Porting Signal Processing Algorithms to CuPy for precision measurement

Porting Signal Processing Algorithms to CuPy for precision measurement

Acknowledgements: D.Cobas, M.Lipinski, M.Sosin, P.Peronnard, T.Gingold, C.Franco, T.Wlostowski (CERN)



Signal Processing in FSI

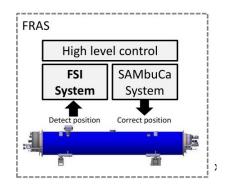
CuPy and Signal Processing

- Butterworth Filter
- Hilbert Transform
- Savitzky-Golay Filter

Outlook



- Frequency scanning interferometry measurement system for Full Remote Alignment System (FRAS), which can determine distance from measuring head to target upto micrometer precision in real time
- Monitoring the position of magnet and crab cavity cold masses inside their cryostats
- Based on Michelson Interferometry Principle and uses sweeping laser to identify distance of target system

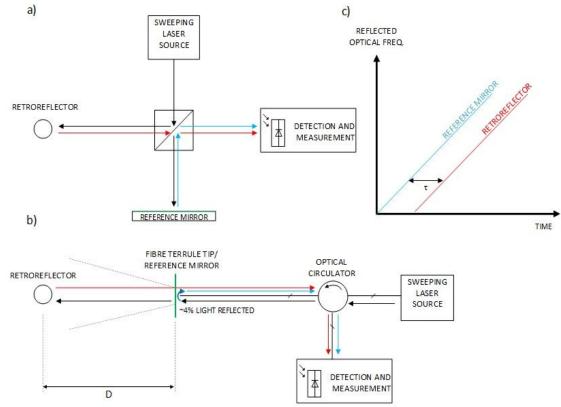


https://home.cern/news/news/accelerators/aligning-hl-lhc-magnets-interferometry

- Based on Michelson Interferometry Principle and uses sweeping laser to identify distance of target system
- Reference beam and the beam reflected from the target are recombined, creating an interference signal -

 $I(t,\tau) = A \cdot \cos[2\pi(\alpha \tau t + f \circ \tau)]$

- A magnitude of the signal
- т time delay between signals
- α sweep rate of the laser



- Based on Michelson Interferometry Principle and uses sweeping laser to identify distance of target system
- Reference beam and the beam reflected from the target are recombined, creating an interference signal -

 $I(t,\tau) = A \cdot \cos[2\pi(\alpha \tau t + f \circ \tau)]$

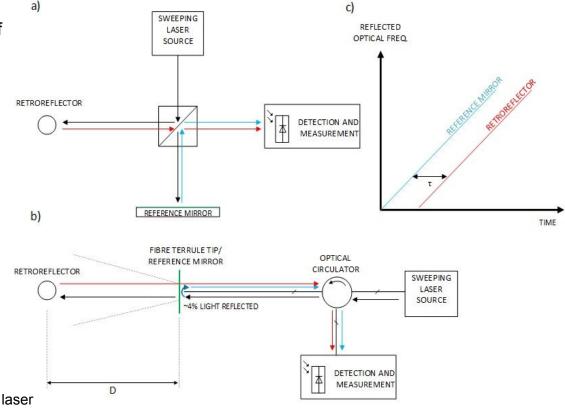
- A magnitude of the signal
- τ time delay between signals
- α sweep rate of the laser

• Distance D is calculated - D =
$$c \frac{N}{2\Delta v n}$$

 $\varDelta \nu$ - change of the laser frequency during sweep n -refractive index

c -speed of light

N -number of cycles of the signal measured during the laser sweep (above equation)



- Based on Michelson Interferometry Principle and uses sweeping laser to identify distance of target system
- Reference beam and the beam reflected from the target are recombined, creating an interference signal -

 $I(t,\tau) = A \cdot \cos[2\pi(\alpha \tau t + f \circ \tau)]$

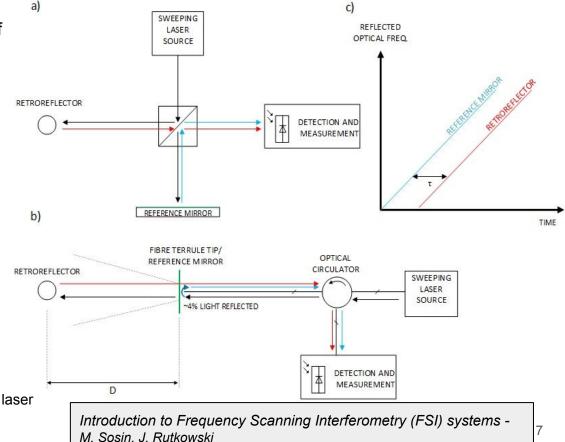
- A magnitude of the signal
- τ time delay between signals
- α sweep rate of the laser

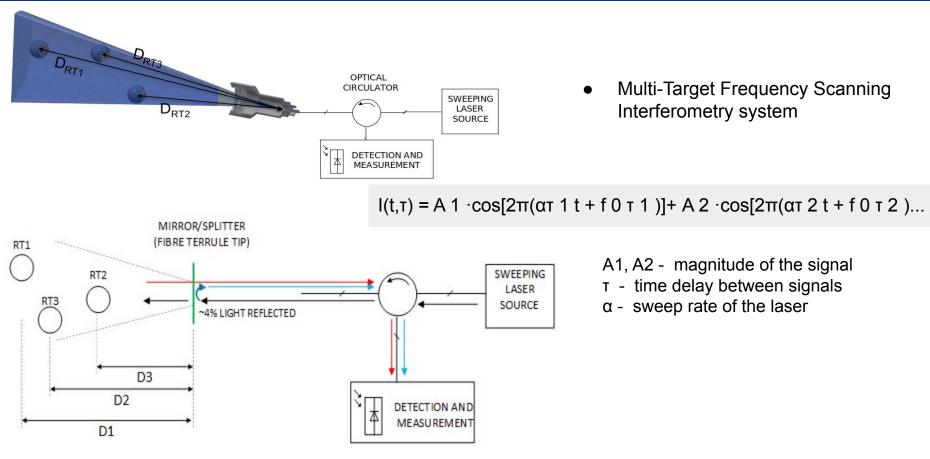
• Distance D is calculated - D =
$$c \frac{N}{2\Delta v n}$$

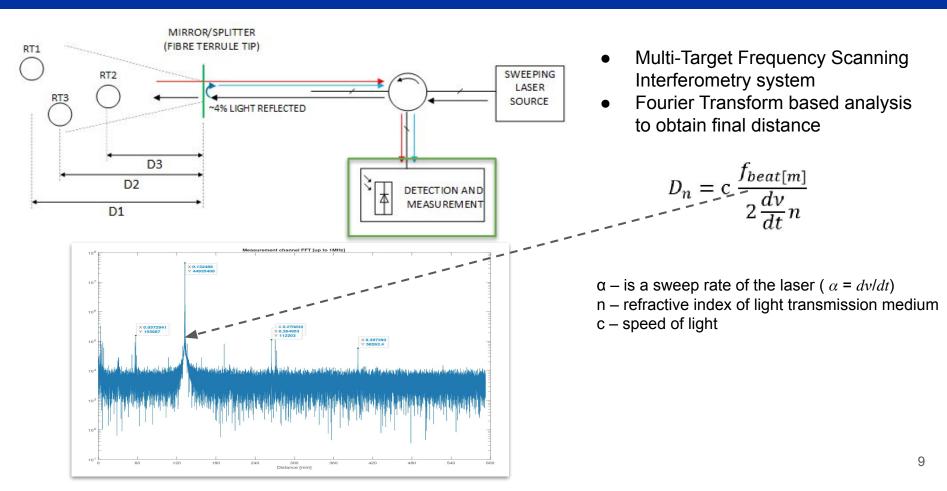
 $\varDelta \nu$ - change of the laser frequency during sweep n -refractive index

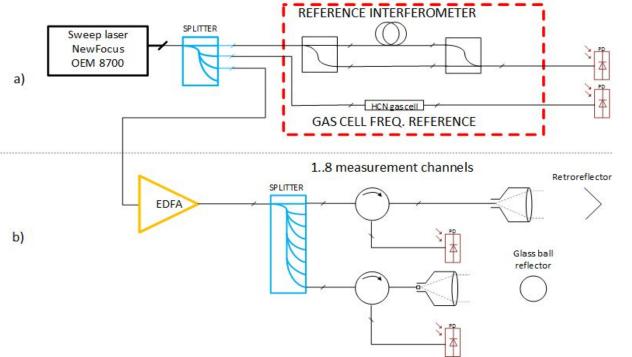
c -speed of light

N -number of cycles of the signal measured during the laser sweep (above equation)

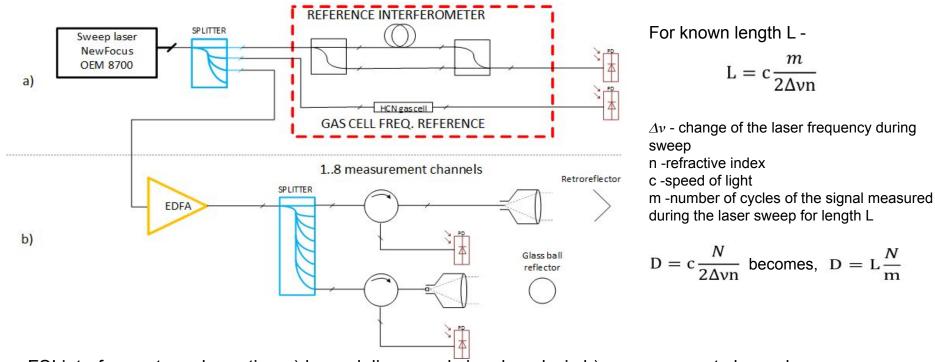




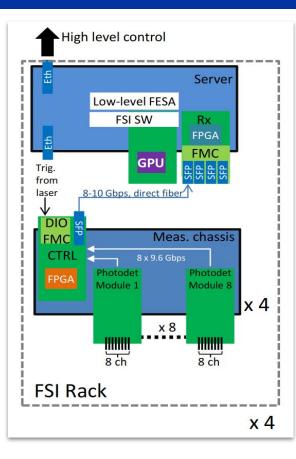




- FSI interferometer schematic a) laser delivery and signal analysis b) measurement channels
- Reference Interferometer to identify laser sweep (α)

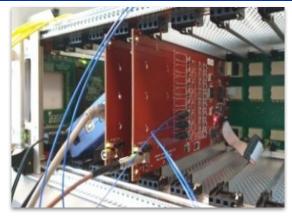


- FSI interferometer schematic a) laser delivery and signal analysis b) measurement channels
- Reference Interferometer to identify laser sweep (Δv) or (α)





FSI Test Setup



FSI Photodetector Module

GPU: Nvidia RTX 3060

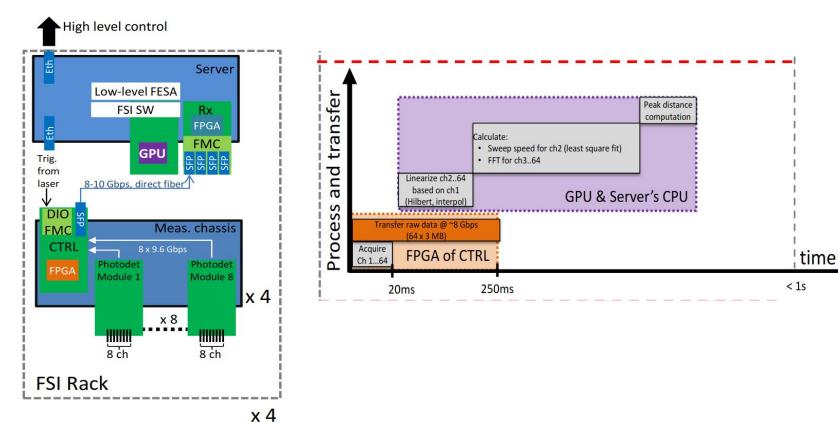


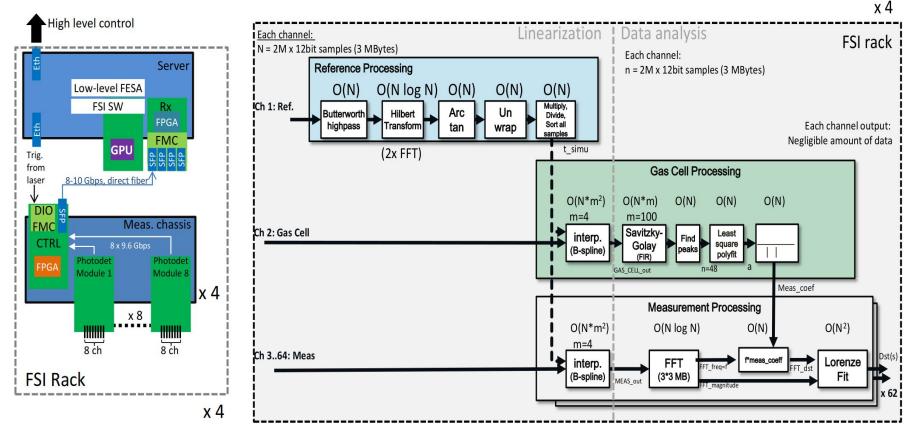
Signal Processing in FSI

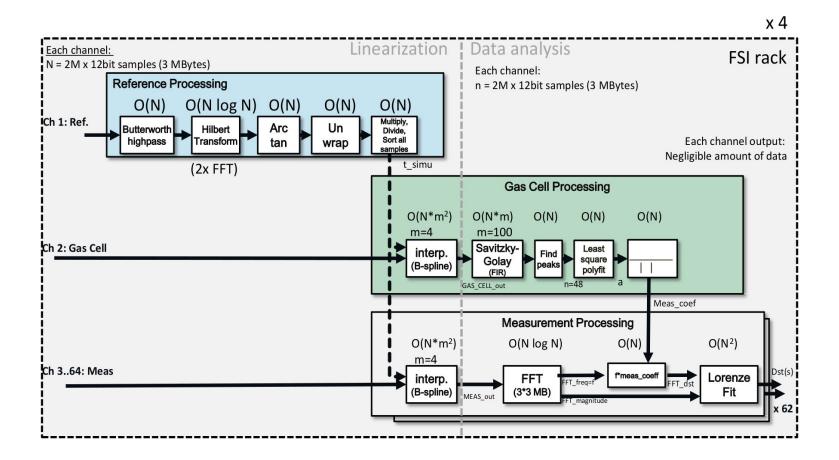
CuPy and Signal Processing

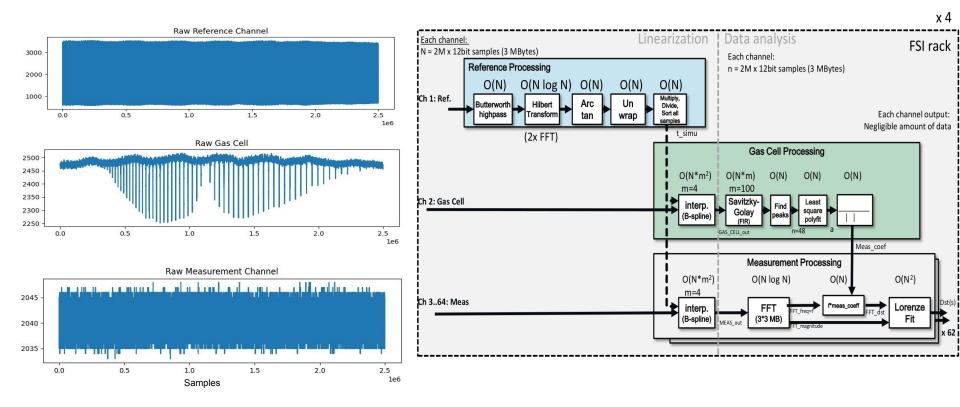
- Butterworth Filter
- Hilbert Transform
- Savitzky-Golay Filter

Outlook











Signal Processing in FSI

CuPy and Signal Processing

- Butterworth Filter
- Hilbert Transform
- Savitzky-Golay Filter

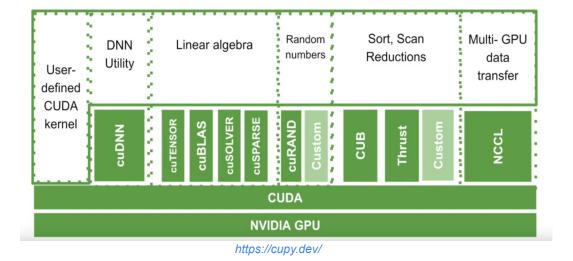
Outlook

- It is an open-source matrix library accelerated with NVIDIA CUDA.
- It uses CUDA-related libraries including cuBLAS, cuDNN, cuRand, cuSolver, cuSPARSE, cuFFT, and NCCL to make full use of the GPU architecture



- It is an open-source matrix library accelerated with NVIDIA CUDA.
- It uses CUDA-related libraries including cuBLAS, cuDNN, cuRand, cuSolver, cuSPARSE, cuFFT, and NCCL to make full use of the GPU architecture





- It is an open-source matrix library accelerated with NVIDIA CUDA.
- It uses CUDA-related libraries including cuBLAS, cuDNN, cuRand, cuSolver, cuSPARSE, cuFFT, and NCCL to make full use of the GPU architecture
- Provides High performance N-dimensional array computation
- Drop in replacement for Numpy https://docs.cupy.dev/en/stable/reference/comparison.html



- It is an open-source matrix library accelerated with NVIDIA CUDA.
- It uses CUDA-related libraries including cuBLAS, cuDNN, cuRand, cuSolver, cuSPARSE, cuFFT, and NCCL to make full use of the GPU architecture
- Provides High performance N-dimensional array computation
- Drop in replacement for Numpy https://docs.cupy.dev/en/stable/reference/comparison.html
- Open Source and distributed under MIT License
- Easy to start with and scale and test
- Develop custom Kernels using JIT NUMBA



CuPy and Signal Processing Algorithms

Support for some of the Scipy routines is available:

- Discrete Fourier Transform fft, rfft, ifft, fft2, irfft, fftshift
- Linear Algebra lu, eigsh, lsqr
- Multidimensional Image processing gaussian_filter, laplace, convolve, grey_dilation, grey_erosion
- Signal Processing fftconvolve, correlate, medfit
- Sparse Matrices and many more

https://docs.cupy.dev/en/stable/reference/scipy.html#

CuPy and Signal Processing Algorithms

Support for some of the Scipy routines is available:

- Discrete Fourier Transform fft, rfft, ifft, fft2, irfft, fftshift
- Linear Algebra lu, eigsh, lsqr
- Multidimensional Image processing gaussian_filter, laplace, convolve, grey_dilation, grey_erosion
- Signal Processing fftconvolve, correlate, medfit
- Sparse Matrices ... and many more

https://docs.cupy.dev/en/stable/reference/scipy.html#

• Cusignal - RAPIDS

https://docs.rapids.ai/api/cusignal/stable/api.html

CuPy and Signal Processing Algorithms

Considerations while porting to GPU:

1] Check the data format

2] Check number of Device to Host and Host to Device Memory Transactions

3] No recursion functions are present

4] GPU is good if you have large data set to process and have possibility of either Data parallelism or Task parallelism



Signal Processing in FSI

CuPy and Signal Processing

- Butterworth Filter
- Hilbert Transform
- Savitzky-Golay Filter

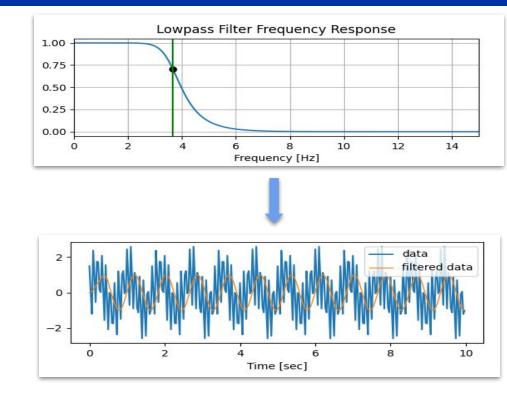
Outlook

Butterworth Filter

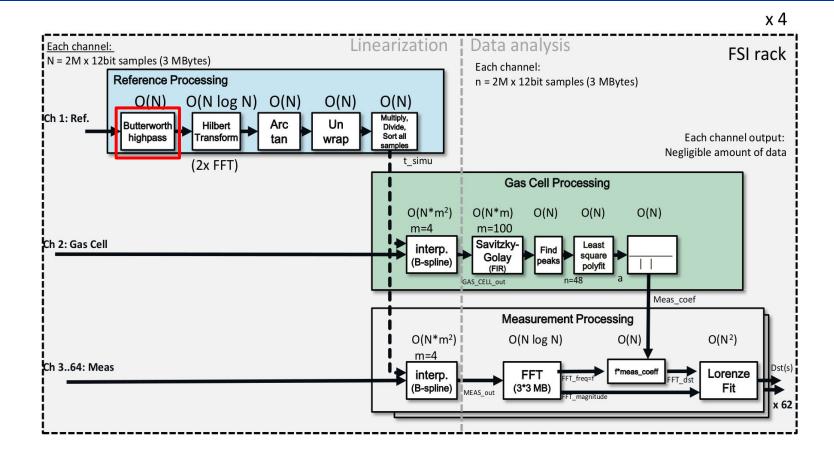
- To reduce the background noise and suppress the interfering signals by removing some frequencies - filters are used
- The frequency range which is allowed : passband and the range which is suppressed is stopband
- Butterworth filter provides maximum flat response in passband i.e least ripple
- Transfer Function of Butterworth Filter:

$$\left|H_{b}\left(jw\right)\right| = \frac{1}{\sqrt{1 + \left(w/w_{c}\right)^{2N}}}$$

 w_c = cut-off frequency N = Order of Filter



Butterworth High Pass Filter



Butterworth High Pass Filter in CuPy

1] Calculate z,p,k for Lowpass analog prototype

```
z = cp.array([])
m = cp.arange(-N+1, N, 2)
p = -cp.exp(1j * pi * m / (2 * N))
k = 1
```

2] Pre-warp frequencies for Digital Filter

```
warped = 2 * fs * cp.tan(pi * Wn / fs)
```

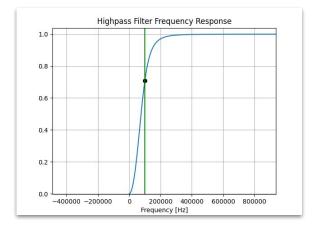
3] Convert Lowpass analog prototype to Highpass, wo= cutoff frequency

```
z_hp = wo / z
p_hp = wo / p
z_hp = cp.append(z_hp, cp.zeros(degree))
k_hp = k * cp.real(cp.prod(-z) / cp.prod(-p))
```

4] Return digital filter parameters using Bilinear Transformation fs = 2.0*fs

```
z_z = (fs + z) / (fs - z)
p_z = (fs + p) / (fs - p)
z_z = cp.append(z_z, -cp.ones(degree))
k_z = k * cp.real(cp.prod(fs - z) / cp.prod(fs - p))
```

5] Convert to b/a form from z,p,k



Performance Analysis: Butterworth Filter

1] Calculate filter Transfer Function

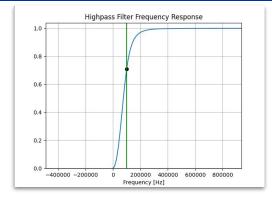
```
nyq = 0.5 * fs
normal_cutoff = cutoff / nyq
b, a = butter(order, normal_cutoff, btype='high', analog=False)
```

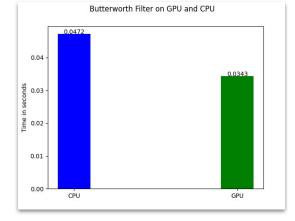
2] Apply using lfilter

```
data=Reference_cell
ret = lfilter(b, a, data)
```

3] Apply using FFT

```
delta = np.zeros(np.size(t))
delta[1] = 1;
filter_butter = lfilter(b, a, delta)
filter_butter = cp.array(filter_butter)
filter_fft = cupyx.scipy.fft.fft(filter_butter)
data_fft = cupyx.scipy.fft.fft(data)
res_fft = cp.multiply(data_fft, filter_fft)
res_fft = cp.array(res_fft)
res = cupyx.scipy.fft.irfft(res_fft)
```







Signal Processing in FSI

CuPy and Signal Processing

- Butterworth Filter
- Hilbert Transform
- Savitzky-Golay Filter

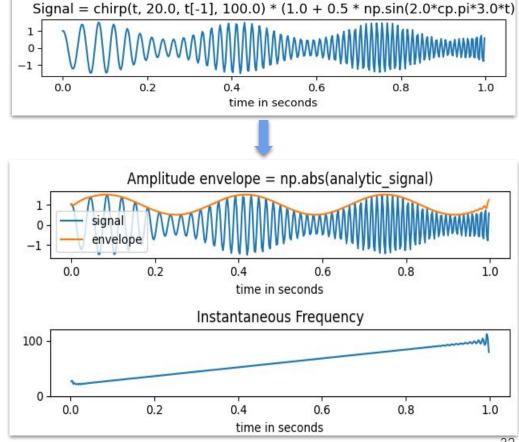
Outlook

Hilbert Transform

- It is useful for calculating instantaneous attributes of a time series, especially the amplitude and the frequency.
- The instantaneous amplitude is the amplitude of the complex Hilbert transform and the instantaneous frequency is the time rate of change of the instantaneous phase angle.
- It returns Analytic Signal 'x'

 $x = x_r + jx_i$ x_r is the original data

 x_i and an imaginary part, which contains the Hilbert transform. The imaginary part is a version of the original real sequence with a 90° phase shift



Hilbert Transform in CuPy

1] Compute Fast Fourier Transform of Real-valued Signal

```
Xf = cupyx.scipy.fft.fft(x, N, axis=axis)
h = cp.zeros(N)
```

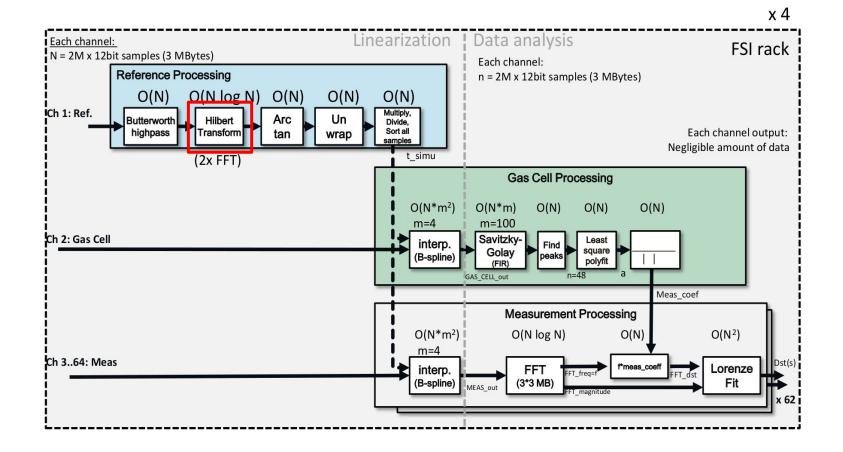
2] Rotate the Fourier Coefficients to obtain imaginary part

```
if N % 2 == 0:
    h[0] = h[N // 2] = 1
    h[1:N // 2] = 2
else:
    h[0] = 1
    h[1: (N + 1) // 2] = 2
if x.ndim > 1:
    ind = [cp.newaxis] *
    x.ndim
    h = [cp.newaxis] *
    x.ndim
```

3] Compute Inverse Fourier Transform to get the Analytic Signal

```
x = cupyx.scipy.fft.ifft(Xf * h, axis=axis)
```

4] Calculate Instantaneous Frequency and phase



Hilbert Transform of Reference Cell Data

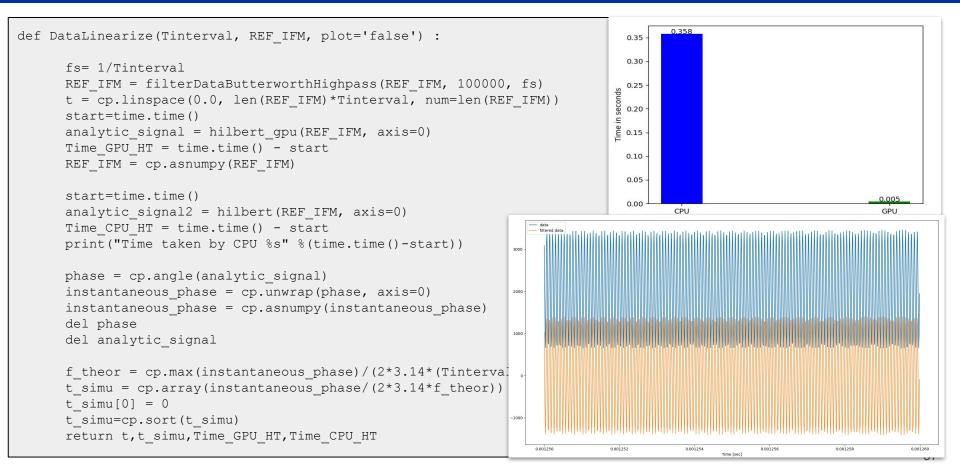
```
def DataLinearize(Tinterval, REF IFM, plot='false') :
```

```
fs= 1/Tinterval
REF IFM = filterDataButterworthHighpass(REF IFM, 100000, fs)
t = cp.linspace(0.0, len(REF IFM)*Tinterval, num=len(REF IFM))
start=time.time()
analytic signal = hilbert gpu(REF IFM, axis=0)
Time GPU HT = time.time() - start
REF IFM = cp.asnumpy(REF IFM)
start=time.time()
analytic signal2 = hilbert(REF IFM, axis=0)
Time CPU HT = time.time() - start
print("Time taken by CPU %s" %(time.time()-start))
phase = cp.angle(analytic signal)
instantaneous phase = cp.unwrap(phase, axis=0)
instantaneous phase = cp.asnumpy(instantaneous phase)
del phase
del analytic signal
f theor = cp.max(instantaneous phase)/(2*3.14*(Tinterval*len(instantaneous phase)))
t simu = cp.array(instantaneous phase/(2*3.14*f theor))
t simu[0] = 0
t simu=cp.sort(t simu)
return t,t simu, Time GPU HT, Time CPU HT
```

Performance Analysis: Hilbert Transform

```
def DataLinearize(Tinterval, REF IFM, plot='false') :
                                                                                      0.358
                                                                                 0.35
      fs= 1/Tinterval
                                                                                 0.30
      REF IFM = filterDataButterworthHighpass(REF IFM, 100000, fs)
      t = cp.linspace(0.0, len(REF IFM)*Tinterval, num=len(REF IFM))
                                                                                0.25
                                                                               seconds
      start=time.time()
      analytic signal = hilbert gpu(REF IFM, axis=0)
                                                                                0.20
                                                                               .⊑
      Time GPU HT = time.time() - start
                                                                               Time i
                                                                                0.15
      REF IFM = cp.asnumpy(REF IFM)
                                                                                0.10
      start=time.time()
      analytic signal2 = hilbert(REF IFM, axis=0)
                                                                                 0.05
      Time CPU HT = time.time() - start
                                                                                                                  0.005
      print("Time taken by CPU %s" %(time.time()-start))
                                                                                 0.00
                                                                                       CPU
                                                                                                                  GPU
      phase = cp.angle(analytic signal)
      instantaneous phase = cp.unwrap(phase, axis=0)
      instantaneous phase = cp.asnumpy(instantaneous phase)
      del phase
      del analytic signal
      f theor = cp.max(instantaneous phase)/(2*3.14*(Tinterval*len(instantaneous phase)))
      t simu = cp.array(instantaneous phase/(2*3.14*f theor))
      t simu[0] = 0
      t simu=cp.sort(t simu)
      return t,t simu, Time GPU HT, Time CPU HT
```

Performance Analysis: Hilbert Transform



Performance Analysis: Hilbert Transform

Timeline View 👻						📼 Q 1x	0	A 3 warnings, 12 n
3s - 🖛 + 368.7ms	+368.72ms +368.741 3s 3	368.7536ms ms +368.78ms +368.8ms	+368.82ms	+368.84ms	+368.86ms	+368.88ms	+368.9ms	+368.92ms +368.94ms
PU (12)								
UDA HW (NVIDIA GeForce R								
11.8% Kernels			regular_fft					
76.2% regular_fft			regular_fft					
46.1% void regular_fft<		regular fft Begins: 3.36868s	regular_fft					
30.8% void regular_fft<2		Ends: 3.36895s (+271.072 µs)						
23.1% void regular_fft<1		grid: <<<400, 1, 1>>> block: <<<8, 25, 1>>>						
6.7% cupy_true_dividecc		Launch Type: Regular Static Shared Memory: 0 bytes						
5.1% cupy_sinfloat64_flo		Dynamic Shared Memory: 8,000 bytes						
6 kernel groups hidd — +		Registers Per Thread: 48 Local Memory Per Thread: 0 bytes						
88.2% Memory		Local Memory Total: 24,772,608 bytes Shared Memory executed: 65,536 bytes						
hreads (31)		Shared Memory Bank Size: 4 B						
		Theoretical occupancy: 72.9167 % Launched from thread: 16759						
✓ [16759] python3.7 →		Latency: ←176.393 µs Correlation ID: 684						
CUDA API		Stream: Default stream 7						
Profiler overhead								
✔ [16765] python3.7 -								
• [10/03] pytions./ •								
✔ [16764] python3.7 -								
nts View 👻								
							Name	-
 Name 			Start	Duration	GPU	Context	Description:	
regular_fft			3.36868s	271.072 µs	GPU 0	Stream 7	regular_fft	
regular_fft			3.37483s	272.767 µs	GPU 0	Stream 7	Begins: 3.36 Ends: 3.3689	868s 95s (+271.072 μs)
							grid: <<<4 block: <<<8	00, 1, 1>>>
							Launch Type	Begular
							Static Share	d Memory: 0 bytes ared Memory: 8,000 bytes
							Registers Pe	r Thread: 48
							Local Memor	y Per Thread: 0 bytes y Total: 24,772,608 bytes
							Shared Mem	ory executed: 65,536 bytes ory Bank Size: 4 B
							Theoretical of	occupancy: 72.9167 %
							Launched fro Latency: ←1	om thread: 16759 76 393 us

Performance of FFT Cupy Kernel in timeline view



Frequency Scanning Interferometry System

Signal Processing in FSI

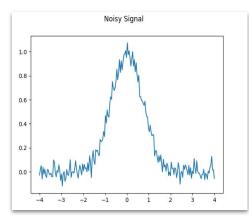
CuPy and Signal Processing

- Butterworth Filter
- Hilbert Transform
- Savitzky-Golay Filter

Outlook

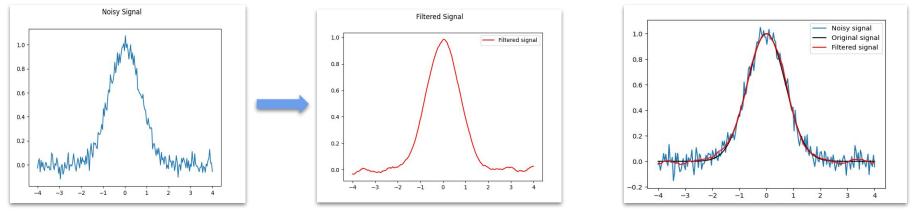
Savitzky-Golay Filter

- It is a digital filter that can be applied to a set of digital data points for smoothing the data without distorting the original signal tendency or to calculate the derivative of signal.
- Find least-square fit for each window and replace each data point with coefficient of that polynomial
- But the smoothed output obtained by fitting polynomial to each window is equivalent to convolution of convolution coefficients(weighting coefficients) with each window/segment



Savitzky-Golay Filter

- It is a digital filter that can be applied to a set of digital data points for smoothing the data without distorting the original signal tendency or to calculate the derivative of signal.
- Find least-square fit for each window and replace each data point with coefficient of that polynomial
- But the smoothed output obtained by fitting polynomial to each window is equivalent to convolution of 'convolution coefficients(weighting coefficients)' with each window/segment



Savitzky-Golay Filter in CuPy

1] Precompute the coefficients based on order and window length

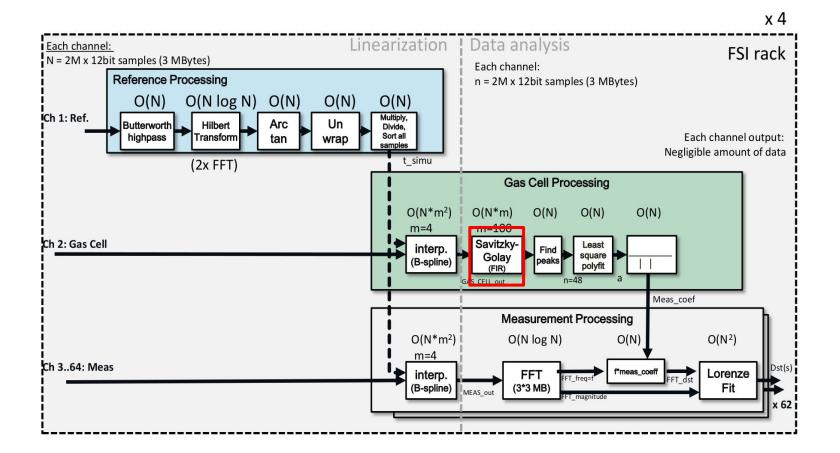
```
b = cp.array([[k**i for i in range(order+1)] for k in range(-half_window, half_window+1)])
m = cp.linalg.pinv(b)
m = cp.multiply(m[deriv] , cp.multiply(cp.power(rate,deriv), factorial(deriv)))
```

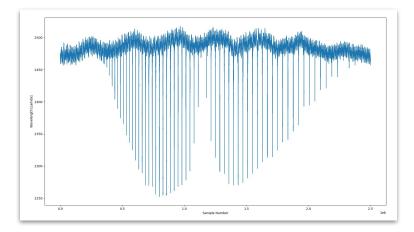
2] Pad the signal at the extremes

```
extr_begin = y[0] - cp.abs( y[1:half_window+1][::-1] - y[0] )
extr_end = y[-1] + cp.abs(y[-half_window-1:-1][::-1] - y[-1])
y = cp.concatenate((extr_begin, y, extr_end))
```

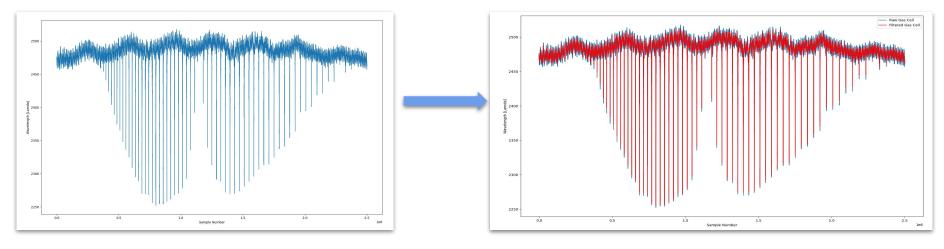
3] Convolve signal with calculated coefficients

result = cp.convolve(m[::-1], y, mode='valid')



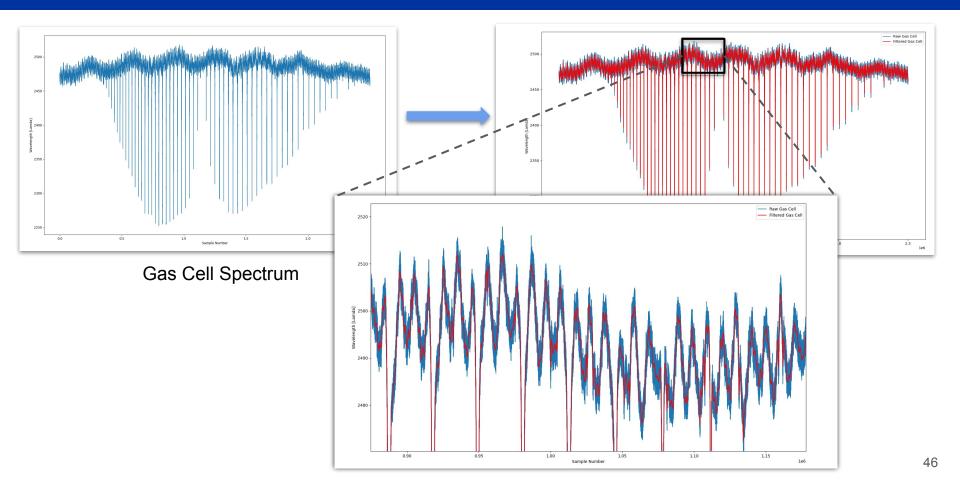


Spectrum of Hydrogen Cyanide (HCN) SRM2519a absorption gas cell -used to track the "true" frequency of the sweeping laser



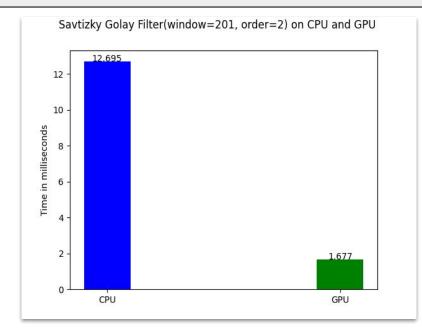
Gas Cell Spectrum

Filtered Gas Cell



Performance Analysis: Savitzky-Golay Filter

```
start=time.time()
savg_cpu = scipy.signal.savgol_filter(FilteredGasCell, 201, 2)
Time_CPU_SG=time.time()-start
FilteredGasCell = cp.array(FilteredGasCell)
start=time.time()
FilteredGasCell = savgol_filter_gpu(FilteredGasCell, window_size=201, order=2)
Time_GPU_SG=time.time()-start
```



Performance Analysis: Savitzky-Golay Filter

	stics:	

Time(%) Total Time (ns) Instances Average Minimum Maximum

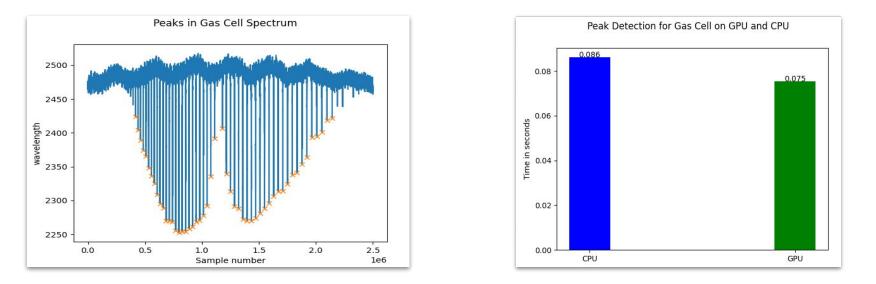
N		

1 131,649.0 131,649 131,649 void generate_seed_pseudo<rng_config<curandStateXORWOW, (curandOrdering)101> >(unsigned long long

12.2	54,272					m_32x32_16x5_nn_align1>(cutlass_80_tensorop_d884gem… t, double*, unsigned long, double*, int)	CUDA Ker	nels usin	na nrot	filinc	r		
6.2 4.0	27,424 18,015	2 9,007.5 8,8				ng_t)0, (twiddle_t)0, (loadstore_modifier_t)2, 10u,…			ig piùi		1		
2.8	12,672	3 4.224.0 3.		768 void larfg kernel fast <double, do<="" td=""><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></double,>									
2.5	11,296	4 2,824.0 1,9		552 cupy copy float64 float64	00010, 02(111, 0	ouble, double, inc, double)	tools - nsy	's protile	nsvs-	111			
2.3	10,112				tensoron d884aem	m 32x32 16x5 nt align1>(cutlass 80 tensorop d884gem		e preme,		0.1			
2.2	9,760	1 9,760.0 9,				ng t)0, (twiddle t)0, (loadstore modifier t)2, 10u,							
2.1	9,280	5 1,856.0 1,8				, double const*, unsigned long, double*, unsigned l							
2.0	8,832			496 void ger kernel <double, :<="" double,="" td=""><td></td><td>ublaccarDagame doubla_doubla.</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></double,>		ublaccarDagame doubla_doubla.							
1.7	7,521	4 1,880.3 1,1		984 void larf minus tau <double>(doub</double>									
1.5	6,560	3 2,186.7 2,0		272 cupy subtract float64 float64 f									
1.4	6,080	2 3,040.0 2,9		136 void larft T 32 <double, 256="">(int</double,>		vthon3.7 +							_
	6,047	2 3,023.5 2,9	43 3,	104 void cuds_gemv_t_kernel <double, 1<="" td=""><td>8,</td><td></td><td></td><td>1 1</td><td></td><td></td><td></td><td></td><td></td></double,>	8,			1 1					
	5,728	1 5,728.0 5,7		728 cupy_powerfloat64_float_float64		cudaFree	cudaHostAlloc		cuMod			Lenery cu	daFree
	5,664	2 2,832.0 2,0	80 3,	584 cupy_addfloat64_float64_float64	4 Profiler ove	rhand		Call to cudaMemcpyAsync					
	5,632	3 1,877.3 1,		952 void gebd2_set_D <double, double=""></double,>	·(i	Ineau		Memory copies					
	5,472	3 1,824.0 1,		952 void gebd2_set_AII <double, double<="" td=""><td></td><td></td><td></td><td>Begins: 9.92745s</td><td></td><td></td><td></td><td></td><td>•</td></double,>				Begins: 9.92745s					•
	5,312	2 2,656.0 2,0		688 void larft_vtv_32 <double, 128="">(in</double,>		1		Ends: 9.9275s (+50.748 µs)		_			•
	5,281	1 5,281.0 5,2		281 void gen_sequenced <curandstatexo< td=""><td></td><td></td><td></td><td>Return value: 0</td><td></td><td></td><td></td><td></td><td></td></curandstatexo<>				Return value: 0					
	5,184	3 1,728.0 1,0		760 void batch_eye_kernel <double, 5,<="" p=""></double,>	2			Correlation ID: 654					
	5,120	2 2,560.0 2,		<pre>592 void larfb_vtc<double, 128="">(int,</double,></pre>								Name 👻	Q
	4,832	2 2,416.0 2,0		<pre>816 cupy_multiplyfloat64_float_float</pre>	at								
1.0	4,609	2 2,304.5 2,0		560 cupy_absolutefloat64_float64	# *	Name			Start	Duration	TID	 Description: 	
0.8	3,680	2 1,840.0 1,8		856 void rot_kernel <double, double,<="" td=""><td></td><td>cudaMalloc</td><td></td><td></td><td>8.81416s</td><td>498.186 ms</td><td>24478</td><td>Call to cudaMalloc</td><td></td></double,>		cudaMalloc			8.81416s	498.186 ms	24478	Call to cudaMalloc	
0.8	3,616	2 1,808.0 1,		824 void gebd2_set_AIIP1 <double, dou<="" td=""><td></td><td></td><td></td><td></td><td>- Contraction Action</td><td></td><td></td><td>Memory allocation</td><td></td></double,>					- Contraction Action			Memory allocation	
0.8	3,616	2 1,808.0 1,		824 void swap_kernel <double, 256="">(in</double,>		cudaMemGetInfo			9.53429s	130.069 µs	24478	Begins: 8.81416s	
0.8	3,584 3,584	2 1,792.0 1, 2 1.792.0 1.		792 void gebd2_set_E <double, double=""> 824 void scal kernel<double. double.<="" p=""></double.></double,>		cuModuleLoadData			9.53444s	543.798 µs	24478	Ends: 9.31234s (+498.186 ms)	
0.7	3,330	2 1,665.0 1,0		697 void larfg set tau zero <double>(i</double>		cupy linspace float float float64			9.5351s	64.401 µs	24478	Return value: 0 Correlation ID: 107	
0.7	3,328	2 1,664.0 1,0		664 void orgqr_step1_kernel <double,< td=""><td></td><td>cuModuleLoadData</td><td></td><td></td><td>9.53657s</td><td>343.578 µs</td><td>24478</td><td>Conclusion ID. 107</td><td></td></double,<>		cuModuleLoadData			9.53657s	343.578 µs	24478	Conclusion ID. 107	
0.7	3,232	1 3,232.0 3,3		<pre>232 cupy_true_dividefloat64_float_</pre>						1		Call stack:	
0.6	2,816	1 2,816.0 2,8		816 void transpose_readWrite_alignment	nt 6	fill			9.53695s	23.648 µs	24478	libcuda.so.470.57.02[13 Frames] [Max depth]![Max depth]	
0.6	2,784	1 2,784.0 2,7		784 cupy_expfloat64_float64	7	cudaMalloc			9.53745s	20.278 µs	24478	fring acheritic as acherit	
0.6	2,720 2,655	1 2,720.0 2,7		720 void gemvNSP_kernel <double, doub<br="">655 cupy_max</double,>	0 0	cuModuleLoadData			9.53875s	531.676 µs	24478		
0.6	2,655	1 2,655.0 2,0		655 cupy_max 655 cupy scan naive	0								
0.6	2,464	1 2,464.0 2,4		464 void ormar cta kernel <double, 4,<="" td=""><td>0 9</td><td>cupy_powerfloat64_float_float64</td><td></td><td></td><td>9.53931s</td><td>24.031 µs</td><td>24478</td><td></td><td></td></double,>	0 9	cupy_powerfloat64_float_float64			9.53931s	24.031 µs	24478		
0.5	2,304	1 2,304.0 2,3		304 cupy_multiplyfloat_float64_float		cudaMalloc			9.53945s	15.558 µs	24478		
0.5	2,272	1 2,272.0 2,1		272 cupy_less_equalfloat64_float64		cuModuleLoadData			9.54064s	410.779 µs	24478		
0.5 0.5	2,240 2,176	1 2,240.0 2,2		240 cupy_multiplycomplex128_comple: 176 cupy_linspacefloat_float_float		cupy negative float64 float64			9.54109s	23.428 µs	24478		
0.5	2,145	1 2,145.0 2,1		145 cupy_reciprocal_float64_float64		cuModuleLoadData			9.54236s	443.049 µs	24478		
0.5	2,145	1 2,145.0 2,1		145 cupy_multiplyfloat64_float64_f	10				PERSONAL PROPERTY AND RECO	1000 C C C C C C C C C C C C C C C C C C	50000		
0.5	2,080	1 2,080.0 2,0		080 cupy_copyint64_float64	14	cupy_expfloat64_float64			9.54284s	24.193 µs	24478		
0.5 0.5	2,017 2,016	1 2,017.0 2,0		017 cupy_negativefloat64_float64 016 copy info kernel(int, int*)	15	cuGetProcAddress			9.54304s	1.701 µs	24478		
0.5	2,010	1 2,010.0 2,0		016 copy_info_kernel(int, int^) 016 cupy copy complex128 complex128	16	cuGetProcAddress			9.54304s	3.213 µs	24478		
0.5	2,015	1 2,015.0 2,0		015 fill	17	cuGetProcAddress			9.54305s	303 ns	24478		
0.4	1,984	1 1,984.0 1,9		984 void shift_householder_vector_by	Ro	cuGetProcAddress			9.54305s	827 ns	24478		
0.4	1,856	1 1,856.0 1,8		<pre>856 void geqrf_copy_R_kernel<double,< pre=""></double,<></pre>	-	LUDELTIULAUUIESS			9.040005	02/115	24470	•	
0.4	1,792	1 1,792.0 1,5	92 1,	792 void cuds_set_matrix_kernel <doub< td=""><td>le</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></doub<>	le								

Some more insights about

Peak Detection for Gas Cell



- There is no function like scipy.signal.find_peaks() in CuPy yet.
- Peak detection for Gas Cell on GPU is developed based on idea that- a peak must be greater (or smaller) than its immediate neighbors, but the performance need to be improved.

def detect_peaks_gpu(x, mph=None, mpd=1, threshold=0)

Some more Signal Processing Routines

A comparison of cupy and numpy implementations on 2.5 million data points sample(time in seconds) :

Routines		Numpy	CuPy	Speed
Interpolation:	np.interp -> cp.interp	0.055173	0.001341	41.4x
Unwrap:	np.unwrap->cp.unwrap	0.143882	0.015522	9.2x
Convolution:	np.convolve->cp.convolve	0.326742	0.014102	23.1x
Angle:	np.angle-> cp.angle	0.165760	0.004315	38.41x
Sort:	np.sort->cp.sort	0.071232	0.002608	27.3x
Absolute:	np.abs->cp.abs	0.005381	0.004910	1.09x

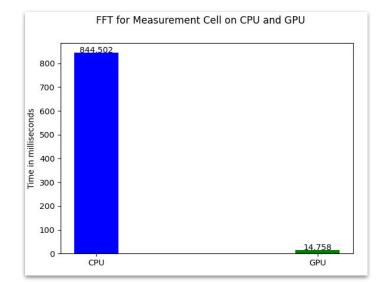
Fast-Fourier Transform in CuPy vs SciPy

cupyx.scipy.fft(x[, n, axis, norm, overwrite_x, plan])

- access advanced routines that cuFFT like get_plan_fft()
- Improve performance and behavior of the FFT routines -

https://docs.cupy.dev/en/stable/user_guide/fft.html

Y = cp.fft.rfft(Meas_Linear, int(len(Meas_Linear)))



	otal Time (ns) In:	stances	Average	Minimum	Maximum				Name		
27.6	7,209,689	3 2	2,403,229.7	1,805,782	3,597,645	void regular fft<625u,	5u, 4u, (pado	ling t)0, (twiddle t)0	, (loadstor	e modifier t)2, (layout t)1, u
24.3	6,349,598	3 2	116.532.7	1.593.272	3.162.159	void regular fft<125u.	5u. 8u. (pado	ling t)1. (twiddle t)0	(loadstor	e modifier t)2, (layout t)1, u
4.4	1,138,906										e_modifier_t)2, (layout_t)1, u
4.3	1,122,426										mplex <double>*, Complex<double< td=""></double<></double>
4.2	1,087,962					cupy angle complex128		ico circy co	and a start of the		inpress address ; compress address
3.8	979,772	2	489.886.0					No. +11 (tud ddl o the	(leadster	e_modifier_t)2, (layout_t)1, u
5.0	919,112		469,660.0	469,720	490,040		, szu, (pauc	ring_i)1, (twiddle_t)o	, (toauston	e_Houtiter_t/2, (tayout_t)1, u
oject 4 🗙 analys											■ Q 1x → A 3 warning:
Timeline View	• 191s • 15	+424ms	+426ms	+428ms	+430r	ns +432ms +4	34ms +	436ms	+438ms	+-440ms	■ Q 1x A <u>3 warning</u> +442ms +444ms
6911 (12)	1915 • 15	+424/05	+420118	+420ms	+4507	ms ++0.22ms ++4	94m8 +	43008	+420ms	+490003	+942705 +994076
CPU (12)			_	_				_			
CUDA HW (NVI	Construction of the second										
* 47.1% Kernel:			ADA		ular_fft ular_fft	regular_ft regular_ft	regular_fft postproce	s cupy_ab	U	c	
* 60.0% regul	iar_m id regular fft≪6			10	laar_in	regular_fft	regular_int				
	d regular fft<1			re	jular fft						
	regular_fft<64						regular_fft				
6.2% void	regular_fft<32										
	eRadixSortDow						_	-			
	rocessC2C_kerr						postproce	5			
	angle_comple:							and the second s			
 3.3% cupy_i 2.8% kernel 	absolute_com							cupy_ab			
36 kernel gr											
+ 52 9% Memor		_							1		
 52.9% Memor Threads (31) 									1		
	ry	_							1	-	
Threads (31)	ry	CuM,	EUX. E. C		1 1	i i i i cuModuleLoadData	1 1	1	(cu), C	- I (u_)	cydattengy
Threads (31)	ry	со М ,]			1 1	culleduret.easData	1 1		Co, C		odaMencpy
 Threads (31) ✓ (11801) py CUDA API Events View 	ry thon3.7 -	COM.				n i i i i culloblecostDita					Name *
 Threads (31) ✓ (11801) py CUDA API Events View ✓ N 	ry thon3.7 -	Contraction of the second seco	.			сиколисоной	Start		on GPU	Context	Name •
Threads (31) ▼ √ (11801) py CUDA API Events View	ry thon3.7 -	CuM.			(c.Motor(conDita	190	: Durati 921s 1.593 922s 1.806	on GPU ms GPU 0		Name • Description: regular fft Beqin: 191.4345
 Threads (31) 	ry thon3.7 - Mo.	COM.			(n aktobiet.aariDita	190 190	.921s 1.593	on GPU ims GPU 0 ims GPU 0	Context Stream 7	Name • Description: • regular fft • 0:05:101.4355 (+1.39 ms) • ord: <<2242, 1.1>>> •
Threads (31)	ry thon2.7	CON.	COS			collosteriostbra	190 190 190	.921s 1.593 .923s 1.806	on GPU ims GPU 0 ims GPU 0 26 µs GPU 0	Context Stream 7 Stream 7	Name Description: regular fft Begin: 191.343 Ends 191.345 Ends 191.
 Threads (31) ✓ (11801) py CUDA API Events View Ψ	ry than3.7	COM				o.MostaccoarDita	190 190 190 190 190	.921s 1.593 .923s 1.806 .925s 489.7 .928s 1.594 .93s 1.806	on GPU ims GPU 0 ims GPU 0 26 µs GPU 0 ims GPU 0 ims GPU 0	Context Stream 7 Stream 7 Stream 7 Stream 7 Stream 7	Name Cescription: regular ff to 10.454 for 10.454 fo
 Threads (31) V (11801) py CUDA API Events View V N 1 2 3 4 5 6 	y then3.7 - Mane regular ft. regular ft. regular ft. regular ft. regular ft. regular ft.	CON		■ 11 · · · · ·		calvaticiastica	190 190 190 190 190 190	921s 1.593 .923s 1.806 .925s 489.7 .928s 1.594 .933s 1.806 .933s 1.806	on GPU ims GPU 0 ims GPU 0 26 µs GPU 0 ims GPU 0 ims GPU 0 46 µs GPU 0	Context Stream 7 Stream 7 Stream 7 Stream 7 Stream 7 Stream 7	Name • Description: • register, #1 • Both 191, 455 (+1, 30 = m) • Biddc <= <<23, 0, 1>>> • Laurch Type, Reput > bytes • Dynamic Started Memory 1.5.34 bytes • Dynamic Started Memory 1.5.34 bytes •
 Threads (31) V (11801) py CUDA API Events View V N 1 2 3 4 5 6 	ry thend 7 - and a me regular fft megalar fft megal	COR.		D)		collegisecontifies	190 190 190 190 190 190 190 190	921s 1.593 923s 1.806 925s 489.7 928s 1.594 933s 1.806 932s 490.0 .4275 3.162	en GPU ms GPU 0 ims GPU 0 26 µs GPU 0 ms GPU 0 ms GPU 0 46 µs GPU 0 ims GPU 0	Context Stream 7 Stream 7 Stream 7 Stream 7 Stream 7 Stream 7 Stream 7	Name Descriptor: Property 11, 14:54 Property 11, 14:54 Property 11, 14:54 Property 11, 14:55 Project 12, 15:55 Project
 Threads (31) V (11801) py CUDA API Events View V N 1 2 3 4 5 6 	ny http://www.internalian.com/ www.internalian.com/ regular_ft: reg	College				colusecandica	190 190 190 190 190 190 190 191 191	921s 1.593 923s 1.806 925s 489.7 928s 1.594 933s 1.806 932s 490.0 .427s 3.162 .435 3.598	en GPU ims GPU 0 ims GPU 0 26 µs GPU 0 ims GPU 0 46 µs GPU 0 ims GPU 0 ims GPU 0	Context Stream 7 Stream 7 Stream 7 Stream 7 Stream 7 Stream 7 Stream 7	Name Descriptor: Tegelar IT Begins
 Threads (31) V (11801) py CUDA API Events View V N 1 2 3 4 5 6 	ry thend 7 - and a me regular fft megalar fft megal	Colt				n n n n n n n n n n n n n n n n n n n	190 190 190 190 190 190 190 191 191	921s 1.593 923s 1.806 925s 489.7 928s 1.594 933s 1.806 932s 490.0 .4275 3.162	en GPU ims GPU 0 ims GPU 0 26 µs GPU 0 ims GPU 0 46 µs GPU 0 ims GPU 0 ims GPU 0	Context Stream 7 Stream 7 Stream 7 Stream 7 Stream 7 Stream 7 Stream 7	Name Descriptor: Perform Perf
 Threads (31) V (11801) py CUDA API Events View V N 1 2 3 4 5 6 	ny http://www.internalian.com/ www.internalian.com/ regular_ft: reg	CON.				n n n n n n n n n n n n n n n n n n n	190 190 190 190 190 190 190 191 191	921s 1.593 923s 1.806 925s 489.7 928s 1.594 933s 1.806 932s 490.0 .427s 3.162 .435 3.598	en GPU ims GPU 0 ims GPU 0 26 µs GPU 0 ims GPU 0 46 µs GPU 0 ims GPU 0 ims GPU 0	Context Stream 7 Stream 7 Stream 7 Stream 7 Stream 7 Stream 7 Stream 7	Nume Descriptor: "equil: n" Bigs: 10, 125 (1, 130 m) grid: excel: 1, 130 m0
 Threads (31) ✓ (11801) py CUDA API Events View 	ny http://www.internalian.com/ www.internalian.com/ regular_ft: reg	CONC. 1				r r colociedada	190 190 190 190 190 190 190 191 191	921s 1.593 923s 1.806 925s 489.7 928s 1.594 933s 1.806 932s 490.0 .427s 3.162 .435 3.598	en GPU ims GPU 0 ims GPU 0 26 µs GPU 0 ims GPU 0 46 µs GPU 0 ims GPU 0 ims GPU 0	Context Stream 7 Stream 7 Stream 7 Stream 7 Stream 7 Stream 7 Stream 7	Name Descriptor: Pagelar # Pagelar
 Threads (31) V (11801) py CUDA API Events View W V N 1 2 3 4 5 6 	ny http://www.internalian.com/ www.internalian.com/ regular_ft: reg	CONC				r textester	190 190 190 190 190 190 190 191 191	921s 1.593 923s 1.806 925s 489.7 928s 1.594 933s 1.806 932s 490.0 .427s 3.162 .435 3.598	en GPU ims GPU 0 ims GPU 0 26 µs GPU 0 ims GPU 0 46 µs GPU 0 ims GPU 0 ims GPU 0	Context Stream 7 Stream 7 Stream 7 Stream 7 Stream 7 Stream 7 Stream 7	Nume Descriptor: "equil: n" Bigs: 10, 125 (1, 130 m) grid: excel: 1, 130 m0
 Threads (31) V (11801) py CUDA API Events View W V N 1 2 3 4 5 6 	ny http://www.internalian.com/ www.internalian.com/ regular_ft: reg					n n n n n n n n n n n n n n n n n n n	190 190 190 190 190 190 190 191 191	921s 1.593 923s 1.806 925s 489.7 928s 1.594 933s 1.806 932s 490.0 .427s 3.162 .435 3.598	en GPU ims GPU 0 ims GPU 0 26 µs GPU 0 ims GPU 0 46 µs GPU 0 ims GPU 0 ims GPU 0	Context Stream 7 Stream 7 Stream 7 Stream 7 Stream 7 Stream 7 Stream 7	Nume Description: "equil: m1 Segment (11, 125 er) Bergin: 11, 125 er)



Frequency Scanning Interferometry System

- Signal Processing in FSI
- CuPy and Signal Processing
 - Butterworth Filter
 - Hilbert Transform
 - Savitzky-Golay Filter



- CuPy a great library to start and test processing on GPU and expand to Signal Processing
- More performance tests and analysis to do with multiple channels and ultimately to improve performance
- Move to more CuPy based custom Kernels
- Upstreaming developments to CuPy repository

- CuPy a great library to start and test processing on GPU and expand to Signal Processing
- More performance tests and analysis to do with multiple channels and ultimately to improve performance
- Move to more CuPy based custom Kernels
- Upstreaming developments to CuPy repository

Now this is not the end. It is not even the beginning of the end. But it is, perhaps, the end of the beginning - Winston Churchill

