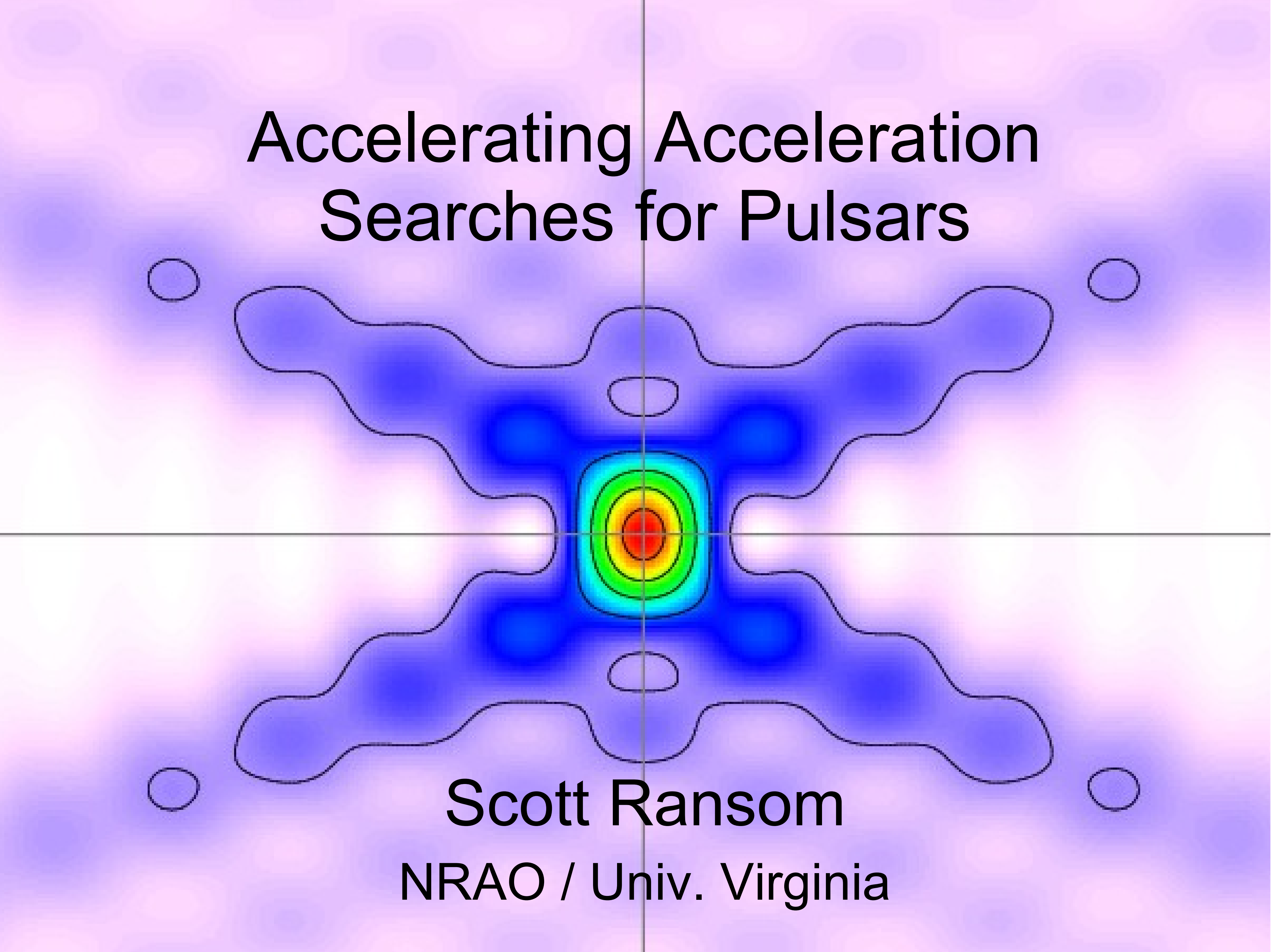


Accelerating Acceleration Searches for Pulsars

Scott Ransom

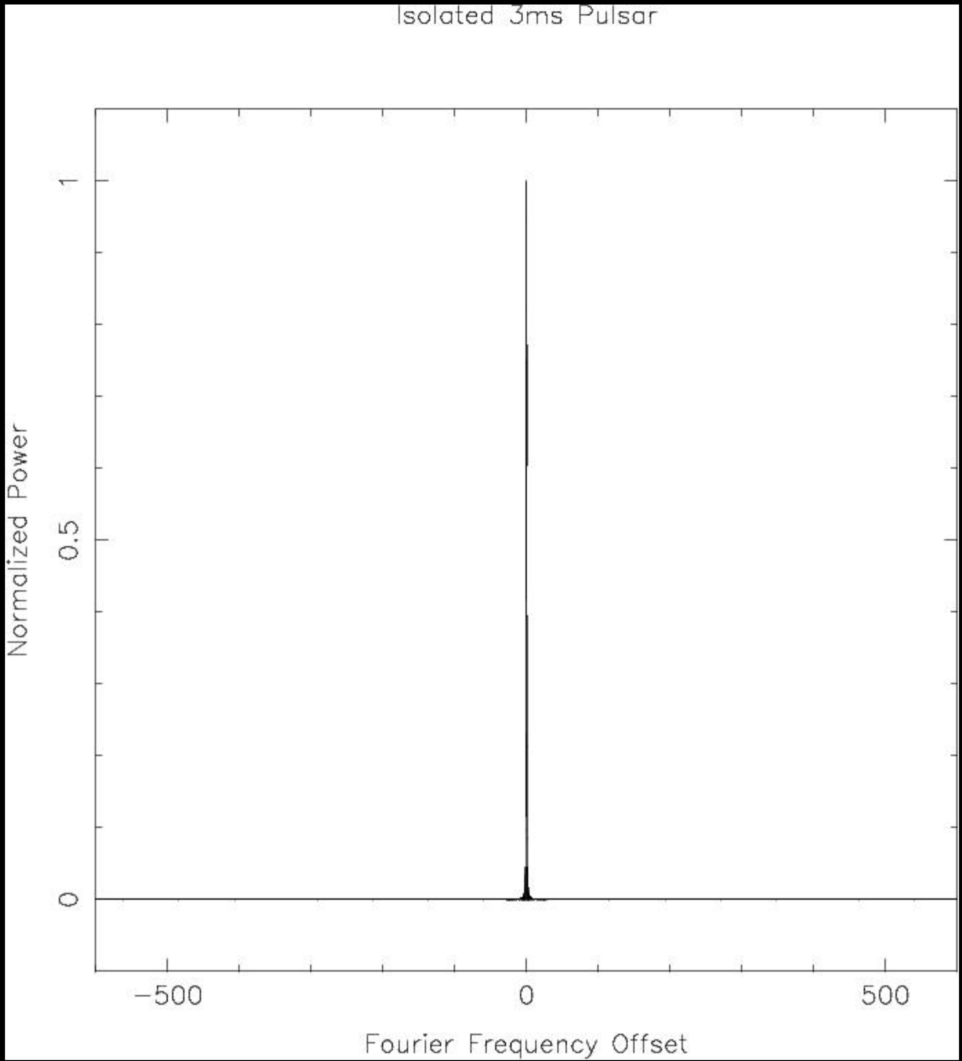
NRAO / Univ. Virginia

Why Search for Binary Pulsars?

- Much of the “sexiest” pulsar science comes from those systems in binaries:
  - Tests of general relativity and other gravity theories
  - Masses of compact objects (equation of state of dense matter)
  - High precision timing from millisecond pulsars may detect gravitational waves
- SKAI non-imaging processing is dominated by pulsar binary searches: ~10 Pops/sec(!)
- Real-time processing is required – the beams will not be permanently stored

Binary Pulsar Search Techniques

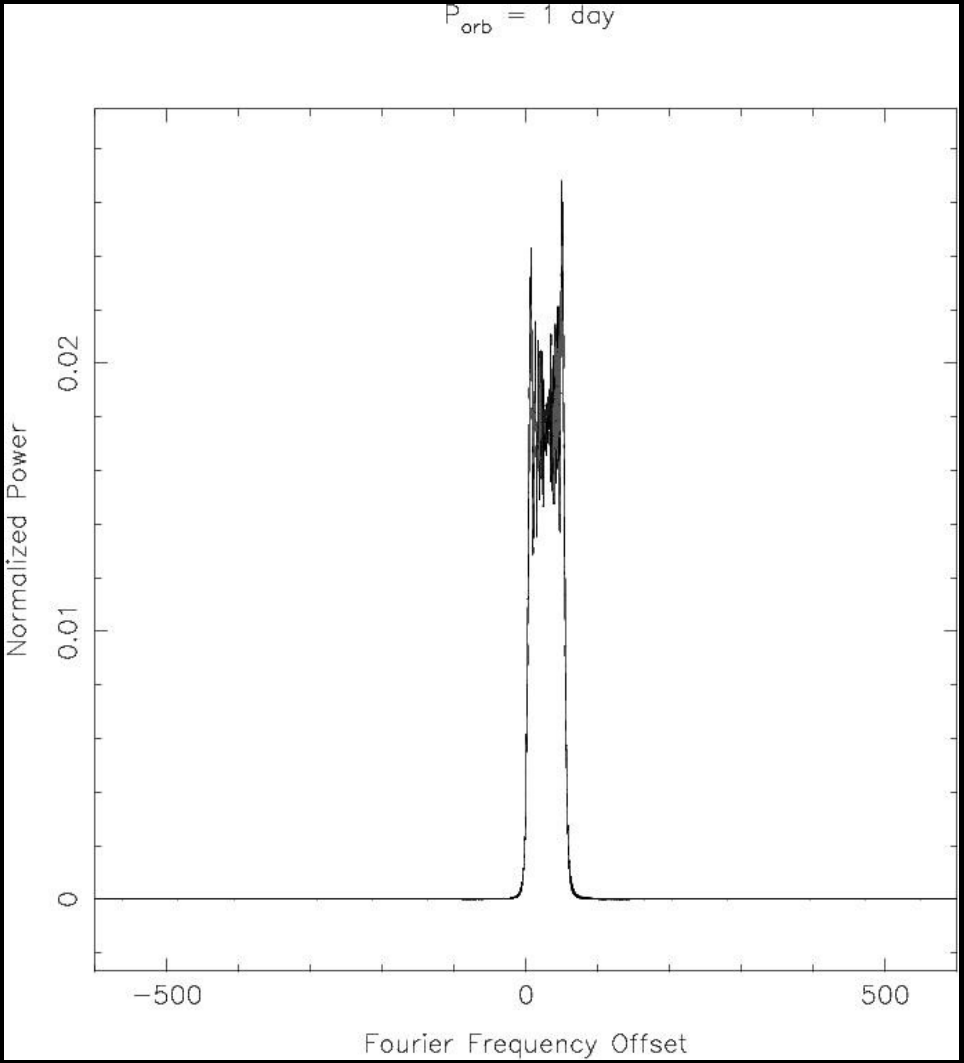
- Isolated Pulsars
  - Fourier analysis



3ms pulsar, 2 hr obs

Binary Pulsar Search Techniques

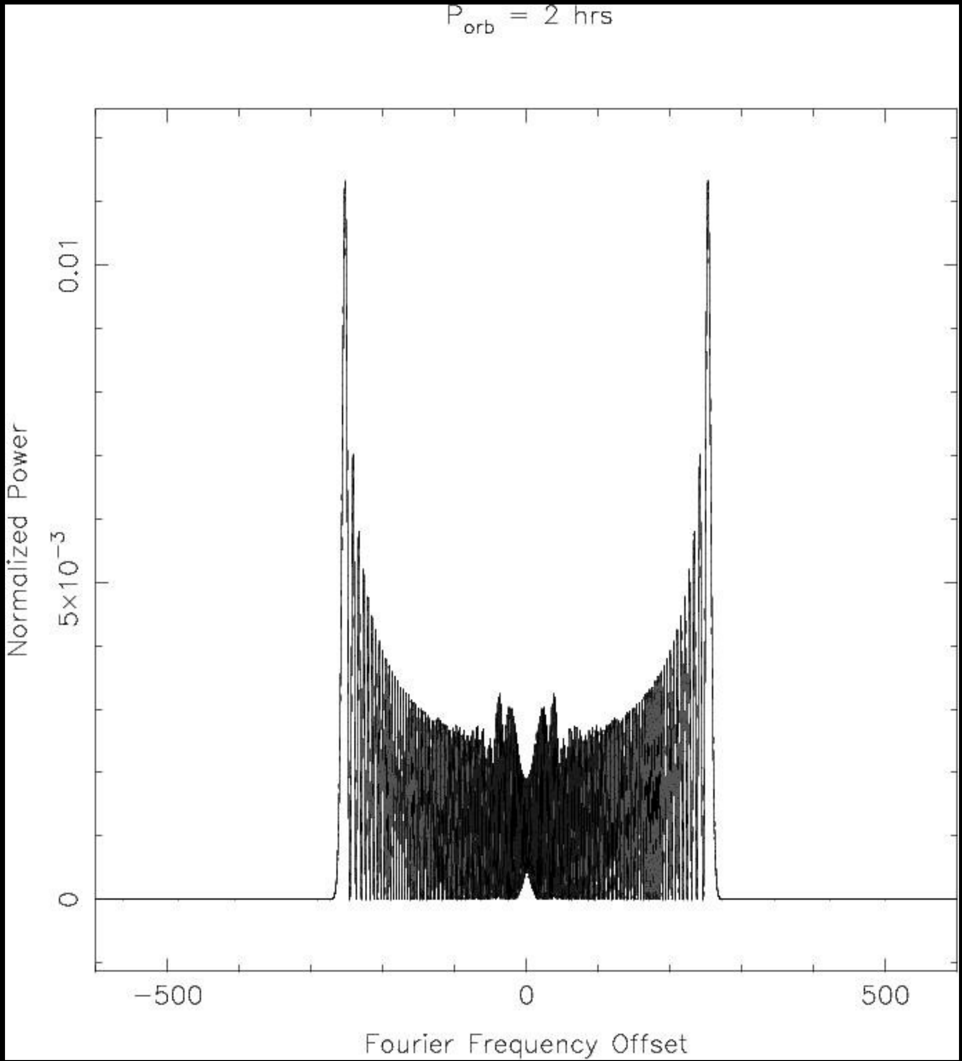
- Isolated Pulsars
  - Fourier analysis
- Binary $P_{orb} > 10T_{obs}$
  - "Acceleration" Searches



3ms pulsar, 2 hr obs

Binary Pulsar Search Techniques

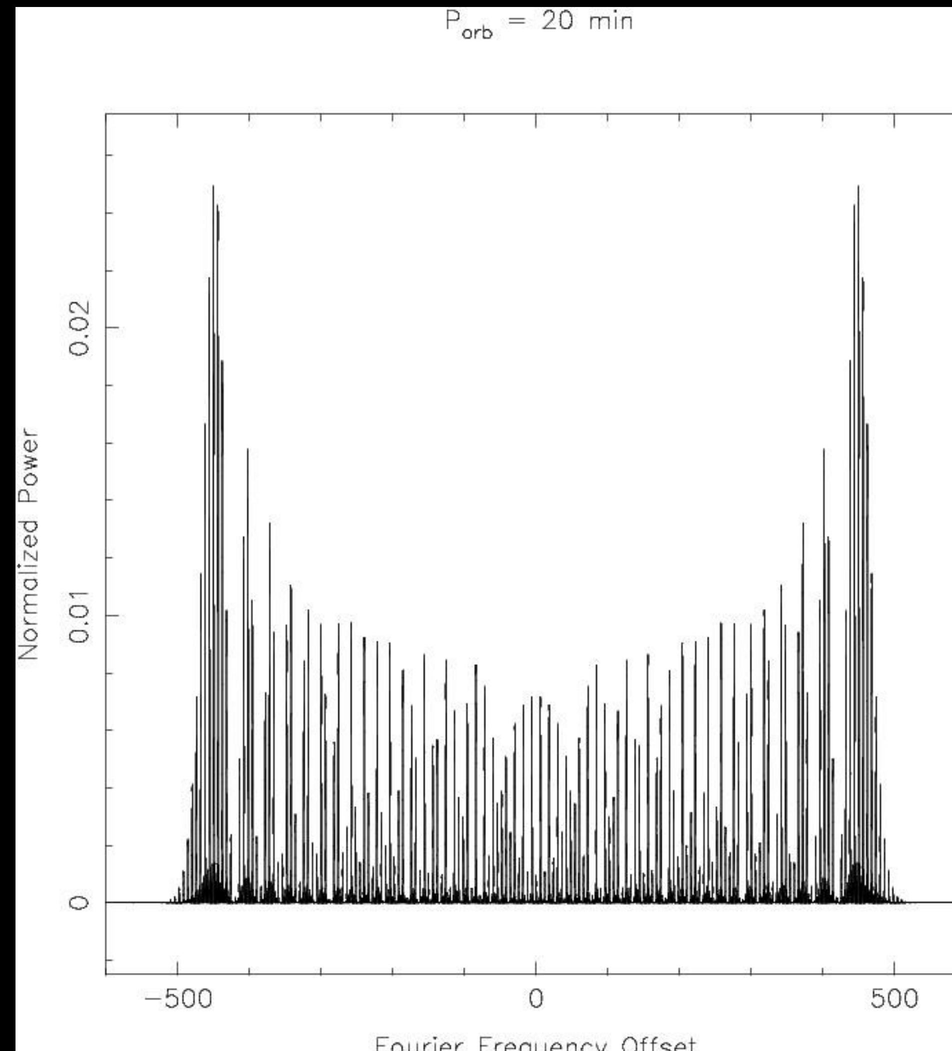
- Isolated Pulsars
  - Fourier analysis
- Binary $P_{orb} > 10T_{obs}$
  - "Acceleration" Searches
- Binary $P_{orb} \sim T_{obs}$
  - "Dynamic" Power Spectra



3ms pulsar, 2 hr obs

Binary Pulsar Search Techniques

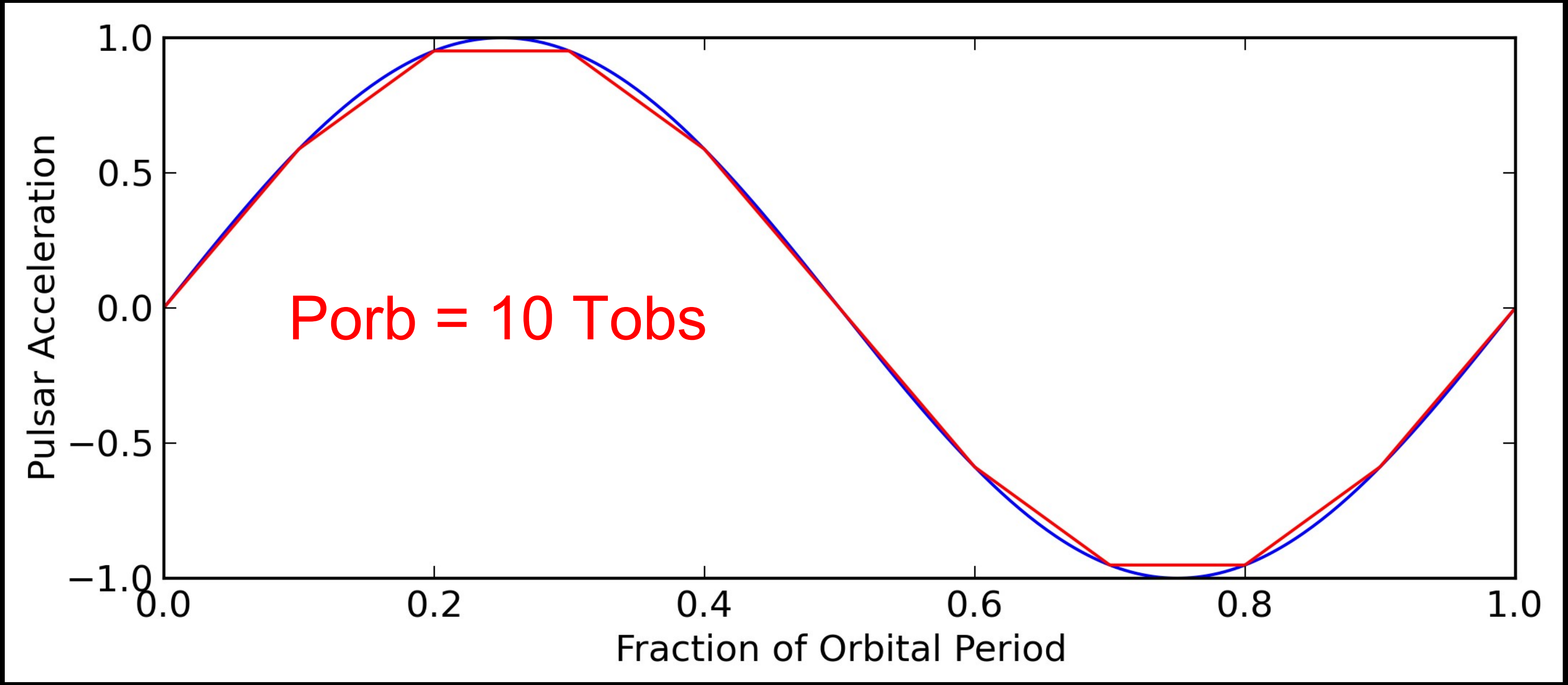
- Isolated Pulsars
  - Fourier analysis
- Binary $P_{orb} > 10T_{obs}$
  - "Acceleration" Searches
- Binary $P_{orb} \sim T_{obs}$
  - "Dynamic" Power Spectra
- Binary $P_{orb} << T_{obs}$
  - "Sideband" Searches



3ms pulsar, 2 hr obs

What are acceleration searches?

- Pulsar binaries typically have circular orbits
- Position, Velocity, and Accel. vs. time are all sinusoids
- If the orbital period >> observation time, then the acceleration is approx constant during the observation
- A “chirp” with small Δf/f (phase changes quadratically)



Two ways to implement...

- Constant acceleration gives a quadratic change of signal phase (i.e. this is a chirp with small Δf/f)
- Time domain (e.g. Jonhston & Kulkarni 1991 etc)
  - Use a time transform to quadratically stretch/compress the full input time series
  - Each acceleration trial is a new stretched time series followed by a long FFT
- Freq domain (e.g. Ransom, Eikenberry & Middleditch 2002)
  - Correct phase change in Fourier domain by applying complex matched filters (ala coherent dedispersion)
  - One long input FFT is operated on by many short filters, usually via FFT convolution/correlation

Two ways to implement...

- Constant acceleration gives a quadratic change of signal phase (i.e. this is a chirp with small Δf/f)
- Time domain (e.g. Jonhston & Kulkarni 1991 etc)

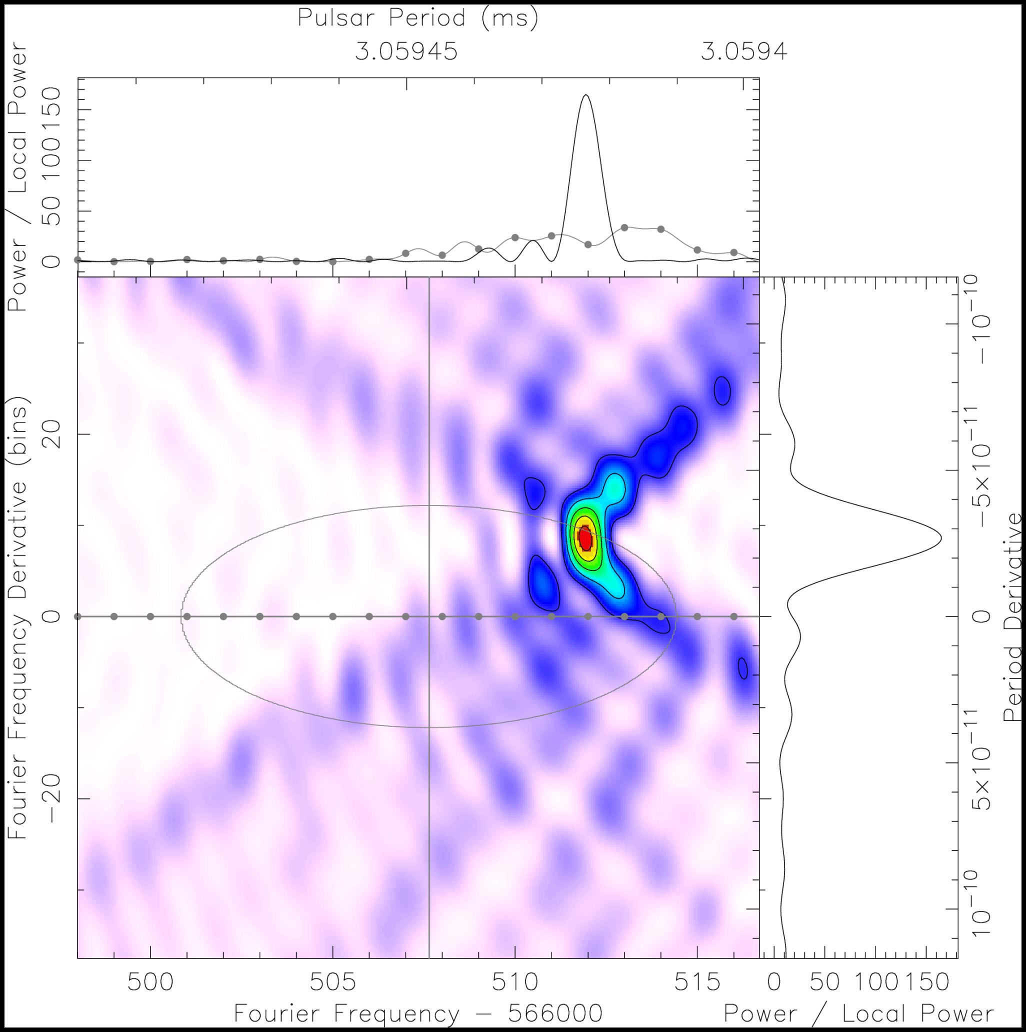
$$\tau(t) = \tau_0(1 + v(t)/c) \sim \tau_0(1 + at/c) \propto at^2/c$$

  - Each acceleration trial is a new stretched time series followed by a long FFT
- Freq domain (e.g. Ransom, Eikenberry & Middleditch 2002)
  - Correct phase change in Fourier domain by applying complex matched filters (ala coherent dedispersion)
  - One long input FFT is operated on by many short filters, usually via FFT convolution/correlation

Fourier-Domain Acceleration Searches

- Short correlations, computed via FFTs, with the Fourier amplitudes of the full time series, exactly remove linear acceleration
- Uniformly tile "f-fdot" plane
- Fourier bins drifted 'z' is related to acceleration:

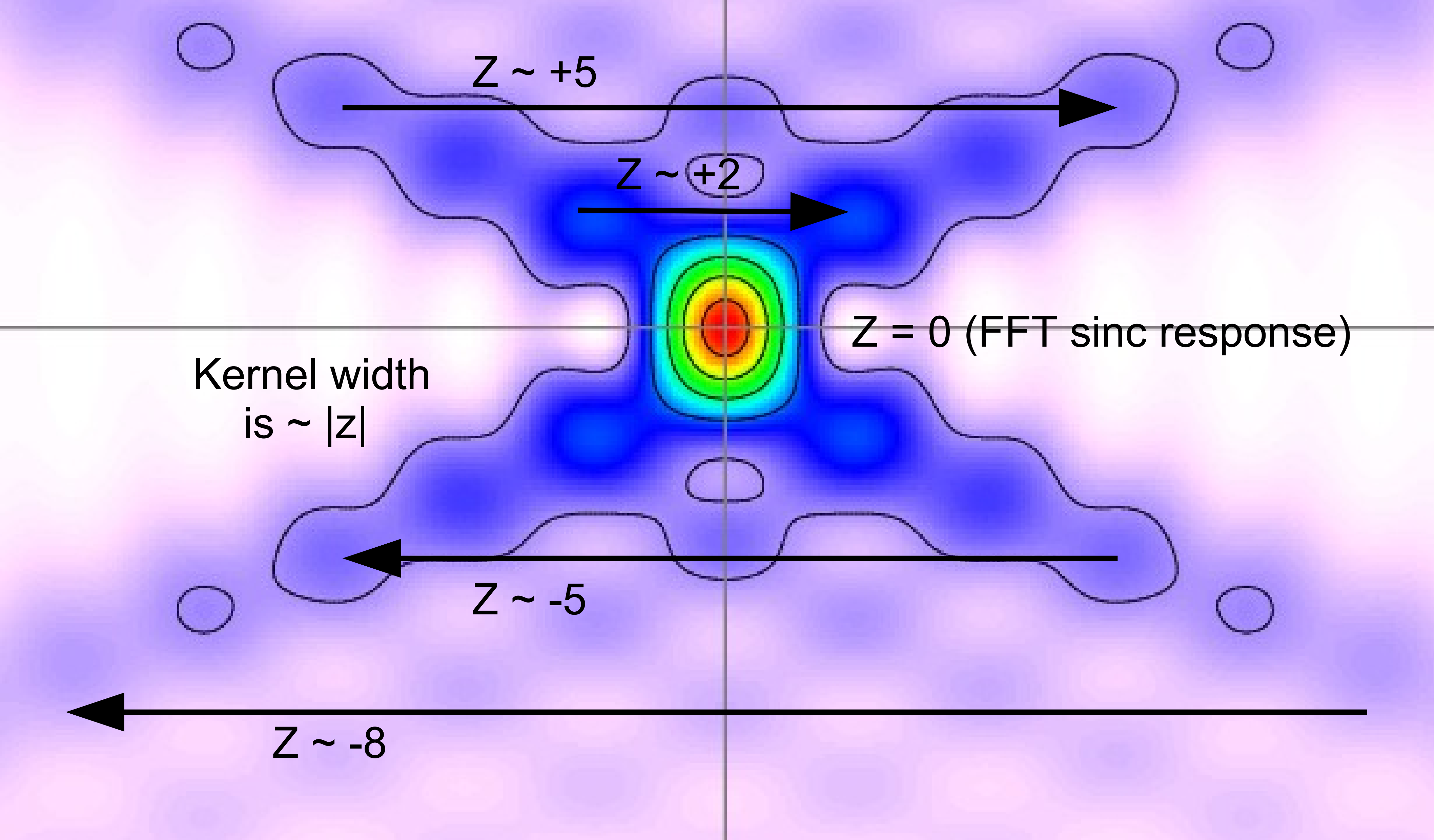
$$\mathrm{accel} = \frac{\dot{f}c}{f} = \frac{zc}{fT^2}$$



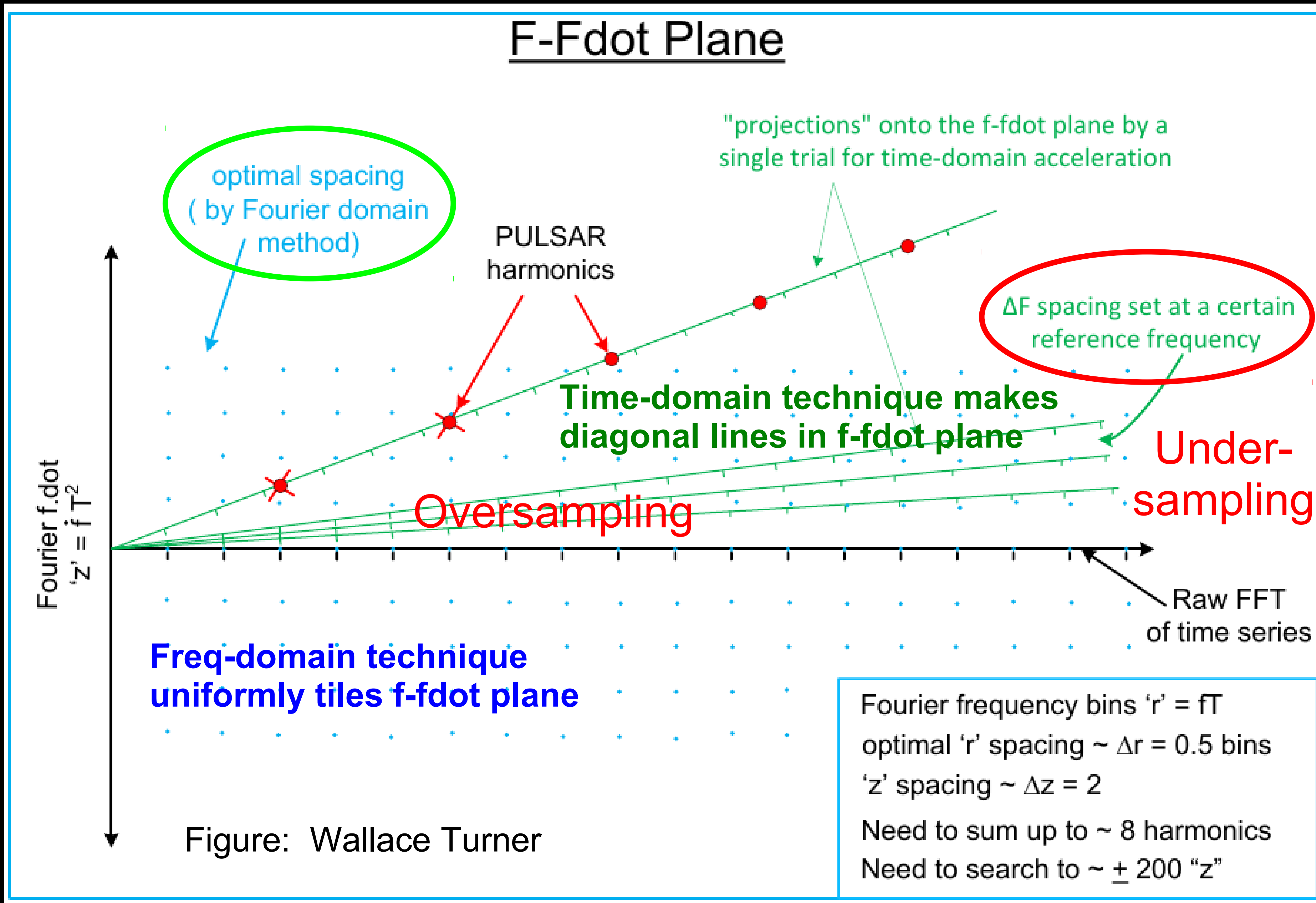
Fundamental harmonic of accelerating MSP J1807-2459A in f-fdot plane

Correlation “kerrnels” are horizontal slices

(highly oversampled; phases not shown)



Comparison with time-domain method

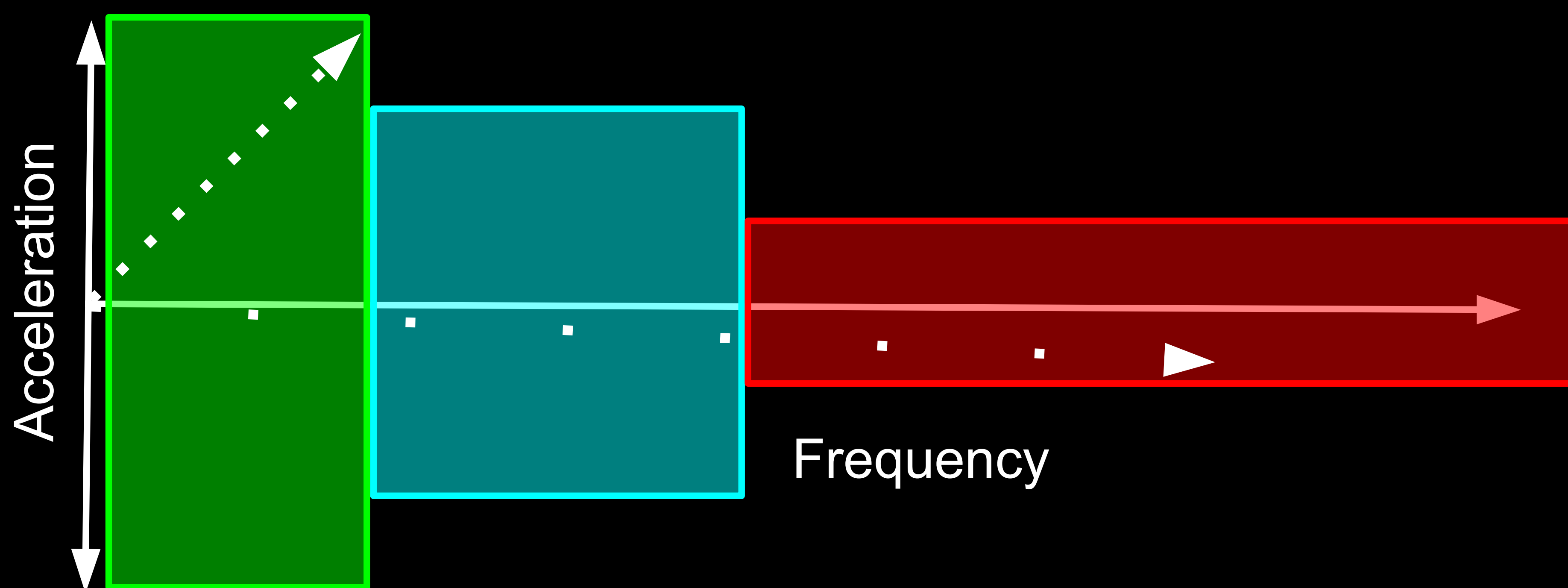


Fourier Domain Pros/Cons

- Pros:
  - The f-fdot plane is optimally sampled in both the 'f' and 'fdot' directions over the full parameter range
  - Correlations to compute parts of f-fdot plane are short, memory local, and therefore fast
  - Maximum “z” value is flexible. Gets high-accel slow pulsars (i.e. PSR-BH system), mid-accel mildly-recycled pulsars (i.e DNS systems), and low-accel MSPs: “a” and “f” offset each other

Tuned Acceleration for Binary Type

- Fourier domain method allows flexibility in frequency vs acceleration amount:
  - Black hole binaries: likely slow PSRs w/ high accel
  - NS-NS binaries: likely 10s-of-ms PSRs w/ med accel
  - MSP binaries: fast PSRs w/ low accel

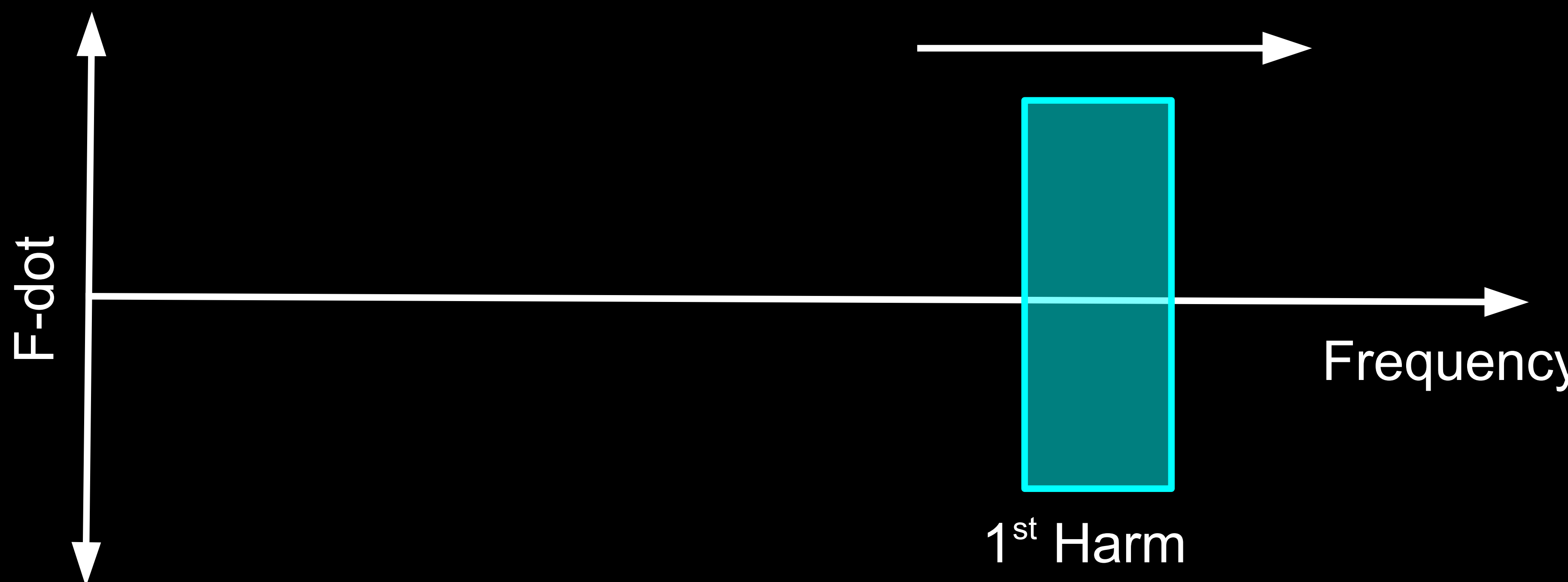


Fourier Domain Pros/Cons

- Pros:
  - The f-fdot plane is optimally sampled in both the 'f' and 'fdot' directions over the full parameter range
  - Correlations to compute parts of f-fdot plane are short, memory local, and therefore fast
  - Maximum “z” value is flexible. Gets high-accel slow pulsars (i.e. PSR-BH system), mid-accel mildly-recycled pulsars (i.e DNS systems), and low-accel MSPs: “a” and “f” offset each other
- Cons:
  - Algorithm (and therefore code) is significantly more complex than time-domain technique
  - Has not yet been fully GPU-ized... work in progress

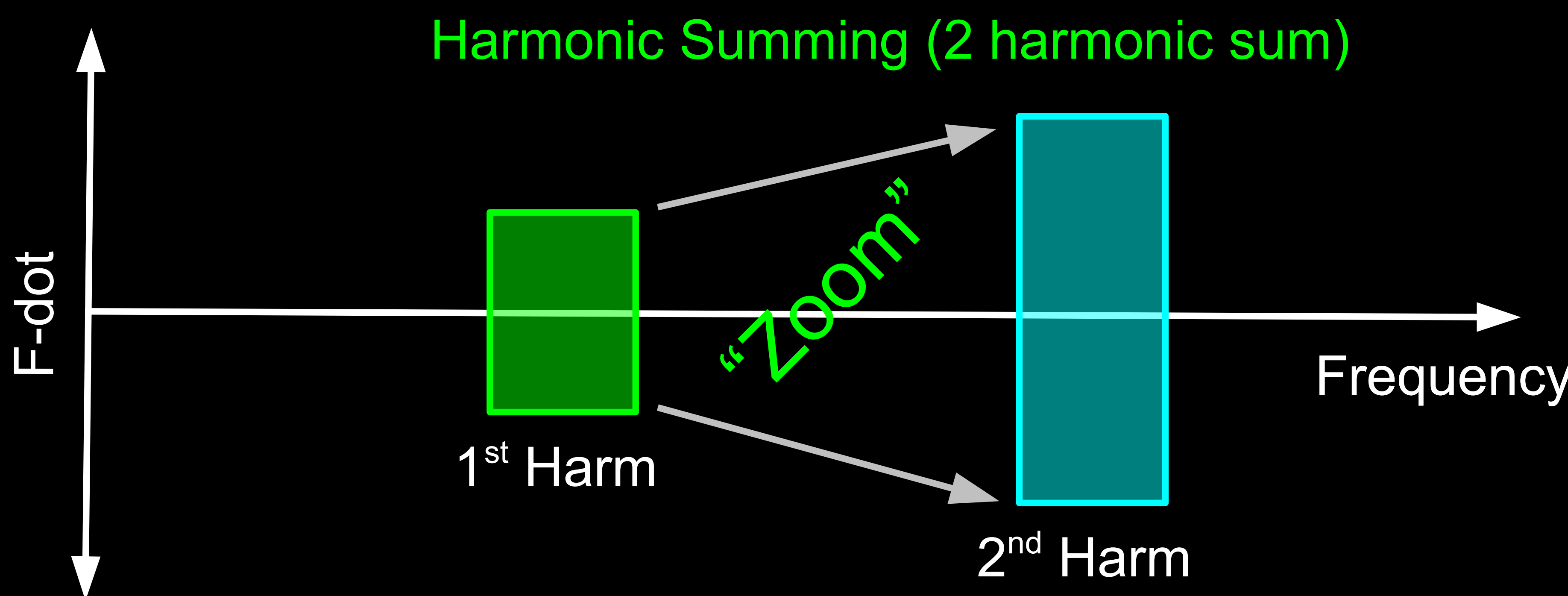
Opt #1: F-Fdot plane in RAM

- Current `accelsearch` (from `PRESTO`) algorithm was designed for very long time series (>100Mpts) and so computes f-fdot plane in chunks as well as re-computes sub-harmonics



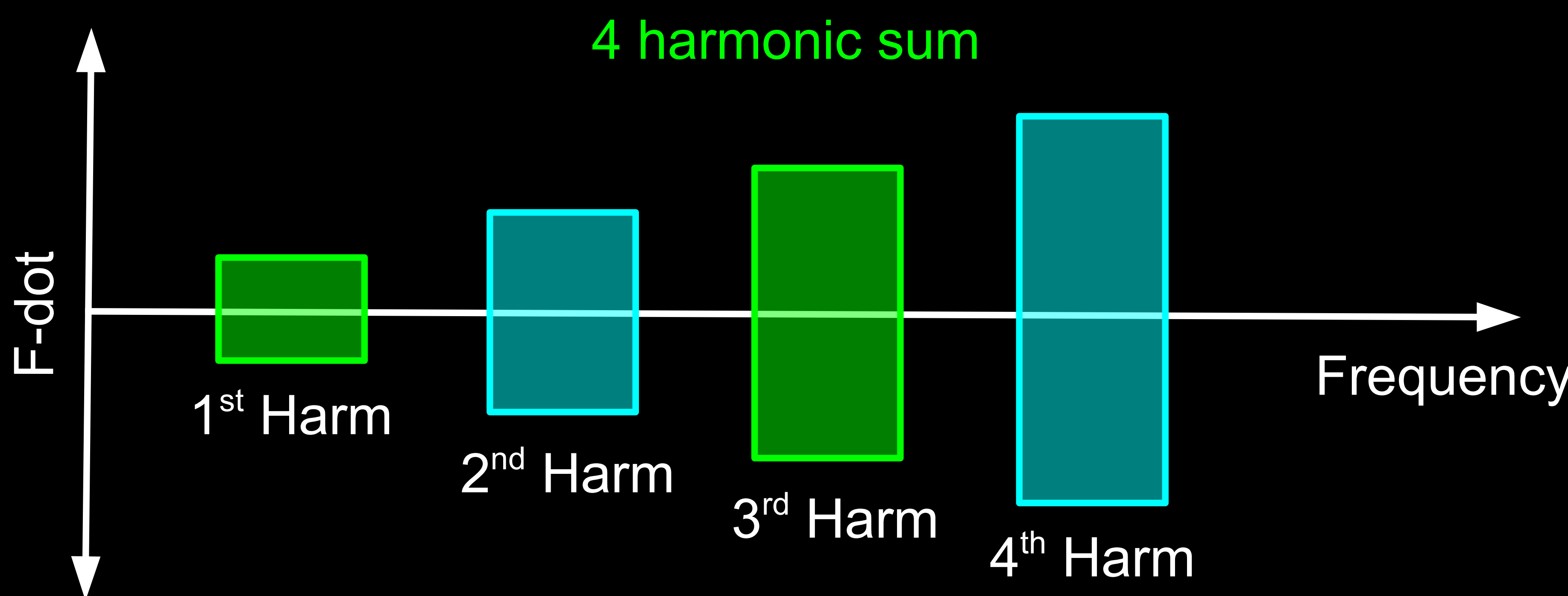
Opt #1: F-Fdot plane in RAM

- Current `accelsearch` (from `PRESTO`) algorithm was designed for very long time series (>100Mpts) and so computes f-fdot plane in chunks as well as re-computes sub-harmonics



Opt #1: F-Fdot plane in RAM

- Current `accelsearch` (from `PRESTO`) algorithm was designed for very long time series (>100Mpts) and so computes f-fdot plane in chunks as well as re-computes sub-harmonics

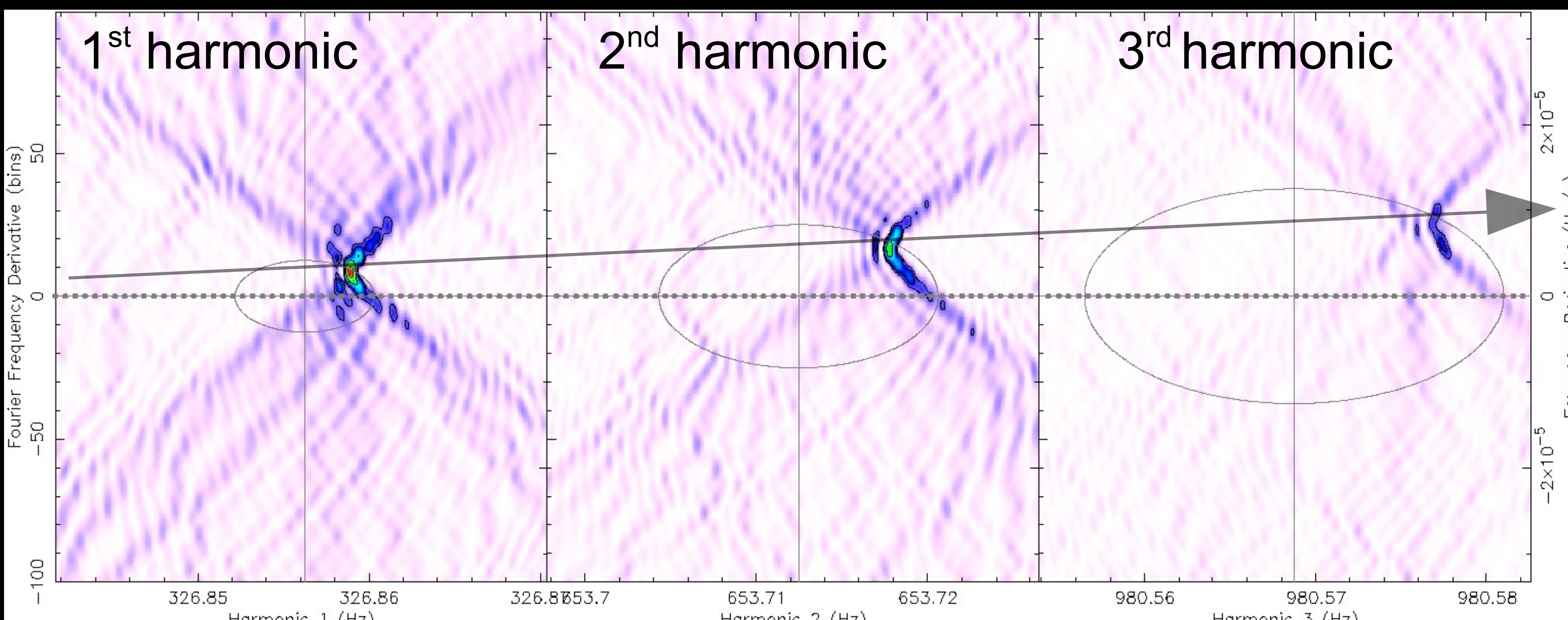


Opt #1: F-Fdot plane in RAM

- Current `accelsearch` (from `PRESTO`) algorithm was designed for very long time series (>100Mpts) and so computes f-fdot plane in chunks as well as re-computes sub-harmonics
- If initial time series is short enough, we can tile the full f-fdot plane into RAM. This would cause an instantaneous speedup of ~Nharm/2 times.
- For SKA1: 600s integrations with 50us dumps gives ~12Mpt time series. For 100 accelerations and Fourier interpolation, we would require only about 10 GB of RAM. Doable on GPU or FPGA.

Opt #2: Clever Harmonic Summing

- Narrow pulses produce many harmonics
- Summing of 2, 4, 8, 16, and sometimes 32 harmonics
- In F-Fdot plane, accomplished by “zooming” in F and Fdot directions, a 2D region around the sub-harmonic
- Interpolation and scaling by using GPU texture memory?



Summary / Request for Ideas

- Acceleration searches are crucial for SKA, and (I argue) frequency-domain versions are better
- GPU-ization of `accelsearch` currently has 8-10x speed-up with extremely minimal code changes (by Jintau Luo)
  - Not fast enough to be worthwhile (Note that the current `accelsearch` is highly optimized on CPU)
- Current algorithm is not optimized for short duration search pointings, or for GPU memory:
  - Put full F-Fdot plane in GPU RAM
  - Improved harmonic summing (i.e. texture memory)
  - ***Other ideas?***