Status of LAL Stochastic Background Codes*

presented by John T. Whelan
The University of Texas at Brownsville
whelan@oates.utb.edu

Presenented at the 8th LSC Meeting
2001 March 15

*See also http://oates.utb.edu/LAL-stochastic/

LIGO-G010112-00-Z

Fundamentals

- Optimally filtered cross-correlation statistic
- Work in frequency domain
- Single precision
- Continuous approximation

Details

- Cross-correlation statistic

$$Y = \sum_{j=0}^{N-1} \sum_{k=0}^{N-1} h_1[j] Q[j-k] h_2[k] = \sum_{\ell=-(N-1)}^{N-1} \tilde{\bar{h}}_1[\ell]^* \, \widetilde{Q}[\ell] \, \tilde{\bar{h}}_2[\ell]$$

- Optimal filter

$$\widetilde{Q}(f) \propto \frac{\Omega_{\text{GW}} \; \gamma(f)}{f^3 P_1(f) P_2(f)}$$

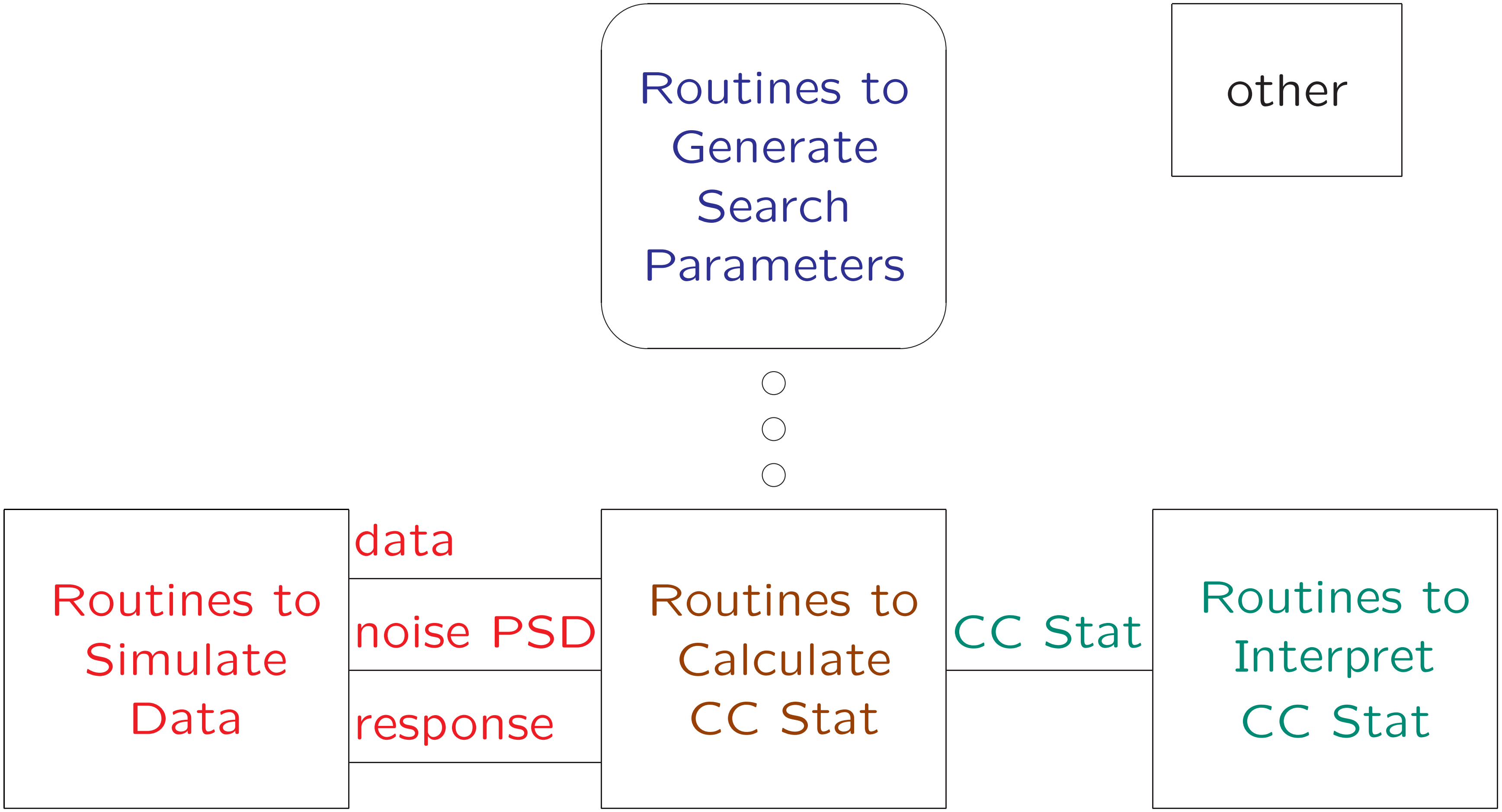
Context

- Analyze data in short ($\sim$10 sec) stretches

- Eventually use “bank” of filters ($\Omega_{\mathrm{GW}}(f) \sim f^{\alpha}$)

- UL search planned with only $\alpha = 0$

Categories of Routines



Routines to Generate Search Parameters

other

Routines to Simulate Data

data

noise PSD

response

Routines to Calculate CC Stat

CC Stat

Routines to Interpret CC Stat

Status of Module Groups

- Routines to Simulate Data:
  ∃ snippets of code, but no complete modules
  (but see `noisemodels` package)

- Routines to Calculate Cross-Correlation Statistic:
  All exist in reasonably complete form (w/o heterodyning)
  Being cleaned up, should be done within the month
  (Main remaining issue is test routines)

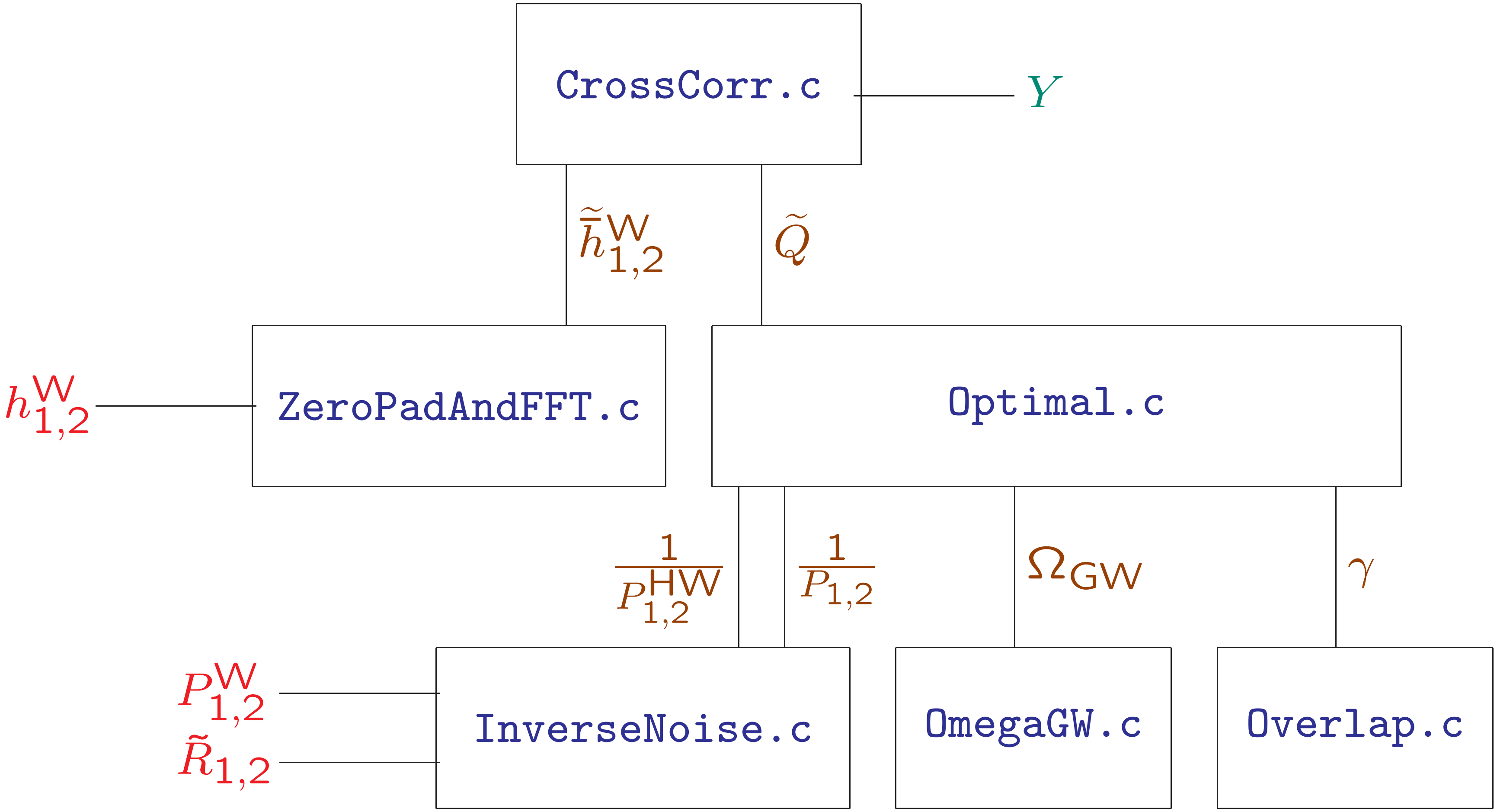
- Routines to Interpret Cross-Correlation Statistic:
  False Alarm/Dismissal module written

Status of Module Groups (cont'd)

- Routines to Generate Search Parameters:
  Not needed for upper limits search

- Other:
  Dirichlet kernel routine written but not used

Calculating CC Stat: Data Pipeline



Calculating CC Stat: Module details

`CrossCorr.c`

Inputs: Two zero-padded DFTed data streams
Optimal filter

Ouputs: Cross-Correlation Statistic

To do: Rewrite test routine
Update documentation
Implement heterodyning support

Calculating CC Stat: Module details

`ZeroPadAndFFT.c`

Inputs: Time-domain data stream

Ouputs: Zero-padded DFTed data stream

To do: Rewrite test routine
Write documentation
Implement heterodyning support

Calculating CC Stat: Module details

`Optimal.c`

Inputs: Two unwhitened & “half-whitened” inverse noise PSDs
Stochastic GW spectrum
Overlap reduction function

Ouputs: optimal filter

To do: Rewrite test routine
Write documentation
Implement heterodyning support

Calculating CC Stat: Module details

`InverseNoise.c`

Inputs: Whitened noise PSD
Whitening filter

Ouputs: Unwhitened & “half-whitened” inverse noise PSDs

To do: Rewrite test routine
Complete documentation
Implement heterodyning support

Calculating CC Stat: Module details

`OmegaGW.c`

Inputs: Power-law exponent
GW background "amplitude"

Ouputs: Stochastic GW spectrum

To do: Implement heterodyning support

Calculating CC Stat: Module details

`Overlap.c`

Inputs: Detector & site geometry for two detectors

Ouputs: Overlap reduction function

To do: Update site structure handling
Rewrite test routine
Update documentation
Implement heterodyning support