W_X \overset{\text{SVD}}{=} UDV$$

yields the projections $U \in \mathbb{R}^{N\times N}$ and $V \in \mathbb{R}^{N\times N}$. Truncating $N - n$ columns of U and rows of V based on the magnitude of singular values $D_{ii} \in \mathbb{R}_{\geq 0}$ gives the truncated reconstructing projection $U_1 \in \mathbb{R}^{N\times n}$ and truncated reducing projection $V_1 \in \mathbb{R}^{n\times N}$. The latter may be obtained by truncating V (Petrov-Galerkin approach) or using $U_1^\intercal$ (Galerkin approach). For further details on balancing methods, see [1].

2.3 Empirical-Cross-Gramian-Based MOR

A cross Gramian matrix can also be computed for nonlinear systems in data-driven manner, which is motivated by the following representation of the linear cross Gramian,

$$W_X = \int_0^\infty (\mathrm{e}^{At}\, B)(\mathrm{e}^{A^\intercal t}\, C^\intercal)^\intercal\, \mathrm{d}t,$$

as a product of the primal and dual (adjoint) impulse responses. For nonlinear systems, an adjoint system is generally not readily available as for linear systems. Yet, a cross Gramian can be computed purely based on state and output trajectory data: the empirical cross Gramian [6], which in a simplified variant is given by:

$$\widehat{W}_X := \frac{1}{M}\sum_{m=1}^{M} \int_0^\infty \Psi^m(t)\, \mathrm{d}t \in \mathbb{R}^{N\times N}, \tag{2}$$

$$\Psi_{ij}^m(t) = (x_i^m(t) - \bar{x}_i^m)(y_m^j(t) - \bar{y}_m^j) \in \mathbb{R}.$$

The $x_i^m(t)$ symbolizes the i-th state component for a simulation with the m-th perturbed input component, while $y_m^j(t)$ symbolizes the m-th output component for a simulation with j-th perturbed initial state component, and $\bar{x}_i^m$, $\bar{y}_m^j$ are averages of the respective trajectory components. for further details see [4].

3 Memory-Economic Computation

A major drawback of the empirical Gramians in general and the empirical cross Gramian in particular, is their dense structure of full order N. In this section we describe a memory-economic algorithm to compute the SVD of the empirical cross Gramian without assembling the full order cross Gramian.

3.1 Memory-Economic Empirical-Cross-Gramian-Based

The definition of the empirical cross Gramian (2) can be computed column-wise [5]:

$$\begin{aligned} W_X &= \begin{pmatrix} \omega_{X,1} & \dots & \omega_{X,n} \end{pmatrix} \in \mathbb{R}^{N \times N}, \\ \omega_{X,j} &= \frac{1}{M} \sum_{m=1}^{M} \int_0^\infty \psi^{mj}(t) \, \mathrm{d}t \in \mathbb{R}^{N \times 1}, \\ \psi_i^{mj}(t) &= (x_i^m(t) - \bar{x}_i^m)(y_m^j(t) - \bar{y}_m^j) \in \mathbb{R}. \end{aligned}$$

Thus the empirical cross Gramian can be assembled in blocks of columns, without any exchange of data between the steps of computing the column blocks. Following, an algorithm is presented to compute the singular vectors from the partitioned empirical cross Gramian incrementally, so only one partition has to kept in memory.

3.2 Memory-Economic SVD

The column-wise partitioning of the empirical cross Gramian is now reused to obtain the full empirical cross Gramian's left singular vectors U_1 associated to the dominant singular values, via a proper orthogonal decomposition (POD). The matrix of singular vectors acts as the truncated reconstructing projection and its transpose as truncated reducing projection. In the scope of this work we restrict ourselves to Galerkin projections $V = U^\intercal$, but Petrov-Galerkin-type projections are computable in a similar manner.

3.2.1 Hierarchical Approximate POD

To obtain the left singular vectors of the empirical cross Gramian, given in a column-wise block partitioning, a method related to the SVD, the hierarchical approximate proper orthogonal decomposition (HAPOD) from [5] is utilized.

The HAPOD algorithms allows to compute the dominant left singular vectors U_1 of a given column-wise partitioned data-set, for example incrementally, such that the mean ℓ_2 projection error is bounded from above by $\|W_X - U_1 U_1^\intercal W_X\|_{\ell_2} < \varepsilon$. Given an upper bound ε and a partitioning $W_X = [\omega_1, \dots, \omega_S]$, with ω_s containing K_s columns, this "live HAPOD" computes as:

$$\begin{aligned} \hat{u}_0 &:= \{\}, \\ [\omega_s, \hat{u}_{s-1}] &\overset{\text{SVD}}{=} u_s d_s v_s \to \hat{u}_s := u_s \hat{d}_s, \quad \hat{d}_{s,ii} = \begin{cases} d_{s,ii} & d_{s,ii} < \varepsilon^2 K_s \sqrt{\frac{\sum_{j=1}^{s} K_j}{S}} \\ 0 & \text{else} \end{cases} \\ U_1 &:= \hat{u}_S \to V_1 = U_1^\intercal. \end{aligned}$$

The combination of the partitioned empirical cross Gramian with the live HAPOD allows a computation of a low-rank reducing Galerkin projection and hence a projection-based reduced order model in a memory economic manner, as the full empirical cross Gramian is never needed and the partitioned only requires communication to forward reduced base components.

4 Numerical Example

To illustrate this memory-economic model reduction technique combining the empirical cross Gramian with the HAPOD, a nonlinear hyperbolic network model is utilized,

$$\begin{aligned} \dot{x}(t) &= A \tanh(Kx(t)) + Bu(t), \\ y(t) &= Cx(t), \\ x(0) &= 0. \end{aligned}$$

Exemplary, a sparse but stable system matrix $A \in \mathbb{R}^{1024 \times 1024}$, a sparse random input matrix $B \in \mathbb{R}^{1024 \times 1}$, a random output matrix $C \in \mathbb{R}^{1 \times 1024}$, and a diagonal random gain matrix $K \in \mathbb{R}^{1024 \times 1024}$, $K_{ii} = \mathcal{U}_{[0,1]}$ is selected for this test.

This model is reduced using the conjoined methods of the empirical cross Gramian and the HAPOD on four different compute systems:

- Desktop Computer:

CPU AMD **A10**-7800 (64-bit Quad-Core x86-64) @ 3.9Ghz

SIMD AVX, FMA3 & FMA4

RAM 32GB DDR3-2133 (dual rank & dual channel)

- Thin Client:

CPU AMD **A12**-9800E (64-bit Quad-Core x86-64) @ 3.1Ghz

SIMD AVX2, FMA3 & FMA4

RAM 32GB DDR4-2133 (dual rank & dual channel)

- Single Board Computer 1:

CPU Allwinner **H3** (32-bit Quad-Core ARM Cortex A7) @ 0.8Ghz

SIMD NEON, VFP4

RAM 0.5GB DDR3-1600 (single rank & single channel)

- Single Board Computer 2:

CPU Allwinner **H5** (64-bit Quad-Core ARM Cortex A8a) @ 0.8Ghz

SIMD NEON, VFP4

RAM 0.5GB DDR3-1600 (single rank & single channel)

using `emgr` - empirical Gramian framework [3] in GNU Octave [10] with OpenBLAS [8] via FlexiBLAS [7].

In Figure 1 average power draw under load for each of the systems is depicted, as well as the computational time and consumed energy for the partitioned and unpartitioned (one partition of order 1024) empirical-cross-Gramian-based computation.

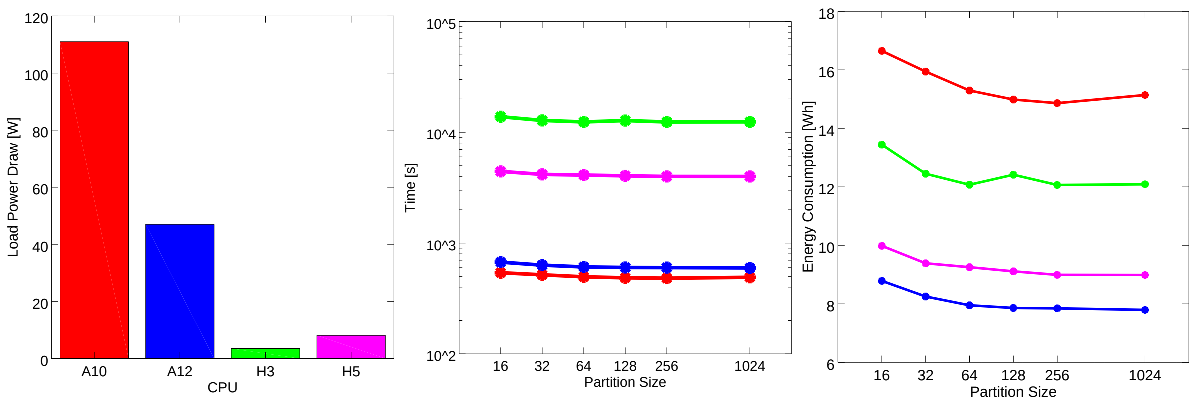

Figure 1: Comparison of power draw under load, computational time (offline phase) and energy consumption on different compute systems and partitionings.

The power draw during computational load among the four architectures ranges from the 111 W (A10), over 47 W (A12), 9.2 W (H5) to 3.5 W (H3). Even though the A12 requires 23% more time on average than the A10 to obtain the solution, the A12 consumes only about half the energy. The H3 requires expectedly significantly longer computational time but still consumes less energy than the A10, while the H5 halves the runtime of the H3 and is almost on par with the most efficent A12 in terms of energy consumption. It should be emphasized that the H3 is a 32-bit (ARM) architecture and additional factors, such as cooling have to be considered more carefully than on x86 platforms, hence this comparison is not completely fair.

The partitioning has a minor effect on computational time and energy consumption: Longer times and more energy are required for small partition sizes. For larger partition sizes a slight reduction in time and thus energy consumption can be observed. Overall, the combination of partitioned empirical cross Gramian and hierarchical approximate proper orthogonal decomposition enables model reduction in compute- or memory-limited environments such as single board computers at little to no additional cost.

Acknowledgements

This work is supported by the German Federal Ministry for Economic Affairs and Energy, in the joint project: “**MathEnergy** – Mathematical Key Technologies for Evolving Energy Grids”, sub-project: Model Order Reduction (Grant number: 0324019**B**).

References

[1] A. Antoulas, *Approximation of Large-Scale Dynamical Systems*, vol. 6 of Advances in Design and Control, SIAM Publications, Philadelphia, PA, 2005.

[2] K. V. Fernando and H. Nicholson, *On the structure of balanced and other principal representations of siso systems*, IEEE Trans. Autom. Control, 28 (1983), pp. 228–231.

[3] C. Himpe, *emgr – EMpirical GRamian framework (Version 5.0)*. `http://gramian.de`, 2016.

[4] ——, *emgr - the Empirical Gramian Framework*, arXiv e-prints 1611.00675, Cornell University, 2016. cs.MS.

[5] C. Himpe, T. Leibner, and S. Rave, *Hierarchical approximate proper orthogonal decomposition*, arXiv e-prints 1607.05210, Cornell University, 2016. math.NA.

[6] C. Himpe and M. Ohlberger, *Cross-Gramian based combined state and parameter reduction for large-scale control systems*, Mathematical Problems in Engineering, 2014 (2014), pp. 1–13.

[7] M. Köhler and J. Saak, *FlexiBLAS - A flexible BLAS library with runtime exchangeable backends*, Tech. Rep. 284, LAPACK Working Note, Jan. 2014.

[8] *OpenBLAS*. `http://www.openblas.net`.

[9] D. C. Sorensen and A. C. Antoulas, *The Sylvester equation and approximate balanced reduction*, Numer. Lin. Alg. Appl., 351–352 (2002), pp. 671–700.

[10] The Octave Developers, *GNU Octave*. `http://octave.org`.