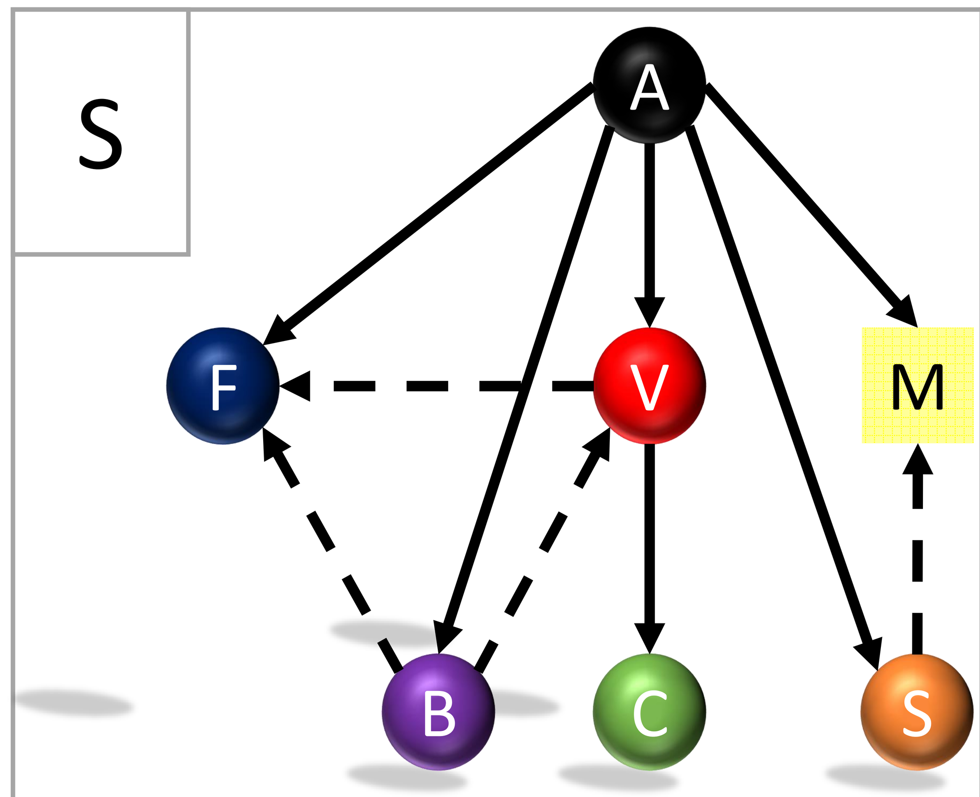
EDGIer APIs: Scalable, Feature-Rich Empirical Orthogonal Function Analysis of Distributed Geoscientific Data That "Just Works"

IN41D-0861

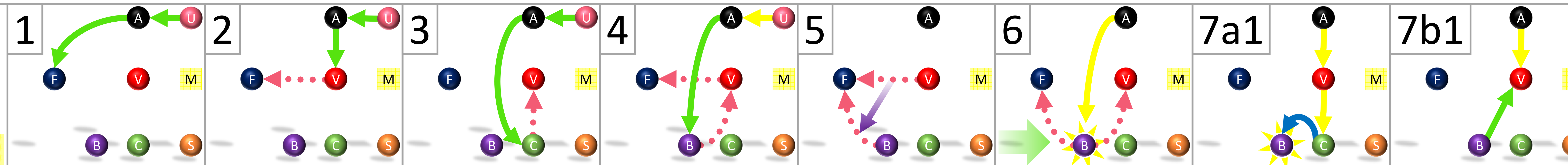
Christopher Dupuis[1], Curtis Bechtel[1,2], and Erixen Cruz[1,2]

[1]Engility at NOAA/GFDL; [2]Drexel University

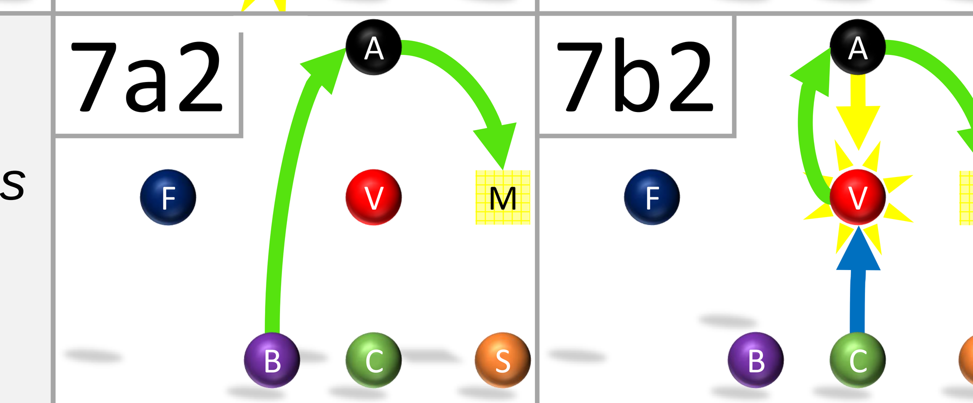
chris.dupuis@engility.com



*Object-oriented class structure of the EDGI package: spheres are classes, solid arrows show composition, and dashed arrows show dependency. Objects are **A**PI, **F**iles, **V**ariables, **M**atrix to solve, **B**locks, **C**losures, and the **S**olver. The **U**ser is also shown.*



1. *File metadata registration by user*
2. *Variable metadata registration by user*
3. *Variable closure registration by user*
4. *Launcher call by user and block initialization*
5. *Copy file dependency from variable to block*
6. *I/O reads for each block*
7. *Closure application and matrix builder; one of two options:*
   a. *Apply closures to each block, then combine data*
   b. *Combine data first, then apply closures to whole variables*
8. *Pass matrix to SVD solver and run it*
9. *Distribute solver results to blocks*
10. *I/O writes for each block*



EOF Analysis in Geoscience

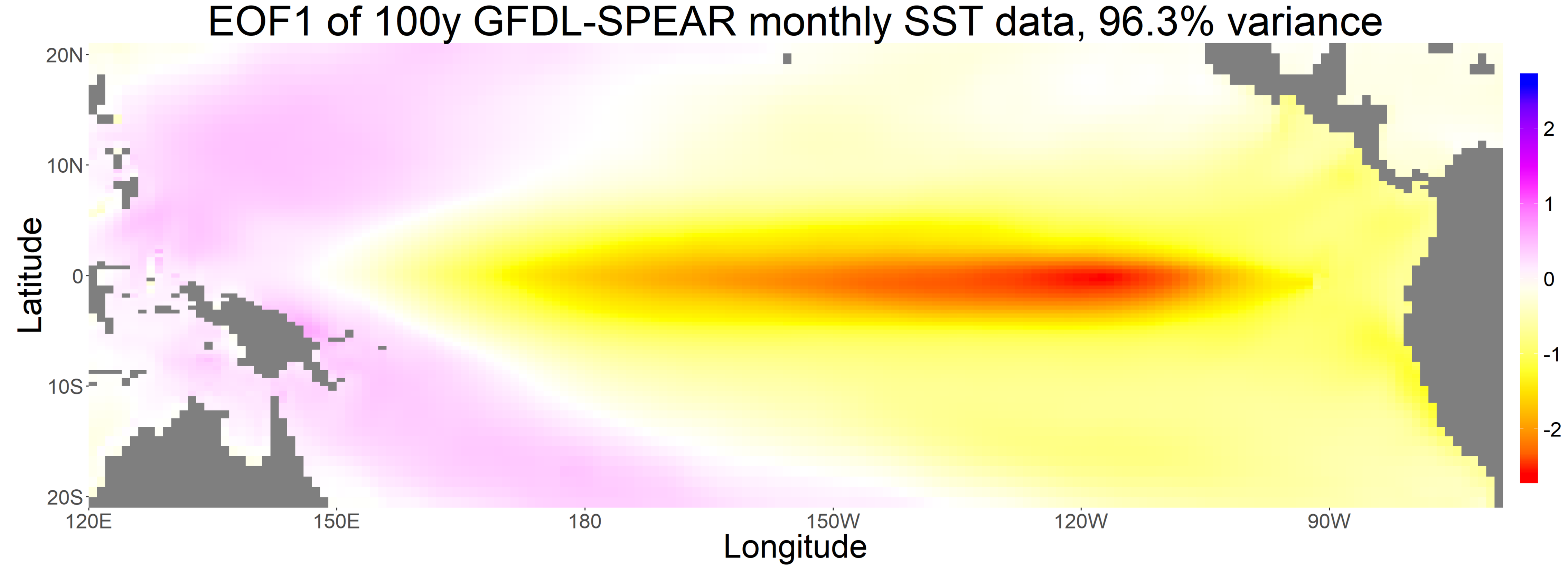
Empirical Orthogonal Function (EOF) analysis is an important part of examining oscillating or propagating patterns in geoscience. EOF analysis offers insights about an oscillation's spatial distribution, the relative importance of oscillations, their strength over time, and how they propagate. EOF Extensions can also shed light on how oscillations are distributed by frequency, the influence of autoregression, connections between different variables, and more. While useful, EOF calculation typically requires burdensome computations and irrelevant file I/O requirements. EOF problems can also easily grow to true HPC scales, requiring massive amounts of memory and compute time. Toward this end, there is little in the way of simple user interfaces that map to terminology in literature. Often, if we want EOFs, we have to hack scripts together ourselves.

The EOF/DFT General Interface (EDGI)

There is a combinatoric explosion of possible EOF algorithms, especially for spectral methods. We seek a general solution for a simple, scalable, and extensible design. Initial attempts converged on a specific design pattern: we differentiate between S-dimensions (dimensions to preserve; "space") and T-dimensions (dimensions to apply closures along, "time"), and separate file I/O from the data. We call this solution the EOF/DFT General Interface, although the EDGI approach can be extended to a large variety of EOF extensions and closures.

Example and Sample Output

EOF1 of 100y GFDL-SPEAR monthly SST data, 96.3% variance




```
edgi -f file1.nc:eofs1.nc file2.nc:eofs2.nc -v sst:sst_eofs -t time -n 32
```

Future Directions

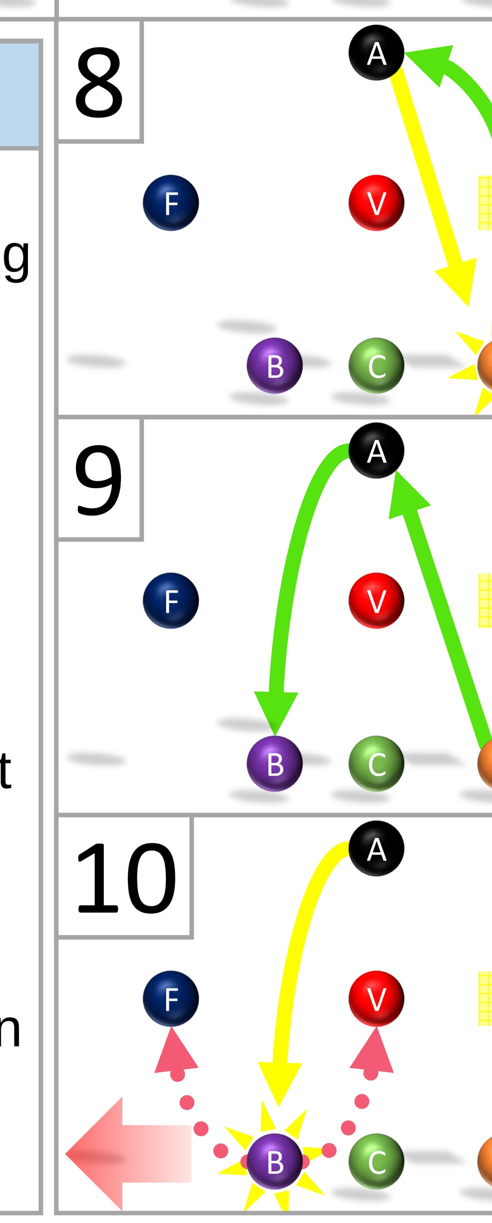
EDGI design and API can incorporate Joint Moment Component Analysis and polyspectra to shed light on coupled oscillations.

This design can be used to support a number of applications with built-in scalability and abstraction from file I/O. Possible applications include consistency tests with built-in and user-specified closures, resampling and self-consistency testing, and generation of machine learning training sets for on-the-fly oscillation recognition.

While the current EDGI package performs well on shared-memory machines, the inclusion of distributed memory linear algebra libraries and node-level affinities will be important for scalability and optimization at true HPC scales.

Design Choices

What is the best way to make EOF analysis EDGIer? We can begin by aiming for scalable I/O and algorithms, generic interfaces, and support for gappy data and parallel linear algebra libraries. We chose to begin with NetCDF I/O and the PLASMA and Intel MKL linear algebra libraries, although more can be added. Critically, the UI should be minimal. Going further, we must be able to support a wide variety of EOF extensions like spectral algorithms, time lags, non-Gaussian and autoregressive distributions, and directional data. We can also support generalizations to higher order moments.



Scalability

Intel Skylake Node Timings for 12966 Time Series (s)