GPU-accelerated interstellar chemistry with WIND, a new general ODE solver

Alex Gurvich
Blue Waters Graduate Fellow
Northwestern University
June 4th, 2019 - Blue Waters Symposium

Northwestern

CIERA

CENTER FOR INTERDISCIPLINARY EXPLORATION AND RESEARCH IN ASTROPHYSICS

I use BW to:
develop a new
GPU-accelerated
ODE solver

Galaxy formation and interstellar chemistry (on FIRE)

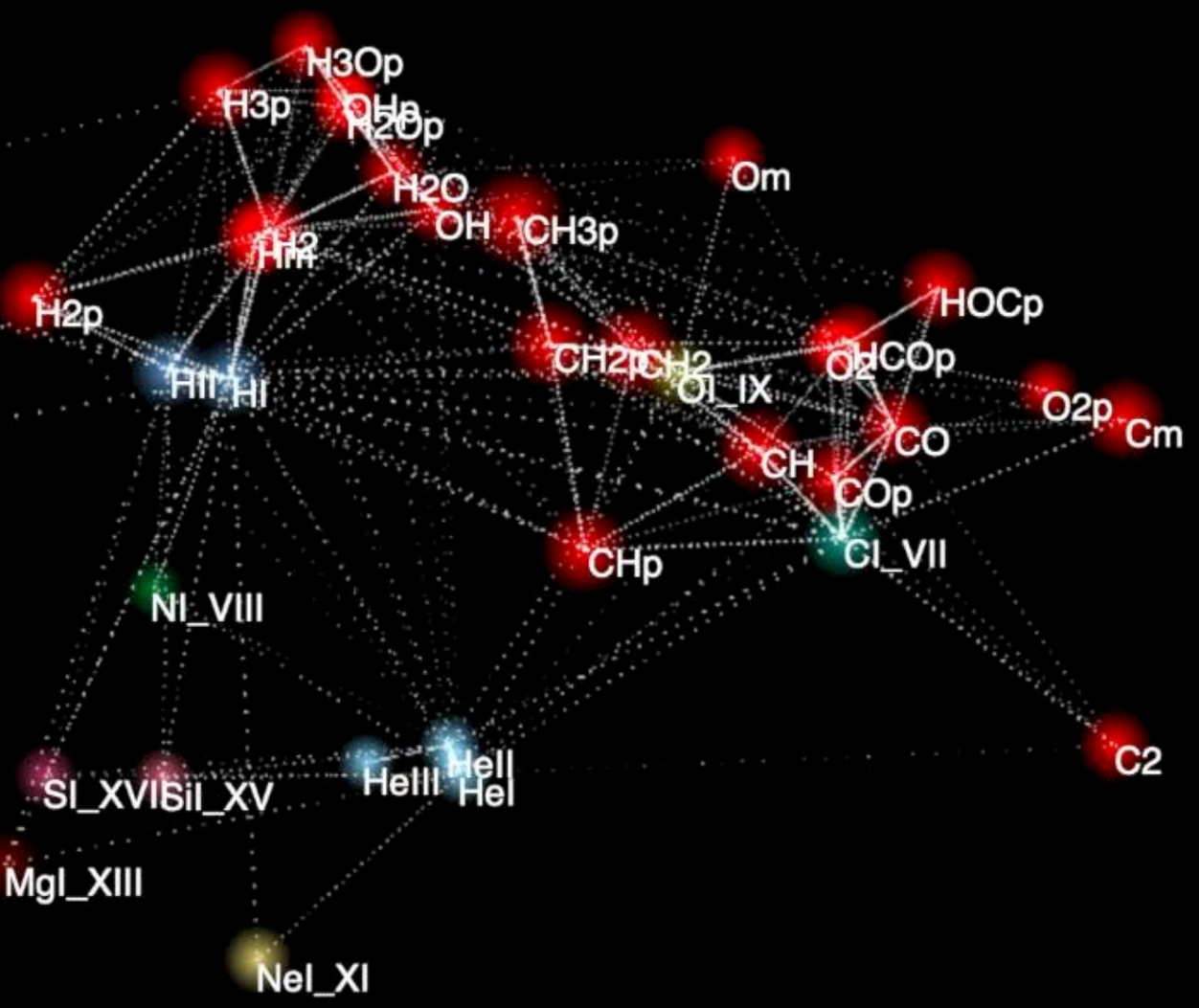
CURRENT:

- Gravity
- Hydrodynamics
- Star formation
  - feedback

NEW: time-dep. chemistry

- Informs gas cooling rate
- Predicts observations
  - abundances/masses
  - emission/absorption spectra

The CHIMES time-dependent chemistry network



- Reaction network of 157 coupled “stiff” ODEs
  - includes metal ions & molecules
    - CO, H_2, OVI, etc…
    - many different timescales
- Prohibitively expensive, up to ~90% of work

How to integrate ODEs

Solving differential equations numerically
both explicitly and implicitly

Explicit: Runge Kutta (RK2)

$$y_{n+1} = y_n + hf\left(y_n + \frac{h}{2}f(y_n)\right)$$

must resolve short timescales or diverges

Implicit: Semi-Implicit Euler (SIE)

$$y_{n+1} = y_n + hf(y_{n+1})$$

$$y_{n+1} \approx y_n + h\left(1 - h\frac{\partial f}{\partial y}\right)^{-1} f(y_n)$$

converges to answer at late times

Solving differential equations numerically
both explicitly and implicitly

Explicit: Runge Kutta (RK2)

$$y_{n+1} = y_n + hf\left(y_n + \frac{h}{2}f(y_n)\right)$$

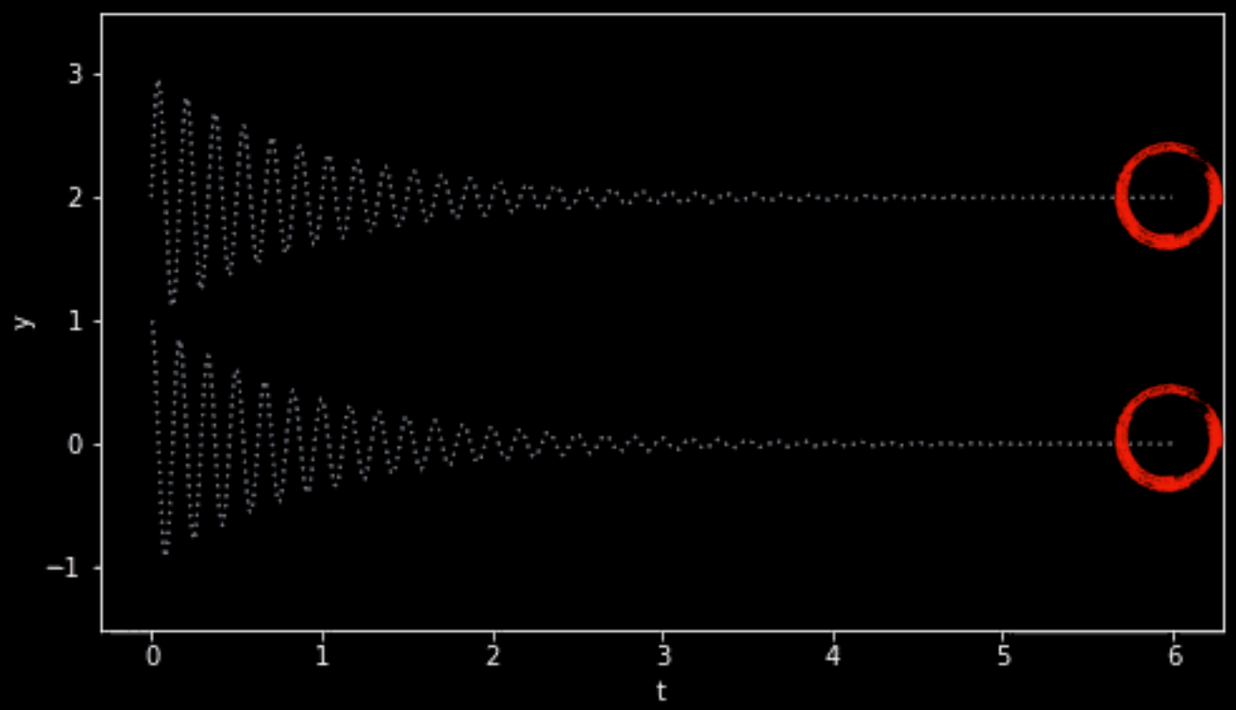
must resolve short timescales or diverges

Implicit: Semi-Implicit Euler (SIE)

$$y_{n+1} = y_n + hf(y_{n+1})$$

$$y_{n+1} \approx y_n + h\left(1 - h\frac{\partial f}{\partial y}\right)^{-1} f(y_n)$$

converges to answer at late times



goal is to find final value

Solving (coupled) ODEs simultaneously with linear algebra

$$\begin{bmatrix} y_0 \\ y_1 \\ y_2 \end{bmatrix} \quad \vec{y}$$

$$\begin{bmatrix} \frac{dy_0}{dt}(y_0, y_1, \cdots) \\ \frac{dy_1}{dt}(y_0, y_1, \cdots) \\ \frac{dy_2}{dt}(y_0, y_1, \cdots) \end{bmatrix} \quad \vec{f}$$

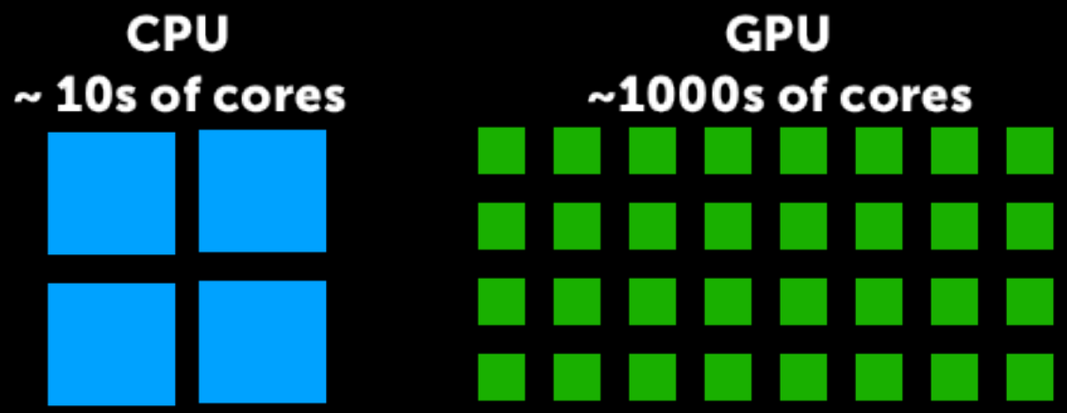
$$\begin{bmatrix} \frac{\partial f_0}{\partial y_0} & \frac{\partial f_0}{\partial y_1} & \frac{\partial f_0}{\partial y_2} \\ \frac{\partial f_1}{\partial y_0} & \frac{\partial f_1}{\partial y_1} & \frac{\partial f_1}{\partial y_2} \\ \frac{\partial f_2}{\partial y_0} & \frac{\partial f_2}{\partial y_1} & \frac{\partial f_2}{\partial y_2} \end{bmatrix} \quad \mathbf{J}$$

Why GPUs?

How are GPUs and CPUs different?

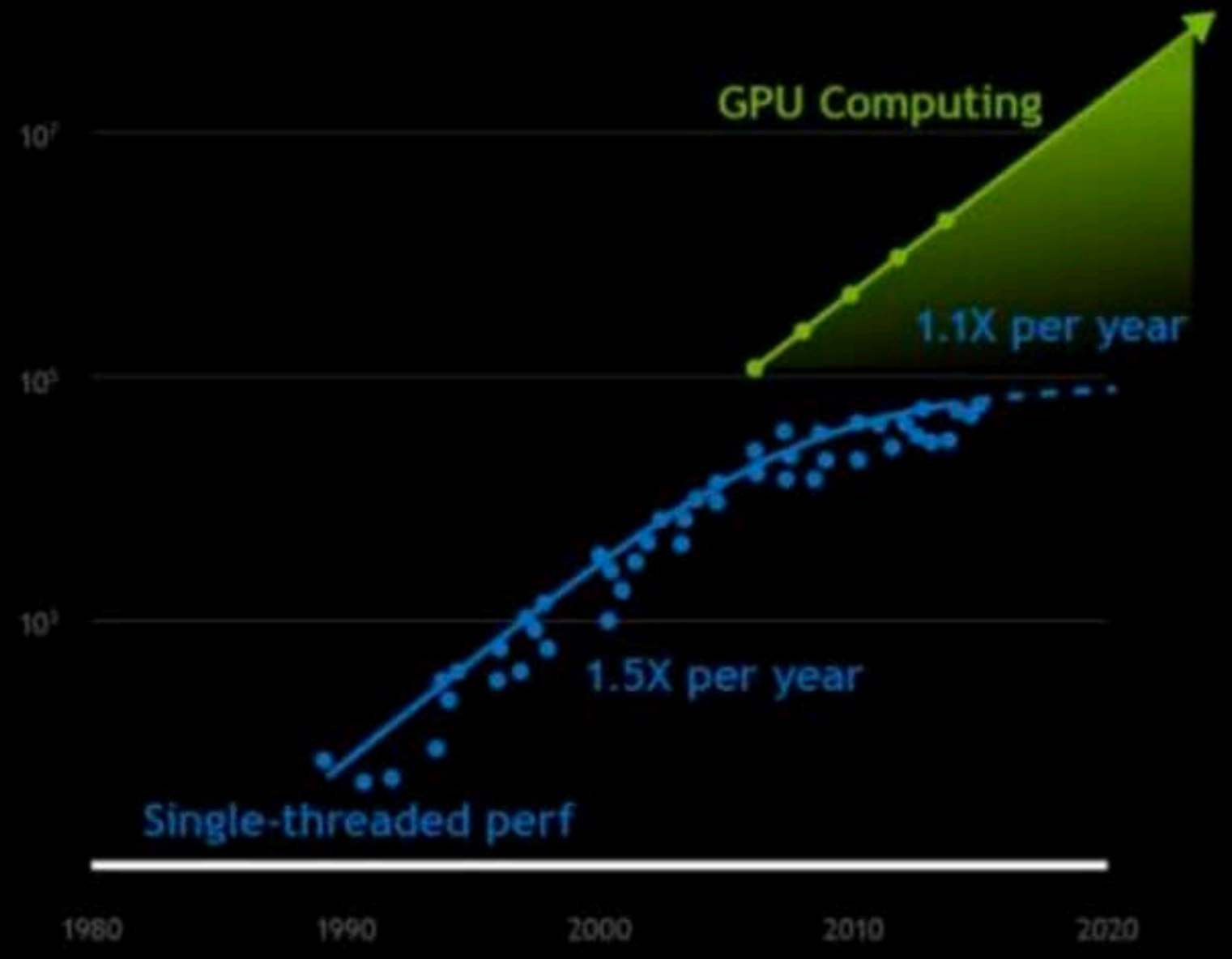
- Many threads that operate concurrently
  - good for vector operations & linear algebra

- Memory bandwidth
- Typically requires substantial code/ algorithm rewrite
  - optimal configuration hardware dependent



GPUs aren't the next big thing, they are the current thing

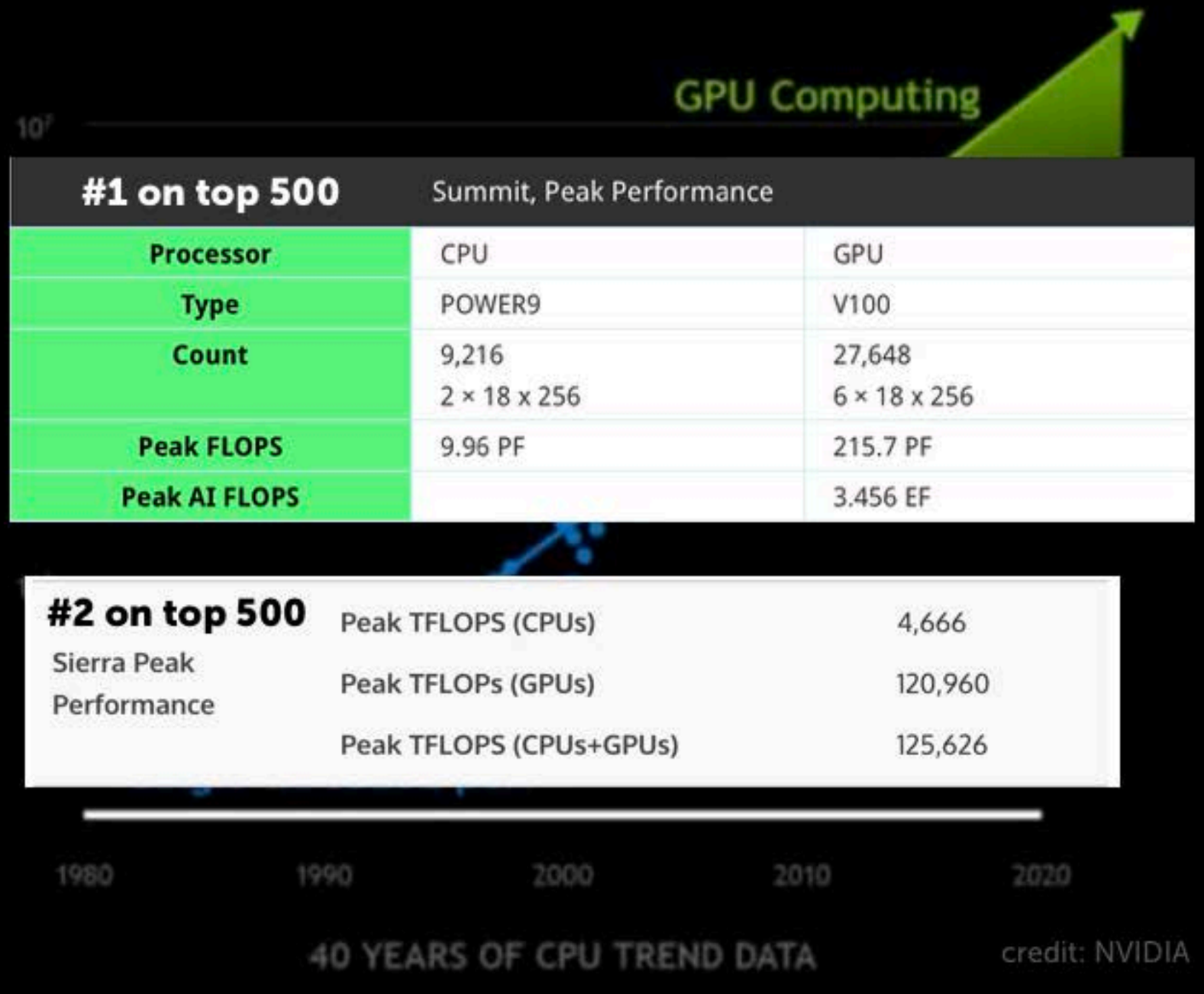
- Modern HPC resources are GPU powered
  - ~95% peak flops are on GPUs
- more power efficient



credit: NVIDIA

GPUs aren't the next big thing, they are the current thing

- Modern HPC resources are GPU powered
  - ~95% peak flops are on GPUs
- more power efficient



https://www.olcf.ornl.gov/summit/ https://hpc.llnl.gov/hardware/platforms/sierra

(accepting suggestions for the backronym)

WIND

Already implemented:
2 solvers, RK2 and SIE on both the GPU and CPU

How does it perform?

Using a five species H-He chemical network as a test case

$$R^a = \kappa^a_{\mu\nu} n^\mu n^\nu$$

$$n^\mu = \begin{bmatrix} \text{HI} \\ \text{HII} \\ \text{HeI} \\ \text{HeII} \\ \text{HeIII} \end{bmatrix}$$

$$n_e = n_{HI} + n_{HeII} + 2n_{HeIII}$$

$$R^{HI} = \alpha_{HII} n_e n_H - \left(\alpha_{HII} + \Gamma_{e,HI} + \Gamma_{\gamma,HI}/n_e\right) n_e n_{HI}$$

$$R^{HII} = -R^{HI}$$

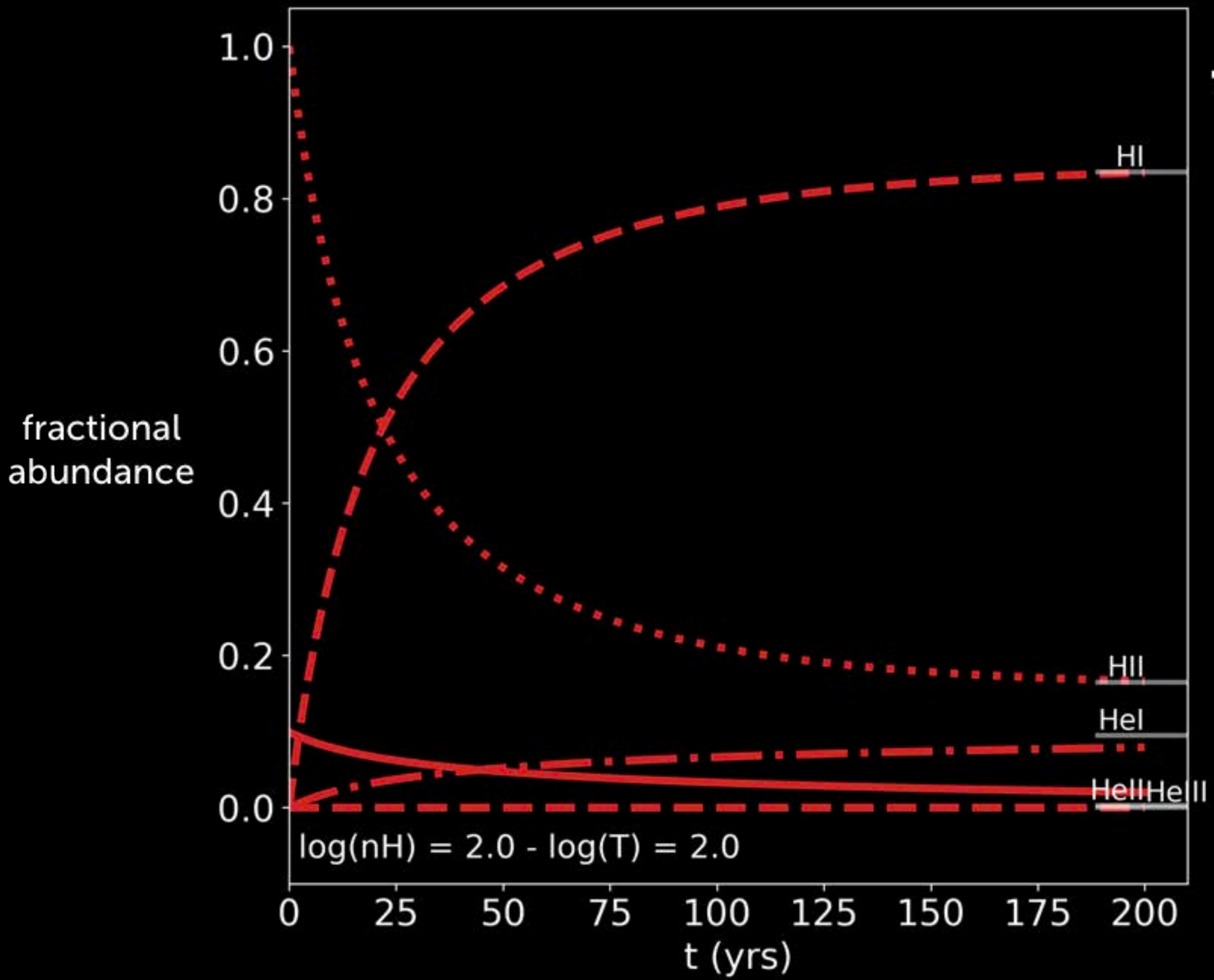
$$R^{HeI} = (\alpha_{HEII} + \alpha_d) n_e n_{HeII} - \left(\Gamma_{e,HeI} + \Gamma_{\gamma,HeI}/n_e\right) n_e n_{HeI}$$

$$R^{HeII} = -(R^{HeI} + R^{HeIII})$$

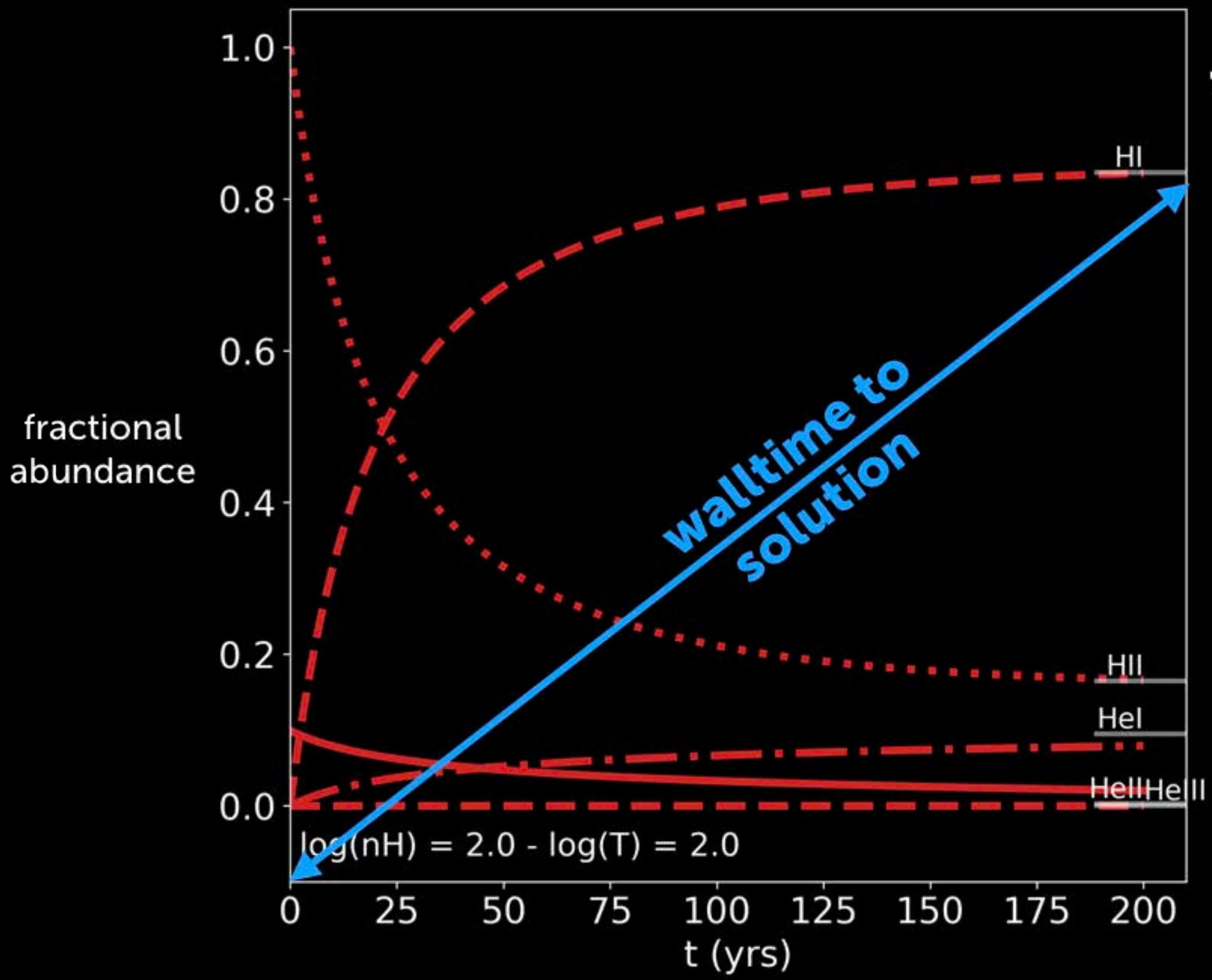
$$R^{HeIII} = \underbrace{\left(\Gamma_{e,HeII} + \Gamma_{\gamma,HeII}/n_e\right) n_e n_{HeII}}_{\text{ionization}} - \underbrace{\alpha_{HeIII} n_e n_{HeIII}}_{\text{recombination}}$$

assume temperature & density is fixed → **evolve to equilibrium**

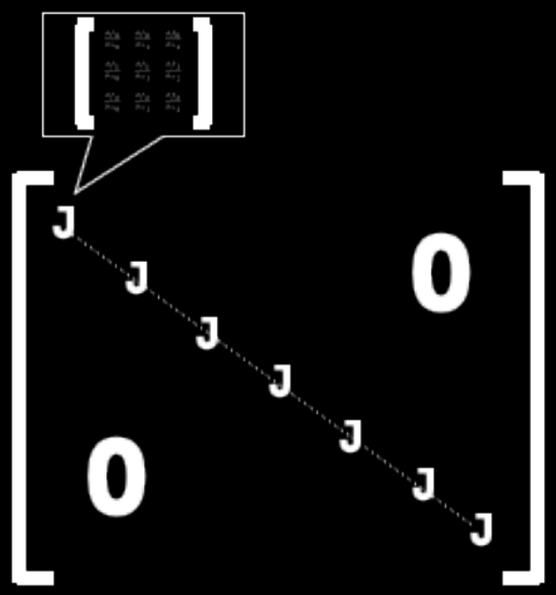
fixed tolerance

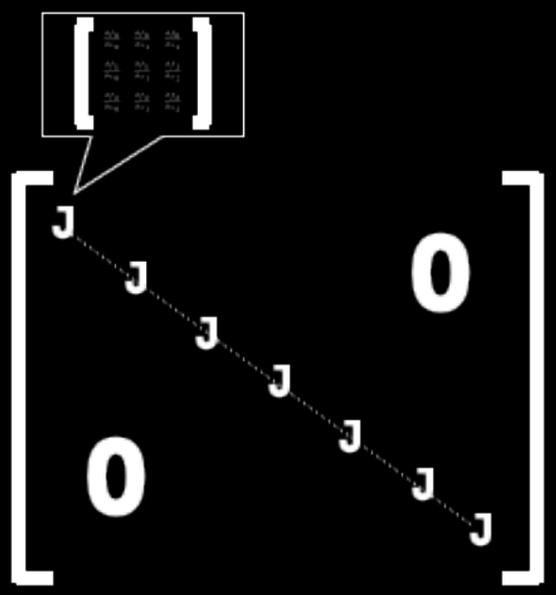


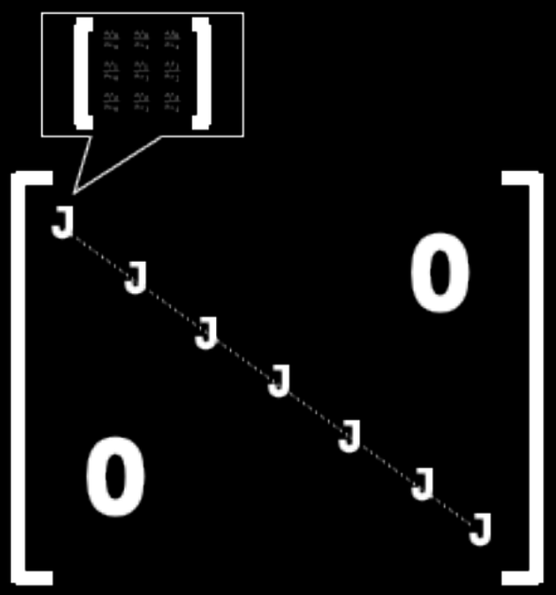
fixed tolerance

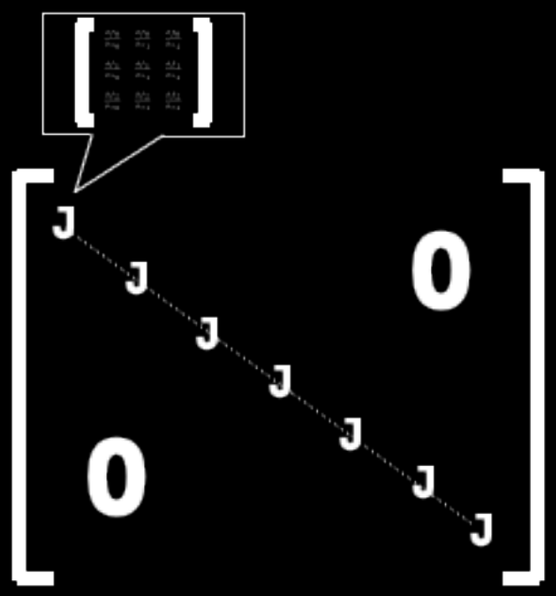


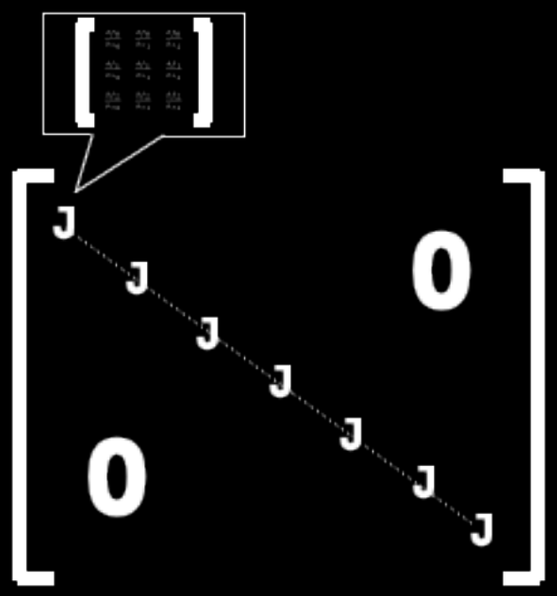
Mimic the computational challenge of the full CHIMES chemistry network

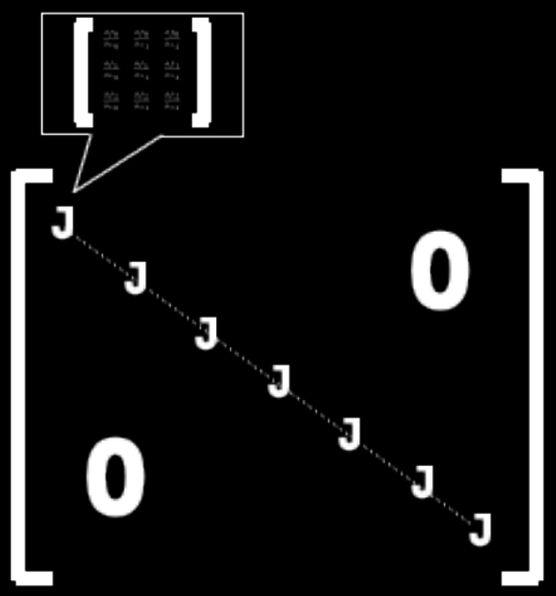








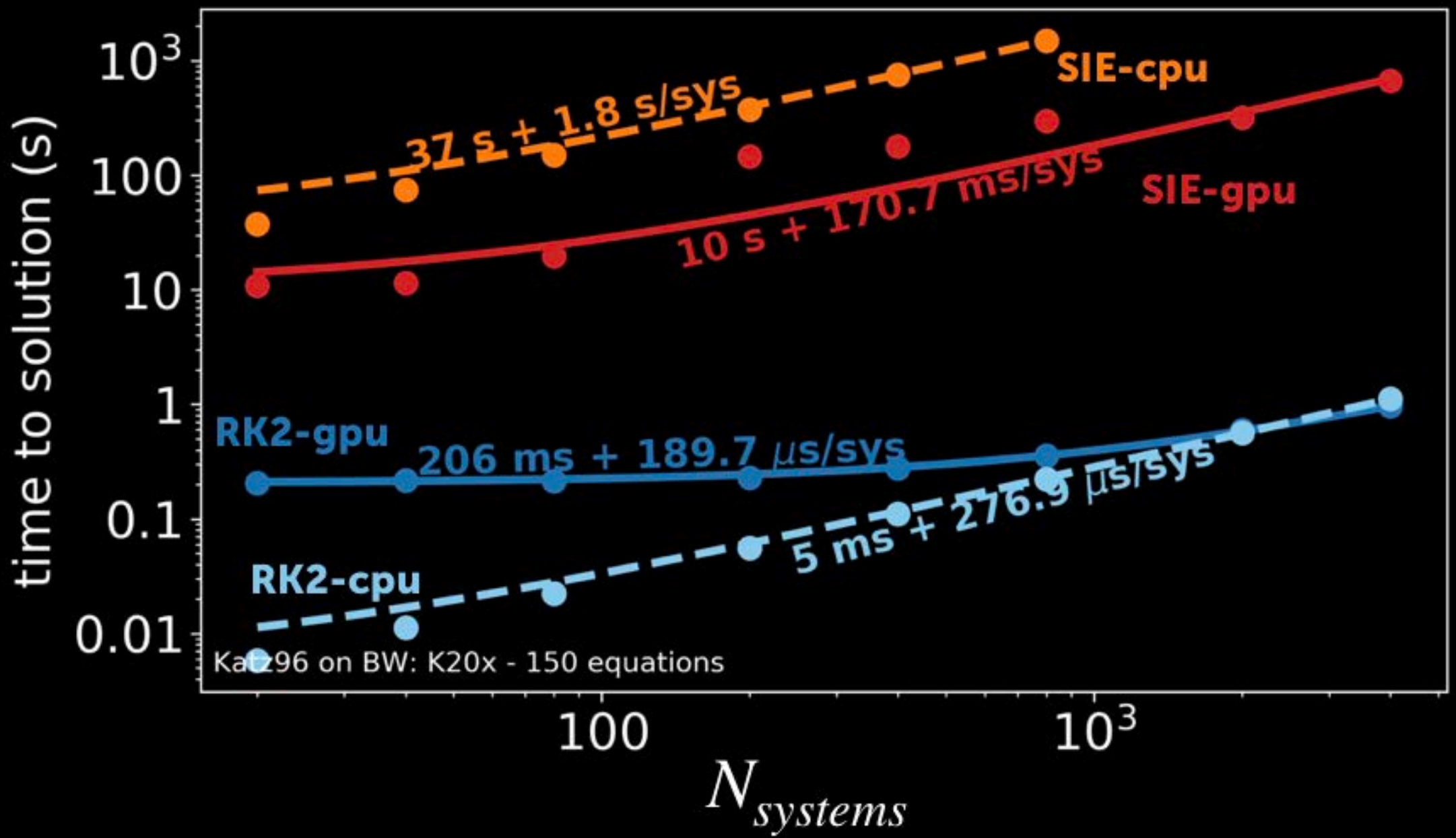




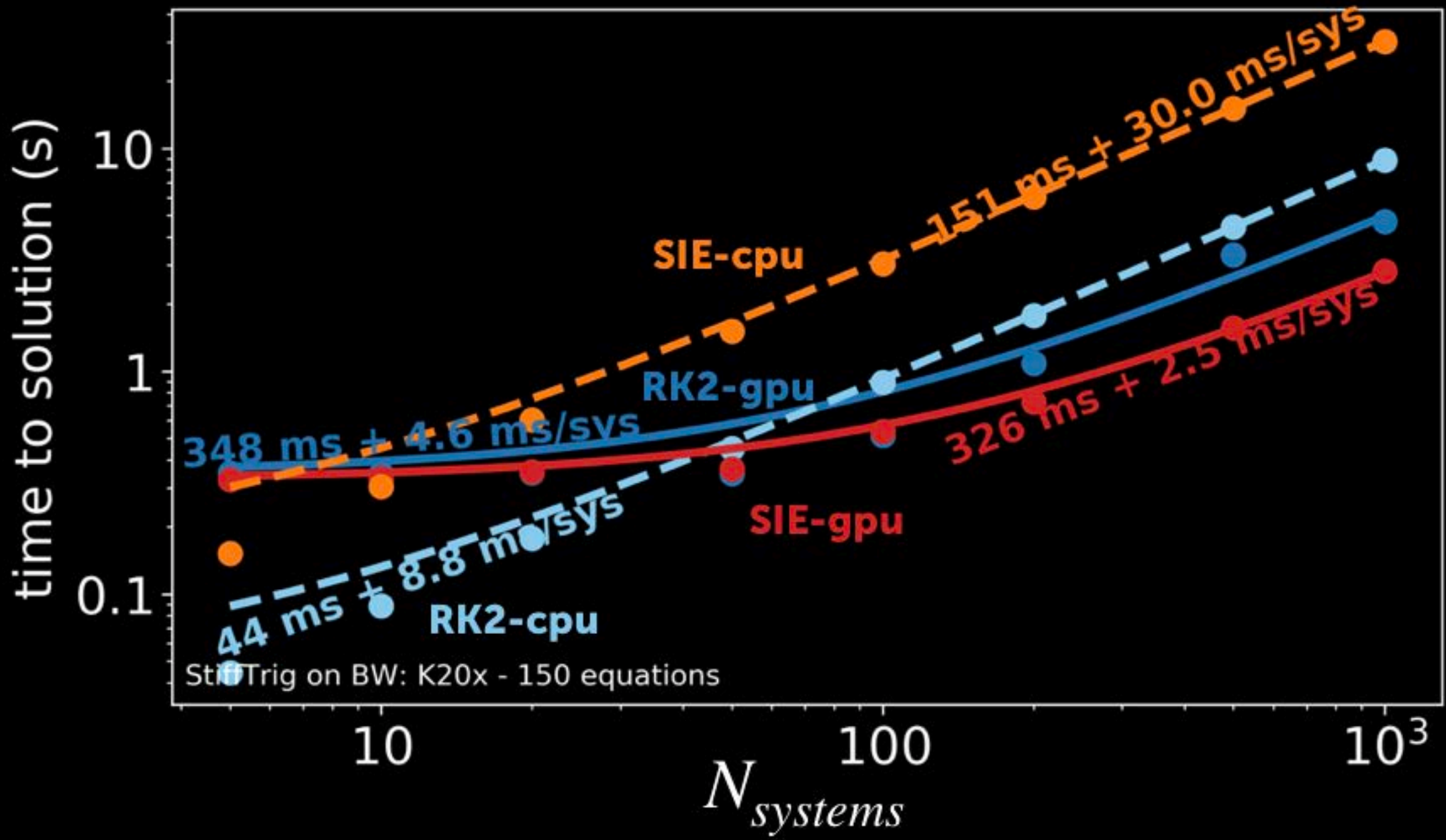
total size: 150x150

~10^6 systems

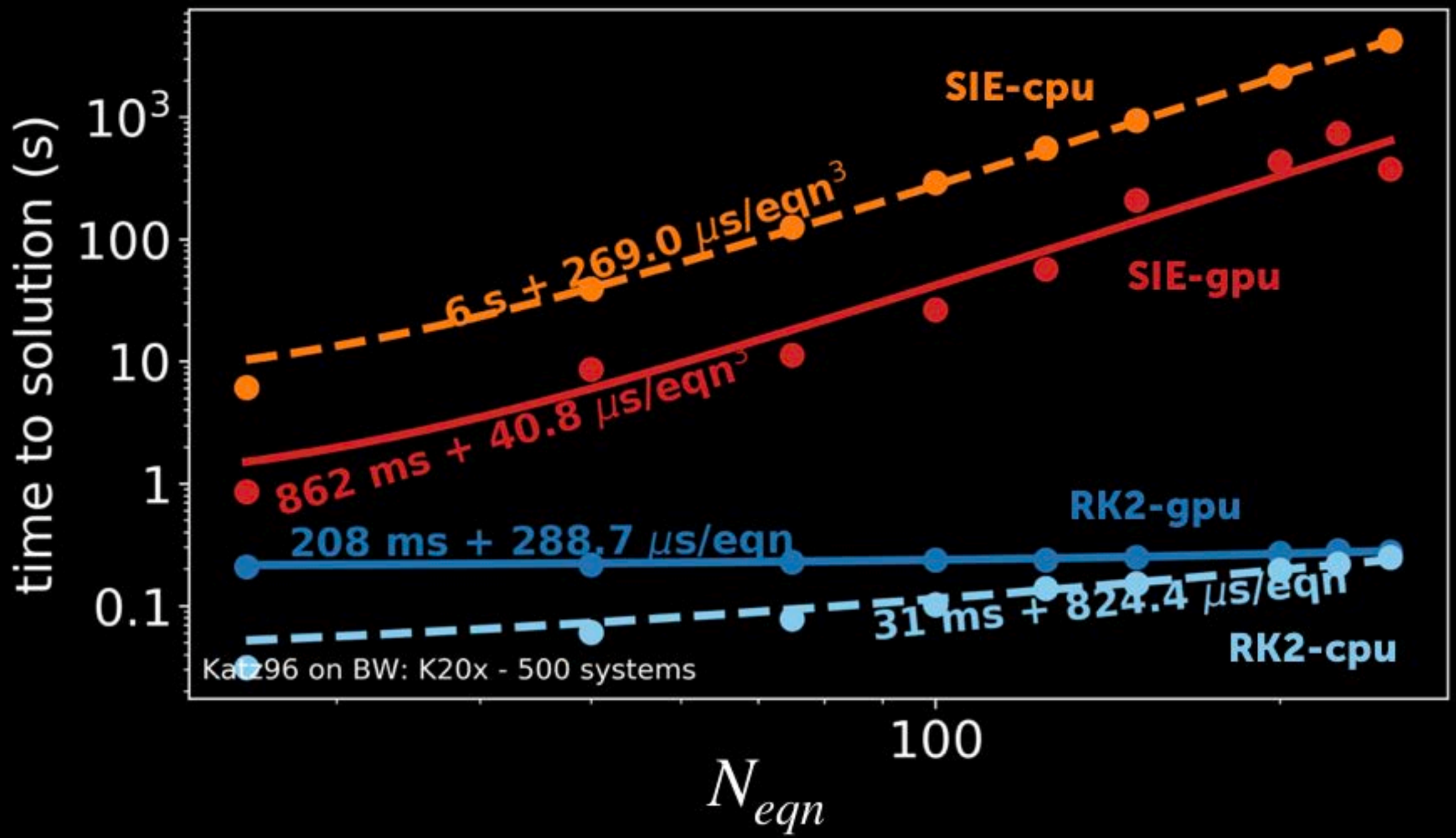
Changing the number of systems



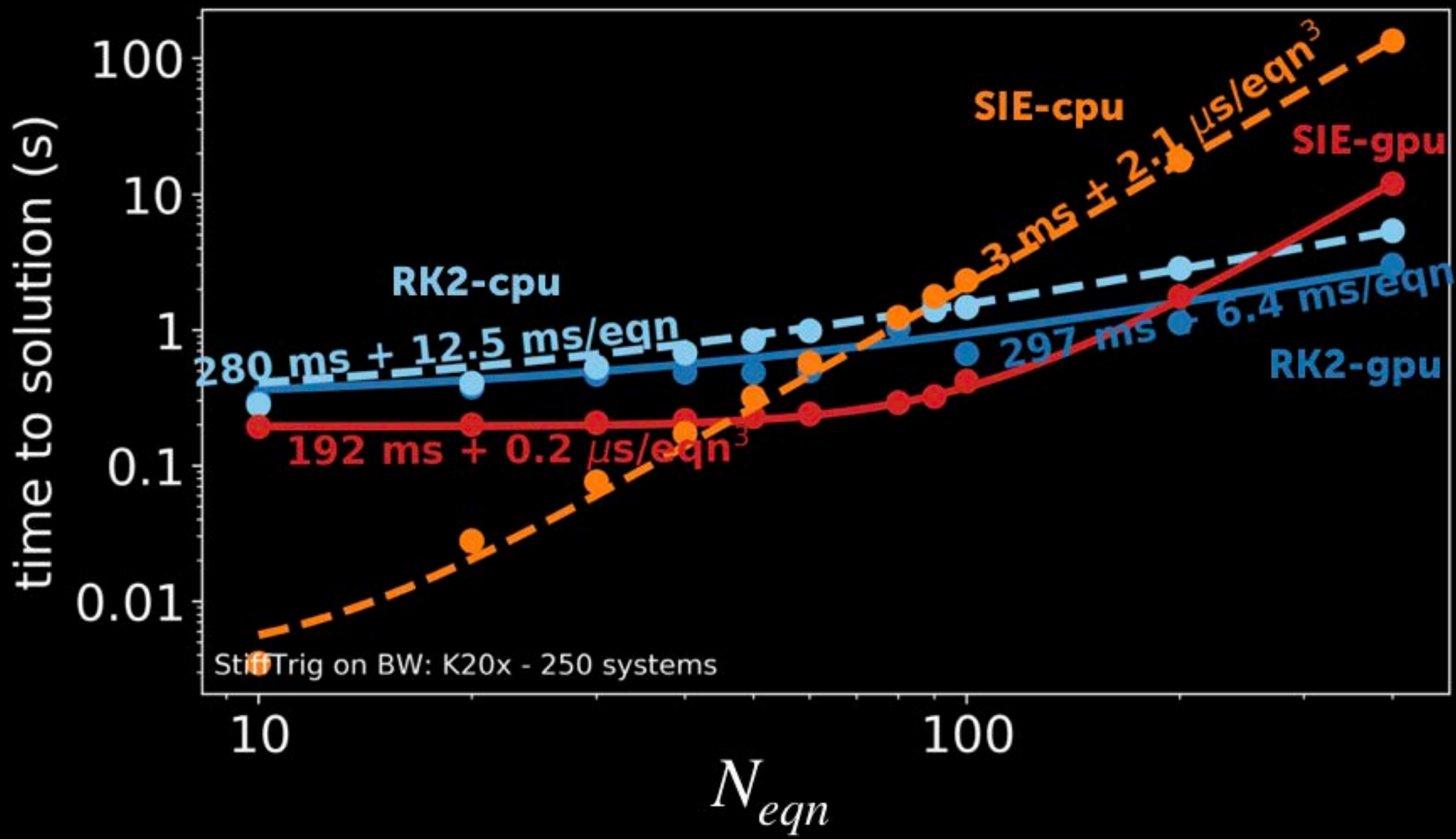
Changing the number of systems



Changing the size of each system



Changing the size of each system



Takeaways

- We have to adapt to a new computing paradigm to take advantage of the most powerful HPC resources
- WIND is a new general, multi-method, GPU-accelerated ODE solver
  - explicit methods are efficient on smooth problems
  - implicit methods are advantageous for stiff systems such as a more complete chemistry network like CHIMES

Next steps

- Implement optimized solvers for sparse systems
  - cost scales as $N^3 \rightarrow N$
  - reduces memory footprint from $N^2 \rightarrow N$
- Testing on more recent GPUs
  - Pascal & Volta
- Encode CHIMES network into WIND and attach to FIRE to run on galaxy simulations

Thanks to Blue Waters Fellowship