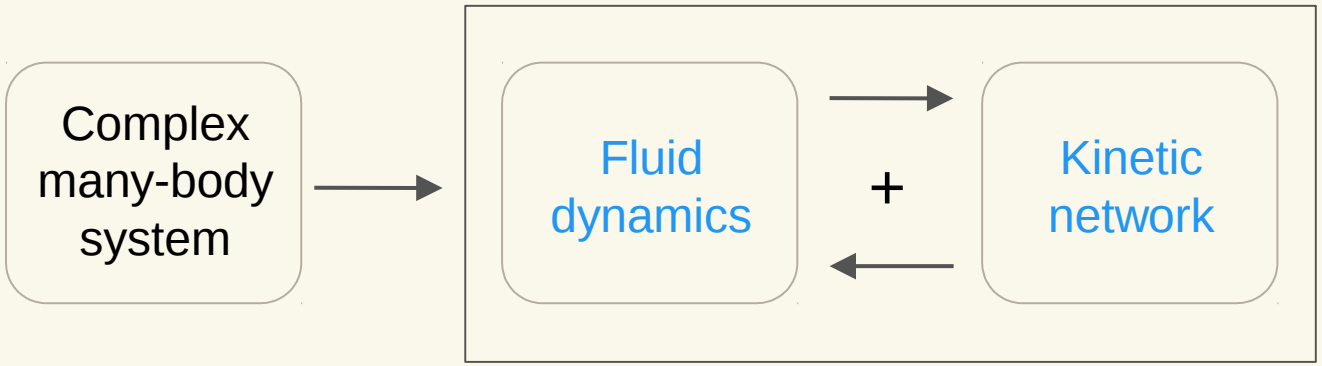
Fast, Scalable, Explicit Integration for a New Generation of Computational Astrophysics Applications

Mike Guidry

Department of Physics and Astronomy, University of Tennessee
Physics Division and Computer Science and Mathematics Division,
Oak Ridge National Laboratory

Fluid Dynamics Plus Kinetics Approximation

Many physical systems can be modeled by a fluid dynamics plus kinetics approximation.



Ordinary Differential Equations of a Typical Kinetic Network

Flux increasing y_i

Flux decreasing y_i

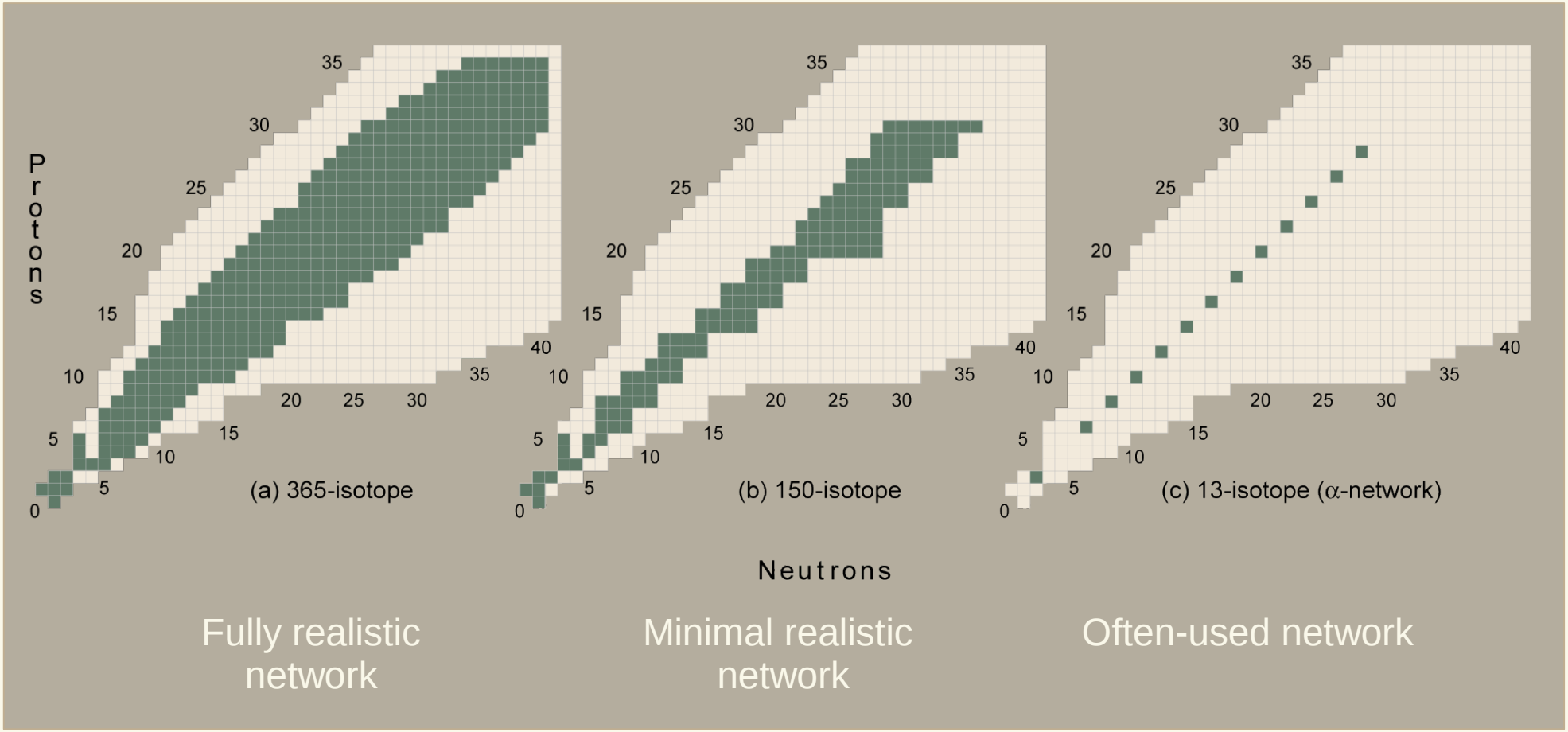
$$\frac{dy_i}{dt} = F_i^+ - F_i^-$$

$$= (f_1^+ + f_2^+ + \ldots)_i - (f_1^- + f_2^- + \ldots)_i$$

$$= (f_1^+ - f_1^-)_i + (f_2^+ - f_2^-)_i + \ldots$$

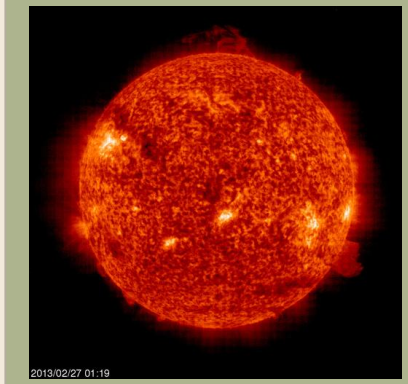
$$= \sum_j (f_j^+ - f_j^-)_i$$

Example: Realistic Thermonuclear Networks for Type Ia Supernova Problem



The Nature of Stiff Equations

- Stiff systems have two or more numerical timescales *differing by many orders of magnitude.*
- One quantitative measure of stiffness is the *ratio of the fastest and slowest timescales* in the system.
- Most physical systems involve processes operating on very different timescales, so *realistic problems tend to be at least moderately stiff.*
- Astrophysical thermonuclear networks, are often *extremely stiff,* with timescales differing by *10-20 orders of magnitude.*



Integrating Stiff Equations Numerically

Explicit numerical integration:

To advance the solution from time t_n to t_{n+1}, only information already available at t_n is required.

Implicit numerical integration:

To advance the solution from time t_n to t_{n+1}, information at the new point t_{n+1} is required, implying an *iterative solution*.

Thus, for numerical integration

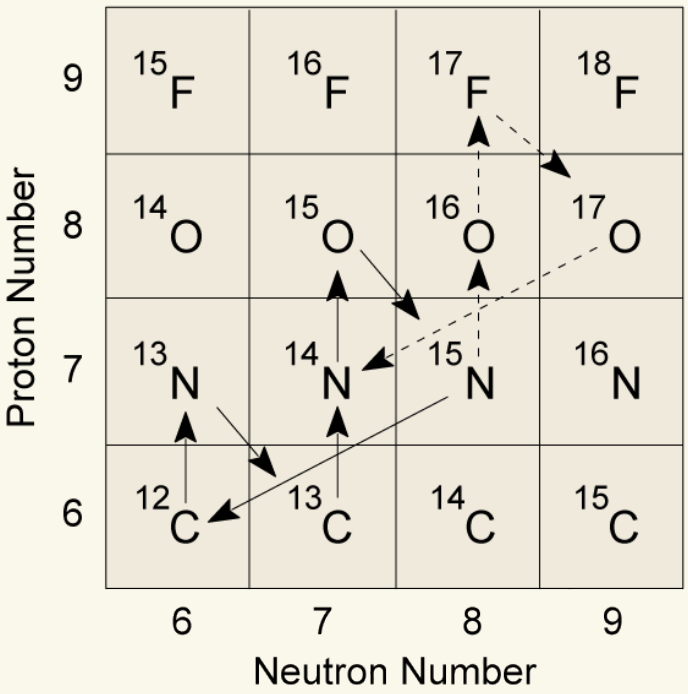
- Explicit methods are *inherently simple, but unstable for large timesteps.*
- Implicit methods are *inherently complicated, but stable for large timesteps.*

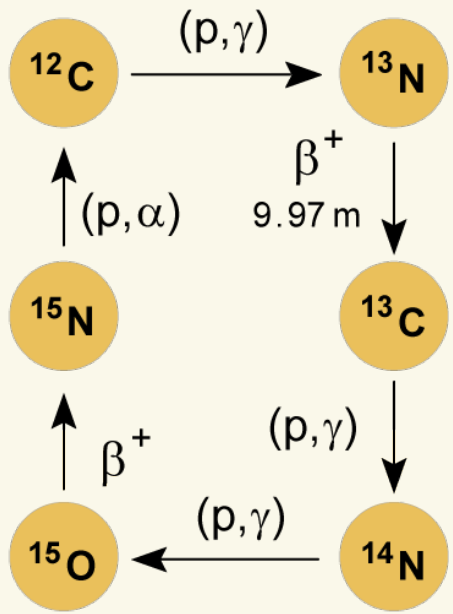
Maximum Stable Explicit Timestep

- In stiff systems the *maximum stable explicit timestep* is set by *the fastest timescales in the system*.

- Thus, for stiff systems *explicit integration is impractical* because *the maximum stable timestep is (much) too short to be useful.*

Example: CNO cycle:





- Fastest rates are beta decays with ~100 s lifetimes.
- Evolution to H depletion may take 1 billion yrs (~10^{16}s)
- Explicit codes could require 10^{14} steps, but implicit can do the job in ~10^{2} steps.

Methods to Integrate Stiff Equations

- There are two general approaches that we might use to deal with stiffness.

  The Traditional Way: Integrate equations *implicitly,* which is stable but requires an iterative solution with matrix inversions at each step (*expensive* for large networks).

  A New Way: Replace equations with some that are more stable and *integrate them explicitly.*

- If we could stabilize explicit integration we could do each timestep more quickly in large networks.

Fundamental Sources of Stiffness

Negative populations

$$\frac{dy_i}{dt} = F_i^+ - F_i^-$$

Macroscopic equilibration

$$= (f_1^+ + f_2^+ + \ldots)_i - (f_1^- + f_2^- + \ldots)_i$$

$$= (f_1^+ - f_1^-)_i + (f_2^+ - f_2^-)_i + \ldots = \sum_j (f_j^+ - f_j^-)_i$$

Microscopic equilibration

$A \rightleftharpoons B$

The key to stable explicit integration is to understand the 3 basic sources of stiffness for a typical reaction network:

- *Negative populations,*
- *Macroscopic equilibration*
- *Microscopic equilibration.*

Emergent Timescales

Emergent properties: Properties of a many-body system that are not present in the non-interacting system and *emerge* because of interactions.

Macroscopic and microscopic equilibrium timescales are emergent: they do not exist in the non-interacting network.

$$\frac{dy_i}{dt} = F_i^+ - F_i^-$$
$$= (f_1^+ + f_2^+ + \ldots)_i - (f_1^- + f_2^- + \ldots)_i$$
$$= (f_1^+ - f_1^-)_i + (f_2^+ - f_2^-)_i + \ldots = \sum_j (f_j^+ - f_j^-)_i$$

Curing Explicit Methods of Their Ills

The clear conceptual distinction among

1. Negative populations
2. Macroscopic equilibrium
3. Microscopic equilibrium

as sources of stiffness is crucial because *the cures are fundamentally different* for these instabilities:

- The first two instabilities are handled well by the *asymptotic (Asy)* or *quasi-steady-state (QSS)* algorithms.
- The microscopic equilibration instability is not fixed by asymptotic or QSS algorithms. It requires a new *partial equilibrium (PE)* algorithm.

Explicit and Implicit Timestep Speeds

We estimate a rough speedup factor for each timestep as $F \sim 1 / (1-f)$ where f is the *fraction of integration time spent in matrix operations*.

With these assumptions we then estimate

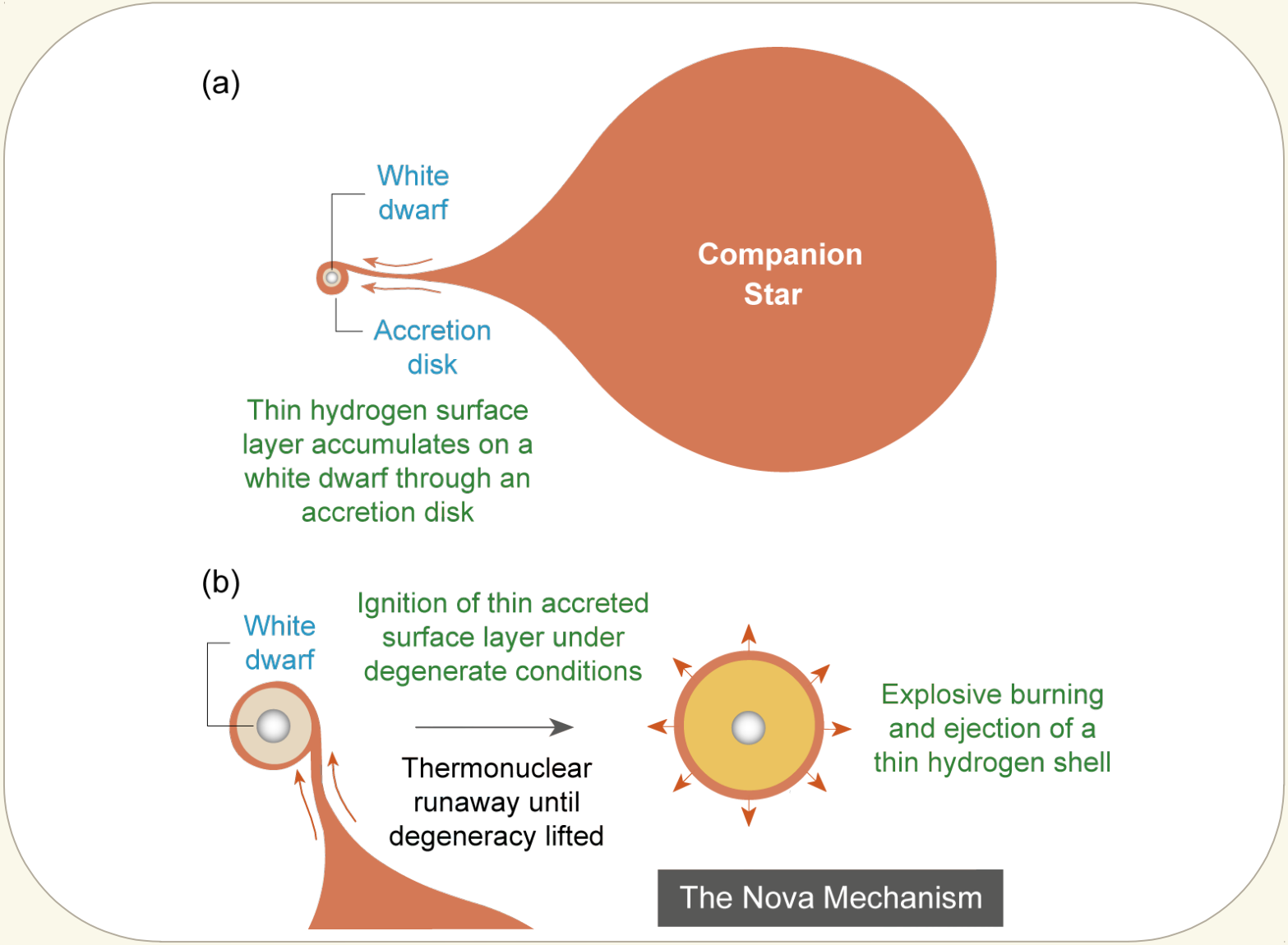
$$\frac{\text{Speed explicit}}{\text{Speed implicit}} \sim F \times \left(\frac{\text{Implicit timesteps}}{\text{Explicit timesteps}} \right)$$

This is a useful basis for comparisons.

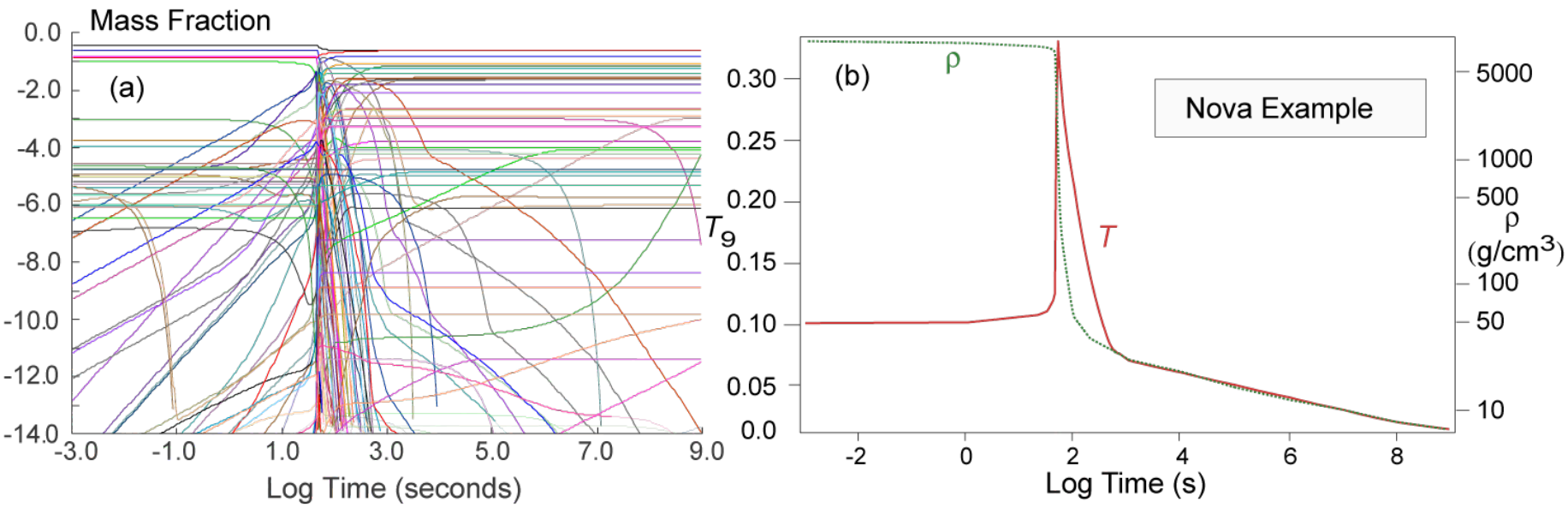
Network	Species	F
PP	7	1.5
α	16	3
Nova	134	~5
150-isotope	150	~ 5-10
365-isotope	365	~ 10-15

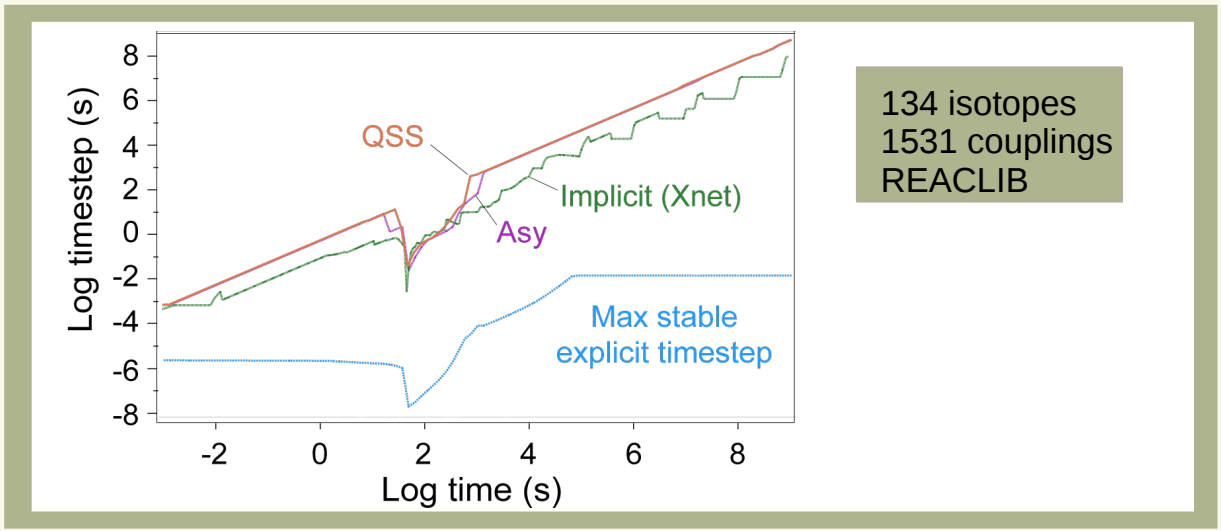
Factors *F* estimated from tests using the implicit backward-Euler code Xnet with several dense and sparse solvers.

Example: Simulating Nova Explosions



Example: Explicit Integration for a Nova Simulation

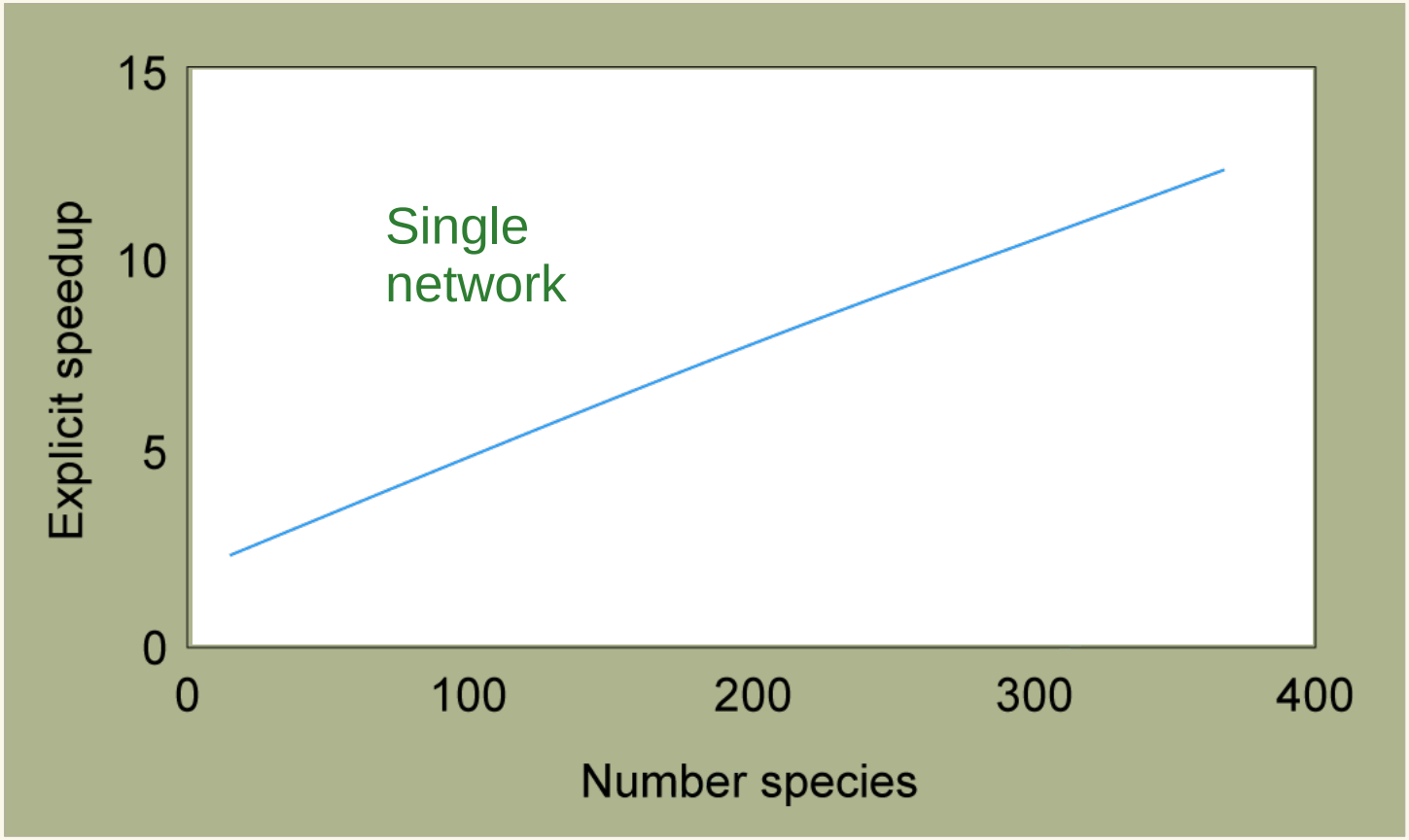




134 isotopes
1531 couplings
REACLIB

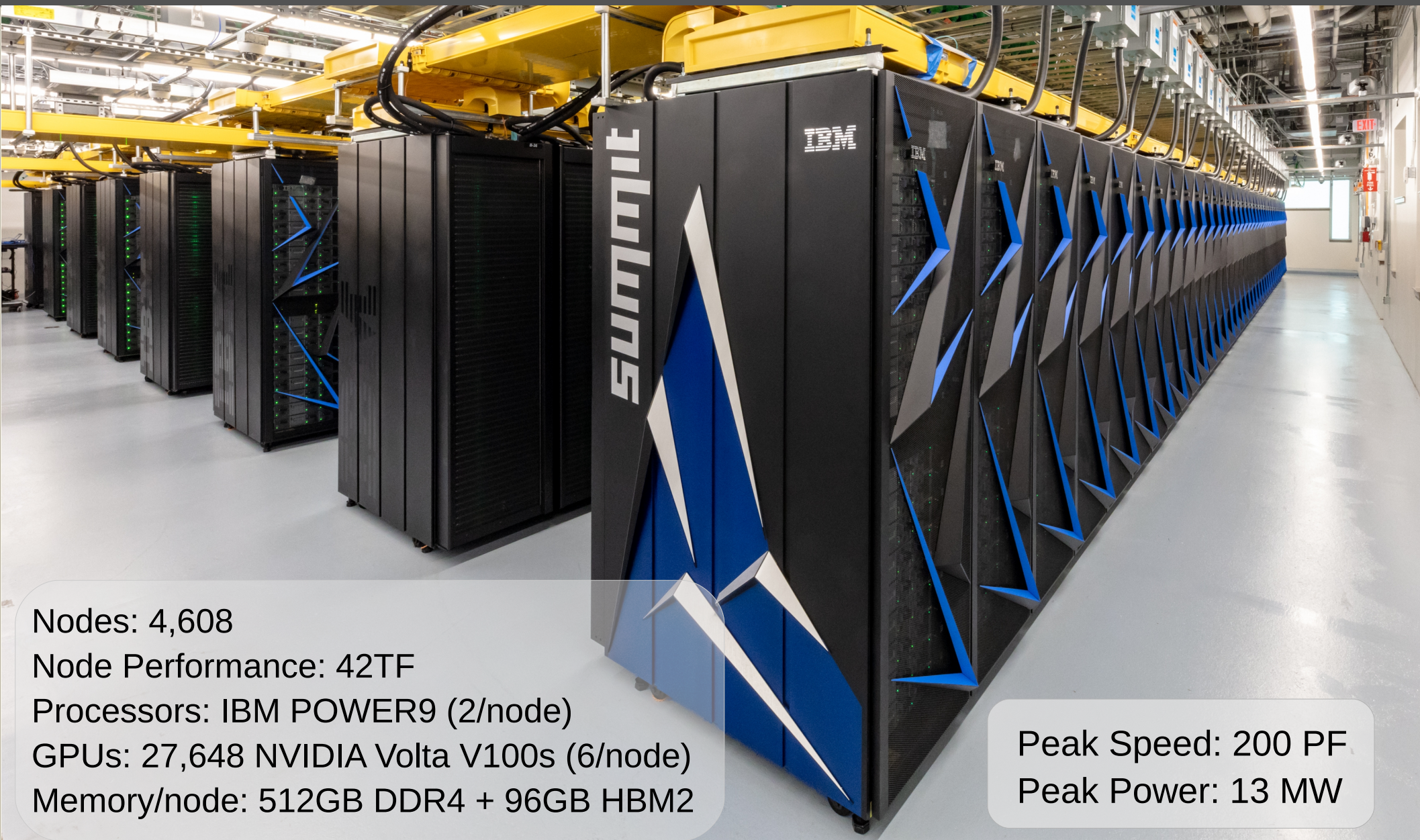
Method	Steps	Speed
Implicit	1332	1
Asy	935	10
QSS	777	12

Summary of Results: Explicit vs Implicit Speedup for a Single Network



Thus our new algorithms can give a speed increase of about an order of magnitude for networks with several hundred species. Now let us consider the *role of modern hardware in this problem.*

Computing Power for Modern Scientific Applications



Nodes: 4,608
Node Performance: 42TF
Processors: IBM POWER9 (2/node)
GPUs: 27,648 NVIDIA Volta V100s (6/node)
Memory/node: 512GB DDR4 + 96GB HBM2

Peak Speed: 200 PF
Peak Power: 13 MW

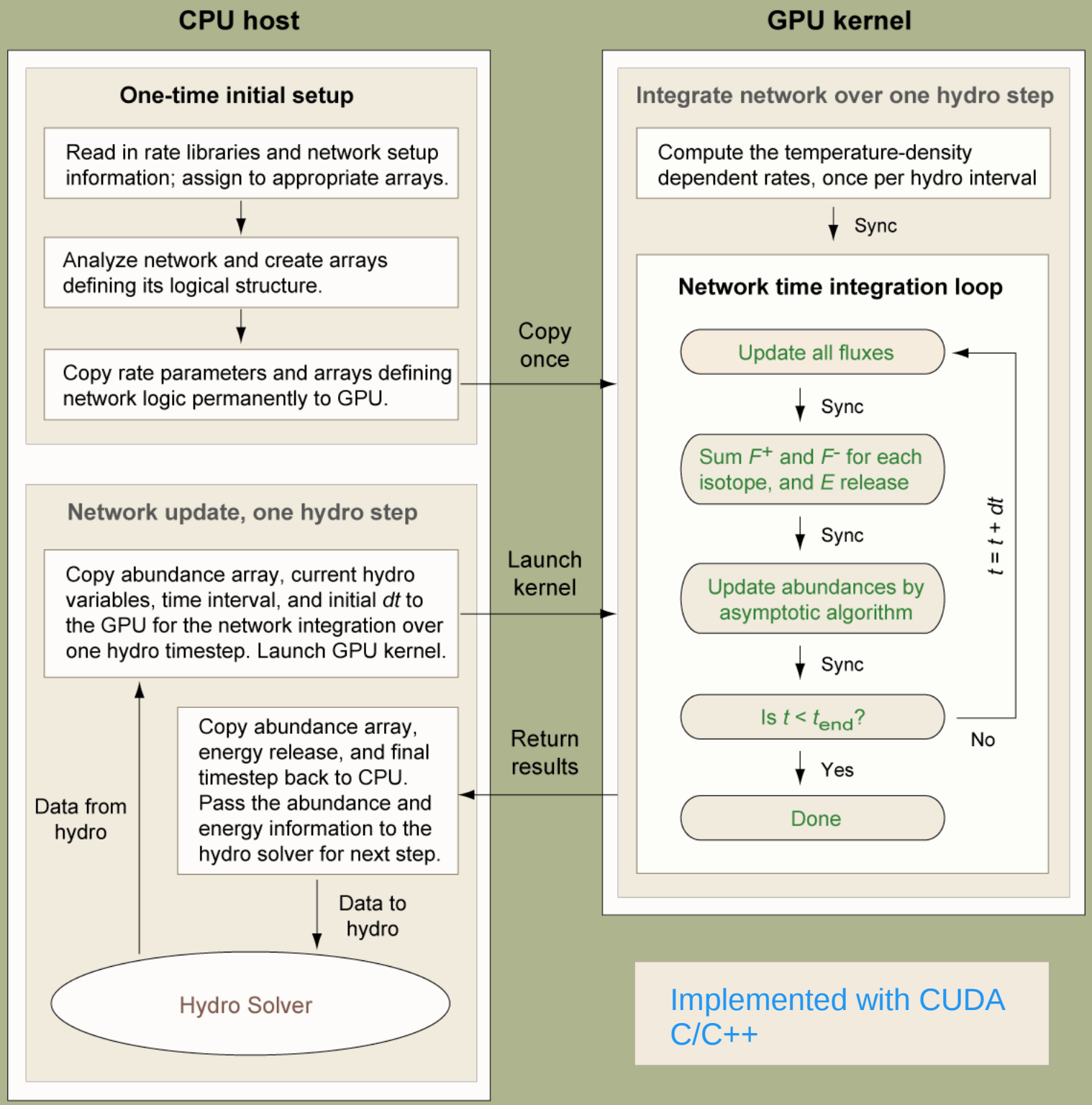
Graphical Processing Units (GPUs) in High-Performance Computing

- Much of the speed for modern supercomputers like Summit comes from graphical processing units (GPUs) on each compute node.

- These are
  - *Massively parallel* but with
  - *many lightweight threads* and
  - *limited fast memory* for each compute node.

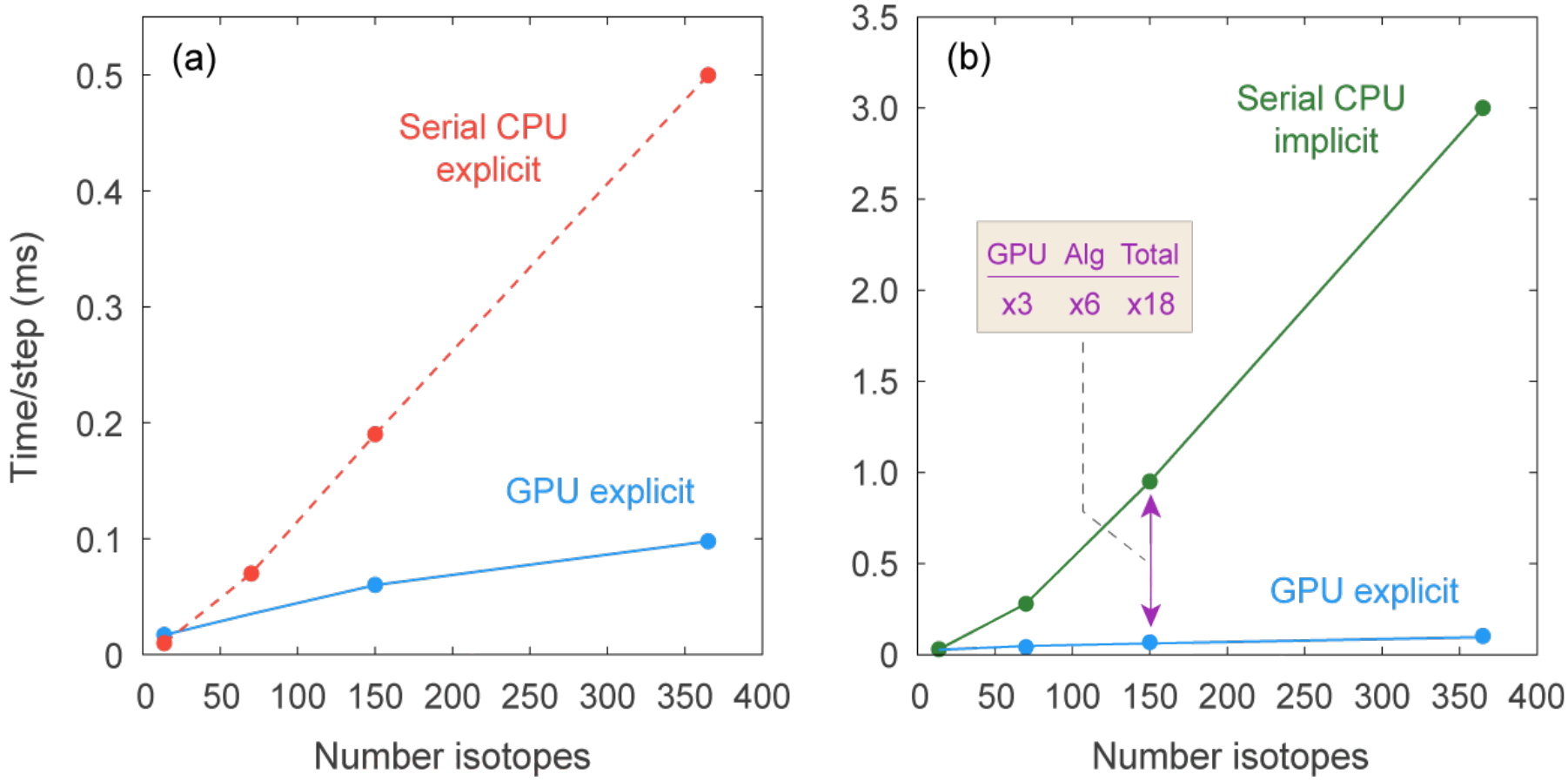
- Thus, such computers can be leveraged if an algorithm can compute effectively with these advantages and limitations.

GPU Acceleration for the Network



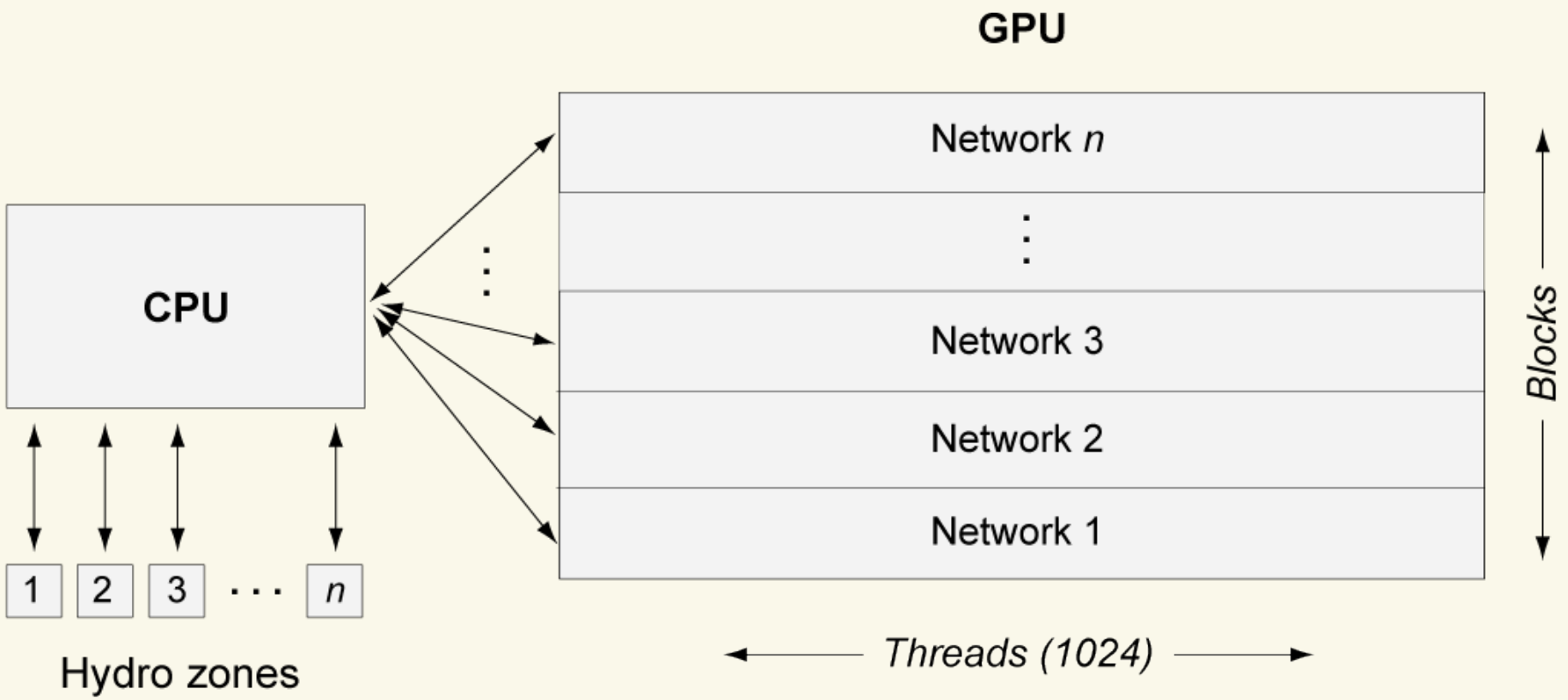
Implemented with CUDA C/C++

Scaling with Network Size for a Single Kinetic Network



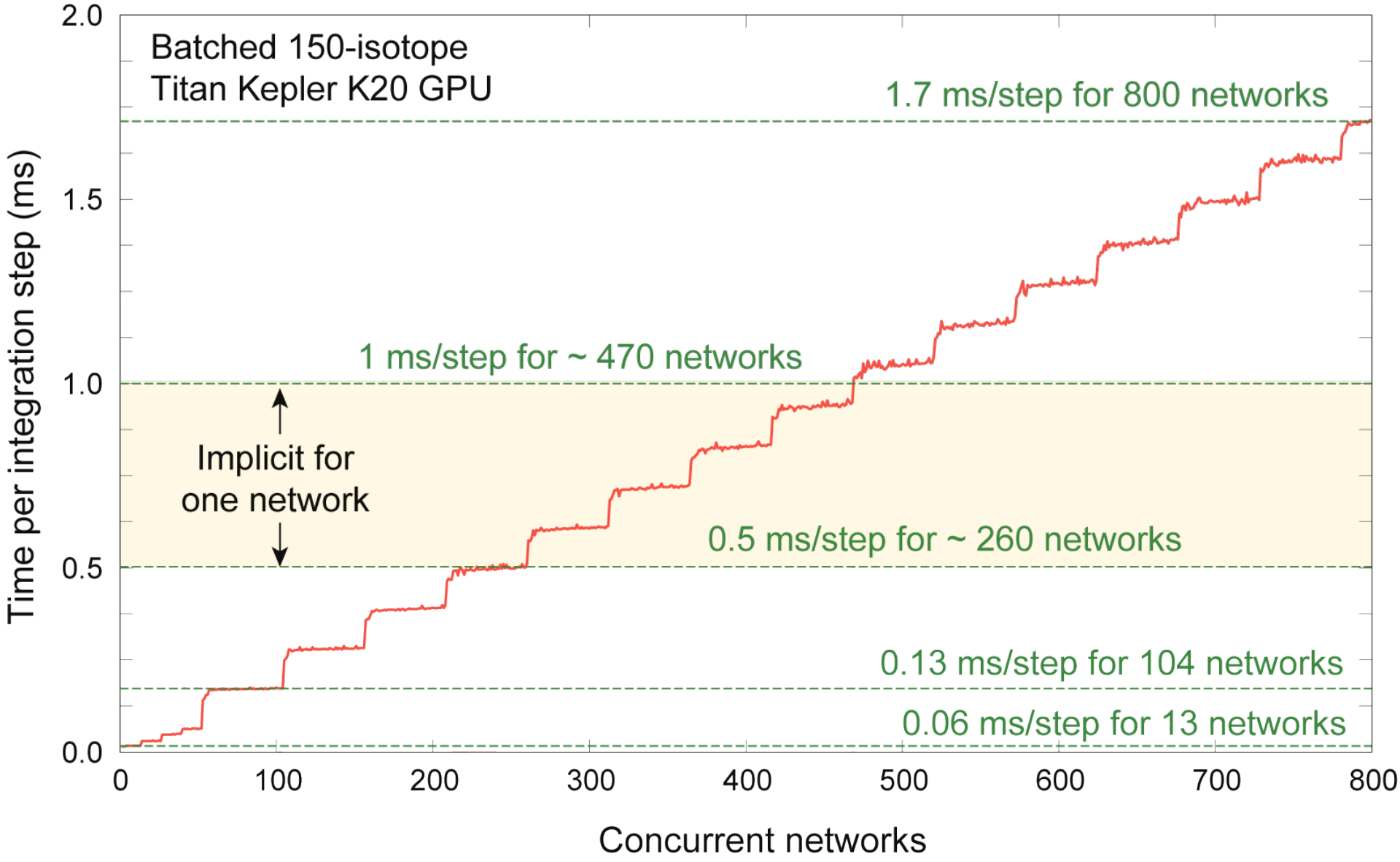
This is impressive speedup but *a single network utilizes only a small fraction of available GPU threads*. Greater efficiency requires that we *give the GPU more work.*

Stacking Multiple Networks on a Graphical Processing Unit (GPU)



It may be possible to run many such networks faster than it is now possible to run one such network.

Timing: Concurrent Network Launches



Explicit Methods for Neutrino Transport in Supernovae and Neutron Star Mergers

Consider electron neutrinos with the energy divided into $N_\varepsilon = 40$ energy bins. At some level of approximation, for the number density of bin i,

$$\frac{dN_i}{dt} = \underbrace{\sum_{k=1}^{N_\varepsilon} R_{ik}^{\text{in}} N_k}_{\text{flux in}} - \underbrace{N_i \sum_{k=1}^{N_\varepsilon} R_{ik}^{\text{out}}}_{\text{flux out}}$$

where R_{ik}^{in} and R_{ik}^{out} are matrices describing the rates populating (in) and depopulating (out) the bin.

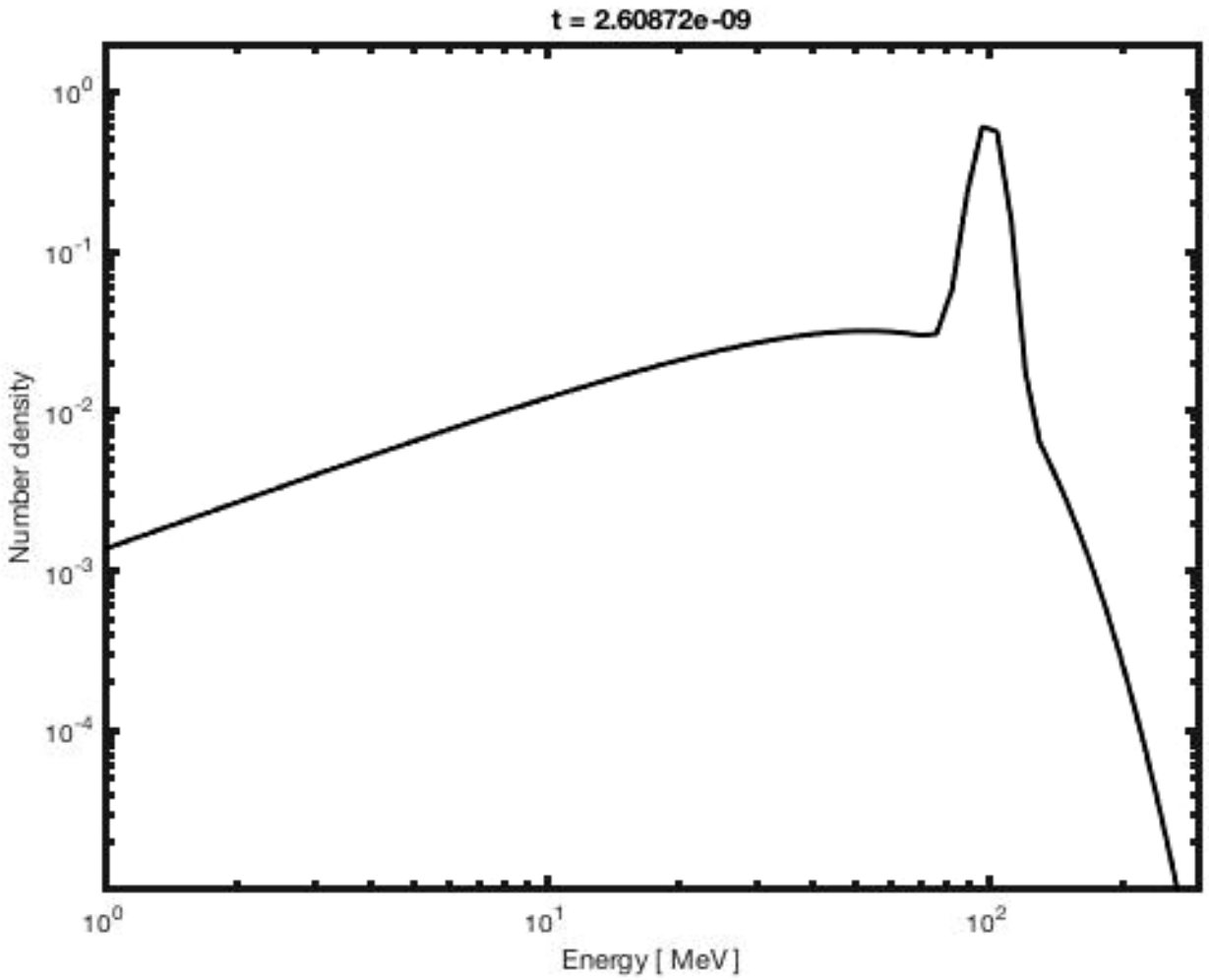
Explicit Methods for Neutrino Transport in Supernovae and Neutron Star Mergers

Thus, we have a set of ordinary differential equations that are *mathematically analogous to the thermonuclear network equations* discussed earlier.

$$\frac{dN_i}{dt} = \underbrace{\sum_{k=1}^{N_\varepsilon} R_{ik}^{\text{in}} N_k}_{\text{flux in}} - \underbrace{N_i \sum_{k=1}^{N_\varepsilon} R_{ik}^{\text{out}}}_{\text{flux out}}$$

Therefore, we can apply the algebraically-stabilized explicit methods that were introduced earlier.

Approach to Equilibrium for a Neutrino Distribution with Neutrino-Electron Scattering

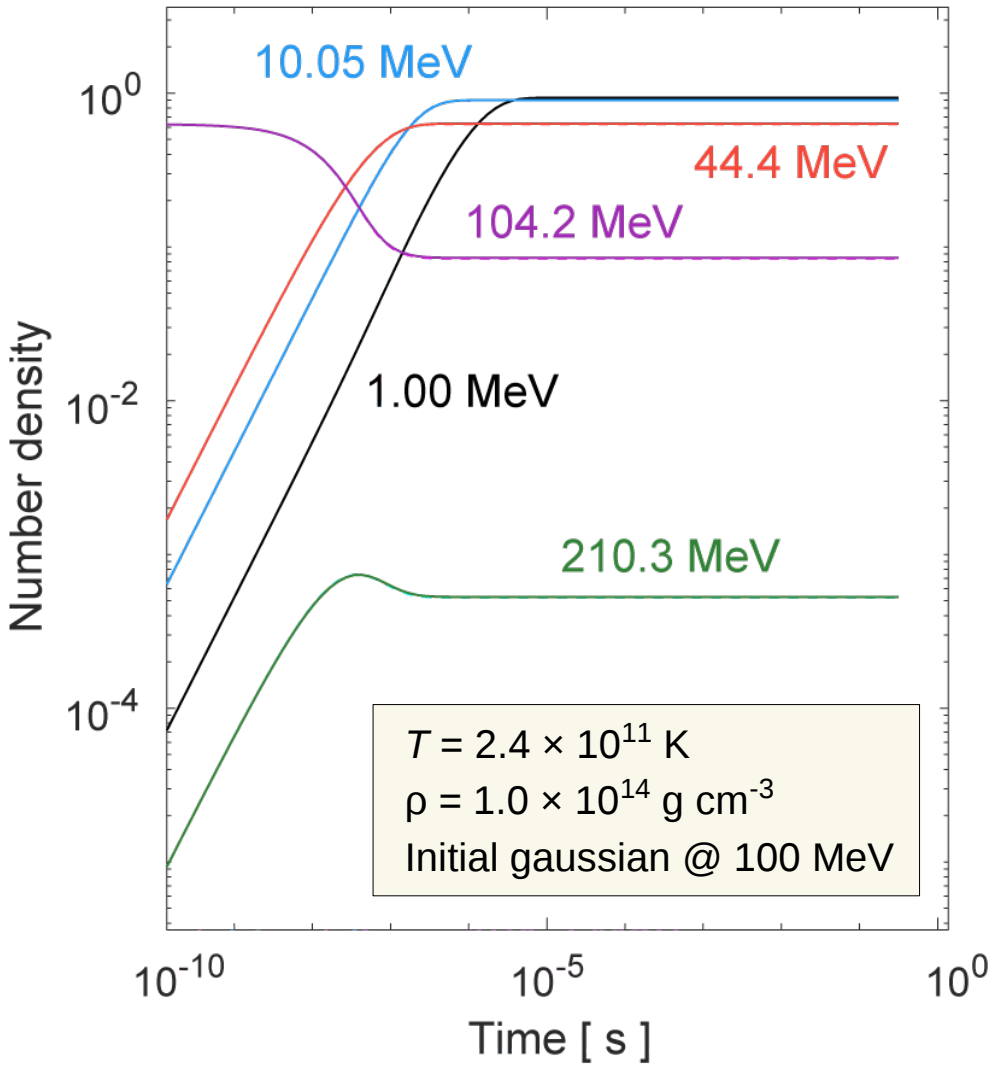


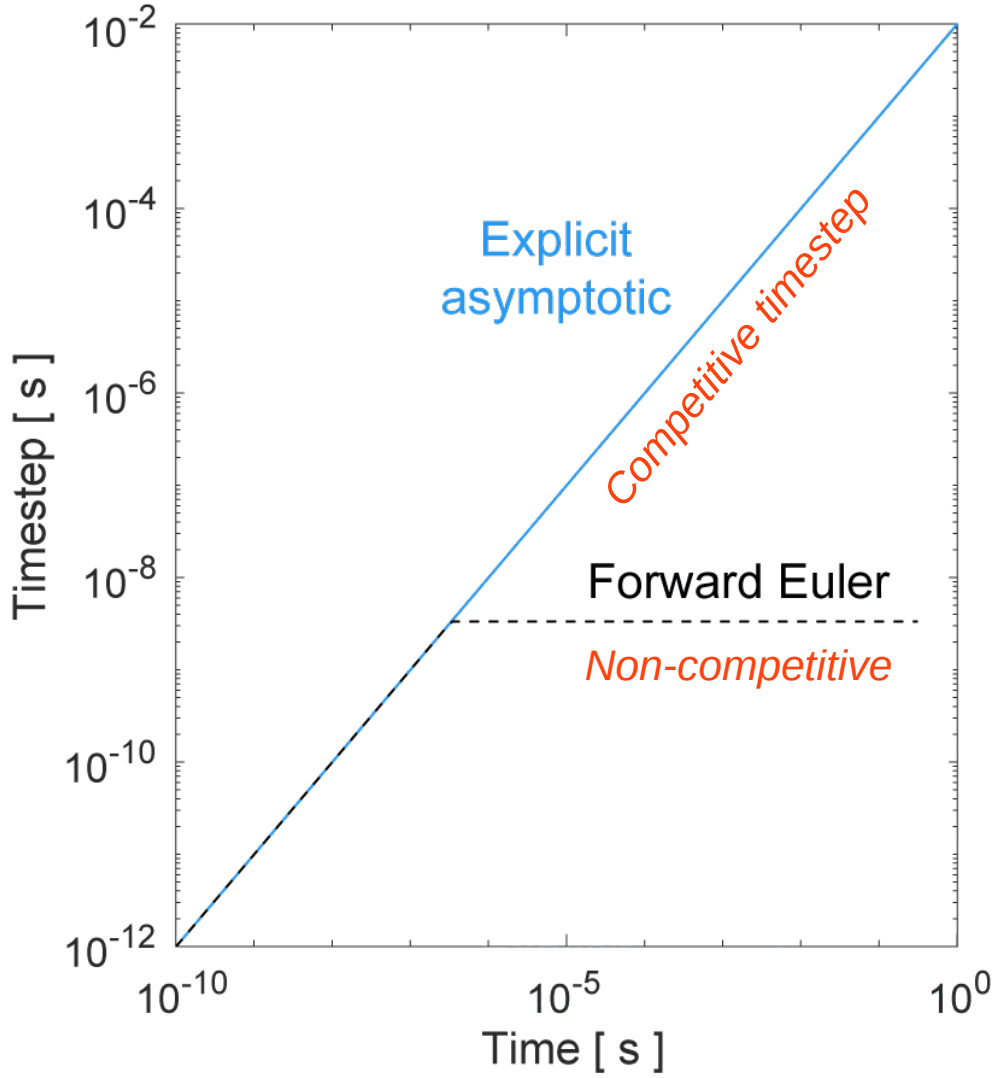
$T = 2.4 \times 10^{11}$ K

$\rho = 1.0 \times 10^{14}$ g cm^{-3}

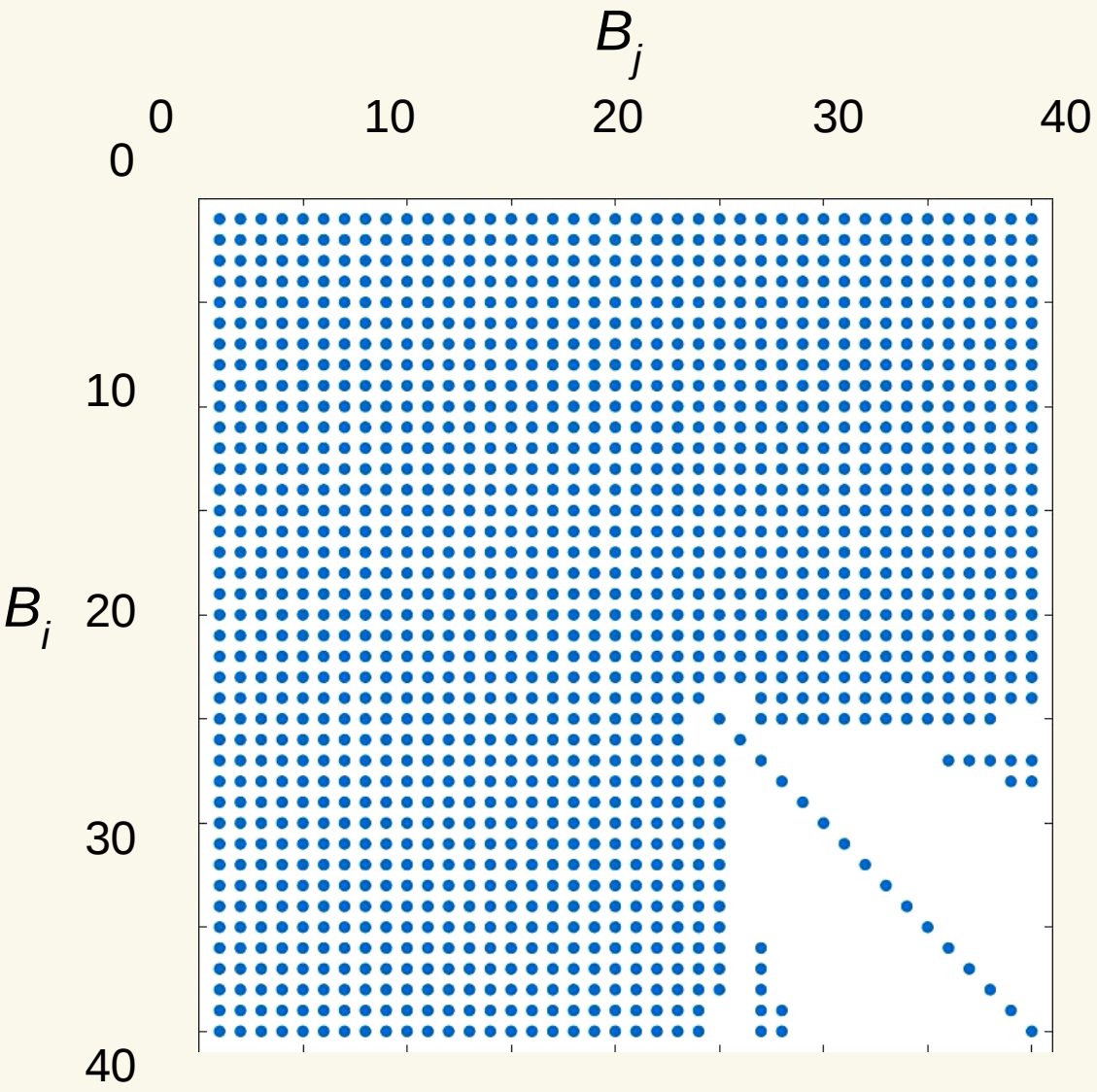
Initial gaussian @ 100 MeV

Explicit Asymptotic Calculation of Number Densities for Neutrinos of Various Energies





Removal of Reaction Groups by the Partial Equilbrium Approximation



The simplest example of a reaction group is forward-backward reaction pairs:

$$B_i \rightleftarrows B_j$$

with energy bins connected by neutrino scattering labeled by B_i and B_j .

Ridding the Galaxy of Jacobians

The computationally intensive matrix inversions required for large networks with implicit methods serve no useful purpose except to stabilize the integration.

1. If the equations could be stabilized by other means, one could do away with matrix inversions, iterative solutions, Jacobians, …

2. The resulting cleaner, more efficient computation would have various (positive) implications for the integration of large networks, particularly those coupled to fluid dynamics simulations.

We now have reason to believe that a combination of asymptotic, quasi-steady-state, and partial equilibrium approximations allows just this possibility for even the stiffest of reaction networks.

Summary

- Our new *algebraically-stabilized explicit algorithms* are intrinsically faster than standard implicit algorithms by factors of 5-10 for kinetic networks with several hundred species.
- For a single network, GPU acceleration increases this to a factor of ~20-40 for networks with several hundred species.
- The GPU is capable of running ~250-500 networks (with several hundred species in each) in parallel in about the same length of time that a standard implicit code can run one such network.
- This implies that much more realistic kinetic networks coupled to fluid dynamics are now feasible in a broad range of large-scale problems in astrophysics and other disciplines.
- We have recently demonstrated that these methods also work for neutrino transport in core-collapse supernovae or neutron star mergers.

Collaborators

- Ben Brock, *University of Tennessee*
- Daniel Shyles, *University of Tennessee*
- Andrew Belt, *University of Tennessee*
- Kyle Gregory, *University of Tennessee*
- Aaron Lackey-Stewart, *University of Tennessee*
- Adam Cole, *University of Tennessee*
- Azzam Haidar, *University of Tennessee*
- Stan Tomov, *University of Tennessee*
- Jay Billings, *Oak Ridge National Laboratory*
- Eirik Endeve, *Oak Ridge National Laboratory*
- Mike Guidry, *University of Tennessee*

References

Algebraic Stabilization of Explicit Numerical Integration for Extremely Stiff Reaction Networks, Mike Guidry, J. Comp. Phys. 231, 5266-5288 (2012). [ArXiv:1112.4778]

Explicit Integration of Extremely-Stiff Reaction Networks: Quasi-Steady-State Methods, M. W. Guidry and J. A. Harris, Comput. Sci. Disc. 6, 015002 (2013) [ArXiv:1112.4750]

Explicit Integration of Extremely-Stiff Reaction Networks: Partial Equilibrium Methods, M. W. Guidry, J. J. Billings, and W. R. Hix, Comput. Sci. Disc. 6, 015003 (2013) [arXiv: 1112.4738]

Explicit Integration of Extremely-Stiff Reaction Networks: Asymptotic Methods, M. W. Guidry, R. Budiardja, E. Feger, J. J. Billings, W. R. Hix, O. E. B. Messer, K. J. Roche, E. McMahon, and M. He, Comput. Sci. Disc. 6, 015001 (2013) [ArXiv: 1112.4716]

Explicit Integration with GPU Acceleration for Large Kinetic Networks, Ben Brock, Andrew Belt, Jay Billings, and Mike Guidry, J. Comp. Phys. 302, 591 (2015) [arXiv:1409.5826]

arXiv = http://arxiv.org/