Distributed Optimization
Alternate Direction Method of Multipliers

Swapnil Saha, Brian B Cheng

ECE 509: Convex Optimization

Outline

- Alternating direction method of multipliers (ADMM): Why it is powerful
- Background: Dual Ascent and Method of Multipliers
- Consensus Optimization
- ADMM applied to LASSO
- Distributed ADMM using MPI

Why ADMM is Powerful

- Modular structure
  Decomposition of a large-scale optimization across objective components

- Good robustness of convergence
  Restrictive convergence conditions and faster convergence rates

Why ADMM is Powerful

Decomposition of Dual Ascent

+

Robustness of Method of Multipliers

Robust Dual Decomposition

Dual Problem

$$\begin{array}{ll} \text{minimize} & f(x) \\ \text{subject to} & Ax = b \end{array}$$

- Lagrangian $L(x, y) = f(x) + y^\top (Ax - b)$
- Dual Function $g(y) = \inf_x L(x, y)$

$$\left.\begin{array}{l} y^* = \operatorname*{argmax}_y g(y) \\ \\ x^* = \operatorname*{argmin}_x L(x, y^*) \end{array}\right\} \text{Strong Duality}$$

Dual Ascent

$$\begin{array}{ll} \text{minimize} & f(x) \\ \text{subject to} & Ax = b \end{array}$$

- Variable Update

$$x^{k+1} := \underset{x}{\operatorname{argmin}}\, L(x, y^k),$$
$$y^{k+1} := y^k + \alpha^k (Ax^{k+1} - b).$$

Dual Decomposition

- f is separable $f(x) = f_1(x_1) + \cdots + f_N(x_N), \quad x = (x_1, \ldots, x_N)$
- L separable wrt x $L(x, y) = \sum_{i=1}^{N} L_i(x_i, y) = \sum_{i=1}^{N} \left(f_i(x_i) + y^\top A_i x_i - \frac{1}{N} y^\top b \right)$

$$\left.\begin{aligned} x_i^{k+1} &:= \underset{x_i}{\operatorname{argmin}}\, L_i(x_i, y^k) \\ y^{k+1} &:= y^k + \alpha^k (A x^{k+1} - b). \end{aligned}\right\}$$

x updates locally
y updates globally

Method of Multipliers

- Add robustness to solving dual problem

- Use augmented Lagrangian

$$L_\rho(x, y) = f(x) + y^\top(Ax - b) + \frac{\rho}{2}\|Ax - b\|_2^2$$

- Dual Function $\quad g_\rho(y) = \inf_x L_\rho(x, y).$ $\Rightarrow$ Differentiable under rather mild conditions

Method of Multipliers

- Improvement of convergence comes with the cost of losing decomposability

$$L_\rho(x, y) = f(x) + y^\top (Ax - b) + \frac{\rho}{2}\|Ax - b\|_2^2$$

ADMM hits this drawbacks by introducing new variable and function

No longer sparable even though f(x) separable

ADMM Method

$$\begin{array}{ll} \text{minimize} & f(x) + g(z) \\ \text{subject to} & Ax + Bz = c \end{array}$$

- Variable z makes the Augmented Lagrangian separable.

- Augmented Lagrangian

$$L_p(x, y, z) = f(x) + g(z) + y^T(Ax + bz - c) + (p/2)\,\|Ax + Bz - c\|_2^2$$

ADMM : Variable Update

$$
\begin{aligned}
x^{k+1} &:= \operatorname{argmin}_x L_\rho(x, z^k, y^k) && \text{// } x\text{-minimization} \\
z^{k+1} &:= \operatorname{argmin}_z L_\rho(x^{k+1}, z, y^k) && \text{// } z\text{-minimization} \\
y^{k+1} &:= y^k + \rho(Ax^{k+1} + Bz^{k+1} - c) && \text{// dual update}
\end{aligned}
$$

- Alternatingly update primal variable x, z
- Variable z makes it decomposable for variable x, y
- z works like a global variable.

ADMM : Decomposability

- Consensus Optimization: Objective function is distributed through N nodes.

$$\text{minimize} \quad \sum_{i=1}^{N} f_i(x)$$

- One "single" x collaboratively decrease overall objective function.

- ADMM Format: Distributed the variable x to each nodes.

$$\begin{array}{ll} \text{minimize} & \sum_{i=1}^{N} f_i(x_i) \\ \text{subject to} & x_i - z = 0 \end{array} \Longrightarrow$$

- Creating Local Variable
- Constraint makes them agree to each other

Consensus Problem with ADMM

- Augmented Lagrangian

$$L_p(x_1, ...x_N, z, y) = \sum_{i=1}^{N}(f_i(x_i) + y_i^T(x_i - z) + (\rho/2)\|x_i - z\|_2^2)$$

- ADMM Update: Subscript i indicating i-th local node

$$x_i^{k+1} := \arg\min_{x_i}\left(f_i(x_i) + y_i^k T(x_i - z_k) + \frac{\rho}{2}\|x_i - z_k\|_2^2\right)$$

$$z_{k+1} := \frac{1}{N}\sum_{i=1}^{N}\left(x_i^{k+1} + \frac{1}{\rho}y_i^k\right)$$

$$y_i^{k+1} := y_k^i + \rho\left(x_i^{k+1} - z^{k+1}\right)$$

Intuition of ADMM Update

- ADMM update can be rewritten as follows

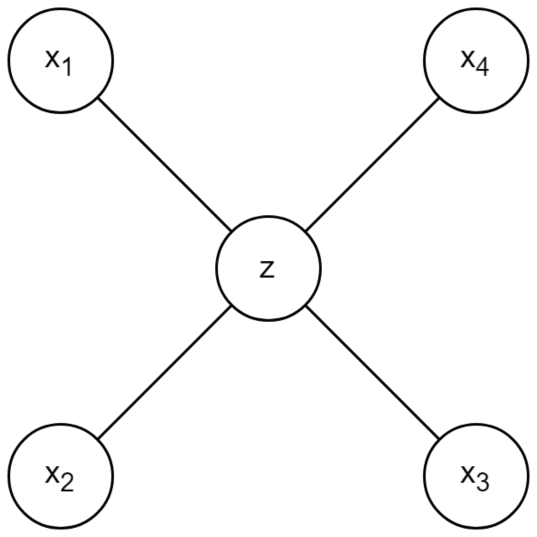
$$z^k = \overline{x^k}$$

$$x_i^{k+1} := \operatorname{argmin}_{x_i} \left(f_i(x_i) + y_i^{kT}(x_i - \bar{x}^k) + \frac{\rho}{2}\|x_i - \bar{x}^k\|^2 \right)$$

$$y_i^{k+1} := y_i^k + \rho(x_i^{k+1} - \bar{x}^{k+1})$$

1. Central Node z: Collect x_i and calculate the average $\overline{x^k}$
2. $\overline{x^k}$ is distributed to each local node.
3. Each local variable updates x_i^k
4. Update local dual variable y_i^k

Step 3 & 4 can be done in parallel



Consensus Setup

ADMM Applied to LASSO Regression

ADMM General Form:

$$\begin{array}{ll} \text{minimize} & f(x) + g(z) \\ \text{subject to} & Ax + Bz = c \end{array}$$

LASSO Objective Function:

$$(1/2)\|Ax - b\|_2^2 + \lambda\|x\|_1$$

$$\begin{array}{ll} \text{minimize} & f(x) + g(z) \\ \text{subject to} & x - z = 0, \end{array}$$

$$(1/2)\|Ax - b\|_2^2 + \lambda\|z\|_1$$

ADMM LASSO Problem:

$$\begin{array}{ll} \text{minimize} & (1/2)\|Ax - b\|_2^2 + \lambda\|z\|_1 \\ \text{subject to} & x - z = 0 \end{array}$$

ADMM Lasso Algorithm

Lasso Obj. Function: f(x) + g(z) = $(1/2)\|Ax - b\|_2^2 + \lambda\|z\|_1$

Augmented Lagrangian: $L_\rho(x, z, y) = f(x) + g(z) + y^T(Ax + Bz - c) + (\rho/2)\|Ax + Bz - c\|_2^2$

A = I, B = -I, c = 0

ADMM Algorithm Applied to LASSO:

$$\begin{aligned} x^{k+1} &:= (A^TA + \rho I)^{-1}(A^Tb + \rho z^k - y^k) \\ z^{k+1} &:= S_{\lambda/\rho}(x^{k+1} + y^k/\rho) \\ y^{k+1} &:= y^k + \rho(x^{k+1} - z^{k+1}) \end{aligned}$$

// Soft thresholding operator

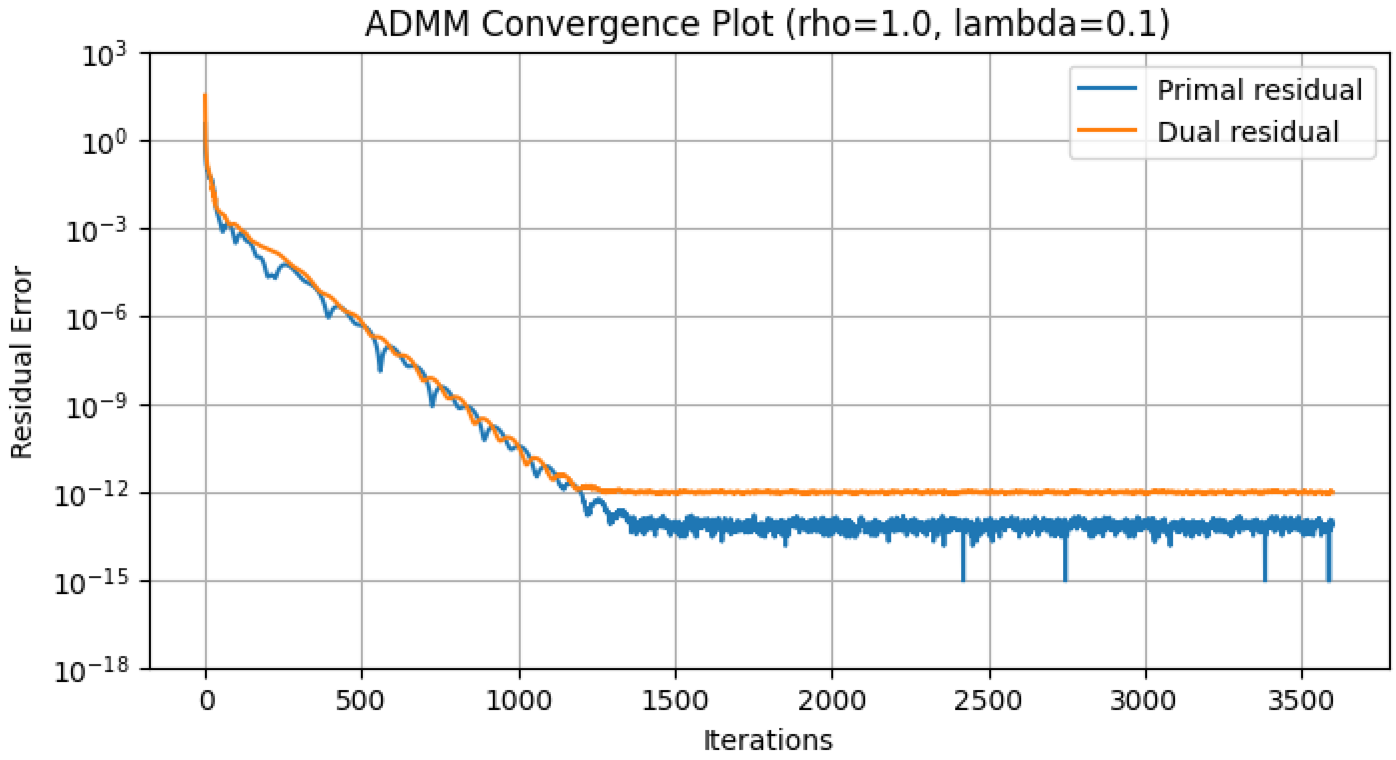
Primal and Dual Residuals:

$$r^{k+1} = x^{k+1} - z^{k+1}$$

$$s^{k+1} = \rho\left(z^{k+1} - z^k\right)$$

Simulation: ADMM Lasso Experiment

- Most simple case
  - Single node, no distribution
  - A = 1200x1200 matrix
    - Dense: All elements generated with normal distribution
- Slow convergence
  - But fast enough for many practical applications
  - Can vary with properties of the input matrix (tallness, wideness, sparsity, etc.)



Distributed ADMM Lasso Algorithm

Lasso Obj. Function: $\frac{1}{2}\|A_i x_i - b_i\|_2^2 + \lambda\|z\|_1$

Dist. Augmented Lagrangian: $L_\rho(x, z, y) = \sum_{i=1}^N \left(f_i(x_i) + y_i^T(x_i - z) + (\rho/2)\|x_i - z\|_2^2\right)$

Distributed ADMM Applied to LASSO:

$$
\left\{
\begin{aligned}
x_i^{k+1} &:= (A_i^T A_i + \rho I)^{-1}(A_i^T b_i + \rho(z^k - u_i^k)). \\
z^{k+1} &:= S_{\lambda/\rho N}(\overline{x}^{k+1} + \overline{u}^k) \\
u_i^{k+1} &:= u_i^k + x_i^{k+1} - z^{k+1}.
\end{aligned}
\right.
$$

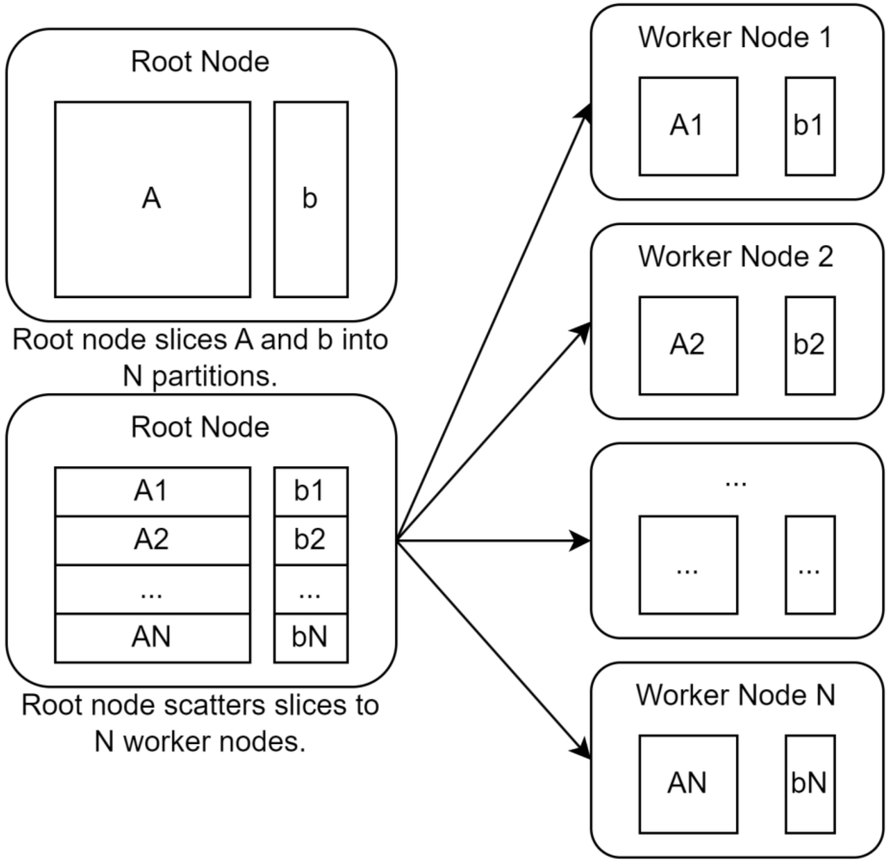
// soft threshold of x_avg and u_avg

Primal and Dual Residuals:

$$
\left\{
\begin{aligned}
r_i^{k+1} &= x_i^{k+1} - z^{k+1} \\
s_i^{k+1} &= \rho\left(z^{k+1} - z^k\right)
\end{aligned}
\right.
$$

Message Passing Interface (MPI)

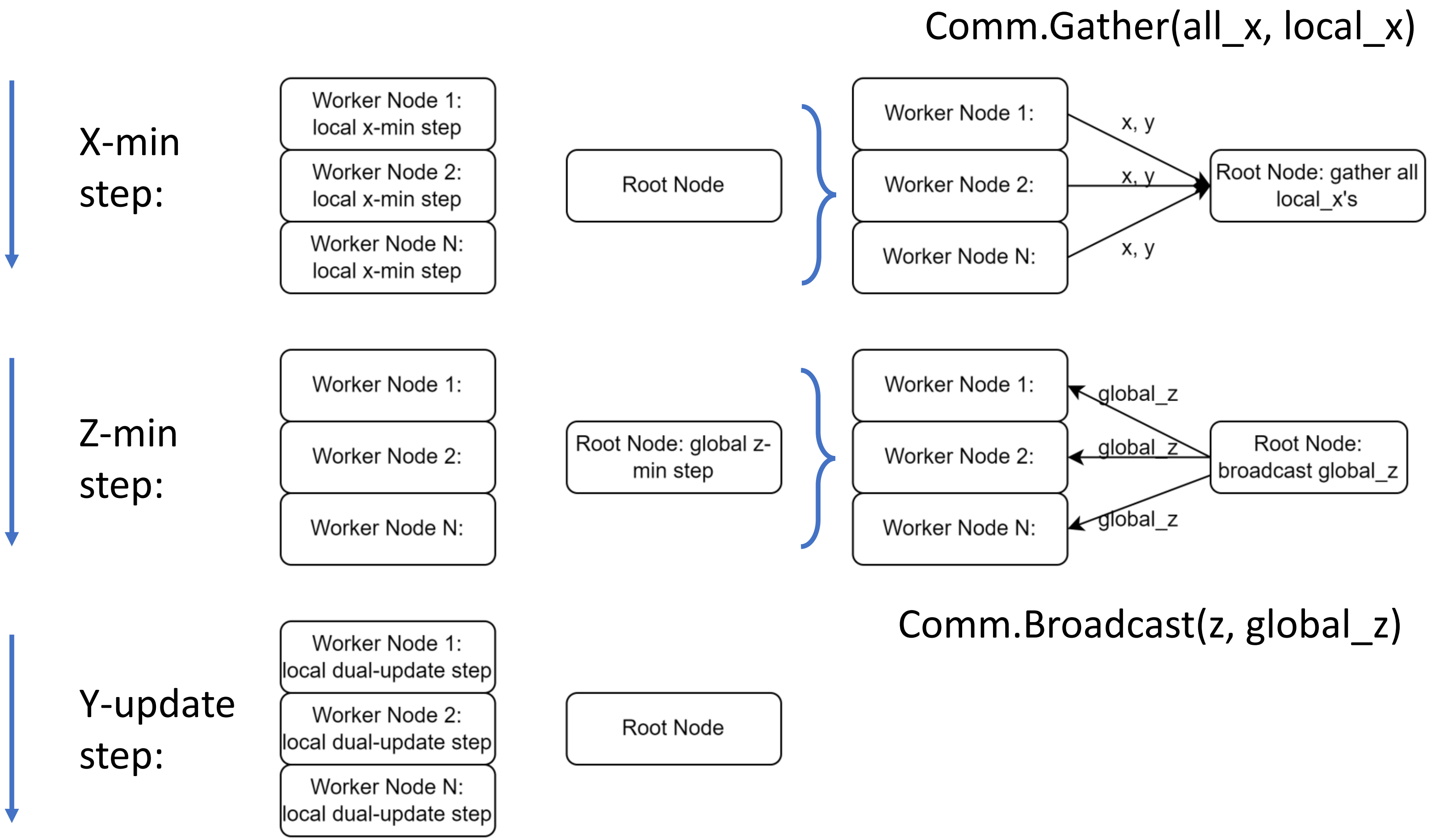
- Before the algorithm starts, the root node broadcasts partitions of the input matrix A and vector b to each of the worker nodes.
- Here, the input matrix is split among examples.



Comm.Scatter(A, local_A)
Comm.Scatter(b, local_b)

Global Consensus

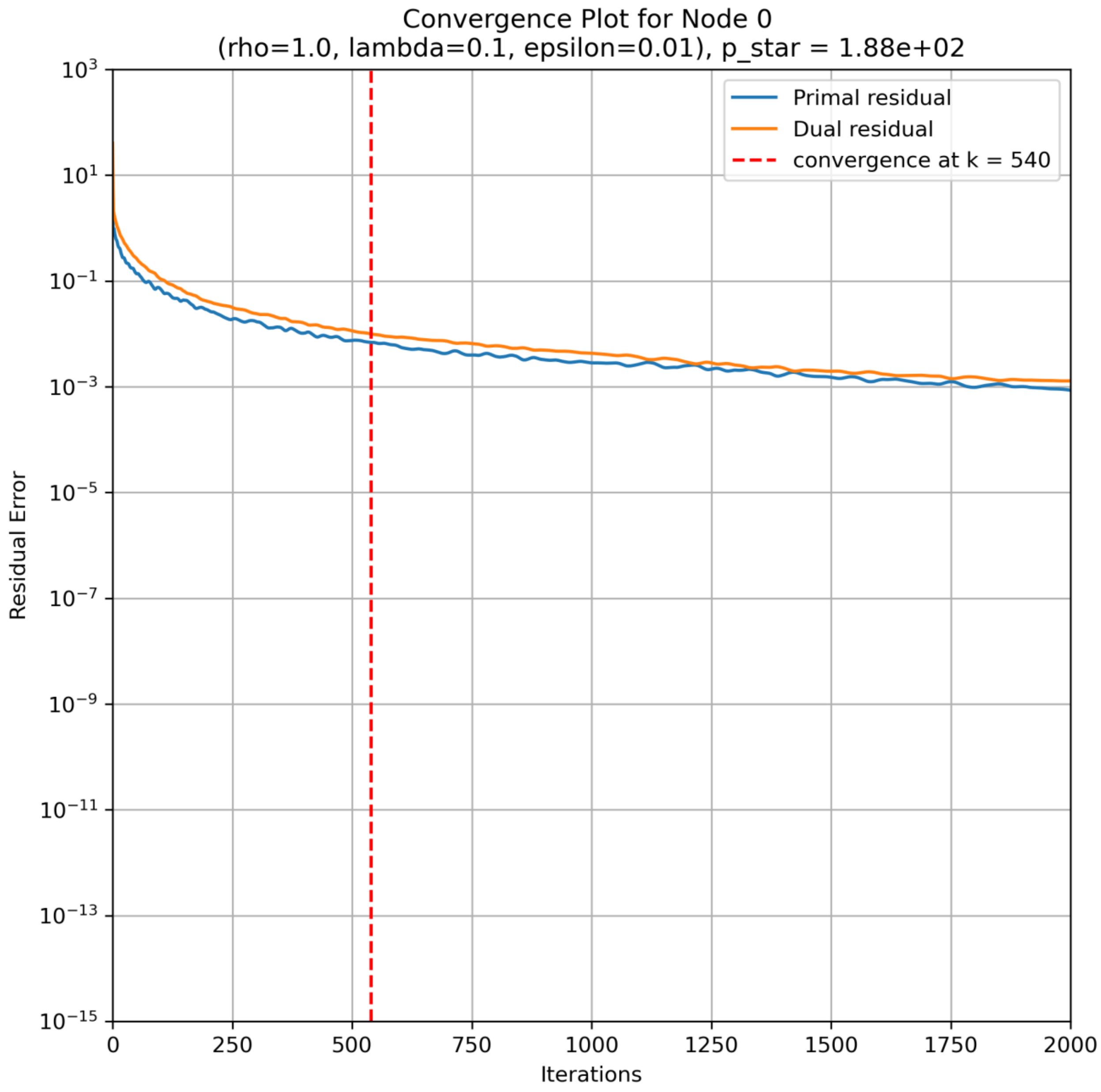
- The worker nodes have their local partitions of the dataset and work on their own local x-min and y-min steps.
- The z-min step acts as the global consensus step



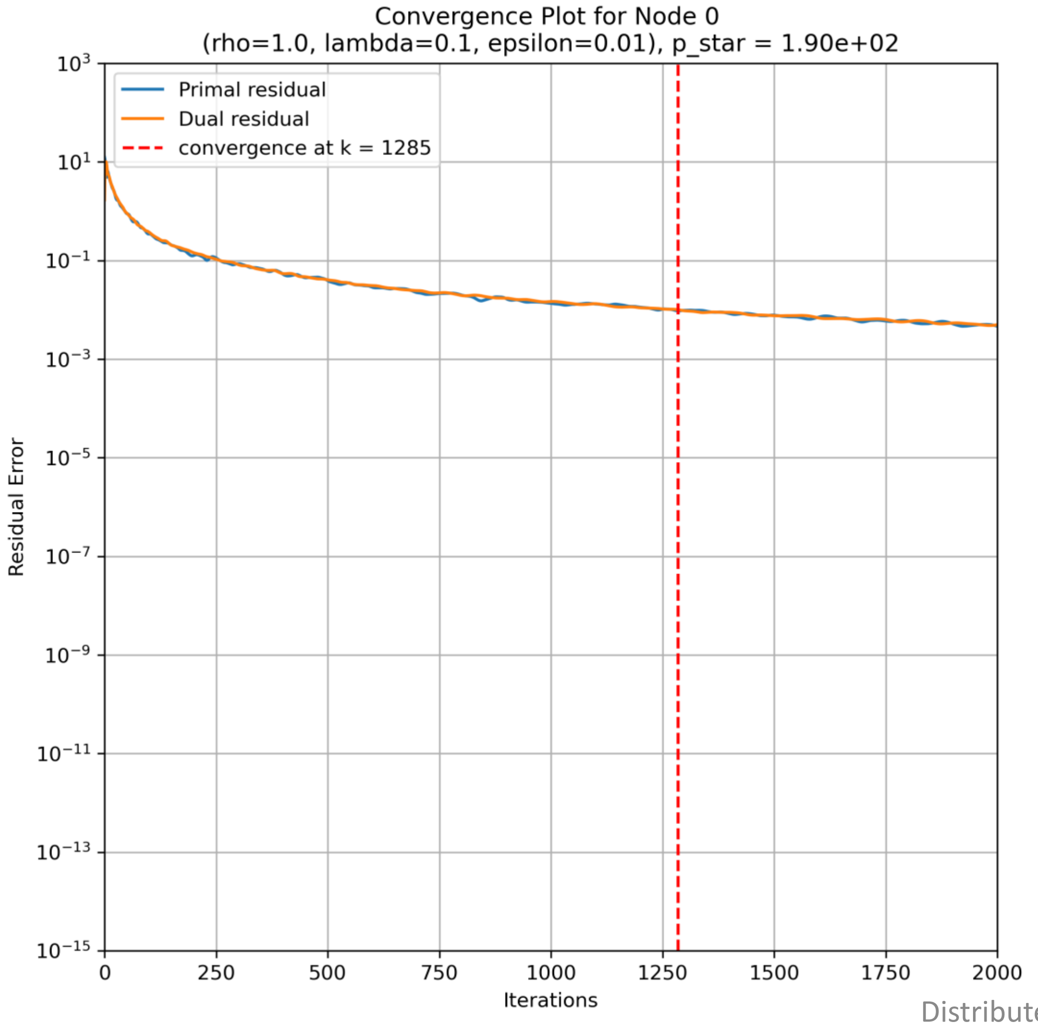
Check stopping criterion.
Loop if criterion not met

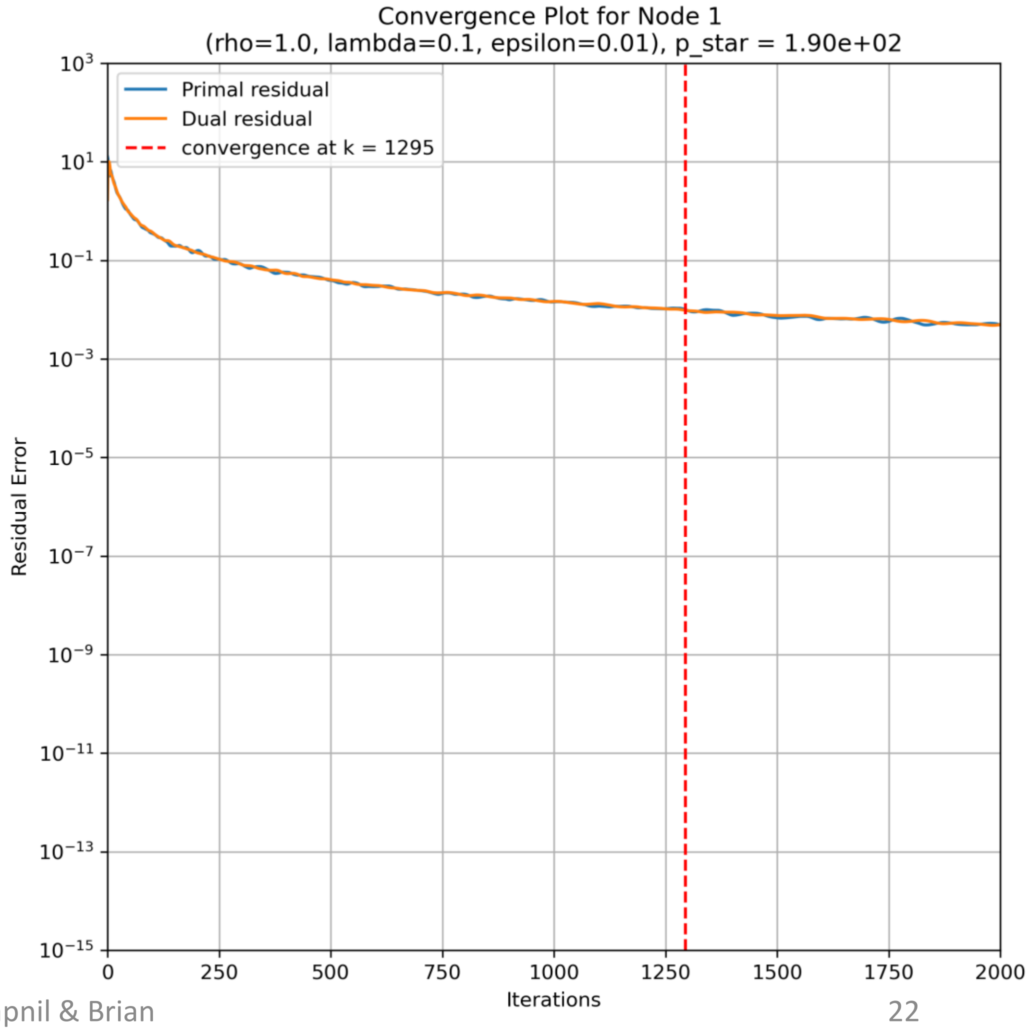
Single Node Simulation Result

- Before running distributed test, run a single node test as validation.
- LASSO Regression Single Node Test:
  - Dataset A = 2000x4000
    - Wide, Dense
  - Dataset splitting strategy:
    - Split across examples
    - Each node receives a horizontal partition of dataset

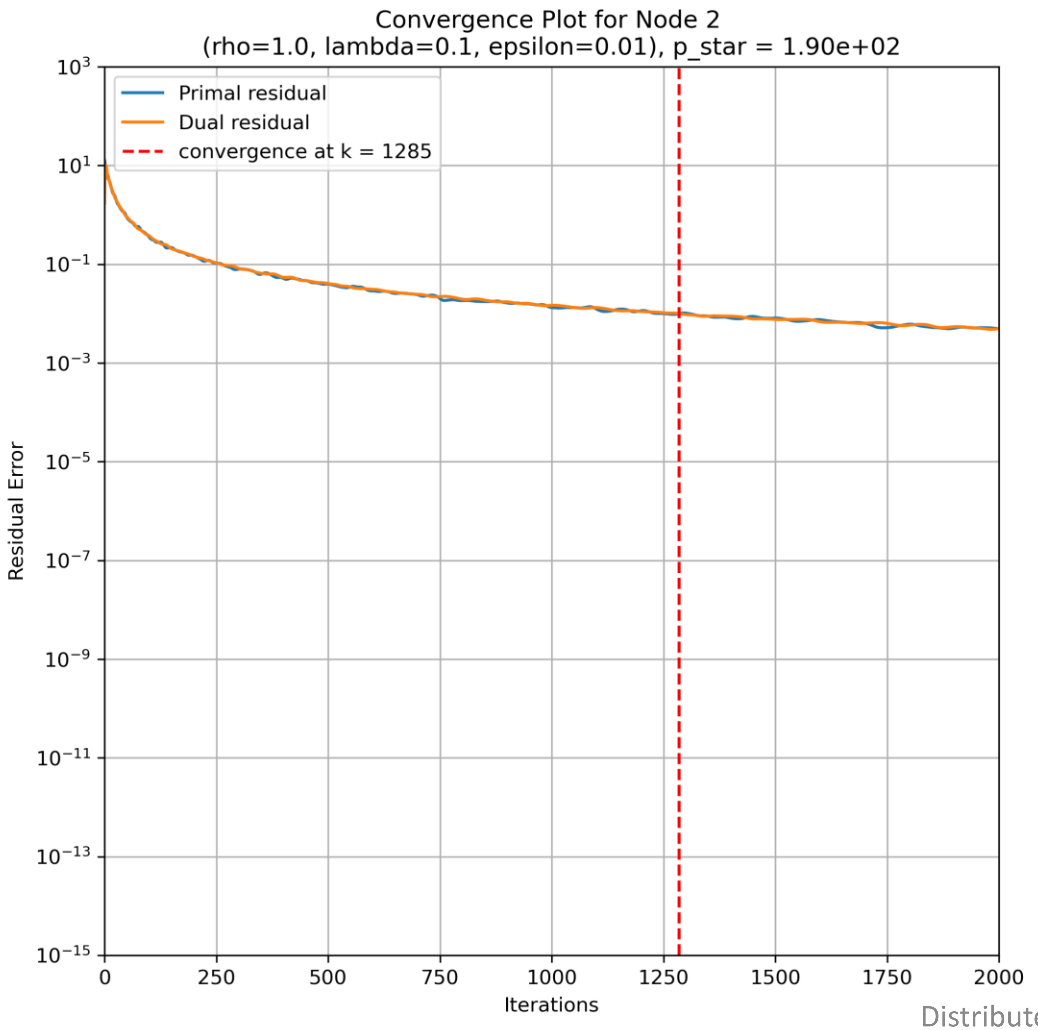


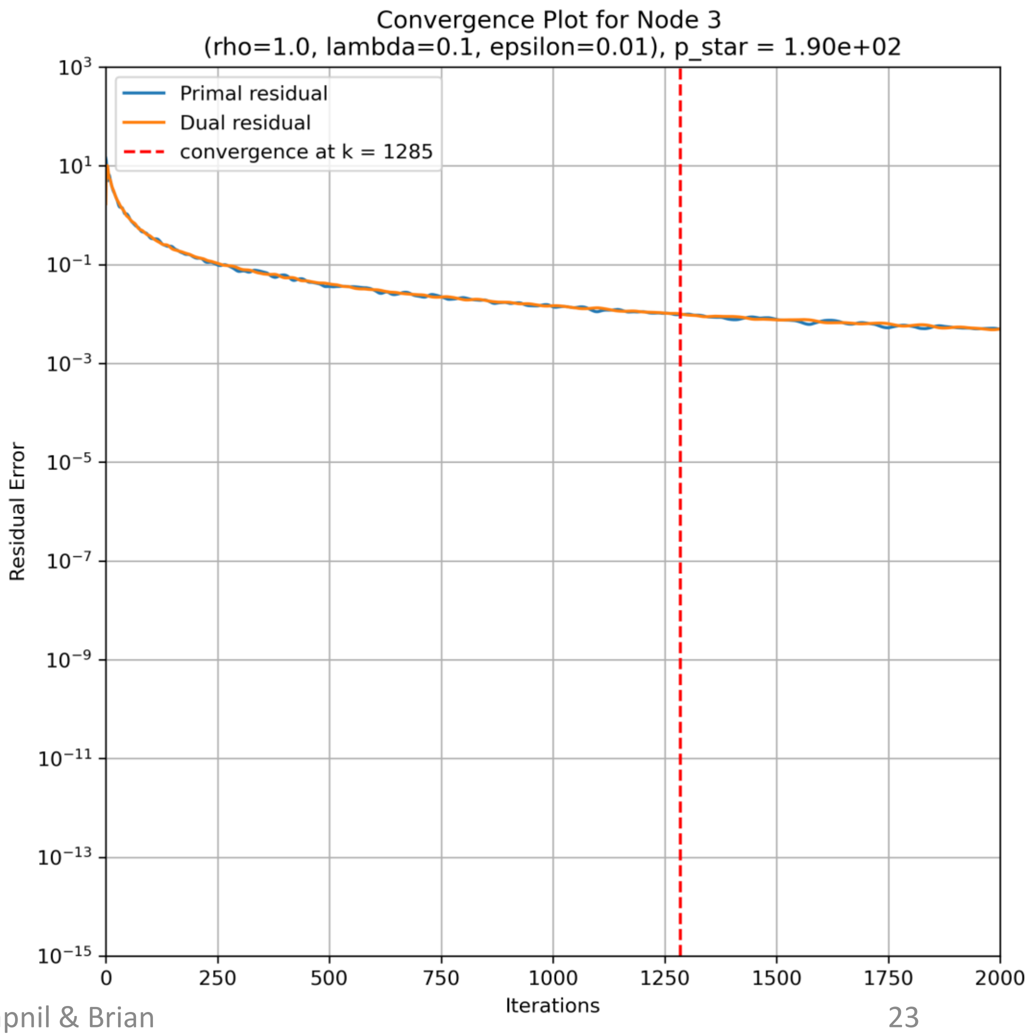
Distributed Simulation Node 0 and 1 Result



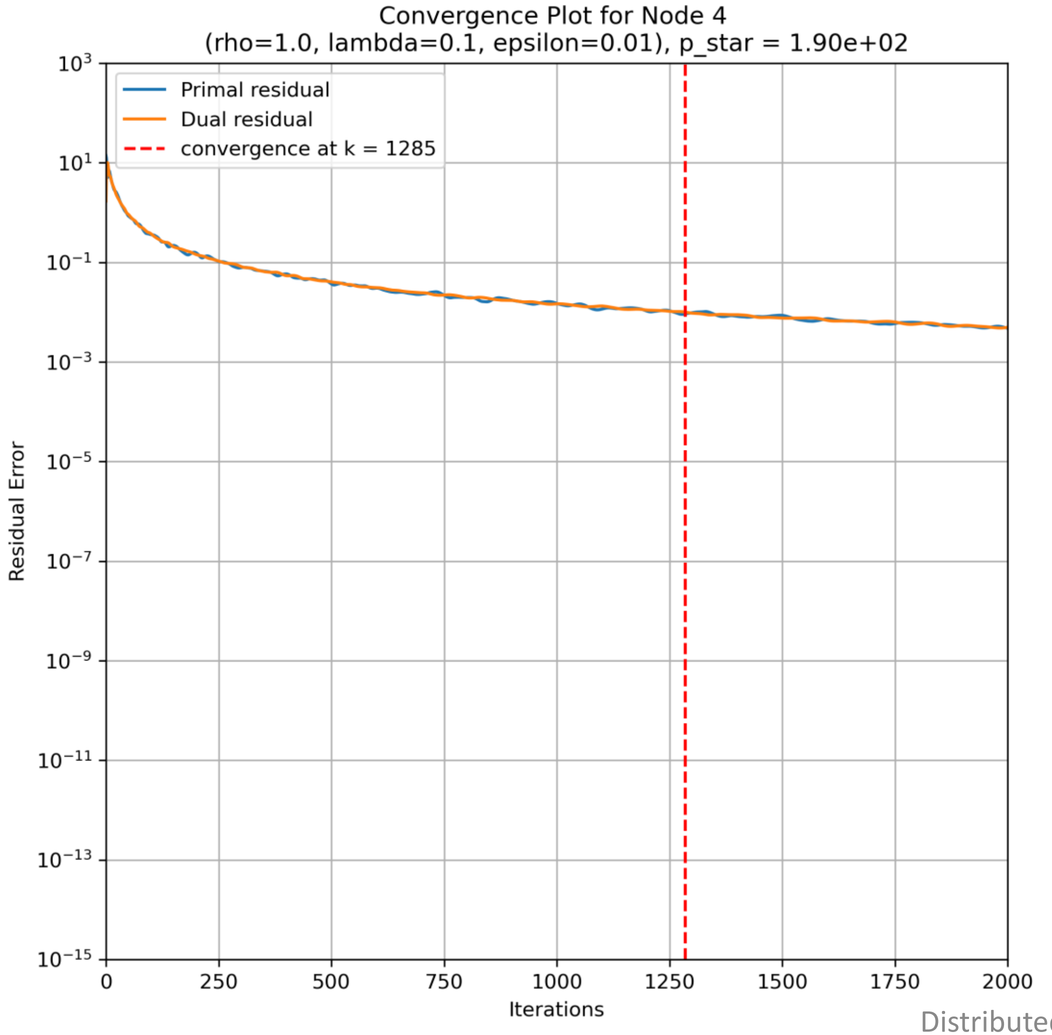


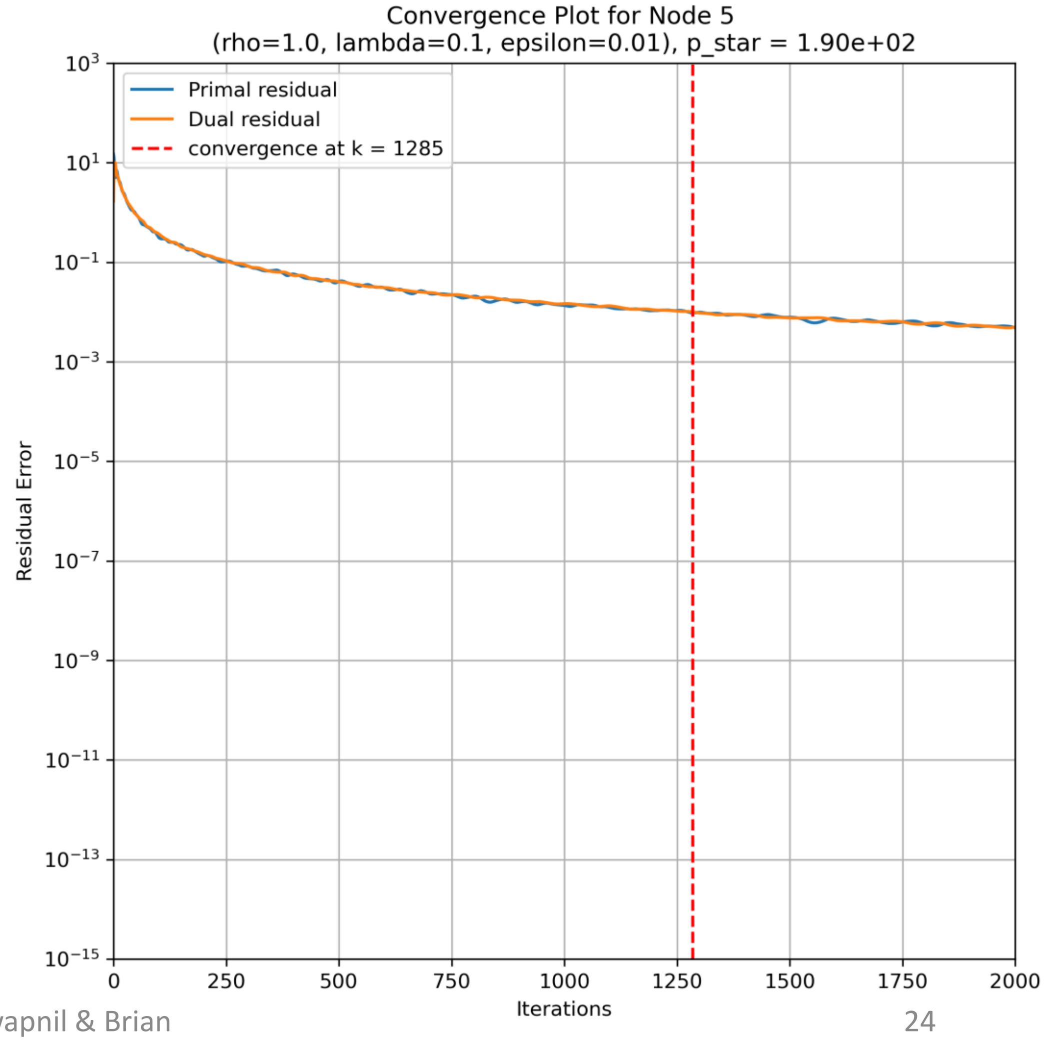
Distributed Simulation Node 2 and 3 Result



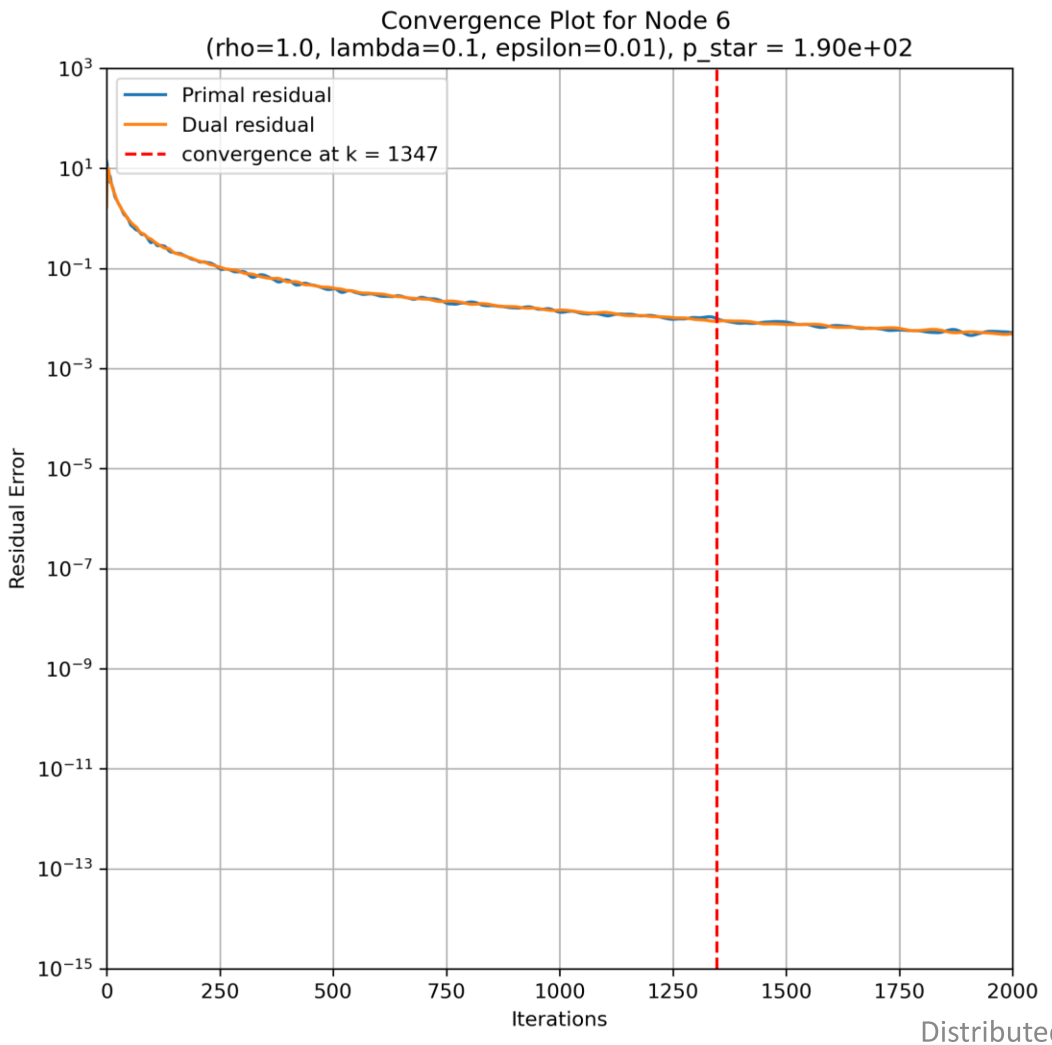


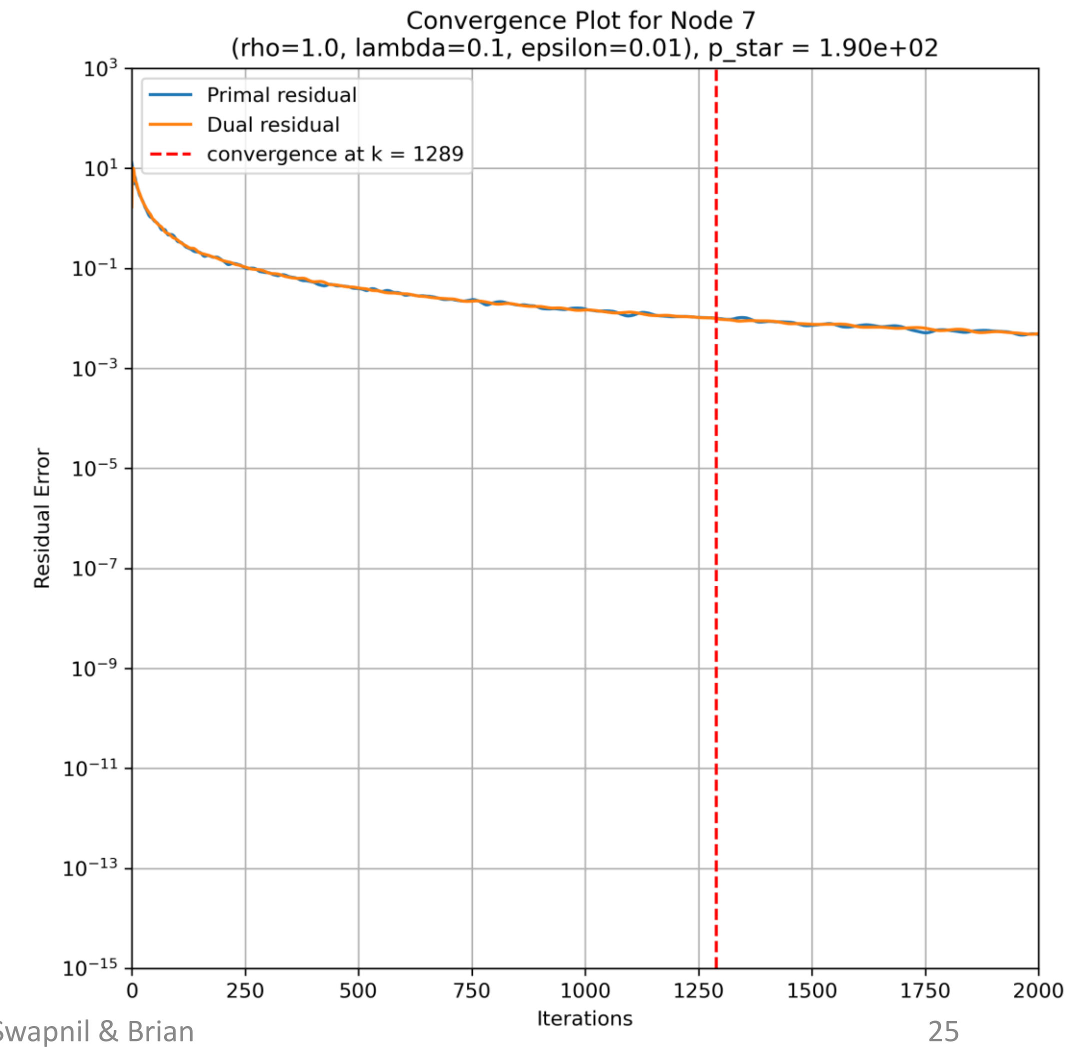
Node 4 and 5 Result



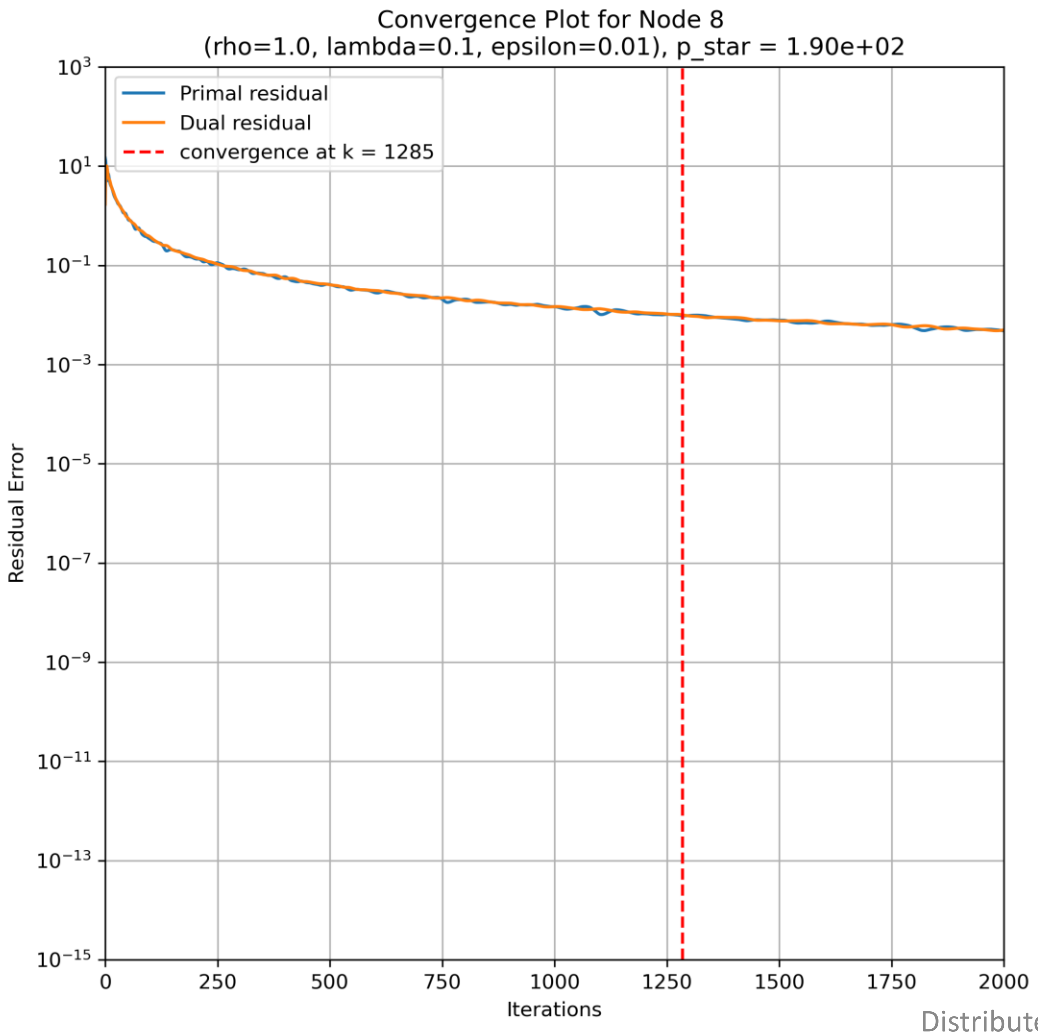


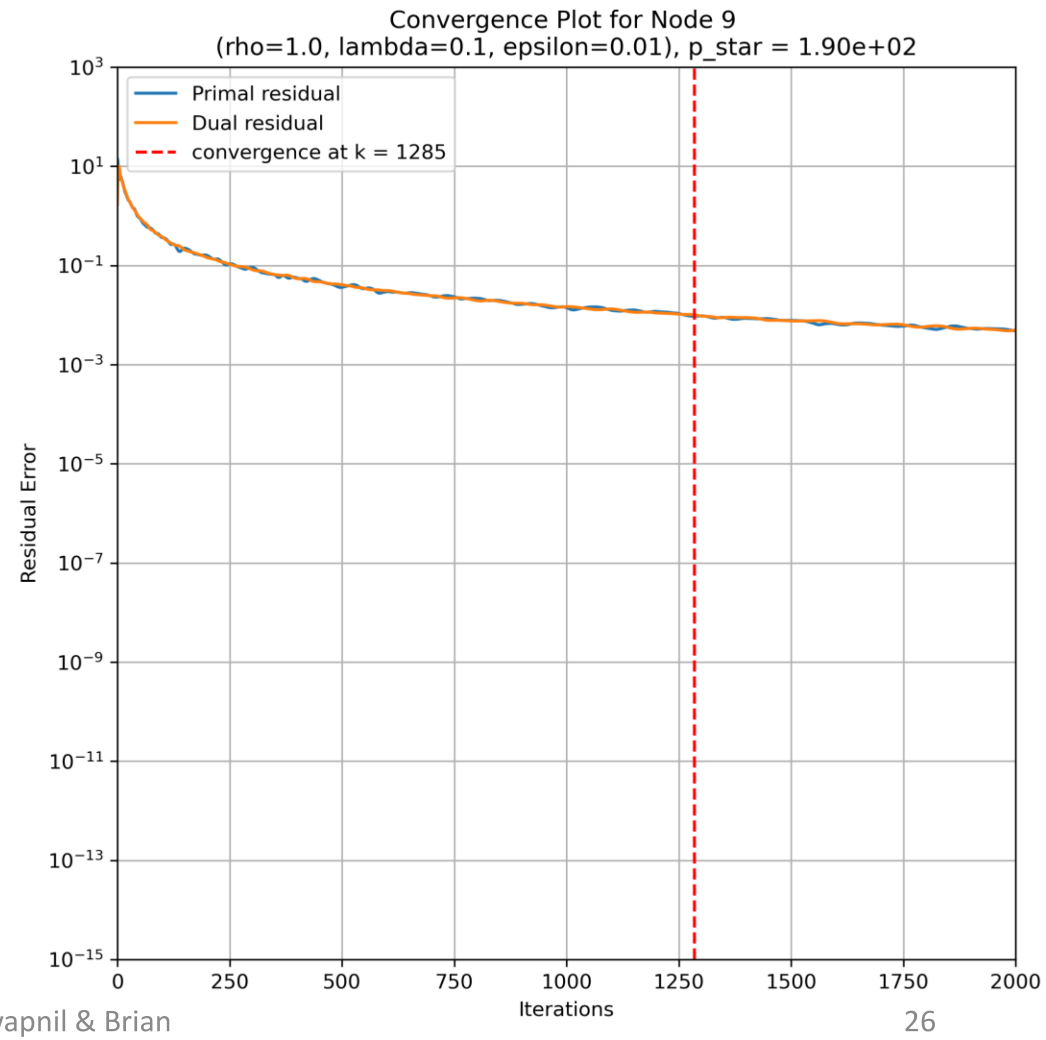
Node 6 and 7 Result





Node 8 and 9 Result





Future Work

- Test different input matrices A
  - Dense vs Sparse matrices
    - We only experimented with dense matrices
  - Tall matrix vs wide matrix
    - ADMM seems to converge with higher precision on tall and square matrices (residual ~10e-12), but with low precision on wide matrices (residual ~10e-3)
  - Very large matrices e.g. (400000x1000)
    - Large enough that the entire dataset does not fit into fast memory on a single node
    - Large enough that the performance boost of distribution outweighs the MPI data transfer overhead
- Faster matrix inversion methods
  - Cholesky Factorization

BACKUP: Dual Problem

- convex equality constrained optimization problem

$$\begin{array}{ll} \text{minimize} & f(x) \\ \text{subject to} & Ax = b \end{array}$$

- Lagrangian: $L(x, y) = f(x) + y^T(Ax - b)$
- dual function: $g(y) = \inf_x L(x, y)$
- dual problem: maximize $g(y)$
- recover $x^\star = \operatorname{argmin}_x L(x, y^\star)$

BACKUP: ADMM Lasso Algorithm

Augmented Lagrangian: $L_\rho(x,z,y) = f(x) + g(z) + y^T(Ax+Bz-c) + (\rho/2)\|Ax+Bz-c\|_2^2$

ADMM Algorithm General Form:

$$\begin{aligned} x^{k+1} &:= \operatorname{argmin}_x L_\rho(x, z^k, y^k) && \text{// } x\text{-minimization} \\ z^{k+1} &:= \operatorname{argmin}_z L_\rho(x^{k+1}, z, y^k) && \text{// } z\text{-minimization} \\ y^{k+1} &:= y^k + \rho(Ax^{k+1} + Bz^{k+1} - c) && \text{// dual update} \end{aligned}$$

Lasso Obj. Function: f(x) + g(z) = $(1/2)\|Ax-b\|_2^2 + \lambda\|z\|_1$

A = B = I, c = 0

ADMM Algorithm Applied to LASSO:

$$\begin{aligned} x^{k+1} &:= (A^TA + \rho I)^{-1}(A^Tb + \rho z^k - y^k) \\ z^{k+1} &:= S_{\lambda/\rho}(x^{k+1} + y^k/\rho) && \text{// Soft thresholding operator} \\ y^{k+1} &:= y^k + \rho(x^{k+1} - z^{k+1}) \end{aligned}$$

BACKUP: Distributed ADMM Lasso Algorithm

Dist. Augmented Lagrangian: $L_\rho(x, z, y) = \sum_{i=1}^N \left(f_i(x_i) + y_i^T(x_i - z) + (\rho/2)\|x_i - z\|_2^2\right)$

Dist. ADMM Algorithm General Form:

$$\begin{aligned} x_i^{k+1} &:= \underset{x_i}{\operatorname{argmin}} \left(f_i(x_i) + y_i^{kT}(x_i - z^k) + (\rho/2)\|x_i - z^k\|_2^2\right) \\ z^{k+1} &:= \frac{1}{N}\sum_{i=1}^N \left(x_i^{k+1} + (1/\rho)y_i^k\right) \\ y_i^{k+1} &:= y_i^k + \rho(x_i^{k+1} - z^{k+1}) \end{aligned}$$

ADMM Algorithm Applied to LASSO:

$$\begin{aligned} x_i^{k+1} &:= (A_i^T A_i + \rho I)^{-1}(A_i^T b_i + \rho(z^k - u_i^k)). \\ z^{k+1} &:= S_{\lambda/\rho N}(\overline{x}^{k+1} + \overline{u}^k) \\ u_i^{k+1} &:= u_i^k + x_i^{k+1} - z^{k+1}. \end{aligned}$$

// soft threshold of x_avg and u_avg