



**Pacific Northwest**  
NATIONAL LABORATORY

*Proudly Operated by **Battelle** Since 1965*

# **A Novel Parallel Approach to Solving Constrained Linear Optimization Problems**

**Stephen Elbert**, [Steve.Elbert@PNNL.gov](mailto:Steve.Elbert@PNNL.gov)

Karan Kalsi, Jamie Lian, Kurt Glaesemann

Pacific Northwest National Laboratory, Richland, WA

FERC Staff Technical Conference on

Increasing Real-Time and Day-Ahead Market Efficiency through Improved Software

Session T1-B

June 24-26, 2013

Washington, DC



## Motivation – Adapt algorithm successfully used with FTR to UC, ACOPF

- ▶ Algorithm has several advantages over commercial FTR solvers
  - Scalability—scales roughly as the square of the problem size
  - Parallelizable—key operation is a (constant) matrix times vector operation
    - A constant (sparse) matrix simplifies load balancing
    - Maintains numerical precision
    - Time per iteration is constant—no backfilling of matrix
  
- ▶ Algorithm is LP solver with limits. Can it be adapted to more generalized LP problems?
  - Linearized ACOPF
  - LP solver within the Branch-and-Bound portion of a MIP solver (Unit Commitment)



## PNNL Algorithm – Parallel Adaptive Dynamical System (PADS)

- ▶ Transform LP into coupled set of non-linear dynamical equations
- ▶ Dynamical system may converge to stable states which are solutions of primal and dual LP problems respectively

### Non-linear Dynamical System

#### Primal

maximize  $c^T x$   
subject to  $Ax \leq b$  and  $x \geq 0$

#### Dual

minimize  $b^T y$   
subject to  $A^T y \geq c$  and  $y \geq 0$



$$\frac{dx}{dt} = k_1 \left( c - A^T \left( y + k \frac{dy}{dt} \right) \right)$$
$$\frac{dy}{dt} = k_2 \left( -b + A \left( x + k \frac{dx}{dt} \right) \right)$$

$$k_1 = \frac{K}{i} \quad i = 1, 2, \dots, M \quad k_2 = \frac{1}{k_1}$$

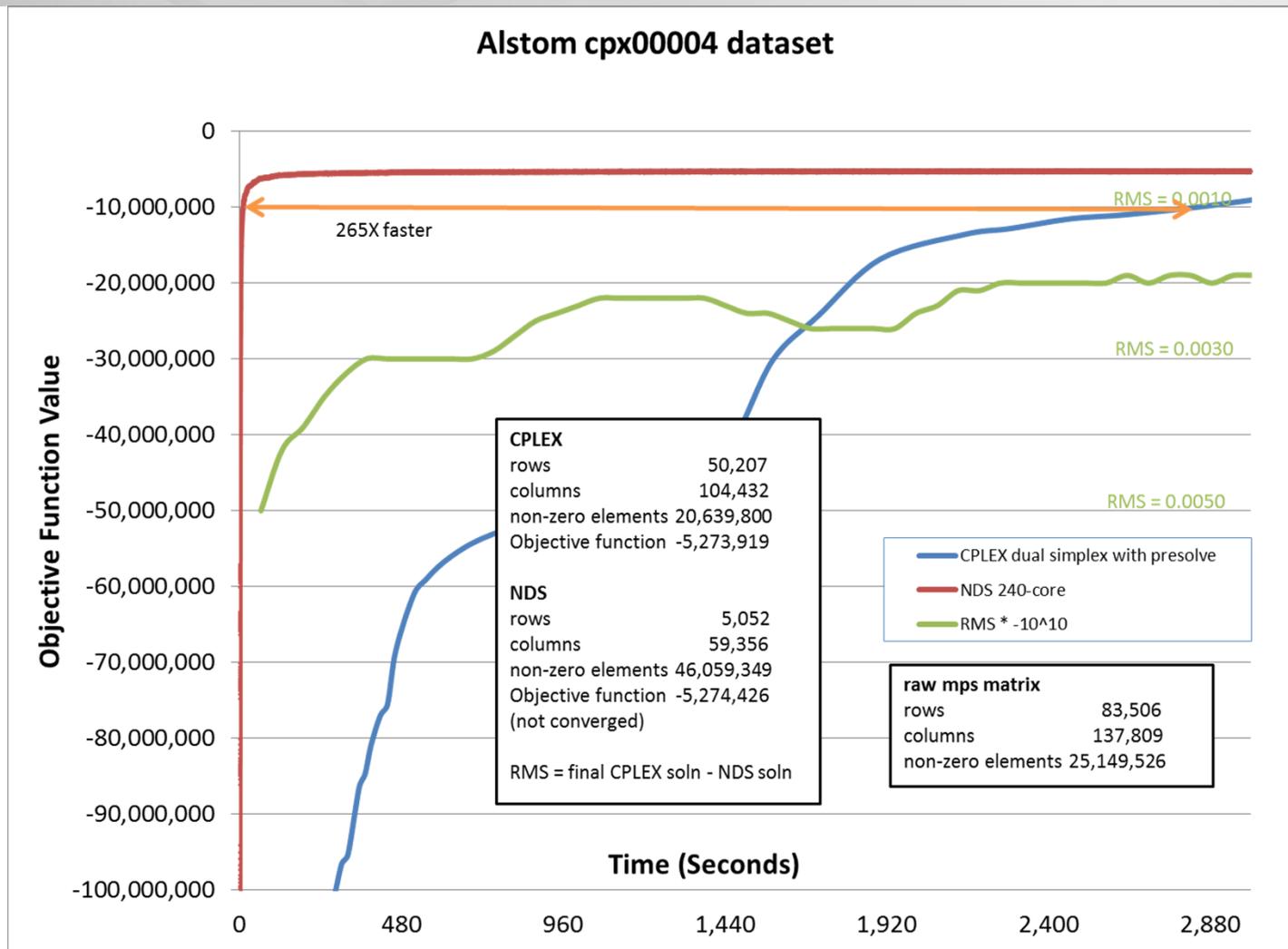
- ▶ Kernel is a pair of easily parallelized matrix-vector operations: scale as square of problem size (constraints x variables)
- ▶  $K, M$ , and  $dt$  are input (tuning) parameters



## Implementation notes

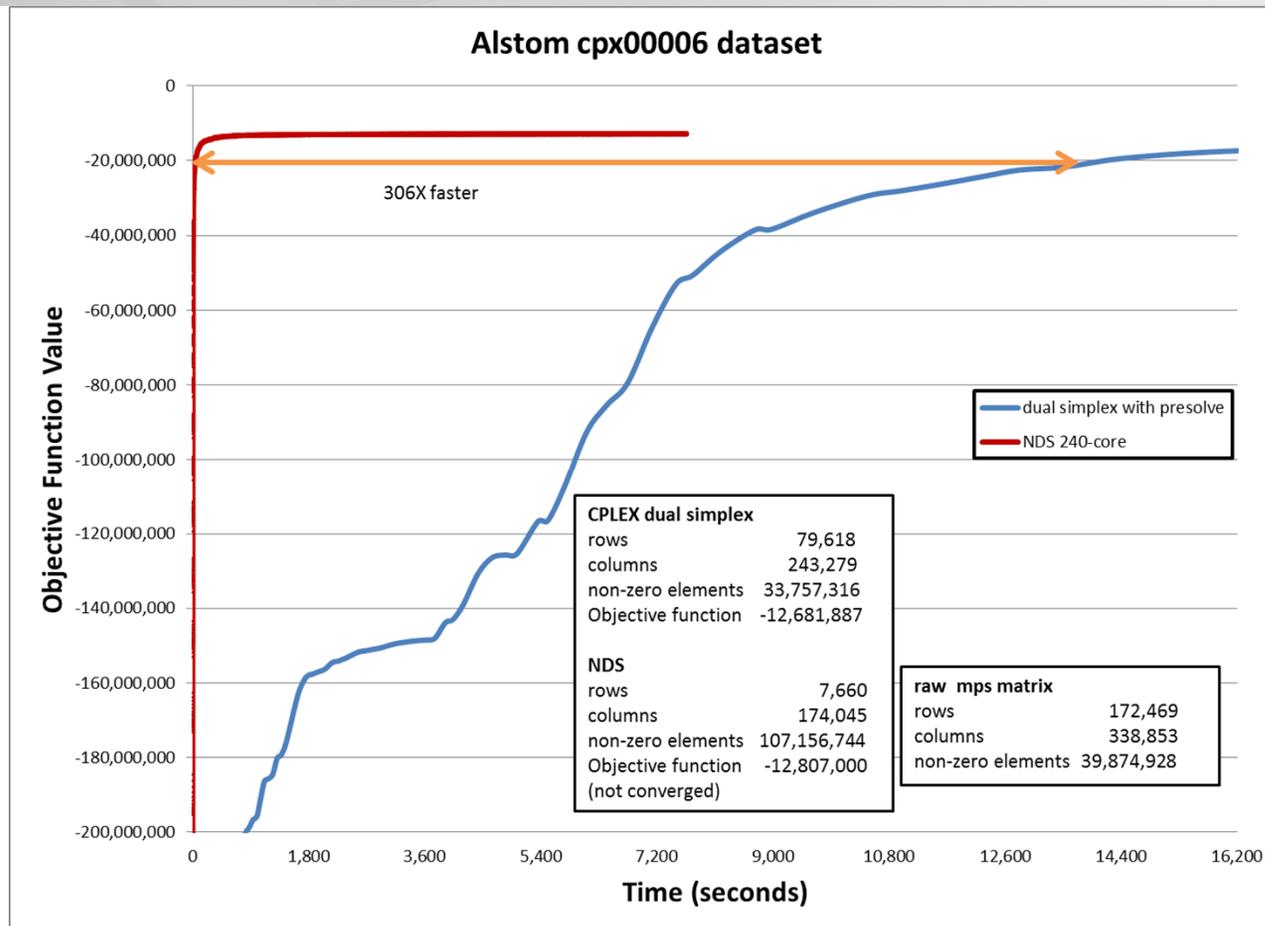
- ▶  $A$  matrix segments communicated once at beginning—the matrix doesn't change.
- ▶ Non-zero elements do not move so load balancing can be predetermined
- ▶  $A$  and  $A^T$  stored separately to maximize unit stride access
- ▶ Global sums for product vectors and distribution of  $x$  and  $y$  vectors are the only communication after initialization
- ▶ Data may be loaded efficiently in parallel

# Real-world data 1 (last year's results)



CPLEX time per iteration slows by 85x from beginning to end due to backfill

# Real World Data 2 (last year's results)



CPLEX time per iteration slows by 269x from beginning to end due to backfill



## Characteristics and Issues

- ▶ At steady-state, primal and dual objective functions yield the same value.
- ▶ Given a solution, the method recognizes it as such and does not change it.
- ▶ Can the method reach steady-state?
  - Understanding the ODEs is an open problem
  - ODEs describe the flow, which is somewhat smooth
  - The constraints introduce discontinuities
- ▶  $x \geq 0$  and  $y \geq 0$  limit the class of problem that can be solved.
  - How serious is this limitation? (not a problem so far)
  - Can it be removed? (probably, with an exponential transformation)
- ▶ Is it useful for ACOPF and UC problems?
  - ACOPF: only applicable to linearized approaches
  - UC: MIP can be represented as branch-and-bound with LP
    - Toy problems successfully solved with MATLAB version of PADS (serial)

# Approach

- ▶ Place PADS in the context of an external optimization infrastructure to improve access to problems
- ▶ For MIP, adapt PADS for use with ParaSCIP
  - Developed at Konrad-Zuse-Zentrum für Informationstechnik Berlin
  - Parallel extension of SCIP (Solving Constraint Integer Programs)
  - Used 2,048 cores to solve two open instances in MIPLIB
    - 86 hours to solve ds (1.17 billion nodes)
    - 114 hours to solve stp3d (14.8 million nodes)
  - Used CPLEX 12.1 as underlying LP solver but also has internal LP solver SOPLEX
  - Currently developing an interface between ParaSCIP and PADS
- ▶ Use GAMS as modeling language
  - GAMS uses many solvers including CPLEX and SCIP
  - Initially will use modified version of SCIP/PADS with GAMS
  - May eventually provide direct PADS interface with GAMS
  - FERC models use GAMS



# Understanding the ODEs: The Predator-Prey Model

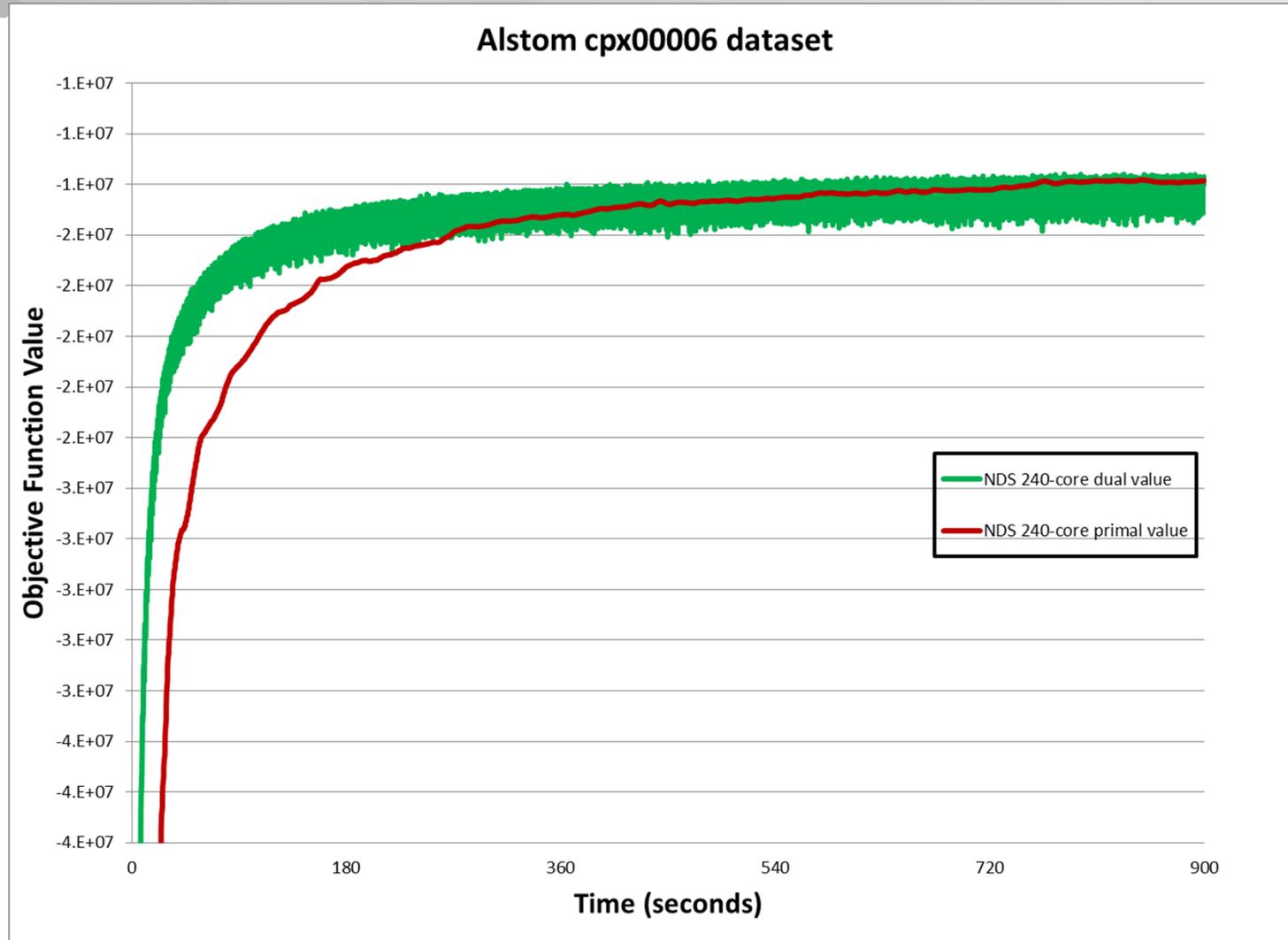
- ▶ Lotka-Volterra equations

$$\frac{dx}{dt} = fx - Axy$$

$$\frac{dy}{dt} = -gy + Bxy$$

- ▶ The two populations oscillate when not at an equilibrium point.
- ▶ Potentially useful in understanding the dynamics of the ODE, but the constraints complicate matters.

# Holding up convergence: Dual objective





# Implicit Method

$$\frac{dx}{dt} = A \frac{dx}{dt} + Bx + C$$

- ▶ Solve for  $\frac{dx}{dt}$
- ▶ Constraints must be carefully managed
- ▶ It works for the small IEEE test cases tested with MATLAB
- ▶ Current MATLAB implementation very inefficient

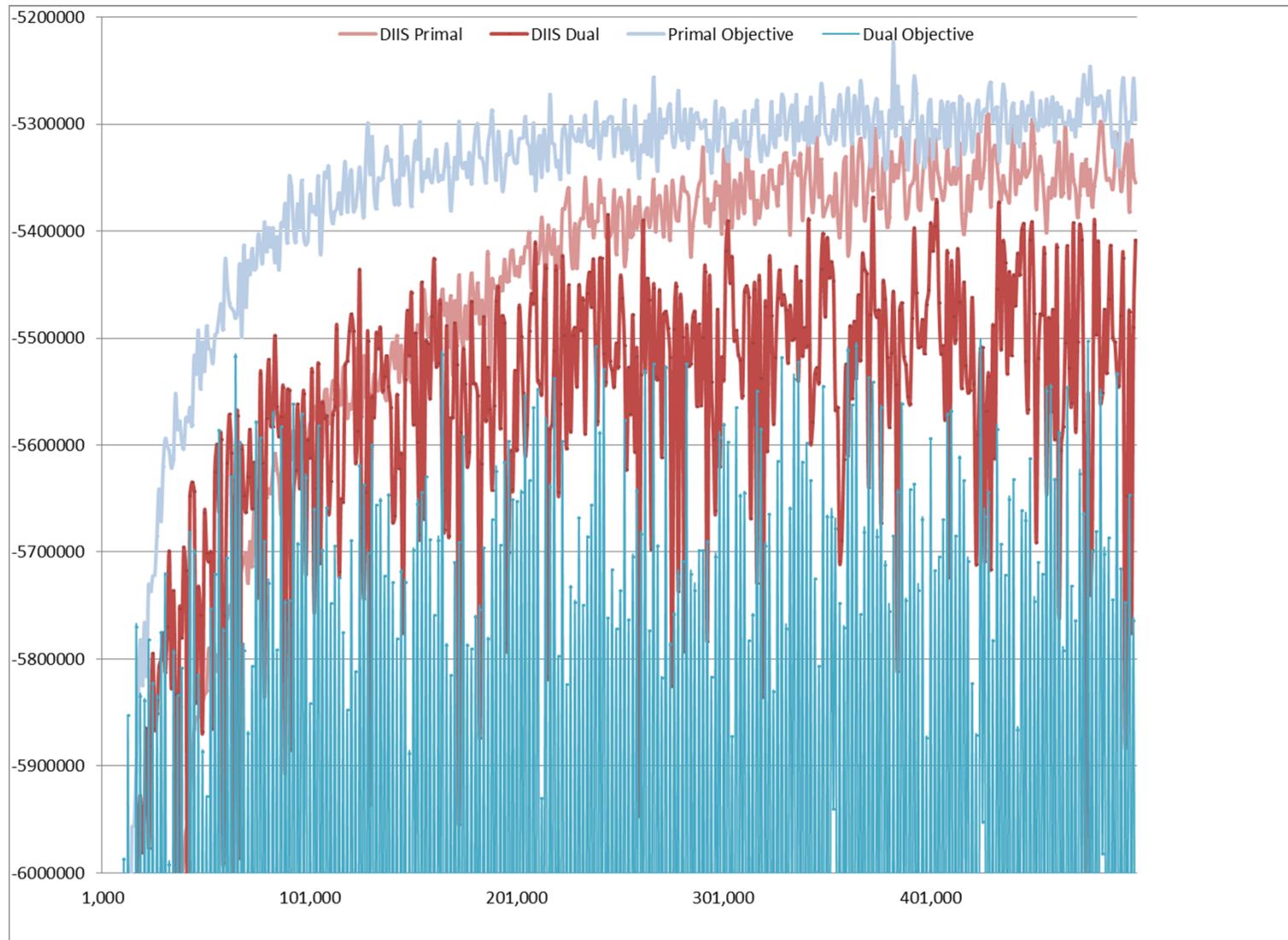
# Direct Inversion in the Iterative Subspace (DIIS)

- ▶ An acceleration technique for solvers of nonlinear problems (Pulay 1980)
- ▶ Can be interpreted as a quasi-Newton method in which the Jacobian is approximated by finite differences
- ▶ Corresponds to a projected backward Broyden's method (Broyden 1965, 1973)
- ▶ Implementation with PADS

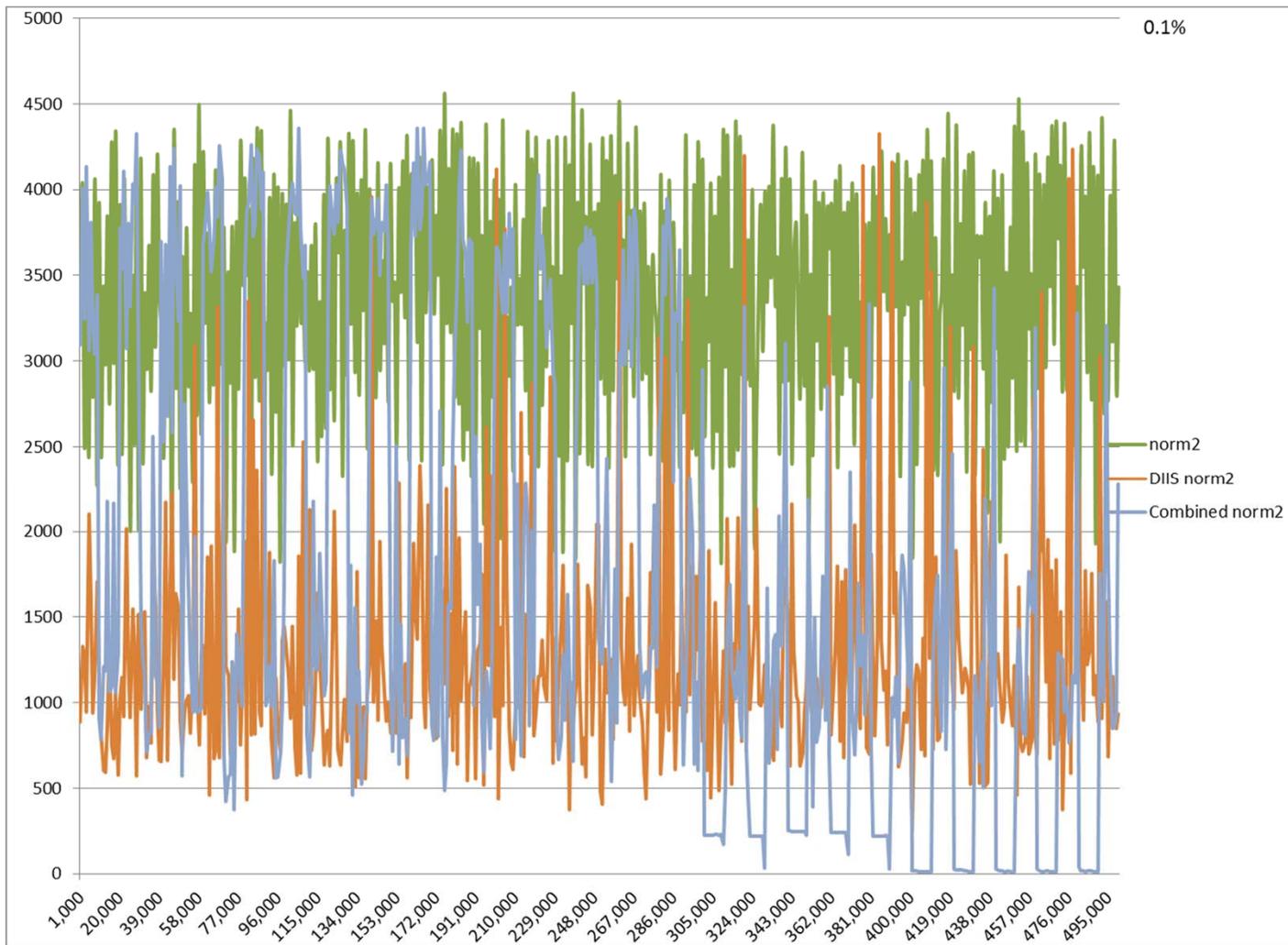
Minimize predicted  $\Delta X$  in  $X_{n+1} = X_n + \Delta X$   
by assuming that  $\Delta X = \sum_i \alpha_i \Delta X_i$   
when  $X = \sum_i \alpha_i X_i$

- Both Y and Y done, but separately
- $\alpha$  weight values are determined with DIIS
- Skip DIIS if any  $\alpha$  component is huge to prevent instability
- Deals with constraints approximately

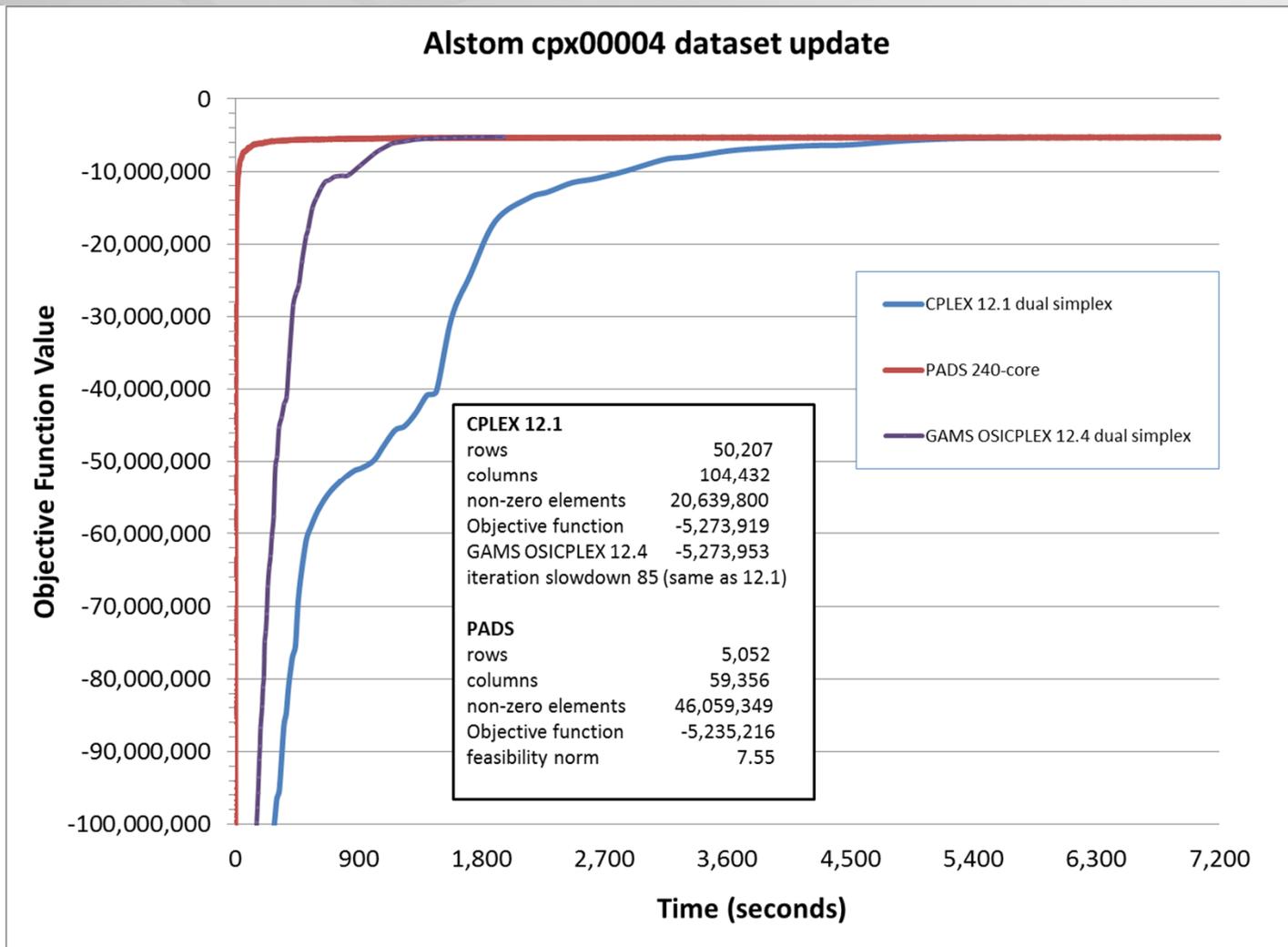
# Effect of using DIIS on cpx00004



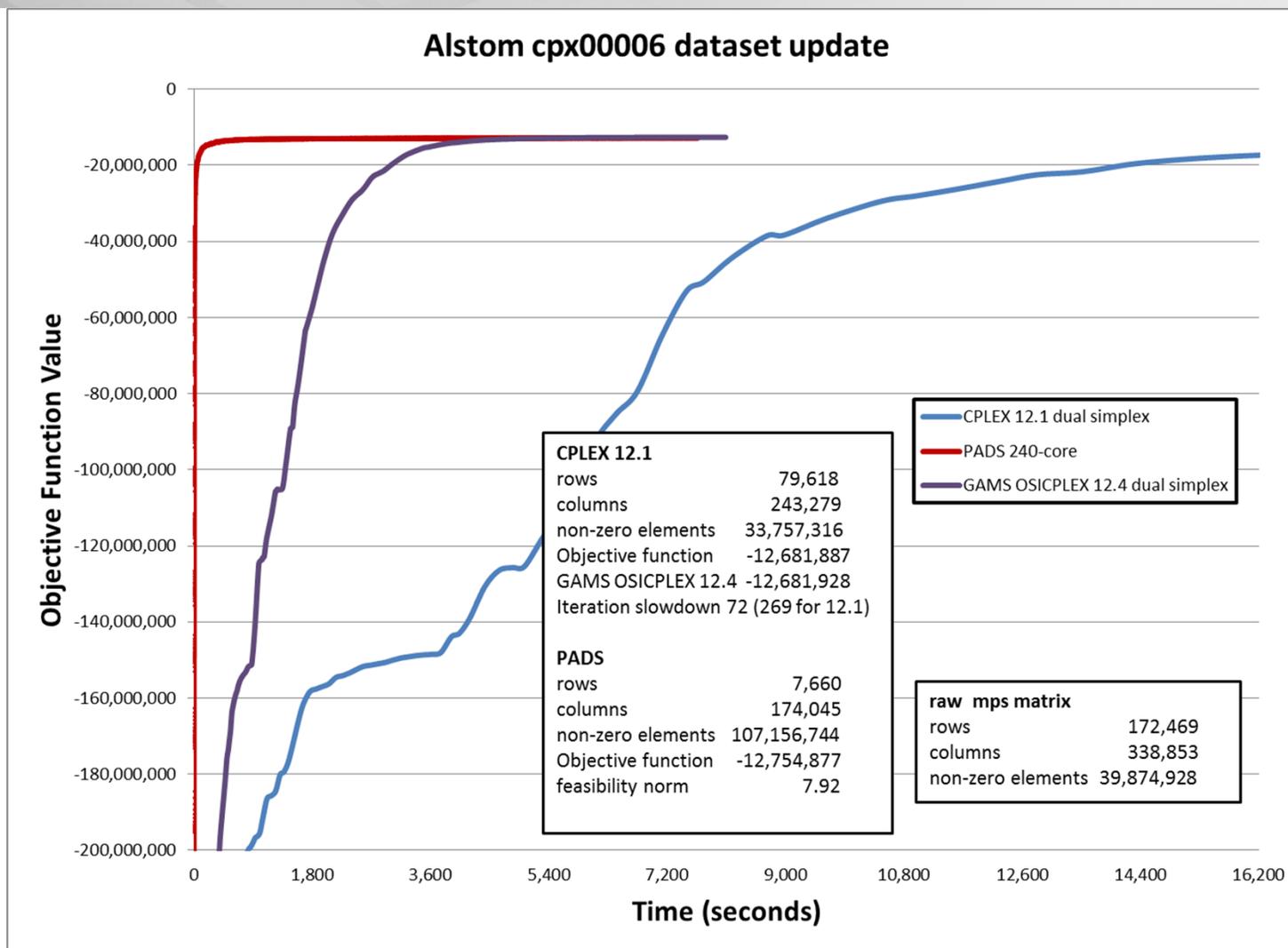
# Infeasibility norms



# Meanwhile...



And...





## Summary

- ▶ Developed novel dynamical system linear programming (LP) solver for use with FTR
- ▶ Easily parallelized to solve large FTR problems
- ▶ More computationally efficient than CPLEX for LP (up to a point)
  - Computational kernel of dual simplex is (sparse) linear solver (LU) that scales as cube of problem size
  - PADS kernel is matrix-vector multiplication that scales as square
  - PADS avoids backfill (filling in zeros) of coupled blocks
- ▶ Implicit method converges to arbitrary accuracy but is slow and only tried on small problems.
- ▶ DIIS method helps a little in some cases
- ▶ Using the PADS algorithm as the LP solver on small MIP problems was successful.
- ▶ Adapting PADS to more general optimization infrastructure
  - Will allow testing of larger problems to see if limitations are fatal or not

# Acknowledgements

- ▶ PNNL colleagues:
  - K. Kalsi (algorithm), K. Glaesemann (parallel programming, DIIS), N. Zhou (FTR), M. Vlachopoulou (CPLEX, GAMS), M. Rice (Power Flow), J. Lian (Implicit Method), Barry Lee (Maths)
- ▶ For providing Alstom data sets
  - Xing Wang (Alstom-Grid)
  - David Sun (Alstom-Grid)

## Questions?