Real-time Modeling and Software Framework for Estimating Greenhouse Gas Emissions

Vineet Yadav

Jet Propulsion Laboratory,

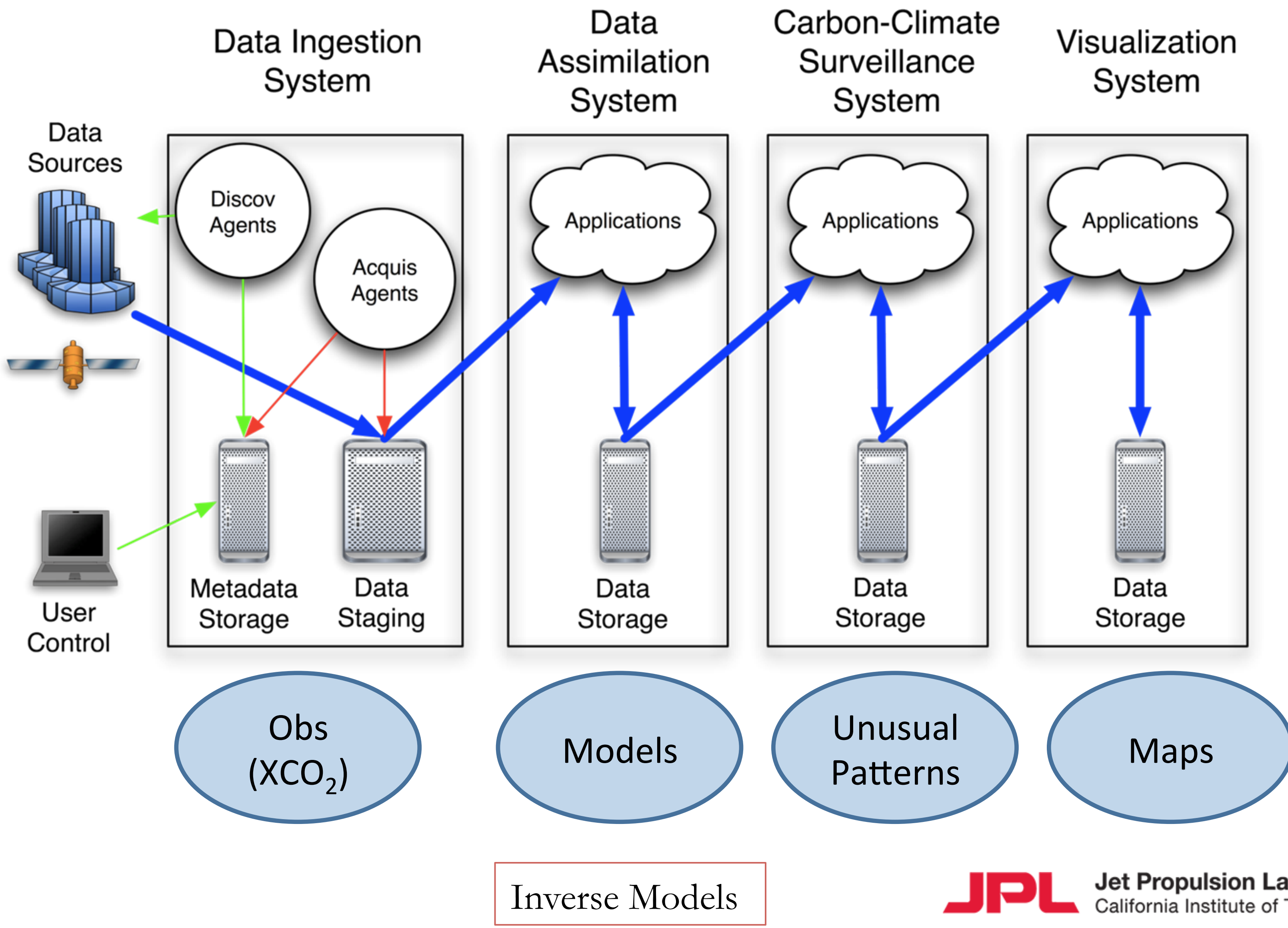
California Institute of Technology

11/11/2015

Objectives

- A software design framework for real-time measurement and monitoring of greenhouse gas emissions
  - Carbon dioxide (CO_2), Methane, Nitrous Oxide, Chlorofluorocarbons etc.
  - Responsible for global climate change.
  - Primary focus in this presentation is on CO_2
- Provide insight into challenges associated with fulfilling the primary goal of OCO2 satellite
  - Estimate fluxes of CO_2
- Computational challenges associated with modeling to get estimates of greenhouse gas emissions

A software pipeline

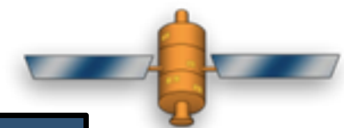


Jet Propulsion Laboratory
California Institute of Technology

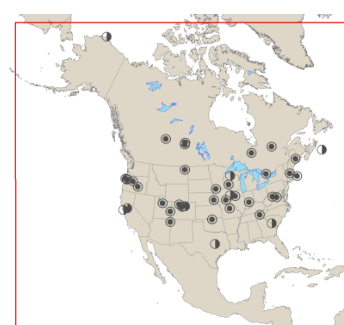
Measurements: Fluxes to Concentration (e.g. CO2)

Transport & Mixing

Molecules of CO2 in Million Molecules of Air (PPM)

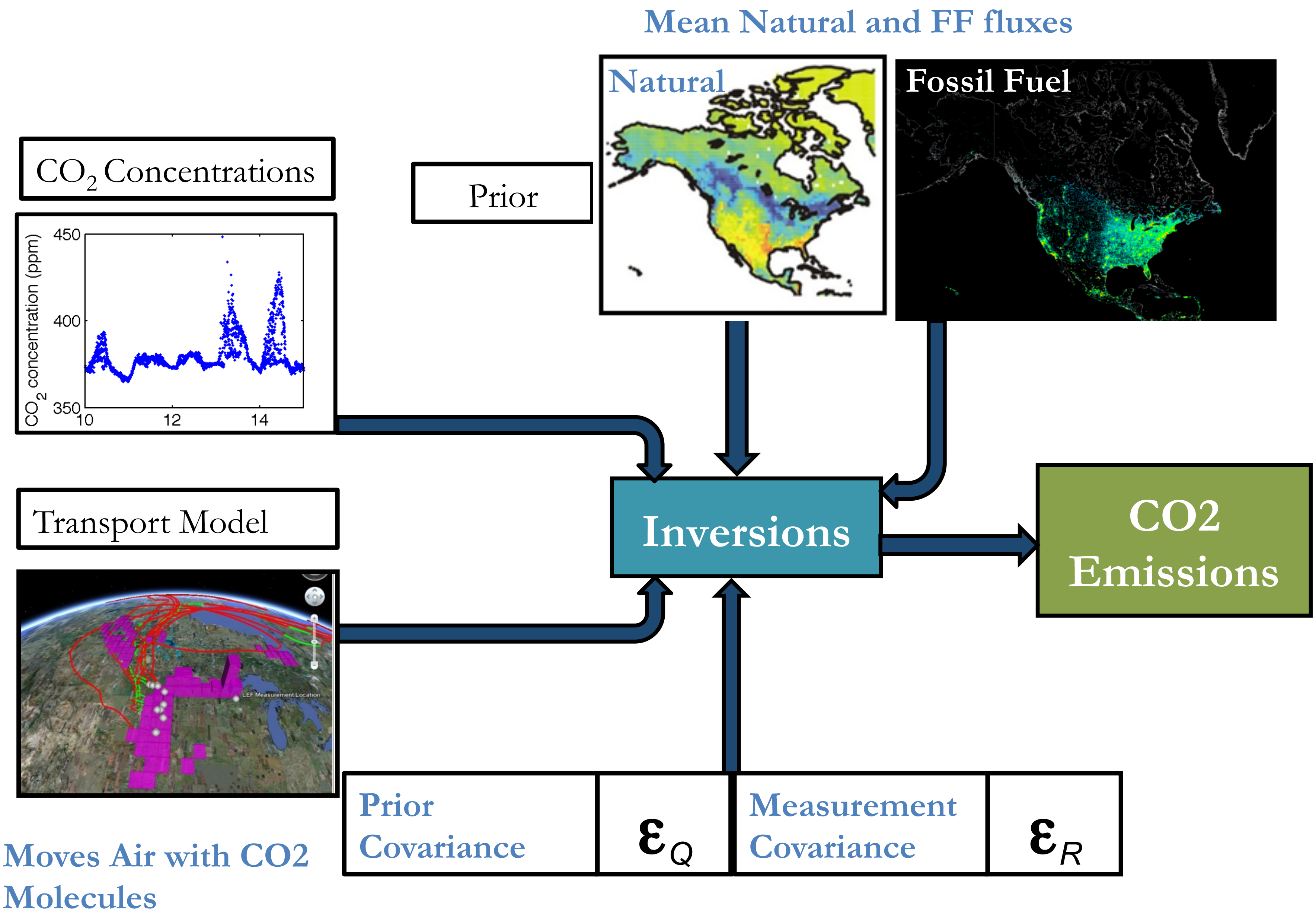


Atmospheric Measurements



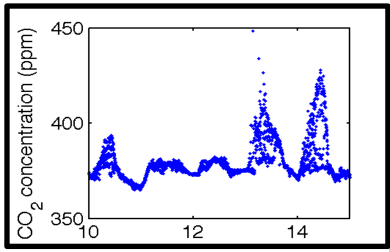
99.1 million metric tons of CO2 (FF) released in atmosphere in 2010 from LA county

Inversions: Concentrations to Fluxes



Inverse Modeling Equation

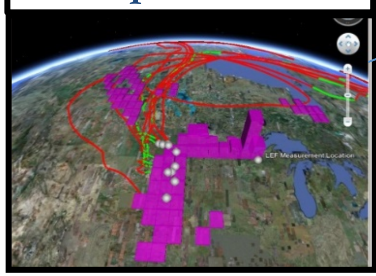
CO_2 Observations



Measurement Covariance | ε_R

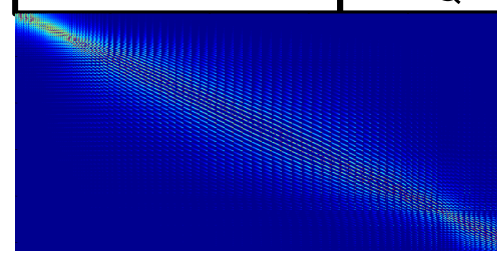
$$L_s = \frac{1}{2}(z - Hs)^T R^{-1}(z - Hs) + \frac{1}{2}(s - s_p)^T Q^{-1}(s - s_p)$$

Transport Model



CO2 Emissions

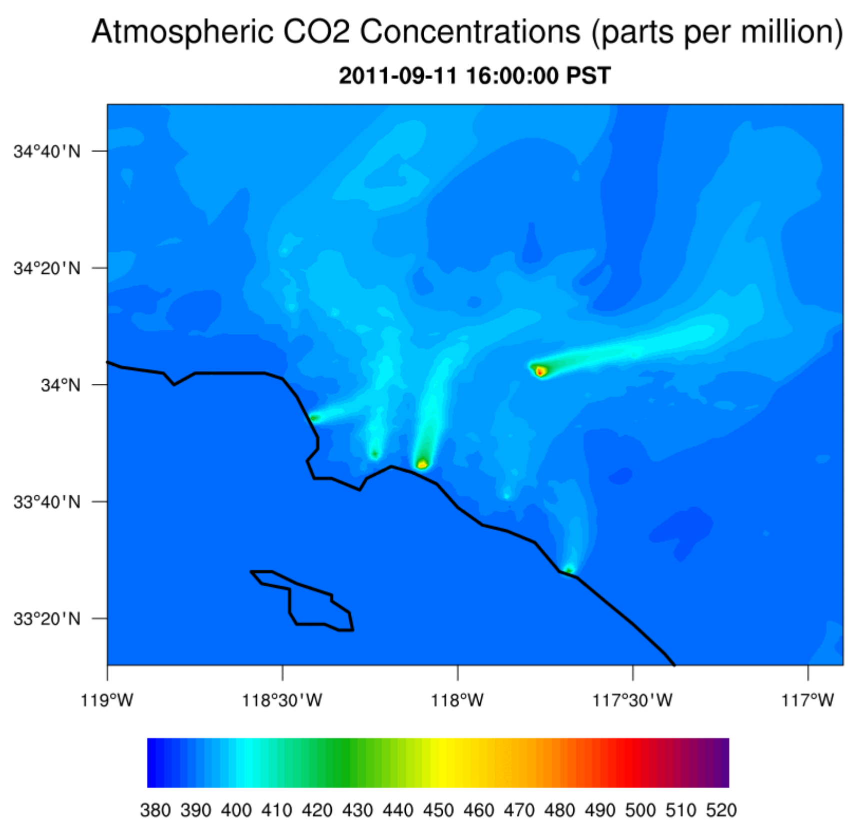
Prior Covariance | ε_Q

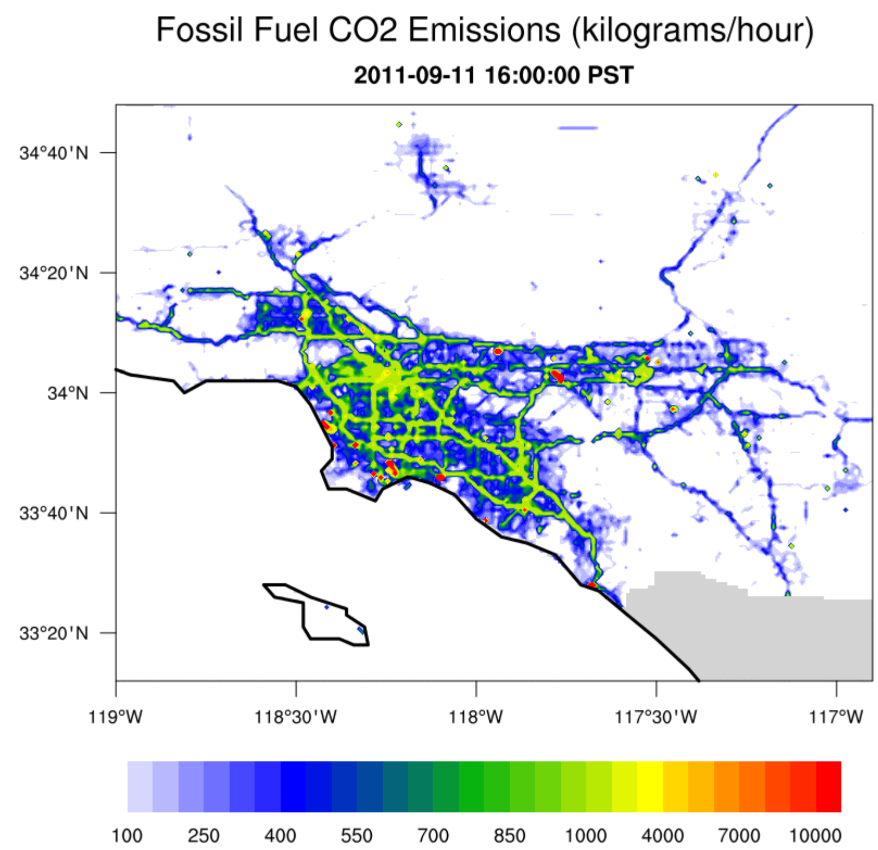


Disaggregated Version

$$L_{s_{bio}, s_{ff}} = \frac{1}{2}(z - [H_{bio} s_{bio} + H_{ff} s_{ff}])^T R^{-1}(z - [H_{bio} s_{bio} + H_{ff} s_{ff}]) + \frac{1}{2}(s_{bio} - s_{pbio})^T Q_{bio}^{-1}(s_{bio} - s_{pbio}) + \frac{1}{2}(s_{ff} - s_{pff})^T Q_{ff}^{-1}(s_{ff} - s_{pff})$$

Measurements in Carbon Cycle Science: Fluxes to Concentration

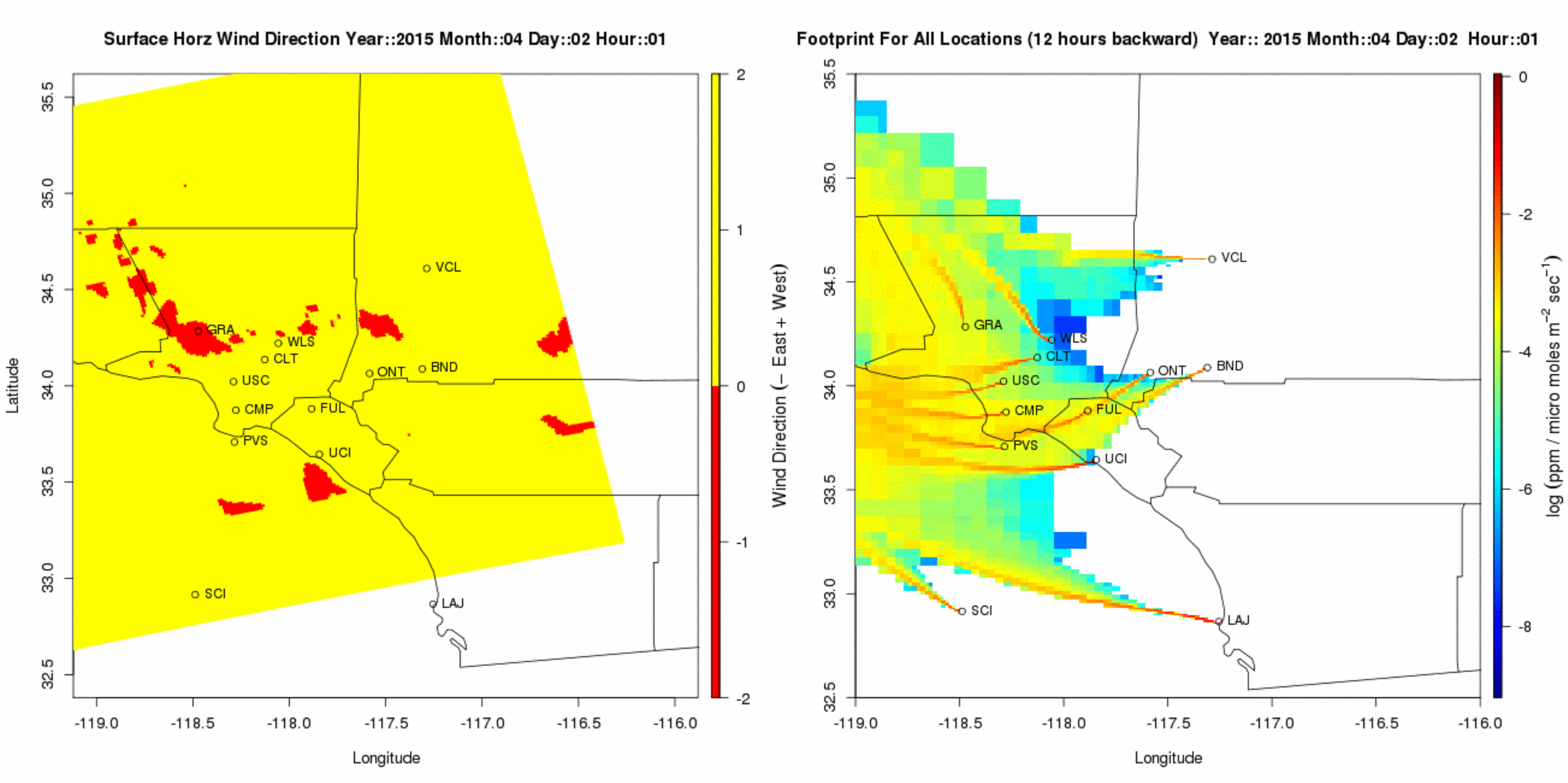




Jet Propulsion Laboratory
California Institute of Technology

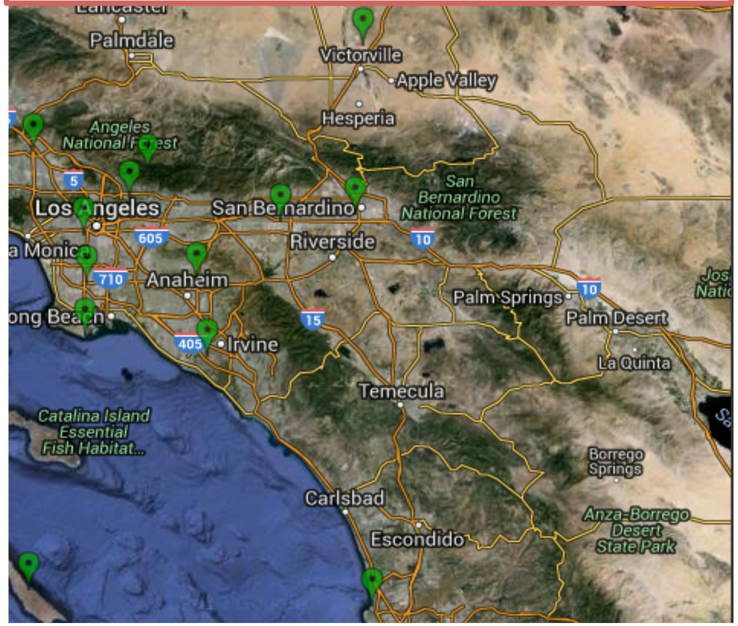
Credits: Sha Feng (Penn State)

Complexity of Transport: Example of Los Angeles



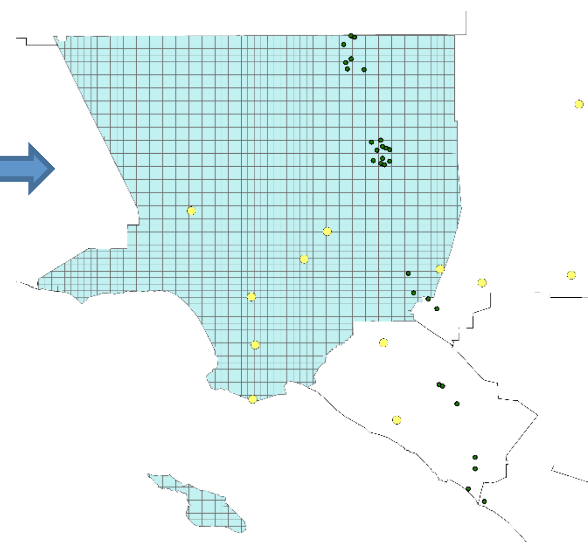
Sparsity of Measurements: Example of Los Angeles

Observations From in-situ Network



OCO2 Observations April 1 2015

- Underdetermined problem
- Illposed-problem



ratory
nnology

Computational Hurdles in Inverse Modeling

- *Matrix Multiplication*
- *Computation of Uncertainties*
- *Solution of Linear Systems of Equations*
- *Optimization of covariance parameters*

The Matrix Multiplication Problem

$$L_{\mathbf{s}} = \frac{1}{2}(\mathbf{z} - \mathbf{Hs})^{\mathrm{T}}\mathbf{R}^{-1}(\mathbf{z} - \mathbf{Hs}) + \frac{1}{2}(\mathbf{s} - \mathbf{s}_{p})^{\mathrm{T}}\mathbf{Q}^{-1}(\mathbf{s} - \mathbf{s}_{p})$$

- $\underbrace{\boldsymbol{\Sigma}}_{\text{PSD}} = \underbrace{\mathbf{H}_{(n,\ m)}\underbrace{\mathbf{Q}_{(m,\ m)}}_{\text{PSD}}\mathbf{H}_{(m,\ n)}^{T}}_{\text{PSD}}$
- Properties:
  - $\boldsymbol{\Sigma}$ is a symmetric matrix
  - $\mathbf{HQH}^{T}$ is a symmetric matrix ($\mathbf{Q}$ is symmetric)
- Cost of Matrix Multiplication
  - $\underbrace{\underbrace{(\mathbf{H}_{(n,\ m)}\mathbf{Q}_{(m,\ m)})}_{O(n^{3})}\mathbf{H}_{(m,\ n)}^{T}}_{O(n^{3})} + O(n^{3})$

Algorithms for Matrix Multiplication (MM)

- $O(n^{3}) > O(n^{\sim 2.80}) > O(n^{\sim 2.37})$ [General MM]

  Naive — Strassen — Coppersmith-Winograd

- Toeplitz/Vandermonde/Circulant/Hankel then MM is $O(n^2 \log n)$ operation

  $O(n^{3}) > O(n^{\sim 2.80}) > O(n^{\sim 2.37}) > O(n^2 \log n)$

- If a matrix is separable and symmetric matrix then MM $O(n^{2.5})$ operation

- $O(n^{3}) > O(n^{\sim 2.80}) > O(n^{2.50}) > O(n^{\sim 2.37}) > O(n^2 \log n)$

Decomposition of the prior covariance matrix

- Suppose there are two matrices :

$$\mathbf{D}_{(p,\ q)} = \begin{pmatrix} d_{(1,1)} & \cdots & d_{(1,q)} \\ \vdots & \ddots & \vdots \\ d_{(p,1)} & \cdots & d_{(p,q)} \end{pmatrix} \quad \mathbf{E}_{(r,\ t)} = \begin{pmatrix} e_{(1,1)} & \cdots & e_{(1,t)} \\ \vdots & \ddots & \vdots \\ e_{(r,1)} & \cdots & e_{(r,t)} \end{pmatrix}$$

$$\mathbf{Q} = \begin{pmatrix} d_{(1,1)}\mathbf{E} & \cdots & d_{(1,q)}\mathbf{E} \\ \vdots & \ddots & \vdots \\ d_{(p,1)}\mathbf{E} & \cdots & d_{(p,q)}\mathbf{E} \end{pmatrix}$$

- Impose condition that $\mathbf{Q}$ is symmetric
- Symbolically, this matrix multiplication can be given as:

$$(\mathbf{HQ})_{(n,\ m)} = \mathbf{H}_{(n,\ m)} \Big(\underbrace{\mathbf{D}_{(p,\ q)}}_{temporal covariance} \otimes \underbrace{\mathbf{E}_{(r,\ t)}}_{spatial covariance} \Big) \underbrace{\mathbf{Q}}_{(pr,\ qt = m,\ m)}$$

Matrix multiplication with prior covariance matrix

$$vec(\mathbf{D}) = \begin{pmatrix} d_{(1,1)} \\ \vdots \\ d_{(p,1)} \\ d \ldots \end{pmatrix}$$

- Definition of vec:

$$\mathbf{D}_{(p \times q)} = \begin{pmatrix} d_{(1,1)} & \cdots & d_{(1,q)} \\ \vdots & \ddots & \vdots \\ d_{(p,1)} & \cdots & d_{(p,q)} \end{pmatrix}$$

- (Modification) Identity for MM of $\mathbf{HQ}$

$$vec\left[\mathbf{H}(\mathbf{D} \otimes \mathbf{E})\right] = (\mathbf{I} \otimes \mathbf{H})\,(vec(\mathbf{D}) \otimes \mathbf{E})$$

- Computational cost for matrix multiplication of $\mathbf{HQ}$:

$$\mathbf{HQ}_{kron} = \underbrace{h\{prq\}}_{scalar\ multiplication} + \underbrace{q\{(p-1)nr\}}_{Addition} + \underbrace{q\{(2r-1)nt\}}_{Matrix\ Multiplication} = O(n^{2.50})$$

Jet Propulsion Laboratory
California Institute of Technology

Matrix multiplication with prior covariance matrix II

- Multiplication of $(\mathbf{HQ})$ and $\mathbf{H}^T$

- Under Condition that $\mathbf{Q}$ is symmetric the product of $(\mathbf{HQ})$ and $\mathbf{H}^T$ is symmetric

- Under these conditions overall cost of computing $\mathbf{HQH}^T$ is

$$\sim O(n^{2.5} + n^{2.81}/2)$$

- Cyclical permutation property: $\mathbf{QH}^T = [\mathbf{H}(\mathbf{D}^T \otimes \mathbf{E}^T)]^T$

- Easily parallelizable, can be implemented within a Hadoop machinery and the cost of storing $\mathbf{Q}$ is extremely low

Jet Propulsion Laboratory
California Institute of Technology

Matrix Multiplication Sparse-Sparse Case

- Consider that both **H** and **Q** are sparse
- Goal is to compute product of **H** and **Q** efficiently
- Sparse-Sparse matrix multiplication is notoriously hard to optimize
  - Branch Mispredictions (if else statements)
  - Pre-fetching problems (Cache hierarchy)
  - low compute-to-memory ratio (more time spent in accessing memory)
  - irregular memory access patterns (jumps from row to row)
- All existing algorithms are based on Gustavson's 1978 algorithm
- Operational Algorithms:
  - Single Pass Algorithm
  - Dual Pass Algorithm

Matrix Multiplication Sparse-Sparse Case II

- Operational algorithms:
  - Matlab ; Single Pass; Non Parallel
  - Intel's Math Kernel Library ; Double Pass; Parallel
  - NVIDIA CUBLAS (GPU's) ; Double Pass; Parallel (Count of non-zeros required)
- Our Algorithm
  - Single Pass Parallel when output is a non-symmetric matrix
  - Single pass parallel when output is a symmetric matrix
  - Can work in various hybrid parallel modes i.e., MPI, OpenMP and GPU's
- Foundation of the algorithm
  - First predicts number of non-zeros in the output matrix based on vector-vector multiplication and then distributes work across machines and threads

Performance of sparse-sparse matrix multiplication

- Asymptotic complexity of sparse-sparse MM is assumption specific.

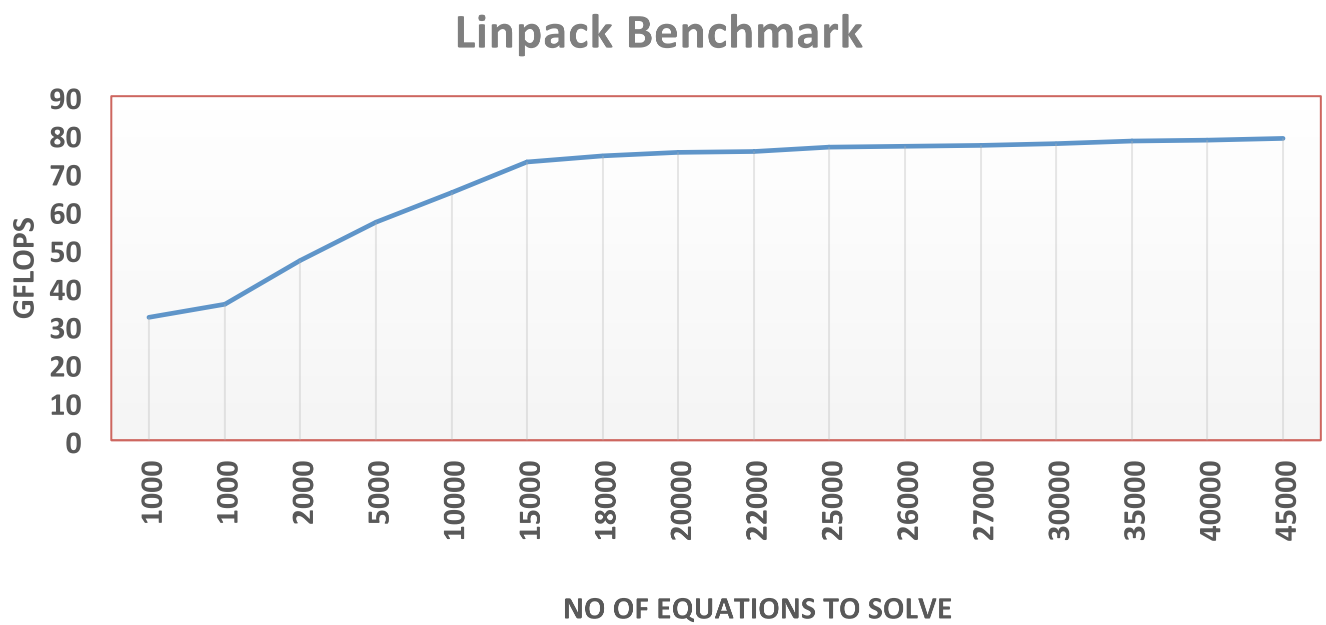
- CPU cycles, Time and Memory consumed in comparison to Intel MKL

- Machine specification:
  - Intel Xeon E5440 Harpertown 2.83 Ghz 12MB L2 Cache
  - Theoretical Gflops: 90.56 or 45.28
  - $Gflops = CPU\ Speed\ in\ GHz \times Flops/Hz \times Cores/Node \times Nodes$

- Test Case
  - Sparse matrices $\mathbf{H}/(1000 \quad 1000000)\ \mathbf{Q}/(1000000$

Benchmarks for the test machine

- Linpack benchmarks



- Performance of the algorithm (double precision)

	Our Algorithm	MKL
CPU cycles	~ 227 billion	~ 362 billion
Time Taken for Execution	~ 28 seconds	~ 42 seconds
Peak Memory Consumed	~ 2.450 Gb	~2.576 Gb
Total Memory Consumed	~4.885 Gb	~3.190 Gb

Features of the sparse-sparse matrix multiplication algorithm

- Compute upper triangular or lower triangular portion of the output matrix
- Return a dense matrix from sparse-sparse MM
- Can analytically compute for non-zeros in the output matrix if you multiply a sparse matrix with a diagonal matrix
- A double pass algorithm has been implemented that has same performance as Intel MKL

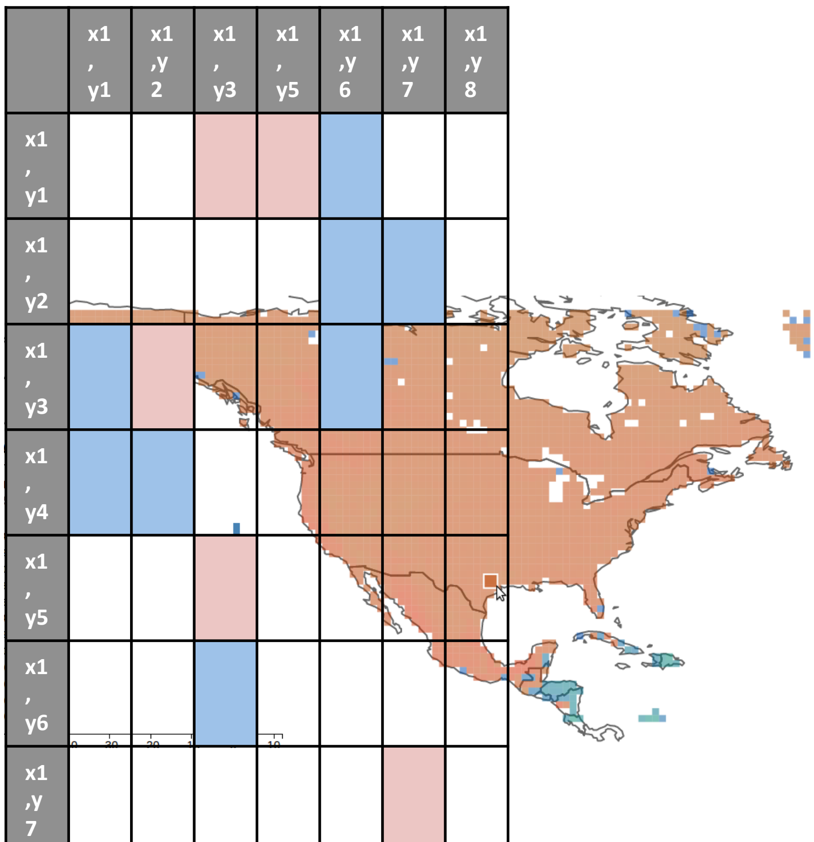
- Further research:
  - Building a Roofline model outlining the performance of out matrix multiplication algorithm

Summary and other areas of research

Further Research

- Covariance Visualization
- Anomaly Detection

Covariance Visualization



Climate Change: Anomaly Detection