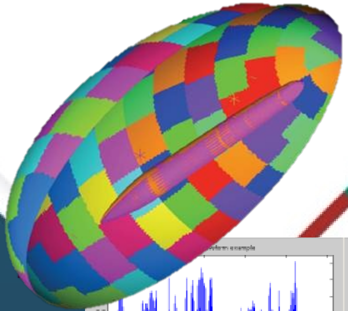
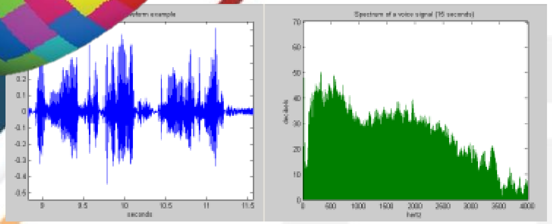
SAND2023-06919C

Sandia National Laboratories

Exceptional service in the national interest

Performance of Modal Random Vibration Calculations for Structural Dynamics on Heterogeneous Computing Architectures





Julia Plews and Nathan Crane

Computational Solid Mechanics & Structural Dynamics
Sandia National Laboratories

USNCCM17, Albuquerque, NM, USA

July 24-27, 2023

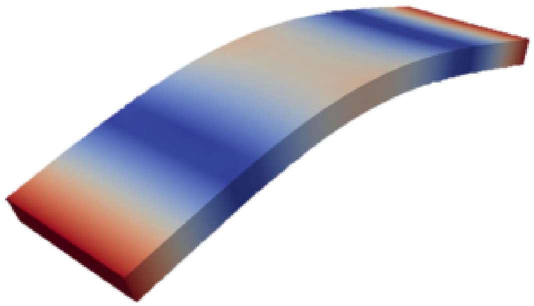
COMPSIM
STRUCTURAL DYNAMICS

Sandia National Laboratories is a multimission laboratory managed and operated by National Technology & Engineering Solutions of Sandia, LLC, a wholly owned subsidiary of Honeywell International Inc., for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-NA0003525.

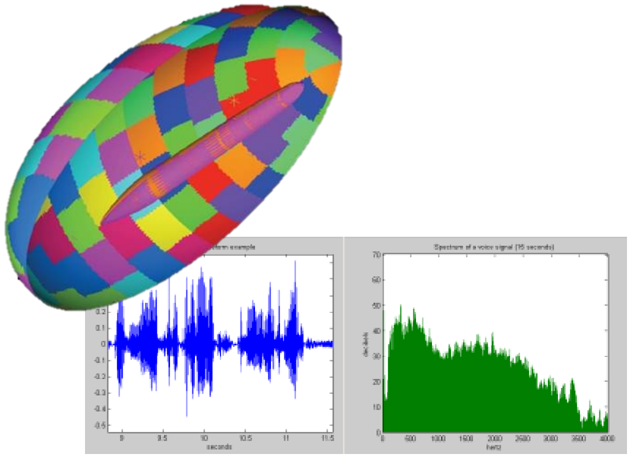
Sandia National Laboratories is a multimission laboratory managed and operated by National Technology and ...wned subsidiary of Honeywell International Inc. for the U.S. Department ...National Nuclear Security Administration under contract DE-NA0003525.

What is Modal Random Vibration?

Modal random vibration is a means to evaluate the statistical behavior of a structure in a steady-state vibration environment

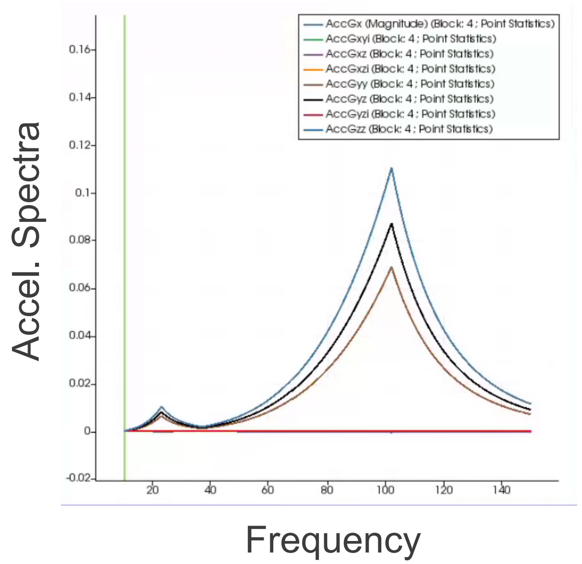


Natural frequency mode shapes



Statistical description of **input loads**

- Power spectra in frequency domain
- Correlation between input loads



Statistical **dynamic system behavior**

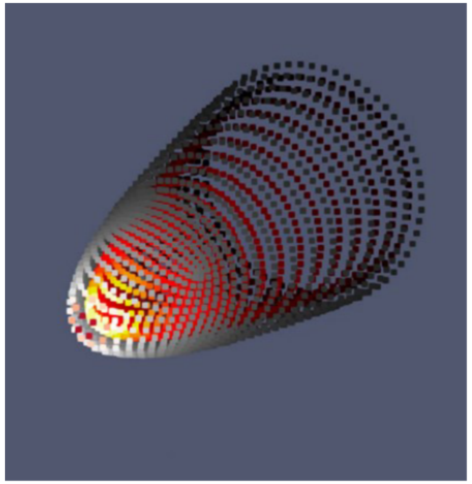
- Acceleration
- Displacement
- Stress

Image: https://en.wikipedia.org/wiki/Spectral_density

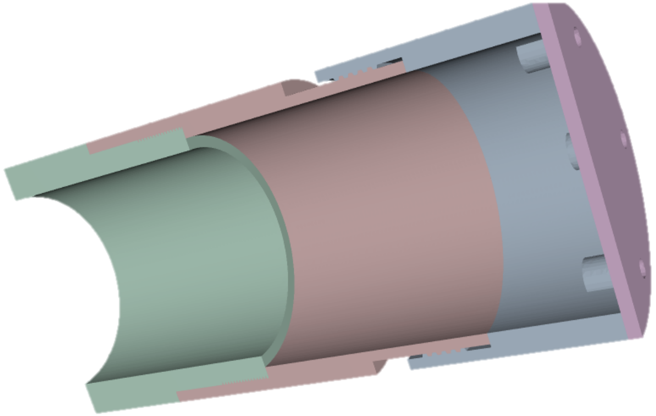
What is Modal Random Vibration?

An efficient alternative to performing large ensembles of transient structural analyses to understand response of structures under random excitation in environments such as

Transportation



Flight and atmospheric reentry



Test fixtures and experimental configurations

Typical probabilistic quantities of interest include

- margin to material yield,
- acceleration spectra near or inside a structural component,
- fatigue crack initiation and growth, etc.

Modal Random Vibration mathematics

$$S(\omega) = Z(\omega)_{\text{modes x loads}}\ E(\omega)_{\text{loads x loads}}\ Z^T(\omega)_{\text{loads x modes}}$$

$$A(\omega) = H^*(\omega)_{\text{modes}}\ S(\omega)_{\text{modes x modes}}\ H(\omega)_{\text{modes}}$$

ω = response frequency

A(ω) = response output (maybe complex-valued)

H(ω) = transfer function (complex-valued)

S(ω) = modal input power spectrum (complex-valued)

Z(ω) = modal excitation for load (real-valued)

E(ω) = input load magnitudes and correlations (complex-valued, input)

Typically engineers deal with up to thousands each of modes, loads, frequencies, and output locations, yielding possibly *trillions of computations* for a single simulation

An opportunity to utilize novel high-performance computing platforms to drastically reduce simulation turnaround times

Sierra Structural Dynamics simulations at Sandia

Sierra/SD (Structural Dynamics): massively parallel C++ finite element analysis code in the **Sierra** suite of tools

- In development for 25+ years
- Designed to run on the fastest supercomputers in the world featuring **CPU** and **GPU** acceleration
- Support national security missions

SD simulations are used to predict the behavior of a wide variety of systems, components, and experimental configurations

COMPSIM
SOLID MECHANICS

COMPSIM
THERMAL FLUIDS

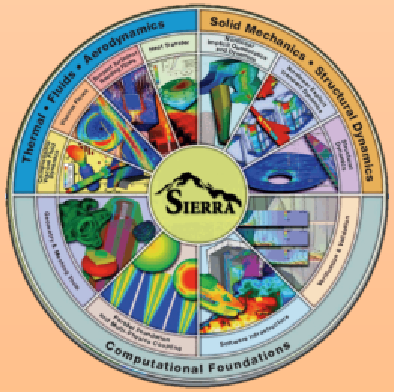
COMPSIM
STRUCTURAL DYNAMICS

Computational science and engineering research for success in large SD simulations

Computing Research

TRILINOS

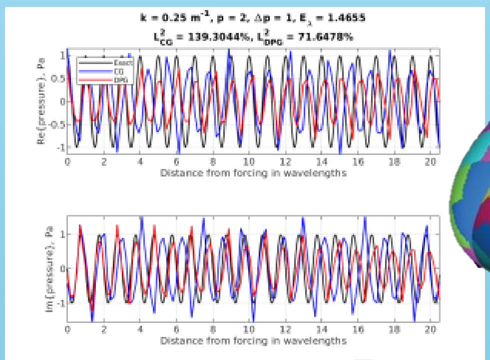
Computational Science Tools

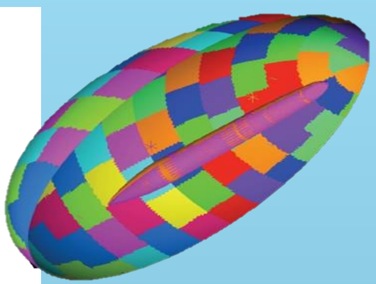


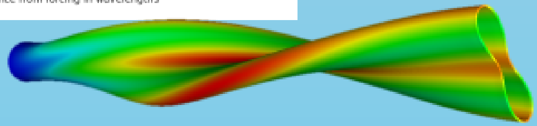
COMPSIM
STRUCTURAL DYNAMICS

Engineering Analysis Tool Development

Engineering Sciences







Analytical and Experimental Structural Dynamics

Porting Sierra/SD to GPUs: Kokkos performance portability library

Sandia-developed C++ Performance Portability Programming EcoSystem[1]

- **Write algorithms just once**, execute on any architecture

```
Kokkos::View<double*, Kokkos::CudaSpace>
    v("myData", 1024);
auto vHost = Kokkos::create_mirror(v);
... // do work on vHost
Kokkos::deep_copy(v, vHost);
```

- **Automatically move data** between memory spaces (e.g., CUDA to host and back)

```
struct MyKernel
{
  void operator()(const size_t i) const {
    printf("Hello, world!\n");
    myGpuData(i) = 2 * i;
  }

  Kokkos::View<double*> myGpuData;
};

Kokkos::parallel_for(1024, MyKernel());
```

- **Execute kernels in parallel** through a tailored template interface

```
KokkosBlas::gemm(Sal::ExecSpace(), "N", "N", 1.0, Z, Eij,
                 0.0, Z_times_E);
KokkosBlas::gemm(Sal::ExecSpace(), "N", "C", 1.0, ZtimesE,
                 Z, 0.0, Sij);
```

- Provide an agnostic interface to **vendor-supplied linear algebra** operations (e.g., cuBlas)

Kokkos on Github:
https://github.com/kokkos

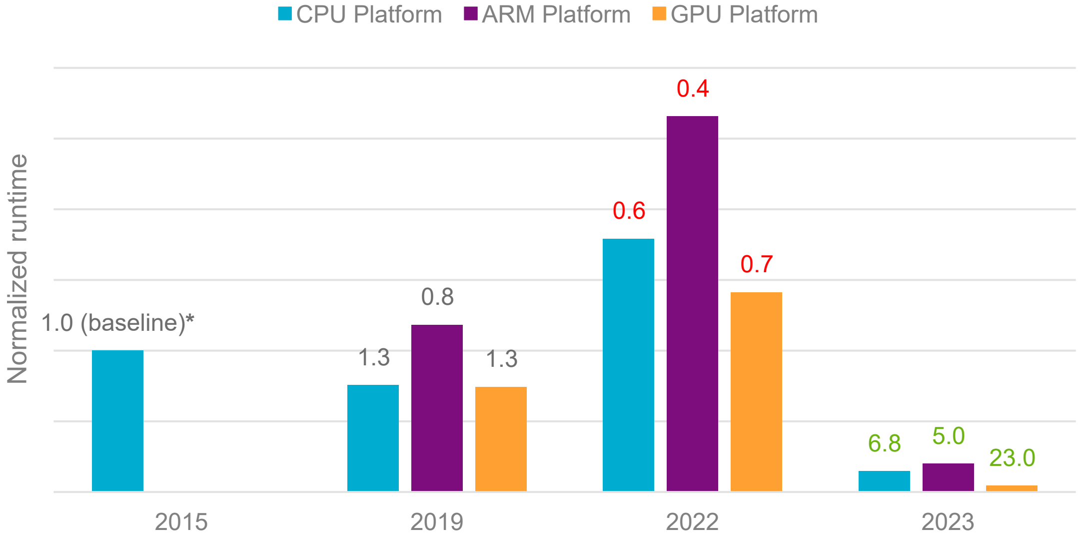
1. Kokkos: Enabling manycore performance portability through polymorphic memory access patterns
H. C. Edwards , C. R. Trott , D. Sunderland, Kokkos, J. Parallel and Distributed Computing, v.74 n.12, p.3202-3216, Dec. 2014

Historical performance of Modal Random Vibration in Sierra/SD

- **Exemplar model performance**: Medium-fidelity simulation of vibration of a system in atmospheric reentry environment

ARM platform: "Astra," 28 cores x 2 sockets Cavium Thunder-X2 per node

GPU platform: "Sierra," 44 cores IBM Power9 + 4x NVIDIA V100 per node



*(*data labels are speedups)*

- **2015**: Historical performance, CPU only
- **2019**: Initial Kokkos port to both ARM and GPU
- **2022**: First attempt at GPU optimization (lessons learned!)
- **2023**: Second, successful attempt at GPU optimization based on KokkosKernels

Historical performance of Modal Random Vibration in Sierra/SD

2019: Initial port to GPU and ARM based on Kokkos

Focused on a single performance hotspot: nodal output at each frequency

- Overall complexity O(frequencies * nodes * modes^2)
- GPU parallelization *only* over nodes; O(1000) concurrency insufficient to saturate GPU
- Modal input power spectrum (S) calculation *not* ported to GPU

```
for (int i = 0; i < numFreq; ++i) {
  Kokkos::parallel_for("NodalOutput", numNodes,
    KOKKOS_LAMBDA(int j) {
      outputVal(j) = Calc_H_S_Ht(H(j), S);
    });
}
```

Serialized calculation per node

No observable performance benefit from GPU use: small concurrency and too many serialized computations

Historical performance of Modal Random Vibration in Sierra/SD

2021: (Unsuccessful) first attempt at GPU optimization

GPU optimization focused on calculation of matrix S (input power spectrum):

- $S = Z_{\text{modes x loads}} E_{\text{loads x loads}} Z^{T}_{\text{loads x modes}}$
- Kokkos **"multi-dimensional range"** to consolidate 4 nested loops

```
Kokkos::MDRangePolicy<Kokkos::Rank<4>> range({0, 0, 0, 0}, {nModes, nModes, nLoads, nLoads});
Kokkos::parallel_for(
    "Scalc", range, KOKKOS_LAMBDA(int iMode, int jMode, int iLoad, int jLoad) {
      unsigned iModeIdx = modeIndices(iMode);
      unsigned jModeIdx = modeIndices(jMode);
      Kokkos::atomic_add(&S(iModeIdx, jModeIdx),
                         Z(iModeIdx, iLoad) * E(iLoad, jLoad) * Z(jModeIdx, jLoad));
    });
```

Accessing data at non-contiguous locations

$O(\text{modes}^2 * \text{loads}^2)$

Performance actually *degraded* due to atomics, frequent data access indirection, and high operator complexity

Historical performance of Modal Random Vibration in Sierra/SD

2023: (Successful) second attempt at GPU optimization

- $S = Z_{\text{modes x loads}} \, E_{\text{loads x loads}} \, Z^T_{\text{loads x modes}}$
- Step one: **remove modal data access indirection**

```
RemoveInactiveModes(S);
RemoveInactiveModes(Z);
Kokkos::MDRangePolicy<Kokkos::Rank<4>> range({0, 0, 0, 0}, {nModes, nModes, nLoads, nLoads});
Kokkos::parallel_for(
    "Scalc", range, KOKKOS_LAMBDA(int iMode, int jMode, int iLoad, int jLoad) {
      Kokkos::atomic_add(&S(iMode, jMode),
                         Z(iMode, iLoad) * E(iLoad, jLoad) * Z(jMode, jLoad));
    });
```

Data access is now contiguous, but complexity is still $O(\text{modes}^2 * \text{loads}^2)$

Historical performance of Modal Random Vibration in Sierra/SD

2023: (Successful) second attempt at GPU optimization

- $ZE = Z_{\text{modes x loads}} E_{\text{loads x loads}}$
- $S = ZE_{\text{modes x loads}} Z^T_{\text{loads x modes}}$
- Step two: **introduce temporary product, use performance-portable BLAS provided by KokkosKernels (optimized dense linear algebra)**

```
RemoveInactiveModes(S);
RemoveInactiveModes(Z);
KokkosBlas::gemm(ExecSpace(), "N", "N", 1.0, Z, E, 0.0, ZtimesE);
KokkosBlas::gemm(ExecSpace(), "N", "C", 1.0, ZtimesE, Z, 0.0, S);
```

Cleaner code, lower **O(modes^2 * loads)** complexity, and truly performance-portable

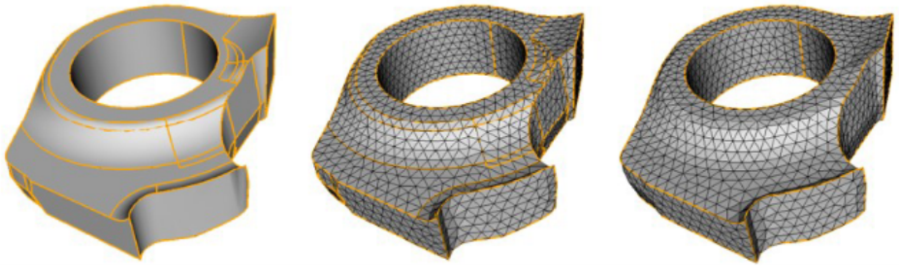
Summary and impacts on statistical analysis of structures at Sandia

	Spring 2022 baseline	Fall 2022	Spring 2023
CPU	--	~3x slower than Spring 22	~4x faster than Spring 22
GPU	on par with CPU	~1.2x slower than Spring 22 on CPU	~12x faster than Spring 22 on CPU

Performance changes in actual SD analyst model, relative to Spring 2022 version of Sierra/SD on CPU HPC platform

Through a combination of

- advances in GPU architectures,
- convenient code porting utilities,
- and algorithm optimizations,



modal random vibration computations in Sierra/SD that *used to take days* may **now take minutes**, enabling statistical analyses of structural designs in real time