FERMILAB-TM-2486-AD

A Bootstrap Algorithm for Temporal Signal Reconstruction in the Presence of Noise from its Fractional Fourier Transformed Intensity Spectra

Cheng-Yang Tan

Accelerator Division/Tevatron

ABSTRACT: A bootstrap algorithm for reconstructing the temporal signal from four of its fractional Fourier intensity spectra in the presence of noise is described. An optical arrangement is proposed which realises the bootstrap method for the measurement of ultrashort laser pulses.

Work supported by Fermi Research Alliance, LLC under Contract No. DE-AC02-07CH11359 with the US Department of Energy.

INTRODUCTION

The measurement of short laser pulses which are less than 1 ps is an ongoing challenge in optical physics. One reason is that no oscilloscope exists today which can directly measure the time structure of these pulses and so it becomes necessary to invent other techniques which indirectly provide the necessary information for temporal pulse reconstruction. One method called FROG (frequency resolved optical gating) has been in use since 1991[1] and is one of the popular methods for recovering these types of short pulses. The idea behind FROG is the use of multiple time-correlated pulse measurements in the frequency domain for the reconstruction. Multiple data sets are required because only intensity information is recorded and not phase, and thus by collecting multiple data sets, there is enough redundant measurements to yield the original time structure, but not necessarily uniquely (or even up to an arbitrary constant phase offset).

The objective of this paper is to describe another method which is simpler than FROG. Instead of collecting many auto-correlated data sets, only two spectral intensity measurements of the temporal signal are needed *in the absence of noise*. The first can be from the intensity components of its usual Fourier transform and the second from its FrFT (fractional Fourier transform). In the presence of noise, a minimum of four measurements are required with the same FrFT order but with two different apertures. Armed with these two or four measurements, a unique solution up to a constant phase offset can be constructed.

The next sections will describe how

(*i*) the input temporal signal can be reconstructed from two different FrFT order measurements in the absence of noise. See *Finding* $s(t)$ *from* $I_\alpha(f)$ *and* $I_\gamma(f)$ *with a bootstrap algorithm*.

(*ii*) in the presence of noise, at least two sets of apertures and two different order

FrFTs, i.e. four measurements makes the algorithm more immune to noise. See *Enhanced bootstrap algorithm*.

(*iii*) an optical setup for implementing the algorithm. See *Method for Measuring Ultrashort Laser Pulses*.

(*iv*) the bootstrap algorithm can reconstruct the temporal signal from the optical setup. See *Numerical Demonstration I* and *II* and *Appendix II: Optics Analysis*.

We define the fractional Fourier transform (FrFT)[2] of a function $s(u)$ to be

$$S_\alpha(v) = \int_{-\infty}^{\infty} du\ K(\alpha, u, v) s(u) \tag{1}$$

where the kernel K is given by

$$\left.\begin{aligned} K(\alpha, u, v) &= \frac{e^{i\frac{1}{2}\alpha}}{\sqrt{i \sin\alpha}} \exp\left[i\pi \frac{(u^2+v^2)\cos\alpha - 2uv}{\sin\alpha}\right] \\ &\equiv A_\alpha \exp\left[i\pi \frac{(u^2+v^2)\cos\alpha - 2uv}{\sin\alpha}\right] \end{aligned}\right\} \tag{2}$$

For the case when $0 < |\alpha| < \pi$, A_α can be written as

$$A_\alpha = \frac{\exp\left[-i\left(\frac{\pi\ \mathrm{sgn}(\alpha)}{4} - \frac{\alpha}{2}\right)\right]}{|\sin\alpha|^{\frac{1}{2}}} \qquad \text{if } 0 < |\alpha| < \pi \tag{3}$$

And in the cases where $\alpha = \pm\pi/2$, (2) reduces to the ordinary Fourier transform. In this paper, we define the Fourier transform to be

$$S_{\frac{1}{2}\pi}(v) = \int_{-\infty}^{\infty} du\ s(u) e^{-i2\pi uv} \tag{4}$$

i.e. when $\alpha = \pi/2$.

Important: the transform pair variables u and v are *dimensionless*. Since we are going to be working in the time and frequency domains, we will identify u as the dimensionless time variable and v as the dimensionless frequency variable. To keep the "bean counting" easier, we will map $u \to t$ and $v \to f$ and call the dimensionless variables t time and f frequency for the rest of this paper.[a]

An N-point α-order DFrFT of $s(t)$ is easily created from (1) and is

$$S_\alpha(m\Delta f) = A_\alpha \Delta t \sum_{k=-N/2}^{N/2-1} s(k\Delta t) \exp\left[i\pi\left[(m\Delta f)^2 + (k\Delta t)^2\right]\cot\alpha - i\frac{2\pi km}{N}\right] \tag{5}$$

[a] Ozaktas *et al* spent section 7.2 in their book[2] discussing the relationship between dimensionless and dimensionfull variables in FrFT.

where Δt is the sampling time, $\Delta f \Delta t = \sin\alpha/N$, and we have assumed that $N = 2q$ where $q \in \mathbb{N}$.

An important FrFT property which we will use to recover the input signal $s(t)$ in the *Method* section is the auto-correlation property[3] (c.f. Wiener-Khinchin Theorem and auto-correlation for ordinary Fourier transforms). The auto-correlation for an α-order FrFT is defined to be

$$(s \star_\alpha s)(\tau) = e^{i\pi\tau^2 \cos\alpha \sin\alpha} \int_{-\infty}^{\infty} dt\ s^*(t) s(t + \tau\cos\alpha) e^{-i2\pi t\tau \sin\alpha} \tag{6}$$

A quick check will show that (6) is the auto-correlation function for the ordinary Fourier Transform when $\alpha = 0$.

It is easy to show that $(s \star_{\alpha-\pi/2} s)(\tau)$ is related to $|S_\alpha(f)|^2$ as follows

$$\left.\begin{aligned}(s \star_{\alpha-\pi/2} s)(\tau) &= e^{-i\pi\tau^2 \cot\alpha} \int_{-\infty}^{\infty} dt\ s^*(t) s(t+\tau) e^{i2\pi t\tau \cot\alpha} \\ &= \int_{-\infty}^{\infty} df\ |S_\alpha(f)|^2 e^{i2\pi f t}\end{aligned}\right\} \tag{7}$$

We can discretise (7) using an N-point α-order DFrFT to get

$$\left.\begin{aligned}&\sum_{\ell=-N/2}^{N/2-1} s^*(\ell\Delta t) s[(\ell+k)\Delta t] e^{i2\pi k\ell(\Delta t)^2 \cot\alpha} \\ &= \frac{\sin\alpha}{N\Delta t^2} e^{-i\pi(k\Delta t)^2 \cot\alpha} \sum_{m=-N/2}^{N/2-1} |S_\alpha(m\Delta f)|^2 e^{i2\pi \frac{km}{N}}\end{aligned}\right\} \tag{8}$$

where $\Delta f \Delta t = \sin\alpha/N$ and $\tau = k\Delta t$ for $0 \le k \le N-1$. We remind ourselves that periodic boundary conditions have been applied to the lhs of (8) because clearly $(\ell + k) > N/2 - 1$ and $(\ell + k) < -N/2$ for some k. This condition will be used in the *Method* section.

Method

We will assume that our setup measures the FrFT spectrum from an input signal $s(t)$. The measured intensity $I_\alpha(f)$ on the detector from an α-order FrFT is

$$|S_\alpha(f)|^2 = \left| \int_{-\infty}^{\infty} dt\ s(t) K(\alpha, t, f) \right|^2 \equiv I_\alpha(f) \tag{9}$$

The goal, of course, is to extract $s(t)$ from two sets of measurements $I_\alpha(f)$ and $I_\gamma(f)$. Note: In the next section we will cavalierly assume that we can, in fact, produce I_α in the form that is compatible with the bootstrap algorithm and with the required accuracy. However, in real life, there is noise in the measurement and a more complicated extraction process which uses the bootstrap method as its basis must be used. We will discuss this algorithm in the subsection *Enhanced Bootstrap Algorithm*.

Finding $s(t)$ from $I_\alpha(f)$ and $I_\gamma(f)$ with a bootstrap algorithm

In general, we need two sets of intensities from two different order FrFTs for extracting $s(t)$. We will describe a *bootstrap algorithm*[b] for obtaining $s(t)$ here. Note: Much of our work has been inspired by the work of Cong *et al.*[5]

Step 1 We divide $s(t)$ into N samples at Δt intervals and then pad with zeros from $-(N_m/2)\Delta t$ to $(-N/2-1)\Delta t$ and from $(N/2)\Delta t$ to $(N_m/2-1)\Delta t$ to get $N_m(\geq 2N)$ points, i.e.

$$\{s_\ell\} = \left\{ \begin{array}{ccccccccc} 0_{-N_m/2}, & 0_{-N_m/2+1}, & \ldots, & \ldots, & \ldots, & 0_{-N/2-1}, & & & \\ s_{-N/2}, & s_{-N/2+1}, & \ldots, & s_{-1}, & s_0, & s_1, & \ldots, & s_{N/2-1}, & \\ 0_{N/2}, & 0_{N/2+1}, & \ldots, & \ldots, & \ldots, & 0_{N_m/2-2}, & 0_{N_m/2-1}, & & \end{array} \right\} \tag{10}$$

[b] At least two algorithms exist, the Gerchberg-Saxton[4] algorithm and a recursive algorithm published by Cong *et al*[5] Unfortunately, we do not understand how the recursive algorithm described by them can work and so have come up with our own bootstrap algorithm.

where $\ell = -N_m/2, -N_m/2+1, \ldots, N_m/2-1$ and $s_\ell = s(\ell\Delta t)$. Note: Here is one major difference between our analysis and Cong *et al*'s: we have extended the analysis for $N_m \geq 2N$.

Therefore, the N_m α-order DFrFT of $\{s_\ell\}$ is

$$S_\alpha(n\Delta f_\alpha) = A_\alpha \Delta t \sum_{\ell=-N_m/2}^{N_m/2-1} s_\ell e^{i\pi[(n\Delta f_\alpha)^2+(\ell\Delta t)^2]\cot\alpha - i2\pi\ell n/N_m} \tag{11}$$

where $\Delta f_\alpha \Delta t = \sin\alpha/N_m$ and $-N_m/2 \leq n \leq N_m/2-1$.

We also perform an N_m γ-order DFrFT on $\{s_\ell\}$, and the solution is

$$S_\gamma(n\Delta f_\gamma) = A_\gamma \Delta t \sum_{\ell=-N_m/2}^{N_m/2-1} s_k e^{i\pi[(n\Delta f_\gamma)^2+(\ell\Delta t)^2]\cot\gamma - i2\pi\ell n/N_m} \tag{12}$$

where $\Delta f_\gamma \Delta t = \sin\gamma/N_m$.

The constraint on α and γ will be determined by the $\{s_\ell\}$ bootstrap process in Step 3 below.

Step 2 We apply the correlation property for the α-order N_m DFrFT to $\{s_\ell\}$ at $\tau = k\Delta t$. Using (8), we get

$$\left.\begin{aligned} &\sum_{\ell=-N/2}^{N/2-1} s_\ell^* s_{\ell+k} e^{i2\pi k\ell(\Delta t)^2\cot\alpha} \\ &= \frac{\sin\alpha}{N_m\Delta t^2} e^{-i\pi(k\Delta t)^2\cot\alpha} \sum_{m=-N_m/2}^{N_m/2-1} I_\alpha(m\Delta f_\alpha) e^{i2\pi\frac{km}{N_m}} \\ &\equiv C_\alpha(k) \end{aligned}\right\} \tag{13}$$

where $I_\alpha(m\Delta f_\alpha) = |S_\alpha(m\Delta f_\alpha)|^2$. (Note that I_α is independent of τ) The range of the summation of $s_\ell^* s_{\ell+k}$ starts and stops with the non-zero values of $\{s_\ell\}$. (Clearly the range can be made tighter, but this is sufficient for our purposes). And $I_\alpha(f)$ is just the intensity which we measure of the α-order DFrFT of $\{s_\ell\}$. This means that $C_\alpha(k)$ is completely known for each k given I_α.

Similarly, the correlation property for the γ-order $2N$ DFrFT of $\{s_\ell\}$ at $\tau = k\Delta t$ is

$$\left.\begin{aligned}&\sum_{\ell=-N/2}^{N/2-1} s_\ell^* s_{\ell+k} e^{i2\pi k\ell(\Delta t)^2\cot\gamma} \\ &= \frac{\sin\gamma}{N_m\Delta t^2} e^{-i\pi(k\Delta t)^2\cot\gamma} \sum_{m=-N_m/2}^{N_m/2-1} I_\gamma(m\Delta f_\gamma) e^{i2\pi\frac{km}{N_m}} \\ &\equiv C_\gamma(k)\end{aligned}\right\} \tag{14}$$

where $I_\gamma(m\Delta f_\gamma) = |S_\gamma(m\Delta f_\gamma)|^2$ is the intensity which we measure of the γ-order DFrFT of $\{s_\ell\}$. And again, $C_\gamma(k)$ is completely known for each k given I_γ.

Step 3 We solve for the terms $s^*_{-N/2}s_{N/2-m}$ and $s^*_{-N/2+m-1}s_{N/2-1}$ which comes from knowing $C_\alpha(k)$ and $C_\gamma(k)$.

When we substitute $k = N-1$ is into (13) and (14), we find that

$$s^*_{-N/2}s_{N/2-1} = C_\alpha(N-1)e^{i\pi N(N-1)(\Delta t)^2\cot\alpha} = C_\gamma(N-1)e^{i\pi N(N-1)(\Delta t)^2\cot\gamma} \tag{15}$$

We notice that the rhs is completely independent of $\{s_\ell\}$.

When we substitute $k = N-2$ into (13) and (14), we find that

$$\left.\begin{aligned}&\begin{pmatrix} e^{i2\pi(-N/2)(N-2)(\Delta t)^2\cot\alpha} & e^{i2\pi(-N/2+1)(N-2)(\Delta t)^2\cot\alpha} \\ e^{i2\pi(-N/2)(N-2)(\Delta t)^2\cot\gamma} & e^{i2\pi(-N/2+1)(N-2)(\Delta t)^2\cot\gamma}\end{pmatrix}\begin{pmatrix} s^*_{-N/2}s_{N/2-2} \\ s^*_{-N/2+1}s_{N/2-1}\end{pmatrix} \\ &\equiv \begin{pmatrix}\mathcal{U}_\alpha(2) & \mathcal{V}_\alpha(2) \\ \mathcal{U}_\gamma(2) & \mathcal{V}_\gamma(2)\end{pmatrix}\begin{pmatrix} s^*_{-N/2}s_{N/2-2} \\ s^*_{-N/2+1}s_{N/2-1}\end{pmatrix} = \begin{pmatrix} C_\alpha(N-2) \\ C_\gamma(N-2)\end{pmatrix}\end{aligned}\right\} \tag{16}$$

which can be solved for $s^*_{-N/2}s_{N/2-2}$ and $s^*_{-N/2+1}s_{N/2-1}$ by inverting the 2×2 matrix on the lhs. When we do this, we have

$$\begin{pmatrix} s^*_{-N/2}s_{N/2-2} \\ s^*_{-N/2+1}s_{N/2-1}\end{pmatrix} = \frac{1}{\mathcal{D}(2)}\begin{pmatrix}\mathcal{V}_\gamma(2) & -\mathcal{V}_\alpha(2) \\ -\mathcal{U}_\gamma(2) & \mathcal{U}_\alpha(2)\end{pmatrix}\begin{pmatrix} C_\alpha(N-2) \\ C_\gamma(N-2)\end{pmatrix} \tag{17}$$

where $\mathcal{D}(2) = \det\begin{pmatrix}\mathcal{U}_\alpha(2) & \mathcal{V}_\alpha(2) \\ \mathcal{U}_\gamma(2) & \mathcal{V}_\gamma(2)\end{pmatrix}$. Again, like for the $k = N-1$ case, the rhs is independent of $\{s_\ell\}$. However, this is not true for $k > N-2$ because in general, for

$k = N - m$, we have

$$\begin{pmatrix} s^*_{-N/2} s_{N/2-m} \\ s^*_{-N/2+m-1} s_{N/2-1} \end{pmatrix} = \frac{1}{\mathcal{D}(m)} \begin{pmatrix} \mathcal{V}_\gamma(m) & -\mathcal{V}_\alpha(m) \\ -\mathcal{U}_\gamma(m) & \mathcal{U}_\alpha(m) \end{pmatrix} \begin{pmatrix} C_\alpha(N-m) - \Sigma_\alpha(m) \\ C_\gamma(N-m) - \Sigma_\gamma(m) \end{pmatrix} \tag{18}$$

where

$$\left.\begin{aligned} \mathcal{U}_\theta(m) &= e^{i2\pi(-N/2)(N-m)(\Delta t)^2 \cot\theta} \\ \mathcal{V}_\theta(m) &= e^{i2\pi(-N/2+m-1)(N-m)(\Delta t)^2 \cot\theta} \\ \Sigma_\theta(m) &= \sum_{j=1}^{m-2} s^*_{-N/2+j} s_{N/2-m+j} e^{i2\pi(N-m)(-N/2+j)(\Delta t)^2 \cot\theta} \\ \mathcal{D}(m) &= \det \begin{pmatrix} \mathcal{U}_\alpha(m) & \mathcal{V}_\alpha(m) \\ \mathcal{U}_\gamma(m) & \mathcal{V}_\gamma(m) \end{pmatrix} \\ &= e^{i2\pi(-N/2)(N-m)(\Delta t)^2(\cot\alpha+\cot\gamma)} e^{i2\pi(N-m)(m-1)(\Delta t)^2 \cot\gamma} \times \\ &\quad \left(1 - e^{i2\pi(N-m)(m-1)(\Delta t)^2(\cot\alpha - \cot\gamma)}\right) \end{aligned}\right\} \tag{19}$$

and the rhs is clearly dependent on $\{s_\ell\}$. We notice that $\mathcal{D} \neq 0$ as long as $m \neq N$ and $m \neq 1$ and so the general solution is always applicable for $2 \leq m \leq N-1$ (or $1 \leq k \leq N-2$).

For the algorithm to work, we must ensure that $\mathcal{D}(m) \neq 0$ for $2 \leq m \leq N/2+1$ (or $N/2-1 \leq k \leq N-2$). This is the only constraint on α and γ.

Step 4 We start the bootstrap method by solving for $s_{N/2-1}$ in terms of $s_{N/2}$ with (15)

$$\left.\begin{aligned} s_{N/2-1} &= C_\alpha(N-1) e^{i\pi N(N-1)(\Delta t)^2 \cot\alpha} / s^*_{-N/2} \\ &\equiv c_\alpha / s^*_{-N/2} \end{aligned}\right\} \tag{20}$$

From (17), we can solve for $s_{N/2-2}$ and $s_{-N/2+1}$ in terms of $s_{-N/2}$

$$\left.\begin{aligned} s_{N/2-2} &= \frac{1}{s^*_{-N/2}\mathcal{D}(2)} \begin{pmatrix} \mathcal{V}_\gamma(2) \\ -\mathcal{V}_\alpha(2) \end{pmatrix}^T \begin{pmatrix} C_\alpha(N-2) \\ C_\gamma(N-2) \end{pmatrix} \\ s_{-N/2+1} &= \frac{s_{-N/2}}{c^*_\alpha \mathcal{D}^*(2)} \begin{pmatrix} -\mathcal{U}_\gamma(2) \\ \mathcal{U}_\alpha(2) \end{pmatrix}^\dagger \begin{pmatrix} C_\alpha(N-2) \\ C_\gamma(N-2) \end{pmatrix}^* \end{aligned}\right\} \tag{21}$$

In general, we have

$$\left.\begin{aligned} s_{N/2-m} &= \frac{1}{s^*_{-N/2}\mathcal{D}(m)} \begin{pmatrix} \mathcal{V}_\gamma(m) \\ -\mathcal{V}_\alpha(m) \end{pmatrix}^T \begin{pmatrix} C_\alpha(N-m) - \Sigma_\alpha(m) \\ C_\gamma(N-m) - \Sigma_\gamma(m) \end{pmatrix} \\ &\equiv \frac{\mathcal{F}_{\alpha,\gamma}(m)}{s^*_{-N/2}} \\ s_{-N/2+m-1} &= \frac{s_{-N/2}}{c^*_\alpha \mathcal{D}^*(m)} \begin{pmatrix} -\mathcal{U}_\gamma(m) \\ \mathcal{U}_\alpha(m) \end{pmatrix}^\dagger \begin{pmatrix} C_\alpha(N-m) - \Sigma_\alpha(m) \\ C_\gamma(N-m) - \Sigma_\gamma(m) \end{pmatrix}^* \\ &\equiv s_{-N/2}\mathcal{G}_{\alpha,\gamma}(m) \end{aligned}\right\} \tag{22}$$

and

$$\left.\begin{aligned} \Sigma_\theta(m) = \sum_{j=1}^{m-2} &\frac{1}{c_\alpha \mathcal{D}(j+1)} \begin{pmatrix} -\mathcal{U}_\gamma(j+1) \\ \mathcal{U}_\alpha(j+1) \end{pmatrix}^T \begin{pmatrix} C_\alpha(N-j-1) - \Sigma_\alpha(j+1) \\ C_\gamma(N-j-1) - \Sigma_\gamma(j+1) \end{pmatrix} \times \\ &\frac{1}{\mathcal{D}(m-j)} \begin{pmatrix} \mathcal{V}_\gamma(m-j) \\ -\mathcal{V}_\alpha(m-j) \end{pmatrix}^T \begin{pmatrix} C_\alpha(N-m+j) - \Sigma_\alpha(m-j) \\ C_\gamma(N-m+j) - \Sigma_\gamma(m-j) \end{pmatrix} \times \\ &e^{i2\pi(N-m)(-N/2+j)(\Delta t)^2 \cot\theta} \end{aligned}\right\} \tag{23}$$

We observe that $\Sigma_\theta(m)$ is independent of $s_{-N/2}$ which means that $\mathcal{F}_{\alpha,\gamma}(m)$ and $\mathcal{G}_{\alpha,\gamma}(m)$ are both independent of $s_{-N/2}$.

Step 5 We solve for $s_{-N/2}$. From (22), we see that for both $s_{N/2-m}$ and $s_{-N/2+m-1}$, $1 \leq m \leq N/2$. And from (13), we have for $k = 0$

$$\sum_{\ell=-N/2}^{N/2-1} |s_\ell|^2 = C_\alpha(0) \tag{24}$$

Thus, by using (22), we have

$$|s_{-N/2}|^2 \sum_{m=1}^{N/2} |\mathcal{G}_{\alpha,\gamma}(m)|^2 + \frac{1}{|s_{-N/2}|^2} \sum_{m=1}^{N/2} |\mathcal{F}_{\alpha,\gamma}(m)|^2 = C_\alpha(0) \tag{25}$$

which we can easily solve for $|s_{-N/2}|$. We can immediately dismiss the negative solutions and only keep the two positive solutions. In principle, both solutions are possible values for $|s_{-N/2}|$. The final selection of $|s_{-N/2}|$ comes after we make the following assumption:

All the phases are measured w.r.t. $s_{-N/2}$. This will allow us to make $s_{-N/2} \in \mathbb{R}$ and $s_{-N/2} > 0$. Once we do this, we can make the identification that $|s_{-N/2}| = s_{-N/2}$. With

this identification, we can select between the two possible $s_{-N/2}$ by demanding that

$$s_0 = s_{-N/2}\mathcal{G}_{\alpha,\gamma}(N/2+1) = \frac{1}{s_{-N/2}}\mathcal{F}_{\alpha,\gamma}(N/2) \tag{26}$$

Therefore, we can bootstrap this solution to obtain the solution for every term in the sequence $\{s_\ell : \ell = -N/2+1, -N/2, \ldots, N/2-1\}$ by using all the equations in Step 4. Q.E.D.

Note: We observe that since every s_ℓ is either multiplied by s_{-N} or $1/s_{-N}$ in the equations of Step 4, we can set $s_{-N} = 1$ first and solve for $\{s_\ell\}$ with this assumption. After we have obtained $s_{-N/2}$ in Step 5, we can then re-calculate $\{s_\ell\}$ with this value. The *Mathematica* input files shown in the appendices and used in all the demonstrations apply this observation.

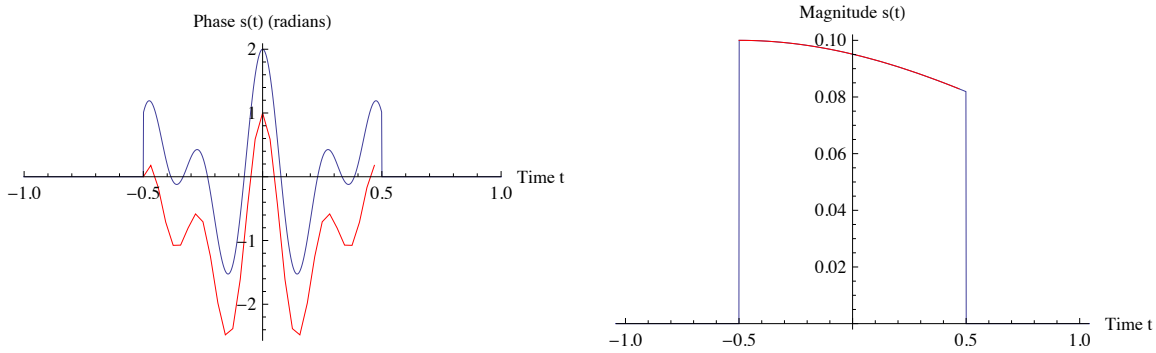


Figure 1 The blue traces are $s(t)$ plotted out in phase and magnitude. The red traces are the results of the bootstrap algorithm applied to the FrFT intensity spectra of the example.

Demonstration of the Bootstrap Algorithm

We can demonstrate how the bootstrap algorithm works with an example.[c] The input

[c] The *Mathematica* input file which was used in this demonstration is attached in *Appendix I*.

signal $s(t)$ is slightly modified from Cong *et al*[5]

$$s(t) = 0.1 \times e^{-0.2(t+0.5)^2} e^{i[\sin 5\pi(t+0.5)+\cos 8\pi(t+0.5)]} \tag{27}$$

For the bootstrap algorithm, we have padded the input with zeroes from -2.0 to -0.5 and 0.5 to 2.0 and set the number of samples $N = 32$ in the $s(t)$ region and $N_m = 256$ from $-2.0 \leq t \leq 2.0$. We have chosen $\alpha = \pi/2$ for the first data set and $\gamma = \pi/4$ for the second data set. We will only show the final results in Figure 1 because the demonstration of the algorithm should be sufficiently clear from the *Mathematica* input file. It is obvious that the bootstrap algorithm faithfully reproduces the magnitude of $s(t)$ while the phase is offset as expected.

Noise

The astute reader will notice that the the bootstrap algorithm solves for $s_{-N/2}$ and then marches towards s_0 by bootstrapping. There are two problems with the algorithm when applied to practical problems

(*i*) The temporal size of the signal is unknown and so N is unknown. For the algorithm to work $s_{-N/2}$ must not be zero.

(*ii*) $|s_{-N/2}|$ is at the tail end of the input pulse, it is usually much smaller than $|s_0|$ and so if there is any noise in the measurement of $I_{\alpha,\gamma}$, the error in $s_{-N/2}$ will propagate through the entire solution for $\{s_\ell\}$. In fact, from our computer simulations, we have discovered that the maximum relative error in the measured $I_{\alpha,\gamma}$ w.r.t. true $I_{\alpha,\gamma}$ must be $< 0.1 \times 10^{-5}$ for the solution to be valid! This is, of course, impossible to achieve in real life. Figure 2 shows the problem.

The first problem highlighted above can only be solved with a change or enhancement of the bootstrap algorithm. We discuss the solution to (*ii*) below. For the noise problem,

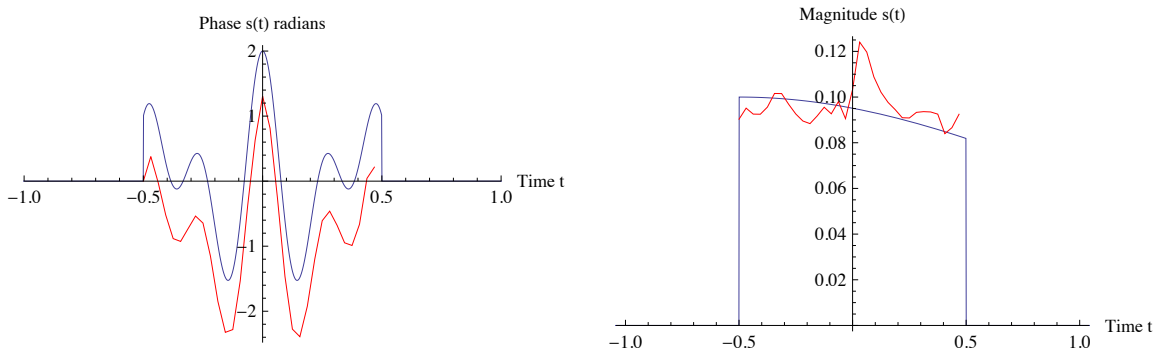


Figure 2 With noise in $I_{\alpha,\gamma}$, the bootstrap algorithm does not work very well. In this example, the maximum relative range of the uniform noise at each point is ± 0.01, $N_m = 256$, and $N = 32$.

the solutions are obvious:

(i) We can average $I_{\alpha,\gamma}$ sufficiently so that we decrease the noise.

(ii) We can enhance the bootstrap algorithm as follows: We take a smaller chunk of the input pulse $s(t)$ by masking out the tails so that the bootstrap algorithm works with $|s_{-N/2}| \neq 0$ and, we hope, is comparable in size to $|s_0|$. We can then increase the chunk and bootstrap these solutions to solve for s_ℓ outside the first chunk. We can continue increasing the chunk size until we obtain the solution for the entire pulse. This forms the basis of our *enhanced* bootstrap algorithm which we will describe in the next section.

Enhanced Bootstrap Algorithm

We can extend the bootstrap algorithm like we have previously described by first taking a smaller chunk of $s(t)$. Suppose $N_1 < N$ is the first chunk that is used to produce the spectra $I_{1,\alpha}$ and $I_{2,\gamma}$. We will assume that by taking this smaller chunk $|s_{-N_1/2}|$ is comparable in size to $|s_0|$ and so we hope will have greater immunity to noise in $I_{1,\alpha}$ and

$I_{1,\gamma}$. We can then use the bootstrap algorithm to calculate $\{s_{-N_1/2},\ldots,s_0,\ldots,s_{N_1/2-1}\} \equiv \{\sigma_{-N_1/2},\ldots,\sigma_0,\ldots,\sigma_{N_1/2-1}\}$ where we will use σ_ℓ to signify that the solutions have already been found.

For our paper, let us choose the size of the next chunk to be $N_2 = 2N_1$.[d] Using (13), we can select only the equations which does not involve $(s_i^* \times s_j)$ we can write down the following complex matrix equation

$$\begin{pmatrix} \boldsymbol{S}_\alpha \\ \boldsymbol{S}_\gamma \end{pmatrix} \boldsymbol{s} = \begin{pmatrix} \boldsymbol{C}_\alpha \\ \boldsymbol{C}_\gamma \end{pmatrix} \tag{28}$$

The entries of the $\boldsymbol{S}_\theta$ matrix are

$$\boldsymbol{S}_\theta = \begin{pmatrix} \bar{\sigma}_{-\frac{N_1}{2},N_1} & \bar{\sigma}_{-\frac{N_1}{2}+1,N_1} & \cdots & \bar{\sigma}_{-1,N_1} & \sigma_{0,N_1} & \sigma_{1,N_1} & \cdots & \sigma_{\frac{N_1}{2}-2,N_1} & \sigma_{\frac{N_1}{2}-1,N_1} \\ \bar{\sigma}_{-\frac{N_1}{2}+1,N_1-1} & \bar{\sigma}_{-\frac{N_1}{2}+2,N_1-1} & \cdots & \bar{\sigma}_{0,N_1-1} & \sigma_{-1,N_1-1} & \sigma_{0,N_1-1} & \cdots & \sigma_{\frac{N_1}{2}-3,N_1-1} & \sigma_{\frac{N_1}{2}-2,N_1-1} \\ \bar{\sigma}_{-\frac{N_1}{2}+2,N_1-2} & \bar{\sigma}_{-\frac{N_1}{2}+3,N_1-2} & \cdots & \bar{\sigma}_{1,N_1-2} & \sigma_{-2,N_1-2} & \sigma_{-1,N_1-2} & \cdots & \sigma_{\frac{N_1}{2}-4,N_1-2} & \sigma_{\frac{N_1}{2}-3,N_1-2} \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ \bar{\sigma}_{0,\frac{N_1}{2}} & \bar{\sigma}_{1,\frac{N_1}{2}} & \cdots & \bar{\sigma}_{\frac{N_1}{2}-1,\frac{N_1}{2}} & \sigma_{-\frac{N_1}{2},\frac{N_1}{2}} & \sigma_{-\frac{N_1}{2}+1,\frac{N_1}{2}} & \cdots & \sigma_{-2,\frac{N_1}{2}} & \sigma_{-1,\frac{N_1}{2}} \end{pmatrix} \tag{29}$$

where $\bar{\sigma}_{k,\ell} \to \bar{\sigma}_{k,\ell}(\theta) = \sigma_k^* e^{+i2\pi k\ell(\Delta t)^2\cot\theta}$ and $\sigma_{k,\ell} \to \sigma_{k,\ell}(\theta) = \sigma_k e^{+i2\pi(k-\ell)\ell(\Delta t)^2\cot\theta}$. The size of the $\boldsymbol{S}_\theta$ matrix is $(N_1/2+1)\times N_1$.

The $\boldsymbol{s}$ vector contains $\left\{s^*_{-\frac{N_2}{2}}, s^*_{-\frac{N_2}{2}+1},\ldots,s^*_{-\frac{N_1}{2}-1}\right\} \cup \left\{s_{\frac{N_1}{2}}, s_{\frac{N_1}{2}+1},\ldots,s_{\frac{N_2}{2}-2}, s_{\frac{N_2}{2}-1}\right\}$ which are the $2N_1$ ($2\times$ because the s_ℓ's are complex) unknowns to be solved and is

$$\boldsymbol{s} = \begin{pmatrix} s_{\frac{N_1}{2}} \\ s_{\frac{N_1}{2}+1} \\ s_{\frac{N_1}{2}+2} \\ \vdots \\ s_{N_2/2-1} \\ s^*_{-N_2/2} \\ s^*_{-N_2/2+1} \\ \vdots \\ s^*_{-\frac{N_1}{2}-2} \\ s^*_{-\frac{N_1}{2}-1} \end{pmatrix} \tag{30}$$

$\boldsymbol{s}$ is an $N_1 \times 1$ vector.

[d] This method works for $N_2 \leq 2N_1$, but the matrix entries in (28) will have to be derived.

The rhs of (28) is

$$\boldsymbol{C}_\theta = \begin{pmatrix} C_\theta(N_1) \\ C_\theta(N_1-1) - \sum_{j=1}^{1} \sigma^*_{-\frac{N_1}{2}+j-1} \sigma_{\frac{N_1}{2}+j-2} e^{i2\pi(-\frac{N_1}{2}+j-1)(N_1-1)(\Delta t)^2 \cot\theta} \\ C_\theta(N_1-2) - \sum_{j=1}^{2} \sigma^*_{-\frac{N_1}{2}+j-1} \sigma_{\frac{N_1}{2}+j-3} e^{i2\pi(-\frac{N_1}{2}+j-1)(N_1-2)(\Delta t)^2 \cot\theta} \\ \vdots \\ C_\theta(\frac{N_1}{2}+1) - \sum_{j=1}^{\frac{N_1}{2}-1} \sigma^*_{-\frac{N_1}{2}+j-1} \sigma_j e^{i2\pi(-\frac{N_1}{2}+j-1)(\frac{N_1}{2}+1)(\Delta t)^2 \cot\theta} \\ C_\theta(\frac{N_1}{2}) - \sum_{j=1}^{\frac{N_1}{2}} \sigma^*_{-\frac{N_1}{2}+j-1} \sigma_{j-1} e^{i2\pi(-\frac{N_1}{2}+j-1)\frac{N_1}{2}(\Delta t)^2 \cot\theta} \end{pmatrix} \tag{31}$$

which is a $(N_1/2+1) \times 1$ vector.

From (28), the number of equations is $2 \times (2 \times (N_1/2+1)) = 2N_1 + 4$ while the number of unknowns is $2N_1$. Therefore, (28) is overdetermined. There are many ways to solve (28) in this situation. Some standard techniques are the *singular value decomposition* (SVD) method and the *least squares fit* (LSF) method. We will use the LSF method `LeastSquares[]` built into *Mathematica* to solve for $\boldsymbol{s}$ in the demonstration.

Once we have the solution for the N_2 chunk, we can expand the size of the chunk $N_3 = 2N_2$ and solve for $\boldsymbol{s}$. This continues *ad infinitum* until we have the complete solution of $s(t)$.

Enhanced Bootstrap Algorithm with Least Squares Fit Demonstration

The enhanced bootstrap algorithm (EBA) is demonstrated with a *Mathematica* programme shown in *Appendix I*. The noise level has been set so that the maximum relative range at each point is ± 0.01 w.r.t. the true value of $I_{\alpha,\gamma}$. The first chunk has length $N_1 = 16$. Once we get the solution to this chunk with the previously described bootstrap algorithm, the second chunk $N_2 = 32$ is solved with the matrix equation (28) with the built-in `LeastSquares[]` function of *Mathematica*. Figure 3 shows the result. We can compare it to Figure 2 where a large chunk $N_1 = 32$ was used with the bootstrap algo-

rithm with the same relative noise level ± 0.01 only and we see that the results are much poorer than with the EBA. Furthermore, the EBA works even if $s_{-N_2/2} = 0$.

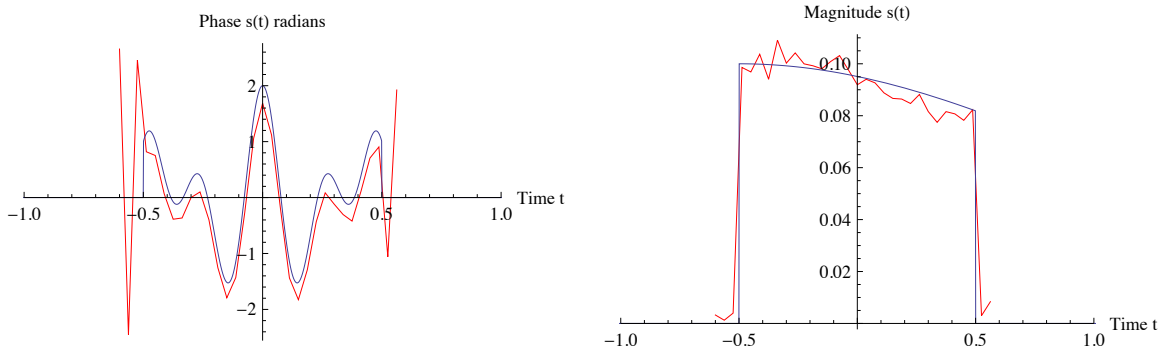


Figure 3 Demonstration of the enhanced bootstrap algorithm with noise in $I_{\alpha,\gamma}$. In this example, the maximum relative range of the uniform noise at each point is ± 0.01, $N_m = 256$, $N_1 = 16$ and $N_2 = 32$. Contrast this to Figure 2 where only the bootstrap algorithm was used. In this demonstration, the chunk N_2 encloses the zero outside $t = \pm 0.5$.

METHOD FOR MEASURING ULTRA-SHORT LASER PULSES

We will describe an optical setup that will allow us to measure ultra-short laser pulses in the femtosecond range. Our method relies on the well-known technique of coupling the longitudinal distribution of a laser pulse to its transverse plane with either a prism or a diffraction grating and then manipulating it with lenses to obtain spectral intensities required for our bootstrap algorithm. Our proposed setup is shown in Figure 4.

In this setup, the elements in the yellow box are completely described in *Appendix II*. We will give a brief synopsis here:

(*i*) The laser beam is incident on the transmission diffraction grating at an incident angle θ_i. When the laser pulse goes through the lens, the longitudinal pulse distribution is coupled to its transverse plane. The diffraction grating also performs the first Fourier transform of $s(t)$.

(*ii*) By placing Lens 1 at a distance equal to its focal length f_1 from the diffraction grating, the spectrum at the output of the grating is quadratic phase compensated. But an extra phase is also introduced when we apply the Fresnel diffraction formula (53) and we have to introduce a new variable $\bar{s}(t)$ (See (61) in Appendix II) to include this phase. The bootstrap algorithm will solve for $\bar{s}(t)$ and we will correct for the extra phases to get $s(t)$.

(*iii*) Lens 2 is placed at a distance equal to its focal length f_2 from Lens 1. The image at the back focal plane of Lens 2 is the exact Fourier transform of the image at P_g. Therefore, the image on P_2 is the result of two Fourier transforms applied to the $\bar{s}(t)$. This is the critical observation: It is well known that the Fourier transform of a Fourier transform of $\bar{s}(t)$ yields $\bar{s}(-t)$ and so the image at P_2 is *the* longitudinal distribution but phase modulated!

(*iv*) We add an adjustable aperture at the focal plane P_2. This is required for the EBA because we want to sample the centre of the distribution first and then increase the aperture to sample the rest of the distribution.

(*iii*) is the critical observation which tells us why the optical setup works because we have successfully transferred the longitudinal distribution to a transverse distribution and so the subsequent application of the bootstrap algorithm on this image will allow us to recover $\bar{s}(t)$.

Of course, what we have described above is completely useless if there is no way to accomplish this optically. It turns out that an invention by Lohmann[6] which uses a single lens (type I) or a couple of lenses (type II) is all that is required. We describe his invention in section *Type I and II Optics*. Using type I optics, we can place Lens 3 with focal length f_3 at a distance RQf_3 from P_2 to produce an image at P_3 which for $RQ = (\tan\alpha/2)(\sin\alpha)$ is the α-order FrFT of the image at P_2. For the bootstrap algorithm, we will need to measure the spectral intensity at α and γ. For example, we can measure the spectral intensity at P_3 for $\alpha = \pi/2$ and $\gamma = \pi/4$ and then apply the bootstrap algorithm for the extraction of $\bar{s}(t)$.

Type I and II Optics

Up to this point, we have used dimensionless variables. In this section we will introduce dimensionfull variables $\hat{x}$ and $\hat{x}'$ which, in this paper, have dimensions of length.

In both type I and II arrangements shown in Figure 5, the α-order FrFT of $\hat{s}(\hat{x})$ is given by

$$\hat{S}_\alpha(\hat{x}') = \hat{A}_\alpha \int_{-\infty}^{\infty} d\hat{x}\ \hat{s}(\hat{x}) \exp\left[i\pi \frac{(\hat{x}^2 + \hat{x}'^2)\cos\alpha - 2\hat{x}\hat{x}'}{\lambda F_3 \sin\alpha}\right] \tag{32}$$

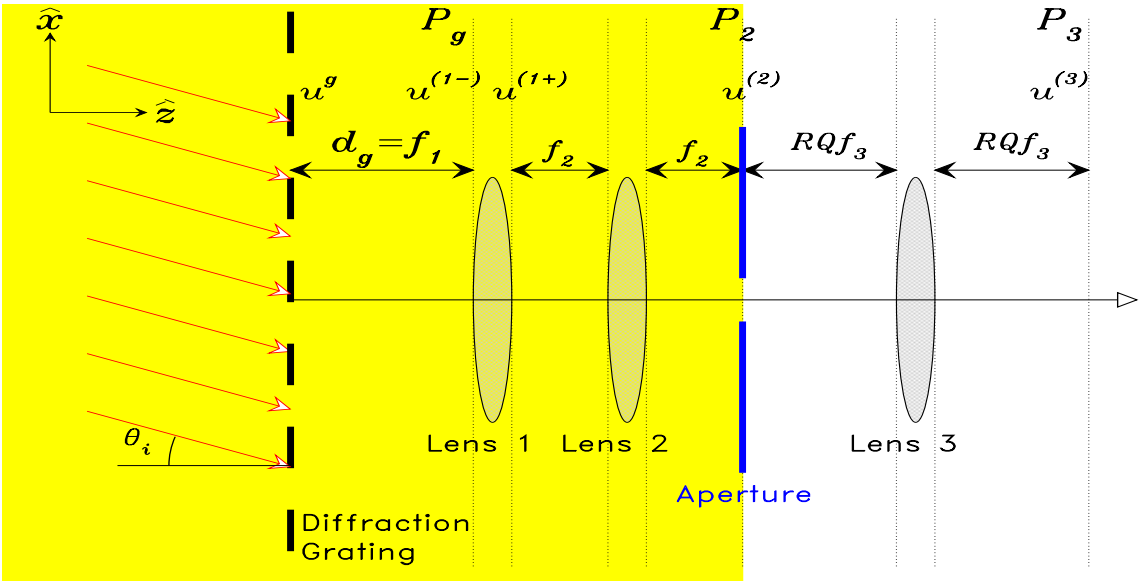


Figure 4 Our proposed setup for the measurement of femtosecond laser pulses. The incoming laser pulse is incident on the diffraction grating at θ_i. Lens 1 compensates the quadratic phase and Lens 2 Fourier transforms the spectral pattern from the diffraction grating. The position of Lens 3 is selected to give the α or γ-order FrFT of the pulse. The elements in the yellow box are completely described in *Appendix II*.

where

$$\hat{A}_\alpha = \frac{\exp\left[-i\left(\frac{\pi\,\mathrm{sgn}(\sin\alpha)}{4} - \frac{\alpha}{2}\right)\right]}{|\lambda F_3 \sin\alpha|^{\frac{1}{2}}} \tag{33}$$

and the scaling parameter λF_3 (which has dimensions of [length^2]), λ is the wavelength of the laser pulse and F_3 is the "local" focal length of the lens and depends on whether the type I or II configuration is chosen

$$\left.\begin{array}{lll} & F_3 = QF & \text{for both type I and II} \\ \text{Type I:} & R = \tan\dfrac{\alpha}{2} & Q = \sin\alpha \\ \text{Type II:} & R = \sin\alpha & Q = \tan\dfrac{\alpha}{2} \end{array}\right\} \tag{34}$$

where F is the focal length of the lens (c.f. local focal length F_3).

We can compare (32) and (33) with the original definition of the FrFT (1), (2) and (3).

They are identical except for $\hat{A}_\alpha = A_\alpha/|\lambda F_3|^{\frac{1}{2}}$ because the Fourier variables here are dimensionfull. The relationship between the dimensionfull and dimensionless representation is:

$$\hat{S}_\alpha(\hat{x}) = \frac{1}{|\lambda F_3|^{\frac{1}{2}}} S_\alpha(x) \tag{35}$$

We can also similarly discretise (32) using the same arguments as before to get

$$\hat{S}_\alpha(m\Delta\hat{x}') = \hat{A}_\alpha \Delta\hat{x} \sum_{k=-N/2}^{N/2-1} \hat{s}(k\Delta\hat{x}) \exp\left[\frac{i\pi}{\lambda F_3}\left[(k\Delta\hat{x})^2 + (m\Delta\hat{x}')^2\right]\cot\alpha - i\frac{2\pi km}{N}\right] \tag{36}$$

where $\Delta\hat{x}$ is the sampling length, $\Delta\hat{x}\Delta\hat{x}'/\lambda F_3 = \sin\alpha/N$ and we have assumed that $N = 2^q$ where $q \in \mathbb{N} \cup \{0\}$.

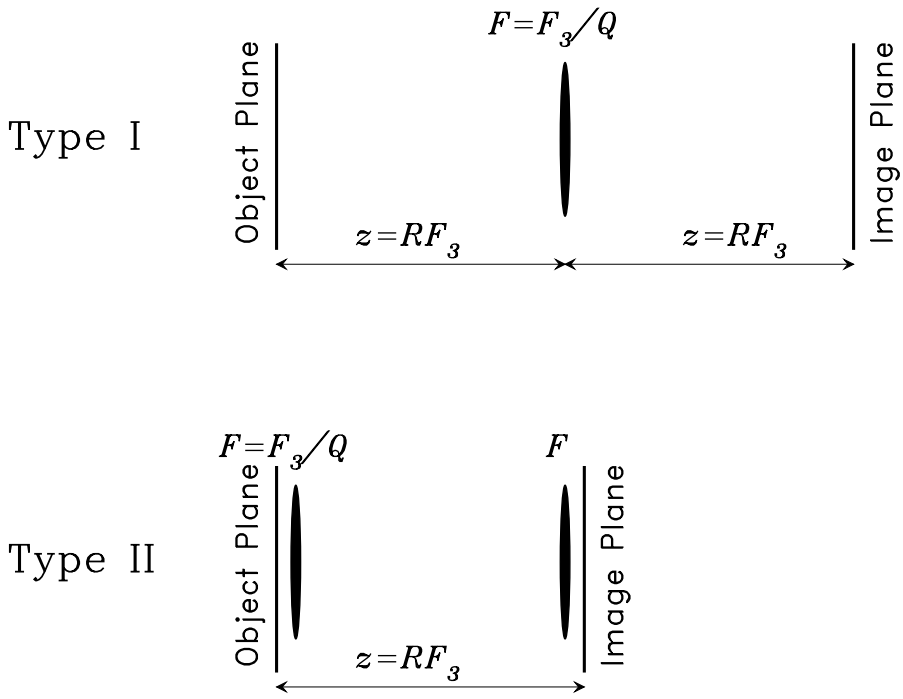


Figure 5 The type I and II FrFT optics.[6]

Therefore, by placing a converging lens represented by Lens 3 (focal length f_3) in

Figure 4 at a distance RQf_3 from Lens 2 we can produce an α-order FrFT of the image on the P_3 plane.

In the spirit of writing (36) into a dimensionless form which we can then easily compare it with our original discretised version of FrFT (5) we will introduce a new variable which has dimensions of length

$$\mu_3 = \sqrt{\lambda F_3} \tag{37}$$

and so $\Delta\hat{x}/\mu_3 \rightarrow \Delta x$ and $\Delta\hat{x}'/\mu_3 \rightarrow \Delta x'$ which are clearly dimensionless. When we substitute these new dimensionless variables into (37), we have

$$S_\alpha(m\Delta x') = A_\alpha \Delta x \sum_{k=-N/2}^{N/2-1} s(k\Delta x) \left(i\pi \left[(k\Delta x)^2 + (m\Delta x')^2 \right] \cot\alpha - i\frac{2\pi km}{N} \right) \tag{38}$$

where $\hat{s}(k\Delta\hat{x}) = \hat{s}(k\mu_3\Delta x) \rightarrow s(k\Delta x)$.

Numerical Demonstration 1

In this numerical demonstration of the optical setup shown in Figure 4, we will use a 100 fs pulse defined as

$$s_{\text{pulse}}(t) = \begin{cases} 1 & \text{if } |t| < 50 \text{ fs} \\ \\ 0 & \text{otherwise} \end{cases} \tag{39}$$

and set the slit size b of the diffraction grating to be 1 nm (which is unrealistically small) so that it satisfies the condition $b/\lambda \ll 1$, and $\theta_i = 85^\circ = 1.48$ rad. These parameters allow us to use (69) for $u^{(1-)}$. The other parameters used in the simulation are summarised in Table AII.2. Using these parameters, we can use the results of *Example A.II.2* and see that we only have 18 sampling points for the temporal pulse. However, 18 points are sufficient to reproduce $s_{\text{pulse}}(t)$ quite nicely with the bootstrap algorithm.

Since this is the first of two numerical demonstrations, we will go through the steps of this demonstration carefully. (Note: All the mathematical details are in *Appendix II* and the *Mathematica* input file in *Appendix IV*):

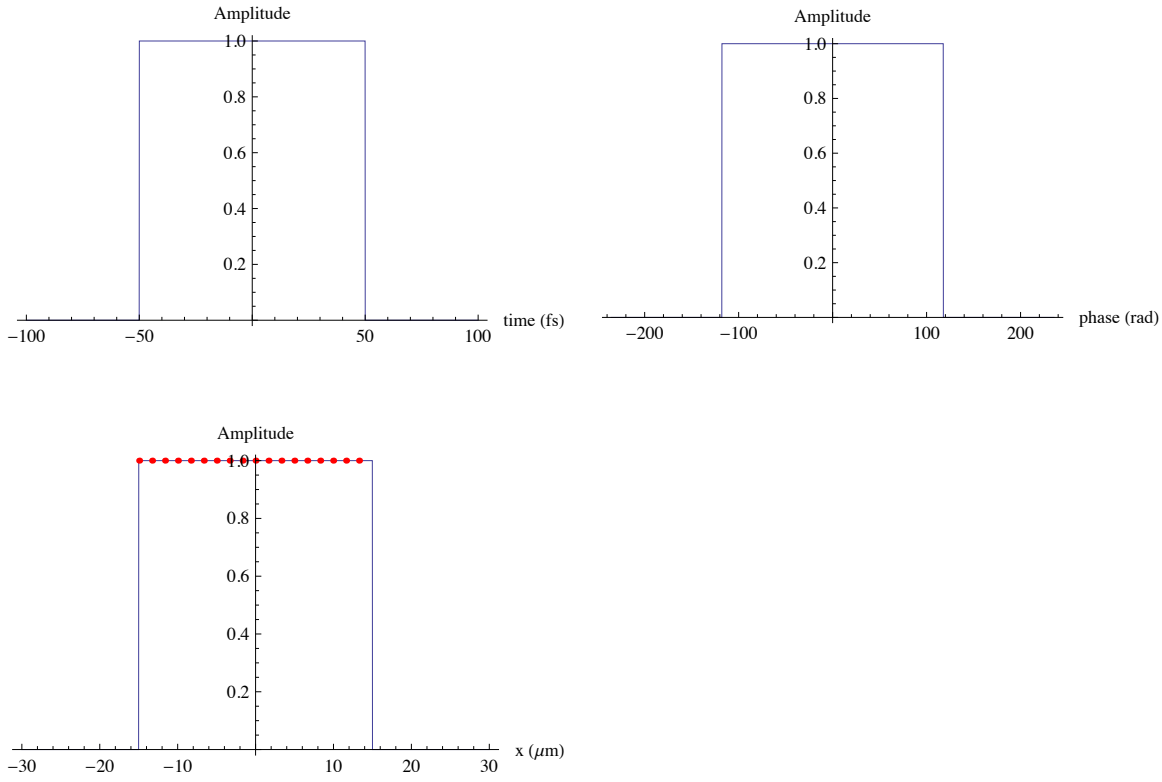


Figure 6 Different parametrisation of s_{pulse}. The red dots on the bottom left plot show how $s_{\text{pulse}}(ct)$ is sampled by the diffraction grid which are spaced $a \sin\theta_i$ apart.

(*i*) Figure 6 shows the input rectangular pulse parametrised in terms of time, phase and space. The red dots on the bottom left plot show the sampling points of the diffraction grid which are spaced $a \sin\theta_i$ apart.

(*ii*) Figure 7 shows the real and imaginary parts of $u^{(1-)}$. The magnification of this image is dependent on f_1. In principle, we should be able to use multiple resonances for averaging. We select $f_1 = 5$ cm because one resonance image fits in about 3 cm of space, i.e. fits in a typical lens. We also notice the high frequency component in this plot which we will correct with Lens 1. We can compare this plot to Figure 8 where $f_1 = 1$ cm. We select only one of these "resonances" for the analysis.

(*iii*) Figure 9 shows the result of correcting the high frequency component of $u^{(1-)}$ with

Lens 1. These plots reveal the underlying sinc() component which is the FT of $s_{\text{pulse}}(t)$.

(*iv*) Figure 10 shows the image $u^{(2)}$ which is a magnified version of $\bar{s}$. This is the key reason why the optics setup works — Lens 2 recreates a magnified version of the input pulse $|s(ct)|$. The magnification factor is f_2/f_1 (See *Appendix III*). The width of the pulse in Figure (a) is 20× the pulse shown in Figure 6. The real and imaginary parts of $u^{(2)}$ show an oscillation which originates from $e^{ikx^2/2f_1}e^{ikax\sin\theta_i}$.

(*v*) For illustrative purposes, we will construct a magnified version of $\bar{s}$ which we see at the focal point of Lens 2, i.e. on P_2. The following equation essentially comes from (91)

$$\bar{s}'(-\nu x) = s(-x)e^{-i\pi/2}e^{ik(-x)^2/2f_1}e^{ik(-x)\sin\theta_i} \tag{40}$$

The term $e^{-i\pi/2}$ has been included to reflect the two Fresnel propagations from the diffraction grid to P_2. We note that the sampling distance is a (note: not $a\sin\theta_i$) *just after* the grid and so the oscillations are grossly under-sampled. See *Appendix III*. In Figure 11 we superimpose the "highly" sampled solution and the under-sampled solution. The under-sampled solution looks nearly identical to Figures 10(c) and (d). In fact, when we superimpose them in Figure 12 the two match up really nicely!

(*vi*) Let us get back to the objective of the demonstration. We will generate four chunks of spectral intensity data for the EBA for $\alpha = \pi/2$ and $\gamma = \pi/4$ and aperture radii $16\Delta x_{\text{pixel}} = 0.2246$ mm and $32\Delta x_{\text{pixel}} = 0.448$ mm. With the selection of α and γ and fixing the "local" focal length $F_3 = 1$ m we can calculate focal length of Lens 3 and where to place it and the CCD camera w.r.t. the aperture. See Table 1. The four chunks of the spectral intensity data are shown in Figure 13.

(*vii*) The spectral intensity data shown in Figure 13 are normalised so that the total

integral power is one. This is necessary because in experiments, we expect to have to integrate many pulses to get a good signal on the CCD. See the *Mathematica* input file. The result of applying the EBA to the normalised spectra is shown in Figure 14. The size of the magnitude is larger than the input signal because the input spectra were normalised. The phase clearly has the contribution from Lens 1 $e^{ikx^2/2f_1}e^{ikax\sin\theta_i}$ and is also offset. Therefore, the last thing that is left for us to do is to correct the phase.

(*viii*) The phase which originates from Lens 1 is corrected using the formula

$$\phi_{\text{correction}}(\nu x) = e^{ikx^2/2f_1}e^{ikx\sin\theta_i} \tag{41}$$

which takes into account the magnification factor ν. The corrected phase is shown in Figure 15 with the original input pulse rescaled without the magnification factor.

Table 1 Type I Optics Parameters for $F_3 = 1.0$ m

α	$f_3 = F_3/\sin\alpha$ (m)	RQf_3 (m)
$\pi/2$	1.0	1.0
$\pi/4$	1.4	0.41

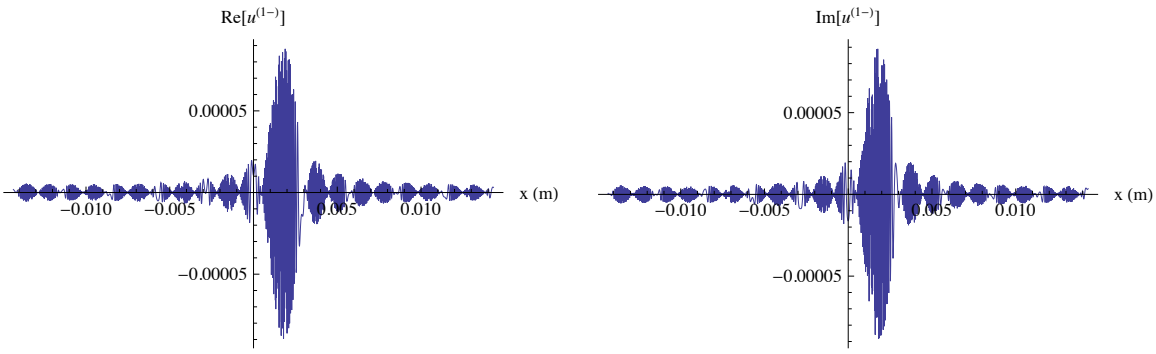


Figure 7 The real and imaginary parts $u^{(1-)}$ which is the image on P_g. The magnification of this image is dependent on f_1. In this case $f_1 = 5$ cm. Notice the high frequency components in $u^{(1-)}$.

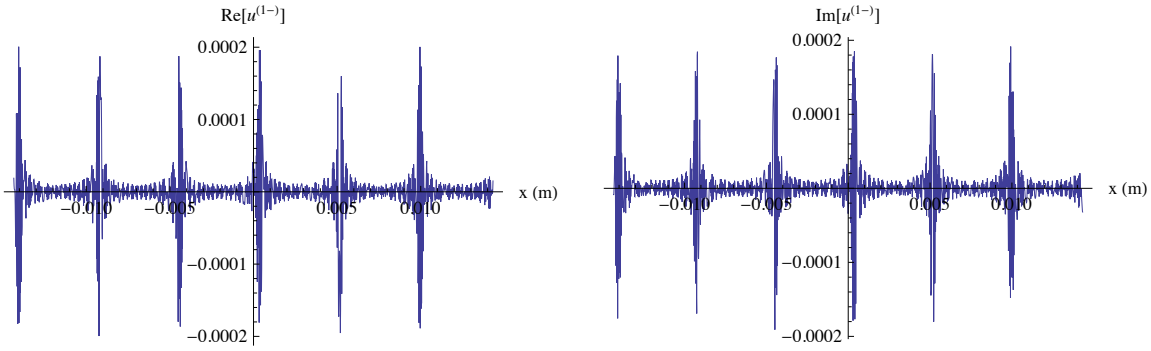


Figure 8 This is $u^{(1-)}$ for $f_1 = 1$ cm. Multiple resonances are seen within the $\sim$3 cm aperture of the lens.

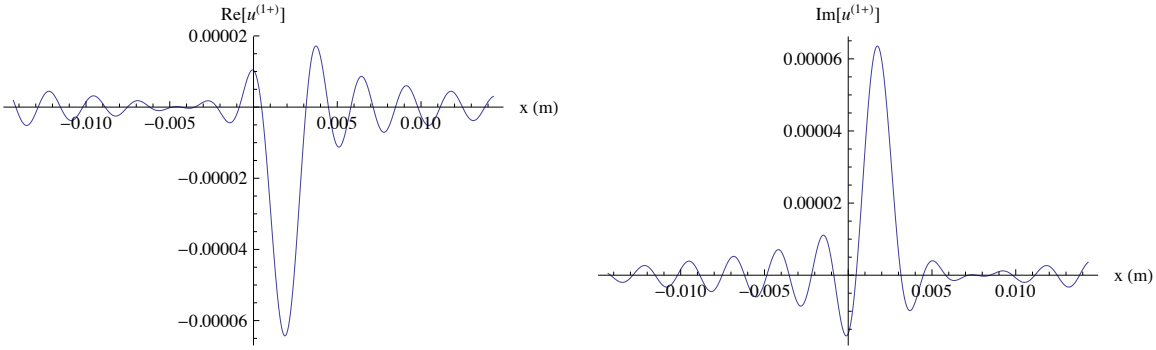


Figure 9 The high frequency component of $u^{(1-)}$ are corrected by Lens 1. The correction reveals the underlying sinc() component which is the FT of $s_{\text{pulse}}(t)$.

Figure 15 compares the input pulse with the reconstructed pulse. The reconstructed magnitude is similar up to a scale factor but the reconstructed phase has structures which are absent from the input phase — it has wiggles at the edges because of the division of small imaginary parts by real parts in this area. The reconstructed phase also has a kink around 0 which comes from the imperfect phase correction which can be seen in Figure 16. This is easily corrected with better data analysis but will not be done here.

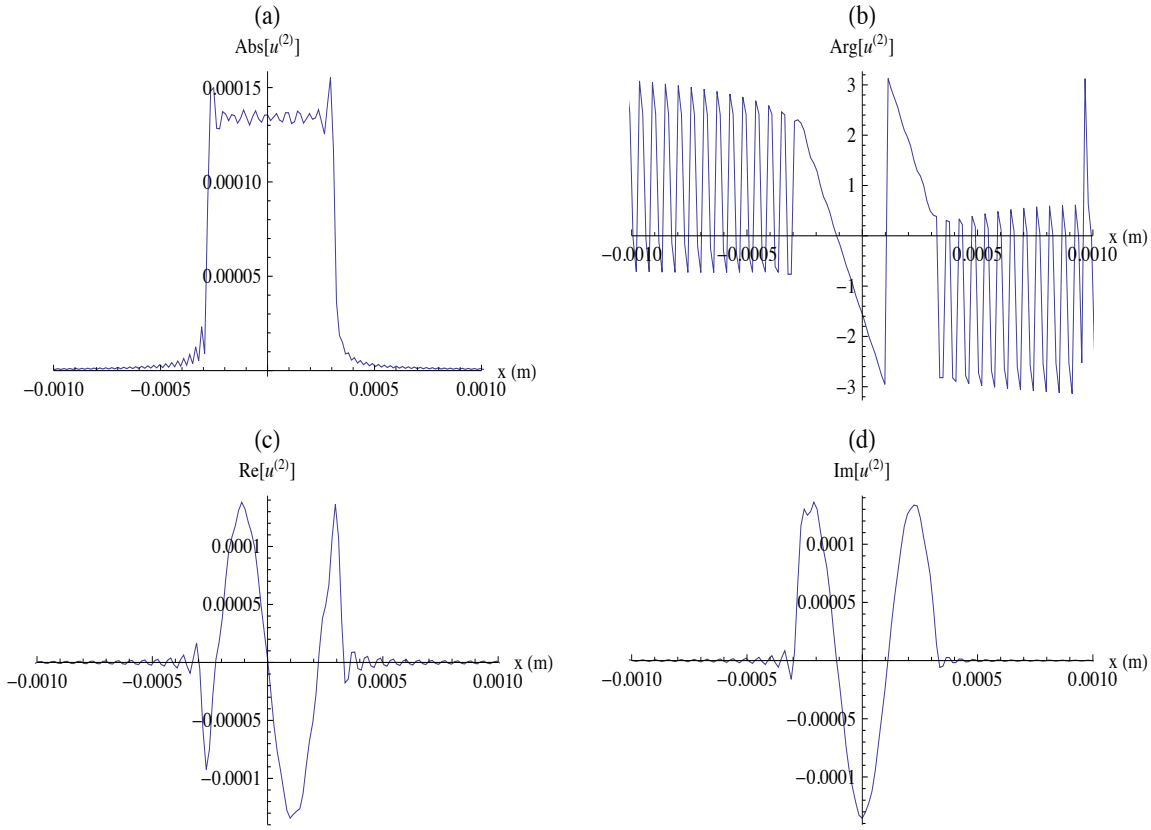


Figure 10 $u^{(2)}$ plotted in (magnitude, phase) and (real, imaginary). This plot shows why the optical setup works — Lens 2 recreates a magnified version of $|s(ct)|$. The phases are more complicated because it contains $e^{ikx^2/2f_1}e^{ikax\sin\theta_i}$.

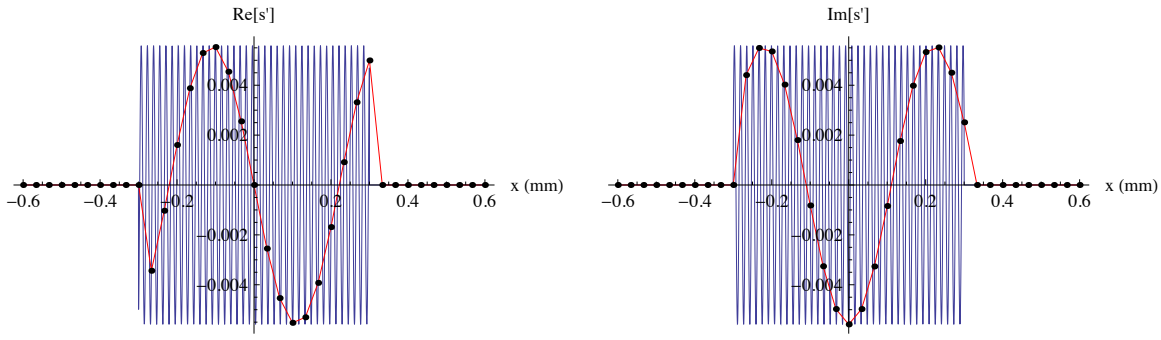


Figure 11 This figure shows the construction of $\bar{s}'$ using (40). The sampling distance is $a \sin\theta_i$ and so under-samples the oscillations. The black dots are the sampling points and the red curve is created when the black dots are joined. The blue curve is the "highly" sampled $\bar{s}'$.

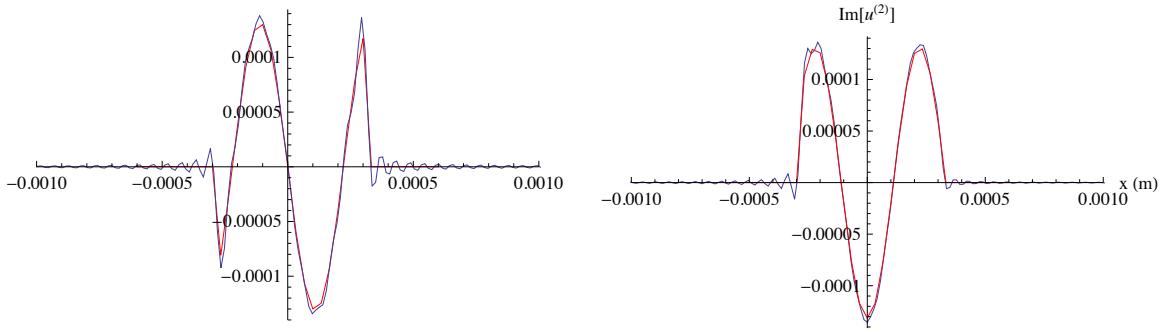


Figure 12 We superimpose $\bar{s}'$ (red curve) and $u^{(2)}$ (blue curve) from Figure 10(c) and (d). Clearly the two match up very well.

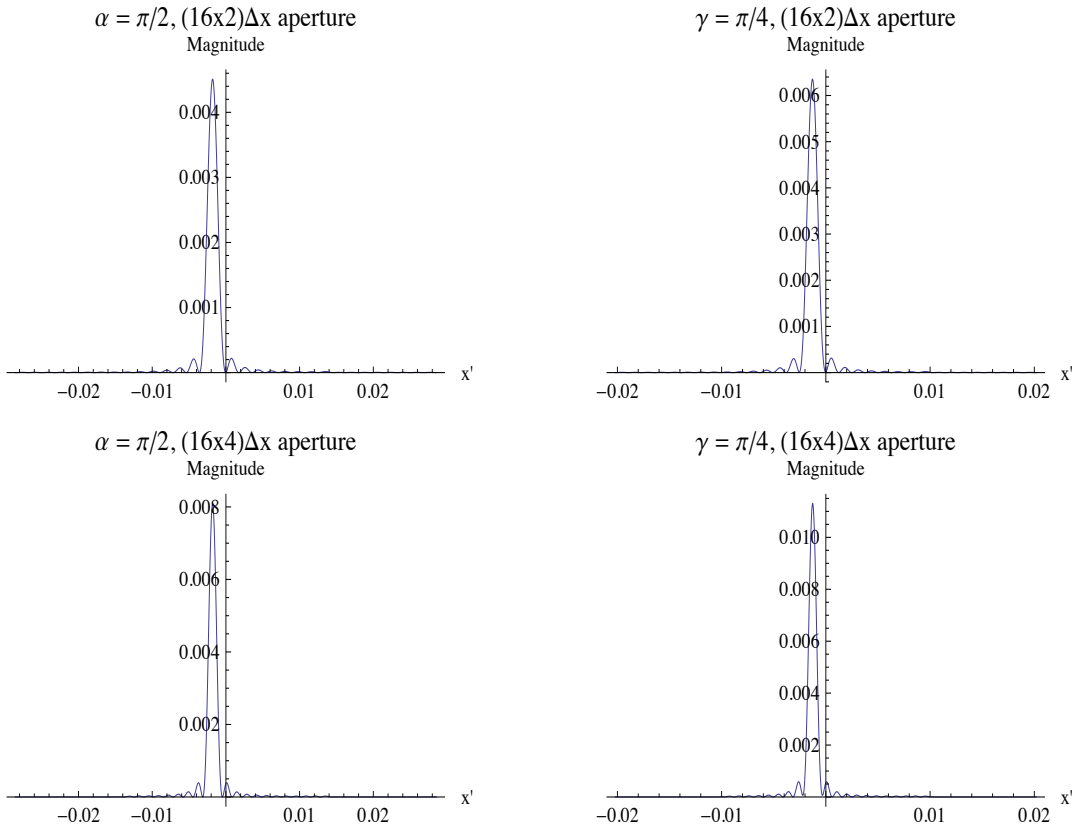


Figure 13 The spectral intensities for $\alpha = \pi/2$ and $\gamma = \pi/4$ and aperture radii $(16\Delta x_{\text{pixel}}) = 0.224$ mm and $(32\Delta x_{\text{pixel}}) = 0.448$ mm. These are the data sets which will be used for the EBA. These abscissa of these plots is the dimensionless variable x' which is the length of the spectral image on the CCD divided by μ_3.

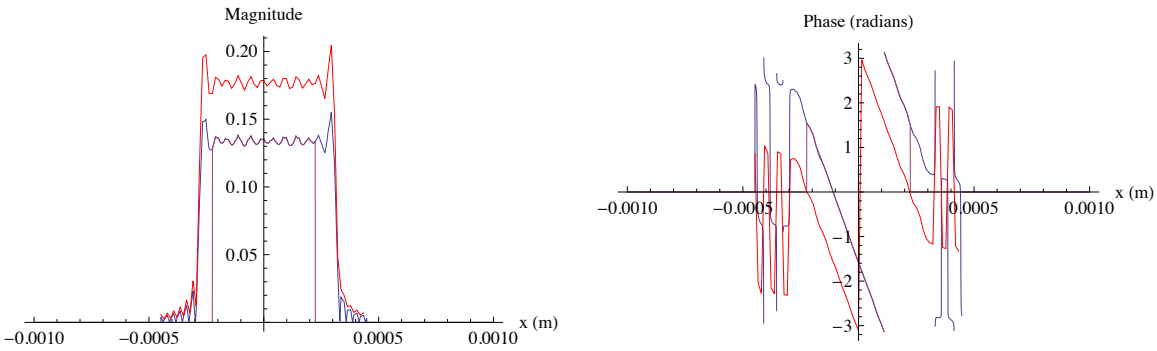


Figure 14 The results of the EBA with the normalised spectra. The red traces are the results from the EBA. It is clear that the magnitude is larger than the input signal because the intensity has been normalised. The phase has a constant offset as expected. The magenta trace comes from the small radius aperture $16\Delta x_{\text{pixel}}$ and the blue is from the large radius aperture $32\Delta x_{\text{pixel}}$.

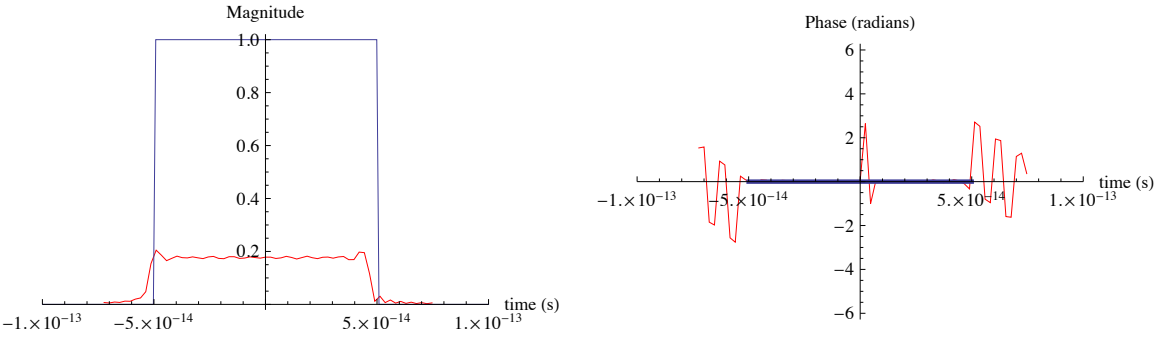


Figure 15 After the phase is corrected using (41) and the magnification factor from Lens 1 and 2 is rescaled out from Figure 14, we can compare the input pulse (blue trace) to the measured pulse (red trace). There is a kink in the reconstructed phase at $t = 0$ because of imperfect the phase correction which we can see from Figure 16.

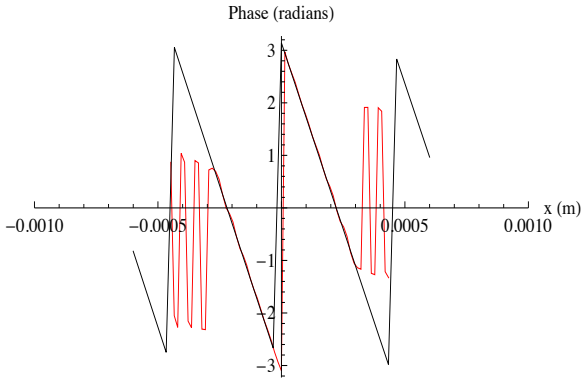


Figure 16 The phase correction formula (41) is plotted here in black with the reconstructed phase in red. Notice the imperfect match in phase between the correction and reconstruction around $x = 0$. Clearly this can be easily corrected with better data analysis.

Numerical Demonstration 2

In this demonstration, we will use the input pulse (27) from Cong *et al* in the *Demonstration of the Bootstrap Algorithm*. It turns out that if we use the same diffraction grating as in *Numerical Demonstration 1*, the 18 points used to sample the input pulse is insufficient to get a good reproduction of it after Lens 2 — which is the key for why the optical setup works at all. Therefore, we will use a diffraction grating with double the number lines per mm $a' = 1/1200 = 0.83 \times 10^{-3}$ mm. With this slit distance and an incident angle of 85°, the number of sampling points are doubled on the input signal from 18 to 36.

Next, we will make the slit width $b = a/8 = 104$ nm so that $b/\lambda = 0.13 < 0.5$ is not so large as to make our approximation of $u^{(1-)}$ from (75) useless and so can still be used here. Otherwise, the other optical parameters remain the same as before.

Figure 17 shows the two cases for a and a' for the same slit width b at the focal plane P_2. It is clear that the number of sampling points is critical for the good reproduction of the input signal.

Finally, using the *Mathematica* programme shown in *Appendix IV*, we can recreate the input signal. The final results are shown in Figure 18. There is a kink in the phase near $t = 0$ which comes from the imperfect phase correction. This can be easily corrected with better data analysis. As expected there is a scale factor difference between the input magnitude and the reconstructed magnitude.

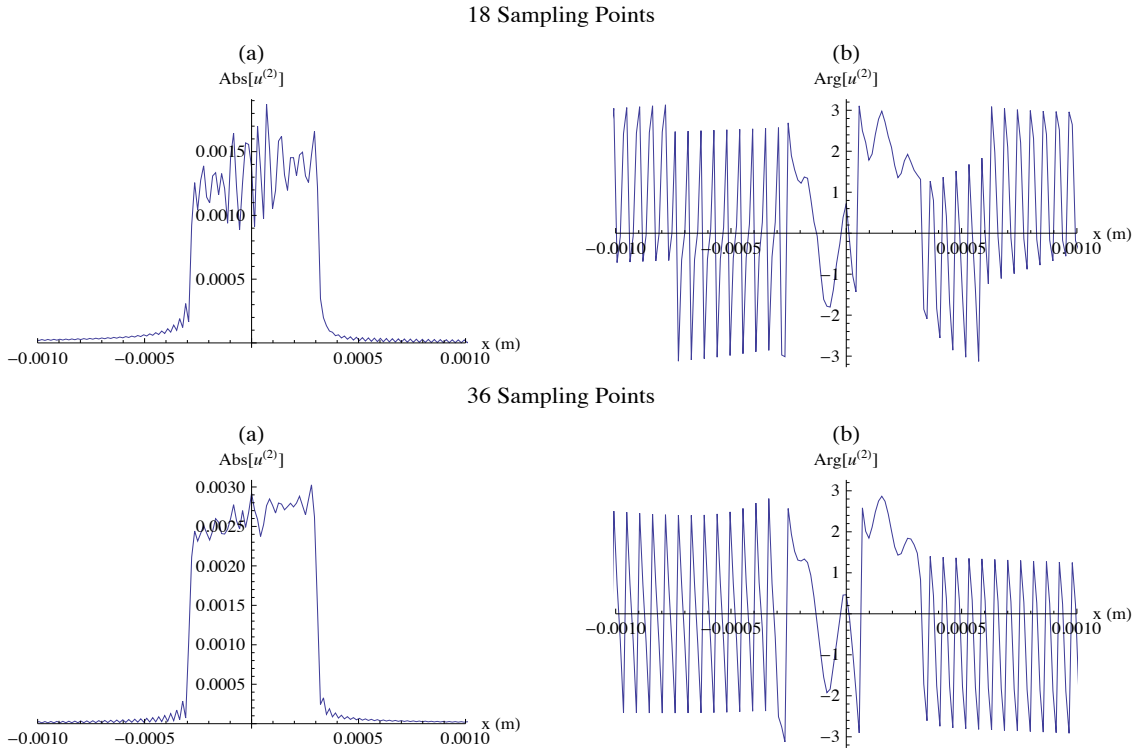


Figure 17 The number of sampling points have a dramatic effect on how the signal looks like at focal plane P_2. It is clear that increasing the number of samples from 18 to 36 makes the signal look more like the magnitude of the input signal shown in Figure 1.

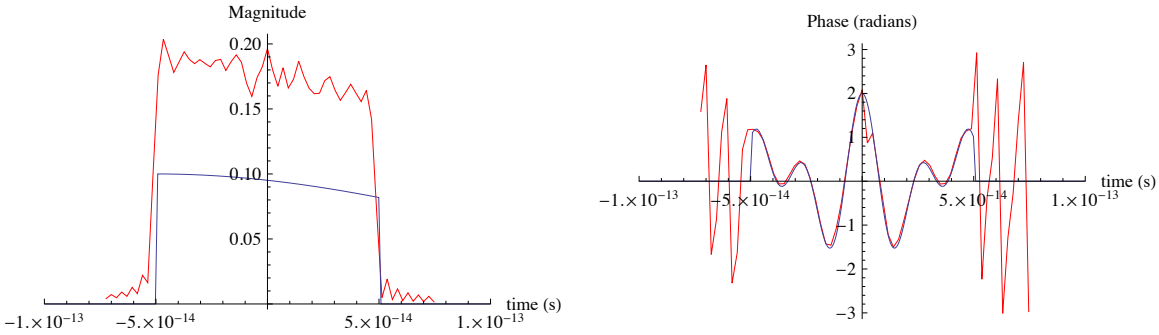


Figure 18 The reconstructed input signal (red) is plotted with the input signal (blue). There is a kink in the phase near $t = 0$ which comes from the imperfect phase correction. This can be easily corrected with better data analysis. As expected there is a scale factor difference between the input magnitude and the reconstructed magnitude.

CONCLUSION

We have demonstrated how the EBA works with and without noise and we have shown that the algorithm works very well. We have also proposed a possible optical setup to perform the EBA on an input signal and demonstrated that it works well in computer simulations. However, in a real setup there are errors in alignment and positions of the optical elements which have not been taken into account in the simulations. These errors will definitely affect the effectiveness and accuracy of the results which we have not attempted to examine in this long paper. This means that further study will need to be done to see if this method is competitive with FROG or even works well at all in real life.

ACKNOWLEDGEMENTS

I would like to thank J.H. Ruan and T. Maxwell for informally discussing FROG with me at a Beams Division party in late spring of 2010. This discussion inspired me to try to see if FrFT which I had worked on can be used to reconstruct temporal signals from intensity spectra. Furthermore, I would like to thank them for reading through this overly long paper.

Appendix I: Demonstration of the Bootstrap Algorithm and the Enhanced Bootstrap Algorithm with Least Squares Fit

Demo of Bootstrap Method

This is a demonstration of the boostrap method which will allow us to reconstruct the input signal from two of its fractional Fourier intensity spectra.

Discrete Fractional Fourier Transform

An n point α-order discrete fractional Fourier transform (DFrFT) of a function s[t] is:

In[1]:=
```
DFrFT[s_, n_, α_, Δt_, m_] := Exp[I α / 2] / Sqrt[I Sin[α]] Δt
    Sum[s[k Δt] Exp[I π (((m Sin[α]) / (n Δt))^2 + (k Δt)^2) Cot[α] - I 2 π k m / n], {k, -n / 2, n / 2 - 1}];
```

Arguments of DFrFT[] are:

- s: function to be transformed
- n: number of points to be transformed. Must be power of 2
- α: order of the transform
- Δt: time step
- m: returns value of transform at m Δf

It is assumed that the time step gives the frequency step with the relationship Δf Δt= Sin[α]/n.

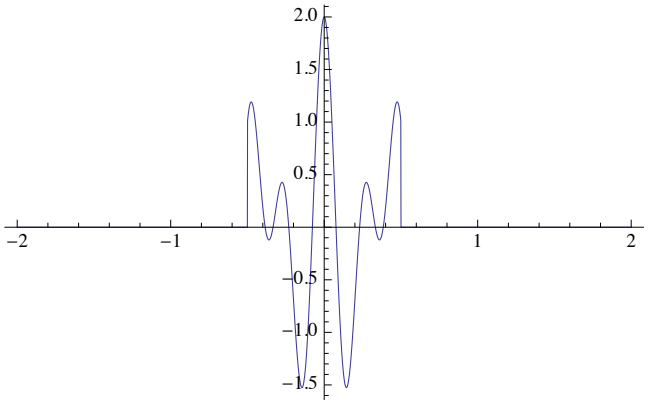
Example Input Function sinput[t]

The example input function sinput[t] comes from Cong et al. (second example). Note: We have shifted Cong's example by 0.5 units so that the symmetry point is at t=0 rather than at t=1. It is assumed that the input signal is zero out side +/- 0.5 units. Note that in the reconstruction, we will only reconstruct s[t] within +/- 0.5. We have also reduced the magnitude from 1 to 0.1.

In[2]:=
```
sinput[t_] := η[t] 0.1 Exp[-0.2 (t + 0.5)^2] Exp[I (Sin[5 π (t + 0.5)] + Cos[8 π (t + 0.5)])];
η[t_] := If[-0.5 ≤ t < 0.5, 1, 0];
```

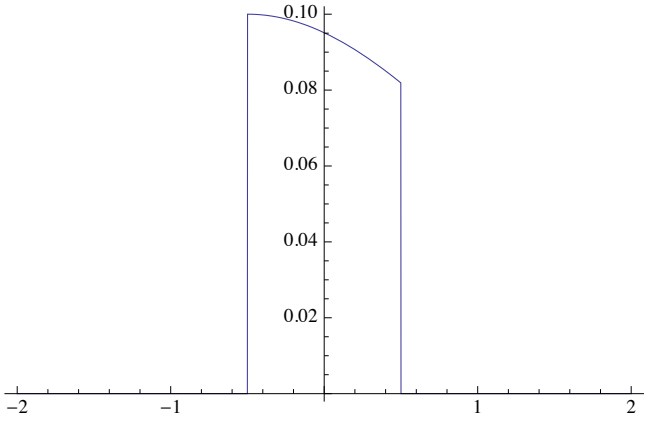
In[4]:=
```
Plot[Arg[sinput[t]], {t, -2, 2}, PlotRange → All]
```

Out[4]=



In[5]:=
```
Plot[Abs[sinput[t]], {t, -2, 2}]
```

Out[5]=



■ Measurement Parameters

We select $\alpha = \pi/2$ and $\gamma = \pi/4$. The nonzero part of the signal is divided into n = 32 samples in the time interval +/-0.5. Therefore $\Delta t = 1/32$. We pad the signal outside this interval with zeroes, we go from -nm/2 to -n/2-1 and n/2 to nm/2-1.

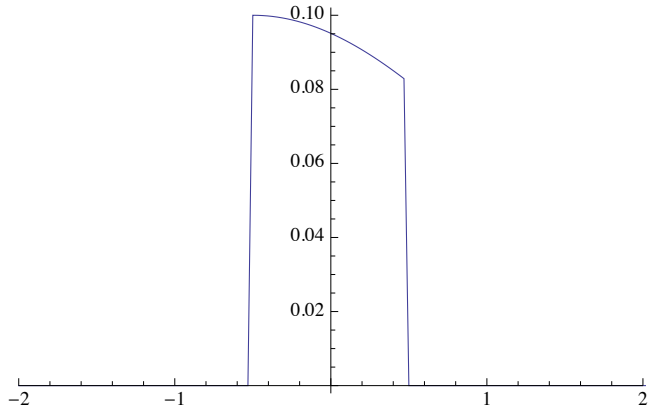
In[6]:=
```
nm = 256; (*total number of samples*)
n = 32;  (* number of samples where s is the real signal and not padding*)
α = π / 2;
γ =  π / 4;
Δt = 1.0 / n;

Δfα = Sin[α] / (nm Δt);
Δfγ = Sin[γ] / (nm Δt);
```

Check that the Δt sampling samples the input data well.

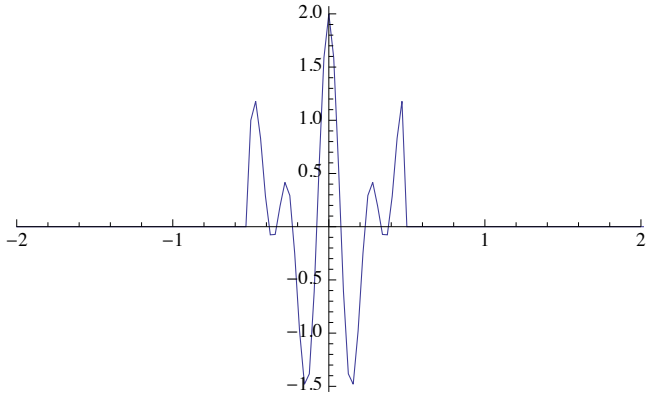
In[13]:=
```
ListPlot[Table[{i Δt, Abs[sinput[i Δt]]}, {i, -nm / 2, nm / 2 - 1}],
 Joined → True, PlotRange → {{-2, 2}, All}]
```

Out[13]=



In[14]:=
```
ListPlot[Table[{i Δt, Arg[sinput[i Δt]]}, {i, -nm / 2, nm / 2 - 1}],
 Joined → True, PlotRange → {{-2, 2}, All}]
```

Out[14]=



■ Create the I_α and I_γ

We need nm points of intensity from I_α and I_γ to create n points of s.

In[15]:=
```
iα = Table[{m Δfα, Abs[ DFrFT[sinput, nm, α, Δt, m]]^2}, {m, -nm / 2, nm / 2 - 1}] // N;
iγ = Table[{m Δfγ, Abs[ DFrFT[sinput, nm, γ, Δt, m]]^2}, {m, -nm / 2, nm / 2 - 1}] // N;
```

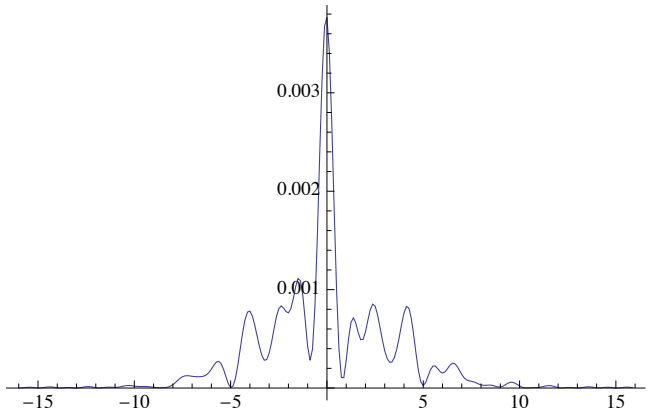
Make functions for the data sets

In[17]:=
```
Iα[f_] := iα[[Round[f / Δfα] + nm / 2 + 1]][[2]];
Iγ [f_] := iγ[[Round[f / Δfγ] + nm / 2 + 1]][[2]];
```

Plot them out

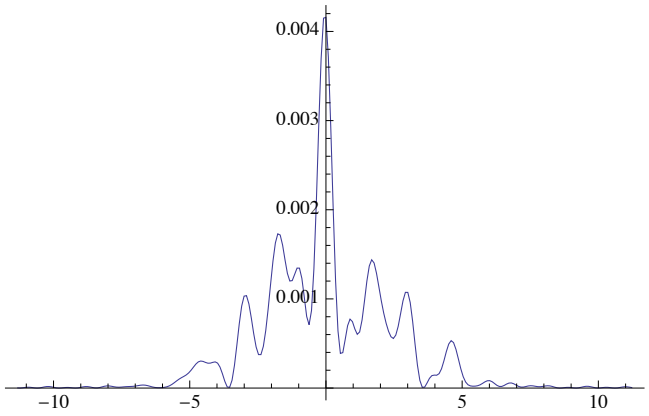
In[19]:=
```
ListPlot[iα, Joined → True, PlotRange → All]
```

Out[19]=



In[20]:=
```
ListPlot[iγ, Joined → True, PlotRange → All]
```

Out[20]=



$I_\alpha$and I_γ are the intensities which we measure.

The Bootstrap Method

We define the equations for the bootstrap method defined as equation (19) in the paper.

In[21]:=
```
u[θ_, m_] := Exp[I 2 π (-n / 2) (n - m) (Δt)^2 Cot[θ]];
v[θ_, m_] := Exp[I 2 π (-n / 2 + m - 1) (n - m) (Δt)^2 Cot[θ]];
Cα[k_] :=
  Sin[α] / (nm Δt^2) Exp[-I π (k Δt)^2 Cot[α]] Sum[Iα[m Δfα] Exp[I 2 π k m / nm], {m, -nm / 2, nm / 2 - 1}];
Cγ[k_] := Sin[γ] / (nm Δt^2) Exp[-I π (k Δt)^2 Cot[γ]] Sum[Iγ[m Δfγ] Exp[I 2 π k m / nm], {m, -nm / 2, nm / 2 - 1}];
Σ[θ_, m_] := Sum[Conjugate[s[-n / 2 + j]]
    s[n / 2 - m + j] Exp[I 2 π (n - m) (-n / 2 + j) (Δt)^2 Cot[θ]], {j, 1, m - 2}];
d[m_] := Det[{{u[α, m], v[α, m]}, {u[γ, m], v[γ, m]}}];
```

Check that Cα[0] = Cγ[0]

In[27]:=
```
Cα[0]
```

Out[27]=
```
0.283636
```

In[28]:=
```
Cγ[0]
```

Out[28]=
```
0.283636
```

They are equal, so at least energy is conserved!

Without loss of generality, we will set s[-N/2]=σ=1 first and then use equation equation (25) to solve for s[-N/2] because every s[n] is either multiplied or divided by s[-N/2]. The correct value of σ in the "Calculate the magnitude s[-n/2]" section below

In[29]:=
```
σ = 1;
```

Calculate cα and cγ using Eq(20)

In[30]:=
```
cα = Cα[n - 1] Exp[I π n (n - 1) Δt^2 Cot[α]]
```
Out[30]=
```
0.00815697 + 0.00147171 i
```

Check that the above is the same when Cγ is used

In[31]:=
```
Cγ[n - 1] Exp[I π n (n - 1) Δt^2 Cot[γ]]
```
Out[31]=
```
0.00815697 + 0.00147171 i
```

From equation (20),

In[32]:=
```
s[n / 2 - 1] = cα / σ
```
Out[32]=
```
0.00815697 + 0.00147171 i
```

They are equal as expected!

From equation (21), we can solve for s[n/2-2] and s[-n/2+1]

In[33]:=
```
s[n / 2 - 2] = Chop[First[1 / Conjugate[σ] d[2] (v[γ, 2] - v[α, 2]).(Cα[n - 2] / Cγ[n - 2])]][[1]]]
```
Out[33]=
```
0.00826918 - 0.00140695 i
```

In[34]:=
```
s[-n / 2 + 1] = Chop[First[σ / Conjugate[cα d[2]] Conjugate[(-u[γ, 2] u[α, 2]).(Cα[n - 2] / Cγ[n - 2])]]][[1]]]
```
Out[34]=
```
0.983918 + 0.177522 i
```

Now we can continue the bootstrap process until every element is done

In[35]:=
```
For[m = 3, m ≤ n / 2, m++,
  s[n / 2 - m] =
   Chop[First[1 / Conjugate[σ] d[m] (v[γ, m] - v[α, m]).(Cα[n - m] - Σ[α, m] / Cγ[n - m] - Σ[γ, m])]][[1]]];
  s[-n / 2 + m - 1] = Chop[First[σ / Conjugate[cα d[m]]
      Conjugate[(-u[γ, m] u[α, m]).(Cα[n - m] - Σ[α, m] / Cγ[n - m] - Σ[γ, m])]]][[1]]];
 ];
```

■ Calculate the magnitude s[-n/2]

We can calculate the magnitude of s[-n/2] using equation (25) of the paper. We will denote s[-n/2] = σ

In[36]:=
```
s[-n / 2] = σ; (* placeholder for the above calculations *)
```

In[37]:=
```
sol = Solve[σ2 Sum[Abs[s[-n / 2 + m - 1]]^2, {m, 1, n / 2}] +
    1 / σ2 Sum[Abs[s[n / 2 - m]]^2, {m, 1, n / 2}] == Cα[0], σ2]
```
Out[37]=
```
{{σ2 → 0.00826499}, {σ2 → 0.01}}
```

We select between the two solutions by using equation (26)

s0 = s[-n/2] G$\alpha\gamma$[n/2 + 1]

```
In[38]:= Sqrt[σ2] Chop[First[1/Conjugate[cα d[n / 2 + 1]] Conjugate[
            (-u[γ, n / 2 + 1] u[α, n / 2 + 1]).(Cα[n - (n / 2 + 1)] - Σ[α, n / 2 + 1]
                                               Cγ[n - (n / 2 + 1)] - Σ[γ, n / 2 + 1])]][[1]]] /. Last[sol]
Out[38]= 0.0513951 + 0.0800432 i
```

The other part which is also s[0] from equation (26)

s0 = $\frac{1}{s[-n/2]}$ F$\alpha\gamma$[$n/2$]

```
In[39]:= 1/Sqrt[σ2] Chop[First[1/d[n / 2] (v[γ, n / 2] - v[α, n / 2]).(Cα[n - n / 2] - Σ[α, n / 2]
                                                                      Cγ[n - n / 2] - Σ[γ, n / 2])]][[1]]] /.
          Last[sol]
Out[39]= 0.0513951 + 0.0800432 i
```

Therefore, the second solution in sol is the correct one because the s0's calculated above are the same whether using σG$\alpha\gamma$[n/2+1] or $\frac{1}{\sigma}$ F$\alpha\gamma$[$n/2$], i.e.

```
In[40]:= σ = Sqrt[σ2] /. Last[sol]
Out[40]= 0.1
```

- **Check other solution**

This is a quick check that the other solution is not correct

```
In[41]:= Sqrt[σ2] Chop[First[1/Conjugate[cα d[n / 2 + 1]] Conjugate[(-u[γ, n / 2 + 1] u[α, n / 2 + 1]).
            (Cα[n - (n / 2 + 1)] - Σ[α, n / 2 + 1]
             Cγ[n - (n / 2 + 1)] - Σ[γ, n / 2 + 1])]][[1]]] /. First[sol]
Out[41]= 0.0467243 + 0.0727688 i

In[42]:= 1/Sqrt[σ2] Chop[First[1/d[n / 2] (v[γ, n / 2] - v[α, n / 2]).(Cα[n - n / 2] - Σ[α, n / 2]
                                                                      Cγ[n - n / 2] - Σ[γ, n / 2])]][[1]]] /.
          First[sol]
Out[42]= 0.0565329 + 0.0880447 i
```

They clearly do not agree, therefore this is the wrong solution to pick.

Normalize s

Once we have σ, we can normalize all the s's

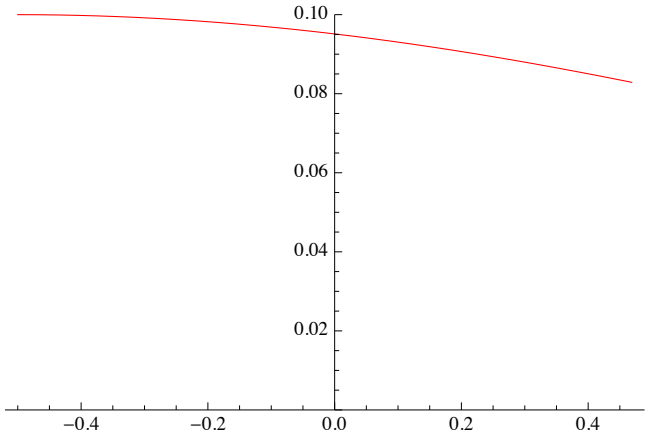
In[43]:=
```
For[m = 1, m ≤ n / 2, m++,
  sn[-n / 2 + m - 1] = σ s[-n / 2 + m - 1];
 ];
For[m = 1, m ≤ n / 2, m++,
  sn[n / 2 - m] = 1/σ s[n / 2 - m];
 ];
```

In[45]:=
```
sMag = Table[{k Δt, Abs[sn[k]]}, {k, -n / 2, n / 2 - 1}];
sArg = Table[{k Δt, Arg[sn[k]]}, {k, -n / 2, n / 2 - 1}];
sRe = Table[{k Δt, Re[sn[k]]}, {k, -n / 2, n / 2 - 1}];
sIm = Table[{k Δt, Im[sn[k]]}, {k, -n / 2, n / 2 - 1}];
```

■ Plots

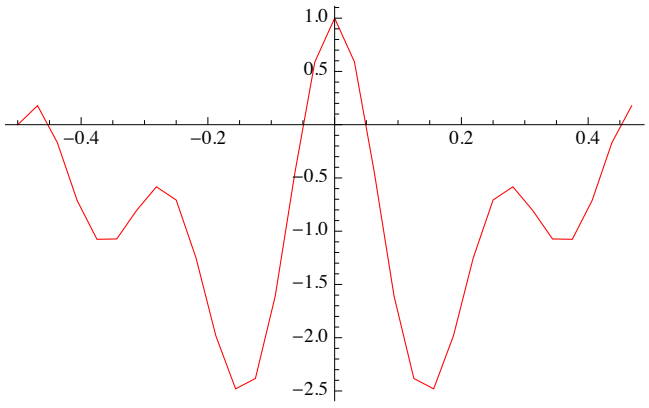
In[49]:=
```
ListPlot[sMag, Joined → True, PlotStyle → RGBColor[1, 0, 0], PlotRange → {0, 0.1}]
```

Out[49]=



In[50]:=
```
ListPlot[sArg, Joined → True, PlotStyle → RGBColor[1, 0, 0]]
```

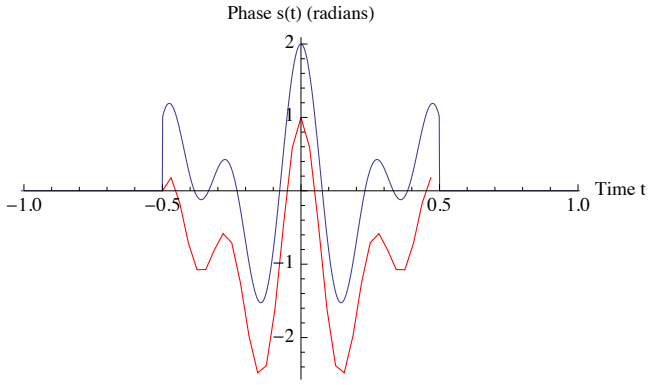
Out[50]=



■ Compare the solution to sinput

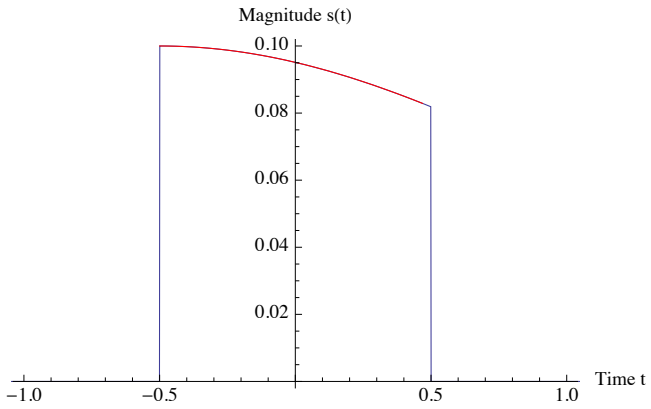
In[54]:=
```
Show[%4, %50, PlotRange → {{-1, 1}, All}, AxesLabel → {"Time t", "Phase s(t) (radians)"}]
```

Out[54]=



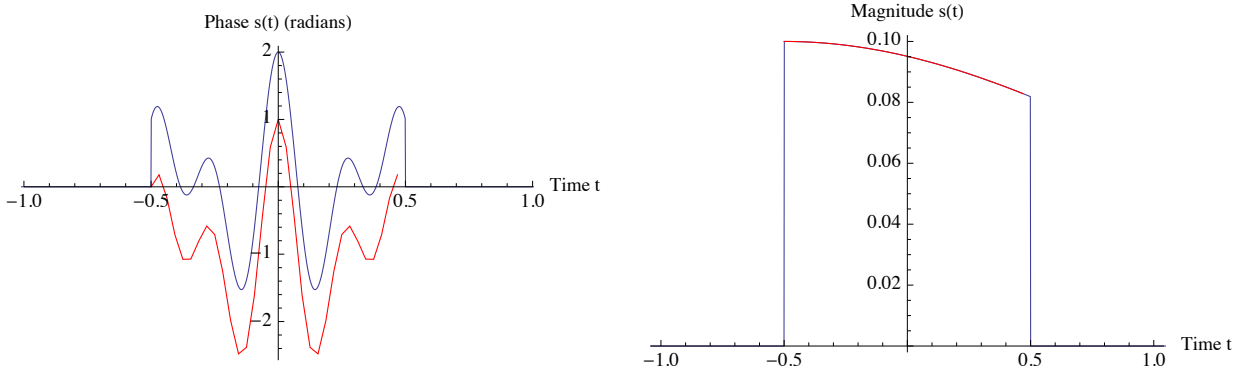
In[55]:=
```
Show[%5, %49,  PlotRange → {{-1, 1}, All}, AxesLabel → {"Time t", "Magnitude s(t)"}]
```

Out[55]=



In[56]:=
```
GraphicsGrid[{{%54, %55}}, ImageSize → Large]
```

Out[56]=



In[57]:=
```
Export["~/papers/frog/phaseAmp.eps", %56]
```

Out[57]= ~/papers/frog/phaseAmp.eps

Enhanced Bootstrap Algorithm with Least Squares Fit

We will demonstrate the enhanced bootstrap algorithm by applying it to the data with noise.
We will use the previous sinput[] but with the addition of noise.

Discrete Fractional Fourier Transform

An n point α-order discrete fractional Fourier transform (DFrFT) of a function s[t] is:

In[1]:=
$$\text{DFrFT}[s\_, n\_, \alpha\_, \Delta t\_, m\_] := \frac{\text{Exp}[I\,\alpha/2]}{\sqrt{I\,\text{Sin}[\alpha]}}\,\Delta t$$
$$\text{Sum}\left[s[k\,\Delta t]\,\text{Exp}\left[I\,\pi\left(\left(\frac{m\,\text{Sin}[\alpha]}{n\,\Delta t}\right)^2 + (k\,\Delta t)^2\right)\text{Cot}[\alpha] - I\,2\,\pi\,k\,m/n\right], \{k, -n/2, n/2-1\}\right];$$

Arguments of DFrFT[] are:

- s: function to be transformed
- n: number of points to be transformed. Must be power of 2
- α: order of the transform
- Δt: time step
- m: returns value of transform at m Δf

It is assumed that the time step gives the frequency step with the relationship $\Delta f\,\Delta t = \text{Sin}[\alpha]/n$.

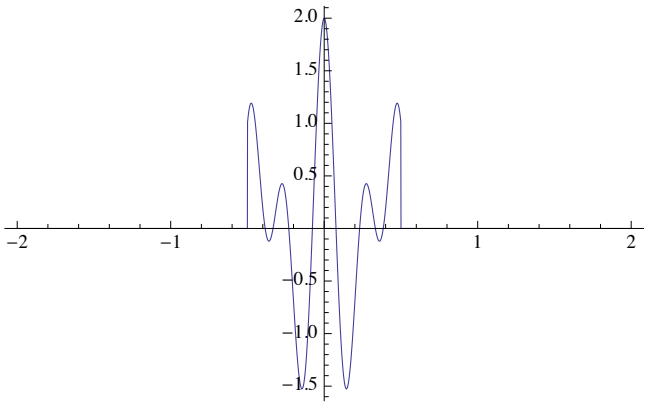
Example Input Function sinput[t]

The input function sinput[t] which we had used previously

In[2]:=
```
sinput[t_, Δτ_] :=
  η[t, Δτ] 0.1 Exp[-0.2 (t + 0.5)^2] Exp[I (Sin[5 π (t + 0.5)] + Cos[8 π (t + 0.5)])];
η[t_, Δτ_] := If[-Δτ ≤ t < Δτ, 1, 0];
```

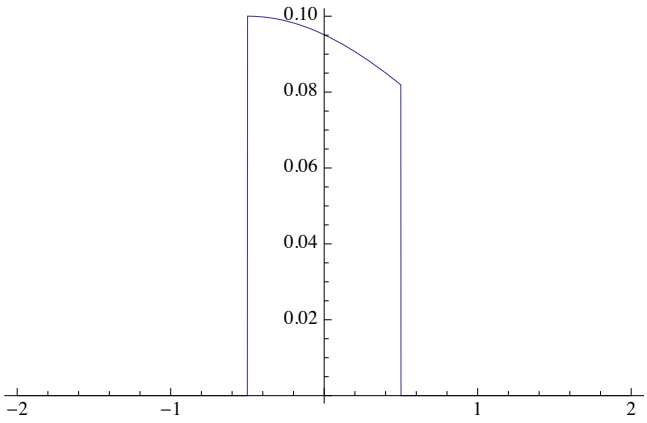
In[4]:=
```
Plot[Arg[sinput[t, 0.5]], {t, -2, 2}, PlotRange → All]
```

Out[4]=



In[5]:=
```
Plot[Abs[sinput[t, 0.5]], {t, -2, 2}]
```

Out[5]=



■ First measurement parameters with a small chunk

We select $\alpha = \pi/2$ and $\gamma = \pi/4$. The first chunk which we will use is size n1=16 in the time interval +/- 0.3. Therefore $\Delta t = 0.6/n1 = 0.6/16$. The signal outside this interval are zeroes which go from -nm/2 to -n1/2-1 and n1/2 to nm/2-1.

In[6]:=
```
nm = 256; (*total number of samples*)
n1 = 16;  (* number of samples where s is the real signal and not padding*)
α = π / 2;
γ =  π / 4;
Δτ = 0.6; (* total temporal interval +Δτ/2 to +Δτ/2*)
Δt = Δτ / n1;

Δfα = Sin[α] / (nm Δt);
Δfγ = Sin[γ] / (nm Δt);
```

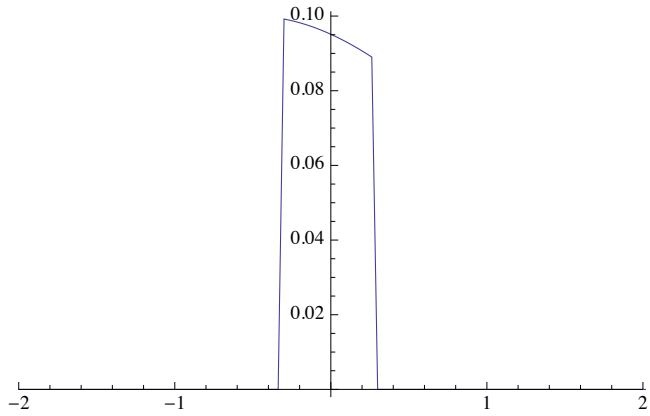
■ Create the new input sinput1[t]

In[13]:=
```
sinput1[t_] := sinput[t, Δτ / 2];
```

Check that the Δt sampling samples the input data well.

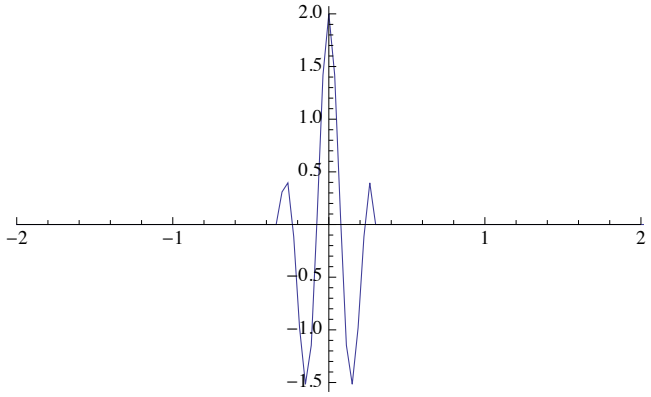
In[14]:=
```
ListPlot[Table[{i Δt, Abs[sinput1[i Δt]]}, {i, -nm / 2, nm / 2 - 1}],
 Joined → True, PlotRange → {{-2, 2}, All}]
```

Out[14]=



In[15]:=
```
ListPlot[Table[{i Δt, Arg[sinput1[i Δt]]}, {i, -nm / 2, nm / 2 - 1}],
 Joined → True, PlotRange → {{-2, 2}, All}]
```

Out[15]=



■ Create the I_α and I_γ with noise

We need nm points of intensity from $I_\alpha$and I_γ from sinput1[t].

In[16]:=
```
iα1 = Table[{m Δfα, Abs[ DFrFT[sinput1, nm, α, Δt, m]]^2}, {m, -nm / 2, nm / 2 - 1}] // N;
iγ1 = Table[{m Δfγ, Abs[ DFrFT[sinput1, nm, γ, Δt, m]]^2}, {m, -nm / 2, nm / 2 - 1}] // N;
```

Add in uniform noise that has a maximum relative range of +/- 0.01

In[18]:=
```
perr = 0.01;
```

In[19]:=
```
iα = Table[{iα1[[i, 1]], iα1[[i, 2]] (1 + Random[Real, {-perr , perr }])}, {i, Length[iα1]}];
iγ = Table[{iγ1[[i, 1]], iγ1[[i, 2]] (1 + Random[Real, {-perr , perr }])}, {i, Length[iγ1]}];
```

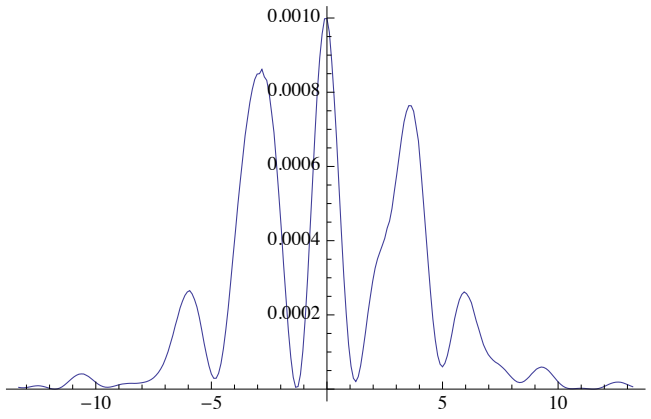
Make functions for the data sets

In[21]:=
```
Iα[f_] := iα[[Round[f / Δfα] + nm / 2 + 1]][[2]];
Iγ [f_] := iγ[[Round[f / Δfγ] + nm / 2 + 1]][[2]];
```

Plot them out

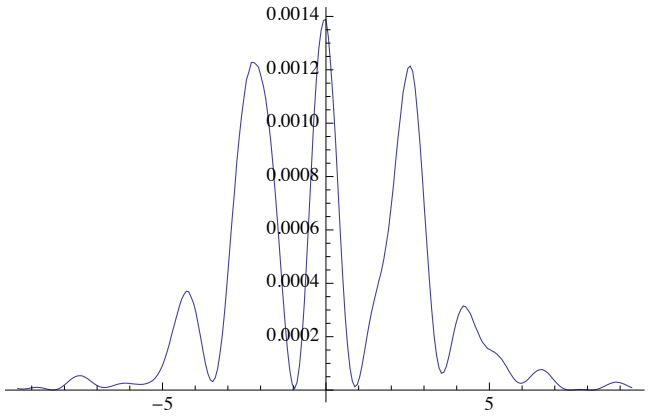
In[23]:=
```
ListPlot[iα, Joined → True, PlotRange → All]
```

Out[23]=



In[24]:=
```
ListPlot[iγ, Joined → True, PlotRange → All]
```

Out[24]=



$I_\alpha$and I_γ are the intensities which we measure.

Let's normalize iα and iγ so that their power is 1

```
(*iα = iα / (Sin[α] / (nm (Δt)^2) Sum[iα[[i,2]], {i, Length[iα]}]);
iγ = iγ / (Sin[γ] / (nm (Δt)^2) Sum[iγ[[i,2]], {i, Length[iγ]}]);*)
```

The Bootstrap Method

We define the equations for the bootstrap method defined as equation (19) in the paper.

In[25]:=
```
u[θ_, m_] := Exp[I 2 π (-n1 / 2) (n1 - m) (Δt)^2 Cot[θ]];
v[θ_, m_] := Exp[I 2 π (-n1 / 2 + m - 1) (n1 - m) (Δt)^2 Cot[θ]];
Cα[k_] :=
  Sin[α] / (nm Δt^2) Exp[-I π (k Δt)^2 Cot[α]] Sum[Iα[m Δfα] Exp[I 2 π k m / nm], {m, -nm / 2, nm / 2 - 1}];
Cγ[k_] := Sin[γ] / (nm Δt^2) Exp[-I π (k Δt)^2 Cot[γ]] Sum[Iγ[m Δfγ] Exp[I 2 π k m / nm], {m, -nm / 2, nm / 2 - 1}];
Σ[θ_, m_] := Sum[Conjugate[s[-n1 / 2 + j]]
    s[n1 / 2 - m + j] Exp[I 2 π (n1 - m) (-n1 / 2 + j) (Δt)^2 Cot[θ]], {j, 1, m - 2}];
d[m_] := Det[(u[α, m]  v[α, m]
              u[γ, m]  v[γ, m])];
```

Check that Cα[0] = Cγ[0]

In[31]:=
```
Cα0 = Cα[0]
```

Out[31]= 0.144402

In[32]:=
```
Cγ0 = Cγ[0]
```

Out[32]= 0.144497

Clearly they are different because of the noise in the measurement of Iα and Iγ.

Solving s[t]

First set s[t] = 0 for -n1/2 to -nm/2+1 and nm/2 to n1/2-1

In[33]:=
```
For[i = -nm / 2, i ≤ -n1 / 2 - 1, i++,
  s[i] = 0;
 ];
For[i = n1 / 2, i ≤ nm / 2 - 1, i++,
  s[i] = 0;
 ];
```

Without loss of generality, we will set $s[-N/2]=\sigma=1$ first and then use equation equation (25) to solve for $s[-n/2]$ because every s[n] is either multiplied or divided by $s[-n/2]$.

From equation (15),

In[35]:=
```
σ = 1;
```

In[36]:=
```
cα = Chop[ Cα[n1 - 1] Exp[I π n1 (n1 - 1) Δt^2 Cot[α]]]
```

Out[36]= 0.00885039 + 0.000728109 i

In[37]:=
```
cγ = Chop[Cγ[n1 - 1] Exp[I π n1 (n1 - 1) Δt^2 Cot[γ]]]
```

Out[37]= 0.00876743 + 0.000753852 i

Clearly $c\alpha$ is not equal to $c\gamma$ because of noise.

Iteration

In[38]:=
```
s[-n1 / 2] = σ
```

Out[38]= 1

Average $c\alpha$ and $c\gamma$ because they are supposed to be the same.

In[39]:=
```
s[n1 / 2 - 1] = (cα + cγ) / (2.0 σ)
```

Out[39]= 0.00880891 + 0.000740981 i

Also set $c\alpha$ to the average since it is used below

In[40]:=
```
cα = s[n1 / 2 - 1]
```

Out[40]= 0.00880891 + 0.000740981 i

From equation (16), we can solve for s[n/2-2] and s[-n/2+1]

In[41]:=
```
s[n1 / 2 - 2] = Chop[First[1/(σ d[2]) (v[γ, 2] - v[α, 2]).{{Cα[n1 - 2]}, {Cγ[n1 - 2]}}][[1]]]
```

Out[41]= 0.00771265 - 0.00355241 i

In[42]:=
```
s[-n1 / 2 + 1] =
 Chop[First[σ/Conjugate[cα d[2]] Conjugate[(-u[γ, 2] u[α, 2]).{{Cα[n1 - 2]}, {Cγ[n1 - 2]}}]][[1]]]
```

Out[42]= 1.0341 + 0.104501 i

Now we can continue the bootstrap process until every element is done

In[43]:=
```
For[m = 3, m ≤ n1 / 2, m++,
  s[n1 / 2 - m] =
   Chop[First[ 1/Conjugate[σ] d[m] (v[γ, m] - v[α, m]).( Cα[n1 - m] - Σ[α, m]
                                                         Cγ[n1 - m] - Σ[γ, m] )]][[1]];
  s[-n1 / 2 + m - 1] = Chop[First[ σ/Conjugate[cα d[m]]
       Conjugate[( -u[γ, m] u[α, m] ).( Cα[n1 - m] - Σ[α, m]
                                        Cγ[n1 - m] - Σ[γ, m] )]][[1]];
 ];
```

■ Calculate the magnitude s[-n/2]

We can calculate the magnitude of s[-n/2] using equation (25) of the paper. We will denote s[-n/2] = σ
σ2 MUST BE REAL so that s[-n/2] is REAL. See eq(25)

In[44]:=
```
s[-n1 / 2] = σ; (* placeholder for the above calculations *)
```

In[45]:=
```
sol = Chop[Solve[σ2 Sum[Abs[s[-n1 / 2 + m - 1]]^2, {m , 1, n1 / 2}] +
      1/σ2 Sum[Abs[s[n1 / 2 - m]]^2, {m, 1, n1 / 2}] == Cα[0], σ2]]
```

Out[45]=
```
{{σ2 → 0.00792997}, {σ2 → 0.0100484}}
```

σ2 must be real. So just select the real parts only for the calculations below

In[46]:=
```
σfirst = Re[σ2] /. First[sol]
```

Out[46]=
```
0.00792997
```

In[47]:=
```
σlast = Re[σ2] /. Last[sol]
```

Out[47]=
```
0.0100484
```

We select between the two solutions by using equation (26)

s0 = s[-n/2] G$\alpha\gamma$[n/2 + 1]

In[48]:=
```
firstsol1 =
  Sqrt[σ2] Chop[First[ 1/Conjugate[cα d[n1 / 2 + 1]] Conjugate[( -u[γ, n1 / 2 + 1] u[α, n1 / 2 + 1] ).
        ( Cα[n1 - (n1 / 2 + 1)] - Σ[α, n1 / 2 + 1]
          Cγ[n1 - (n1 / 2 + 1)] - Σ[γ, n1 / 2 + 1] )]][[1]]] /. σ2 → σfirst
```

Out[48]=
```
-0.0126781 + 0.0862064 i
```

The other part which is also s[0] from equation (26)

s0 = $\frac{1}{s[-n/2]}$ F$\alpha\gamma$[$n/2$]

```
In[49]:= firstsol2 =
          1
         ------- Chop[First[ 1 / d[n1 / 2] (v[γ, n1 / 2] - v[α, n1 / 2]).( Cα[n1 - n1 / 2] - Σ[α, n1 / 2] ) ][[
         Sqrt[σ2]                                                        Cγ[n1 - n1 / 2] - Σ[γ, n1 / 2] )
             1]]] /. σ2 → σfirst

Out[49]= -0.0120715 + 0.102767 i
```

■ Check the second solution

This is a quick check that the other solution is not correct

```
In[50]:= secondsol1 =
          Sqrt[σ2] Chop[First[ 1 / Conjugate[cα d[n1 / 2 + 1]] Conjugate[( -u[γ, n1 / 2 + 1] u[α, n1 / 2 + 1] ).
                  ( Cα[n1 - (n1 / 2 + 1)] - Σ[α, n1 / 2 + 1] ) ]]][[1]]] /. σ2 → σlast
                  ( Cγ[n1 - (n1 / 2 + 1)] - Σ[γ, n1 / 2 + 1] )

Out[50]= -0.0142714 + 0.0970402 i

In[51]:= secondsol2 =
          1
         ------- Chop[First[ 1 / d[n1 / 2] (v[γ, n1 / 2] - v[α, n1 / 2]).( Cα[n1 - n1 / 2] - Σ[α, n1 / 2] ) ][[
         Sqrt[σ2]                                                        Cγ[n1 - n1 / 2] - Σ[γ, n1 / 2] )
             1]]] /. σ2 → σlast

Out[51]= -0.0107238 + 0.0912938 i
```

Select between the two solutions by looking at their differences

```
In[52]:= If[Abs[firstsol1 - firstsol2] < Abs[secondsol1 - secondsol2],
           σ = Sqrt[σ2] /. σ2 → σfirst,
           σ = Sqrt[σ2] /. σ2 → σlast
         ]

Out[52]= 0.100242
```

■ Normalize s

Once we have σ, we can normalize all the s's

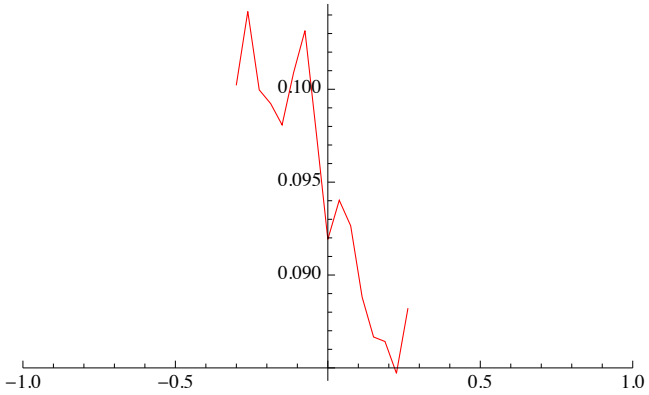
```
In[53]:= For[m = 1, m ≤ n1 / 2, m++,
           sn[-n1 / 2 + m - 1] = σ s[-n1 / 2 + m - 1];
         ];
         For[m = 1, m ≤ n1 / 2, m++,
           sn[n1 / 2 - m] = 1/σ s[n1 / 2 - m];
         ];

In[55]:= sMag = Table[{k Δt, Abs[sn[k]]}, {k, -n1 / 2, n1 / 2 - 1}];
         sArg = Table[{k Δt, Arg[sn[k]]}, {k, -n1 / 2, n1 / 2 - 1}];
         sRe = Table[{k Δt, Re[sn[k]]}, {k, -n1 / 2, n1 / 2 - 1}];
         sIm = Table[{k Δt, Im[sn[k]]}, {k, -n1 / 2, n1 / 2 - 1}];
```

■ Plots

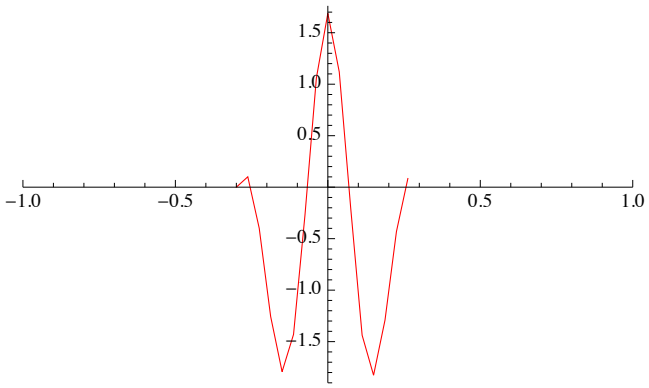
In[59]:=
```
ListPlot[sMag, Joined → True, PlotStyle → RGBColor[1, 0, 0], PlotRange → {{-1, 1}, All}]
```

Out[59]=



In[60]:=
```
ListPlot[sArg, Joined → True, PlotStyle → RGBColor[1, 0, 0], PlotRange → {{-1, 1}, All}]
```

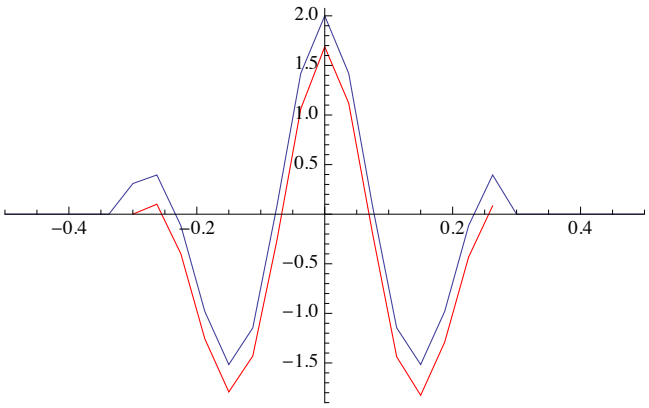
Out[60]=



■ Compare the result and the original chunk

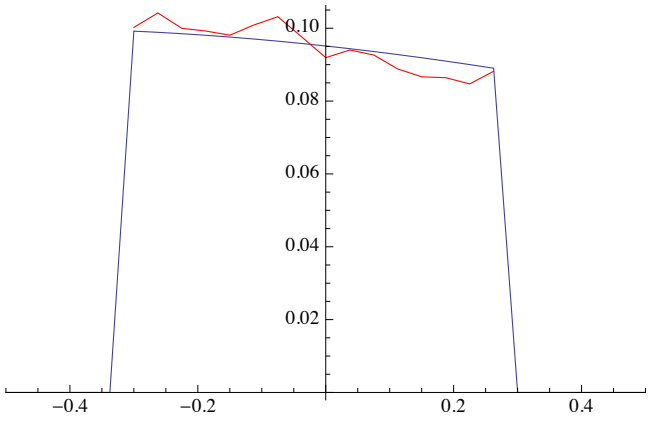
In[62]:=
```
Show[%15, %60, PlotRange → {{-0.5, 0.5}, All}]
```

Out[62]=



In[64]:=
```
Show[%14, %59, PlotRange → {{-0.5, 0.5}, All}]
```

Out[64]=



Extending the Bootstrap Algorithm

Using the sn[t] which we calculated, we will extend the solution for the next chunk.

The chunk has been increased by a factor of 2 from n1 to n2=2*n1.

In[65]:=
```
n2 = 2 n1;  (* number of samples where s is the real signal and not padding*)
Δt = 1.2 / n2;

Δfα = Sin[α] / (nm Δt);
Δfγ = Sin[γ] / (nm Δt);
```

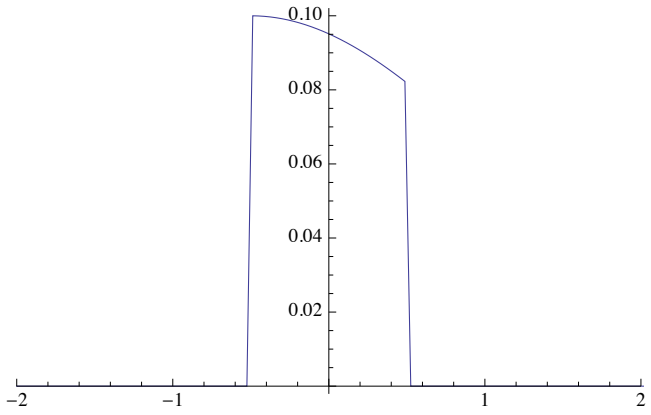
- **Create the new input sinput2[t]**

In[69]:=
```
sinput2[t_] := sinput[t, 0.5];
```

Check that the Δt sampling samples the input data well.

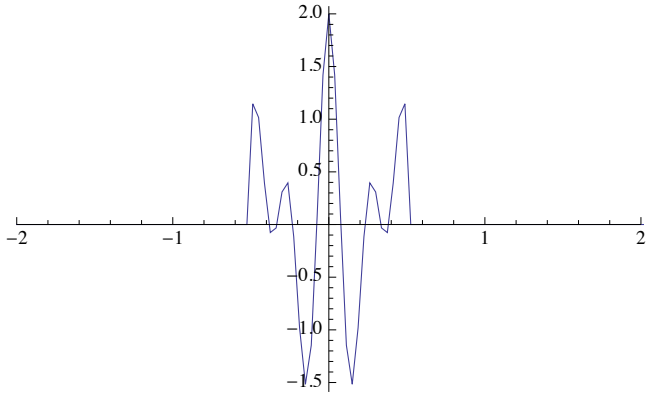
In[70]:=
```
ListPlot[Table[{i Δt, Abs[sinput2[i Δt]]}, {i, -nm / 2, nm / 2 - 1}],
 Joined → True, PlotRange → {{-2, 2}, All}]
```

Out[70]=



In[71]:=
```
ListPlot[Table[{i Δt, Arg[sinput2[i Δt]]}, {i, -nm / 2, nm / 2 - 1}],
 Joined → True, PlotRange → {{-2, 2}, All}]
```

Out[71]=



■ Create the I_α and I_γ with noise

We need nm points of intensity from $I_\alpha$and I_γ from sinput1[t].

In[72]:=
```
iα2 = Table[{m Δfα, Abs[ DFrFT[sinput2, nm, α, Δt, m]]^2}, {m, -nm / 2, nm / 2 - 1}] // N;
iγ2 = Table[{m Δfγ, Abs[ DFrFT[sinput2, nm, γ, Δt, m]]^2}, {m, -nm / 2, nm / 2 - 1}] // N;
```

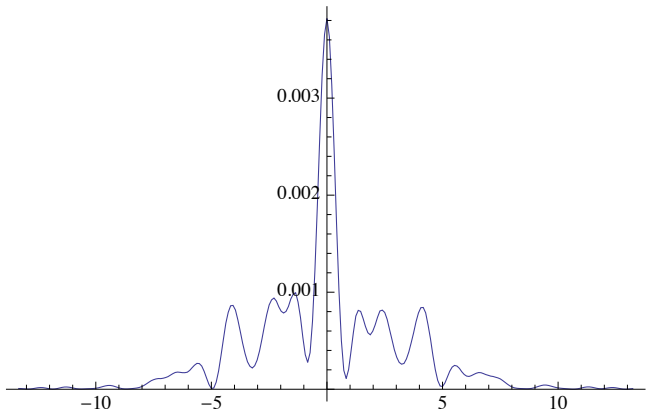
Add in the noise

In[74]:=
```
iα = Table[{iα2[[i, 1]], iα2[[i, 2]] (1 + Random[Real, {-perr , perr }])}, {i, Length[iα2]}];
iγ =  Table[{iγ2[[i, 1]], iγ2[[i, 2]] (1 + Random[Real, {-perr , perr }])}, {i, Length[iγ2]}];
```

Plot them out

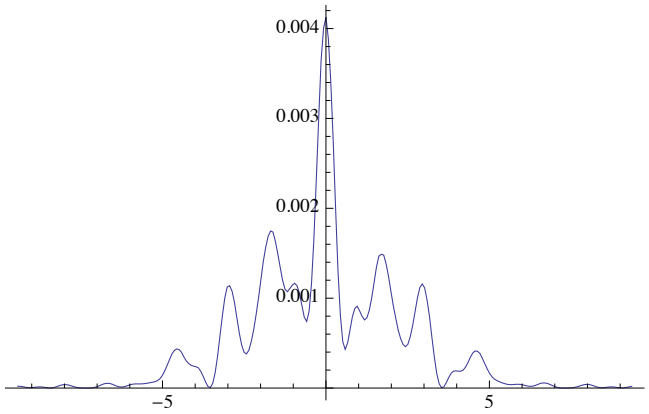
In[76]:=
```
ListPlot[iα, Joined → True, PlotRange → All]
```

Out[76]=



In[77]:=
```
ListPlot[iγ, Joined → True, PlotRange → All]
```

Out[77]=



$I_\alpha$and I_γ are the intensities which we measure.

Create the Matrix Vector Equation

Clear previous definition of σ

In[78]:=
```
Clear[σ];
Clear[σs];
```

Define the equations that are the entries of the matrix S

In[80]:=
```
σs[k_, l_, θ_] := Conjugate[sn[k]] Exp[i 2 π k l (Δt)^2 Cot[θ]];
σ[k_, l_, θ_] := sn[k] Exp[i 2 π (k - l) l (Δt)^2 Cot[θ]];
```

Filling in the $S_\theta$matrix shown in Eq. (29)

In[82]:=
```
Sα = Table[0, {i, n1 / 2 + 1}, {j, n1}];
Sγ = Table[0, {i, n1 / 2 + 1}, {j, n1}];
For[i = 1, i ≤ n1 / 2 + 1, i++,
  For[j = 1, j ≤ n1 / 2, j++,
    Sα[[i, j]] = σs[-n1 / 2 + (j - 1) + (i - 1), n1 - (i - 1), α];
    Sγ[[i, j]] = σs[-n1 / 2 + (j - 1) + (i - 1), n1 - (i - 1), γ];
   ];
 ];

For[i = 1, i ≤ n1 / 2 + 1, i++,
  For[j = 1, j ≤ n1 / 2, j++,
    Sα[[i, j + n1 / 2]] = σ[(j - 1) - (i - 1), n1 - (i - 1), α];
    Sγ[[i, j + n1 / 2]] = σ[(j - 1) - (i - 1), n1 - (i - 1), γ];
   ];
 ];
```

The $C_\theta$vector showin in Eq. (31)

In[86]:=
```
CCα = Table[0, {i, n1 / 2 + 1}, {j, 1, 1}];
CCγ = Table[0, {i, n1 / 2 + 1}, {j, 1, 1}];
For[i = 1, i ≤ n1 / 2 + 1, i++,
  CCα[[i, 1]] = Cα[n1 - (i - 1)] - Sum[Conjugate[sn[-n1/2 + j - 1]]
      sn[n1/2 + j - i] Exp[i 2 π (-n1/2 + j - 1) (n1 - (i - 1)) (Δt)^2 Cot[α]], {j, 1, i - 1}];
  CCγ[[i, 1]] = Cγ[n1 - (i - 1)] - Sum[Conjugate[sn[-n1/2 + j - 1]] sn[n1/2 + j - i]
      Exp[i 2 π (-n1/2 + j - 1) (n1 - (i - 1)) (Δt)^2 Cot[γ]], {j, 1, i - 1}];
 ];
```

Create the matrix equation which we solve for the unknown s[] entries Eq. (28)

In[89]:=
```
S = Table[0, {i, n1 + 2}, {j, n1}];
CC = Table[0, {i, n1 + 2}, {j, 1, 1}];
For[i = 1, i ≤ n1 / 2 + 1, i++,
  For[j = 1, j ≤ n1, j++,
    S[[i, j]] = Sα[[i, j]];
   ];
 ];
For[i = 1, i ≤ n1 / 2 + 1, i++,
  For[j = 1, j ≤ n1, j++,
    S[[i + (n1 / 2 + 1), j]] = Sγ[[i, j]];
   ];
 ];
For[i = 1, i ≤ n1 / 2 + 1, i++,
  CC[[i, 1]] = CCα[[i, 1]];
 ];
For[i = 1, i ≤ n1 / 2 + 1, i++,
  CC[[i + (n1 / 2 + 1), 1]] = CCγ[[i, 1]];
 ];
```

The number of equations is greater than the number of variables. Therefore, we'll have to use a least squares method to solve for s[]

In[95]:=
```
sol = LeastSquares[S, CC];
```

Now, get the answers to the unknowns using Eq (30)

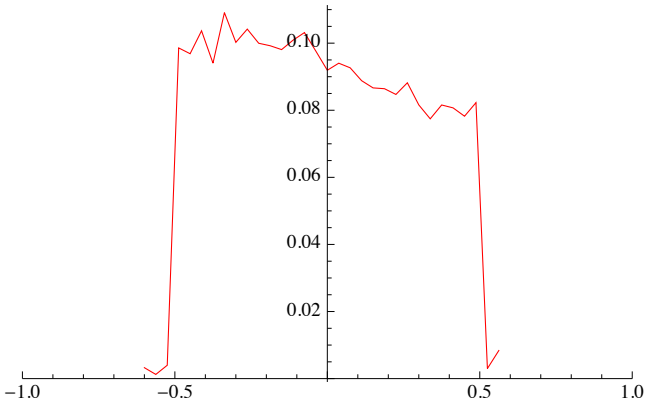
In[96]:=
```
For[i = 1, i ≤ n1 / 2, i++,
  sn[n1 / 2 + (i - 1)] = sol[[i, 1]];
 ];
For[i = 1, i ≤ n1 / 2, i++,
  sn[-n1 + (i - 1)] = Conjugate[sol[[n1 / 2 + i, 1]]];
 ];
```

- **Plots**

In[98]:=
```
sMag = Table[{k Δt, Abs[sn[k]]}, {k, -n2 / 2, n2 / 2 - 1}];
sArg = Table[{k Δt, Arg[sn[k]]}, {k, -n2 / 2, n2 / 2 - 1}];
sRe = Table[{k Δt, Re[sn[k]]}, {k, -n2 / 2, n2 / 2 - 1}];
sIm = Table[{k Δt, Im[sn[k]]}, {k, -n2 / 2, n2 / 2 - 1}];
```

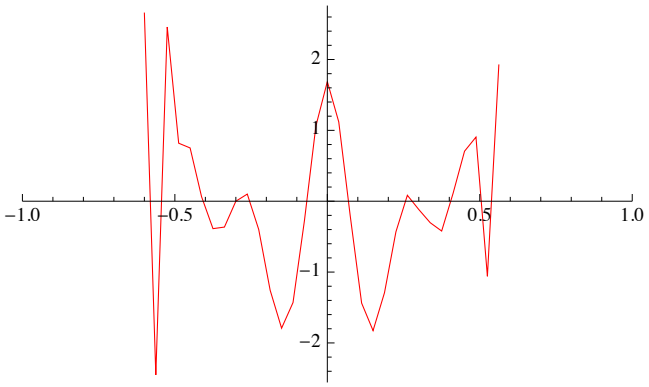
In[102]:=
```
ListPlot[sMag, Joined → True, PlotStyle → RGBColor[1, 0, 0], PlotRange → {{-1, 1}, All}]
```

Out[102]=



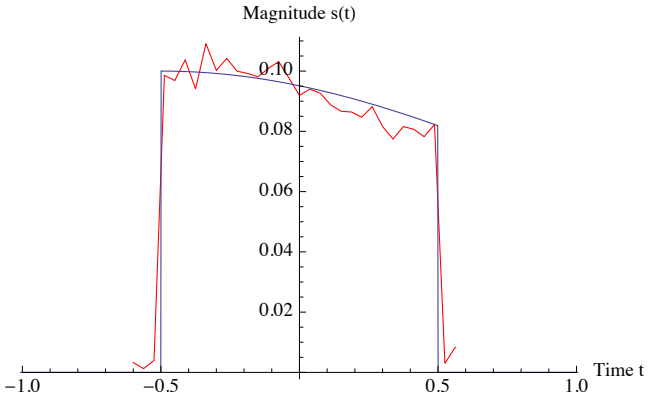
In[103]:=
```
ListPlot[sArg, Joined → True, PlotStyle → RGBColor[1, 0, 0], PlotRange → {{-1, 1}, All}]
```

Out[103]=



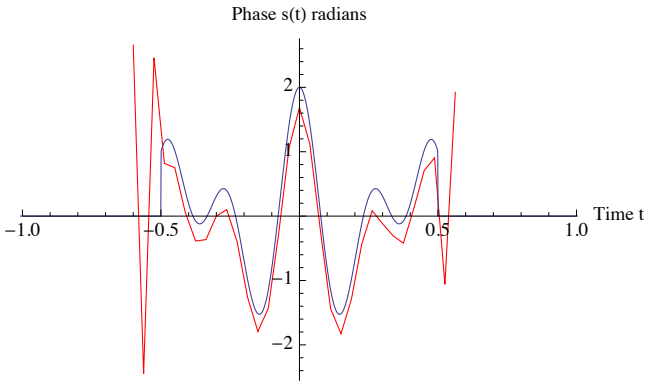
In[106]:=
```
Show[%102, %5, AxesLabel → {"Time t", "Magnitude s(t)"}]
```

Out[106]=



In[107]:=
```
Show[%103, %4, AxesLabel → {"Time t", "Phase s(t) radians"}]
```

Out[107]=



APPENDIX II: OPTICS ANALYSIS

In this appendix, we will analyse of the optics setup shown in the yellow panel of Figure 4.

Transmission grating diffraction grating

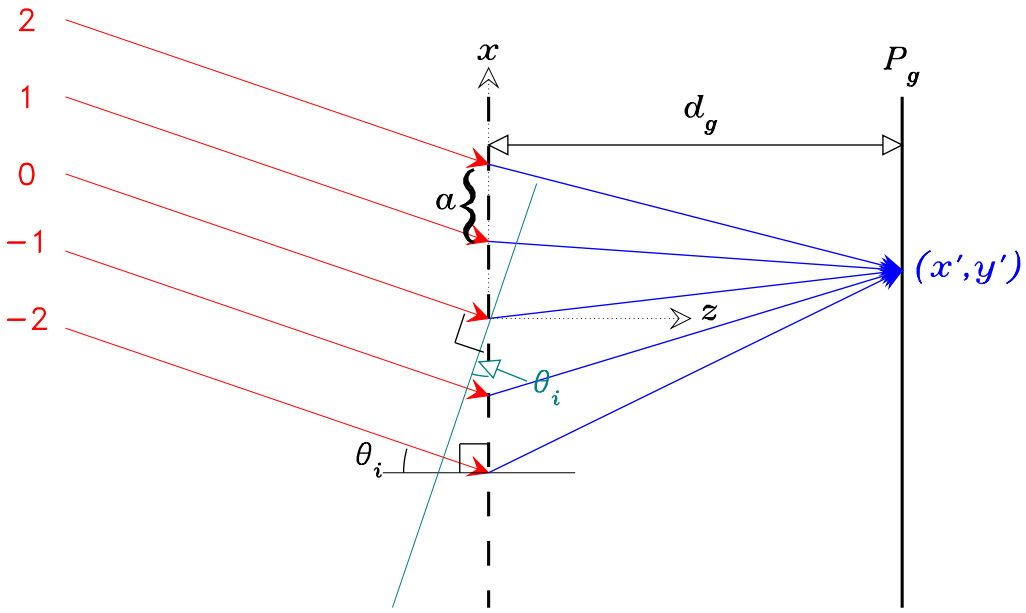


Figure 19 A simple transmission diffraction grating spectrometer.

A simple transmission diffraction grating spectrometer is shown in Figure 19. By tilting the incident beam so that it intersects the diffraction grating at an angle, the diffracted light will contain information about its longitudinal distribution. In this presentation we will apply scalar diffraction theory for the analysis of the diffraction pattern and so there cannot be any explicit time dependence.

In general, the transmission diffraction grating transmission function is the convolution

of a sum of δ-functions which are separated by a period a with the width of each slit b.[e]

$$t_D(x) = \left(\int_{-\infty}^{\infty} dx' \left[\sum_{p=-\infty}^{\infty} \delta(x' - pa) \right] W(x - x') \right) P(x) \tag{42}$$

where the W is described by

$$W(x;\, b) = \begin{cases} 1 & \text{if } |x| < b/2 < a/2 \\ 0 & \text{otherwise} \end{cases} \tag{43}$$

We have also added an aperture function $P(x)$ to describe the finite size of the diffraction grating.

$$P(x) = \begin{cases} 1 & \text{if } |x| < Ma/2 \\ 0 & \text{otherwise} \end{cases} \tag{44}$$

where M is the total number slits.

$M \to \infty$ diffraction grating model

For the analytic calculations, we will simplify diffraction grating model by making it infinite in size, i.e. $M \to \infty$. Therefore, the transmission function is

$$\left.\begin{aligned} t'_D(x) &= \int_{-\infty}^{\infty} dx' \sum_{p=-\infty}^{\infty} \delta(x' - pa) W(x - x';\, b) \\ &= \sum_{p=-\infty}^{\infty} W(x - pa;\, b) \end{aligned}\right\} \tag{45}$$

Diffraction analysis

We will first consider the rays to the left of the diffraction grating and assume that it is a plane wave with a propagation vector $\boldsymbol{k} = k(\sin\theta_i, \cos\theta_i) = \frac{2\pi}{\lambda}(\sin\theta_i, \cos\theta_i)$. Then at

[e] Note: Commercial diffraction gratings have a sinusoidal structure or a sawtooth structure. Sinusoidal gratings have been analysed by Goodman.[7]

$\boldsymbol{r} = (x, 0^-)$, we have

$$\left.\begin{aligned} u^g(x, 0;\ \theta_i;\ \phi) &= s(\phi) e^{-(x\cos\theta_i)^2/w^2} e^{i\boldsymbol{k}\cdot\boldsymbol{r}} \\ &= s(\phi) e^{-(x\cos\theta_i)^2/w^2} e^{ikx\sin\theta_i} \end{aligned}\right\} \tag{46}$$

where $s \in \mathbb{C}$ is the "temporal" description of the laser pulse in its propagation direction which we have reparametrised in terms of phase (Note: we have to be careful in the definition of u^g because u^g satisfies the Helmholtz equation which means it is *time independent*), and $e^{-(x\cos\theta_i)^2/w^2}$ is the Gaussian spatial profile of the laser beam in its transverse direction projected onto the z-axis, and w is the radius of the beam where the field amplitude has fallen by $1/e$.

Since, our calculation cannot have any time dependence, let us consider a snapshot of the fields at the grating when $\phi = 0$ for ray 0 which we will use as the reference ray and so

$$\text{ray } 0 = u^g(0, 0;\ \theta_i;\ 0) \tag{47}$$

For ray 1, we have

$$\text{ray } 1 = u^g\left(a, 0;\ \theta_i;\ ka\sin\theta_i\right) \tag{48}$$

because it arrives earlier than ray 0, and so at any later sampling time, we will see a later part of the pulse compared to ray 0. See Figure (c).

For ray (-1), we have

$$\text{ray } (-1) = u^g\left(-a, 0;\ \theta_i;\ -ka\sin\theta_i\right) \tag{49}$$

because it arrives later than ray 0 and so at any later sampling time, we will see an earlier part of the pulse compared to ray 0.

And in general

$$\text{ray } p = u^g\left(pa, 0;\ \theta_i;\ pka\sin\theta_i\right) \tag{50}$$

Fresnel approximation confined to xz-plane

In general, the Fresnel approximation for diffraction is

$$u(x,y,z) = \frac{e^{ikz}e^{ik(x^2+y^2)/2z}}{i\lambda z}\int_{-\infty}^{\infty}dx'\int_{-\infty}^{\infty}dy'\ \left[u(x,y,0)e^{ik(x'^2+y'^2)/2z}\right]e^{-ik(xx'+yy')/z} \tag{51}$$

We will confine our analysis only to the xz-plane. To do this, we will set $y=0$ in (51) and define $u(x,0,z)\equiv u(x,z)$ to get

$$u(x,z) = \frac{e^{ikz}e^{ikx^2/2z}}{i\lambda z}\int_{-\infty}^{\infty}dx'\ u(x,0)e^{ikx'^2/2z}e^{-ikxx'/z}\int_{-\infty}^{\infty}dy'\ e^{iky'^2/2z} \tag{52}$$

The integral $\int_{-\infty}^{\infty}dy'\ e^{iky'^2/2z} = -i\sqrt{\lambda z/i}$ and when we substitute it into (52), we get the Fresnel approximation which we desire

$$u(x,z) = -\frac{e^{ikz}e^{ikx^2/2z}}{\sqrt{i\lambda z}}\int_{-\infty}^{\infty}dx'\ u(x,0)e^{ikx'^2/2z}e^{-ikxx'/z} \tag{53}$$

Fresnel diffraction pattern

If we describe the diffraction pattern on the the plane P_g by the Fresnel approximation (53), then it is

$$\begin{aligned}
u^{(1-)}(x,d_g;\ \theta_i) &= -\frac{e^{ikd_g}e^{ikx^2/2d_g}}{\sqrt{i\lambda d_g}}\times \\
&\qquad \int_{-\infty}^{\infty}dx'\left[u^g(x',0;\ \theta_i;\ kx'\sin\theta_i)t'_D(x')e^{ikx'^2/2d_g}\right]e^{-ikxx'/d_g} \\
&= -\frac{e^{ikd_g}e^{ikx^2/2d_g}}{\sqrt{i\lambda d_g}}\times \\
&\qquad \sum_{p=-\infty}^{\infty}\int_{-\infty}^{\infty}dx'\ \left[\left(u^g(x',0;\ \theta_i;\ kx'\sin\theta_i)e^{ikx'^2/2d_g}\right)W(x'-pa;\ b)\right]\times \\
&\qquad\qquad e^{-ikxx'/d_g}
\end{aligned} \tag{54}$$

When we only keep the non-zero parts of the integral we get the Fresnel diffraction pattern for a transmission diffraction grating

$$u^{(1-)}(x, d_g;\ \theta_i) = -\frac{e^{ikd_g}e^{ikx^2/2d_g}}{\sqrt{i\lambda d_g}}\times \sum_{p=-\infty}^{\infty}\int_{-b/2}^{b/2} dx'\ \left[u^g\Big(x'+pa, 0;\ \theta_i\ k(x'+pa)\sin\theta_i\Big)e^{ik(x'+pa)^2/2d_g}\right]\times e^{-ikx(x'+pa)/d_g} \tag{55}$$

In general, (55) is not integrable for arbitrary u^g and even if it is, the results will clearly not be very illuminating. Therefore, in the section *Extremely narrow slits*, we will make the slits very small so that we can simplify (55). But first, we will check out (55) for the special case of $u^g = 1$.

Check

We will check that (55) gives us the well known solution for a diffraction grating for the case when $u^g = 1$. Substituting this into (55) we get

$$u^{(1-)}(x, d_g;\ 0) = -\frac{e^{ikd_g}e^{ikx^2/2d_g}}{\sqrt{i\lambda d_g}}\sum_{p=-\infty}^{\infty}\int_{-b/2}^{b/2} dx'\ e^{ik(x'+pa)^2/2d_g}e^{-ikx(x'+pa)/d_g} \tag{56}$$

In the Fraunhofer approximation $k(x'+pa)^2/2d_g \approx 0$ and so

$$\left.\begin{aligned} u^{(1-)}(x, d_g;\ 0) &= -\frac{e^{ikd_g}e^{ikx^2/2d_g}}{\sqrt{i\lambda d_g}}\sum_{p=-\infty}^{\infty}\int_{-b/2}^{b/2} dx'\ e^{-ikx(x'+pa)/d_g} \\ &= -b\frac{e^{ikd_g}e^{ikx^2/2d_g}}{\sqrt{i\lambda d_g}}\left(\frac{\sin kbx/2d_g}{kbx/2d_g}\right)\sum_{p=-\infty}^{\infty} e^{-i2\pi pax/\lambda d_g} \\ &= -b\frac{e^{ikd_g}e^{ikx^2/2d_g}}{a}\sqrt{\frac{\lambda d_g}{i}}\left(\frac{\sin kbx/2d_g}{kbx/2d_g}\right)\sum_{p=-\infty}^{\infty}\delta\left[x - p\frac{\lambda d_g}{a}\right] \end{aligned}\right\} \tag{57}$$

where the last line comes from the Poisson sum formula for complex exponentials

$$\sum_{p=-\infty}^{\infty} e^{\pm i2\pi pfT} = \frac{1}{T}\sum_{p=-\infty}^{\infty} \delta(f - p/T) \tag{58}$$

Hence, we have obtained the well-known solution of a diffraction grating in (57). The width of the slit b gives the sinc() modulation of the spectrum which are an infinite number of δ-functions separated by $\lambda d_g/a$.

Extremely narrow slits

The Fresnel solution for a finite sized slit is very complicated. We can simplify it by making the slits extremely narrow, i.e. $b/\lambda \ll 1$ so that $u^g(x)e^{ikx^2/2d_g}$ is constant in the slit. We can simplify (55) to

$$u^{(1-)}(x, d_g;\ \theta_i) = -b\frac{e^{ikd_g}e^{ikx^2/2d_g}}{\sqrt{i\lambda d_g}} \sum_{p=-\infty}^{\infty} u^g(pa, 0;\ \theta_i;\ pka\sin\theta_i)e^{ik(pa)^2/2d_g}e^{-ipkax/d_g} \tag{59}$$

When we substitute (46) into (59), we have

$$\begin{aligned} u^{(1-)}(x, d_g;\ \theta_i) = -b\frac{e^{ikd_g}e^{ikx^2/2d_g}}{\sqrt{i\lambda d_g}}\times \\ \sum_{p=-\infty}^{\infty} \left[s\left(pka\sin\theta_i\right) e^{ik(pa)^2/2d_g}e^{ipka\sin\theta_i}\right] e^{-(pa\cos\theta_i)^2/w^2}e^{-ipkax/d_g} \end{aligned} \tag{60}$$

Notice the quadratic phase term $e^{ik(pa)^2/2d_g}e^{ipka\sin\theta_i}$ above. This means that the phase at every point $pka\sin\theta_i$ is increased by $(k(pa)^2/2d_g + pka\sin\theta_i)$. Since all the parameters in this phase term are known, we can correct the for it in the phase calculated by the bootstrap algorithm.

Let us define a new variable

$$\left.\begin{aligned} \bar{s}(pka\sin\theta_i) &\equiv s(pka\sin\theta_i)e^{ik(pa)^2/2d_g}e^{ipka\sin\theta_i} \\ \text{or} \qquad \bar{s}(\phi) &= s(\phi)e^{i\phi^2/(2kd_g\sin^2\theta_i)}e^{i\phi} \end{aligned}\right\} \tag{61}$$

because we can always reconstruct s from $\bar{s}$ and so (60) becomes

$$u^{(1-)}(x, d_g;\ \theta_i) = -b\frac{e^{ikd_g}e^{ikx^2/2d_g}}{\sqrt{i\lambda d_g}} \sum_{p=-\infty}^{\infty} \bar{s}\,(pka\sin\theta_i)\, e^{-(pa\cos\theta_i)^2/w^2} e^{-ipkax/d_g} \tag{62}$$

Note: Although the reconstruction of $s(\phi)$ from $\bar{s}(\phi)$ is trivial, there are some technicalities which must be understood for the reconstruction to work. *Appendix III* details the solution of these technicalities.

Case I: $\theta_i \ = \ \frac{\pi}{2}$

We want to show that (62) is the Fourier transform of $\bar{s}(\varphi')$. Let us consider the special case when the incident angle $\theta_i = \pi/2$ then

$$u^{(1-)}(x, d_g; \tfrac{1}{2}\pi) = -b\frac{e^{ikd_g}e^{ikx^2/2d_g}}{\sqrt{i\lambda d_g}} \sum_{p=-\infty}^{\infty} \bar{s}\,(pka)\, e^{-ipkax/d_g} \tag{63}$$

If we only consider the sum on the rhs of (63), and identify $ka = \mu\Delta\varphi'$ and $\mu x/d_g/2\pi = \varphi$ where $\mu \in \mathbb{R}^+$ is an arbitrary constant (see *Example A.II.1*) then

$$\bar{S}(\varphi) = \sum_{p=-\infty}^{\infty} \bar{s}(pka)e^{-ipka(x/d_g-1)} = \sum_{p=-\infty}^{\infty} \bar{s}(p\Delta\varphi')e^{-i2\pi(p\Delta\varphi')\varphi} \tag{64}$$

It is clear that (64) is the Fourier transform of $\bar{s}(\varphi')$.

We can transform (64) into a DFT by first defining a new variable

$$\Delta\varphi = \frac{\mu}{2\pi}\left(\frac{\Delta x}{d_g}\right) \tag{65}$$

and demanding that

$$\Delta\varphi\ \Delta\varphi' = \frac{\mu}{2\pi}\left(\frac{\Delta x}{d_g}\right) \times \frac{ka}{\mu} = 1/N \tag{66}$$

where N is the number of points used to sample $\bar{s}(\varphi')$ which we have assumed to be a finite pulse, thus

$$\Delta x = \left(\frac{2\pi}{Nka}\right) d_g \tag{67}$$

Therefore, the DFT of $\bar{s}(\varphi')$ is

$$\bar{S}(q\Delta\varphi) = \sum_{p=-N/2}^{N/2-1} \bar{s}(p\Delta\varphi')e^{-i2\pi pq/N} \tag{68}$$

and $u^{(1-)}(x, d_g;\ \frac{1}{2}\pi)$ when discretised is

$$u^{(1-)}(q\Delta x, d_g;\ \tfrac{1}{2}\pi) = -b\frac{e^{ikd_g}e^{ik(q\Delta x)^2/2d_g}}{\sqrt{i\lambda d_g}}\bar{S}(q\Delta\varphi) \tag{69}$$

Example A.II.1

We will verify that the DFT of $\bar{s}(\phi')$ for a rectangle function

$$\text{rect}(\phi'/\phi'_{\text{len}}) = H(\phi' + \phi'_{\text{len}}/2) - H(\phi' - \phi'_{\text{len}}/2) \tag{70}$$

where $H(\phi')$ is the Heaviside operator, is the well known normalised sinc function

$$\text{FT}\Big[\text{rect}(\phi'/\phi'_{\text{len}})\Big](\phi) = |\phi'_{\text{len}}|\frac{\sin\pi\phi'_{\text{len}}\phi}{\pi\phi'_{\text{len}}\phi} \tag{71}$$

using (68).

The parameters used for our example are shown in Table AII.2. For a rectangle pulse which is "on" for $t_p = 100$ fs or $\phi'_{\text{len}} = 75\pi$ radians and $\mu = 4$ so as to reduce the sampling period to $\Delta\phi' = ka/4$, we can plot $S(\phi)/S(0)$ and $\sin\pi\phi'_{\text{len}}\phi/\pi\phi'_{\text{len}}\phi$ and superimpose them in Figure 20.

Case II: $\theta_i = \frac{\pi}{2} - \delta\theta_i$

In general, the incident angle θ_i is at some value smaller than $\pi/2$. For the success of the optical setup, θ_i must be as close to $\pi/2$ as possible, i.e.

$$\theta_i = \pi/2 - \delta\theta_i \tag{72}$$

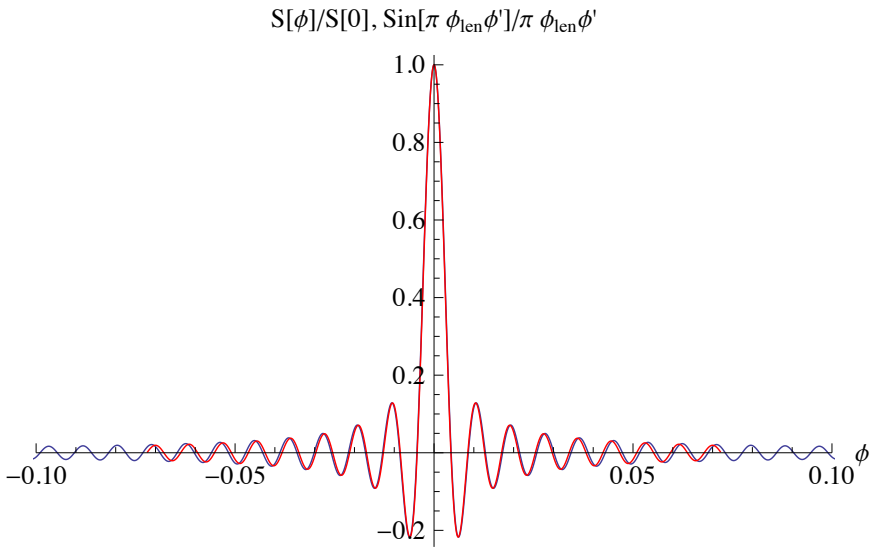


Figure 20 We superimpose $S(\phi)/S(0)$ (blue trace) and $\frac{\sin \pi \phi_{\rm len} \phi'}{\pi \phi_{\rm len} \phi'}$ (red trace) and see that the two results are very close. The phase difference between these two functions gets smaller as the sampling period $\Delta\phi' \to 0$.

where $\delta\theta_i \ll 1$. Therefore,

$$\left.\begin{aligned} \cos\theta_i &= \delta\theta_i + \mathrm{O}(\delta\theta^3) \\ \sin\theta_i &= 1 + \mathrm{O}(\delta\theta^2) \end{aligned}\right\} \tag{73}$$

When we substitute the above into (62), we have

$$u^{(1-)}(x, d_g;\ \tfrac{1}{2}\pi - \delta\theta_i) = -b\frac{e^{ikd_g} e^{ikx^2/2d_g}}{\sqrt{i\lambda d_g}} \sum_{p=-\infty}^{\infty} \bar{s}\,(pka)\, e^{-(pa\delta\theta_i)^2/w^2} e^{-ipkax/d_g} \tag{74}$$

and so the sum on the rhs of (74) is still the Fourier transform of $s(\phi')$ *without any coupling to the transverse plane* as long as

(i) the sum is not to infinity, i.e. $s(\pm p_{\rm max}\Delta\phi') = 0$ for $|p| > p_{\rm max}$.

(ii) $(p_{\rm max} a\delta\theta_i/w)^2 \ll 1$.

If the above conditions are met then

$$u^{(1-)}(x, d_g;\ \tfrac{1}{2}\pi - \delta\theta_i) \approx -b\frac{e^{ikd_g} e^{ikx^2/2d_g}}{\sqrt{i\lambda d_g}} \sum_{p=-p_{\rm max}/2}^{p_{\rm max}/2-1} \bar{s}\,(pka)\, e^{-ipkax/d_g} \tag{75}$$

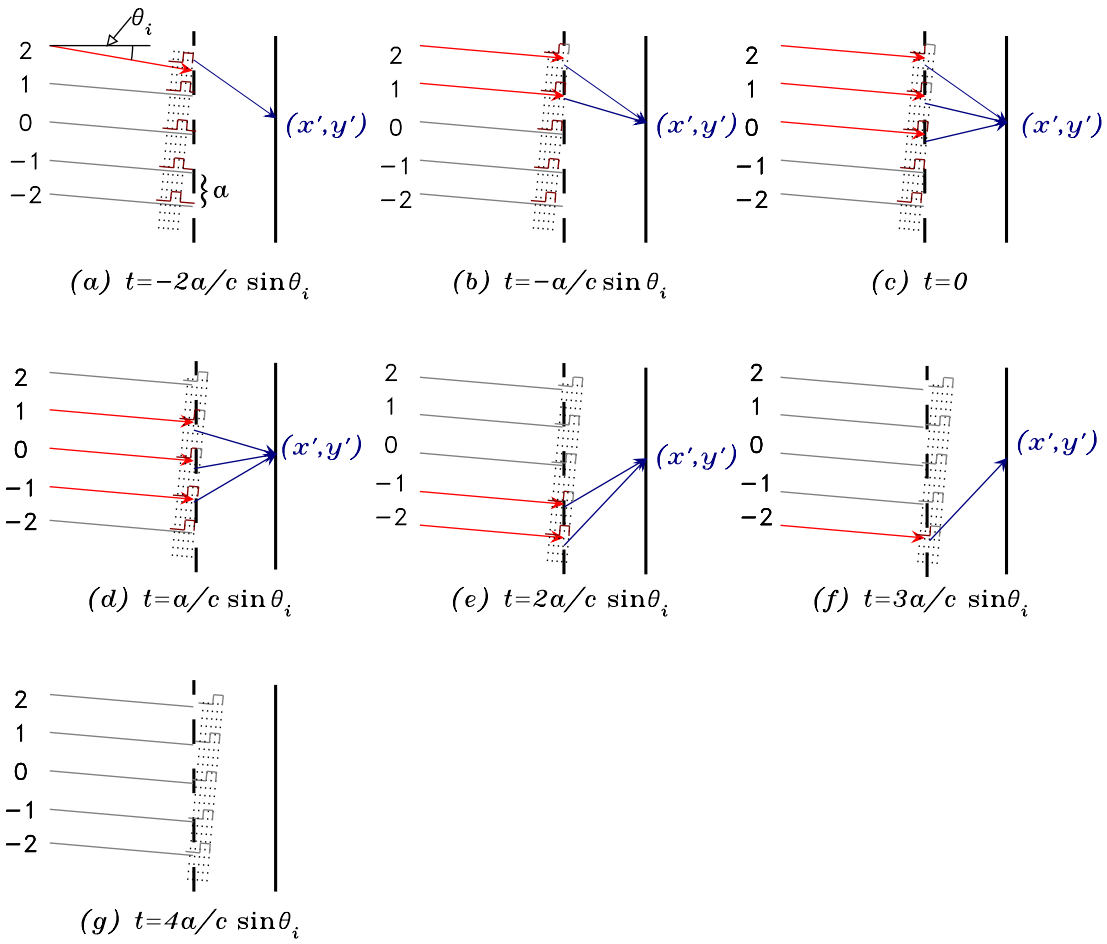


Figure 21 We have made the incident angle θ_i very shallow in this figure in order to illustrate the temporal dependence of the Fourier spectrum. The leading edge of the pulse represented by ray 2 intersects the diffraction grating first. As time progresses, the entire pulse will eventually leave the grating.

and the previous arguments of the section *Case I:* $\theta_i = \frac{1}{2}\pi$ apply. We will not consider the case when $\delta\theta_i$ is large.

Sampling frames

In the previous sections, we have considered only the static situation of one snapshot.

However, we have a dynamical situation where each snapshot shows a sampled pulse which moves across the grating. See Figures 21 and 22. Because of this motion, we require the number of frames which contain the entire pulse to be much greater than the frames which only contain a partial part of the pulse. If the full frames are dominant then the only concern that we have is the phase shift between each frame, i.e.

$$\begin{aligned}
\text{Frame } 1 &= u(x) \\
\text{Frame } 2 &= e^{i\psi}u(x) \\
\text{Frame } 3 &= e^{i2\psi}u(x) \\
\vdots &= \vdots
\end{aligned}$$

But even in this case the CCD camera is only sensitive to the magnitude of each frame and so the integral is simply

$$\begin{aligned}
\text{CCD integral} &= u(x)u^*(x) + \left[e^{i\psi}u(x)\right]\left[e^{i\psi}u(x)\right]^* + \left[e^{i2\psi}u(x)\right]\left[e^{i2\psi}u(x)\right]^* + \dots \\
&= \sum_{N_{\text{frames}}} |u(x)|^2
\end{aligned}$$

which means that any phase shift ψ arising from each frame is irrelevant.

The final part of this section is to calculate the optimum θ_i and number of samples n per frame for a pulse width t_p.

The angle θ_i must not be so large that the pulse from the first incident ray completely leaves the slit before the subsequent rays enter their slits. If the sampling time is Δt, we want

$$\Delta t = \frac{a}{c}\sin\theta_i \tag{76}$$

If we want n samples of the pulse, we must have

$$\Delta t = t_p/n \qquad \Rightarrow \qquad \frac{a}{c}\sin\theta_i = t_p/n \tag{77}$$

Therefore, the maximum incident angle allowed is

$$\theta_i^{\max} = \sin^{-1}\left(\frac{ct_p}{na}\right) \qquad \text{where } 0 \le \theta_i^{\max} \le \pi/2 \tag{78}$$

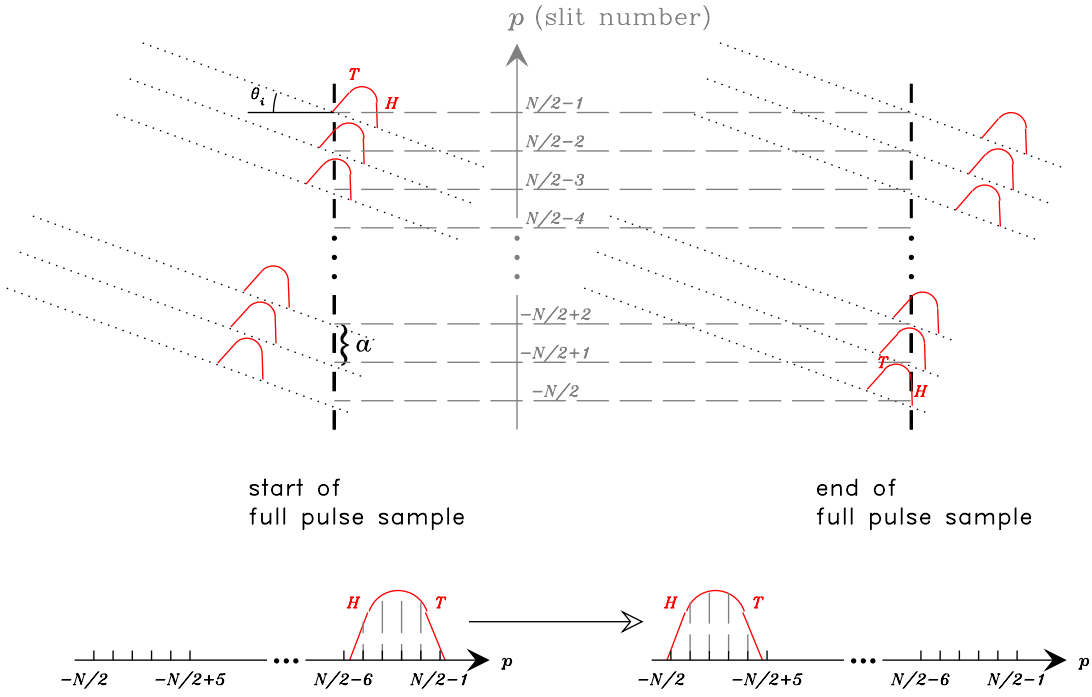


Figure 22 The sampling frames of the input pulse. As the pulse moves across the diffraction grating, some of the slits sample the pulse.

We need to ensure that the number of full frames exceed the partial frames which depends on the transverse width of the laser beam. If the transverse $1/e$ width is w, then the projected width w_{proj} onto the grid is

$$\cos\theta_i = 2w/w_{\text{proj}} \tag{79}$$

(Note: the "2" in the above equation is because w is the "radius" of the laser beam) which means that the total number of full frames are

$$N_{\text{frames}} = w_{\text{proj}}/na \qquad \Rightarrow \qquad N_{\text{frames}} = \frac{2w}{na\cos\theta_i} \tag{80}$$

Example A.II.2

We can calculate the number of sample points n per frame using (78) when we use the

numbers in Table AII.2 and set $\theta_i = (90^\circ - 5^\circ) = (\pi/2 - 0.087)$ rad.

$$n = \frac{ct_p}{a \sin\theta} = \frac{(3 \times 10^8)[\text{m/s}] \times (100 \times 10^{-15})[\text{s}]}{(1.67 \times 10^{-6})[\text{m}] \times \sin(\pi/2 - 0.087)} = 18 \tag{81}$$

We can compare this number, $n = 18$, to FROG numbers in Trebino's review paper.[8] We see that this number seems to be typical.

The number of full frames using (80) is ($w = 0.25$ cm)

$$N_{\text{frames}} = \frac{2 \times (0.25 \times 10^{-2})[\text{m}]}{18 \times (1.67 \times 10^{-6})[\text{m}] \cos(\pi/2 - 0.087)} = 1914 \tag{82}$$

Correcting the quadratic phase

Let us recall from (69) that $u^{(1-)}$ is

$$u^{(1-)}(q\Delta x, d_g) = -b \frac{e^{ikd_g} e^{ik(q\Delta x)^2/2d_g}}{\sqrt{i\lambda d_g}} \bar{S}(q\Delta\varphi) \tag{83}$$

where we have suppressed the θ_i argument of $u^{(1-)}$ because we will always be in the regime which satisfies the conditions for this approximation. (See section *Case II:* $\theta_i = \frac{\pi}{2} - \delta\theta_i$.)

We can correct for the quadratic phase $e^{ikx^2/2d_g}$ in (83) with a converging lens by a suitable choice of parameters because it is well known that the transmission function of a lens is

$$t_\ell = e^{-ikx^2/2f_\ell} \tag{84}$$

where f_ℓ is the focal length of the lens. And so if we choose the focal length of lens 1 to be d_g then the quadratic phase in $u^{(1-)}$ is compensated and thus we get

$$u^{(1+)}(x) = t_\ell u^{(1-)} = -b \frac{e^{ikd_g}}{\sqrt{i\lambda d_g}} \bar{S}\Big[\varphi(x)\Big] \tag{85}$$

Propagating through Lens 2

We have to add two more lenses after lens 1 to recover $\bar{S}$ without introducing anymore quadratic phases because when we place lens 2 a distance equal to its focal length f_2 from

lens 1 and look at the image at its back focal plane we have

$$u^{(2)}(x, f_2) = b\frac{e^{ikd_g}}{i\lambda\sqrt{d_g f_2}} \int_{-\infty}^{\infty} dx' \; \bar{S}\Big[\varphi(x')\Big] e^{-i2\pi x x'/\lambda f_2} \tag{86}$$

Note: it is trivial to prove that applying the Fourier transform twice to a function $s(t)$ produces $s(-t)$.

Propagating through Lens 3

Lens 3 is used for creating the FrFT of the image from Lens 2. We have fixed the "local" focal length to F_3 and so the position for Len 3 is $RQF_3 = (\tan\alpha/2)(\sin\alpha)f_3$ from (34) (also see Figure 4).

For $\alpha = \pi/2$, we can set $f_3 = f_2$ we can show that $\bar{S}$ can be recovered because at this position Lens 3 performs a Fourier transform of an object placed at the front focal length of Lens 3.

Let us consider the case when we place lens 3 f_2 from from the focal plane of lens 2 and look at the image f_2 from the back of lens 3. The final image is thus

$$\begin{aligned}
u^{(3)}(x, f_2) &= -b\frac{e^{ikd_g}}{i\lambda^{3/2} f_2\sqrt{d_g}} \int_{-\infty}^{\infty} dx'' \left(\int_{-\infty}^{\infty} dx' \; \bar{S}\Big[\varphi(x')\Big] e^{-i2\pi x' x''/\lambda f_2} \right) e^{-i2\pi x'' x/\lambda f_2} \\
&= -b\frac{e^{ikd_g}}{\lambda^{3/2} f_2\sqrt{d_g}} \int_{-\infty}^{\infty} dx' \; \bar{S}\Big[\varphi(x')\Big] \int_{-\infty}^{\infty} dx'' \; e^{-i2\pi(x'+x)x''/\lambda f_2} \\
&= -b\frac{e^{ikd_g}}{\sqrt{\lambda d_g}} \int_{-\infty}^{\infty} dx' \; \bar{S}\Big[\varphi(x')\Big] \delta(x + x') \\
&= -b\frac{e^{ikd_g}}{\sqrt{\lambda d_g}} \bar{S}\Big[\varphi(-x)\Big]
\end{aligned} \tag{87}$$

The complete setup for producing the α-order FrFT of $\bar{s}$ is shown in Figure 4.

Table AII.2 Simulation Parameters

Parameter	Value	Comments
λ	800 ns	Ti-Sapphire laser wavelength
t_p	100 fs	longitudinal pulse width
w	0.25 cm	transverse $1/e$ radius of the laser
a	1.67 μm	600 lines/mm diffraction grating (600 lines/mm)
a'	0.83 μm	1200 lines/mm diffraction grating for *Numerical Demonstration 2*
Δx_{pixel}	14 μm	length of CCD pixel (Hamamatsu S10420)
N_{pixel}	2048	true number of CCD pixels is 2068
$d_g = f_1$	5 cm	focal length of lens 1
f_2	100 cm	focal length of lens 2
F_3	100 cm	"local" focal length of lens 3

APPENDIX III: EXTRACTING $s(\phi)$ FROM $\bar{s}(\phi)$

We note that the input pulse $s(t)$ is sampled spatially at a distance $a\sin\theta_i$ *just before* the diffraction grating. See Figure 23. If the number of slits used is $p_{\max}$, then the maximum transverse length of the sampled pulse is $p_{\max}a\sin\theta_i$

However, also from Figure 23, *just after* the diffraction grating, the intensity pattern is spaced a apart. Therefore, the maximum transverse length is $p_{\max}a = \Delta x_{\text{grid}}$. The magnification between the object at the diffraction grid and the image on P_2 is given by[f]

$$\text{magnification } \nu = f_2/f_1 \tag{88}$$

For example, if we use the numbers in Table AII.2 and (82), $p_{\max} = 18$ fs, then $\Delta x_{\text{grid}} = 30$ μm. We expect the size of the image Δx_{P_2} on P_2 to be

$$\Delta x_{P_2} = \left(\frac{f_2}{f_1}\right)\Delta x_{\text{grid}} \tag{89}$$

For $f_1 = 0.05$ m and $f_2 = 1.0$ m, we have

$$\Delta x_{P_2} = \left(\frac{1.0}{0.05}\right) \times 30[\mu\text{m}] = 20 \times 30[\mu\text{m}] = 0.60\text{mm}$$

This is exactly what we see in Figure 10.

$\bar{s}(\varphi)$ was defined in (61) which we repeat here but with $s(\varphi)$ on the lhs

$$s(\varphi) = \bar{s}(\varphi)e^{-i\varphi^2/(2kd_g\sin^2\theta_i)}e^{-i\varphi} \tag{90}$$

But because of the magnification factor shown in (88), we actually measure $\bar{s}(-\nu x)$ (the negative sign of the argument comes the two Fourier Transforms of $\bar{s}(x)$. See (87))and so our correction must reflect this magnification factor (including the sign)

$$s(-\nu x) = \bar{s}(-x)e^{-i(-x)^2/(2kd_g\sin^2\theta_i)}e^{-i(-x)} \tag{91}$$

[f] The proof is trivial: The Fresnel integral between the grid and P_g contains $e^{-ixx'/\lambda f_1}$ while between P_g and P_2 it contains $e^{-ixx'/\lambda f_2}$. Therefore, the ratio f_2/f_1 relates the size of the object on the grid to the image on P_2.

Finally, we must take into account the slow sampling compared to the oscillations of $e^{ikx\sin\theta_i} = e^{i\varphi}$. We can superimpose the under-sampled version of $\bar{s}$ to the "highly" sampled $\bar{s}$. This is shown in Figure 11. The *Mathematica* programme in the section *Numerical Demonstration 1* shows how we perform the extraction.

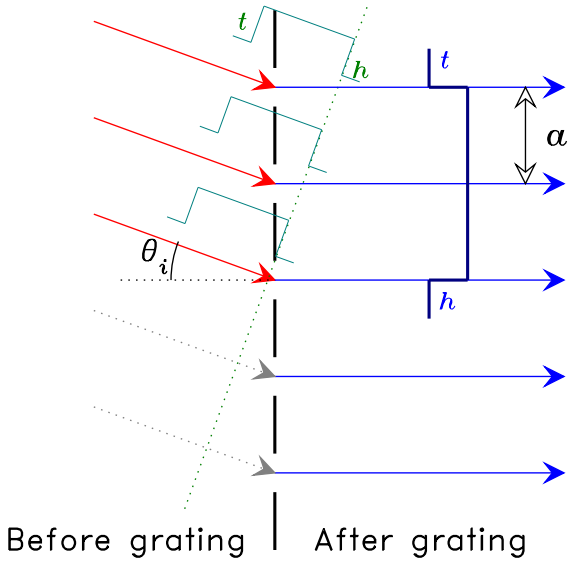


Figure 23 The input pulse is sampled at an interval of $a\sin\theta_i$ in the spatial domain to the left of the diffraction grating. To the right of the grid, the optics looks at a pulse which is sampled at a intervals.

Appendix IV: Numerical Demonstration of the Optical Setup

Demonstration 1

The goal is to see how well we can reconstruct the s[t] signal using reasonable parameters for the Ti:Sapphire laser. In this demonstration grating size is very small here b=1nm and incident angle $\theta_i = 85$ deg = 1.48 rad is near grazing. The input is a rectangle pulse so that we can understand the mechanics.

Simulation Parameters

Ti : Sapphire laser parameters

```
In[1]:= λ = 800 × 10^-9; (*wavelength of TiS, m*)
        c = 3 × 10^8; (* speed of light, m/s*)

In[3]:= k = 2 π / λ; (* wave number m*)
```

1/e radius of the laser

```
In[4]:= w = 0.25 × 10^-2; (*m*)
```

Grating parameters

The parameters of the grating, assuming 600 grooves/mm.
The period of the grating

```
In[5]:= a = 10^-3 / 600.; (*m*)
```

The size of the slit

```
In[6]:= b = 10^-9; (*m*)
```

CCD parameters

The size of a pixel and number of pixels on a CCD camera, for example Hamamatsu S10420

```
In[7]:= Δxpixel = 14 × 10^-6; (*m*)

In[8]:= Npixel = 2048; (* set it to 2048, actual number of pixels is 2068x70 *)
```

Integration Step Size

For convenience, we will make the integration step size, the same as the CCD camera

```
In[9]:= Δx = Δxpixel;

In[10]:= NΔx = Npixel;
```

■ Optics parameters

Lens 1 focal length

In[11]:=
```
f1 = 0.05; (*m*)
```

Lens 2 focal length

In[12]:=
```
f2 = 1.0; (*m*)
```

■ Input pulse parameters

The incident angle

In[13]:=
```
θi = 85 π / 180;
```

The input pulse is 100fs long

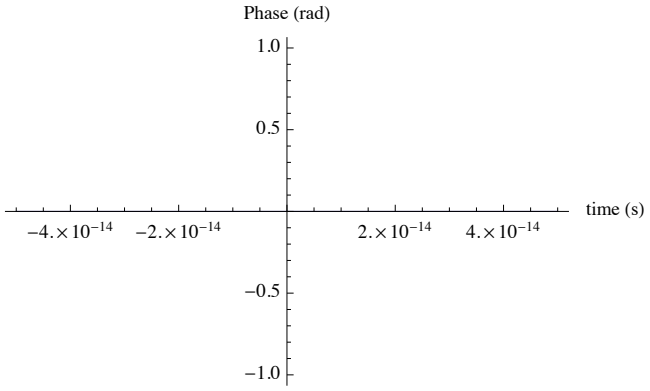
In[14]:=
```
tp = 100 × 10^-15; (*s*)
```

In[15]:=
```
η[t_] := If[-0.5 ≤ t / (tp) < 0.5, 1, 0];
```

In[16]:=
```
sinput[t_] := η[t]
```

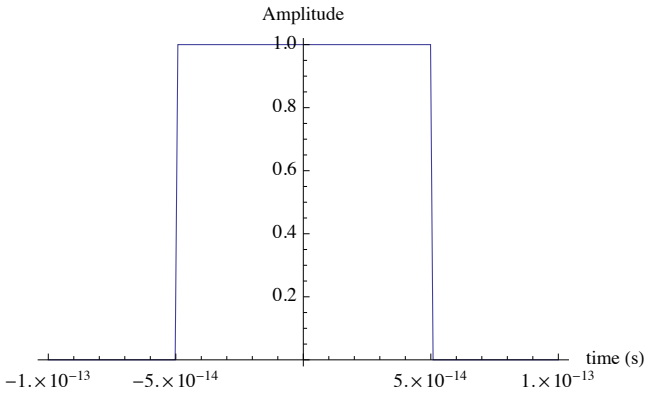
In[17]:=
```
Plot[Arg[sinput[t]], {t, -tp / 2, tp / 2},
 PlotRange → All, AxesLabel → {"time (s)", "Phase (rad)"}]
```

Out[17]=



In[18]:=
```
Plot[Abs[sinput[t]], {t, -tp, tp}, AxesLabel → {"time (s)", "Amplitude"}]
```

Out[18]=



This is from Eq (46) which describes the distribution just before the diffraction grating. Note that the argument of sinput is converted from phase ϕ to time.

In[19]:=
```
ug[x_, ϕ_] := sinput[(ϕ / (2 π)) λ / c] Exp[-(x Cos[θi])^2 / w^2] Exp[i k x Sin[θi]];
```

The Diffraction Grating

From Example AII.2, Eq (81), we know that pmax=18

In[20]:=
```
pmax = 18;
```

Use the u1 approximation

We will use the approximation from Eq (75)

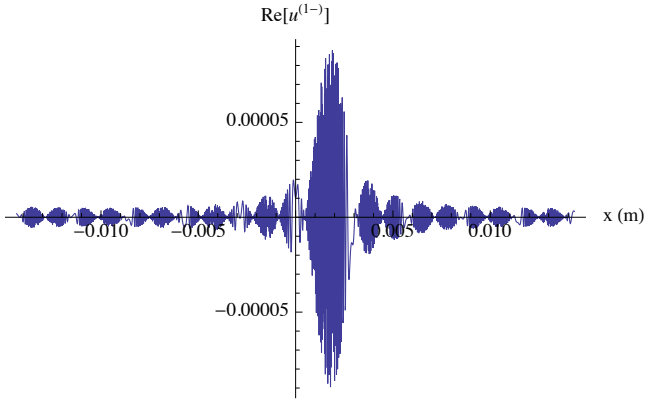
In[21]:=
```
u1approx[x_] :=
  -b Exp[i k f1] Exp[i k x^2 / (2 f1)] / Sqrt[i λ f1] Sum[sinput[(p k a Sin[θi] / (2 π)) λ / c] Exp[i k (p a)^2 / (2 f1)] Exp[i p k a Sin[θi]]
    Exp[-(p a Cos[θi])^2 / w^2] Exp[-i p k a x / f1], {p, -pmax / 2, pmax / 2 - 1}]
```

In[22]:=
```
Lu1approx = Table[{i Δx, u1approx[i Δx]}, {i, -NΔx / 2, NΔx / 2 - 1}];
```

In[23]:=
```
Lu1approxr = Table[{Lu1approx[[i, 1]], Re[Lu1approx[[i, 2]]]}, {i, Length[Lu1approx]}];
Lu1approxi = Table[{Lu1approx[[i, 1]], Im[Lu1approx[[i, 2]]]}, {i, Length[Lu1approx]}];
```

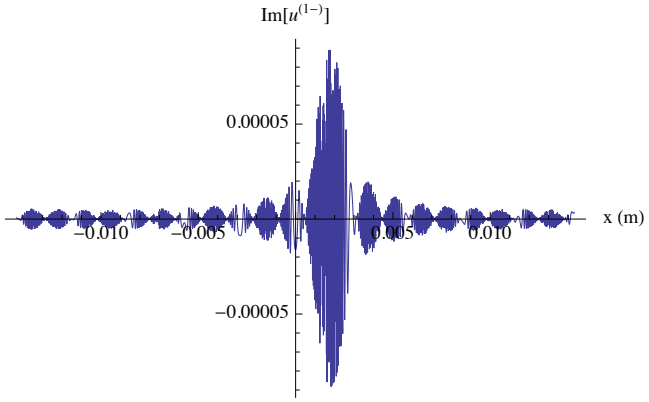
In[25]:=
```
ListPlot[Lu1approxr, Joined → True, PlotRange → All, AxesLabel → {"x (m)", "Re[u^(1-)]"}]
```

Out[25]=



In[26]:=
```
ListPlot[Lu1approxi, Joined → True, PlotRange → All, AxesLabel → {"x (m)", "Im[u(1-)]"}]
```

Out[26]=



Looks pretty good! The rapid oscillations are from Exp[i k x^2 / 2 f1] which we will correct with lens 1 in the next section. Note: the x axis has dimensions of length which is required for input into Lens 2.

■ Correct Quadratic Phase

Correct the quadratic phase with lens 1.

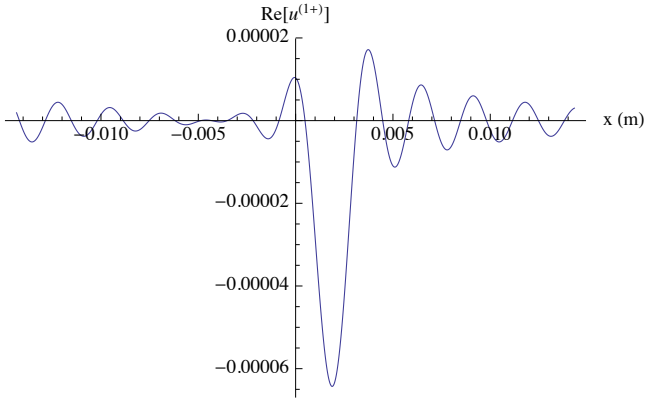
In[27]:=
```
u1p[x_] := Exp[-i k x^2 / (2 f1)] Lu1approx[[Round[x / Δx + NΔx / 2 + 1], 2]]
```

In[28]:=
```
Lu1p = Table[{i Δxpixel, u1p[i Δxpixel]}, {i, -NΔx / 2, NΔx / 2 - 1}];
```

In[29]:=
```
Lu1pr = Table[{Lu1p[[i, 1]], Re[Lu1p[[i, 2]]]}, {i, Length[Lu1p]}];
Lu1pi = Table[{Lu1p[[i, 1]], Im[Lu1p[[i, 2]]]}, {i, Length[Lu1p]}];
```

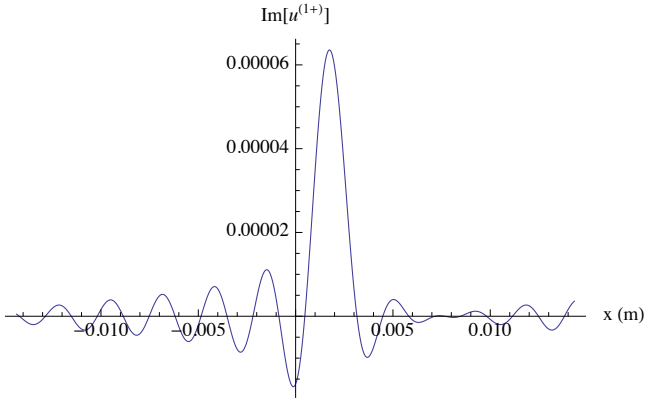
In[31]:=
```
ListPlot[Lu1pr, Joined → True, PlotRange → All, AxesLabel → {"x (m)", "Re[u(1+)]"}]
```

Out[31]=



In[32]:=
```
ListPlot[Lu1pi, Joined → True, PlotRange → All, AxesLabel → {"x (m)", "Im[u^(1+)]"}]
```

Out[32]=



■ Propagating through Lens 2

x must have dimensions of length for input into lens 2. Fortunately we do have this criteria satisfied in the construction of Eq(86). The output x axis also has the dimensions of length.

Instead of performing the integral in Eq(86) with NIntegral[], we will replace it with a Sum[] instead to speed things up.

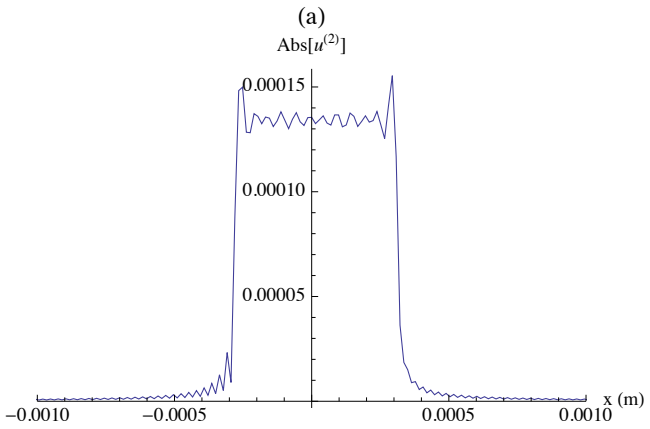
In[33]:=
```
u2[x_] := -1/Sqrt[i λ f2] Sum[u1p[i Δx] Exp[-i 2 π x (i Δx)/(λ f2)], {i, -NΔx/2, (NΔx/2 - 1)}] Δx
```

In[34]:=
```
Lu2 = Table[{i Δx, u2[i Δx]}, {i, -NΔx/2, NΔx/2 - 1}];
```

In[35]:=
```
Lmagu2 = Table[{Lu2[[i, 1]], Abs[Lu2[[i, 2]]]}, {i, Length[Lu2]}];
Largu2 = Table[{Lu2[[i, 1]], Arg[Lu2[[i, 2]]]}, {i, Length[Lu2]}];
Lu2r = Table[{Lu2[[i, 1]], Re[Lu2[[i, 2]]]}, {i, Length[Lu2]}];
Lu2i = Table[{Lu2[[i, 1]], Im[Lu2[[i, 2]]]}, {i, Length[Lu2]}];
```

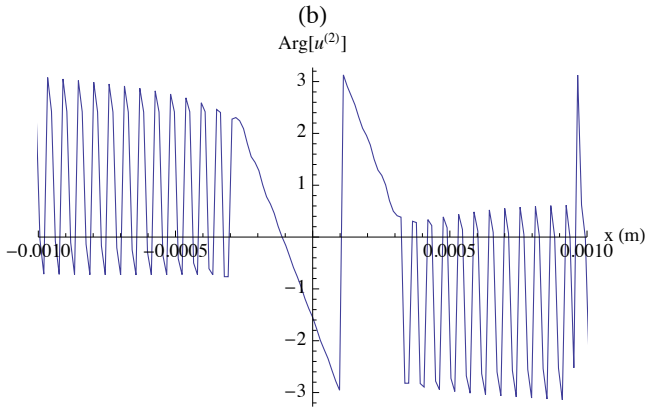
In[39]:=
```
ListPlot[Lmagu2, Joined → True, PlotRange → {{-0.001, 0.001}, All},
 AxesLabel → {"x (m)", "Abs[u^(2)]"}, PlotLabel → "(a)"]
```

Out[39]=



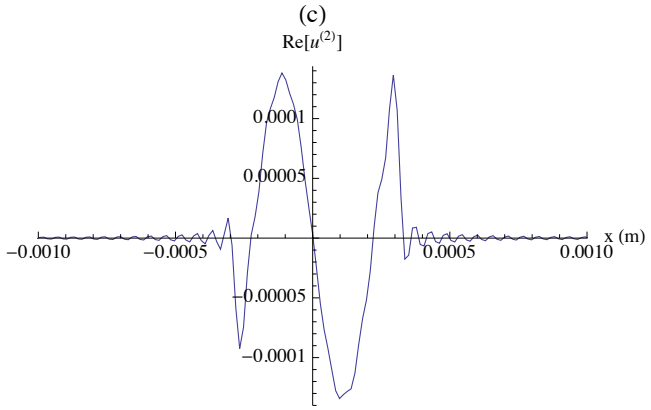
In[40]:=
```
ListPlot[Largu2, Joined → True, PlotRange → {{-0.001, 0.001}, All},
 AxesLabel → {"x (m)", "Arg[u(2)]"}, PlotLabel → "(b)"]
```

Out[40]=



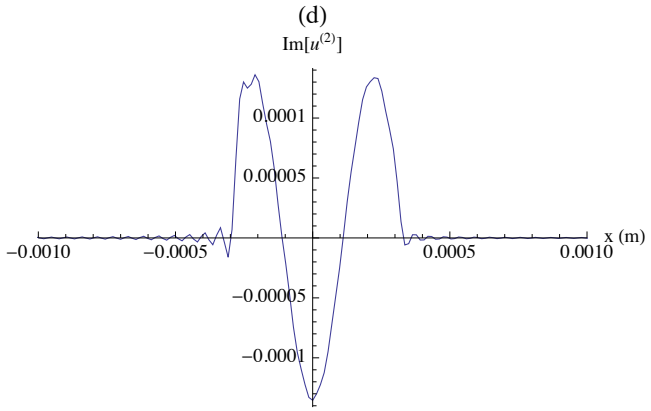
In[41]:=
```
ListPlot[Lu2r, Joined → True, PlotRange → {{-0.001, 0.001}, All},
 AxesLabel → {"x (m)", "Re[u(2)]"}, PlotLabel → "(c)"]
```

Out[41]=



In[42]:=
```
ListPlot[Lu2i, Joined → True, PlotRange → {{-0.001, 0.001}, All},
 AxesLabel → {"x (m)", "Im[u(2)]"}, PlotLabel → "(d)"]
```

Out[42]=



Aperture Stop

We have to put in an aperture stop for the Enhanced Bootstrap Algorithm to work.

Define the aperture function

In[43]:=
```
ap[x_, xstop_] := If[ -xstop ≤ x ≤ xstop, 1, 0]
```

Create a interpolation of Lu2 at the exit of Lens 2

In[44]:=
```
u2fi = Interpolation[Lu2]
```

Out[44]=
```
InterpolatingFunction[{{-0.014336, 0.014322}}, <>]
```

and adding the aperture function

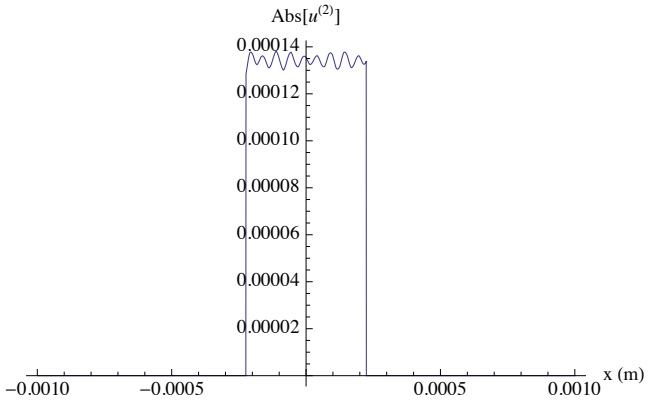
In[45]:=
```
u2f[x_, xstop_] := ap[x, xstop] u2fi[x]
```

■ Chunk 1

The first chunk which picks out the centre of the distribution (16 Δx)

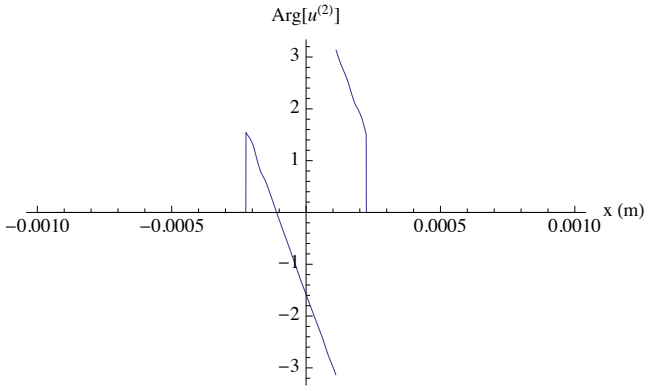
In[46]:=
```
Plot[Abs[u2f[x, 16 Δx]], {x, -0.001, 0.001},
 PlotRange -> All, AxesLabel → {"x (m)", "Abs[u(2)]"}]
```

Out[46]=



In[47]:=
```
Plot[Arg[u2f[x, 16 Δx]], {x, -0.001, 0.001},
 PlotRange -> All, AxesLabel → {"x (m)", "Arg[u(2)]"}]
```

Out[47]=



In[48]:=
```
Export["~/papers/frog/math/demo1chunk1.dat",
 Table[{N[i Δx], N[u2f[i Δx, 16 Δx]]}, {i, -NΔx / 2, (NΔx / 2 - 1)}]]
```

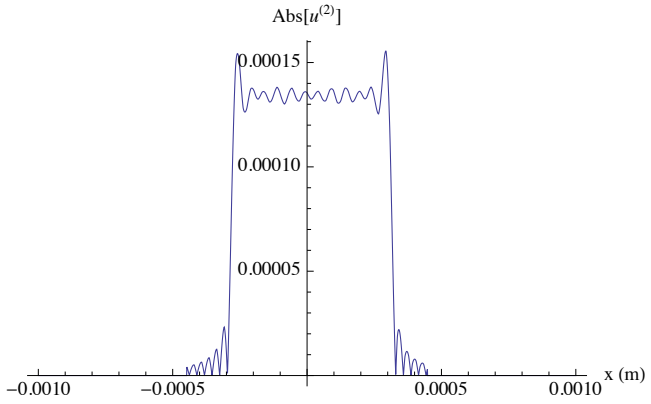
Out[48]=
```
~/papers/frog/math/demo1chunk1.dat
```

Chunk 2

The second chunk which encompasses the entire pulse (32 Δx)

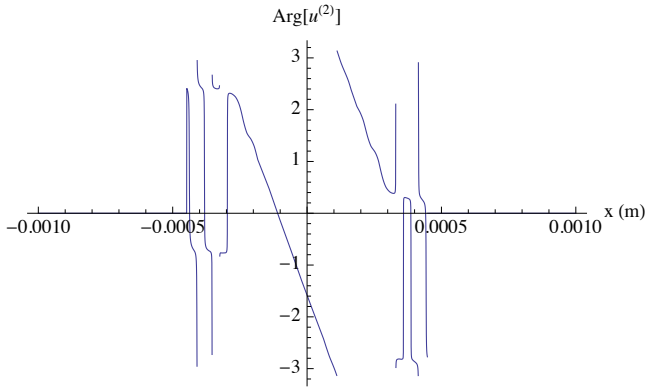
In[49]:=
```
Plot[Abs[u2f[x, 32 Δx]], {x, -0.001, 0.001},
 PlotRange -> All, AxesLabel → {"x (m)", "Abs[u(2)]"}]
```

Out[49]=



In[50]:=
```
Plot[Arg[u2f[x, 32 Δx]], {x, -0.001, 0.001},
 PlotRange -> All, AxesLabel → {"x (m)", "Arg[u(2)]"}]
```

Out[50]=



In[51]:=
```
Export["~/papers/frog/math/demo1chunk2.dat",
 Table[{N[i Δx], N[u2f[i Δx, 32 Δx]]}, {i, -NΔx / 2, (NΔx / 2 - 1)}]]
```

Out[51]= ~/papers/frog/math/demo1chunk2.dat

Chunks 1 and 2 will be analysed in demo1a.nb

Demonstration 1 using Type 1 Optics

Demonstration 1 produced chunk1 and chunk2 for the "Enhanced Bootstrap Algorithm" to be analysed here with the type I optics.

Type 1 Optics DFrFT

The Type 1 dimensionfull DFrFT is given by Eq (36)

```
In[1]:= DFrFT[s_, n_, α_, Δt_, F_, λ_, m_] :=
          Block[{Δxp, F3}, F3 = (Sin[α] F); Δxp = λ F3 Sin[α] / (n Δt); Return[Exp[I α / 2] / Sqrt[I λ F3 Sin[α]] Δt
            Sum[s[k Δt] Exp[I π / (λ F3) ((m Δxp)^2 + (k Δt)^2) Cot[α] - I 2 π k m / n], {k, -n / 2, n / 2 - 1}]]];
```

Arguments of DFrFT[] are:

- s: function to be transformed
- n: number of points to be transformed. Must be power of 2
- α: order of the transform
- Δxp: pixel size of the CCD camera
- F: focal length of the lens
- λ: wavelength of the laser
- m: returns value of transform at m Δxp

It is assumed that the time step gives the frequency step with the relationship Δx Δxp= λ F1 Sin[α]/n.
For type I configuration, F1 = Q F and z = R F1 = RQ F. For type I: Q = sin[α] R = Tan[α/2]

Type I Optics Setup

Laser wavelength

```
In[2]:= λ = 800 × 10^-9; (*m*)
```

Fixed "local" focal length

```
In[3]:= F3 = 1; (*m*)
```

Focal length of the lens used for Type 1 optics for α and γ

```
In[4]:= α = π / 2;
        γ = π / 4;

In[6]:= F3α = F3 / Sin[α] (*m*)

Out[6]= 1

In[7]:= F3γ = F3 / Sin[γ] (*m*)

Out[7]= √2
```

Size of CCD pixel

In[8]:=
```
Δxpixel = 14 × 10^-6; (*m*)
```

Number of CCD pixels. Technically there are 2068 pixels, but just to make things a power of 2, we are setting it to 2048

In[9]:=
```
Npixel = 2048;
```

For convenience, set the sampling size as the CCD pixel size

In[10]:=
```
Δx = Δxpixel;
```

Define the normalization factor from Eq (37)

In[11]:=
```
μ = Sqrt[λ F3] // N
```

Out[11]=
```
0.000894427
```

Read in the chunk1 and chunk2 data

The input function sinput[t] which we had used previously

In[12]:=
```
Lchunk1 = ReadList["~/papers/frog/math/demo1chunk1.dat", {Number, Expression}];
Lchunk2 = ReadList["~/papers/frog/math/demo1chunk2.dat", {Number, Expression}];
```

Select out the signal part

In[14]:=
```
Lchunk1a = {};
For[i = 1, i ≤ Length[Lchunk1], i++,
  If[Abs[Lchunk1[[i, 2]]] ≠ 0,
    Lchunk1a = Join[Lchunk1a, {Lchunk1[[i]]}];
   ];
 ];
```

In[16]:=
```
fchunk1 = Interpolation[Lchunk1a, InterpolationOrder → 1]
```

Out[16]=
```
InterpolatingFunction[{{-0.000224, 0.000224}}, <>]
```

For numeric reasons, we are increasing the size of the signal by 10^3. The first aperture is +/- 16Δx

In[17]:=
```
fapchunk1[x_] := If[-16 Δx ≤ x < 16 Δx, 10^3 fchunk1[x], 0]
```

In[18]:=
```
Lchunk2a = {};
For[i = 1, i ≤ Length[Lchunk2], i++,
  If[Abs[Lchunk2[[i, 2]]] ≠ 0,
    Lchunk2a = Join[Lchunk2a, {Lchunk2[[i]]}];
   ];
 ];
```

In[20]:=
```
fchunk2 = Interpolation[Lchunk2a, InterpolationOrder → 1]
```

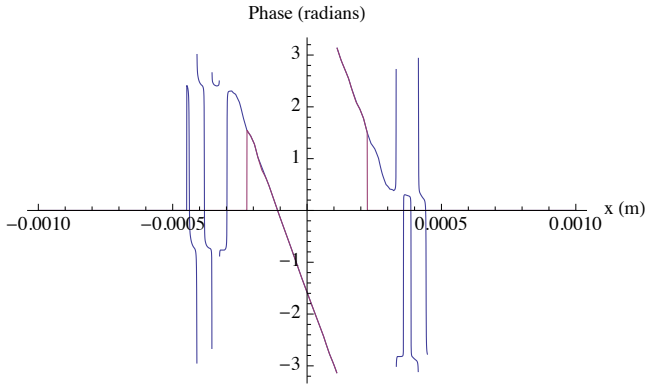
Out[20]=
```
InterpolatingFunction[{{-0.000448, 0.000448}}, <>]
```

The second aperture is +/- 32Δx

In[21]:=
```
fapchunk2[x_] := If[-32 Δx ≤ x < 32 Δx, 10^3 fchunk2[x], 0]
```

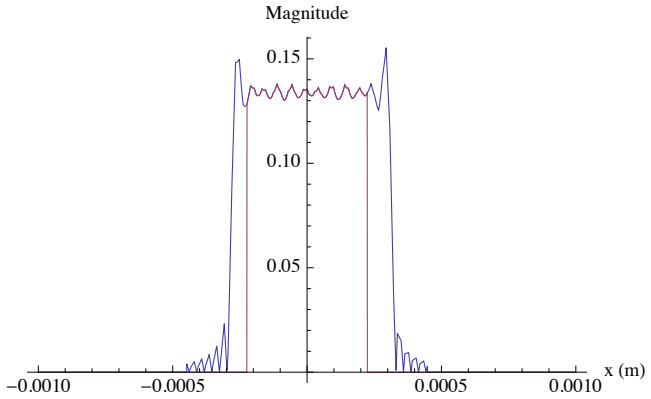
In[22]:=
```
Plot[{Arg[fapchunk2[x]], Arg[fapchunk1[x]]}, {x, -10^-3, 10^-3},
 PlotRange → All, AxesLabel → {"x (m)", "Phase (radians)"}]
```

Out[22]=



In[23]:=
```
Plot[{Abs[fapchunk2[x]], Abs[fapchunk1[x]]},
 {x, -10^-3, 10^-3}, PlotRange → All, AxesLabel → {"x (m)", "Magnitude"}]
```

Out[23]=



■ First measurement parameters with a small chunk

We select $\alpha = \pi/2$ and $\gamma = \pi/4$ and work on the first chunk

In[24]:=
```
nm = Npixel;
(*total number of samples which is
 approximately equal to the number of pixels on the CCD*)
n1 = 16 × 2;  (* number of samples where s is the real signal and not padding*)
α = π / 2;
γ =  π / 4;
Δt = Δx / μ; (*dimensionaless temporal variable created from Δx*)
```

Make dimensionless $\Delta f\alpha$ and $\Delta f\gamma$. Note: $\lambda F1/\mu = \mu$

In[29]:=
```
Δfα = λ F3 Sin[α] / (μ nm Δt)  // N;
Δfγ = λ F3 Sin[γ] / (μ nm Δt)  // N;
```

■ Create the I_α and I_γ from chunk 1

We need nm points of intensity from I_α and I_γ from sinput1[t].

In[31]:=
```
iα =
  Table[{m Δfα, Abs[ DFrFT[fapchunk1, nm, α, Δx, F3α, λ, m]]^2}, {m, -nm / 2, nm / 2 - 1}] // N;
iγ = Table[{m Δfγ, Abs[ DFrFT[fapchunk1, nm, γ, Δx, F3γ, λ, m]]^2},
    {m, -nm / 2, nm / 2 - 1}] // N;
```

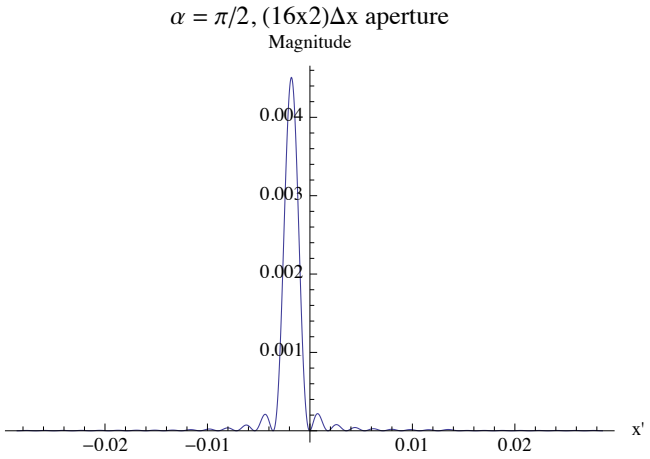
Make functions for the data sets

In[33]:=
```
Iα[f_] := iα[[Round[f / Δfα] + nm / 2 + 1]][[2]];
Iγ [f_] := iγ[[Round[f / Δfγ] + nm / 2 + 1]][[2]];
```

Plot them out

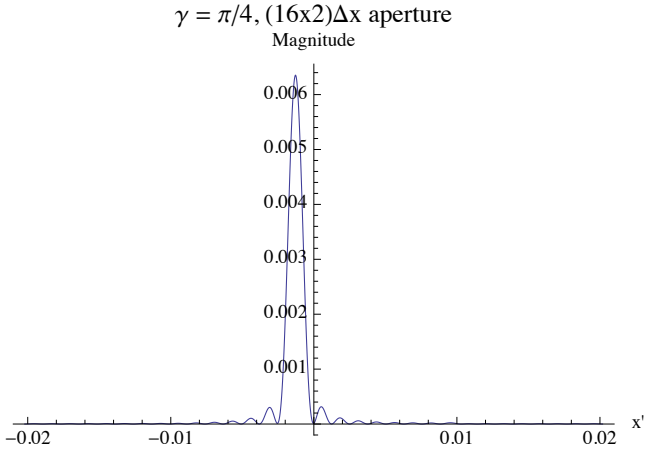
In[35]:=
```
ListPlot[iα, Joined → True, PlotRange → All,
 AxesLabel → {"x'", "Magnitude"}, PlotLabel → "α = π/2, (16x2)Δx aperture"]
```

Out[35]=



In[36]:=
```
ListPlot[iγ, Joined → True, PlotRange → All,
 AxesLabel → {"x'", "Magnitude"}, PlotLabel → "γ = π/4, (16x2)Δx aperture"]
```

Out[36]=



$I_\alpha$and I_γ are the intensities which we measure. And note that the x-axis which is now frequency is dimensionless.

- **Let's normalize iα and iγ so that their power is 1**

In[37]:=
```
totalPower = Sin[α] / (nm (Δt)^2) Sum[iα[[i, 2]], {i, Length[iα]}];
```

```
In[38]:= iα = iα / totalPower;
         iγ = iγ / totalPower;
```

The Bootstrap Method

We define the equations for the bootstrap method defined as equation (19) in the paper.

```
In[40]:= u[θ_, m_] := Exp[I 2 π (-n1 / 2) (n1 - m) (Δt)^2 Cot[θ]];
         v[θ_, m_] := Exp[I 2 π (-n1 / 2  + m - 1) (n1 - m) (Δt)^2 Cot[θ]];
         Cα[k_] :=
           Sin[α] / (nm Δt^2) Exp[-I π (k Δt)^2 Cot[α]] Sum[Iα[m Δfα] Exp[I 2 π k m / nm], {m, -nm / 2, nm / 2 - 1}];
         Cγ[k_] := Sin[γ] / (nm Δt^2) Exp[-I π (k Δt)^2 Cot[γ]] Sum[Iγ[m Δfγ] Exp[I 2 π k m / nm], {m, -nm / 2, nm / 2 - 1}];
         Σ[θ_, m_] := Sum[ Conjugate[s[-n1 / 2 + j]]
              s[n1 / 2 - m + j] Exp[I 2 π (n1 - m) (-n1 / 2 + j) (Δt)^2 Cot[θ]], {j, 1, m - 2}];
         d[m_] := Det[( u[α, m]  v[α, m]
                        u[γ, m]  v[γ, m] )];
```

Check that $C\alpha[0] = C\gamma[0]$

```
In[46]:= Cα0 = Cα[0]
Out[46]= 1.
In[47]:= Cγ0 = Cγ[0]
Out[47]= 1.
```

Solving s[t]

First set s[t] = 0 for -n1/2 to -nm/2+1 and nm/2 to n1/2-1

```
In[48]:= For[i = -nm / 2, i ≤ -n1 / 2 - 1, i++,
           s[i] = 0;
         ];
         For[i = n1 / 2, i ≤ nm / 2 - 1, i++,
           s[i] = 0;
         ];
```

Without loss of generality, we will set $s[-N/2]=\sigma=1$ first and then use equation equation (25) to solve for $s[-n/2]$ because every s[n] is either multiplied or divided by $s[-n/2]$.

```
In[50]:= σ = 1;
In[51]:= cα = Chop[ Cα[n1 - 1] Exp[I π n1 (n1 - 1) Δt^2 Cot[α]]]
Out[51]= 0.0288895 + 0.00669871 i
In[52]:= cγ = Chop[Cγ[n1 - 1] Exp[I π n1 (n1 - 1) Δt^2 Cot[γ]]]
Out[52]= 0.0288895 + 0.00669871 i
```

Clearly $c\alpha$ and $c\gamma$ are equal as required.

Iteration

```
In[53]:= s[-n1 / 2] = σ
Out[53]= 1
```

Average $c\alpha$ and $c\gamma$ because they are supposed to be the same.

```
In[54]:= s[n1 / 2 - 1] = (cα + cγ) / (2.0 σ)
Out[54]= 0.0288895 + 0.00669871 i
```

Also set $c\alpha$ to the average since it is used below

```
In[55]:= cα = s[n1 / 2 - 1]
Out[55]= 0.0288895 + 0.00669871 i
```

From equation (16), we can solve for s[n/2-2] and s[-n/2+1]

```
In[56]:= s[n1 / 2 - 2] = Chop[First[1 / Conjugate[σ] d[2] (v[γ, 2] - v[α, 2]).(Cα[n1 - 2] / Cγ[n1 - 2])][[1]]]
Out[56]= 0.0278015 + 0.0121123 i
```

```
In[57]:= s[-n1 / 2 + 1] =
  Chop[First[σ / Conjugate[cα d[2]] Conjugate[(-u[γ, 2] u[α, 2]).(Cα[n1 - 2] / Cγ[n1 - 2])]][[1]]]
Out[57]= 1.06447 - 0.117289 i
```

Now we can continue the bootstrap process until every element is done

```
In[58]:=
For[m = 3, m ≤ n1 / 2, m++,
  s[n1 / 2 - m] =
   Chop[First[1 / Conjugate[σ] d[m] (v[γ, m] - v[α, m]).(Cα[n1 - m] - Σ[α, m] / Cγ[n1 - m] - Σ[γ, m])][[1]]];
  s[-n1 / 2 + m - 1] = Chop[First[σ / Conjugate[cα d[m]]
      Conjugate[(-u[γ, m] u[α, m]).(Cα[n1 - m] - Σ[α, m] / Cγ[n1 - m] - Σ[γ, m])]][[1]]];
 ];
```

■ Calculate the magnitude s[-n/2]

We can calculate the magnitude of s[-n/2] using equation (25) of the paper. We will denote s[-n/2] = σ
σ2 MUST BE REAL so that s[-n/2] is REAL. See eq(25)

```
In[59]:= s[-n1 / 2] = σ; (* placeholder for the above calculations *)
```

```
In[60]:= sol = Chop[Solve[σ2 Sum[Abs[s[-n1 / 2 + m - 1]]^2, {m, 1, n1 / 2}] +
             1/σ2 Sum[Abs[s[n1 / 2 - m]]^2, {m, 1, n1 / 2}] == Cα[0], σ2]]
Out[60]= {{σ2 → 0.0285318}, {σ2 → 0.0285889}}
```

$\sigma 2$ must be real. So just select the real parts only for the calculations below

```
In[61]:= σfirst = Re[σ2] /. First[sol]
Out[61]= 0.0285318

In[62]:= σlast = Re[σ2] /. Last[sol]
Out[62]= 0.0285889
```

We select between the two solutions by using equation (26)

s0 = s[-n/2] G$\alpha\gamma$[n/2 + 1]

```
In[63]:= firstsol1 =
           Sqrt[σ2] Chop[First[1/Conjugate[cα d[n1 / 2 + 1]] Conjugate[(-u[γ, n1 / 2 + 1] u[α, n1 / 2 + 1]).
                 {{Cα[n1 - (n1 / 2 + 1)] - Σ[α, n1 / 2 + 1]}, {Cγ[n1 - (n1 / 2 + 1)] - Σ[γ, n1 / 2 + 1]}}]][[1]]] /. σ2 → σfirst
Out[63]= -0.178392 - 0.00763894 i
```

The other part which is also s[0] from equation (26)

s0 = $\frac{1}{s[-n/2]}$ F$\alpha\gamma$[$n/2$]

```
In[64]:= firstsol2 =
           1/Sqrt[σ2] Chop[First[1/d[n1 / 2] (v[γ, n1 / 2] - v[α, n1 / 2]).{{Cα[n1 - n1 / 2] - Σ[α, n1 / 2]}, {Cγ[n1 - n1 / 2] - Σ[γ, n1 / 2]}}][[
             1]]] /. σ2 → σfirst
Out[64]= -0.178392 - 0.00763894 i
```

- **Check the second solution**

This is a quick check that the other solution is not correct

```
In[65]:= secondsol1 =
           Sqrt[σ2] Chop[First[1/Conjugate[cα d[n1 / 2 + 1]] Conjugate[(-u[γ, n1 / 2 + 1] u[α, n1 / 2 + 1]).
                 {{Cα[n1 - (n1 / 2 + 1)] - Σ[α, n1 / 2 + 1]}, {Cγ[n1 - (n1 / 2 + 1)] - Σ[γ, n1 / 2 + 1]}}]][[1]]] /. σ2 → σlast
Out[65]= -0.17857 - 0.00764658 i
```

In[66]:=
```
secondsol2 =
 1/Sqrt[σ2] Chop[First[1/d[n1/2] (v[γ, n1/2] - v[α, n1/2]).(Cα[n1 - n1/2] - Σ[α, n1/2]
                                                           Cγ[n1 - n1/2] - Σ[γ, n1/2])]][[
    1]]] /. σ2 → σlast
```

Out[66]= $-0.178213 - 0.0076313\,i$

Select between the two solutions by looking at their differences

In[67]:=
```
If[Abs[firstsol1 - firstsol2] < Abs[secondsol1 - secondsol2],
 σ = Sqrt[σ2] /. σ2 → σfirst,
 σ = Sqrt[σ2] /. σ2 → σlast
]
```

Out[67]= 0.168914

■ Normalize s

Once we have σ, we can normalize all the s's

In[68]:=
```
For[m = 1, m ≤ n1/2, m++,
  sn[-n1/2 + m - 1] = σ s[-n1/2 + m - 1];
 ];
For[m = 1, m ≤ n1/2, m++,
  sn[n1/2 - m] = 1/σ s[n1/2 - m];
 ];
```

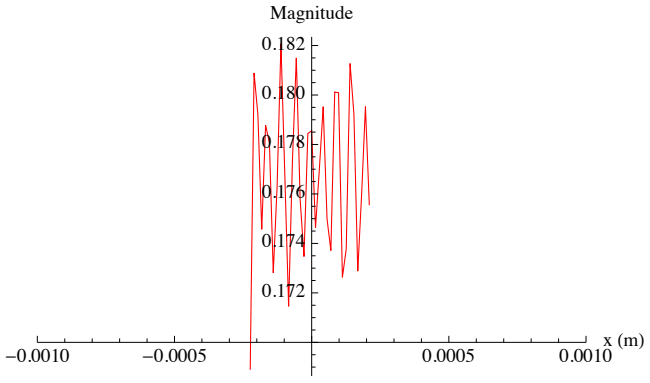
Plot out dimensionfull pictures

In[70]:=
```
sMag = Table[{k Δt μ, Abs[sn[k]]}, {k, -n1/2, n1/2 - 1}];
sArg = Table[{k Δt μ, Arg[sn[k]]}, {k, -n1/2, n1/2 - 1}];
sRe = Table[{k Δt μ, Re[sn[k]]}, {k, -n1/2, n1/2 - 1}];
sIm = Table[{k Δt μ, Im[sn[k]]}, {k, -n1/2, n1/2 - 1}];
```

■ Plots

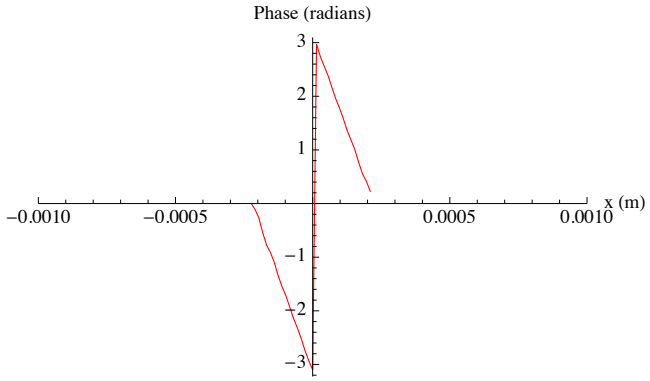
In[74]:=
```
ListPlot[sMag, Joined → True, PlotStyle → RGBColor[1, 0, 0],
 PlotRange → {{-10^-3, 10^-3}, All}, AxesLabel → {"x (m)", "Magnitude"}]
```

Out[74]=



In[75]:=
```
ListPlot[sArg, Joined → True, PlotStyle → RGBColor[1, 0, 0],
 PlotRange → {{-10^-3, 10^-3}, All}, AxesLabel → {"x (m)", "Phase (radians)"}]
```

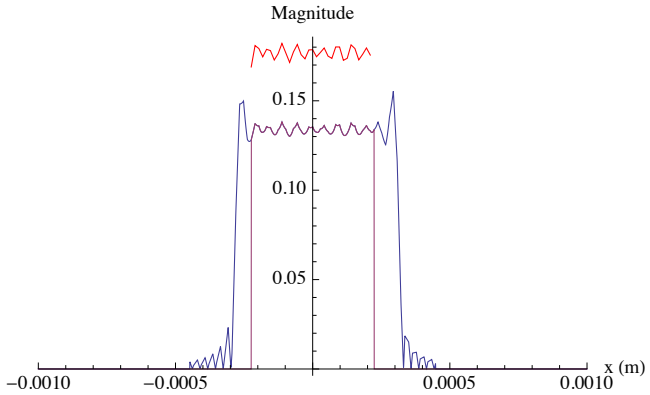
Out[75]=



- **Compare the result and the original chunk**

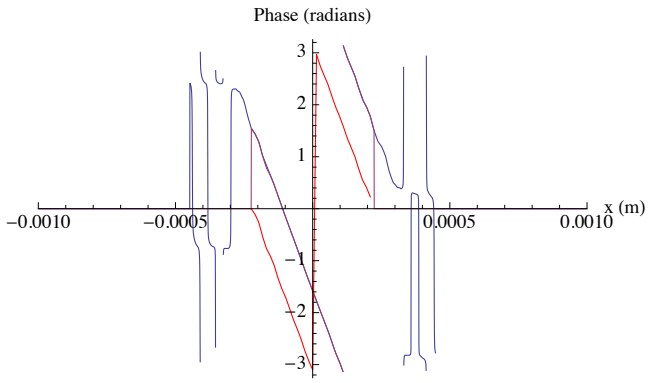
In[76]:=
```
Show[%23, %74, PlotRange → {{-10^-3, 10^-3}, All}]
```

Out[76]=



In[77]:=
```
Show[%22, %75, PlotRange → {{-10^-3, 10^-3}, All}]
```

Out[77]=



These two plots match very well the input phase up to a constant and magnitude up to a gain which is expected because the input power was normalised to 1.

Enhanced Bootstrap Algorithm

Using the sn[t] which we calculated, we will extend the solution for the next chunk.

The chunk has been increased by a factor of 2 from n1 to n2=2*n1.

In[78]:=
```
n2 = 2 n1;  (* number of samples where s is the real signal and not padding*)
```

■ Create the I_α and I_γ from chunk 2

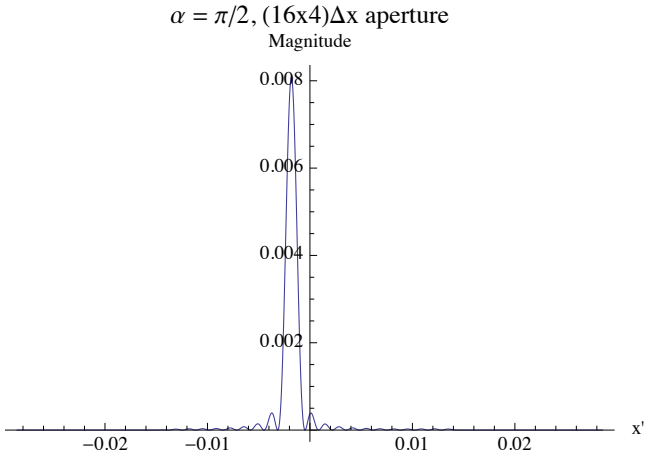
We need nm points of intensity from $I_\alpha$and I_γ from sinput1[t].

In[79]:=
```
iα =
  Table[{m Δfα, Abs[ DFrFT[fapchunk2, nm, α, Δx,  F3α, λ, m]]^2}, {m, -nm / 2, nm / 2 - 1}] // N;
iγ = Table[{m Δfγ, Abs[ DFrFT[fapchunk2, nm, γ, Δx,  F3γ, λ, m]]^2},
    {m, -nm / 2, nm / 2 - 1}] // N;
```

Plot them out

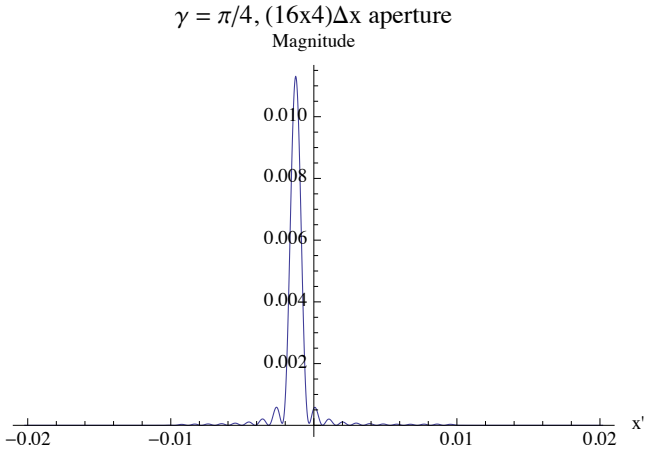
In[81]:=
```
ListPlot[iα, Joined → True, PlotRange → All,
 AxesLabel → {"x'", "Magnitude"}, PlotLabel → "α = π/2, (16x4)Δx aperture"]
```

Out[81]=



In[82]:=
```
ListPlot[iγ, Joined → True, PlotRange → All,
 AxesLabel → {"x'", "Magnitude"}, PlotLabel → "γ = π/4, (16x4)Δx aperture"]
```

Out[82]=



$I_\alpha$and I_γ are the intensities which we measure.

- **Normalize w.r.t. the power from the initial bootstrap**

```
In[83]:= iα = iα / totalPower;
         iγ = iγ / totalPower;
```

Create the Matrix Vector Equation

Clear previous definition of σ

```
In[85]:= Clear[σ];
         Clear[σs];
```

Define the equations that are the entries of the matrix S

```
In[87]:= σs[k_, l_, θ_] := Conjugate[sn[k]] Exp[i 2 π k l (Δt)^2 Cot[θ]];
         σ[k_, l_, θ_] := sn[k] Exp[i 2 π (k - l) l (Δt)^2 Cot[θ]];
```

Filling in the $S_\theta$matrix shown in Eq. (29)

```
In[89]:= Sα = Table[0, {i, n1 / 2 + 1}, {j, n1}];
         Sγ = Table[0, {i, n1 / 2 + 1}, {j, n1}];
         For[i = 1, i ≤ n1 / 2 + 1, i++,
           For[j = 1, j ≤ n1 / 2, j++,
             Sα[[i, j]] = σs[-n1 / 2 + (j - 1) + (i - 1), n1 - (i - 1), α];
             Sγ[[i, j]] = σs[-n1 / 2 + (j - 1) + (i - 1), n1 - (i - 1), γ];
            ];
          ];

         For[i = 1, i ≤ n1 / 2 + 1, i++,
           For[j = 1, j ≤ n1 / 2, j++,
             Sα[[i, j + n1 / 2]] = σ[(j - 1) - (i - 1), n1 - (i - 1), α];
             Sγ[[i, j + n1 / 2]] = σ[(j - 1) - (i - 1), n1 - (i - 1), γ];
            ];
          ];
```

The $C_\theta$vector showin in Eq. (31)

```
In[93]:= CCα = Table[0, {i, n1 / 2 + 1}, {j, 1, 1}];
         CCγ = Table[0, {i, n1 / 2 + 1}, {j, 1, 1}];
         For[i = 1, i ≤ n1 / 2 + 1, i++,

           CCα[[i, 1]] = Cα[n1 - (i - 1)] - Sum[Conjugate[sn[-n1/2 + j - 1]]
                sn[n1/2 + j - i] Exp[i 2 π (-n1/2 + j - 1) (n1 - (i - 1)) (Δt)^2 Cot[α]], {j, 1, i - 1}];

           CCγ[[i, 1]] = Cγ[n1 - (i - 1)] - Sum[Conjugate[sn[-n1/2 + j - 1]] sn[n1/2 + j - i]
                Exp[i 2 π (-n1/2 + j - 1) (n1 - (i - 1)) (Δt)^2 Cot[γ]], {j, 1, i - 1}];

          ];
```

Create the matrix equation which we solve for the unknown s[] entries Eq. (28)

In[96]:=
```
S = Table[0, {i, n1 + 2}, {j, n1}];
CC = Table[0, {i, n1 + 2}, {j, 1, 1}];
For[i = 1, i ≤ n1 / 2 + 1, i++,
  For[j = 1, j ≤ n1, j++,
    S[[i, j]] = Sα[[i, j]];
   ];
 ];
For[i = 1, i ≤ n1 / 2 + 1, i++,
  For[j = 1, j ≤ n1, j++,
    S[[i + (n1 / 2 + 1), j]] = Sγ[[i, j]];
   ];
 ];
For[i = 1, i ≤ n1 / 2 + 1, i++,
  CC[[i, 1]] = CCα[[i, 1]];
 ];
For[i = 1, i ≤ n1 / 2 + 1, i++,
  CC[[i + (n1 / 2 + 1), 1]] = CCγ[[i, 1]];
 ];
```

The number of equations is greater than the number of variables. Therefore, we'll have to use a least squares method to solve for s[]

In[102]:=
```
sol = LeastSquares[S, CC];
```

Now, get the answers to the unknowns using Eq (30)

In[103]:=
```
For[i = 1, i ≤ n1 / 2, i++,
  sn[n1 / 2 + (i - 1)] = sol[[i, 1]];
 ];
For[i = 1, i ≤ n1 / 2, i++,
  sn[-n1 + (i - 1)] = Conjugate[sol[[n1 / 2 + i, 1]]];
 ];
```

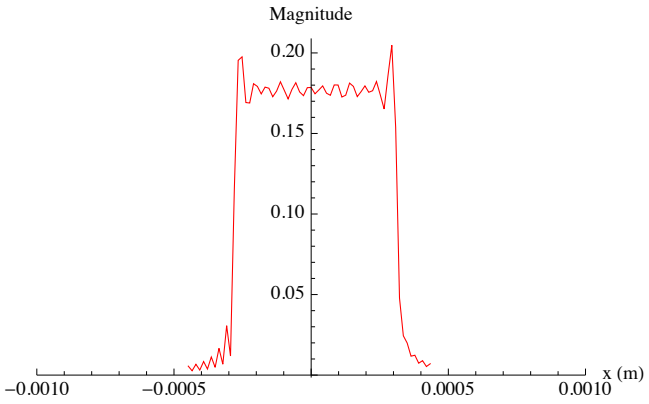
■ Plots

Plot out dimensionfull pictures

In[105]:=
```
sMag = Table[{k Δt μ, Abs[sn[k]]}, {k, -n2 / 2, n2 / 2 - 1}];
sArg = Table[{k Δt μ, Arg[sn[k]]}, {k, -n2 / 2, n2 / 2 - 1}];
sRe = Table[{k Δt μ, Re[sn[k]]}, {k, -n2 / 2, n2 / 2 - 1}];
sIm = Table[{k Δt μ, Im[sn[k]]}, {k, -n2 / 2, n2 / 2 - 1}];
```

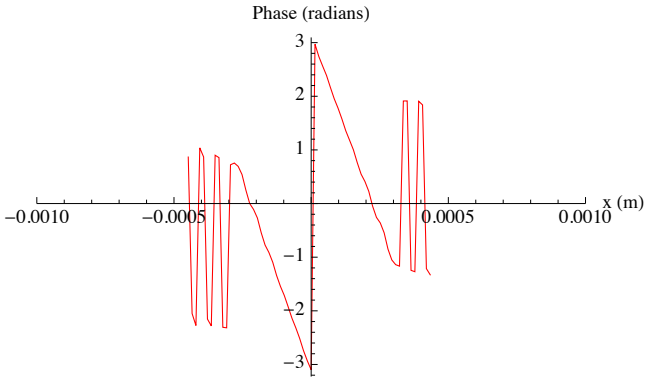
In[109]:=
```
ListPlot[sMag, Joined → True, PlotStyle → RGBColor[1, 0, 0],
 PlotRange → {{-10^-3, 10^-3}, All}, AxesLabel → {"x (m)", "Magnitude"}]
```

Out[109]=



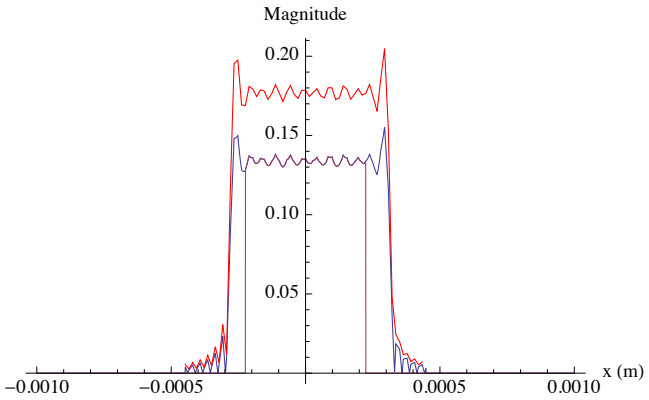
In[110]:=
```
ListPlot[sArg, Joined → True, PlotStyle → RGBColor[1, 0, 0],
 PlotRange → {{-10^-3, 10^-3}, All}, AxesLabel → {"x (m)", "Phase (radians)"}]
```

Out[110]=



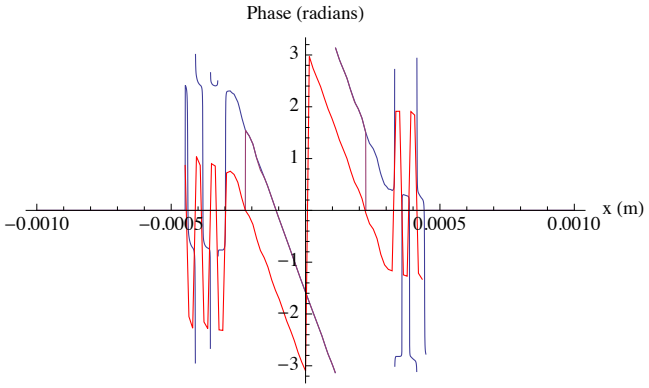
In[111]:=
```
Show[%23, %109, AxesLabel → {"x (m)", "Magnitude"}]
```

Out[111]=



In[112]:=
```
Show[%22, %110, AxesLabel → {"x (m)", "Phase (radians)"}]
```

Out[112]=



EBA works as expected!

Correct the quadratic phase

There is a phase contribution which was introduced by lens 1. See Eq (90). We can correct for this.

The size of the diffraction grating and incident angle

In[113]:=
```
a = 10^-3 / 600.; (*m*)
```

In[114]:= `θi = 85 π / 180;`

focal length of lens 1

In[115]:= `f1 = 0.05; (*m*)`

focal length of lens 2

In[116]:= `f2 = 1.0; (*m*)`

The magnification factor

In[117]:= `ν = f2 / f1`

Out[117]= `20.`

The wave number of the laser

In[118]:= `k = 2 π / λ;`

From Example AII.2, Eq (81), we can calculate the number of samples

In[119]:= `c = 3 × 10^8; (* speed of light, m/s*)`

In[120]:= `tp = 100 × 10^-15;`

In[121]:= `pmax = IntegerPart[ c tp / (a Sin[θi]) ]`

Out[121]= `18`

The phase correction comes from Eq (61) and add in a possible phase offset

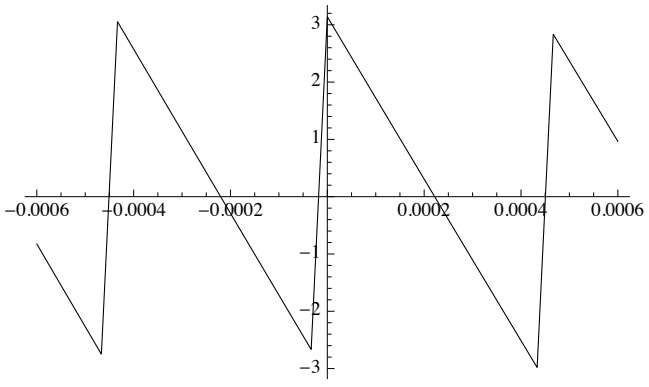
In[122]:= `phasecorr[x_, θ_] := Arg[Exp[i θ] Exp[i k x^2 / (2 f1)] Exp[i k x Sin[θi]]]`

We generate a table of the phase corrections which are spaced a apart. Note that we are undersampling the phase correction and also the magnification factor ν (the negative sign in the argument of phasecorr[] comes from performing the Fourier transform twice, see Eq (40)). We anticipate that the phase offset is π.

In[123]:= `pcorr = Table[{p a ν, phasecorr[-p a, π]}, {p, -pmax, pmax}];`

In[124]:= `ListPlot[pcorr, Joined → True, PlotStyle → RGBColor[0, 0, 0]]`

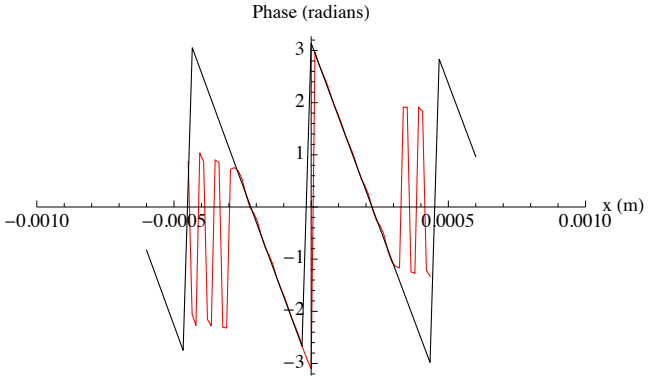
Out[124]=



Show that the phase matches what we have calculated

In[125]:=
```
Show[%110, %124]
```

Out[125]=



Create the phase correction function which includes the magnification factor from pcorr

In[126]:=
```
fcorr = Interpolation[pcorr, InterpolationOrder → 1]
```

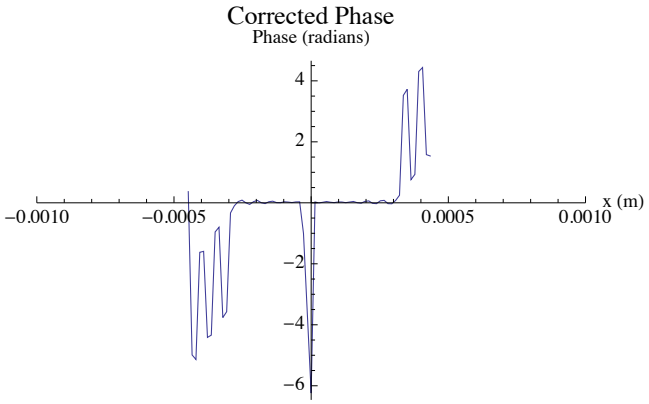
Out[126]=
```
InterpolatingFunction[{{-0.0006, 0.0006}}, <>]
```

Perform the corection

In[127]:=
```
corrArg = Table[{sArg[[i, 1]], sArg[[i, 2]] - fcorr[sArg[[i, 1]]]}, {i, Length[sArg]}];
```

In[128]:=
```
ListPlot[corrArg, Joined → True, PlotRange → {{-10^-3, 10^-3}, All},
 AxesLabel → {"x (m)", "Phase (radians)"}, PlotLabel → "Corrected Phase"]
```

Out[128]=



■ Unwind the phase

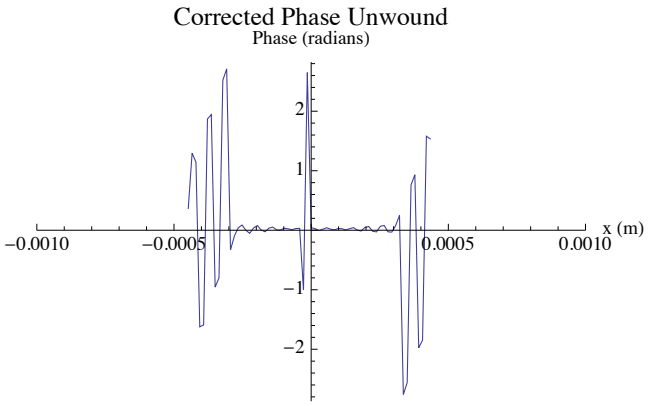
There are phases which outside the range +/- π. We can correct this with phaseUnwind below.

In[129]:=
```
phaseUnwind[θ_] := If[-π ≤ θ ≤ π, Return[θ],
    If[θ < -π, Return[θ + 2 π], Return[θ - 2 π]]
  ];
```

In[130]:=
```
ucorrArg = Table[{corrArg[[i, 1]], phaseUnwind[corrArg[[i, 2]]]}, {i, Length[corrArg]}];
```

In[131]:=
```
ListPlot[ucorrArg, Joined → True, PlotRange → {{-10^-3, 10^-3}, All},
 AxesLabel → {"x (m)", "Phase (radians)"}, PlotLabel → "Corrected Phase Unwound"]
```

Out[131]=



Notice the small discontinuity near the centre of the corrected phase. This is comes from imperfect phase correction.

Comparing with the input pulse

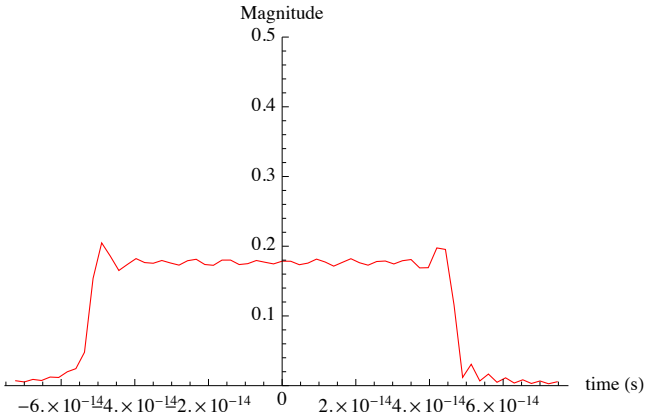
Rescale the magnitude and phase by ν to take out the magnification from Lens1 and Lens2 and change it to time on the xaxis. Also reverse the x axis, i.e. x-> -x because of the mirroring of the object by lens1 and lens2.

In[132]:=
```
rescaleMag = Table[{-sMag[[i, 1]]/(c ν), sMag[[i, 2]]}, {i, Length[sMag]}];

rescaleArg = Table[{-ucorrArg[[i, 1]]/(c ν), ucorrArg[[i, 2]]}, {i, Length[corrArg]}];
```

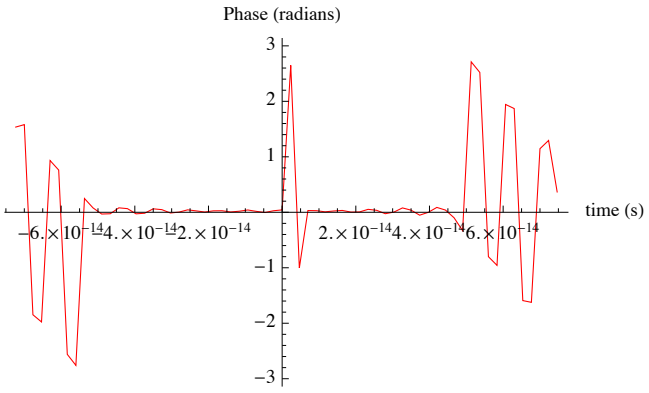
In[134]:=
```
ListPlot[rescaleMag, Joined → True, PlotStyle → RGBColor[1, 0, 0],
 PlotRange → {0, 0.5}, AxesLabel → {"time (s)", "Magnitude"} ]
```

Out[134]=



In[135]:=
```
ListPlot[rescaleArg, Joined → True, PlotStyle → RGBColor[1, 0, 0],
 PlotRange → {-π, π}, AxesLabel → {"time (s)", "Phase (radians)"} ]
```

Out[135]=



■ Input pulse

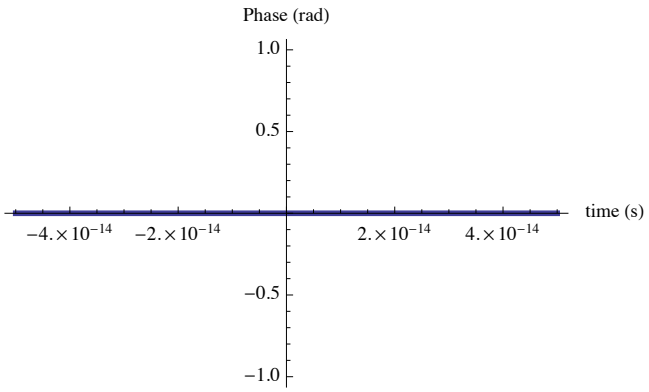
The input pulse is 100fs long and a constant pulse

In[136]:=
```
η[t_] := If[-0.5 ≤ t / (tp) < 0.5, 1, 0];
```

In[137]:=
```
sinput[t_] := η[t]
```

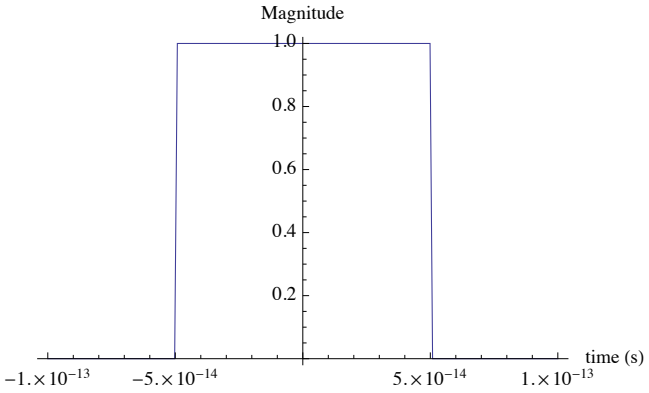
In[138]:=
```
Plot[Arg[sinput[t]], {t, -tp / 2, tp / 2}, PlotRange → All,
 AxesLabel → {"time (s)", "Phase (rad)"}, PlotStyle → Thickness[0.01]]
```

Out[138]=



In[139]:=
```
Plot[Abs[sinput[t]], {t, -tp, tp}, AxesLabel → {"time (s)", "Magnitude"}]
```

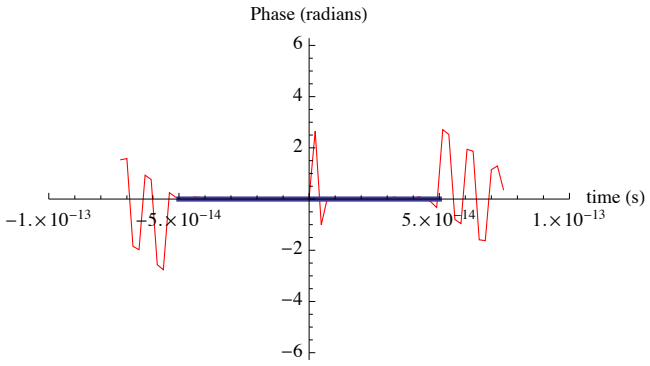
Out[139]=



Comparison

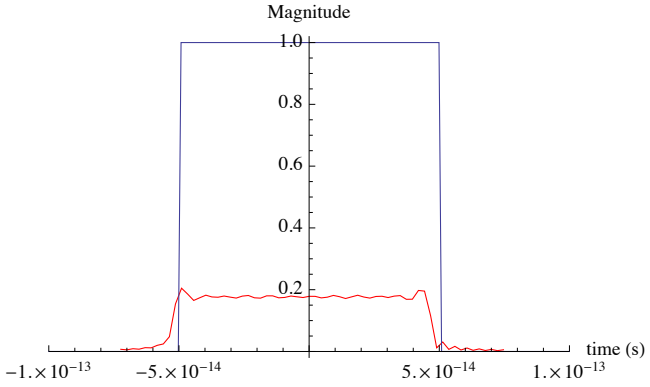
In[140]:=
```
Show[ %135, %138, PlotRange → {{-10^-13, 10^-13}, {-2 π, 2 π}}]
```

Out[140]=



In[141]:=
```
Show[%134, %139, PlotRange → {{-10^-13, 10^-13}, All}]
```

Out[141]=



Everything looks good! Magnitude needs rescaling as expected. Phase match looks great.

Demonstration 2

The goal is to see how well we can reconstruct the s[t] signal using reasonable parameters for the Ti:Sapphire laser. In this demonstration grating size is b=a/8=0.1 μm and a = 0.83μm, and incident angle θ_i = 85 deg = 1.48 rad is near grazing.

Simulation Parameters

Ti : Sapphire laser parameters

```
In[1]:= λ = 800 × 10^-9; (*wavelength of TiS, m*)
        c = 3 × 10^8; (* speed of light, m/s*)

In[3]:= k = 2 π / λ; (* wave number m*)
```

1/e radius of the laser

```
In[4]:= w = 0.25 × 10^-2; (*m*)
```

Grating parameters

The parameters of the grating, assuming 1200 lines/mm.
The distance between slits

```
In[5]:= a = 10^-3 / 1200.; (*m*)
```

The size of the slit

```
In[6]:= b = a / 8 (*m*)

Out[6]= 1.04167 × 10^-7
```

CCD parameters

The size of a pixel and number of pixels on a CCD camera, for example Hamamatsu S10420

```
In[7]:= Δxpixel = 14 × 10^-6; (*m*)

In[8]:= Npixel = 2048; (* set it to 2048, actual number of pixels is 2068x70 *)
```

Integration Step Size

For convenience, we will make the integration step size, the same as the CCD camera

```
In[9]:= Δx = Δxpixel;

In[10]:= NΔx = Npixel;
```

■ Optics parameters

Lens 1 focal length

In[11]:=
```
f1 = 0.05; (*m*)
```

Lens 2 focal length

In[12]:=
```
f2 = 1.0; (*m*)
```

■ Input pulse parameters

The incident angle

In[13]:=
```
θi = 85 π / 180;
```

The input pulse is 100fs long

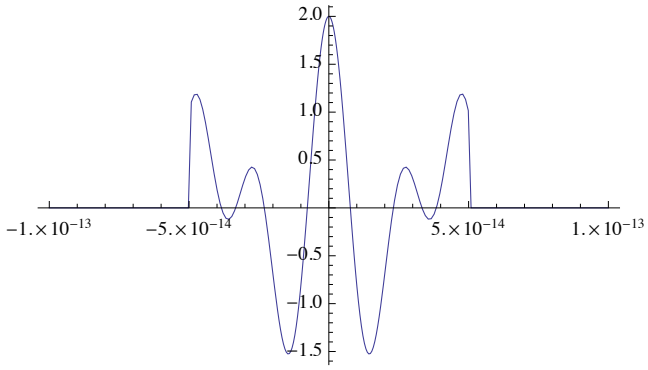
In[14]:=
```
tp = 100 × 10^-15; (*s*)
```

Using Cong's example Eq (27)

In[15]:=
```
sinput[t_] :=
  η[t] 0.1 Exp[-0.2 (t / (tp) + 0.5)^2] Exp[I (Sin[5 π (t / (tp) + 0.5)] + Cos[8 π (t / (tp) + 0.5)])];
η[t_] := If[-0.5 ≤ t / (tp) < 0.5, 1, 0];
```

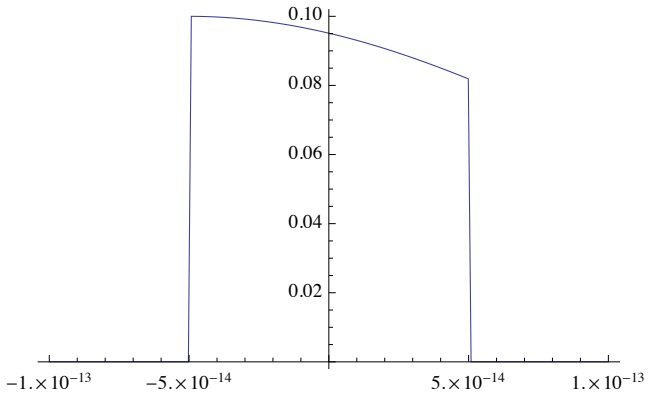
In[17]:=
```
Plot[Arg[sinput[t]], {t, -tp, tp}, PlotRange → All]
```

Out[17]=



In[18]:=
```
Plot[Abs[sinput[t]], {t, -tp, tp}]
```

Out[18]=



This is from Eq (46) which describes the distribution just before the diffraction grating. Note that the argument of sinput is converted from phase ϕ to time.

In[19]:=
```
ug[x_, ϕ_] := sinput[(ϕ / (2 π)) λ / c] Exp[-(x Cos[θi])^2 / w^2] Exp[i k x Sin[θi]];
```

The Diffraction Grating

From Example AII.2, Eq (81), we can calculate the number sampling frames

In[20]:=
```
pmax = IntegerPart[c tp / (a Sin[θi])]
```

Out[20]=
```
36
```

The Diffraction Grating

Check that b/λ is much less than 1 so that the u1 approximation from Eq(75) can be used

In[21]:=
```
b / λ // N
```

Out[21]=
```
0.130208
```

This should be small enough for the u1 approximation

Use the u1 approximation

We will use the approximation from Eq (75)

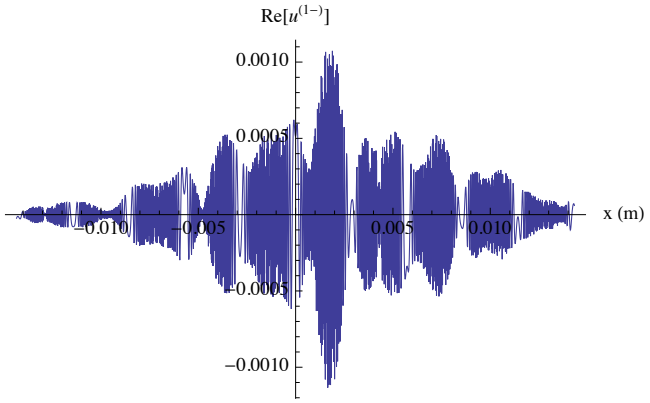
In[22]:=
```
u1approx[x_] :=
  -b Exp[i k f1] Exp[i k x^2 / (2 f1)] / Sqrt[i λ f1] Sum[sinput[(p k a Sin[θi] / (2 π)) λ / c] Exp[i k (p a)^2 / (2 f1)] Exp[i p k a Sin[θi]]
    Exp[-(p a Cos[θi])^2 / w^2] Exp[-i p k a x / f1], {p, -pmax / 2, pmax / 2 - 1}]
```

In[23]:=
```
Lu1approx = Table[{i Δx, u1approx[i Δx]}, {i, -NΔx / 2, NΔx / 2 - 1}];
```

In[24]:=
```
Lu1approxr = Table[{Lu1approx[[i, 1]], Re[Lu1approx[[i, 2]]]}, {i, Length[Lu1approx]}];
Lu1approxi = Table[{Lu1approx[[i, 1]], Im[Lu1approx[[i, 2]]]}, {i, Length[Lu1approx]}];
```

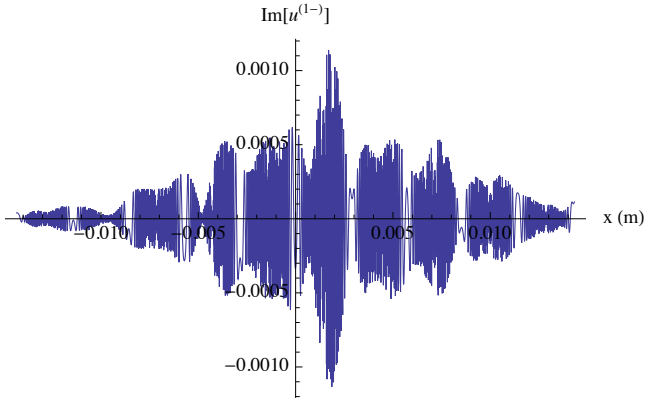
In[26]:=
```
ListPlot[Lu1approxr, Joined → True, PlotRange → All, AxesLabel → {"x (m)", "Re[u^(1-)]"}]
```

Out[26]=



In[27]:=
```
ListPlot[Lu1approxi, Joined → True, PlotRange → All, AxesLabel → {"x (m)", "Im[u(1-)]"}]
```

Out[27]=



Looks pretty good! The rapid oscillations are from Exp[i k x^2 / 2 f1] which we will correct with lens 1 in the next section. Note: the x axis has dimensions of length which is required for input into Lens 2.

■ Correct Quadratic Phase

Correct the quadratic phase with lens 1.

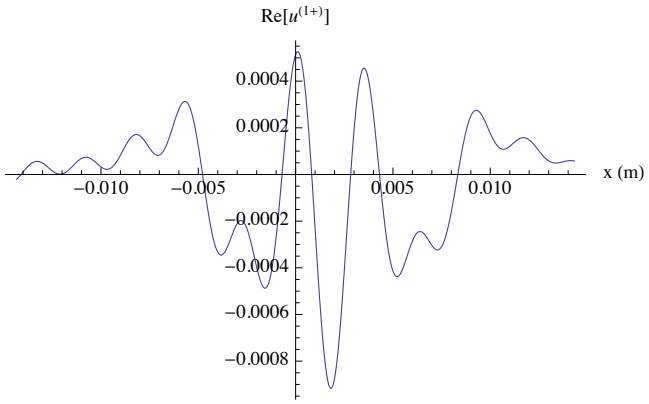
In[28]:=
```
u1p[x_] := Exp[-i k x^2 / (2 f1)] Lu1approx[[Round[x / Δx + NΔx / 2 + 1], 2]]
```

In[29]:=
```
Lu1p = Table[{i Δx, u1p[i Δx]}, {i, -NΔx / 2, NΔx / 2 - 1}];
```

In[30]:=
```
Lu1pr = Table[{Lu1p[[i, 1]], Re[Lu1p[[i, 2]]]}, {i, Length[Lu1p]}];
Lu1pi = Table[{Lu1p[[i, 1]], Im[Lu1p[[i, 2]]]}, {i, Length[Lu1p]}];
```

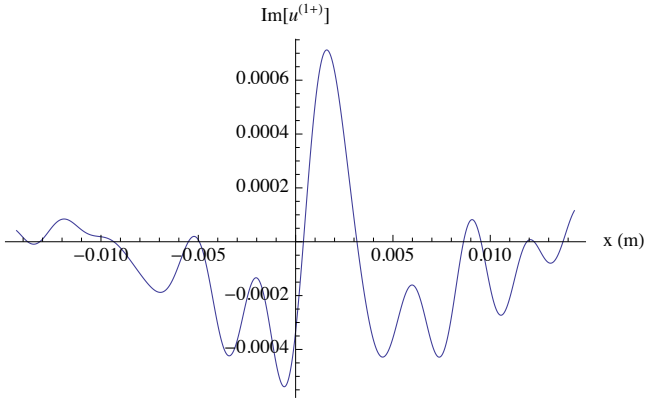
In[32]:=
```
ListPlot[Lu1pr, Joined → True, PlotRange → All, AxesLabel → {"x (m)", "Re[u(1+)]"}]
```

Out[32]=



In[33]:=
```
ListPlot[Lu1pi, Joined → True, PlotRange → All, AxesLabel → {"x (m)", "Im[u^(1+)]"}]
```

Out[33]=



■ Propagating through Lens 2

x must have dimensions of length for input into lens 2. Fortunately we do have this criteria satisfied in the construction of Eq(86). The output x axis also has the dimensions of length.

Instead of performing the integral in Eq(86) with NIntegral[], we will replace it with a Sum[] instead to speed things up.

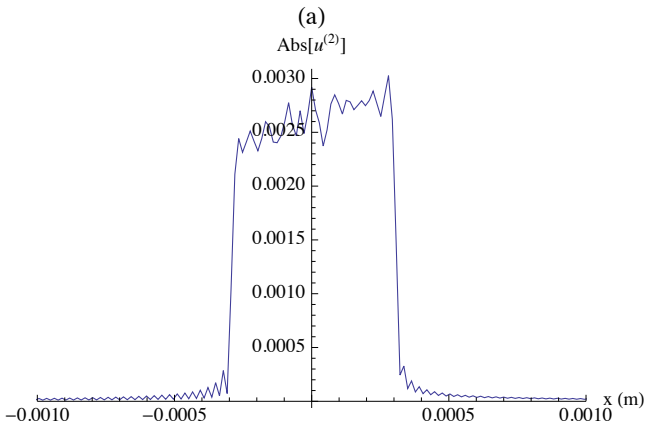
In[34]:=
```
u2[x_] := -1/Sqrt[i λ f2] Sum[ u1p[i Δx] Exp[-i 2 π x (i Δx)/(λ f2)], {i, -NΔx / 2, (NΔx / 2 - 1)}] Δx
```

In[35]:=
```
Lu2 = Table[{i Δx, u2[i Δx]}, {i, -NΔx / 2, NΔx / 2 - 1}];
```

In[36]:=
```
Lmagu2 = Table[{Lu2[[i, 1]], Abs[Lu2[[i, 2]]]}, {i, Length[Lu2]}];
Largu2 = Table[{Lu2[[i, 1]], Arg[Lu2[[i, 2]]]}, {i, Length[Lu2]}];
Lu2r = Table[{Lu2[[i, 1]], Re[Lu2[[i, 2]]]}, {i, Length[Lu2]}];
Lu2i = Table[{Lu2[[i, 1]], Im[Lu2[[i, 2]]]}, {i, Length[Lu2]}];
```

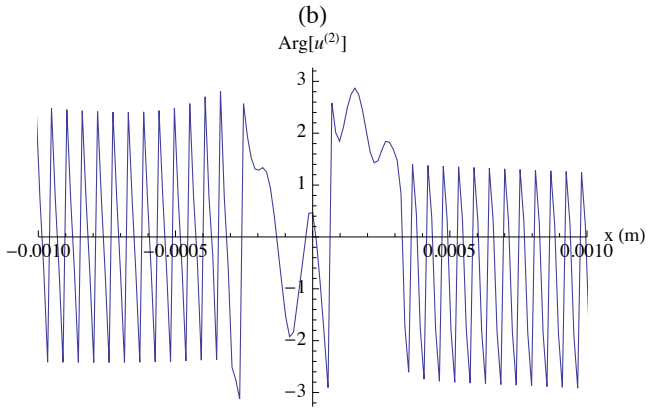
In[40]:=
```
ListPlot[Lmagu2, Joined → True, PlotRange → {{-0.001, 0.001}, All},
 AxesLabel → {"x (m)", "Abs[u^(2)]"}, PlotLabel → "(a)"]
```

Out[40]=



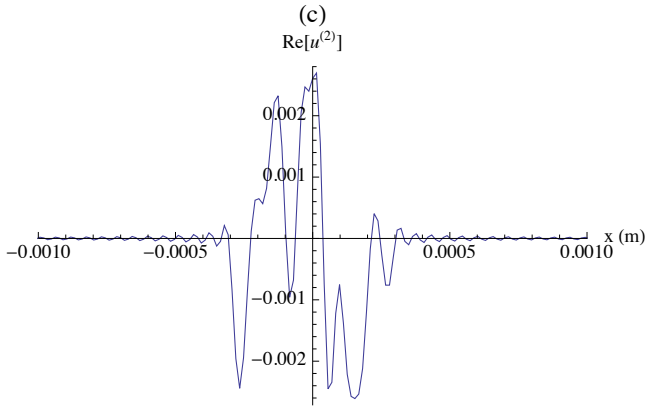
In[41]:=
```
ListPlot[Largu2, Joined → True, PlotRange → {{-0.001, 0.001}, All},
 AxesLabel → {"x (m)", "Arg[u(2)]"}, PlotLabel → "(b)"]
```

Out[41]=



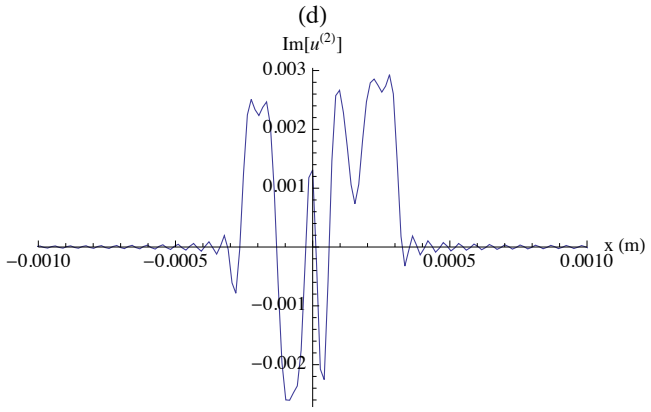
In[42]:=
```
ListPlot[Lu2r, Joined → True, PlotRange → {{-0.001, 0.001}, All},
 AxesLabel → {"x (m)", "Re[u(2)]"}, PlotLabel → "(c)"]
```

Out[42]=



In[43]:=
```
ListPlot[Lu2i, Joined → True, PlotRange → {{-0.001, 0.001}, All},
 AxesLabel → {"x (m)", "Im[u(2)]"}, PlotLabel → "(d)"]
```

Out[43]=



Aperture Stop

We have to put in an aperture stop for the Enhanced Bootstrap Algorithm to work.

Define the aperture function

In[44]:=
```
ap[x_, xstop_] := If[ -xstop ≤ x ≤ xstop, 1, 0]
```

Create a interpolation of Lu2 at the exit of Lens 2

In[45]:=
```
u2fi = Interpolation[Lu2, InterpolationOrder → 1]
```

Out[45]=
```
InterpolatingFunction[{{-0.014336, 0.014322}}, <>]
```

and adding the aperture function

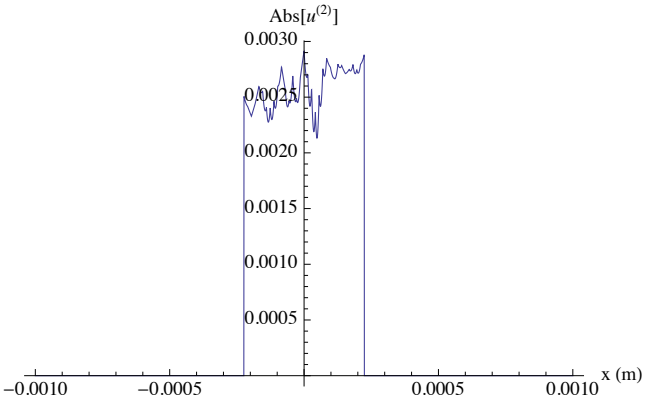
In[46]:=
```
u2f[x_, xstop_] := ap[x, xstop] u2fi[x]
```

■ Chunk 1

The first chunk which picks out the centre of the distribution (16 Δx)

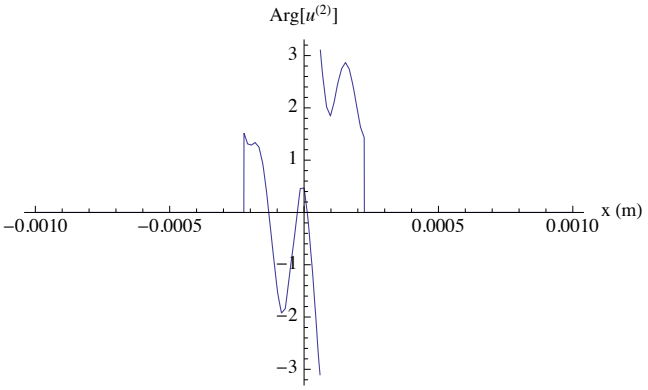
In[47]:=
```
Plot[Abs[u2f[x, 16 Δx]], {x, -0.001, 0.001},
 PlotRange -> All, AxesLabel → {"x (m)", "Abs[u(2)]"}]
```

Out[47]=



In[48]:=
```
Plot[Arg[u2f[x, 16 Δx]], {x, -0.001, 0.001},
 PlotRange -> All, AxesLabel → {"x (m)", "Arg[u(2)]"}]
```

Out[48]=



In[49]:=
```
Export["~/papers/frog/math/demo2chunk1.dat",
 Table[{N[i Δx], N[u2f[i Δx, 16 Δx]]}, {i, -NΔx / 2, (NΔx / 2 - 1)}]]
```

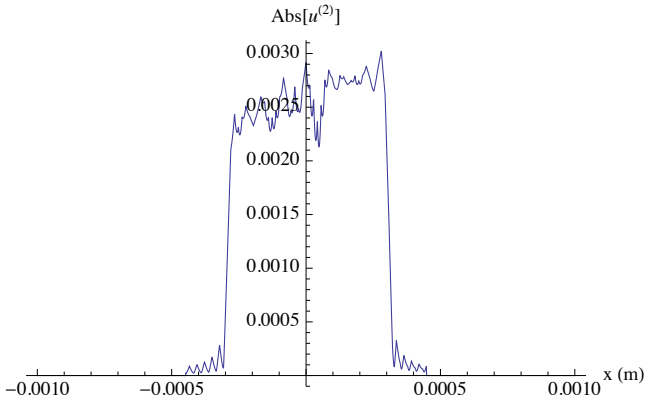
Out[49]=
```
~/papers/frog/math/demo2chunk1.dat
```

Chunk 2

The second chunk which encompasses the entire pulse (32 Δx)

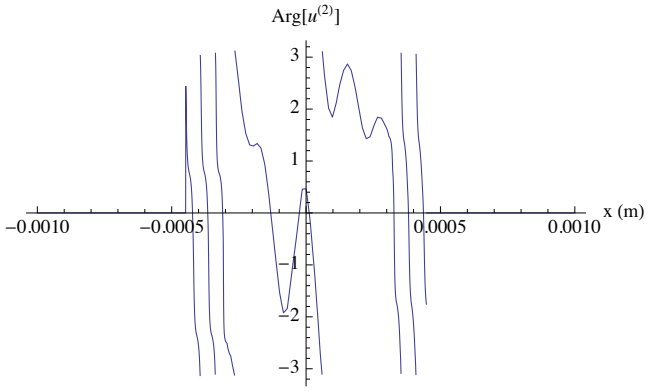
In[50]:=
```
Plot[Abs[u2f[x, 32 Δx]], {x, -0.001, 0.001},
 PlotRange -> All, AxesLabel → {"x (m)", "Abs[u(2)]"}]
```

Out[50]=



In[51]:=
```
Plot[Arg[u2f[x, 32 Δx]], {x, -0.001, 0.001},
 PlotRange -> All, AxesLabel → {"x (m)", "Arg[u(2)]"}]
```

Out[51]=



In[52]:=
```
Export["~/papers/frog/math/demo2chunk2.dat",
 Table[{N[i Δx], N[u2f[i Δx, 32 Δx]]}, {i, -NΔx / 2, (NΔx / 2 - 1)}]]
```

Out[52]= ~/papers/frog/math/demo2chunk2.dat

Chunks 1 and 2 will be analysed in demo2a.nb

Demonstration 2 using Type 1 Optics

Demonstration 2 produced chunk1 and chunk2 for the "Enhanced Bootstrap Algorithm" to be analysed here with the type I optics.

Type 1 Optics DFrFT

The Type 1 dimensionfull DFrFT is given by Eq (36)

In[1]:=
```
DFrFT[s_, n_, α_, Δt_, F_, λ_, m_] :=
  Block[{Δxp, F3}, F3 = (Sin[α] F); Δxp = λ F3 Sin[α] / (n Δt); Return[Exp[I α / 2] / Sqrt[I λ F3 Sin[α]] Δt
    Sum[s[k Δt] Exp[I π / (λ F3) ((m Δxp)^2 + (k Δt)^2) Cot[α] - I 2 π k m / n], {k, -n / 2, n / 2 - 1}]]];
```

Arguments of DFrFT[] are:

- s: function to be transformed
- n: number of points to be transformed. Must be power of 2
- α: order of the transform
- Δxp: pixel size of the CCD camera
- F: focal length of the lens
- λ: wavelength of the laser
- m: returns value of transform at m Δxp

It is assumed that the time step gives the frequency step with the relationship Δx Δxp= λ F1 Sin[α]/n.
For type I configuration, F1 = Q F and z = R F1 = RQ F. For type I: Q = sin[α] R = Tan[α/2]

Type I Optics Setup

Laser wavelength

In[2]:=
```
λ = 800 × 10^-9; (*m*)
```

Fixed "local" focal length

In[3]:=
```
F3 = 1; (*m*)
```

Focal length of the lens used for Type 1 optics for α and γ

In[4]:=
```
α = π / 2;
γ = π / 4;
```

In[6]:=
```
F3α = F3 / Sin[α] (*m*)
```

Out[6]= 1

In[7]:=
```
F3γ = F3 / Sin[γ] (*m*)
```

Out[7]= $\sqrt{2}$

Size of CCD pixel

In[8]:=
```
Δxpixel = 14 × 10^-6; (*m*)
```

Number of CCD pixels. Technically there are 2068 pixels, but just to make things a power of 2, we are setting it to 2048

In[9]:=
```
Npixel = 2048;
```

For convenience, set the sampling size as the CCD pixel size

In[10]:=
```
Δx = Δxpixel;
```

Define the normalization factor from Eq (37)

In[11]:=
```
μ = Sqrt[λ F3] // N
```

Out[11]=
```
0.000894427
```

Read in the chunk1 and chunk2 data

The input function sinput[t] which we had used previously

In[12]:=
```
Lchunk1 = ReadList["~/papers/frog/math/demo2chunk1.dat", {Number, Expression}];
Lchunk2 = ReadList["~/papers/frog/math/demo2chunk2.dat", {Number, Expression}];
```

Select out the signal part

In[14]:=
```
Lchunk1a = {};
For[i = 1, i ≤ Length[Lchunk1], i++,
  If[Abs[Lchunk1[[i, 2]]] ≠ 0,
    Lchunk1a = Join[Lchunk1a, {Lchunk1[[i]]}];
   ];
 ];
```

In[16]:=
```
fchunk1 = Interpolation[Lchunk1a, InterpolationOrder → 1]
```

Out[16]=
```
InterpolatingFunction[{{-0.000224, 0.000224}}, <>]
```

For numeric reasons, we are increasing the size of the signal by 10^3. The first aperture is +/- 16Δx

In[17]:=
```
fapchunk1[x_] := If[-16 Δx ≤ x < 16 Δx, 10^3 fchunk1[x], 0]
```

In[18]:=
```
Lchunk2a = {};
For[i = 1, i ≤ Length[Lchunk2], i++,
  If[Abs[Lchunk2[[i, 2]]] ≠ 0,
    Lchunk2a = Join[Lchunk2a, {Lchunk2[[i]]}];
   ];
 ];
```

In[20]:=
```
fchunk2 = Interpolation[Lchunk2a, InterpolationOrder → 1]
```

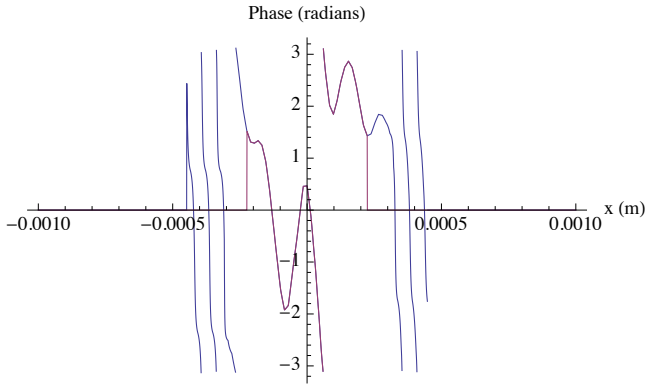
Out[20]=
```
InterpolatingFunction[{{-0.000448, 0.000448}}, <>]
```

The second aperture is +/- 32Δx

In[21]:=
```
fapchunk2[x_] := If[-32 Δx ≤ x < 32 Δx, 10^3 fchunk2[x], 0]
```

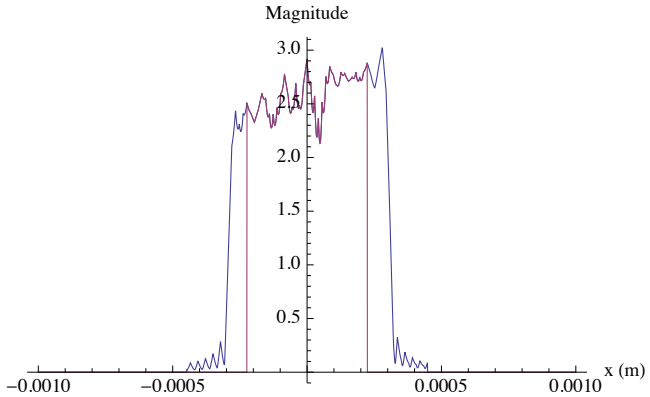
In[22]:=
```
Plot[{Arg[fapchunk2[x]], Arg[fapchunk1[x]]}, {x, -10^-3, 10^-3},
 PlotRange → All, AxesLabel → {"x (m)", "Phase (radians)"}]
```

Out[22]=



In[23]:=
```
Plot[{Abs[fapchunk2[x]], Abs[fapchunk1[x]]},
 {x, -10^-3, 10^-3}, PlotRange → All, AxesLabel → {"x (m)", "Magnitude"}]
```

Out[23]=



■ First measurement parameters with a small chunk

We select $\alpha = \pi/2$ and $\gamma = \pi/4$ and work on the first chunk

In[24]:=
```
nm = Npixel;
(*total number of samples which is
 approximately equal to the number of pixels on the CCD*)
n1 = 16 × 2;  (* number of samples where s is the real signal and not padding*)
α = π / 2;
γ =  π / 4;
Δt = Δx / μ; (*dimensionaless temporal variable created from Δx*)
```

Make dimensionless $\Delta f\alpha$ and $\Delta f\gamma$. Note: $\lambda F1/\mu = \mu$

In[29]:=
```
Δfα = λ F3 Sin[α] / (μ nm Δt)  // N;
Δfγ = λ F3 Sin[γ] / (μ nm Δt)  // N;
```

■ Create the I_α and I_γ from chunk 1

We need nm points of intensity from I_α and I_γ from sinput1[t].

In[31]:=
```
iα =
  Table[{m Δfα, Abs[ DFrFT[fapchunk1, nm, α, Δx, F3α, λ, m]]^2}, {m, -nm / 2, nm / 2 - 1}] // N;
iγ = Table[{m Δfγ, Abs[ DFrFT[fapchunk1, nm, γ, Δx, F3γ, λ, m]]^2},
    {m, -nm / 2, nm / 2 - 1}] // N;
```

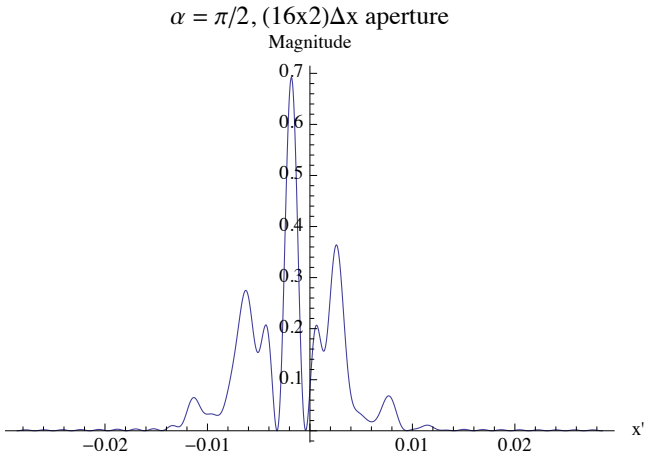
Make functions for the data sets

In[33]:=
```
Iα[f_] := iα[[Round[f / Δfα] + nm / 2 + 1]][[2]];
Iγ [f_] := iγ[[Round[f / Δfγ] + nm / 2 + 1]][[2]];
```

Plot them out

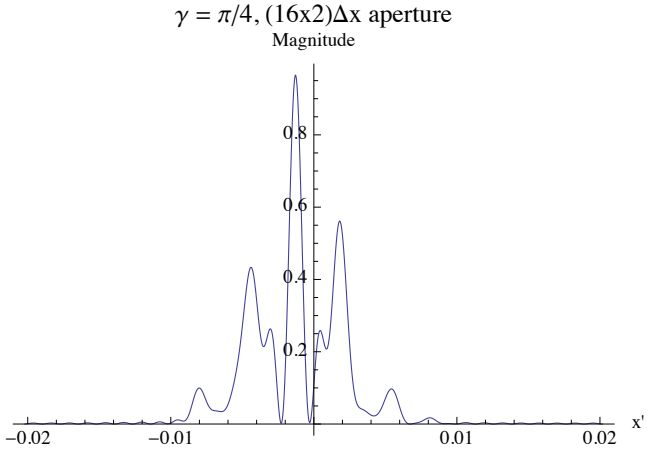
In[35]:=
```
ListPlot[iα, Joined → True, PlotRange → All,
 AxesLabel → {"x'", "Magnitude"}, PlotLabel → "α = π/2, (16x2)Δx aperture"]
```

Out[35]=



In[36]:=
```
ListPlot[iγ, Joined → True, PlotRange → All,
 AxesLabel → {"x'", "Magnitude"}, PlotLabel → "γ = π/4, (16x2)Δx aperture"]
```

Out[36]=



I_α and I_γ are the intensities which we measure. And note that the x-axis which is now frequency is dimensionless.

- **Let's normalize iα and iγ so that their power is 1**

In[37]:=
```
totalPower = Sin[α] / (nm (Δt)^2) Sum[iα[[i, 2]], {i, Length[iα]}];
```

```
In[38]:= iα = iα / totalPower;
         iγ = iγ / totalPower;
```

The Bootstrap Method

We define the equations for the bootstrap method defined as equation (19) in the paper.

```
In[40]:= u[θ_, m_] := Exp[I 2 π (-n1 / 2) (n1 - m) (Δt)^2 Cot[θ]];
         v[θ_, m_] := Exp[I 2 π (-n1 / 2  + m - 1) (n1 - m) (Δt)^2 Cot[θ]];
         Cα[k_] :=
           Sin[α] / (nm Δt^2) Exp[-I π (k Δt)^2 Cot[α]] Sum[Iα[m Δfα] Exp[I 2 π k m / nm], {m, -nm / 2, nm / 2 - 1}];
         Cγ[k_] := Sin[γ] / (nm Δt^2) Exp[-I π (k Δt)^2 Cot[γ]] Sum[Iγ[m Δfγ] Exp[I 2 π k m / nm], {m, -nm / 2, nm / 2 - 1}];
         Σ[θ_, m_] := Sum[ Conjugate[s[-n1 / 2 + j]]
              s[n1 / 2 - m + j] Exp[I 2 π (n1 - m) (-n1 / 2 + j) (Δt)^2 Cot[θ]], {j, 1, m - 2}];
         d[m_] := Det[( u[α, m]  v[α, m]
                        u[γ, m]  v[γ, m] )];
```

Check that $C\alpha[0] = C\gamma[0]$

```
In[46]:= Cα0 = Cα[0]
Out[46]= 1.
In[47]:= Cγ0 = Cγ[0]
Out[47]= 1.
```

Solving s[t]

First set s[t] = 0 for -n1/2 to -nm/2+1 and nm/2 to n1/2-1

```
In[48]:= For[i = -nm / 2, i ≤ -n1 / 2 - 1, i++,
           s[i] = 0;
         ];
         For[i = n1 / 2, i ≤ nm / 2 - 1, i++,
           s[i] = 0;
         ];
```

Without loss of generality, we will set $s[-N/2]=\sigma=1$ first and then use equation equation (25) to solve for $s[-n/2]$ because every s[n] is either multiplied or divided by $s[-n/2]$.

```
In[50]:= σ = 1;
In[51]:= cα = Chop[ Cα[n1 - 1] Exp[I π n1 (n1 - 1) Δt^2 Cot[α]]]
Out[51]= 0.0315972 + 0.0034225 i
In[52]:= cγ = Chop[Cγ[n1 - 1] Exp[I π n1 (n1 - 1) Δt^2 Cot[γ]]]
Out[52]= 0.0315972 + 0.0034225 i
```

Clearly $c\alpha$ and $c\gamma$ are equal as required.

Iteration

In[53]:=
```
s[-n1 / 2] = σ
```

Out[53]= 1

Average $c\alpha$ and $c\gamma$ because they are supposed to be the same.

In[54]:=
```
s[n1 / 2 - 1] = (cα + cγ) / (2.0 σ)
```

Out[54]= 0.0315972 + 0.0034225 i

Also set $c\alpha$ to the average since it is used below

In[55]:=
```
cα = s[n1 / 2 - 1]
```

Out[55]= 0.0315972 + 0.0034225 i

From equation (16), we can solve for s[n/2-2] and s[-n/2+1]

In[56]:=
```
s[n1 / 2 - 2] = Chop[First[1 / Conjugate[σ] d[2] (v[γ, 2] - v[α, 2]).(Cα[n1 - 2] / Cγ[n1 - 2])]][[1]]
```

Out[56]= 0.0273862 + 0.0150159 i

In[57]:=
```
s[-n1 / 2 + 1] =
 Chop[First[σ / Conjugate[cα d[2]] Conjugate[(-u[γ, 2] u[α, 2]).(Cα[n1 - 2] / Cγ[n1 - 2])]][[1]]]
```

Out[57]= 0.940244 - 0.203553 i

Now we can continue the bootstrap process until every element is done

In[58]:=
```
For[m = 3, m ≤ n1 / 2, m++,
  s[n1 / 2 - m] =
   Chop[First[1 / Conjugate[σ] d[m] (v[γ, m] - v[α, m]).(Cα[n1 - m] - Σ[α, m] / Cγ[n1 - m] - Σ[γ, m])]][[1]];
  s[-n1 / 2 + m - 1] = Chop[First[σ / Conjugate[cα d[m]]
      Conjugate[(-u[γ, m] u[α, m]).(Cα[n1 - m] - Σ[α, m] / Cγ[n1 - m] - Σ[γ, m])]][[1]]];
 ];
```

■ Calculate the magnitude s[-n/2]

We can calculate the magnitude of s[-n/2] using equation (25) of the paper. We will denote s[-n/2] = σ
σ2 MUST BE REAL so that s[-n/2] is REAL. See eq(25)

In[59]:=
```
s[-n1 / 2] = σ; (* placeholder for the above calculations *)
```

```
In[60]:= sol = Chop[Solve[σ2 Sum[Abs[s[-n1 / 2 + m - 1]]^2, {m, 1, n1 / 2}] +
             1/σ2 Sum[Abs[s[n1 / 2 - m]]^2, {m, 1, n1 / 2}] == Cα[0], σ2]]
Out[60]= {{σ2 → 0.028556}, {σ2 → 0.0331599}}
```

$\sigma 2$ must be real. So just select the real parts only for the calculations below

```
In[61]:= σfirst = Re[σ2] /. First[sol]
Out[61]= 0.028556

In[62]:= σlast = Re[σ2] /. Last[sol]
Out[62]= 0.0331599
```

We select between the two solutions by using equation (26)

s0 = s[-n/2] G$\alpha\gamma$[n/2 + 1]

```
In[63]:= firstsol1 =
           Sqrt[σ2] Chop[First[1/Conjugate[cα d[n1 / 2 + 1]] Conjugate[(-u[γ, n1 / 2 + 1] u[α, n1 / 2 + 1]).
             {{Cα[n1 - (n1 / 2 + 1)] - Σ[α, n1 / 2 + 1]}, {Cγ[n1 - (n1 / 2 + 1)] - Σ[γ, n1 / 2 + 1]}}]][[1]]] /. σ2 → σfirst
Out[63]= 0.0964345 - 0.171092 i
```

The other part which is also s[0] from equation (26)

s0 = $\frac{1}{s[-n/2]}$ F$\alpha\gamma$[$n/2$]

```
In[64]:= firstsol2 =
           1/Sqrt[σ2] Chop[First[1/d[n1 / 2] (v[γ, n1 / 2] - v[α, n1 / 2]).{{Cα[n1 - n1 / 2] - Σ[α, n1 / 2]}, {Cγ[n1 - n1 / 2] - Σ[γ, n1 / 2]}}][[
             1]]] /. σ2 → σfirst
Out[64]= 0.0964345 - 0.171092 i
```

- **Check the second solution**

This is a quick check that the other solution is not correct

```
In[65]:= secondsol1 =
           Sqrt[σ2] Chop[First[1/Conjugate[cα d[n1 / 2 + 1]] Conjugate[(-u[γ, n1 / 2 + 1] u[α, n1 / 2 + 1]).
             {{Cα[n1 - (n1 / 2 + 1)] - Σ[α, n1 / 2 + 1]}, {Cγ[n1 - (n1 / 2 + 1)] - Σ[γ, n1 / 2 + 1]}}]][[1]]] /. σ2 → σlast
Out[65]= 0.103918 - 0.184368 i
```

In[66]:=
```
secondsol2 =
 1/Sqrt[σ2] Chop[First[1/d[n1/2] (v[γ, n1/2] - v[α, n1/2]).(Cα[n1 - n1/2] - Σ[α, n1/2]
                                                           Cγ[n1 - n1/2] - Σ[γ, n1/2])]][[
    1]]] /. σ2 → σlast
```

Out[66]= 0.08949 - 0.158771 i

Select between the two solutions by looking at their differences

In[67]:=
```
If[Abs[firstsol1 - firstsol2] < Abs[secondsol1 - secondsol2],
 σ = Sqrt[σ2] /. σ2 → σfirst,
 σ = Sqrt[σ2] /. σ2 → σlast
]
```

Out[67]= 0.168985

■ Normalize s

Once we have σ, we can normalize all the s's

In[68]:=
```
For[m = 1, m ≤ n1/2, m++,
  sn[-n1/2 + m - 1] = σ s[-n1/2 + m - 1];
 ];
For[m = 1, m ≤ n1/2, m++,
  sn[n1/2 - m] = 1/σ s[n1/2 - m];
 ];
```

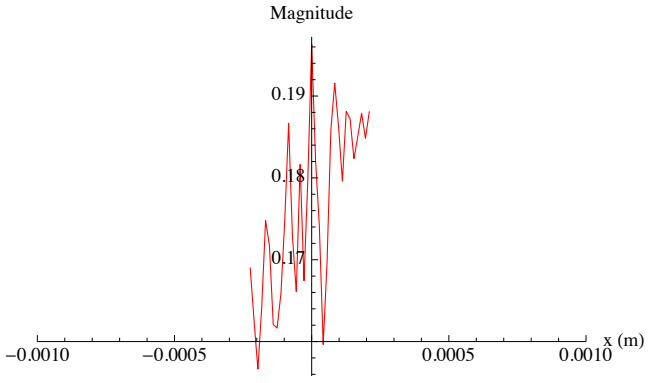
Plot out dimensionfull pictures

In[70]:=
```
sMag = Table[{k Δt μ, Abs[sn[k]]}, {k, -n1/2, n1/2 - 1}];
sArg = Table[{k Δt μ, Arg[sn[k]]}, {k, -n1/2, n1/2 - 1}];
sRe = Table[{k Δt μ, Re[sn[k]]}, {k, -n1/2, n1/2 - 1}];
sIm = Table[{k Δt μ, Im[sn[k]]}, {k, -n1/2, n1/2 - 1}];
```

■ Plots

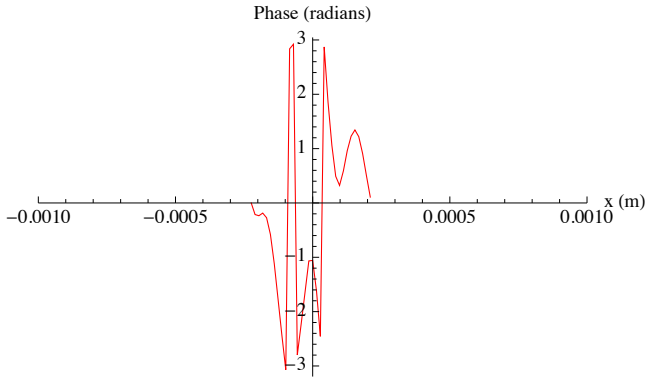
In[74]:=
```
ListPlot[sMag, Joined → True, PlotStyle → RGBColor[1, 0, 0],
 PlotRange → {{-10^-3, 10^-3}, All}, AxesLabel → {"x (m)", "Magnitude"}]
```

Out[74]=



In[75]:=
```
ListPlot[sArg, Joined → True, PlotStyle → RGBColor[1, 0, 0],
 PlotRange → {{-10^-3, 10^-3}, All}, AxesLabel → {"x (m)", "Phase (radians)"}]
```

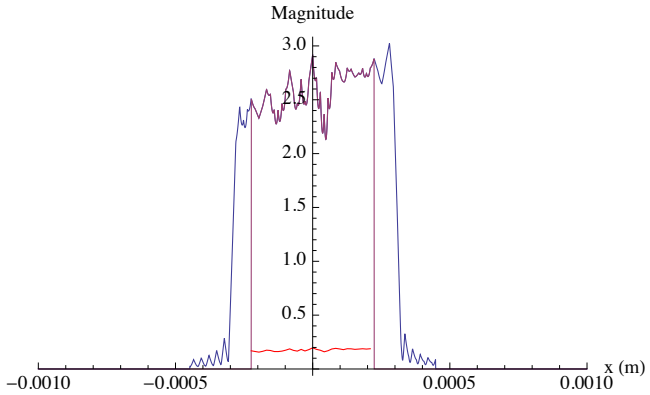
Out[75]=



- **Compare the result and the original chunk**

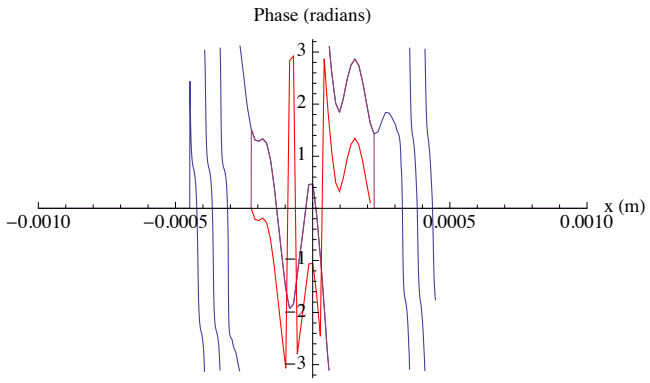
In[76]:=
```
Show[%23, %74, PlotRange → {{-10^-3, 10^-3}, All}]
```

Out[76]=



In[77]:=
```
Show[%22, %75, PlotRange → {{-10^-3, 10^-3}, All}]
```

Out[77]=



Enhanced Bootstrap Algorithm

Using the sn[t] which we calculated, we will extend the solution for the next chunk.

The chunk has been increased by a factor of 2 from n1 to n2=2*n1.

In[78]:=
```
n2 = 2 n1; (* number of samples where s is the real signal and not padding*)
```

■ Create the I_α and I_γ from chunk 2

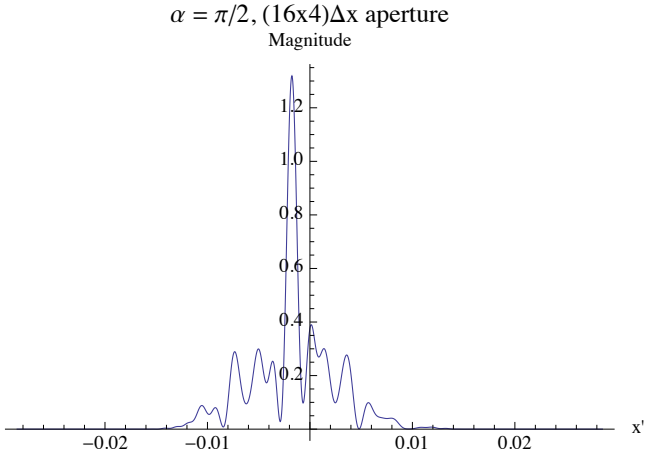
We need nm points of intensity from I_α and I_γ from sinput1[t].

In[79]:=
```
iα =
  Table[{m Δfα, Abs[ DFrFT[fapchunk2, nm, α, Δx, F3α, λ, m]]^2}, {m, -nm / 2, nm / 2 - 1}] // N;
iγ = Table[{m Δfγ, Abs[ DFrFT[fapchunk2, nm, γ, Δx, F3γ, λ, m]]^2},
    {m, -nm / 2, nm / 2 - 1}] // N;
```

Plot them out

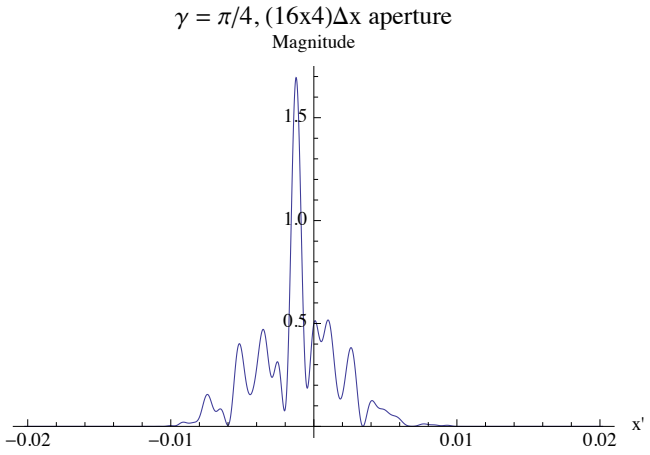
In[81]:=
```
ListPlot[iα, Joined → True, PlotRange → All,
 AxesLabel → {"x'", "Magnitude"}, PlotLabel → "α = π/2, (16x4)Δx aperture"]
```

Out[81]=



In[82]:=
```
ListPlot[iγ, Joined → True, PlotRange → All,
 AxesLabel → {"x'", "Magnitude"}, PlotLabel → "γ = π/4, (16x4)Δx aperture"]
```

Out[82]=



I_α and I_γ are the intensities which we measure.

■ Normalize w.r.t. the power from the initial bootstrap

In[83]:=
```
iα = iα / totalPower;
iγ = iγ / totalPower;
```

Create the Matrix Vector Equation

Clear previous definition of σ

```
In[85]:= Clear[σ];
         Clear[σs];
```

Define the equations that are the entries of the matrix S

```
In[87]:= σs[k_, l_, θ_] := Conjugate[sn[k]] Exp[i 2 π k l (Δt)^2 Cot[θ]];
         σ[k_, l_, θ_] := sn[k] Exp[i 2 π (k - l) l (Δt)^2 Cot[θ]];
```

Filling in the $S_\theta$matrix shown in Eq. (29)

```
In[89]:= Sα = Table[0, {i, n1 / 2 + 1}, {j, n1}];
         Sγ = Table[0, {i, n1 / 2 + 1}, {j, n1}];
         For[i = 1, i ≤ n1 / 2 + 1, i++,
           For[j = 1, j ≤ n1 / 2, j++,
             Sα[[i, j]] = σs[-n1 / 2 + (j - 1) + (i - 1), n1 - (i - 1), α];
             Sγ[[i, j]] = σs[-n1 / 2 + (j - 1) + (i - 1), n1 - (i - 1), γ];
            ];
          ];

         For[i = 1, i ≤ n1 / 2 + 1, i++,
           For[j = 1, j ≤ n1 / 2, j++,
             Sα[[i, j + n1 / 2]] = σ[(j - 1) - (i - 1), n1 - (i - 1), α];
             Sγ[[i, j + n1 / 2]] = σ[(j - 1) - (i - 1), n1 - (i - 1), γ];
            ];
          ];
```

The $C_\theta$vector showin in Eq. (31)

```
In[93]:= CCα = Table[0, {i, n1 / 2 + 1}, {j, 1, 1}];
         CCγ = Table[0, {i, n1 / 2 + 1}, {j, 1, 1}];
         For[i = 1, i ≤ n1 / 2 + 1, i++,

           CCα[[i, 1]] = Cα[n1 - (i - 1)] - Sum[Conjugate[sn[-n1/2 + j - 1]]
                sn[n1/2 + j - i] Exp[i 2 π (-n1/2 + j - 1) (n1 - (i - 1)) (Δt)^2 Cot[α]], {j, 1, i - 1}];

           CCγ[[i, 1]] = Cγ[n1 - (i - 1)] - Sum[Conjugate[sn[-n1/2 + j - 1]] sn[n1/2 + j - i]
                Exp[i 2 π (-n1/2 + j - 1) (n1 - (i - 1)) (Δt)^2 Cot[γ]], {j, 1, i - 1}];

         ];
```

Create the matrix equation which we solve for the unknown s[] entries Eq. (28)

In[96]:=
```
S = Table[0, {i, n1 + 2}, {j, n1}];
CC = Table[0, {i, n1 + 2}, {j, 1, 1}];
For[i = 1, i ≤ n1 / 2 + 1, i++,
  For[j = 1, j ≤ n1, j++,
    S[[i, j]] = Sα[[i, j]];
   ];
 ];
For[i = 1, i ≤ n1 / 2 + 1, i++,
  For[j = 1, j ≤ n1, j++,
    S[[i + (n1 / 2 + 1), j]] = Sγ[[i, j]];
   ];
 ];
For[i = 1, i ≤ n1 / 2 + 1, i++,
  CC[[i, 1]] = CCα[[i, 1]];
 ];
For[i = 1, i ≤ n1 / 2 + 1, i++,
  CC[[i + (n1 / 2 + 1), 1]] = CCγ[[i, 1]];
 ];
```

The number of equations is greater than the number of variables. Therefore, we'll have to use a least squares method to solve for s[]

In[102]:=
```
sol = LeastSquares[S, CC];
```

Now, get the answers to the unknowns using Eq (30)

In[103]:=
```
For[i = 1, i ≤ n1 / 2, i++,
  sn[n1 / 2 + (i - 1)] = sol[[i, 1]];
 ];
For[i = 1, i ≤ n1 / 2, i++,
  sn[-n1 + (i - 1)] = Conjugate[sol[[n1 / 2 + i, 1]]];
 ];
```

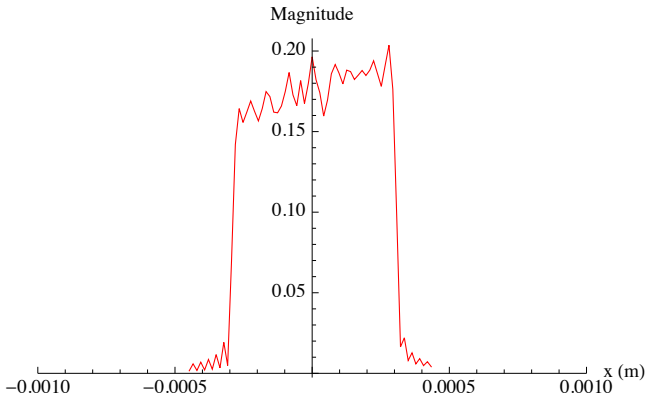
■ Plots

Plot out dimensionfull pictures

In[105]:=
```
sMag = Table[{k Δt μ, Abs[sn[k]]}, {k, -n2 / 2, n2 / 2 - 1}];
sArg = Table[{k Δt μ, Arg[sn[k]]}, {k, -n2 / 2, n2 / 2 - 1}];
sRe = Table[{k Δt μ, Re[sn[k]]}, {k, -n2 / 2, n2 / 2 - 1}];
sIm = Table[{k Δt μ, Im[sn[k]]}, {k, -n2 / 2, n2 / 2 - 1}];
```

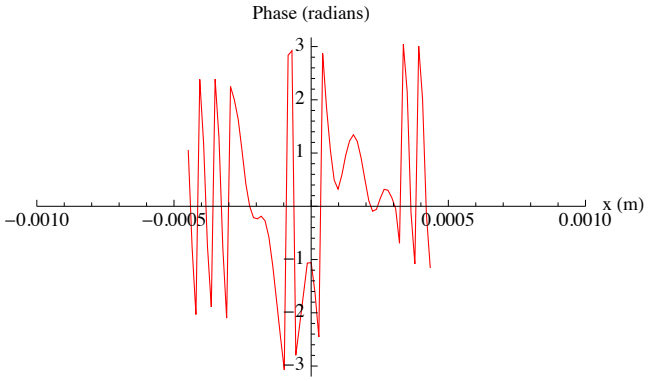
In[109]:=
```
ListPlot[sMag, Joined → True, PlotStyle → RGBColor[1, 0, 0],
 PlotRange → {{-10^-3, 10^-3}, All}, AxesLabel → {"x (m)", "Magnitude"}]
```

Out[109]=



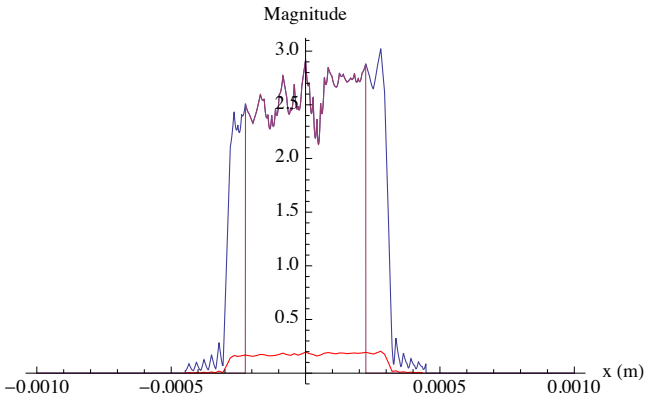
In[110]:=
```
ListPlot[sArg, Joined → True, PlotStyle → RGBColor[1, 0, 0],
 PlotRange → {{-10^-3, 10^-3}, All}, AxesLabel → {"x (m)", "Phase (radians)"}]
```

Out[110]=



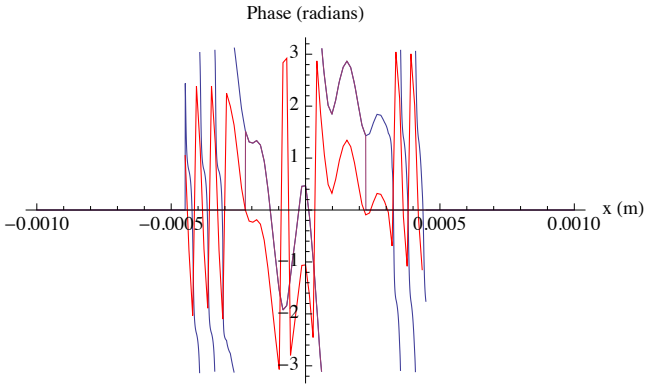
In[111]:=
```
Show[%23, %109, AxesLabel → {"x (m)", "Magnitude"}]
```

Out[111]=



In[112]:=
```
Show[%22, %110, AxesLabel → {"x (m)", "Phase (radians)"}]
```

Out[112]=



EBA works as expected, but quadratic phase needs to be corrected

Correct the quadratic phase

There is a phase contribution which was introduced by lens 1. See Eq (90). We can correct for this.

The size of the diffraction grating and incident angle

In[113]:=
```
a = 10^-3 / 1200.; (*m*)
```

In[114]:=
```
θi = 85 π / 180;
```

focal length of lens 1

In[115]:=
```
f1 = 0.05; (*m*)
```

focal length of lens 2

In[116]:=
```
f2 = 1.0; (*m*)
```

The magnification factor

In[117]:=
```
ν = f2 / f1
```

Out[117]=
```
20.
```

The wave number of the laser

In[118]:=
```
k = 2 π / λ;
```

From Example AII.2, Eq (81), we can calculate the number of samples

In[119]:=
```
c = 3 × 10^8; (* speed of light, m/s*)
```

In[120]:=
```
tp = 100 × 10^-15;
```

In[121]:=
```
pmax = IntegerPart[ c tp / (a Sin[θi]) ]
```

Out[121]=
```
36
```

The phase correction comes from Eq (61) and add in a possible phase offset

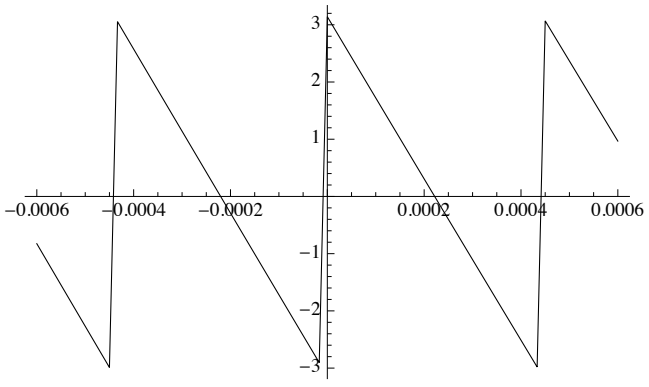
In[122]:=
```
phasecorr[x_, θ_] := Arg[ Exp[i θ] Exp[ i k x^2 / (2 f1) ] Exp[i k x Sin[θi]] ]
```

We generate a table of the phase corrections which are spaced a apart. Note that we are undersampling the phase correction and also the magnification factor ν (the negative sign in the argument of phasecorr[] comes from performing the Fourier transform twice, see Eq (40)). We anticipate that the phase offset is π.

In[123]:=
```
pcorr = Table[{p a ν, phasecorr[-p a, π]}, {p, -pmax, pmax}];
```

In[124]:=
```
ListPlot[pcorr, Joined → True, PlotStyle → RGBColor[0, 0, 0]]
```

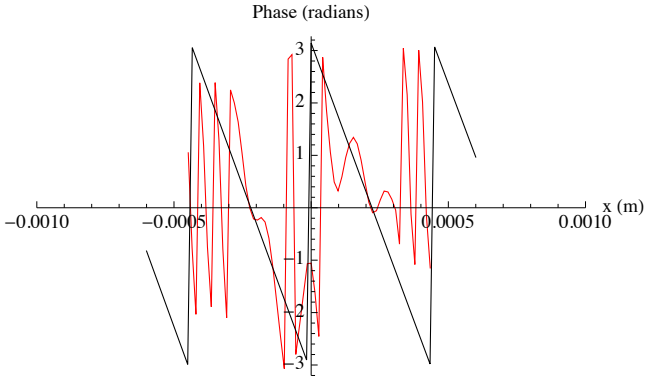
Out[124]=



Show that the phase matches what we have calculated

In[125]:=
```
Show[%110, %124]
```

Out[125]=



Create the phase correction function which includes the magnification factor from pcorr

In[126]:=
```
fcorr = Interpolation[pcorr, InterpolationOrder → 1]
```

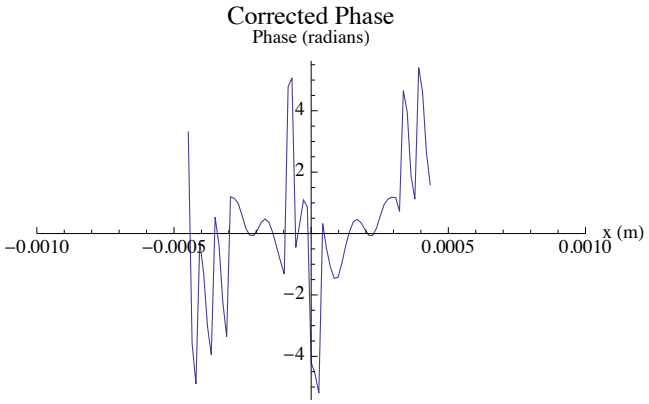
Out[126]=
```
InterpolatingFunction[{{-0.0006, 0.0006}}, <>]
```

Perform the corection

In[127]:=
```
corrArg = Table[{sArg[[i, 1]], sArg[[i, 2]] - fcorr[sArg[[i, 1]]]}, {i, Length[sArg]}];
```

In[128]:=
```
ListPlot[corrArg, Joined → True, PlotRange → {{-10^-3, 10^-3}, All},
 AxesLabel → {"x (m)", "Phase (radians)"}, PlotLabel → "Corrected Phase"]
```

Out[128]=



■ Unwind the phase

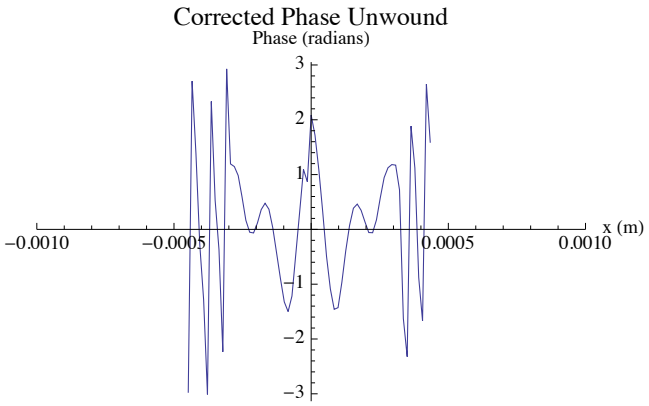
There are phases which outside the range +/- π. We can correct this with phaseUnwind below.

In[129]:=
```
phaseUnwind[θ_] := If[-π ≤ θ ≤ π, Return[θ],
    If[θ < -π, Return[θ + 2 π], Return[θ - 2 π]]
  ];
```

In[130]:=
```
ucorrArg = Table[{corrArg[[i, 1]], phaseUnwind[corrArg[[i, 2]]]}, {i, Length[corrArg]}];
```

In[131]:=
```
ListPlot[ucorrArg, Joined → True, PlotRange → {{-10^-3, 10^-3}, All},
 AxesLabel → {"x (m)", "Phase (radians)"}, PlotLabel → "Corrected Phase Unwound"]
```

Out[131]=



Notice the small discontinuity near the centre of the corrected phase. This is comes from imperfect phase correction.

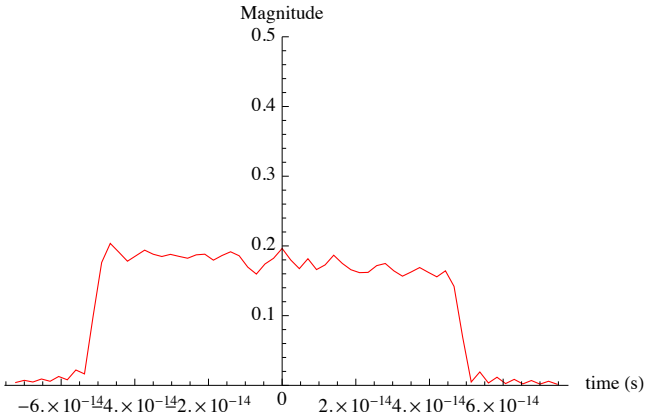
Comparing with the input pulse

Rescale the magnitude and phase by ν to take out the magnification from Lens1 and Lens2 and change it to time on the xaxis. Also reverse the x axis, i.e. x-> -x because of the mirroring of the object by lens1 and lens2.

In[132]:=
```
rescaleMag = Table[{-sMag[[i, 1]]/(c ν), sMag[[i, 2]]}, {i, Length[sMag]}];
rescaleArg = Table[{-ucorrArg[[i, 1]]/(c ν), ucorrArg[[i, 2]]}, {i, Length[corrArg]}];
```

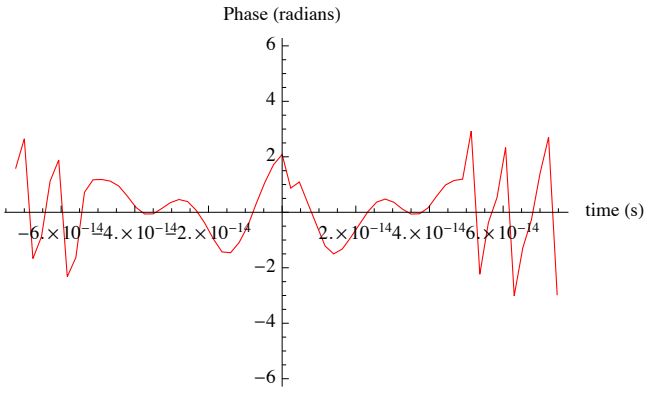
In[134]:=
```
ListPlot[rescaleMag, Joined → True, PlotStyle → RGBColor[1, 0, 0],
 PlotRange → {0, 0.5}, AxesLabel → {"time (s)", "Magnitude"} ]
```

Out[134]=



In[135]:=
```
ListPlot[rescaleArg, Joined → True, PlotStyle → RGBColor[1, 0, 0],
 PlotRange → {-2 π, 2 π}, AxesLabel → {"time (s)", "Phase (radians)"} ]
```

Out[135]=



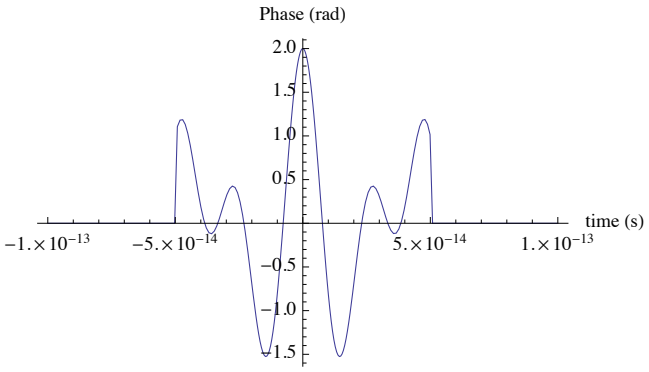
■ Input pulse

Recreate the input pulse so that we can compare with the reconstructed pulse.

In[136]:=
```
sinput[t_] :=
  η[t] 0.1 Exp[-0.2 (t / (tp) + 0.5)^2] Exp[I (Sin[5 π (t / (tp) + 0.5)] + Cos[8 π (t / (tp) + 0.5)])];
η[t_] := If[-0.5 ≤ t / (tp) < 0.5, 1, 0];
```

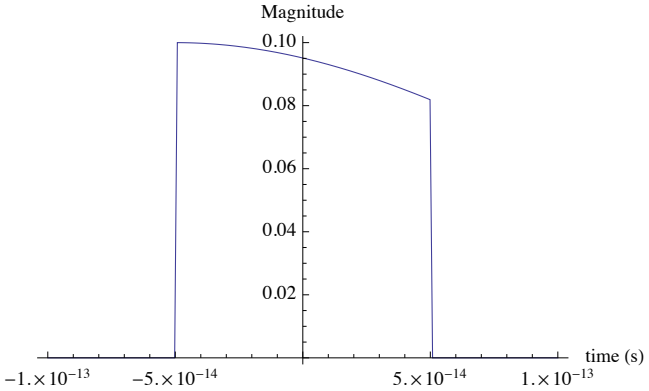
In[138]:=
```
Plot[Arg[sinput[t]], {t, -tp, tp},
 PlotRange → All, AxesLabel → {"time (s)", "Phase (rad)"}]
```

Out[138]=



In[139]:=
```
Plot[Abs[sinput[t]] , {t, -tp, tp}, AxesLabel → {"time (s)", "Magnitude"}]
```

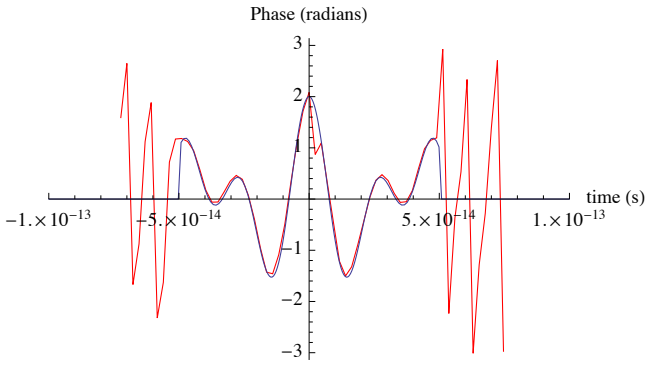
Out[139]=



Comparison

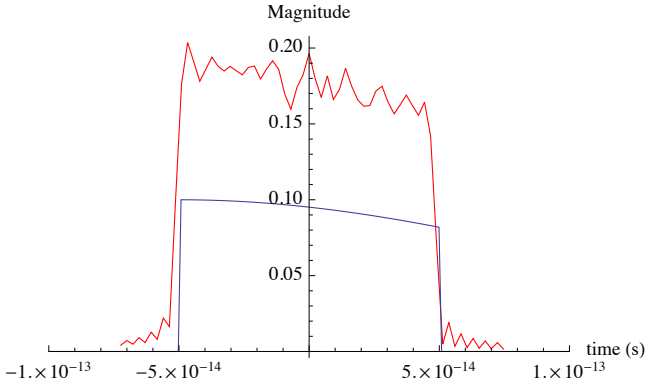
In[140]:= `Show[ %135, %138, PlotRange → {{-10^-13, 10^-13}, {-π, π}}]`

Out[140]=



In[141]:= `Show[%134, %139, PlotRange → {{-10^-13, 10^-13}, All}]`

Out[141]=



Everything looks good! Magnitude needs rescaling as expected. Phase match looks great.

REFERENCES

[1] D.J. Kane and R. Trebino, *Single-Shot Measurement of the Intensity and Phase of an Arbitrary Ultrashort Pulse By Using Frequency-Resolved Optical Gating.*, Opt. Lett., 18(10), pg. 823 – 825, 1993.

[2] H.M. Ozaktias *et al*, *Introduction to the Fractional Fourier Transform and its Applications*, Chapter 4, John Wiley & Sons, 2001.

[3] O. Akay and G.F. Boudreaux-Bartels, *Fractional Autocorrelation and its Applications to Detection and Estimation of Linear FM Signals*, IEEE, pg. 213 – 216, 1998.

[4] R.W. Gerchberg and W.O. Saxton, *A Practial Algorithm for the Determination of Phase from Image and Diffraction Plane Pictures*, Optik (Stuttgart) **35**, pg. 237−246, 1972.

[5] W.-X. Cong *et al*, *Recursive Algorithm for Phase Retrieval in the Fractional Fourier Transform Domain*, App. Opt., Vol. 37, No. 29, pg. 6906 – 6910, 1998.

[6] A.W. Lohmann, *Image Rotation, Wigner Rotation, and the Fractional Fourier Transform*, J. Opt. Soc. Am. A **10**, pg. 2181 – 2186, 1993.

[7] J.W. Goodman, *Introduction to Fourier Optics, 2nd Ed.*, McGraw-Hill, pg. 78 – 81, 1996.

[8] R. Trebino *et al*, *Measuring Ultrashort Laser Pulses in the Time-Frequency Domain Using Frequency-Resolved Optical Gating*, Rev. Sci. Instrum., Vol. 68, No. 9, 1997.