Implicit Integration Methods for Dislocation Dynamics

ICERM

August 31, 2015

David J. Gardner[1],

C. S. Woodward[1] , D. R. Reynolds[2], K. Mohror[1],

G. Hommes[1], S. Aubry[1], M. Rhee[1], and A. Arsenlis[1]

[1]LLNL, [2]SMU

Lawrence Livermore National Laboratory

LLNL-PRES-676689

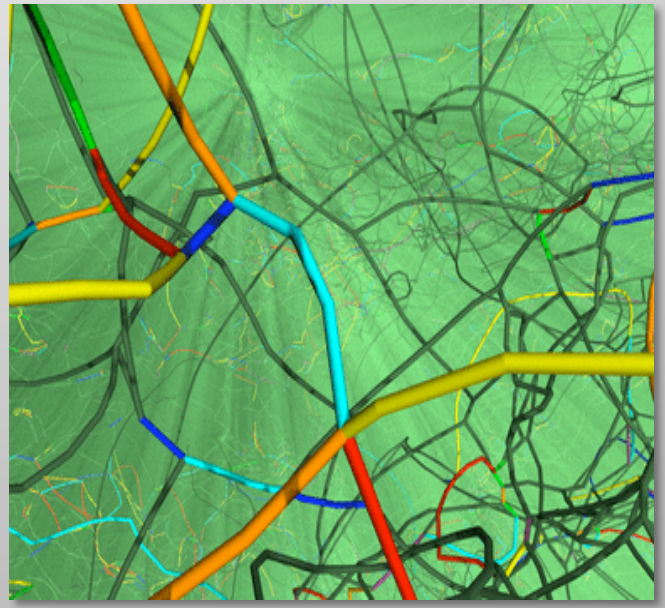
This work was performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under Contract DE-AC52-07NA27344. Lawrence Livermore National Security, LLC

Outline

- Dislocation Dynamics
- Time Integrators
- Nonlinear Solvers
- Numerical Results
- Conclusions

Dislocation Dynamics

- The strength of a crystalline material depends on the motion, multiplication, and interaction line defects in the crystal lattice
- These *dislocations* are the carriers of *plasticity* and play an important role in *strain hardening* where continued deformation increases the material's strength
- Dislocation dynamics simulations attempt to connect aggregate dislocation behavior to macroscopic material response
- A typical simulation involves millions of segments evolved for several hundred thousand time steps to reach 1% of plastic strain



Parallel Dislocation Simulator (ParaDiS)

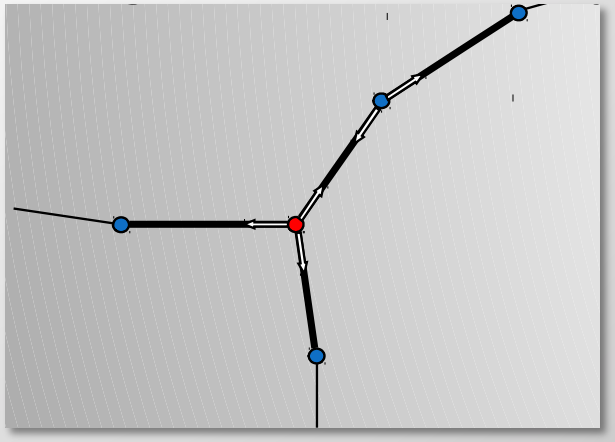
- Discretize dislocation lines as line segments terminated by nodes
- Node locations evolve in time $\frac{dx_i(t)}{dt} = v_i(t)$
- Velocities are determined from nodal forces and a material specific mobility law

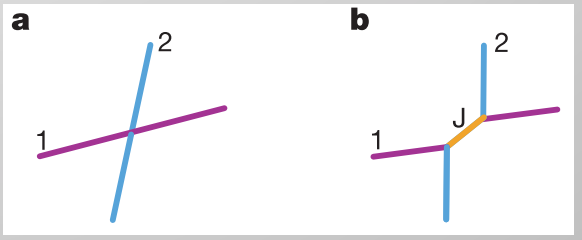
$$v_i(t) = M(F_i(t))$$

- Nodal forces are computed using local and Fast Multipole Methods

$$F_i(t) = f_i^{\text{self}} + f_i^{\text{core}} + f_i^{\text{external}} + f_i^{\text{interaction}}$$

- Discretization adaptation and topology changes occur between each time step



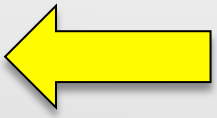


Two dislocation lines intersect and zip to form a binary junction

Computational Challenges

- Dislocation dynamics simulations are computationally challenging
  - Expensive force calculations
  - Discontinuous topological events
  - Rapidly changing problem size
- Standard time integration methods for dislocation dynamics are explicit Euler and the trapezoid algorithm
- ParaDiS uses the trapezoid method paired with a fixed point iteration
- Previously, other integration methods were not systematically studied for dislocation dynamics
- To enable larger time steps and faster simulations we focus on
  - Enhancing the native trapezoid method with more robust nonlinear solvers
  - Utilizing higher-order multistage implicit integration methods

Outline

- Dislocation Dynamics
- Time Integrators 
  - Trapezoid Integrator
  - DIRK Integrators
- Nonlinear Solvers
- Numerical Results
- Conclusions

Trapezoid Integrator

- Consider the initial value problem $y'(t) = f(t, y(t)), \qquad y(t_0) = y_0$
- The trapezoid method is the default integrator in ParaDiS,

$$y_{n+1} = y_n + \frac{h_n}{2}\left(f(t_n, y_n) + f(t_{n+1}, y_{n+1})\right)$$

- This is the simplest second-order one-step implicit method and only requires the most recent solution value
- The new solution is computed by solving a nonlinear residual equation such that $\|g(y)\|_\infty \leq \epsilon_n$

$$g(y) = y - y_n - \frac{h_n}{2}\left(f(t_n, y_n) + f(t_{n+1}, y)\right) = 0$$

- In ParaDiS, time step sizes are adapted based on the success or failure of solving the residual equation

DIRK Integrators

- Consider the initial value problem $y'(t) = f(t, y(t)), \qquad y(t_0) = y_0$

- High-order embedded diagonally implicit Runge-Kutta (DIRK) time integration methods are defined by

$$z_i = y_n + h_n \sum_{j=1}^{i} A_{i,j}\, f(t_n + c_j h_n, z_j), \quad i = 1, \ldots, s,$$

$$y_{n+1} = y_n + h_n \sum_{j=1}^{s} b_j\, f(t_n + c_j h_n, z_j),$$

$$\tilde{y}_{n+1} = y_n + h_n \sum_{j=1}^{s} \tilde{b}_j\, f(t_n + c_j h_n, z_j).$$

- Computing the next time step value requires solving *s* implicit systems for the internal stages

$$g(z) = z - h_n\, A_{i,i}\, f(t_n + c_i h_n, z) - y_n - h_n \sum_{j=1}^{i-1} A_{i,j}\, f(t_n + c_j h_n, z_j) = 0$$

DIRK Integrators

- Embedded Runge-Kutta methods allow for time stepping adaptivity based on an estimate of the solution error

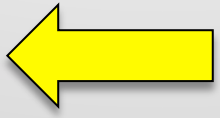
$$e_n = \|y_n - \tilde{y}_n\|_{WRMS} = \left(\frac{1}{N} \sum_{k=1}^{N} \left(\frac{y_{n,k} - \tilde{y}_{n,k}}{r_{tol}|y_{n-1,k}| + a_{tol}} \right)^2 \right)^{1/2}$$

- Steps with $e_n \leq 1$ are accepted, otherwise the step is repeated
- Up to the last three error estimates are used to predict the step size for a new or repeated step (PID, PI, I time-adaptivity controllers)
- The nonlinear equations for the internal stages are solved such that

$$\|g(z)\|_{WRMS} \leq \epsilon_n$$

- Note that here $\epsilon_n \leq 1$, with the trapezoid integrator ϵ_n is the absolute error of nodal positions

Outline

- Dislocation Dynamics
- Time Integrators
- Nonlinear Solvers 
  - Anderson Accelerated Fixed Point
  - Newton's Method
- Numerical Results
- Conclusions

Anderson Acceleration

- The simplest approach for solving the nonlinear systems arising from the implicit integration methods is a fixed point iteration

$$y^{(k+1)} = \phi(y^{(k)}) \qquad \text{where} \qquad \phi(y) \equiv y - g(y)$$

- For a sufficiently small h_n, the iteration is linearly convergent

- Significant speedups can be obtained with Anderson Acceleration

Algorithm AA: ANDERSON ACCELERATION

Given $y^{(0)}$ *and* $m \geq 1$.
Set $y^{(1)} = \phi(y^{(0)})$.
For $k = 1, 2, \ldots,$ *until* $\|y^{(k+1)} - y^{(k)}\| < \epsilon_n$
  Set $m_k = \min\{m, k\}$.
  Set $F_k = [f_{k-m_k}, \ldots, f_k]$, *where* $f_i = \phi(y^{(i)}) - y^{(i)}$.
  Determine $\alpha^{(k)} = \left[\alpha_0^{(k)}, \ldots, \alpha_{m_k}^{(k)}\right]^T$ *that solves*
    $\min_\alpha \|F_k \alpha\|_2$ *such that* $\sum_{i=0}^{m_k} \alpha_i = 1$.
  Set $y^{(k+1)} = \sum_{i=0}^{m_k} \alpha_i^{(k)} \phi(y^{(k-m_k+i)})$.

Newton's Method

- For solving $g(y) = 0$, the k[th] Newton iteration update is the root of the linear model

$$g(y^{(k+1)}) \approx g(y^{(k)}) + J_g(y^{(k)})(y^{(k+1)} - y^{(k)})$$

- The linear system is solved inexactly with GMRES
- Jacobian-vector products are approximated using a finite difference

$$J_g(y)v \approx \frac{g(y + \epsilon v) - g(y)}{\epsilon}$$

- For a good initial guess, the iteration is quadratically convergent

Algorithm INI: INEXACT NEWTON ITERATION

Given $y^{(0)}$.

For $k = 0, 1, \ldots,$ *until* $\|g(y^{(k)})\| < \epsilon_n$

For $tol_L \in [0, 1)$, *approximately solve* $J_g(y^{(k)})\Delta y^{(k)} = -g(y^{(k)})$

so that $\|J_g(y^{(k)})\Delta y^{(k)} + g(y^{(k)})\| \leq tol_L \|g(y^{(k)})\|$.

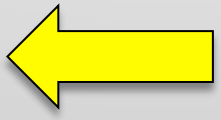
Set $y^{(k+1)} = y^{(k)} + \Delta y^{(k)}$.

SUNDIALS

- The nonlinear solvers and DIRK integration methods employed in the following tests are part of the SUNDIALS suite of codes
- KINSOL: nonlinear solvers including Fixed Point with Anderson acceleration and Newton-Krylov method
- ARKode: Adaptive-step time integration package for stiff, nonstiff, and multi-rate systems of ordinary differential equations using additive Runge Kutta methods
- Written in C with interfaces to Fortran and Matlab
- Designed to be incorporated into existing codes and modular structure allows user to supply their own data structures

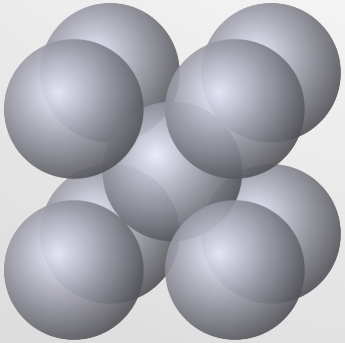
https://computation.llnl.gov/casc/sundials

Outline

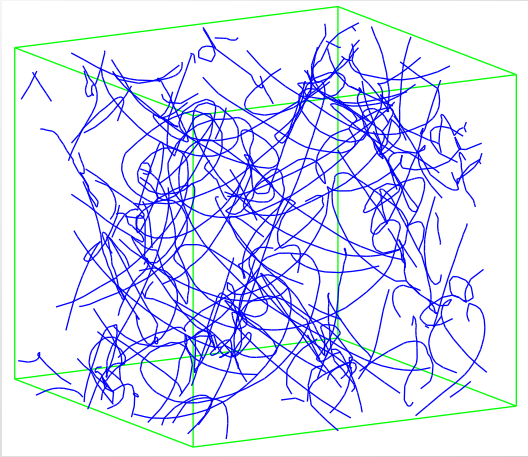
- Dislocation Dynamics
- Time Integrators
- Nonlinear Solvers
- Numerical Results 
  - BCC Crystal
  - Large Scale Strain Hardening
  - Annealing
- Conclusions

BCC Crystal

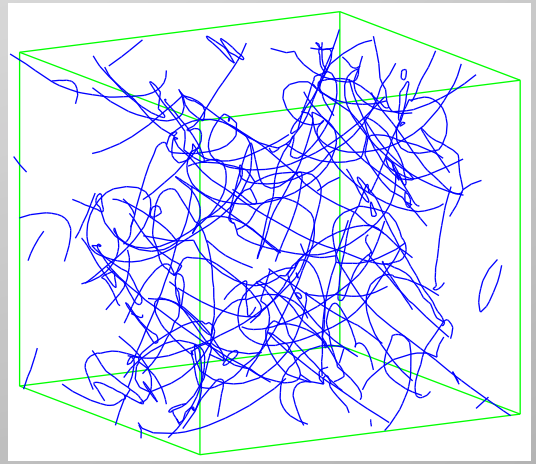
- Body-centered cubic crystal, 4.25 μm^3



- Temp = 600K, Pres. = 0 GPa
- Constant 10^3 s^{-1} strain along the x-axis
- Tolerance = 0.5 |b|
- Tests run on 16 cores of LLNL Cab machine:
  - 430 Teraflop Linux cluster system
  - 1,296 nodes, 16 cores and 32 GB memory per node



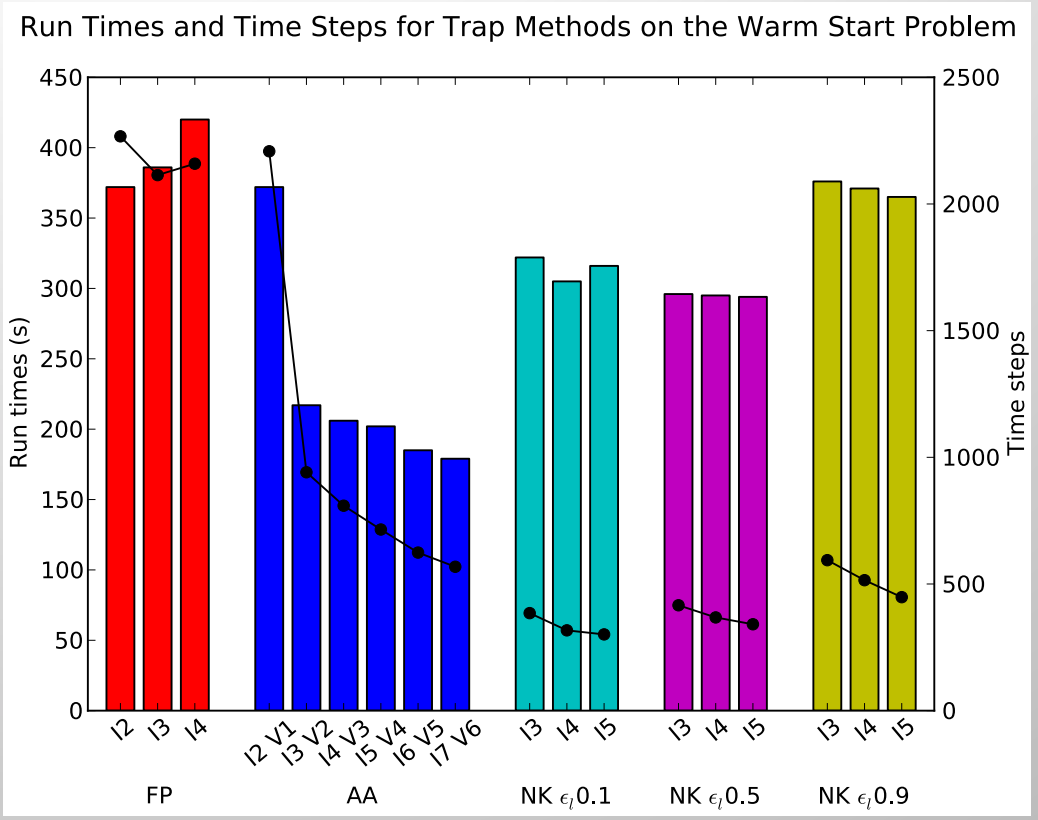
Initial State, t = 3.3 μs
~2,850 nodes



Final State, t = 4.4 μs
~2,920 nodes

BCC Crystal: Trapezoid

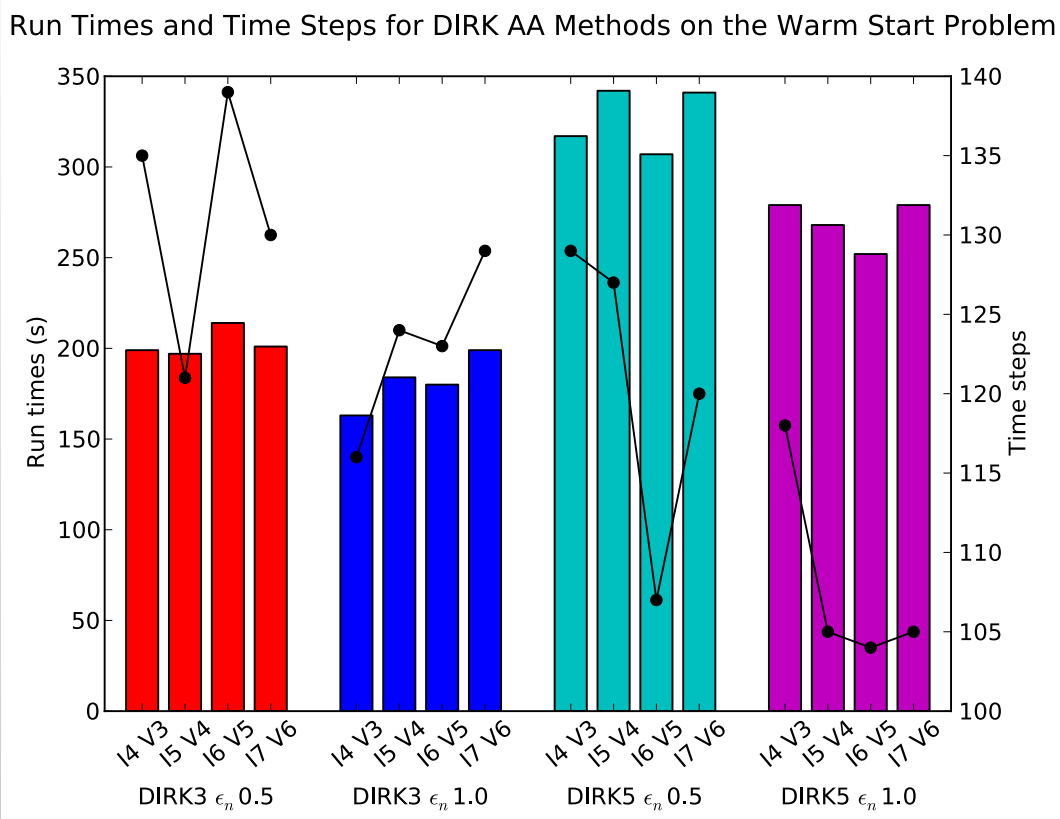
- Additional fixed point iterations do not improve results
- Anderson acceleration runtimes and number of time steps decrease monotonically with increased iterations
- Newton takes the fewest time steps but is slower than accelerated fixed point



Trapezoid with Anderson Acceleration
52% speedup
over trapezoid with fixed point

BCC Crystal: DIRK with Anderson

- Both 3rd and 5th order methods are faster than trapezoid with the fixed point iteration
- The 3rd order method is uniformly faster than the 5th order method
- Fastest results are with a looser nonlinear tolerance
- Runtime does not depend heavily on the max number of iterations



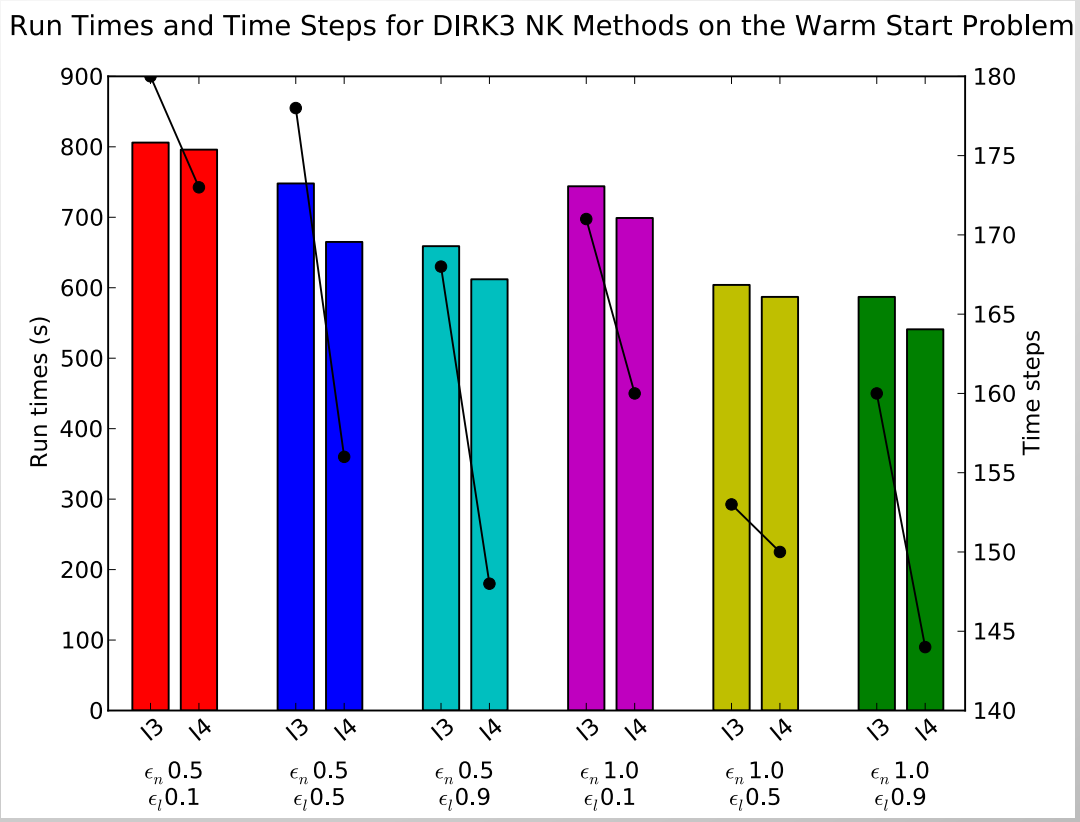
3rd order DIKR with Anderson Acceleration

56% speedup

over trapezoid with fixed point

BCC Crystal: DIRK with Newton

- Newton's method takes approximately 1/15 of the number of time steps but is 68% slower than trapezoid with fixed point
- Multiple implicit stage solves and additional function evaluations for finite difference Jacobian-vector products lead to slower runtimes



Potential with Newton's Method

- The finite difference Jacobian-vector products require an additional force evaluation *each linear iteration*
- Assuming the force calculations dominate the runtime cost then we can estimate the runtime with an analytic Jacobian

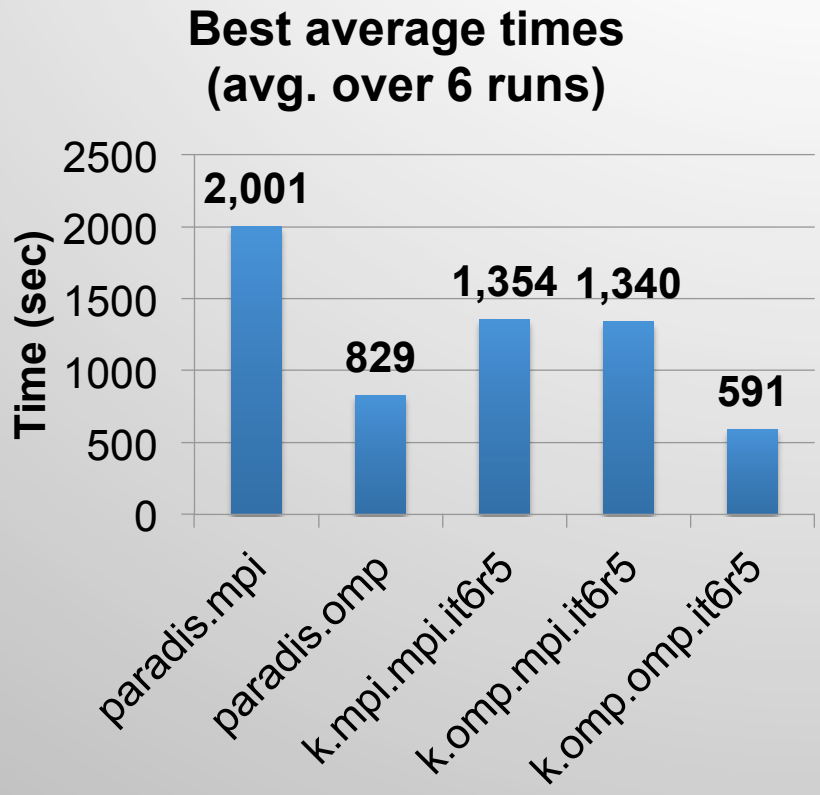
Method	Steps	Nonlin Iter	Linear Iter	Jv Eval	Fcn Eval	Run Time (s)	Est. Run Time (s)
TRAP FP I2	10,434	15,627	—	—	15,627	2,615	—
TRAP AA I7 V6	2,866	10,466	—	—	10,466	1,627	—
TRAP NK I3 $\epsilon_l 0.5$	1,544	3,386	5,875	7,643	12,804	2,135	1,157
TRAP NK I4 $\epsilon_l 0.5$	1,396	3,279	5,763	7,627	12,440	2,089	1,121
DIRK3 NK I4 $\epsilon_n 1.0$ $\epsilon_l 0.9$	483	4,041	10,029	13,909	20,647	2,953	1,519

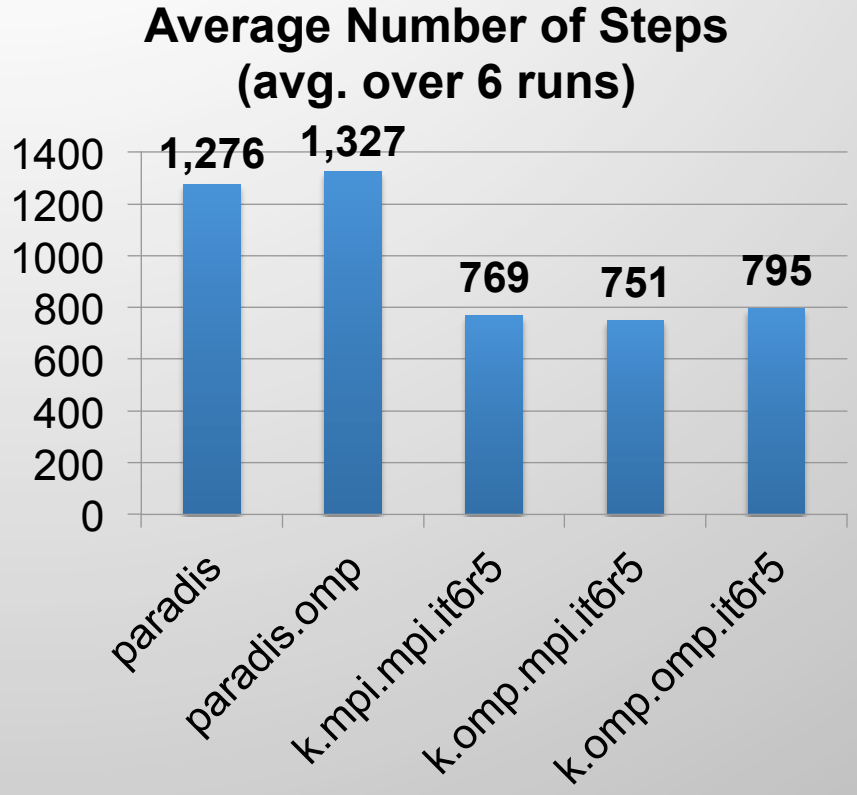
- There is a significant potential benefit with an analytic Jacobian

Large Scale Simulation on Vulcan

- Test case with 1,094,620 initial dislocation nodes
- Run 1.35 x 10^{-9} μs, starting at 1.286713 x 10^{-5} μs
- Temp = 600K; Pressure = 1 Gpa; Domain is 1.0 μm^3
- Strain rate = 10^4 s^{-1}; Tolerance = 1.25 |b|
- Run on 4,096 cores of LLNL Vulcan machine:
  - 5 Petaflop IBM Blue Gene/Q, small version of Sequoia
  - 24,576 nodes, 16 cores and 16 GB of memory per node
- Tested MPI + OpenMP parallelization with Trapezoid using the fixed point iteration and with Anderson Accelerated fixed point

Vulcan Results





ParaDiS with OpenMP threading gives speedup over MPI-only

For trapezoid with Anedrson Acceleration (I6 V5), OpenMP threading shows little benefit unless ParaDiS is also threaded

~25% speedup over OpenMP threaded ParaDiS

Large Scale Simulation on Sequoia

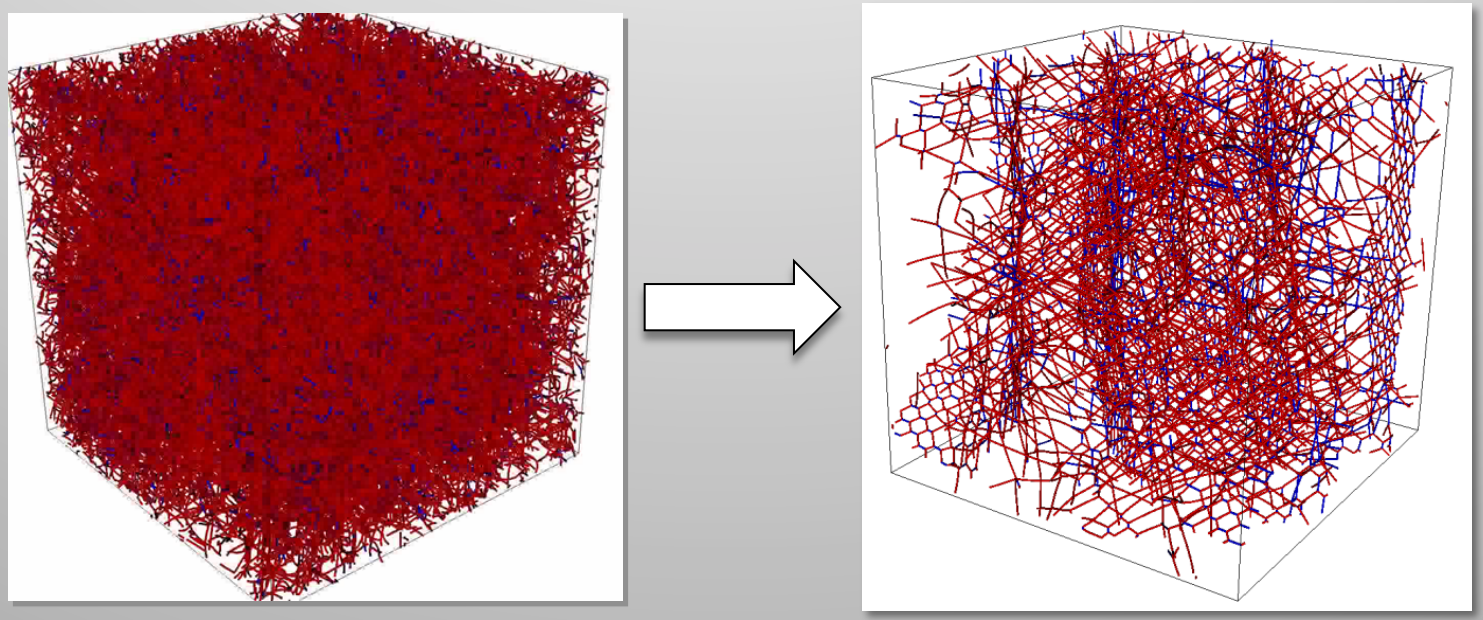
- Test case with 55,456,000 initial dislocation nodes
- Run $9.552642101 \times 10^{-9}$ μs, starting at $1.269053578987361 \times 10^{-9}$ μs
- Temp = 300K; Pressure = 0 Gpa; Domain is 1.0 μm^3
- Strain rate = 1.0 s^{-1}, Tolerance = 1.25 |b|
- Run on 262,144 cores of LLNL Sequoia machine
- MPI + OpenMP threading with 4 threads per core

	Trap FP I2	Trap AA I6 V5
Total Time	1,774 s	1,566 s
Time Steps	551	394
Avg. Step	2.23e-11 s	3.24e-11 s

Anderson Accelerated fixed point gives a **12% speedup**

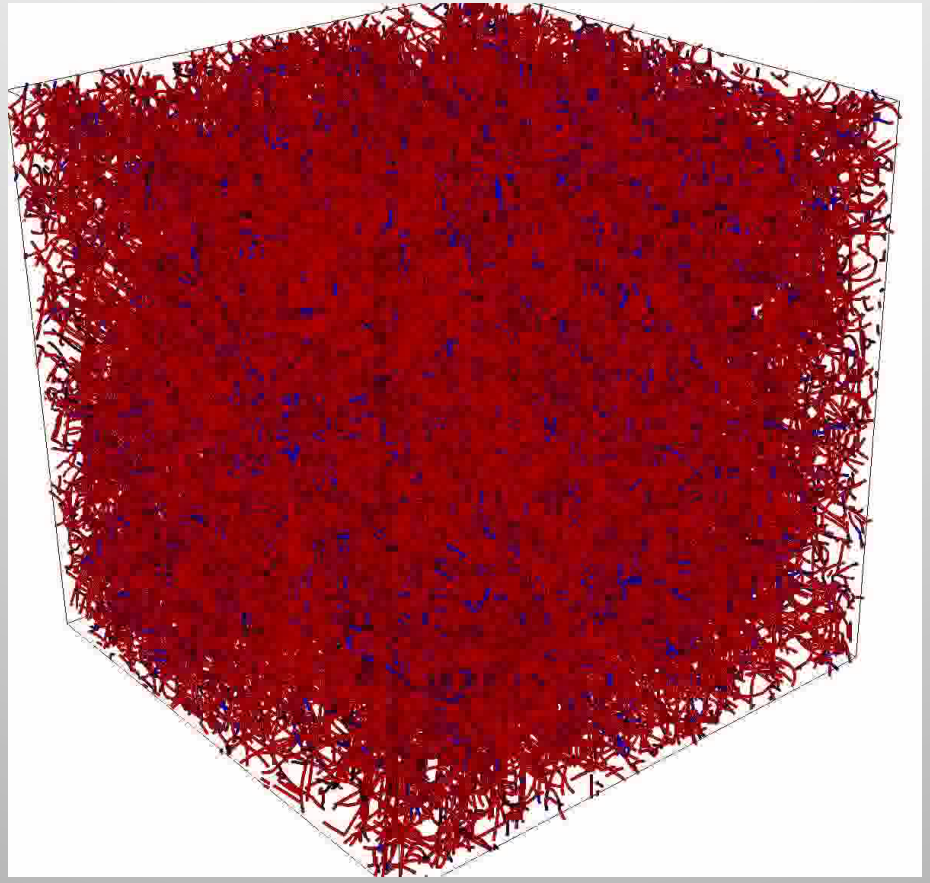
Annealing Simulation

- The numerical results thus far have focused on *strain hardening* simulations where the number of dislocations increases in time
- *Annealing* is when a metal is heated and then allowed to cool in order to remove internal stresses and increase toughness
- As a result, the number of dislocations decreases over the course of the simulation



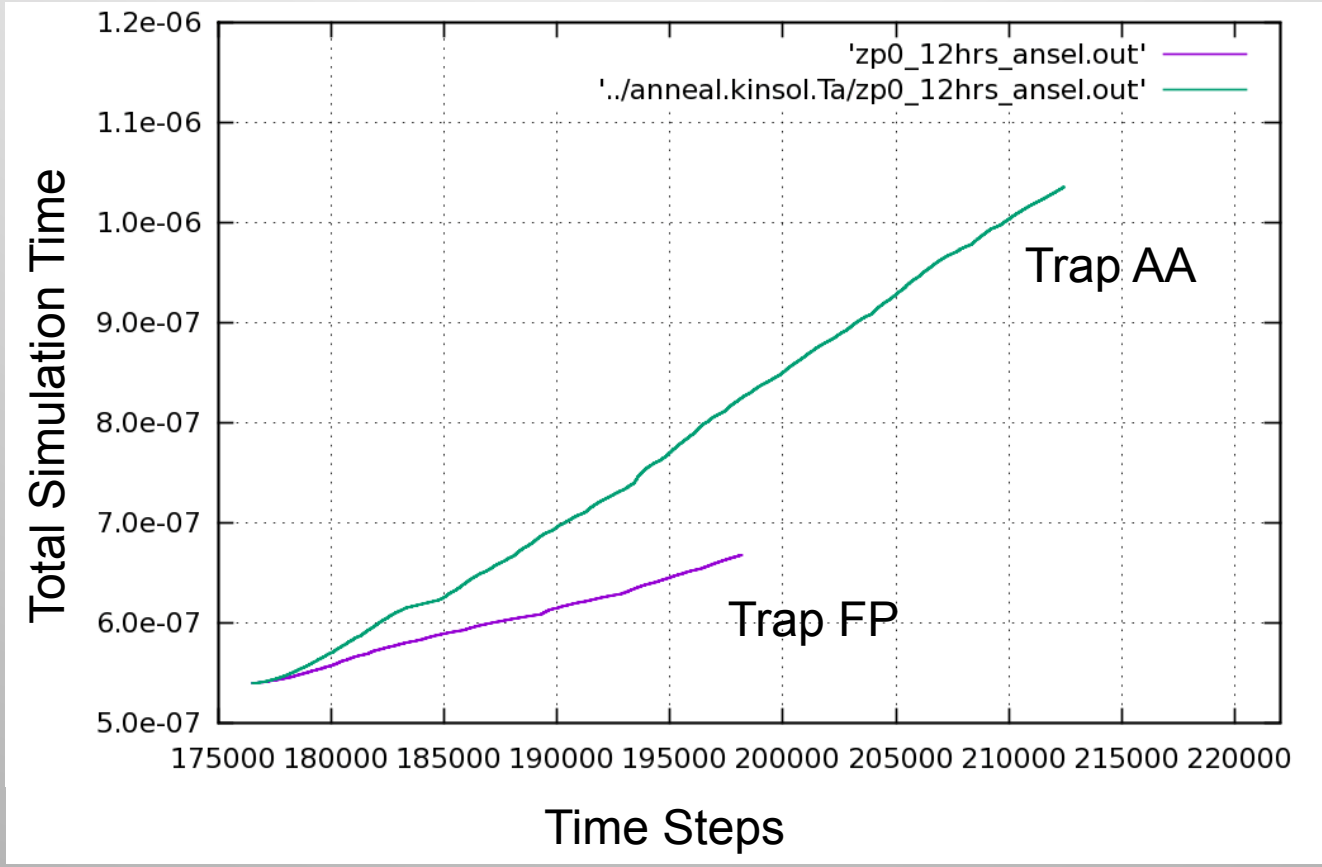
Annealing Simulation

- ~425,000 nodes to ~10,000 nodes
- 12 hour run on 256 cores of LLNL Ansel system

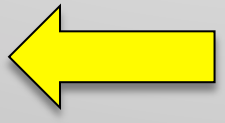


Annealing Runtime

- The trapezoid integrator with the Anderson accelerated fixed point solver gives a **~50% increase** in simulated time over 12 wall clock hours



Outline

- Dislocation Dynamics
- Time Integrators
- Nonlinear Solvers
- Numerical Results
- Conclusions 

Summary

- Simulating dislocation dynamics presents several challenges for nonlinear solvers and implicit time integrators
  - Expensive force calculations
  - Discontinuous topological events
  - Rapidly changing problem size
- To increase time step sizes and improve runtimes we interfaced with the SUNDIALS suite of codes to utilize advanced nonlinear solvers and higher-order implicit time integrators
  - KINSOL: Anderson accelerated fixed point, Newton's method
  - ARKode: High-order embedded DIRK methods
- In various test cases these approaches have shown the ability improve performance in both number of time steps and runtime versus the native trapezoid integrator with a fixed point iteration

Conclusions

- Both Anderson accelerated fixed point and higher order integrators gave significant speedup to the dislocation dynamics simulations
- Newton's method allowed for larger time steps but the additional function evaluations for Jacobian-vector products lead to slower runs
- The 3rd order DIRK method produced greatly improved runtimes in initial tests with a BCC crystal
- In large scale strain hardening and annealing tests the accelerated fixed point method is very effective and is now the *default nonlinear solver in ParaDiS*
- Current and future work focus on
  - Utilizing an analytic Jacobian in nonlinear solves with Newton
  - Investigating performance with Anderson acceleration and GPUs

Acknowledgements

Support for this work was provided through the Scientific Discovery through Advanced Computing (SciDAC) program funded by the U.S. Department of Energy Office of Advanced Scientific Computing Research.

computation.llnl.gov/casc/sundials


Lawrence Livermore
National Laboratory