

# High-Level Overview: High Performance Simulations of Neutron Stars in C++

---

Omokuyani Chibuzor Udiani

Michigan State University: FRIB  
June 10, 2025



Photo Credit: NASA's Goddard Space Flight Center

# Neutron Stars

- ❑ Composed of neutrons, protons, and electrons
- ❑ Gravity compresses them, but are held up by nuclear pressure
  - ❖ We need to understand this pressure
    - \* New states of matter
    - \* Origin of heavy elements
    - \* Probe physics beyond Standard Model

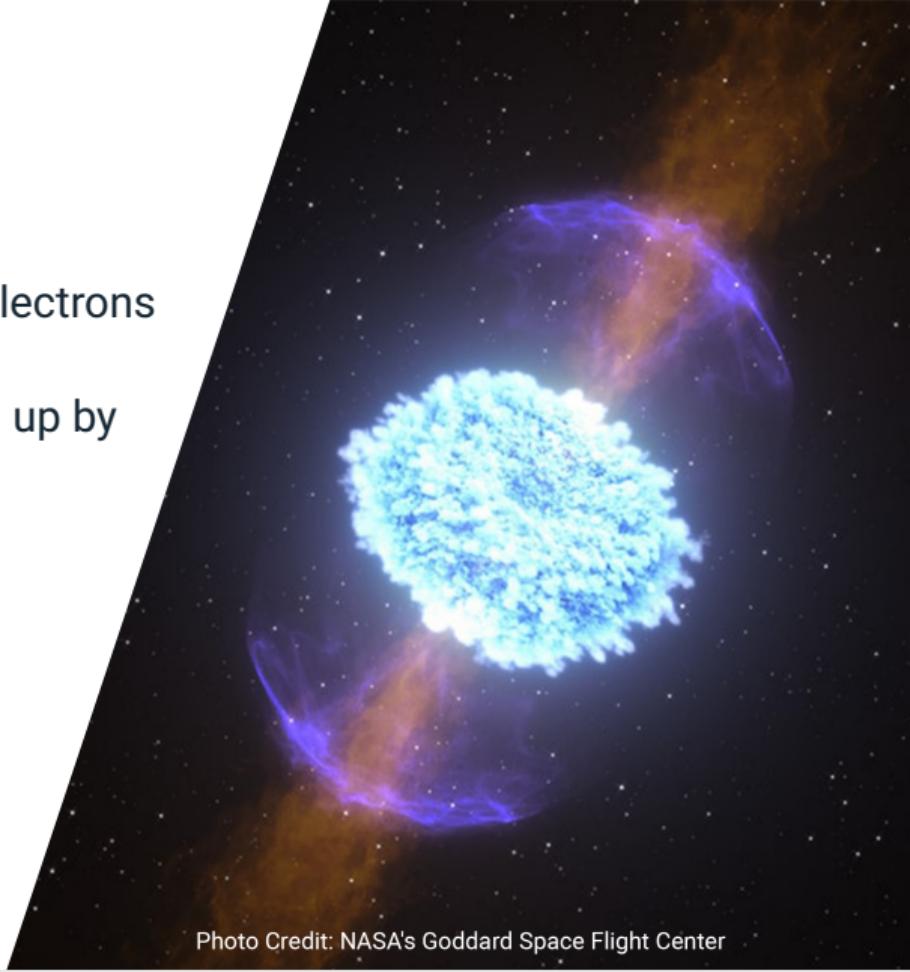
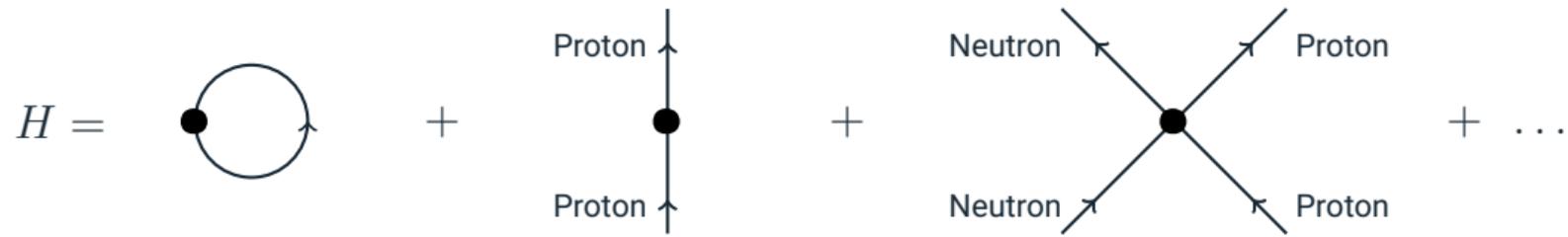
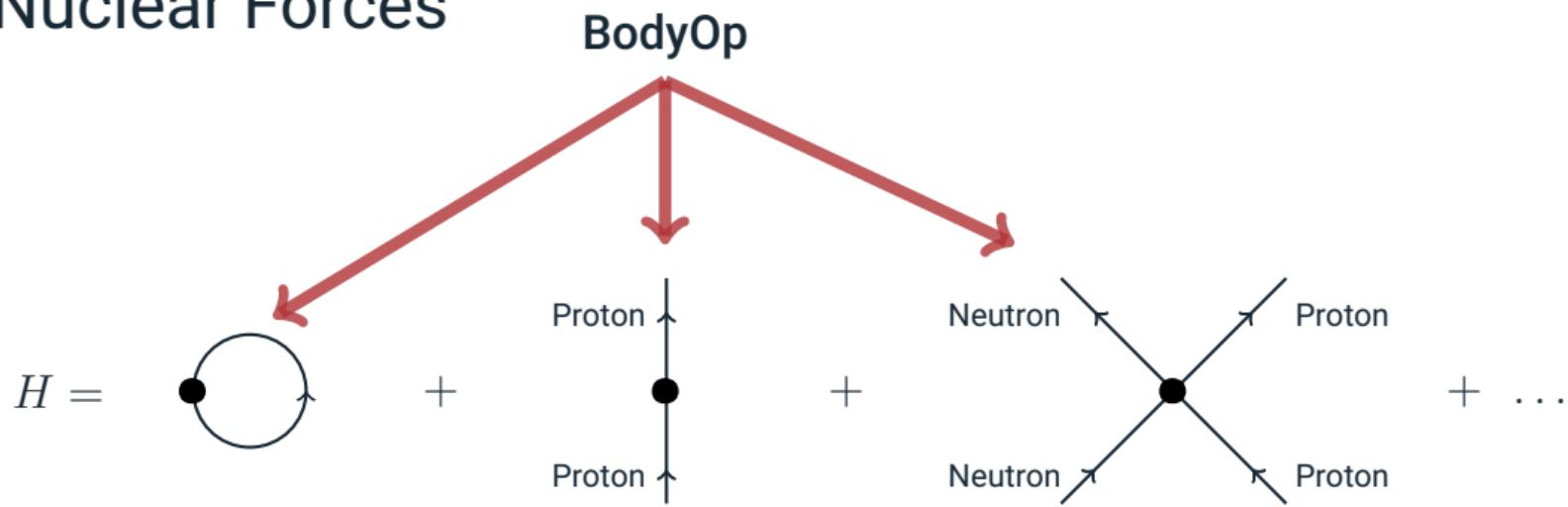


Photo Credit: NASA's Goddard Space Flight Center

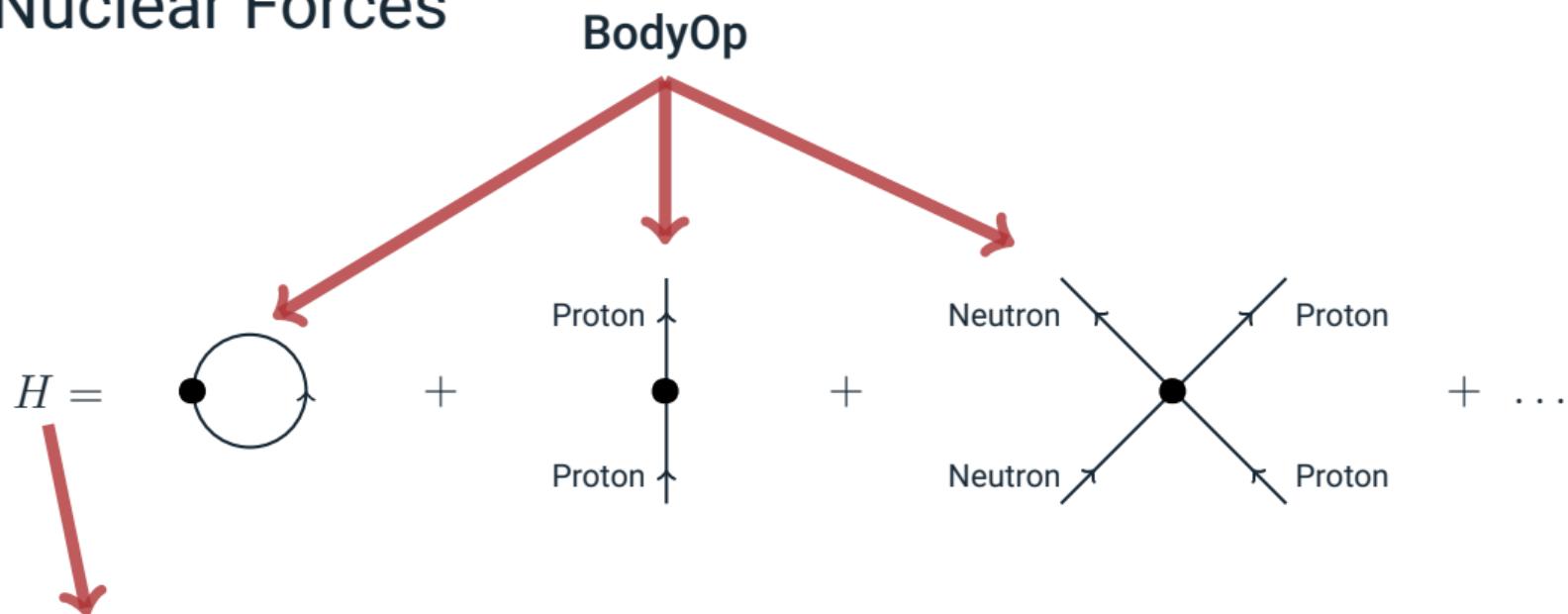
# Nuclear Forces

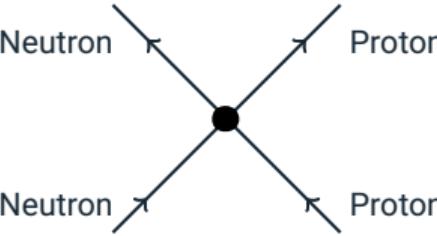


# Nuclear Forces



# Nuclear Forces



$H =$   +  +  + ...

A **BodyOp**  $\sim 80$  GB—size of a matrix with  $\sim 7e9$  **std::complex double** elements  
(We store  $\sim 10$  of these in RAM)

# In-Medium Similarity Renormalization Group

$$\frac{dH}{ds} = [\eta, H] \longrightarrow \text{Gives nuclear pressure!}$$



# Performance Critical Points

Copy assign ( $\sim 80$  GBs)

$$\frac{dH}{ds} = [\eta, H]$$


Tensor contractions  
(Compute heavy!)

# High Performance Optimizations

## For compute

- ✓ Manual profile like cavemen: we need real time profiling for calculations at scale
- ✓ Use shared (CUDA and OpenMP) and distributed parallelism (MPI)
- ✓ Write/rewrite algorithms to optimize cache utilization
- ✓ Precompute data and reuse operations
- ✓ Use approximations when worthwhile
- ✓ Exploit symmetries of underlying system
- ✓ Accelerate convergence using novel physics and data-driven insights

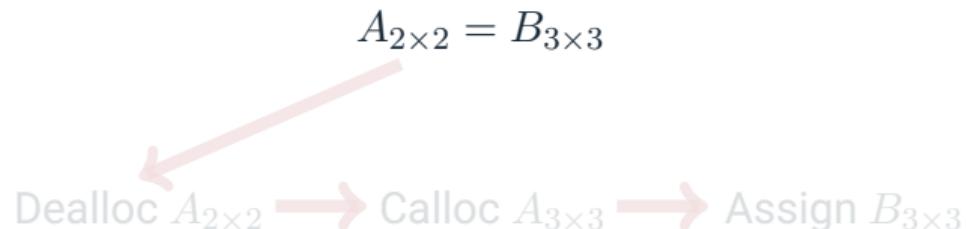
# High Performance Optimizations

## Memory management

- ✓ Custom destructors, move and copy constructors, move and copy assigners
- ✓ ABodyOp resources gatekept (accessible with performant getters)
- ✓ All expensive objects are allocated once at program start, and cleaned up upon termination
- ✓ Profiled with Valgrind and Cachegrind

# Why Custom Resource Management?

Default copy assignment is designed to handle:

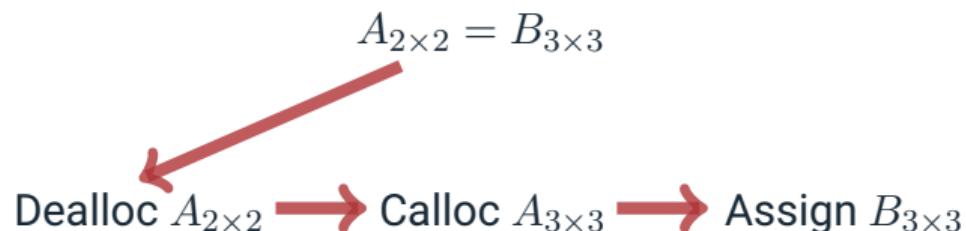


~1K copy assignments invoked throughout our program's lifetime (worst case)

We check if  $\text{shape}(A) = \text{shape}(B)$  (true in production) before dealloc + realloc, bypassing at least 1K kernel calls each worth ~80 GB—performance killer!

# Why Custom Resource Management?

Default copy assignment is designed to handle:

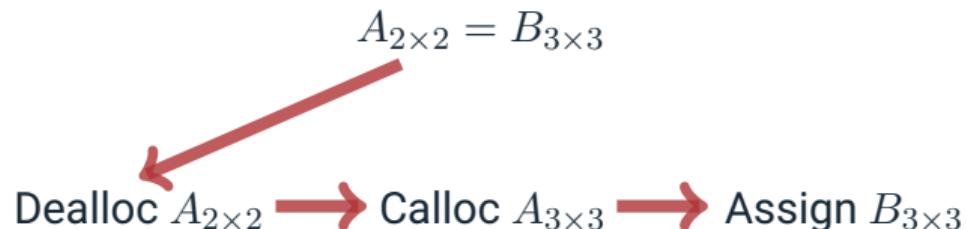


~1K copy assignments invoked throughout our program's lifetime (worst case)

We check if  $\text{shape}(A) = \text{shape}(B)$  (true in production) before dealloc + realloc, bypassing at least 1K kernel calls each worth ~80 GB—performance killer!

# Why Custom Resource Management?

Default copy assignment is designed to handle:



~1K copy assignments invoked throughout our program's lifetime (worst case)

We check if  $\text{shape}(A) = \text{shape}(B)$  (true in production) before dealloc + realloc, bypassing at least 1K kernel calls each worth ~80 GB—performance killer!