Chapter 2

RMPI

This chapter introduces the Random Modulation Pre-Integrator (RMPI) and details the status of an integrated circuit implementing the architecture. To our knowledge, this is the first full-speed hardware RMPI in existence. The goal of this chapter is to describe some key parameters of the RMPI that are essential to its performance. This is not a hardware chapter and does not go into detail of the designs; the reader does not need any knowledge of Op-amps, though standard mathematical and signal processing background is helpful. The idea behind the parameter selection is that of trade offs. We presuppose that a competent engineer is able to make several designs, and the task is to decide which design is better.

This introduction describes the setting of the RMPI and its basic principles, and tries to make clear the intuition behind the receiver, as well as outline when the RMPI is an appropriate architecture and when it is not. Further sections of the chapter describe the system in more detail, and report a some of the many results from 3 years of study and simulation.

The hardware implementation of the RMPI has been returned from the manufacturer, and is in the process of being calibrated, which has required significant work (to be discussed in §2.3.4), but results are encouraging. An earlier chip, which we refer to as "version 1," was manufactured and some reconstructed pulses obtained, but otherwise all plots in this chapter are from simulated data. The simulations took place at various levels of realism, which will are described in §2.3. Most computational work was performed in Matlab and/or Simulink.

The RMPI is due to the work of a large number of people, but we especially want to mention the efforts of Juhwan Yoo. He has done most of the hardware design, collaborated on the system design, and contributed figures and occasional text to this chapter. Funding for work was graciously provided by DARPA grant FA8650-08-C-7853.

2.1 Introduction

2.1.1 Signal class

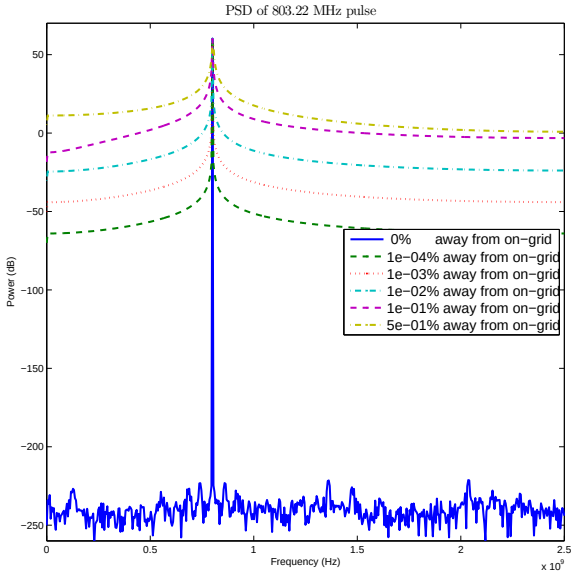


Figure 2.1: The problem of spectral leakage. Shown are FFT values. The FFT is a bad way to do spectral estimation [Tho82].

The data acquisition of the RMPI works well for several signal models, but each signal model has certain issues, so we restrict ourselves to a specific model. The goal of this RMPI is recovery of radar pulses, so our signal model is tailored to this. Let $x(t)$ be a continuous time signal, and $x[n] = x(n\Delta T)$ be a sampled version, where $\Delta T \triangleq 1/f_s$ is the Nyquist time; samples are not actually acquired at this resolution, but this Nyquist rate version of $x[n]$ is used in the digital post-processing. See Table 2.1 for a reference of the notation that is used.

The first assumption is that if the signal $x(t)$ can be characterized if we limit time to an interval $[0, T]$. This is true if $x(t)$ is periodic with period T, or if $x(t)$ is zero outside a narrow time window, as it is for a short radar pulse. An implicit requirement is that T is not too great. The postprocessing stage can handle up to about $N = 8192 = 2^{13}$ length vectors in the case of radar signals, and up to perhaps $262144 = 2^{18}$ for pure tones; the radar pulses are limited because of the size of the Gabor dictionary used in analysis, while pure tones use a simpler over-sampled FFT dictionary. Typical simulations in this chapter work with $N = 1024$ to $N = 8192$, so the corresponding time periods are 204.8 ns to 1.638 μs.

The next assumption is actually at odds with the the first: it is assumed that $x(t)$ is bandlimited to the range $[-f_s/2, f_s/2]$. This appears to rule out radar pulses, since it is impossible for signal to have compact support in both time and frequency. However, we simply consider the periodic extension of a radar pulse. By forcing the signal to be periodic, it means the spectrum is discrete

and given by the discrete Fourier transform (DFT).

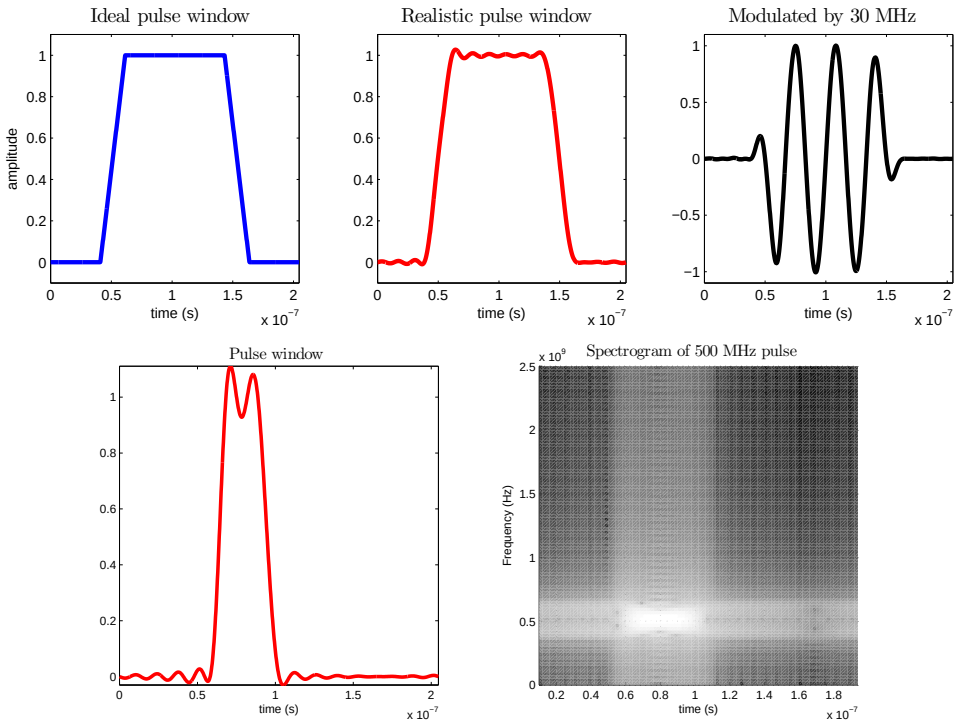


Figure 2.2: Radar pulses. Top row: on the left and middle, two sample pulse windows. On the right, a radar pulse in the time domain. Bottom row: sample pulse window on left, and on right, same pulse, modulated at 500 MHz, in the time-frequency plane

The pragmatic definition of the signal class is any signal that is well-approximated by N of its Nyquist-rate samples. One type of signal which does *not* belong to this class is the pure tone that has frequencies which are not on the periodic grid that has been imposed by the finite number of samples. Such a tone is called "off-grid," and Figure 2.1 shows the DFT of such tones. The frequency grid runs from 0 to $f_s/2$ in spacings of $\Delta f = 1/T$, which makes it clear that as $T \to \infty$, the spacing approaches 0 and the signal class approaches all band-limited signals. Viewed another way, the sampling from $[0, T]$ is like windowing a signal with the boxcar filter, so the spectrum is convolved with the Fourier transform of the boxcar filter, which is the sinc function. The sinc function has slow decay, so the off-grid tones cannot be well-approximated by just N Nyquist-rate samples. In §2.7.2.6 we discuss the problem further and propose solutions using windows. For now, these signals are not part of the signal model and do not concern us.

Signals that do fit this category are radar pulses, as shown in Figure 2.2. A radar pulse consists of an envelope, modulated by a carrier frequency. This modulation is just time-domain multiplication. The carrier is in the RF band between $[0, f_s/2]$, and is necessary for transmission. Other than the carrier frequency, the information content of a pulse is its time-of-arrival and duration, since these are used in radar systems to estimate range, angle, and speed. In order for time-of-arrival estimation

to be consistent, it is desirable for a pulse to have a well-defined leading edge that rises sharply, and hence pulses are approximately just trapezoids or even rectangles. However, a trapezoid is similar to the boxcar function, and so it has high spectral content due to the sharp corners. Radar emitters have only so much frequency bandwidth, so the radar pulse envelope is bandpass filtered either intentionally or unintentionally. This is beneficial for the RMPI system, since it makes the pulse sparser in frequency. This fact has been taken into account in pulse simulations. Another deviation from the trapezoidal model is that receivers often see multiple reflections of a signal, which means that the trailing edge of the pulse is generally degraded. This is not accounted for in our simulations, but it is not expected to significantly affect performance.

Looking at Figure 2.2, it is apparent that the radar signals are slightly sparse in time, and slightly sparse in frequency, but very sparse in the time-frequency plane. Using a time-frequency dictionary is essential to recovery, and is discussed in §2.7.2.2. The pulses considered in the model have duration varying between 100 and 1000 ns, and typically 200 ns.

Having stated the signal model, what is the goal of the system? The goal is an ambitious one: full reconstruction of the signal up to the accuracy permitted by the finite model. This means not just frequency domain estimation, but also time-domain estimation. Of all the works discussed in the related literature §2.2.3, none of them attempt time-domain reconstruction. We don't preclude the possibility of recovering multiple pulses simultaneously. Simulation results indicate that recovering two pulses of the same size, even if they overlap in time, is not much different than recovering a single pulse. The biggest challenge lies with recovering two pulses of extremely disparate size; this is discussed in §2.8.

In addition to this primary goal, a secondary goal is more practical: we wish to estimate the pulse parameters, termed "pulse descriptor words" (PDW), which are significant for radar processing. These include carrier frequency, phase, time-of-arrival, duration, and pulse repetition rate. For this kind of estimation, we propose a "compressive matched filter" (see [ERW11] for very recent results), discussed further in §2.7.1.

When is the RMPI a useful system?

- For high bandwidth designs, requiring 1 GHz bandwidth or more.
- When the input is sparse in the time-frequency plane; for example, it might consist of a few narrowband communication channels and a few radar pulses.
- For applications that need high precision, due to high dynamic range. For example, one might be interested in recovering very faint radar pulses that occur concurrently with some very large signals from a cell phone tower or airport.
- For applications that have power restrictions, so that massive filter banks are not an option. An example would be an airborne systems with a tight power budgets.

- For systems that transmit signal information to a central processing location. With the RMPI, the compressed measurements, or even the PDW, are transmitted, so less channel capacity is needed. The Herschel satellite telescope is an example: it has limited capability for processing, and a low-rate down-link channel to the Earth, so it sends compressed measurements and processing is done on the ground.

- For identifying unconventional signals, such as ultra-wideband spread spectrum. A channelized receiver is limited to analyzing 50 MHz (or similar) of spectrum at a time, while the RMPI analyzes the entire bandwidth simultaneously, and can make use of prior knowledge about the signal.

- For noisy environments that contain signals with relatively low-information content (such as so-called "finite rate of innovation" signals). The RMPI recovery process applies digital denoising, so for some signals, the RMPI reconstruction is much more accurate than even high ENOB Nyquist-rate measurements, since a conventional receiver does not denoise.

- Similarly, spectral estimation [Tho82] with the RMPI can be more accurate then conventional spectral estimation, since compressed spectral estimation [DB10] is done in digital and takes into account all the measurements at once.

- Other estimation, such as angle-of-incidence, can be done in the compressed framework, and like spectral estimation, they may benefit from using all measurements at once.

The RMPI is not appropriate for all applications. One of the current drawbacks is that it requires a computer for the recovery and does not yet process data in real-time (meaning that it takes the computer longer than time T to process the data generated from T worth of samples). These limitations are not fundamental but rather just a limit due to the current state of technology. The compressive matched filter is a promising technique for accelerating computation, and it is conceivable that this could be performed in real-time in hardware or via an FPGA.

Representation of the system. Characterizing the system will be the subject of §2.3.4, but one aspect of this is conceptually very important, so we mention it now. Because of the sampling, our system is not time-invariant, so we cannot analyze it in a pure linear-time-invariant (LTI) framework. However, it is linear (or at least approximately so; this is discussed in §2.7.2.5). This is fundamental for many reasons. First, it gives us a way to calibrate our system and determine how it is really functioning. It also allows us to write down the measurements in the linear inverse framework described in the introduction:

$$b = \Phi x, \quad \text{or, in the noisy case,} \quad b = \Phi(x + \sigma_1 z_1) + \sigma_2 z_2.$$

Notation	Typical values	Fixed or variable	Description
f_s	5 GHz	fixed	Nyquist rate
$\Delta T \triangleq 1/f_s$	200 ps	fixed	Nyquist period
N	1024, 4096	variable	length of digital signal x_n
$T \triangleq N\Delta T$	204 ns, 820 ns	variable	period of digital signal
$c(t)$	NA	NA	chipping sequence with PRBS $\{c_n\}$
N_{chip}	128	variable	periodicity of chipping sequence $\{c_n\}$
$T_{\text{chip}} \triangleq N_{\text{chip}}\Delta T$	25.6 ns	variable	period of $c(t)$, assuming $f_{\text{chip}} = f_s$
$1/T_{\text{chip}}$	39 MHz	variable	frequency of repetition of $c(t)$
f_{chip}	f_s	mainly fixed	rate of modulation with c_n
N_{int}	100	mainly fixed	length of integration in units of ΔT
$f_{\text{ADC}} \triangleq f_s/N_{\text{int}}$	50 MHz	mainly fixed	sampling rate of ADCs
$T_{\text{ADC}} \triangleq 1/f_{\text{ADC}}$	20 ns	mainly fixed	period of ADC samples
$T_{\text{int}} \triangleq T_{\text{ADC}}$	20 ns	mainly fixed	synonym for T_{ADC}
r_{ch}	8	mainly fixed	number of channels

Table 2.1: Notation used with the RMPI. We will use $i = \sqrt{-1}$ (rather than j). "Fixed or variable" refers to whether we often change the value of the parameter. For example, determining an appropriate value of r_{ch} is the topic of §2.4.3, but is usually fixed at $r_{\text{ch}} = 8$ in the rest of the sections.

Variant	Time	Frequency
Fourier Transform, $\hat{x} = \mathcal{F}x$	$x(t) = \int_{-\infty}^{\infty} \hat{x}(f) e^{i2\pi ft} df$	$\hat{x}(f) = \int_{-\infty}^{\infty} x(t) e^{-i2\pi ft} dt$
Fourier Series	$x_T(t) = \sum_{k=-\infty}^{\infty} \hat{x}[k] e^{i2\pi kt/T}$	$\hat{x}[k] = \frac{1}{T}\int_0^T x_T(t) e^{-i2\pi kt/T} dt$
DFT, $\hat{x} = Fx$	$x_n = \frac{1}{N}\sum_{k=0}^{N-1} \hat{x}_k e^{i2\pi kn/N}$	$\hat{x}_k = \sum_{n=0}^{N-1} x_n e^{-i2\pi kn/N}$
Unitary DFT, $\hat{x} = F_u x$	$x_n = \frac{1}{\sqrt{N}}\sum_{k=0}^{N-1} \hat{x}_k e^{i2\pi kn/N}$	$\hat{x}_k = \frac{1}{\sqrt{N}}\sum_{n=0}^{N-1} x_n e^{-i2\pi kn/N}$

Table 2.2: Variants of the Fourier transform. Listed to show normalization and notation conventions. The notation x_T means x is periodic with period T. Frequency variables written as f are in Hz, and written as ω are angular frequencies in radian/s. The notation $\hat{x}$ is used for both $\mathcal{F}x$, and occasionally to denote an estimate of x; context should distinguish the two uses. DFT stands for discrete Fourier transform, and is synonymous with the fast Fourier transform (FFT).

The system itself is represented by the Φ matrix, and we make much use of this fact. The linearity is also important since all our reconstruction algorithms assume linear measurements.

We also make use of many symbols, which have been collected in Table 2.1, and also speak of various types of Fourier transforms. Table 2.2 shows the normalization conventions used.

2.2 The design

2.2.1 Basic design

The design of the RMPI can be motivated by considering our goals, and then reasoning about what is necessary. At first, we need only consider a single channel, since extra channels simply give more information of the same type.

Overall, we aim to digitally sample a wideband signal at a low-rate. The incoming bandwidth to the system is $f_s/2 = 2.5$ GHz, and the ADC samples at 50 MHz, which is $100\times$ slower than the Nyquist rate f_s. We allow a few channels, but 100 channels is impractical, since this would have the same giant power consumption as traditional channelized receivers, so a filter bank approach is not valid, and thus it is fair to assume that the input to a single channel has the full bandwidth of 2.5 GHz. Sampling this at 50 MHz causes aliasing, so that an input of frequency f_{in} and an input of $f_{in} + 25$ MHz are indistinguishable.

In order to make these inputs distinguishable, the signals must be modified at some point before the ADC. A reasonable strategy is to mix the signals with another known signal; this is equivalent to multiplication in the time domain (note that *adding* another signal would be no use, since the ADC sampling is linear). Let this known signal be called the "chipping sequence," to borrow the term from spread-spectrum design [PSM82], and denote it by $c(t)$. Given two distinct signals x and x', we desire that $x(t)c(t)$ and $x'(t)c(t)$ are distinguishable even after being aliased by 50 MHz sampling.

Unfortunately, this is not possible just via mixing. Suppose the signals x and x' are distinct, but become identical after aliasing, which means $x(kT_{\text{int}}) = x'(kT_{\text{int}})$ for any k. Then clearly multiplying x and x' by the same signal c will not resolve the situation. In order to distinguish the two signals, the samples must involve information from in between $(k-1)T_{\text{int}}$ and kT_{int}. Let the samples of our function be $y = y(x)$ and $y' = y(x')$. Linear samples y_k can be represented by convolving with some kernel h known as the transfer function:

$$y_k = \int_{(k-1)T_{\text{int}}}^{kT_{\text{int}}} c(t)x(t)h(T_{\text{int}} - t)dt.$$

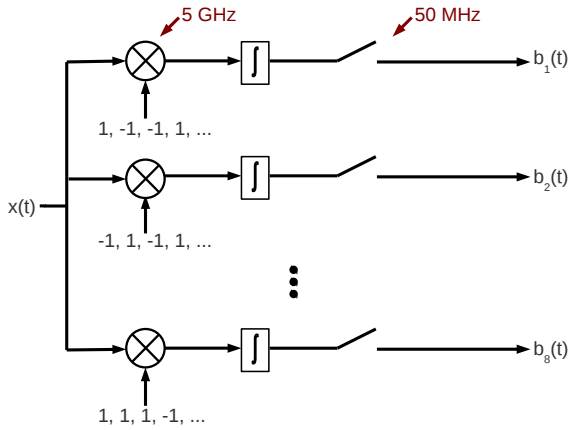


Figure 2.3: Principles of the RMPI. Each horizontal section represents a channel, and each channel has a unique PRBS sequence of ± 1's. After integration, each channel is sampled at 50 MHz.

For now, let $h = 1$ and consider

$$y_k - y_k' = \int_{(k-1)T_{\text{int}}}^{kT_{\text{int}}} (x(t) - x'(t))c(t)dt = \langle (x - x'), c \rangle$$

as a linear operator $\langle (x - x'), \cdot \rangle$. Thus $y_k = y_k'$ if c is in the null space of this operator. If $x \neq x'$ (by which we mean they differ on a set of nonzero measure), then this is not the zero operator, so there is a non-trivial complement of the null space (assume $x, x' \in L_2[(k-1)T_{\text{int}}, kT_{\text{int}}]$). Though we do not make the notion precise, a "random" function $c(t)$ drawn from L_2 is almost surely not entirely in the null space of this operator, and so $y_k \neq y_k'$.

The RMPI is born from this idea of making all signals have a unique signature after the mixing and integration. It is also motivated by the approximation of a signed Bernoulli matrix, as was mentioned in the introduction. Before going into further details of the system, we pause to codify the two guiding principles of the RMPI, which will be essential to bear in mind as we depart from the idealized RMPI to make a realistic system.

Principle 2.2.1 (Uniqueness)**.** *All distinct signals in the signal class should produce distinct measurements.*

Restriction to the signal class is required, since for undersampling, it is not possible to produce distinct measurements for all signals.

The next principle means that noise will effect all measurements about the same.

Principle 2.2.2 (Democracy)**.** *All equi-energy inputs should produce measurements of similar energy.*

These ideas can be interpreted as the RMPI analog of the "minimum distance" concept used in coding theory [McE02].

Practical version. The first design question for the RMPI is choosing a suitable chipping sequence $c(t)$. Since c must be known to the reconstruction algorithm, it must be predictable, and furthermore, it must be implementable. A reasonable choice is choosing c to be a pseudo-random bit sequence (PRBS), which flips pseudo-randomly between $+1$ and -1. The flips always occur at a multiple of the Nyquist grid ΔT, so that they may be controlled by a digital clock running at f_s. Technology for generating a chip sequence is available since this is the same principle behind spread-spectrum systems [PSM82]. In fact, the RMPI may be thought of as a spread spectrum device, using spreading in order to increase robustness to jamming and interference from foreign signals. Figure 2.4 shows a frequency domain representation of what this mixing looks like. The PRBS is discussed in detail in §2.5. Since our PRBS operates at the rate f_s, the RMPI system is a high-rate system, and it may appear to have just shifted the burden from high-rate sampling to high-rate mixing. This is true, but high-rate mixing is actually much easier than high-rate sampling, so the trade is worth it. The clock jitter introduced in the mixing problem is an issue, so our design is careful to keep the standard deviation of jitter to less than 0.5 ps; see §2.7.2.7.

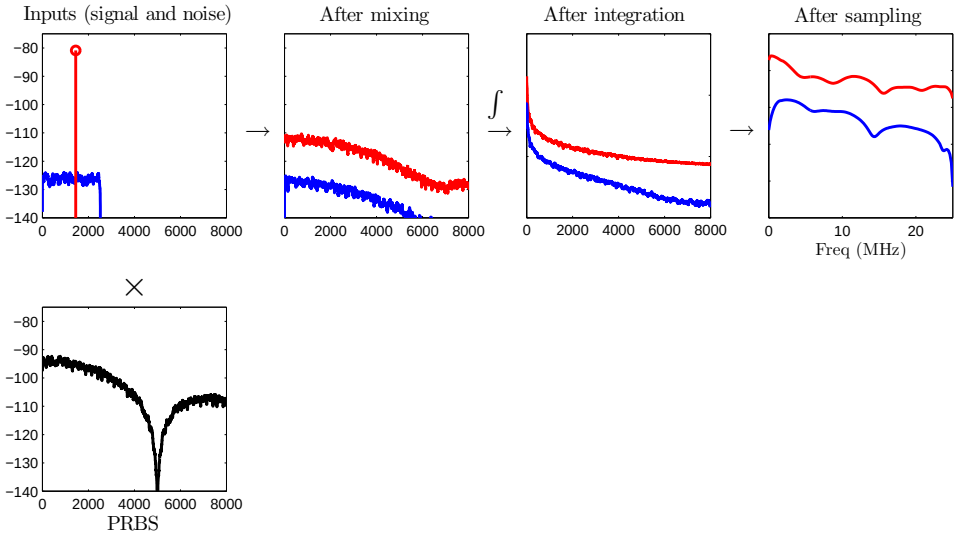


Figure 2.4: Frequency domain picture of one channel of the RMPI system. Red represents a signal, blue represents band-limited noise. The inputs are shown in the upper left, and are then convolved with the spectrum of the PRBS (shown below in black). Continuing to the right, the plots represent the signal after this mixing, then after integration (which attenuates high frequencies; the y-axis scale is shifted here), and then after the low-rate sampling which aliases high frequencies down to low frequencies. The x-axis of every plot is in MHz.

The next stage of the device is integration. This relies on capacitors, which actually charge and discharge at exponential rates. If this rate is too fast, then the integration becomes similar to a point sample, and as we argued above, point samples do not distinguish between pulses that have carrier frequencies separated by multiples of f_{ADC}. Another reason point samples are bad is that our signals include pulses, so they are sparse in time, and any sparse sampling mechanism will be coherent with these signals. For example, a pulse with duration less than T_{ADC} could possibly be

missed altogether if it was directly sampled (even after the mixing stage). To think about this in the Fourier domain, an ideal integrator has transfer function

$$H_{int}(s) = \frac{1}{s}$$

which means it attenuates high frequencies. A capacitor is similar to single pole filter

$$H(s) = \frac{1}{s+a}$$

for a pole frequency a. At high frequencies, this has little effect, but at low frequencies $f \ll a$, it means the response is nearly flat, like a band-pass filter. This can cause problems: in the extreme case, if the response is flat in the whole band, it means that $h(t)$ is a dirac, and this is just point sampling, which performs poorly. The location of this pole is a major factor of determining the performance of the system, and is described in detail in §2.6.

We briefly mention one more component of a practical RMPI, the low-noise amplifier (LNA). This amplifies the input signal from an antenna to make it suitable for processing. Though not part of the actual compressed sensing of the chip, it is still a crucial component in a useful hardware device, and it is what makes the RMPI a true receiver and not just an ADC.

2.2.2 Theoretical performance

The results from compressed sensing theory give an idea of what performance to expect from the RMPI. This analysis assumes the signal is perfectly sparse in the Fourier dictionary, which is not true, but the assumption simplifies the error estimates and gives meaningful qualitative relationships. Let Φ be the measurement matrix of the system, and suppose measurements are of the forming

$$b = \Phi x + \sigma_{\text{RMPI}} z$$

where $z \sim \mathcal{N}(0,1)$. Assume $x \in \mathbb{R}^n$ has exactly $k/2$ on-grid pure tones, each of rms amplitude $V_{\text{rms}} = A/\sqrt{2}$, and we observe for time T. The frequencies and phases are unknown, so there are K degrees-of-freedom. In this section, $\hat{x}$ refers to an estimate, not the Fourier transform.

Then if k is small enough, the ℓ_1 recovery gives an estimator $\hat{x}_{\text{RMPI}}$ such that

$$\text{MSE}(\hat{x}_{\text{RMPI}}) \leq c\frac{k}{nr_{\text{ch}}}\sigma^2_{\text{RMPI}} \tag{2.2.1}$$

for some constant $c \simeq 1$, where we define $\text{MSE}(\hat{x}) = \mathbb{E}\|\hat{x} - x\|_2^2/n$. Later sections of the chapter will define MSE in a different normalized sense, but we stick with the above definition in this subsection.

This bound (2.2.1) can be found in [CT07a] for the case of using the Dantzig selector for recovery,

or using [CP07a] (which gives a bound on $\|\Phi(x-\hat{x}_{\text{RMPI}})\|_2$) in conjunction with assuming the RIP on Φ. We show below a heuristic argument that derives this bound, based on the canonical compressed sensing results.

Let $A_i \in \mathbb{R}^{m_i \times n}$ be a Bernoulli ± 1 matrix scaled to have unit-normed columns, and let $b_i = A_i x + \sigma z_i$ with $z_i \sim \mathcal{N}(0,1)$ and x is k-sparse. Then

Theorem 2.2.3 ([Can08] plus [BDDW08]). *If $m_i \geq \bar{c}k\log(n/k)$, then with high probability*

$$MSE(\hat{x}_i) \equiv \|x - \hat{x}_i\|_2^2/n \leq c\|z_i\|_2^2/n = c\frac{m_i}{n}\sigma^2.$$

Suppose we really have a matrix $\Phi = [A_1; A_2; \cdots; A_L] \in \mathbb{R}^{m \times n}$ with dimensions $m = Lm_1$ where $m_1 = m_2 = \ldots = m_L = \bar{c}k\log(n/k)$, and $b = [b_1; \cdots; b_L]$ where $b = \Phi x + \sigma z$. If we form an estimator $\hat{x}$ by averaging the $\hat{x}_i$, then we improve the MSE by $1/L \leq 1$ since we are averaging the errors z_i:

$$\text{MSE}(\hat{x}) \leq c\frac{m_1}{nL}\sigma^2.$$

Now, we wish to allow arbitrary scalings of Φ. For RMPI, the average of the squared column norms in Φ_{RMPI} is r_{ch} (this is just $\|\Phi_{\text{RMPI}}\|_F^2/n = \text{trace}(\Phi_{\text{RMPI}}^T\Phi_{\text{RMPI}})/n$).

The average of the squared columns norms of the Φ constructed above is L. Thus by change-of-variables, using a scaling of $\sqrt{L/r_{\text{ch}}}$ for the noise z, we recover (2.2.1):

$$MSE(\hat{x}_{\text{RMPI}}) \leq c\frac{m_1}{nr_{\text{ch}}}\sigma^2 = \tilde{c}\frac{k\log(n/k)}{nr_{\text{ch}}}\sigma^2 \simeq \tilde{c}\frac{k}{nr_{\text{ch}}}\sigma^2. \tag{2.2.2}$$

Note that Φ *gains* the signal in power by r_{ch}, which improves the MSE.

According to numerical simulations (see Figure 2.5), the constant $\tilde{c}$ for the 8 channel RMPI is about .39.

$$\text{MSE}(\hat{x}) \simeq .389\frac{k}{n}\sigma_{\text{RMPI}}^2 = \frac{3.11}{r_{\text{ch}}n}k\sigma_{\text{RMPI}}^2 \tag{2.2.3}$$

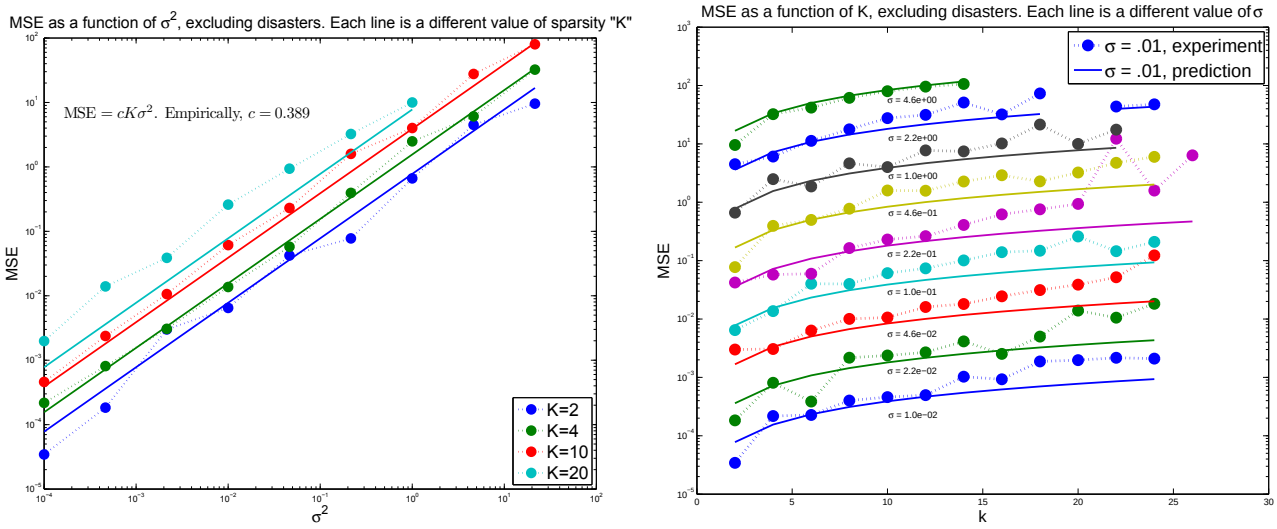
2.2.2.1 Input noise and channelized receivers

In a well-designed system, error should be dominated by input noise:

$$b = \Phi(x + \sigma_{\text{input}}z), \quad z \sim \mathcal{N}(0,1).$$

This is the same as modeling $y = \Phi x + \tilde{z}$ for a modified noise vector $\tilde{z}$ where $\tilde{z} \sim \mathcal{N}(0, \sigma_{\text{input}}^2\Phi\Phi^T)$. Then

$$\sigma_{\text{RMPI}}^2 \equiv \mathbb{E}\|\tilde{z}\|_2^2/m = \sigma_{\text{input}}^2\frac{n}{m}r_{\text{ch}}.$$



(a) MSE as function of noise variance

(b) MSE as function of sparsity k

Figure 2.5: Simulations to find an estimate of the constant in (2.2.1). Shown are averages over 10 trials of non-catastrophic recoveries, using k pure tones (on-grid, but random phase), $n = 1000$, and $m = 100$ measurements. Matrix Φ modeled with ± 1 matrix. Used re-weighted ℓ_1 using [GB10].

Let $\rho \equiv \frac{m}{n} = \frac{1}{12.5}$ be the under-sampling ratio. Thus the error of the RMPI in terms of *input* noise is

$$\mathrm{MSE}(\hat{x}_{\mathrm{RMPI}}) = 3.11 \frac{k}{\rho n} \sigma^2_{\mathrm{input}}. \tag{2.2.4}$$

Is this error good? We compare it to three other systems: a Nyquist-rate system with infinite precision ADCs and no post-processing; the same Nyquist-rate system but in a so-called "channelized" architecture; and a smart "oracle" system that does post-processing.

Nyquist-rate ADC. Neglecting quantization error, if we could make a high-bit ADC at this rate f_s (which is *not* possible), then with no post-processing, the error is

$$\mathrm{MSE}(\hat{x}_{\mathrm{ADC}}) = \sigma^2_{\mathrm{input}} = \frac{\rho n}{3.11 k} \mathrm{MSE}(\hat{x}_{\mathrm{RMPI}})$$

which is terrible compared to the RMPI, since this grows with n

Channelized receiver. Channelized receivers are the current state-of-the-art for high bandwidth systems; see Figure 2.6 for a frequency domain depiction. The input signal x is split into many channels, and each channel processes a small portion of the spectrum after downconverting to baseband via the heterodyne principle discussed in the introduction. For concreteness, let a channelized receiver split the 2.5 GHz of bandwidth into 50 chunks of 50 MHz each. There are two advantages to this: first, each channel processes only 50 MHz of bandwidth, so it can use a 100 MHz ADC which is lower power and more accurate than a higher rate ADC; second, each channel sees only 50 MHz

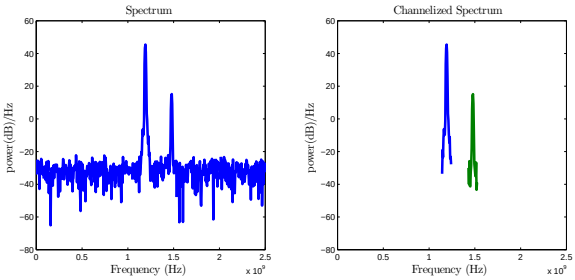


Figure 2.6: Left: two pulses with noise in 2.5 GHz of bandwidth. Right: channelizing the input into channels of width 50 MHz. Only channels with power above a threshold are recovered. The out-of-band noise around the pulses is removed, which helps the SNR.

of noise. Assuming the noise is white, the noise power is directly proportional to the bandwidth B:

$$\sigma^2_{\text{input}} = B\sigma_0^2.$$

If there is only one input tone, then only the channel that contains this tone is used to collect data, so the noise power is proportional to the 50 MHz bandwidth. In contrast, the RMPI is inherently wideband and sees the entire $B = f_s/2$ band, so it has 50 times more noise-power, and hence 17 dB ($=10\log_{10} 50$) less SNR. The MSE for a 50 channel ADC, where σ_{input} is defined over the entire bandwidth $f_s/2$, is

$$\text{MSE}(\hat{x}_{\text{ADC}}) = \sigma^2_{\text{input}}/50 = \frac{\rho n}{50 \cdot 3.11k}\text{MSE}(\hat{x}_{\text{RMPI}})$$

which still grows with n compared to the RMPI MSE.

In practice, the bands should overlap slightly, which requires extra channels. Since each channel requires an ADC, and there are 50 or more channels, this design is extremely power hungry, and uses 50 to 200 watts, depending on the specifics of the ADCs. For comparison, the RMPI uses on the order of 1 to 4 watts.

Oracle. Consider the (un-channelized) infinite precision Nyquist rate samples again. If we knew in advance which tones were non-zero (such information is called *oracle* information), then with post-processing denoising techniques, the error is

$$\text{MSE}(\hat{x}_{\text{ideal}}) = \frac{k}{n}\sigma^2_{\text{input}}.$$

This does not change if the input is from a channelized receiver, since the denoising process reduces out-of-band noise already.

Comparison. Comparing the RMPI to the non-post-processing techniques is tricky. The RMPI error is a function of the information content in the signal. For a fixed information content, the

RMPI error improves as more samples are collected. With the Nyquist-rate receiver, there is no estimation occurring, and more samples do not improve the estimate.

The RMPI penalty compared to an *oracle* pays only a penalty of $3.11\rho^{-1} \simeq 15.9$ dB. This is quite reasonable, since the oracle result requires knowing the frequency of the signals and having an ADC that samples at f_s with very high ENOB. This comparison is only valid when k is sufficiently small.

2.2.3 Related literature

Since 2006, there have been several analog-to-information (AIC or A2I) systems discussed in the literature. A few such systems are reviewed below. Many of these papers contain major ideas but offer few details. All prior work has assumed exact spectral sparsity, and used the Fourier basis for reconstructions; this assumption is relaxed in this thesis. Systems that have been implemented in hardware will be specifically mentioned. Our RMPI design has been influenced by conversations with many people, but among published papers, [TLD+10] was most influential.

2.2.3.1 Other RMPI systems

The RMPI was proposed in 2005 [SOS+05], and has been discussed in the literature by a team at Rice University [KLW+06] in 2006, and further discussed by the same group in [LKD+07, KRL+06]. The first main theoretical work was the "Beyond Nyquist" paper [TLD+10] discussed below. Another DARPA group led by Hoyos has also proposed multi-channel RMPI designs [YHS08, CYH+11].

The initial work [KLW+06] proposed the basic RMPI design, including the concepts of a PRBS (and declared the rate must be at f_s or higher), mixing, integration, and sampling. The signal model assumes that signals are on-grid sinusoids or sparse in some ortho-basis. Notably, they propose a high-level calibration scheme. The design was further expanded in [LKD+07], which assumes the same on-grid signal model, although it also discusses the Gabor dictionary. The authors implement a transistor level SPICE simulation of the basic system, though without any front-end amplifier, and only at schematic level since there was no layout. The implementation includes non-idealities at a few stages, although their estimates of the limiting non-idealities differ than what we consider the most important; according to their design, the non-linearity of the mixer is the most significant non-ideality, since they do not model the LNA. The integrator is allowed to be a single pole, though the pole is set to 100 Hz which may be quite optimistic. It is not clear if thermal noise is included. Using the end-to-end simulation, a superposition of on-grid tones is reconstructed and the spectrum of the reconstruction looks reasonably accurate.

The work [KRL+06] was a precursor to [LKD+07] and had simulation results from SPICE, and built simpler non-SPICE behavioral models to mimic the effects of non-idealities. They tested the effect of a nonzero pole in the integrator, and found that performance degraded as the pole moved

farther from DC, although this may have been using a non-calibrated model, in which case this is due to model mismatch and not any intrinsic effect of the pole. Their dominant non-ideality was the non-linearity in the mixer, which was more significant than clock jitter in the mixer. Our RMPI has been designed carefully to keep the mixer very linear, and we find that the LNA non-linearity is just as significant. The work of [KLW+06,KRL+06,LKD+07] is due to a Rice University team. Rice University has a contract with the same DARPA BAA as the Caltech team, and they are actively working to implement a type of compressed sensing system in hardware, though their device has very different bandwidth and recovery goals than the RMPI [RLN+08].

The first published work on a multi-channel design appears in a report by a team at Texas A & M University [YHS08]; this application is tailored for spectrum sensing in cognitive radio. These authors also have a contract with the DARPA BAA, so it is not surprising that their design, the Rice University design, and the Caltech design are all related. Their methodology is called the "Parallel Segmented Compressed Sensing" (PSCS) structure, and at a high level, it is identical to the RMPI. They cite worries over the "10th order" integrating filter in [LKD+07] (though this may be due to a misinterpretation), and suggest that the single-channel design's samples are redundant, which can be fixed with the multi-channel design. Regarding the number of channels, they suggest that an upper limit is necessary, due to practical implementation issues. Notably, they introduce a stagger to the channels, suspecting that samples would have higher fidelity in the interior of a sampling period and thus the edges of the channels should not align. Using orthogonal matching pursuit (OMP), they demonstrate basic feasibility with Matlab simulations using on-grid tones and a signed Bernoulli matrix.

The same team recently published [CYH+11], detailing a 90 nm CMOS chip they plan to fabricate. The layout was fully simulated in Cadence Spectre. The chip consumes 120 mW per channel, which includes specs for ADCs, although they assume that a LNA is off-chip and that clock distribution, which typically accounts for 30% or so of power, is also off-chip. Their signal model assumes on-grid tones, of which 4% are occupied. They predict a 44 dB SNR. The effective bandwidth is 1.5 GHz, using 8 channels, each sampling at 110 MHz. This is very similar to the Caltech design, though less ambitious in terms of the undersampling ratio (they undersample by $\times.587$, our design undersamples by $\times.08$). Similar to what we will find in this chapter, they conclude that mixing jitter is a limiting non-ideality, and assume their jitter will have 0.5 ps rms (same as our design). Non-DC poles in the integrator are not discussed, but they mention using a dual-integrator design which samples on one branch while resetting the other branch, then swapping, so that railing/clipping of the integrator is not an issue. They briefly discuss calibration. Like in the earlier paper, signal recovery is done straightforwardly via OMP and a Fourier dictionary, since off-grid tones are not considered. A thorough analysis of noise at different blocks is presented. Due to the similarities with the Caltech RMPI, it will be interesting to see the results once their chip is fabricated, since

both teams face similar problems.

We note recent work by [TV11] that considers parallel branches of mixers and integrators, which is similar to already proposed designs except for an extra step at the sampling which introduces extra samples; the extra samples are correlated to earlier samples, but they claim it gives processing gains. Another new A2I device has been mentioned in the dissertation of Michael Trakimas at Tufts, but we are unaware of any papers on the subject and so do not know the details. The recent work of [MPAL11] also proposes a RMPI design, and they have constructed a prototype in 500 nm CMOS with 200 MHz of bandwidth, and appear to be planning construction of a 130 nm device. They demonstrate on-grid recovery with Matlab simulated data.

Beyond Nyquist. The paper [TLD+10] called "Beyond Nyquist" is probably the most influential A2I paper, so we give it an extended discussion. Though authors of the paper are associated with both the Rice and Caltech hardware projects, the paper itself is mainly theoretical. Their setup is identical to the model considered in this chapter, except they only consider a single channel, and with different assumptions on non-idealities and practical considerations. Using a particular signal model, they prove the first rigorous theorems, which state, in effect, that the RMPI mixing operations are not much worse than using a standard compressed sensing matrix like a random signed Bernoulli matrix. This signal model is not overly realistic, but it is important because the result suggests that the operations of the system adhere to the compressed sensing principle, and follow a phase-transition similar to the Donoho-Tanner phase transition for Gaussian matrices.

Specifically, assume x is a periodic signal with K on-grid tones. For example, amplitude-modulation of on-grid signals falls into this class, while frequency-modulation does not. Their model uses a single channel which undersamples the Nyquist rate by the amount M/N. Empirically, they suggest that recovery happens when

$$M = 1.7K \log(N/K + 1).$$

This is complemented by two theoretical results, which we state here. Fix a time T and let $N = T/\Delta T$. Write X for the unitary DFT of the discrete signal x.

Theorem 2.2.4 (Noiseless recovery of a random on-grid random signals, [TLD+10])**.** *Suppose x has K on-grid frequencies chosen uniformly at random, so X has up to $2K$ non-zeros, including the negative frequencies. Let the non-zero entries of X have arbitrary amplitudes, and random complex phases. Pick a PRBS sequence at random. Then a single-channel ideal RMPI with M measurements (and assume M divides N)*

$$M \geq C \left(K \log(N) + \log^3(N) \right)$$

will exactly recover the signal from the noiseless measurements

$$b = \Phi x, \quad \Phi \in \mathbb{R}^{M \times N}$$

by solving basis pursuit (1.1.5)*, except with probability* $\mathcal{O}(N^{-1})$*. The constant* C *is independent of* N *and* K*.*

This theorem might be interpreted as saying that most PRBS sequences will work to reconstruct most inputs, as long as M is large enough. However, since the PRBS sequence is typically fixed and not changed, we might wonder if there is a big risk of choosing a "bad" PRBS. It turns out as long as the PRBS is initially selected at random, then most likely, recovery will still happen. Furthermore, this is robust to additive noise, and adapts to compressible signals. To be specific:

Theorem 2.2.5 (Noisy recovery of arbitrary signals, [TLD+10])**.** *Fix any PRBS sequence. Let*

$$M \geq CK \log^6(N)$$

and assume M *divides* N*. Take measurements*

$$b = \Phi x + \eta$$

and estimate $\hat{x}$ *be the solution to the basis pursuit denoising problem* (1.1.6) *with parameter* $\varepsilon = \|\eta\|_2$*. Then except with probability* $\mathcal{O}(N^{-1})$*,*

$$\|\hat{x} - x\|_2 \leq C \max\left\{\varepsilon, \frac{1}{\sqrt{K}}\|X - X_K\|_1\right\}$$

where X_K *is any* K*-sparse vector in* $\mathbb{C}^N$*, i.e., the best approximation of* X*.*

Another improvement of this theorem is that it doesn't require x to be exactly K sparse in frequency; good reconstruction occurs as long as its DFT is compressible. However, a major issue still happens for off-grid tones. These tones are not periodic, so instead of the DFT, it is appropriate to analyze their continuous Fourier transform. Because only a finite time sample T is observed, this has the effect of a boxcar filter in the time domain, which means the Fourier transform of the time-limited function will show a sinc decay away from the active tone. Since the sinc function decays like $1/f$ in frequency, it is not well-approximated by a sparse DFT.

This is a common problem in signal processing, and the usual solution is windowing, as will be discussed in detail in §2.7.2.6. The "Beyond Nyquist" paper proposes implementing a window in hardware via a time-varying amplifier; furthermore, because two shifted windows are needed, the number of channels must double. Whether such a scheme is practical remains to be seen.

Though calibration is not discussed in detail, the paper suggests that non-ideal integrators should be acceptable as long as they can be characterized well, which agrees with the findings presented in this chapter. Some Matlab simulations are presented using IRLS.

2.2.3.2 Related systems

NUS. The non-uniform sampler (NUS) was described in the introduction. It is ideal for sensing signals composed of only a few frequencies or frequency bands, such as in typical narrowband modulation schemes. In [ME10], a multi-co-set sampling strategy is discussed, which is effectively the same as NUS except with periodicity of the samples made more explicit, and uses a bank of ADCs (each sampling at the same rate, but offset from each other in time). It is suggested that a drawback of the system is that it needs specialized implementation, since even though the ADCs only need to sample at a low-rate, they still sample a high-bandwidth signal. Since ADCs are designed to sample signals that have already been low-pass filtered to the ADC sampling rate, this would introduce errors due to parasitic response in the ADC. The Northrop-Grumman NUS overcomes this by a specialized implementation which includes their own sample-and-hold block.

Random convolution. The random convolution has been proposed in 2006 by [TWD$^+$06] and analyzed in detail in [Rom09]. In spirit, it is quite similar to the RMPI: the idea is that a signal is convoluted with a random complex point-spread function that changes the phase of the spectrum while keeping magnitudes the same. Following this, two schemes are analyzed: either sampling the output directly, or sampling a Rademacher sum (similar to the RMPI). The latter technique is shown to be a universally good system, meaning that the input can be sparse in any fixed ortho-basis.

The random convolution and the random sums make the system more complicated than the RMPI, and it has not yet been implemented. Applications are suggested in radar sensing (as opposed to passive radar observation like the RMPI), and Fourier optics. In the latter setting, it is assumed the receiver can detect the phase of the light, which may make it less enticing in practice.

Multiband sampling theory. The "Xampling" methodology [ME10, MEE09, Eld09, MEDS11], which follows in the steps of the authors' blind multiband sampling theorem [ME09a], is similar to the RMPI in many respects. Unlike most other proposals, a prototype has been developed in the lab [MEDS11]. The signal model is that x has only N bands of frequency, and each band has size at most B, so the overall bandwidth is NB, and it is assumed $NB \ll f_s$. The approach is implemented in a "modulated wide-band converter" (MWC) which splits a signal into channels, mixes each channel with a unique periodic sequence, then low-pass filters the output before sampling at a low rate. The periodicity of the mixing sequence, the rate of the sampling, and the number of channels are discussed in detail in [ME10]. The design works for both known and unknown

(a.k.a. "blind") bands.

If the band locations are unknown, then after collecting a reasonable number of samples (for example, 1 to 10 μs worth), an ℓ_1 block sparsity problem is solved to find locations of non-zero blocks of spectrum; this is referred to as the "continuous-to-finite" (CTF) block. The computational problem may be reasonably sized, though perhaps not possible for real-time implementation. This processing is cheapest when the bands are large; if there are many bands that are very small and evenly spread out, the computational problem may be expensive. In the follow-up paper [MEE09], the authors describe a digital processing step that recovers the base-band versions of the occupied frequency bands.

The methodology behind Xampling is motivated by working directly with analog signals, and doing all processing at base-band, rather than Nyquist rate. The work seems promising, though it may be best adapted for spectrum sensing, such as in cognitive radio, rather than full recovery; or suited for recovering signals with known types of modulation (such as QPSK). Another key feature of the model is that the system is relatively simple to implement and does not rely on delicate hardware; to this extent, they seem to have achieved the goal admirably.

The hardware of both the MWC and the RMPI are not too dissimilar, the main difference being the chipping sequence rate, which is much higher for the RMPI case. The essential difference of the system is the processing. The RMPI makes full digital processing, while the MWC uses digital processing only to identify the support, and then works with base-band digital processing. Our view is that this methodology is exciting and worth further investigation. However, below we detail a few minor criticisms of the MWC and address some comments from [MEE09] about the RMPI.

We first note the difference in signal models, which is simply a matter of application. Both the MWC and RMPI do poorly when analyzed using the wrong signal model. The MWC assumes a static spectral occupancy. Whenever the spectrum changes significantly, the CTF block must recompute the spectral support. Furthermore, this spectral estimation requires many time samples, and would give inaccurate estimates for time-dependent inputs like a radar pulse. Similarly, the RMPI adds unnecessary complexity and processing if applied to the problem of finding signals with infinite support. It doesn't exploit continuity in the spectrum from one reconstruction window to the next, and it does not automatically offer full analog resolution since it works with finite discrete samples.

It is mentioned that RMPI systems have a high computational load, and this is true. Real-time processing of the RMPI is not yet achievable, though as this thesis shows, the processing time is reasonable. The standard RMPI reconstruction works at the Nyquist rate, although the compressive matched filter doesn't rely on this, so is closer to the baseband processing suggested in Xampling. Other valid criticisms are the lack of flexibility for changing the bandwidth f_s on-the-fly, and sensitivity to windowing issues. In particular, the on-grid tone model is viewed as too academic.

For the radar signals studied in this thesis, windowing and on-grid frequencies are not a major issue since the radar envelope is itself a window with finite bandwidth. More about windowing is discussed in §2.7.2.6. It is true that if our RMPI were restricted to measuring on-grid pure tones, it would perform much better, but we find adequate performance without this assumption.

It is suggested that in the RMPI, any deviation of the integrator filter from being a true integrator will introduce signal-dependent errors, and the system really must be calibrated *continuously*! This is quite misleading. The RMPI system certainly needs to be calibrated, but this can be done once. In particular, the post-processing does not rely in any way on the ideal integrator model. The calibration will not be exact, and the error due to this will be signal dependent, but this is not necessarily dangerous, and is inherent in many hardware devices, including the MWC. For example, the system may be mis-calibrated because of an absolute timing shift, but this is harmless because the reconstruction will just be a time-shifted version of the input.

Cognitive radio. Cognitive radio systems are designed to sense a broad range of RF spectrum to detect unoccupied channels, and as such, this technique is not tied to a particular hardware implementation. For example, both the MWC and the RMPI can be used for this. Many papers on this area have appeared recently; see [ME10] for a discussion.

Nyquist Folding Receiver (NYFR). Proposed in [FBC+08] and also part of the same DARPA project as the RMPI, this design samples at a time-varying rate. For example, the sampling rate averages $f_{avg} = 2$ GHz, but varies sinuisoidally in time between 1950 and 2050 MHz. For input tones of high frequency, say $f_{in} = 10$ GHz, this induces aliasing. The aliased signal will change frequency over time since the sampling rate is changing over time. The amount this aliased signal changes over time is determined by what "Nyquist zone" the carrier signal was in. The Nyquist zone L is the smallest integer such that $f_{avg}L > f_{in}$. Thus even from aliased signals, it is possible to determine what the original absolute frequency was, since the time dependence introduces a splitting.

This design is examined in [FBC+08] for a few narrowband tones, and it is shown experimentally that their original frequencies can be successfully recovered; this is compared to a standard time-independent sampler, which cannot recover the original frequency. However, the implications are not clear, since many applications are not actually concerned with recovery of absolute frequency information and only need the narrowband content. It has not been shown that this design is robust to recovering arbitrary sparse tones, e.g., tones that are close in frequency, or tones with significant bandwidth that overlap two Nyquist zones. The method is claimed not to rely on digital Nyquist-rate ℓ_1 recovery, though the actually recovery algorithm is not discussed.

2.3 Modeling the system

This section describes how to simulate the system numerically. The process of describing the modeling should also make it clear what the RMPI does. The results in this chapter present many different types of simulations and models. Some models were improved over time, so preliminary simulations were necessarily not as accurate. The best models, that of the Simulink and SPICE simulations, are orders of magnitude slower than the simplest models, so in some cases we had to choose between a single accurate simulation and many less accurate ones. We discuss the modeling inaccuracies and when we think they may affect results.

There are two purposes behind modeling the system. The first is to generate realistic measurements for testing purposes. The second purpose is to create a model matrix Φ that the reconstruction solvers will use. In a single computational experiment, we may use two different models: for example, a highly accurate model incorporating non-idealities is used for the measurements, and a simpler model is used to generate Φ.

This section concludes with a discussion of calibrating the hardware device. This is extremely important, so some time is spent discussing the issues and various strategies.

First, we review the mathematical representation of the system. Unless otherwise specified, we will work with only a single channel, since including the representation from other channels is trivial (except perhaps for alignment issues during hardware calibration). Let $x(t)$ be in the input to the integrator, and $y(t)$ be the output. Then if $h(t)$ is the system's *impulse response* (or "Green's function" in PDE terminology), we have $y(t) = h(t) * x(t)$ where "$*$" is the convolution operator:

$$y(t) = \int_{-\infty}^{\infty} h(\tau)x(t-\tau)d\tau.$$

Our interest is in causal systems, which means $h(t) = 0$ for $t < 0$, so the convolution integral can be taken from $[0, \infty]$. Let $\mathcal{L}$ be the Laplace transform operator (in the subsequent chapters, this symbol will be used instead for the Lagrangian), defined by $\mathcal{L}(x)(s) = \int_{-\infty}^{\infty} x(t)e^{-st}dt$. The transfer function $H(s)$ of a system is defined as $H(s) = Y(s)/X(s)$, or equivalently as $H = \mathcal{L}(h)$. If the output is an integral, $y(t) = \int_{-\infty}^{t} x(\tau)d\tau$, and if $x(t)$ has finite support, then using integration-by-parts gives

$$h(t) = u(t) = \begin{cases} 1 & t \geq 0 \\ 0 & \text{else} \end{cases}, \qquad H(s) = 1/s.$$

Another important system is the single-pole system:

$$h(t) = \begin{cases} e^{-at} & t \geq 0 \\ 0 & \text{else} \end{cases}, \qquad H(s) = \mathcal{L}(h) = \frac{1}{s+a}.$$

The frequency response of the system to a frequency f is $H(if)$, since this is just the Fourier transform of h. A fundamental fact of linear time-invariant (LTI) systems is that they can be represented exactly by an impulse response. The RMPI system is linear but not time-invariant since it is periodically sampled. Restricted to within a sampling period, the system is LTI. When using the $H(s)$ terminology, we are implicitly restricting our attention to a single sampling period and assuming that the input signal is 0 before the period of interest.

Let $c(t)$ be the continuous time version of the chipping sequence (a.k.a. pseudo-random bit sequence PRBS). For now, H and h may be arbitrary, but are still called the "integrating" filter since the system should be similar to an integrator. Then the 1^{st} output is

$$y_1 = \int_0^{T_{\text{int}}} c(t)x(t)h(T_{\text{int}} - t)dt = \sum_{n=0}^{N_{\text{int}}-1} c_n \int_{n\Delta T}^{(n+1)\Delta T} x(t)h(T_{\text{int}} - t)dt \tag{2.3.1}$$

and the m^{th} output is defined similarly, with integration limits $[(m-1)T_{\text{int}}, mT_{\text{int}}]$ and the discrete chip sequence shifted appropriately. In particular, h will be the same regardless of m; for this reason, the RMPI system is not an LTI system on large timescales, but for the period $[(m-1)T_{\text{int}}, mT_{\text{int}}]$ it is.

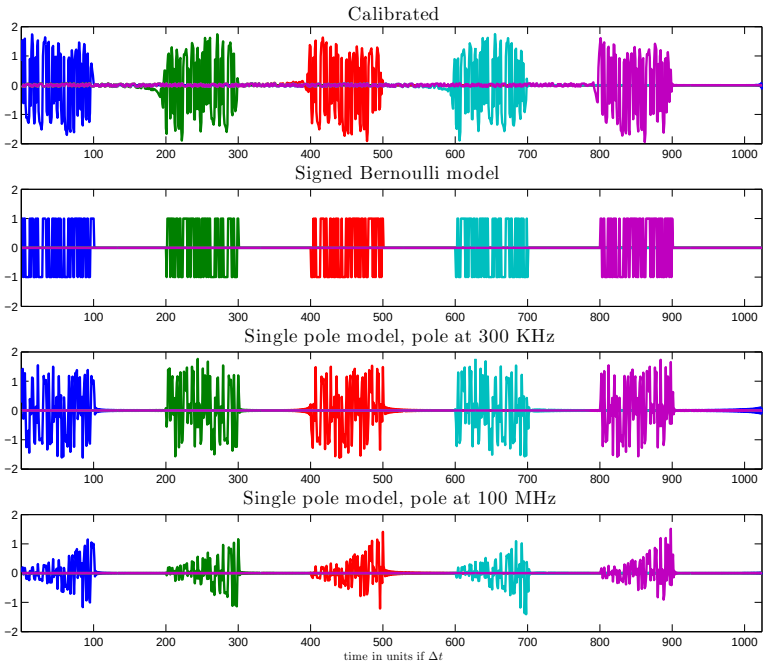


Figure 2.7: Showing samples for one channel of various models (showing every other sample for visual clarity), all with the same chip sequence. The Fourier-based models are not guaranteed to be causal. The entries of the single-pole model are not the same as windowing the Bernoulli model by e^{-at}.

2.3.1 Simple models

The simplest model of the system is as a block-diagonal matrix with ± 1 entries on the nonzero portions. Recall that the discrete vector x represents $x(t_n)$ for $t_n = n\Delta T$, and $n = 0, \ldots, N-1$; we may occasionally change between 0-based and 1-based representation when one representation is cleaner. Thus multiplication with a row c of this matrix is

$$b = \sum_{i=0}^{N_{\text{int}}-1} c_n x_n$$

which can be thought of as a quadrature approximation of the integral $\int_0^{T_{\text{ADC}}} c(t)x(t)dt$. For high frequencies, this is quite inaccurate. If instead we decide that x_n represents $\int_{n\Delta T}^{(n+1)\Delta T} x(t)$, then now this is an exact representation of the integral. However, this discrete representation is problematic, since it is not clear how to extract frequency information, and this representation does not represent the system if $h \neq 1$.

With the discrete representation of x, it is very easy to consider the discrete Fourier transform coefficients $\hat{x} = Fx$ since this is fast to compute and involves no loss of information. Using the conventions of Table 2.2 on page 45,

$$x_n = \frac{1}{N} \sum_{k=0}^{N-1} \hat{x}_k e^{i2\pi kn/N}. \tag{2.3.2}$$

Note that we can equivalently write

$$x_n = \frac{1}{N} \hat{x}_{N/2} \cos(2\pi(N/2)(n/N)) + \frac{1}{N} \sum_{k=-(N/2-1)}^{N/2-1} \hat{x}_k e^{i2\pi kn/N} \tag{2.3.3}$$

using the convention that $\hat{x}_{-k} = \hat{x}_{N-k}$. For convenience, we have chosen N to be even. Since x is real, $\hat{x}$ is conjugate-symmetric: $\hat{x}_k = \bar{\hat{x}}_{-k}$.

To approximate the integral in (2.3.1), the plan is to approximate $x(t)$ using $\hat{x}$. Recall that our pragmatic definition of the signal class was any signal that can be well approximated by its DFT, so now we put this assumption to use. If $x(t)$ is evaluated at an arbitrary point in time, rather than a grid point, the two inversion formulas (2.3.2) and (2.3.3) are not equivalent. Both of them interpolate x at the nodes t_n, but are different in between. One benefit of using the second formula (2.3.3) is that by the conjugate symmetry of $\hat{x}$, this guarantees that $x(t) \in \mathbb{R}$; using (2.3.2), there is no reason to expect $x(t)$ to be real-valued. The approximation is

$$x(t) = \frac{1}{N} \hat{x}_{N/2} \cos(\pi t N/T) + \frac{1}{N} \sum_{k=-(N/2-1)}^{N/2-1} \hat{x}_k e^{i2\pi kt/T} \tag{2.3.4}$$

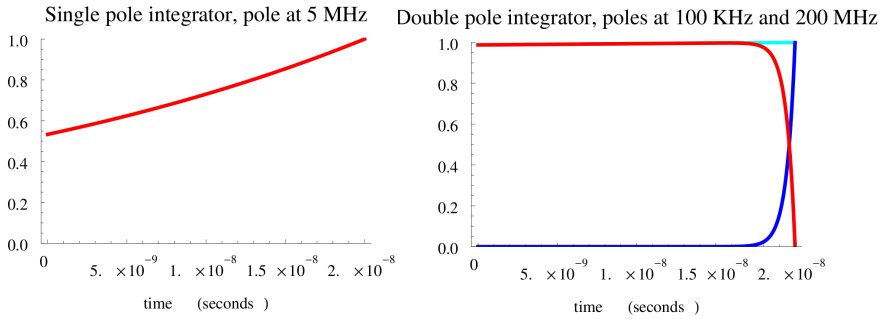


Figure 2.8: Time domain representation of integration. $h(T_{\text{ADC}} - t)$ for the integrator. Left: single-pole system. Right: double-pole system, showing the effects of each pole, and their combined effect (light blue)

for $T = N\Delta T$.

We can substitute this expression in (2.3.1) and then exchange the order of summation and integration. Thus

$$y_1 = \sum_{k=0}^{N-1} \hat{x}_k q_k \tag{2.3.5}$$

for some q which depends only on the discrete sequence c and the impulse response h. This is the form of an inner product, so we can build a representation of the system by stacking and shifting the q vectors for different time periods. This builds a matrix Φ_{FFT} which operates on $\hat{x}$, and hence a time-domain matrix representation of the system is $\Phi = \Phi_{\text{FFT}} F$ where F is the DFT matrix. The computation is easy to perform since $\Phi_{\text{FFT}} F = (F^* \Phi^*_{\text{FFT}})^*$, and applying F^* is equivalent to applying the inverse FFT (and scaling by N).

The computation of q, which needs only be done once, requires evaluating definite integrals of the form $\int e^{i2\pi kt/T} h(T_{\text{int}} - t)$. If h represents an ideal integrator, then $h = 1$ and the integral can be done in closed form; this is the system analyzed in [TLD+10]. If the poles and zeros of H are known, then h can be written concisely in terms of exponentials, and this can also be integrated easily.

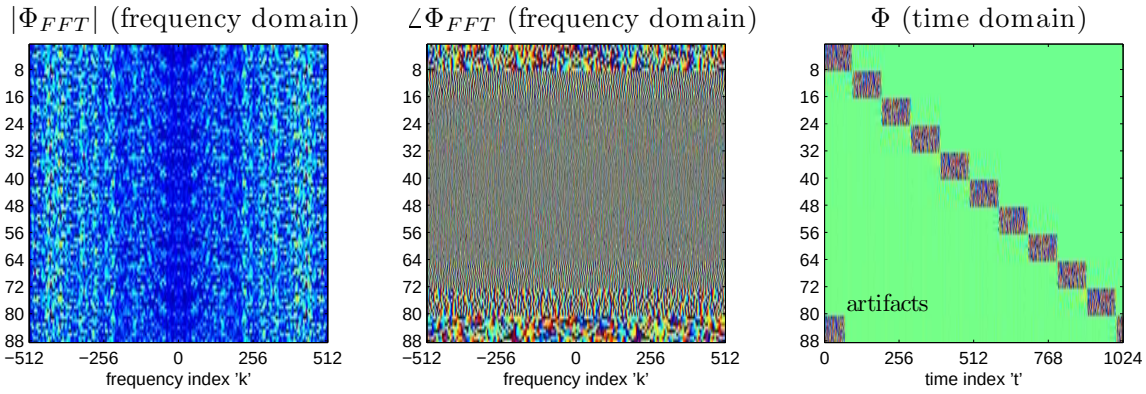


Figure 2.9: The left and center matrices show the magnitude and phase of the complex entries of Φ_{FFT}, which is designed to act on Fourier coefficients $\hat{x}$. The time-domain version Φ is shown on the right, and is generated by taking the inverse FFT of Φ_{FFT}.

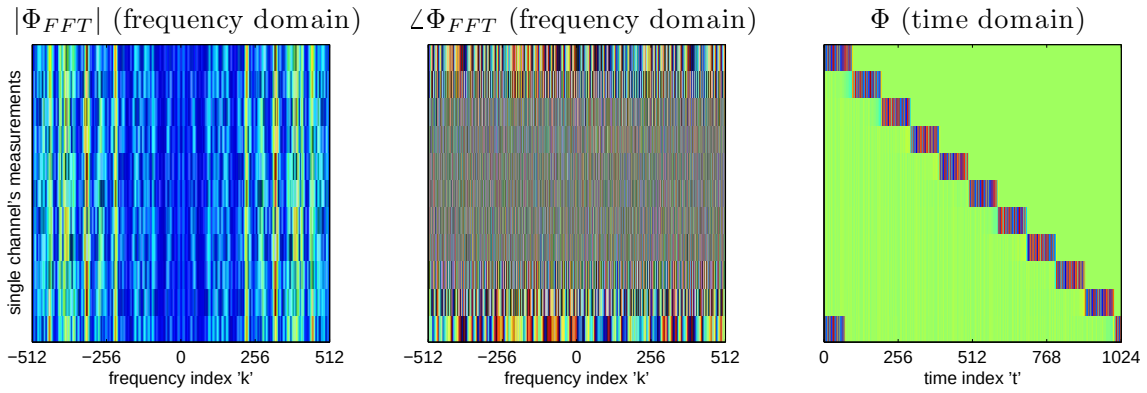


Figure 2.10: Showing the same matrix as Figure 2.9, but now only showing a single channel

The advantage of the previous model is that it easily extends to other versions of h and H. As we discuss in the section on integration §2.6, it is not possible to implement $H(s) = 1/s$, but rather we consider the *single-pole model* $H(s) = \frac{1}{s+a}$ for a pole frequency a. The corresponding impulse response is $h(t) = e^{-at}$ (where a is in Hz); this is sometimes characterized by the *time constant*, which is the time it takes h to decay to e^{-1}, so clearly the time constant is $1/a$. See Figure 2.8

There is a bizarre effect of using the frequency domain representation. In the simple Bernoulli model, a row of Φ consists of N_{int} nonzero entries. For example, the first row has nonzero entries in locations $1, \ldots, N_{\text{int}}$, and zero entries from $N_{\text{int}} + 1, \ldots, N$. This makes sense, because the system is causal, and the measurement y_1 taken at time T_{ADC} should only depend on $x(t), t \leq T_{\text{ADC}}$. However, when $\Phi = \Phi_{\text{FFT}} F$, there is no guarantee that the entries $N_{\text{int}} + 1, \ldots, N$ in the first row are zero. Typically, these entries are small but nonzero. This seems to violate causality. Really, there is no paradox, because the discrete vector x is no longer the time samples of $x(t)$ but rather just the inverse DFT of $\hat{x}$. Or, viewed differently, we know that the continuous signal $x(t)$ can be represented exactly by its DFT only if x is band-limited and periodic with period T, and if x is periodic there is no violation of causality. The matrix should not be "cleaned" by removing the non-causal portions, as this reduces the modeling power. An example of this effect can be seen by closely examining the gaps between samples in the top row of Figure 2.7.

A related phenomenon is shown in Figures 2.9 and 2.10. The last sample is incomplete, since it goes from $n = 1001$ to 1100, but $N = 1024$. Using the frequency approach, it appears as if the sample is still there, but has wrapped back to $n = 1, \ldots, 76$. This sample is inaccurate and we remove these rows from the matrix.

2.3.2 SPICE

The schematics and layout of the chip were done with the Cadence design suite. Cadence includes a number of circuit and transient simulators which allow high-accuracy simulations. In particular,

verilog-ams was used for behavioral models, and spectre-RF was used for AC analysis and transient simulations. These simulators include various forms of SPICE (Simulation Program with Integrated Circuit Emphasis), and rely on the device models from BSIM4 [BSI09]. For simplicity, we refer to these tools collectively under the generic label "SPICE".

Because a single license for Cadence is extremely expensive, these simulations cannot be run on a cluster and instead are run on a single high-performance server. Simulations with these models take on the order of a few minutes up to a few hours, depending on how many blocks are being simulated and which effects are included (e.g., just schematics, or considering the physical layout as well). Simulations can be used to determine parameters such as the frequency response of a specific block, or they can perform and end-to-end simulation of an arbitrary input.

2.3.3 Simulink

Because the SPICE simulation is slow, it is complemented with a behavioral model created in Simulink. The Simulink model also had the advantage of interfacing easily with Matlab, which is the language used for most of the experiments. The Simulink model consists of the following key blocks:

1. A low-noise amplifier (LNA) to gain the signal and remove out-of-band noise. After this point, the signal is split into 8 channels, and so there are 8 copies of each of the next components.
2. A transconductor (Gm) to convert the voltage signal into a current signal.
3. A mixer to multiply the signal with the chip sequence.
4. A common gate (CG) amplifier to provide current gain, which improves the noise figure.
5. An integrator.
6. A sample-and-hold block, and samplers.

Schematics of the entire system are shown in Figure 2.11, which includes various input sources, the LNA, the clock, channel blocks, and sampling. The schematics of a single channel block, which includes the mixing, CG, and integrator, are shown in Figure 2.12. The parameters for each component, such as the poles and zeros of the filters, were extracted from simulations with the SPICE model.

Additionally, the model included non-idealities, which could be turned on or off. White noise was injected into the LNA (this was the input referred noise, and takes into account the noise sources from the whole block), using an estimated value from SPICE. We also tested adding in noise directly to the individual components. The LNA, Gm, and CG were distorted by polynomials, with the non-linearity fitted by extrapolating to IIP2/IIP3 values taken from SPICE. Similarly, the frequency response of the LNA, Gm, CG,

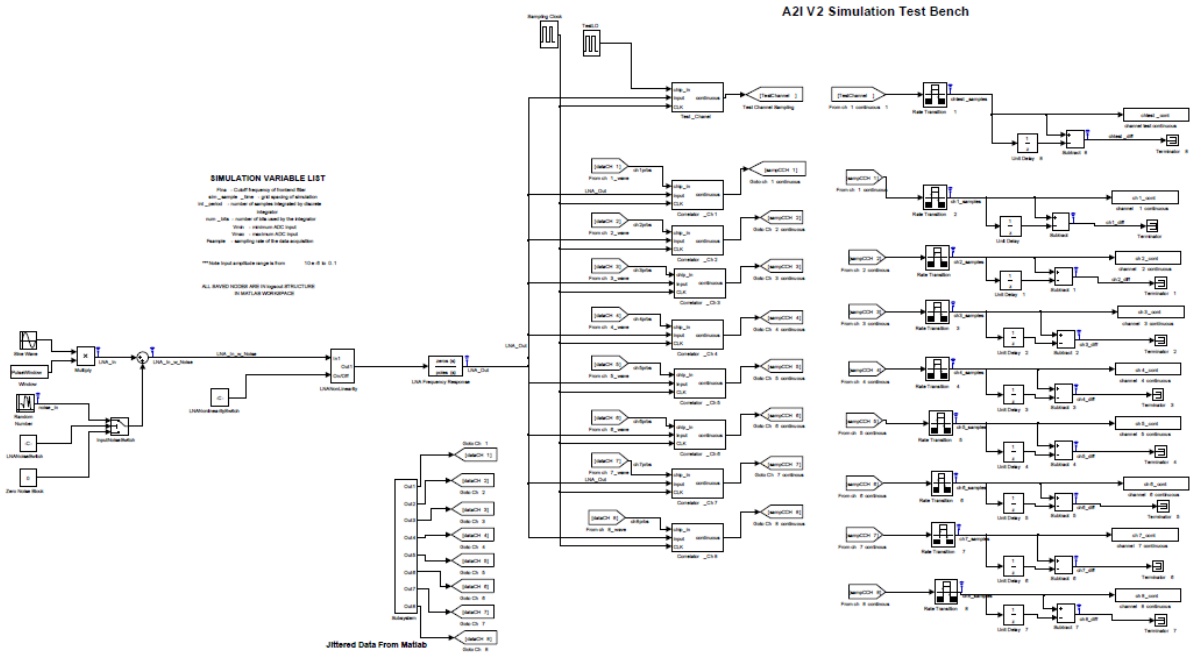


Figure 2.11: Simulink model. This was constantly updated as the chip design was updated.

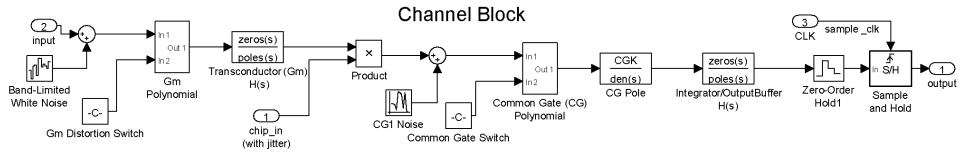


Figure 2.12: Details of a single channel in the Simulink model. The various sources of non-idealities are present, and were able to be switched on and off.

To solve the system, we set Simulink to use Matlab's `ode3` solver, with a fixed time step that we varied from 0.1 to 0.5 ps (recall the Nyquist time ΔT is 200 ps). Depending on the total period T and the size of the fixed time step, simulations took between 10 seconds and a few minutes, if the model was pre-compiled into binaries (compilation takes several minutes).

The Simulink model was used for two purposes:

1. With various non-idealities turned on, it generated test samples, which were fed into the reconstruction algorithms to test performance of the system.

2. With non-idealities turned off, it was used to generate a frequency domain model Φ_{FFT}, which in turn generates Φ. This process is referred to as "calibration" and is described in the next section.

The purpose of the first mode of use was to generate noisy samples. Specifically, this goes beyond the additive white Gaussian noise (AWGN) model, and includes non-linear effects such as the jitter

noise and non-linearities. The Simulink model is only an approximation, but it is useful as long as its noisy measurements test the robustness of the measurement and recovery process.

2.3.4 Calibration

The Simulink model, despite its flaws, is much more accurate than modeling the entire system as mixing followed by a single- or double-pole filter, as considered in in the simple models. To get the Simulink model to provide a matrix representation of the system Φ, we build up an "impulse response" by sending in a series of known inputs. Let S represent the system as a "black-box", which can be probed indirectly by sending in signals and recording the output. Furthermore, S is linear and approximately finite-dimensional, so we seek to represent it with a matrix Φ. If Φ is a $M \times N$ matrix and $S(x) = \Phi x$, then sending in any set of N linearly independent vectors into S and recording the output is sufficient to characterize Φ. This was initially tried with the unit impulse functions e_i, in order to provide the traditional impulse response. This approach does not work well since e_i have sharp rise and fall times, and thus slowly decaying Fourier coefficients, so they are not in the signal class. A much better approach is via the Fourier domain. Using the representation (2.3.4), we see that we need only the response of S to sines and cosines on the frequency grid. Thus we build a frequency-domain impulse response to Φ_{FFT}, and then invert via the IFFT to get Φ. The basis functions are $e^{i2\pi kt/T}$ for $k = -N/2+1, \ldots, N/2$. To measure $S(e^{i2\pi kt/T})$ we expand the complex exponential and use linearity to get $S(e^{i2\pi kt/T}) = S(\cos 2\pi kt/T) + iS(\sin 2\pi kt/T)$. Thus a single basis function requires two measurements, but since the negative frequencies can be inferred from the positive frequencies only N overall measurements are needed.

This procedure defines the calibration of the system. The response of the system is measured for each input from a set of $N/2$ cosines and $N/2-1$ sines (and also a DC signal); see Figure 2.13 for a diagram. A Simulink model represents the RMPI system, since for $N = 1024$, a SPICE simulation would take about a month of computation time. This approach is more accurate than the simpler single pole models, and has the added benefit that it does not require estimation of the location of poles or other parameters. Because this only requires a black-box system, it can use the actual hardware system to generate measurements; this is useful when the noise in the hardware is small and has less effect than the modeling errors in the simulators. Since the inputs can have arbitrary amplitude, they are chosen in the "sweet-spot" amplitudes which is small enough that non-linearities can be ignored yet large enough that thermal noise plays a small role.

To validate the calibration approach, there are two considerations.

1. Given a known input x, do the actual measurements $S(x)$ agree with the predicted measurements Φx?

2. Given a known input x with measurements $b = S(x)$, does the reconstruction $\tilde{x}$ (a function of

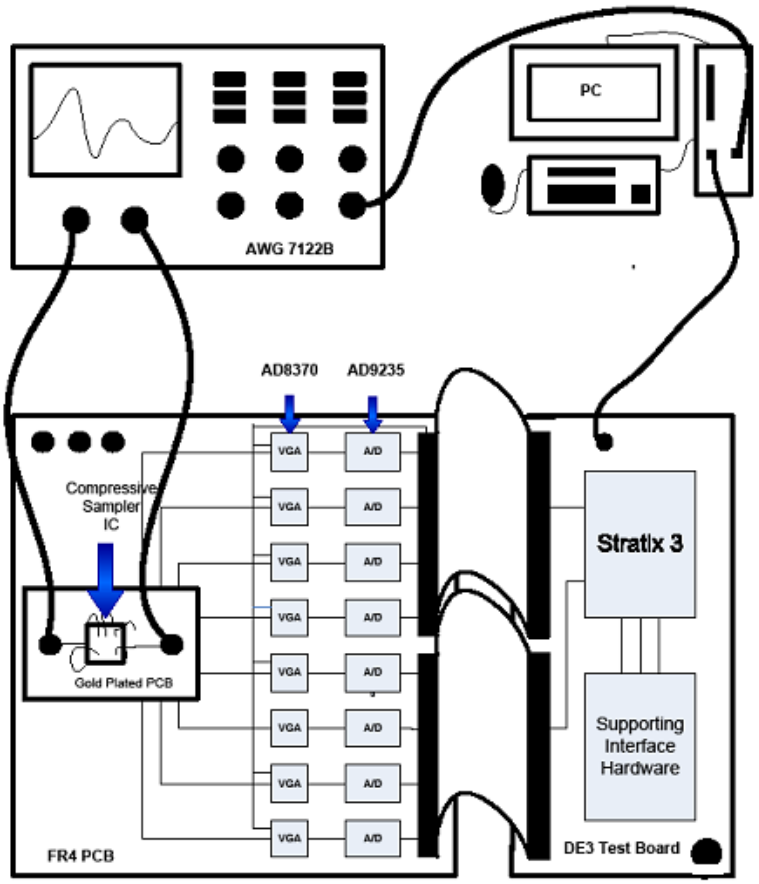


Figure 2.13: Cartoon of the calibration procedure. The arbitrary waveform generator (AWG) generates basis functions $x(t) = e^{i2\pi kt/T}$, the RMPI chip generates measurements which are recorded on the Stratix 3 FPGA, which then uploads to a computer. The computer controls the AWG and tells it which grid frequency k to use.

Φ and b) closely match x?

Note that the first item is more strict than the second item. If the second condition holds, and $\tilde{x} \simeq x$, this implies that $\Phi\tilde{x} \simeq S(x)$ but does not imply that $\Phi x \simeq S(x)$. For example, suppose Φ is an incorrect model of S because it uses a chip sequence which is shifted forward or backward in time by an amount $n\Delta T$. This has a strong effect on the measurements of any particular signal, and $\Phi(x) \neq S(x)$. However, this is not actually significant, since the recovery algorithm will just recover the estimate $\tilde{x}(t) = x(t + n\Delta T)$, which is a systematic error that is of little consequence.

Validation of the first type is considered in Figure 2.14, which plots the values of $S(x)$ and Φx for a signal x of medium (left) and small (right) amplitudes. The measurements S are performed with the Simulink model, and include a constant level of noise z. Also shown is $\Phi(x+z)$, which should be a better match to the Simulink model than just Φx; it does provide a better match, but not a perfect match, since white noise is not band-limited and is not well represented in the finite-dimensional signal model. Without the effect of noise, the agreement between $S(x)$ and Φx is excellent, but this

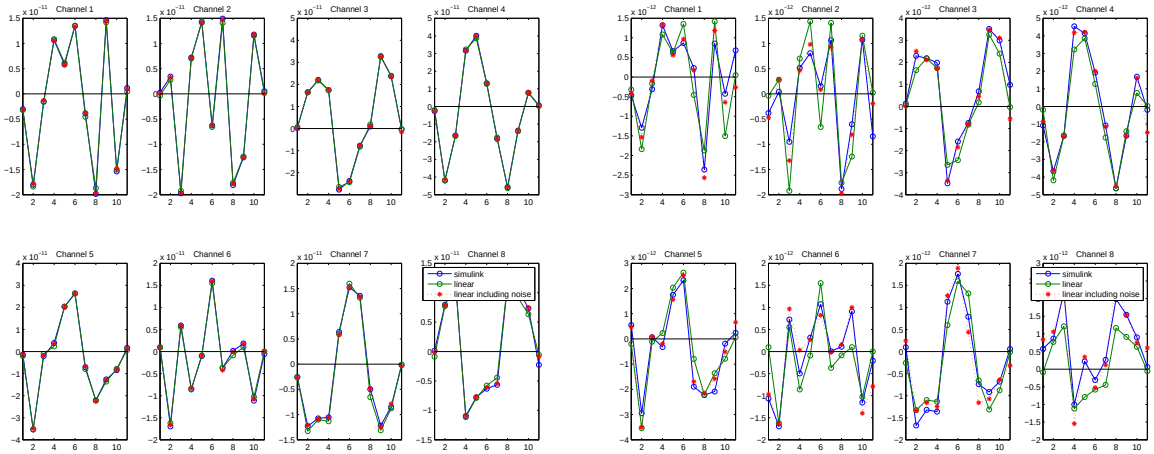


Figure 2.14: Testing if $S(x) = \Phi x$. Left: medium amplitude input. Right: small amplitude input, so noise has a significant effect. A large amplitude input is not shown because agreement is nearly perfect.

is because Φ was calibrated using the same Simulink model. In real hardware calibration, we would not expect such agreement.

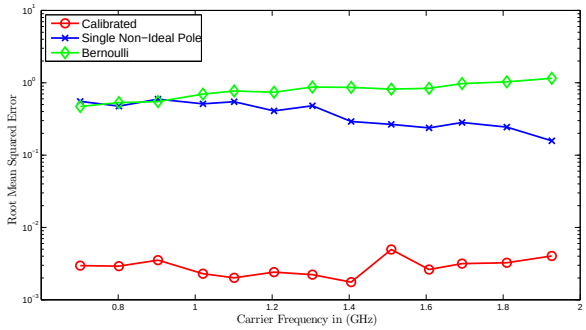


Figure 2.15: Reconstruction error for different types of calibration, as a function of frequency. Calibration is highly beneficial.

Figure 2.15 shows results from validation of the second type, plotting RMSE of the recovered signal, as a function of carrier frequency. Noisy samples $S(x)$ are generated with a Simulink model using non-idealities. Reconstructions are performed using three models of Φ: the simple ± 1 signed Bernoulli model, the single-pole model, and the calibrated model. As expected, the calibrated model performs much better. Also note that the single-pole model performs better than the Bernoulli model at high frequencies, which is reasonable since the Bernoulli model (which is equivalent to quadrature evaluation of an integral) is most accurate at low frequencies.

The value of N has so far been considered arbitrary. As N increases, the DFT approximation of the signal $x(t)$ becomes increasingly accurate, but typically a value $N = 1024$ provides sufficient approximation to the band-limited signal and it is not necessary to go larger.

However, there is a value $N_{\text{min}} = \text{LCM}(N_{\text{int}}, N_{\text{chip}})$ (LCM is the least common multiple) which is the smallest N that is needed to fully characterize the system. Both the chipping sequence and the

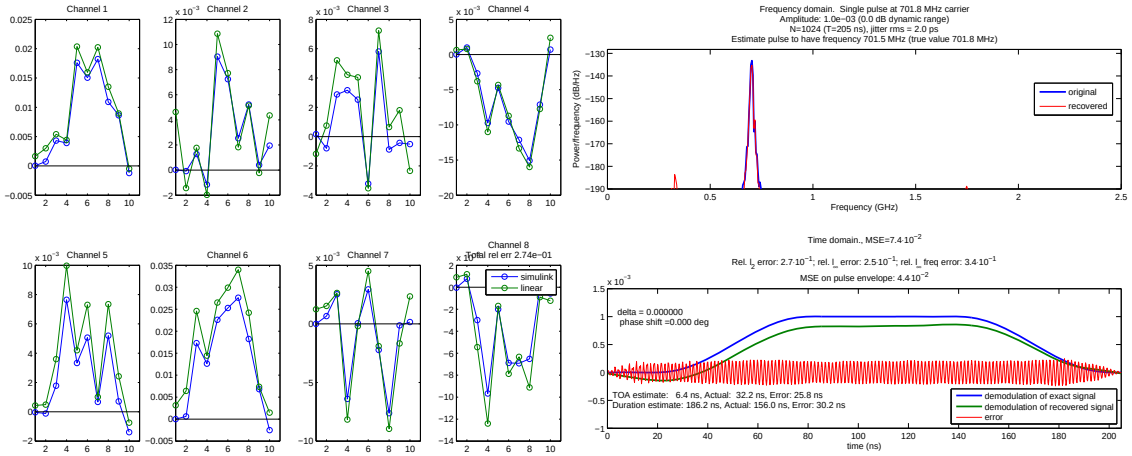


Figure 2.16: Phase blind calibration, using Simulink to generate "real" data, and using a simple Bernoulli model for Φ_{crude}. The Bernoulli model is inaccurate and does not capture phase response of the system, but it is sufficiently accurate to allow estimation of the phase of input signals since this is a massively overdetermined problem. From this phase estimate, a linear model Φ is created. The plot on the left shows how well Φx predicts measurements $S(x)$ from a test pulse; and the right plot shows reconstruction of this test pulse using. In both plots, the errors appear to be systematic in a harmless manner, affecting mainly the amplitude.

ADC samples are periodic, but the actual rows of the Φ matrix will not repeat until both periodic sequences line up. For $N_{\text{int}} = 100$ and $N_{\text{chip}} = 128$, this is $N_{\min} = 3200$. The system is fully characterized by a $\Phi_{\min}$ matrix with 3200 columns. To simulate a larger system, a new Φ matrix can be created by using copies of $\Phi_{\min}$ on the diagonals.

2.3.5 Phase blind calibration

The calibration procedure builds up the N rows of the frequency-domain matrix using

$$\Phi_{\text{FFT}}(:, k) = S(e^{i2\pi kt/T}) = S(\cos 2\pi kt/T) + iS(\sin 2\pi kt/T)$$

where the last equation follows from linearity of S and Euler's formula. The signals sin and cos are said to be "in quadrature" because they have 90° phase difference. This is convenient but not necessary: the measurement $S(e^{i2\pi kt/T})$ can be determined by measuring any two pure tones of frequency kt/T as long as they have distinct phases modulo 180°, since then the two input tones are linearly independent. The calibration is more robust if the phases are close to quadrature.

For the moment, we will consider the problem of just measuring a single-tone $\cos(2\pi kt/T + \theta)$. For calibration using hardware, an arbitrary waveform generator (AWG) can very accurately produce a sinusoid at the frequency kt/T; see Figure 2.13 for a diagram of the experimental setup. The issue with this procedure is that the phase θ of the incoming signal is unknown. The AWG cannot control phase, nor is it possible to accurately estimate the phase delay during travel from the AWG to the chip.

We propose the following method to estimate the phase. This requires a previous estimate of the

system, Φ_{crude}. This can be created from any of the methods described above, such as a Bernoulli model, or a Simulink calibrated model. The better this estimate is, the more accurate the phase estimation will be. Let $b = S(\cos 2\pi kt/T + \theta)$ be the measurements. Then we solve

$$\hat{\theta} = \measuredangle \operatorname*{argmin}_{A \in \mathbb{C}} \|b - \Phi_{\text{crude}} A e^{i2\pi kt/T}\|_2 \tag{2.3.6}$$

where t is a Nyquist time grid. This estimation is a 2-dimensional least squares problem which can be efficiently solved, similar to the method mentioned in §2.4.2.

It is likely that the crude estimate Φ_{crude} has different alignment than the true hardware, so the estimate $\hat{\theta}$ will be off. The biggest source of error is a shift in the bit sequence. Thus the above procedure can be repeated for a shifted version of Φ_{crude}. For every shift, we generate a new estimate of the phase; using this phase estimate, we compute the sample error, and pick the shift that has the least error; see Figure 2.17. If Φ_{crude} is a Bernoulli matrix, then shifting it is trivial. For a Simulink calibrated matrix of size $N_{\min}$, exactly $N_{\min}/N_{\text{chip}}$ of the possible N_{int} shifts can be easily generated by shifting the matrix. If N_{chip} and N_{int} are relatively prime, this is all of the shifts. In our system, $N_{\text{chip}} = 128$ and $N_{\text{int}} = 100$, so only 32 of the possible 128 shifts are generated by shifting the rows and columns of Φ_{crude}. To generate the other shifts, we need to build up the impulse response of 3 other Φ_{crude} matrices, each with a shift from a different co-set.

The estimation problem is parametric, with θ being determined by 2 parameters (or 1 complex parameter), and the shifts being determined by N_{int} parameters. If T is short, then b only has a few entries, so this problem is ill-posed. However, hardware measurements can easily generate long T samples. Furthermore, the shift of the chipping sequence is fixed regardless of the signal measured, so all the data can be combined to estimate the best shift of the chipping sequence. Hence the problem is estimating about 256 parameters from a dataset of size on the order of 10^4 to 10^6, and hence should be extremely accurate. The shift of the PRBS is estimated separately for each channel since each channel may see slightly different delays, but the shifts are expected to be similar, so this helps the estimation.

The crude model Φ_{crude} may not account for some complicated properties of the system, such as the phase response of the various filters, and this will lead to systematic errors in the phase estimation which are frequency dependent. However, preliminary tests (using Simulink as the "real" device, and a Bernoulli Φ_{crude}) suggest that the error is a smooth function of phase and also concentrated at low frequency. Thus it may be possible to correct for this; even if it is not corrected, the effect on reconstruction may be small; see Figure 2.16.

We now return to the issue of finding the response of the system to two signals of the same frequency but of different phase. Let θ_1 and θ_2 be the phases of the two input tones, each of grid frequency kt/T for some integer $|k| \leq N$. One possibility is to separately estimate θ_1 and θ_2 in

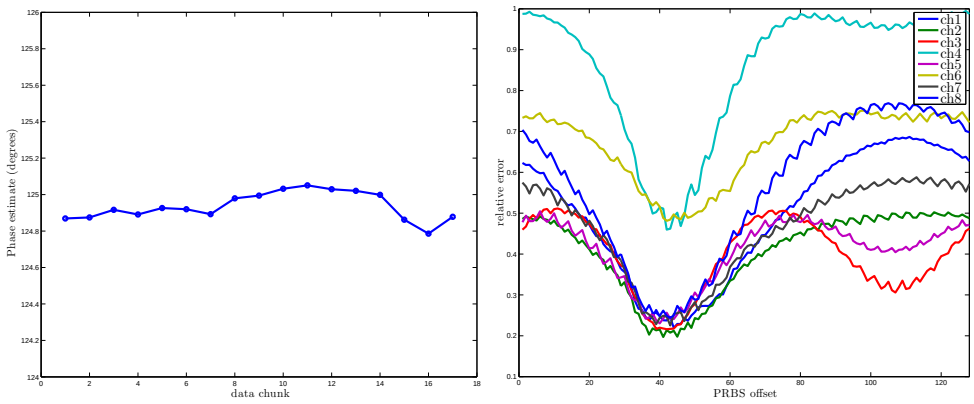


Figure 2.17: Phase blind calibration. Showing preliminary results from late March, 2011, using a single frequency input. This is real hardware data, using a Bernoulli Φ_{crude}. Left: the estimated phase for different data sets (single channel shown). Right: since there is no absolute time reference, the PRBS sequence needs to be aligned. The estimated shift of the sequence is the location of the minimum. Each channel shows a similar minimum, which indicates the method works. This is calculated from a single data set; by including more data sets, the data will improve.

the manner just described. A more accurate method is to record $2N_{\text{min}}$ measurements of a single input tone; let $T_{\text{min}} = N_{\text{min}}\Delta T$. Recall that $T = N\Delta T$. If the input is $x(t) = \cos(2\pi kt/T + \theta_1)$, then we consider $x_1 = x(t)$ as the first signal, and $x_2 = x(t + T_{\text{min}})$ as the second signal. This is equivalent to $x_2 = x(2\pi kt/T + \theta_2)$ where $\theta_2 = \theta_1 + 2\pi kT_{\text{min}}/T$. Thus x_2 is linearly dependent on x_1 whenever $kT_{\text{min}}/(2T) = 2kN_{\text{min}}/(N) \in \mathbb{Z}$. If $N > N_{\text{min}}$, then there are some frequencies k for which this is not an integer, and hence we simply use the second set of measurements to complete the characterization of this frequency (and θ_1 and θ_2 are now jointly estimated). If the quantity is not an integer but lies close to an integer, then we can improve robustness by skipping ahead by a few more blocks of N_{min}.

For frequencies k where this is an integer (if $N = N_{\text{min}}$, this is all frequencies) then $\theta_2 = \theta_1 \mod 180$ so the samples are linearly dependent. In this situation, we generate several sets of measurements from the AWG, and most likely each waveform will have a different phase. These phases are estimated separately and then combined. Note that if N is prime, then this situation will never occur, though it may not be robust. A large prime value of N, with $N \gg N_{\text{min}}$, will be a more robust choice, though it will require more measurements.

When $N > N_{\text{min}}$, then the Φ and Φ_{FFT} matrices have unnecessary columns. If necessary, the time-domain version Φ can just remove the extra columns *after* being created by an inverse DFT of the larger Φ_{FFT}. This will induce a few errors (since the time vectors should really be thought of as the inverse DFT of the frequency vectors, not as a inherently meaningful themselves), but should not be significant.

This calibration system is in the process of being implemented on the RMPI chip. A computer controls the entire process, but it is far from trivial since several thousand sets of precise measurements must be taken. Preliminary results are shown in Figure 2.17 which indicate the

method consistently estimates phase and shift parameters. These results show the the phase and shift estimates of a 1 GHz signal collected from hardware measurements, and using Φ_{crude} for the estimation.

We note that changing the ADC rate f_{ADC}, and hence T_{int}, is one of the easiest system parameters to modify. Changing T_{int} from 100 to 96 makes a big change, since now $N_{\min} = \text{LCM}(N_{\text{chip}}, N_{\text{int}})$ is 384 instead of 3200, so it would be possible to build a calibration model using a smaller N since the only requirement is that $N \geq 384$.

An alternative to the phase blind calibration is to use the accurate SPICE model for calibration, since then the input signal is known exactly. However, since the Cadence software is run on a single computer, it would take about a month of computational time to record $N = 1024$ signals, so we favor the hardware-based approach.

2.4 General design considerations

This section and the following two sections consider some important design choices. Below are some of the major questions we aim to answer.

- At what rate does the chipping sequence need to operate?
- What periodicity does the chipping sequence need?
- What characteristics should the PRBS have?
- Given a fixed information rate (400 MHz), how many channels are necessary?
- How closely must the integrating filter approximate an integrator?
- What bandwidth does the integrating filter need need?
- Which blocks of the system must be highly linear?
- Which blocks of the system are affected by clock jitter?

We are also concerned about whether the system is robust to non-idealities, such as thermal noise, over which we have little control. This section discusses a few of the design decisions and rational behind them. The chipping sequence and integrator are discussed in detail in sections §2.5 and §2.6, respectively. By necessity, components are introduced one-at-a-time, but of course it is impossible to simulate changing one part of the system without first having a model for the entire system, so occasionally we reference a system block out of order.

The design parameters are coupled to each other. For example, a PRBS with short repetition rate can be partially compensated for by using a nearly ideal integrator, so these two systems

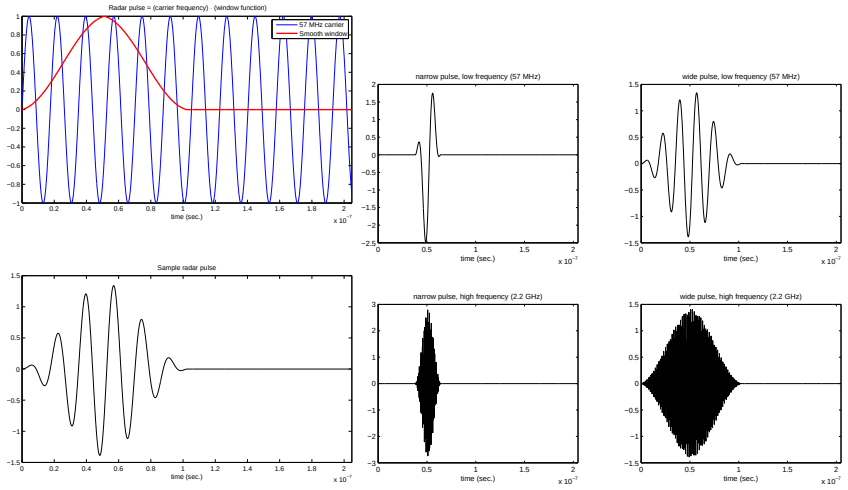


Figure 2.18: (Left) A radar pulse consists of a pulse envelope and a carrier frequency. (Right) For initial tests beginning in 2008, four types of pulses were used in "corner" sims. The four test pulses cover low frequency (57 MHz) to high frequency (2.2 GHz), and narrow (20 ns) to wide (100 ns), all with smooth pulse envelopes.

cannot be considered in isolation. Unfortunately, due to time limitations, it is not practical to optimize all the parameters at once, and instead we perform one round of "coordinate-descent": fix all parameters but one, and then find the best value for this parameter; then repeat for the other parameters. Furthermore, as our simulation model gains complexity over time, certain previous design decisions become fixed, so it is not simple to revisit old choices. Another complication is that recovery, discussed in §2.7, involves many parameters as well, so the overall parameter space is high dimensional. However, results indicate that the approach has worked remarkably well.

2.4.1 Test signals

The performance of the system was primarily evaluated on its accuracy for reconstructing single pulses, although multiple pulses were tested on occasion. To capture various types of behavior, it is important that the system tests a wide range of carrier frequencies and pulse widths. The simulations consisted of either "corner simulations" using extreme types of pulses (see Figure 2.18) or a combination of systematic parameters, such as pulse length, and random parameters, such as phase and frequency of the carrier.

The window of the pulse was occasionally changed, since this affects the spectral sparsity of reconstruction. Window functions varied from smooth windows, like a Gaussian pulse, to trapezoidal windows. Most of the recent simulations used filtered trapezoidal windows. For the compressive matched filter recovery described in §2.7.1, the pulse window has little effect, since spectral leakage is not an issue. In fact, a matched filter works best when the window is a pure box-car with infinitely fast initial rise time.

Simulations were careful to either use frequencies chosen uniformly at random in $[0, f_s/2]$, or

fixed frequencies that are off-grid. The system can perform much better with on-grid signals, but this is an unrealistic assumption.

Testing dynamic range was done in two manners. The first test was with a single signal, and varying the amplitude. This gives an absolute limit to the dynamic range. The other test was with two signals, and varying the relative amplitudes. This limit is always less than the single-pulse dynamic range since the large pulse introduces extra difficulties for the small pulse, such as creating extra error due to jitter, as well as creating non-linear effects.

2.4.2 Error metrics

For synthetic simulations, the true signal x_0 is known, so this can be used as a reference. The most common metric is the root-mean-squared error (RMSE), which we define as

$$\mathrm{RMSE}(x) \triangleq \frac{\|x - x_0\|_2}{\|x_0\|_2}. \tag{2.4.1}$$

This metric is perhaps more correctly referred to as *relative* RMSE. On occasion, we reference MSE, which is just the square of this quantity. One benefit of RMSE and MSE is that they are the same regardless of whether we consider time-domain samples or frequency components. Other reconstruction-based metrics we consider are $\|x - x_0\|_\infty$ and $\|\hat{x} - \hat{x}_0\|_\infty$, where $\hat{x}$ is the (non-unitary) DFT of x.

For pulses, the parameters of interest are the pulse descriptor words (PDW), such as time-of-arrival, so these can be the basis of error metrics, though we do not use them much. The envelope of the function itself is also of much interest, so the error can be cast in terms of this. To find the envelope of a pulse x, aka demodulation, we perform the following, which requires an estimate of the carrier frequency f (found via the Welch method or similar):

1. Using f, estimate the phase ϕ of the carrier. This is done by matched filter in the following manner:

   $$(a, b) \leftarrow \operatorname*{argmin}_{a,b} \|x - a\sin(2\pi t f) - b\cos(2\pi t f)\|$$

   where t is a time grid. This is a 2D least-squares problem that can be solved efficiently. The phase is estimated $\phi = \tan^{-1}(b/a)$. The operation is similar to phase estimation of the calibration scheme, although in this case the measurements are from full Nyquist-rate samples.

2. The signal is mixed with $\sin(2\pi t f + \phi)$ and then digitally low-pass filtered by truncating its DFT coefficients.

Another demodulation method is taking the absolute value of the Hilbert transform of a signal, since the Hilbert transform puts the phase information into the complex phase of the analytic signal.

Number of channels, r_{ch}	1	2	4	8	16	32
f_{ADC} (in MHz)	400	200	100	50	25	12.5
T_{ADC} (in ns)	2.5	5	10	20	40	80
N_{int}	12.5	25	50	100	200	400

Table 2.3: Various system values as a function of r_{ch}, using a fixed overall sampling rate of 400 MHz. The current RMPI uses 8 channels.

However, this approach works poorly when there are several signals present. Our demodulation method works well in multi-signal environments, since the filtering discards the background signals. The main disadvantage is that it requires an accurate estimate of the frequency, but it is not overly sensitive to this, so we find that it works well in practice.

For some of the tests on hardware, the original signal x_0 is unknown, or only approximately known, in which case "recovery" is more subjective, and usually based on visual inspection of time and frequency plots.

Some reconstruction methods, such as the matched filter, are more appropriately analyzed only in the frequency domain. Error is then recorded as a function of how far the estimated frequency is from the true frequency, $|f - f_0|$. Often this type of error is thresholded and the reconstruction is declared a "success" if $|f - f_0| < f_{\text{cutoff}}$, and a "failure" otherwise, with f_{cutoff} typically 1 MHz. These reconstructions can be aggregated into a sample recovery percentage statistic.

2.4.3 Number of channels

For a fixed ADC sampling rate, increasing the number of channels will always improve performance. However, each channel requires extra power; in the extreme case, we could have 100 channels each sampling at $f_{\text{ADC}} = 50$ MHz, but this is just a conventional channelized receiver. But on the other hand, a single channel receiver at $f_{\text{ADC}} = 50$ MHz would undersample too much.

In order to analyze the appropriate number of channels, consider a fixed information rate system that undersamples the Nyquist rate by 12.5×. This means samples are taken at an average rate of 400 MHz, since $f_s = 5$ GHz. The final Caltech RMPI design uses 8 channels at 50 MHz each, whereas the Northrop Grumman design uses 4 channels at 100 MHz each. This section discusses this trade off.

From a theoretical point-of-view, more channels are better. The matrix representation Φ of such a system closely approximates a signed Bernoulli matrix, which is known to be incoherent with any basis. In contrast, a single channel operating at a fast rate becomes closer to a simple underclocked regular ADC, which is not a good measurement system.

However, multi-channel systems cause quite a few problems. Consider the situation of recovering a large pulse and a small pulse. If these pulses are separated in time, then we don't expect the

large pulse to have any effect on the recoverability of the small pulse. If the pulses overlap in time, then because clock jitter induces error proportional to the amplitude of the largest signal, the small pulse will see a relatively large amount of additive noise which will make recovery difficult. If the RMPI has many channels, then each channel only takes measurements every T_{ADC}, where T_{ADC} can be quite large. This means that even if a big pulse and small pulse do not overlap in time but are within T_{ADC} of each other, then they will affect each other. As the number of channels increases, T_{ADC} increases, so this is more likely to occur.

The longer integration window means that more noise is added to each measurement. By itself, this does not affect the signal-to-noise ratio, since more signal is included in each measurement, but it also means that the noise included is more correlated (i.e., covariance matrix $\Phi\Phi^T$), so it is more important to account for this correlation in recovery algorithms. A longer T_{ADC} also increases the chance of clipping the signal by reaching the maximum amplitude; see §2.8.1.5.

Another effect is caused by the non-ideal integration. Consider a single-pole integrator, and assume the time constant (which is the inverse of the pole location) is constant, regardless of the number of channels. The time constant measures how long it takes an equivalent capacitor to discharge or charge e^{-1} of its max capacity. For the initial RMPI design, the time constant was 22 ns (this has been subsequently improved). In a 32 channel system, $T_{\mathrm{ADC}} = 80$ ns, which means that samples arriving early in the time interval will be attenuated by about $e^{-4} \approx 0.018$. The measurements are heavily weighted toward the end of the interval. To a crude approximation, we are only taking about 20 ns worth of information, so this may cause problems when sampling very short pulses. Decreasing T_{ADC} helps the problem.

For reconstruction, having many channels causes more difficult calibration and more difficult reconstructions. To see how reconstructions are affected, consider taking m measurements per channel, so the system matrix Φ has $M = mr_{\mathrm{ch}}$ rows. It is convenient to choose the number of columns, N, to be a multiple of m. However, this is not always possible for two reasons: the first is that analysis with the Gabor dictionary is much more efficient when N is a power of 2, and in fact the code only handles this case. The second reason is that we would like to consider the possibility of *staggering* each channel (to be discussed in §2.6), which means that the ADC samples of each successive channel are offset by a small amount Δm from the previous channel. Thus we want all of $N, N + \Delta m, \ldots, N + (r_{\mathrm{ch}} - 1)\Delta m$ to all be divisible by m, and the offset is most naturally $\Delta m = N_{\mathrm{int}}/r_{\mathrm{ch}}$. This is impossible to satisfy in general. Whenever there is mismatch, it leads to zeros in the Φ matrix, since we cannot make use of a partial sample. For a small number of channels, this is not an issue, but with a larger number of channels, it happens with increasing frequency. For example, with 8 channels, N_{int} is 100, and if $N = 1024$ and no stagger is used, this leaves 24 columns of zeros in the Φ matrix and each channel takes $\lfloor N/N_{\mathrm{int}} \rfloor = 10$ measurements. Now consider staggering each channel by an offset of about 12.5 Nyquist samples on average, so

that channel two is shifted by 25 Nyquist samples. The last sample of channel two runs from 926 to 1025, which exceeds the length of N, so this sample is discarded, and this channel only keeps 9/10 of its data. For a fixed length of N, this loss of data is exacerbated as the N_{int} increases, so it gets worse for a many-channel system.

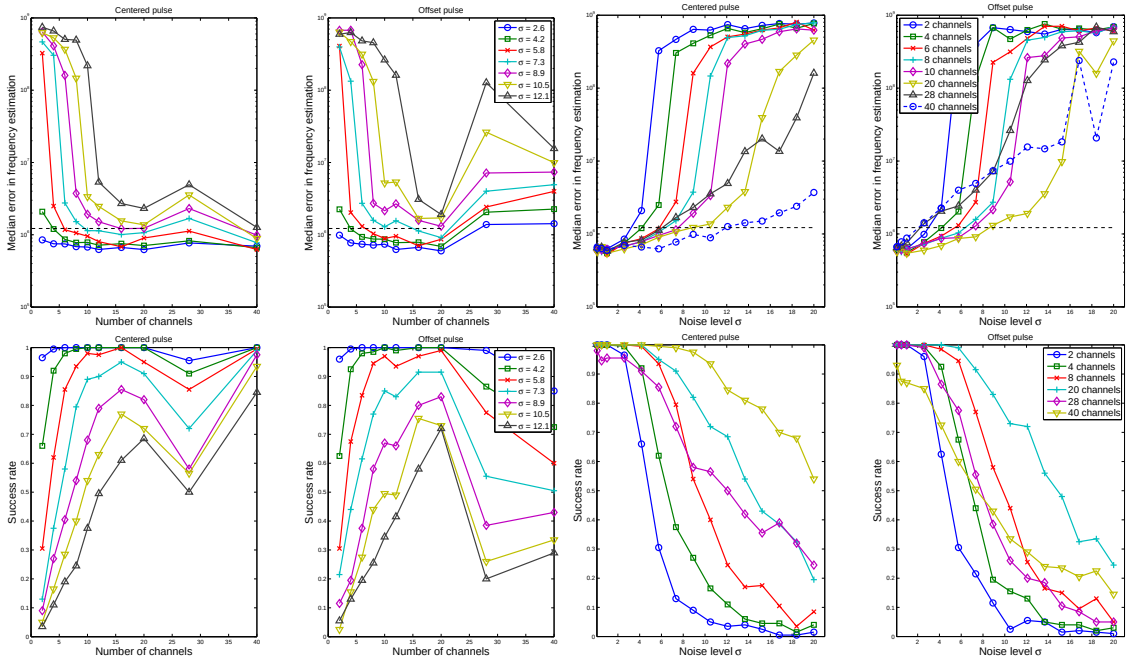


Figure 2.19: Matched filter simulations. Top row plots the error of frequency estimation, bottom row uses a 1 MHz frequency estimation cutoff to determine "successes" and "failures", and plots the success rate (using 200 independent trials). Plots either show results as a function of noise standard deviation σ, and each line represents a fixed r_{ch}; or plots as a function of r_{ch}, and each line represents a fixed σ. The title of the plot is "centered" or "offset", which refers to the location of the pulse. Offset pulses are near the boundary, and are more affected by missing measurements whenever N_{int} does not exactly divide N. In all plots, the dashed black line represents the resolution of an FFT of length N.

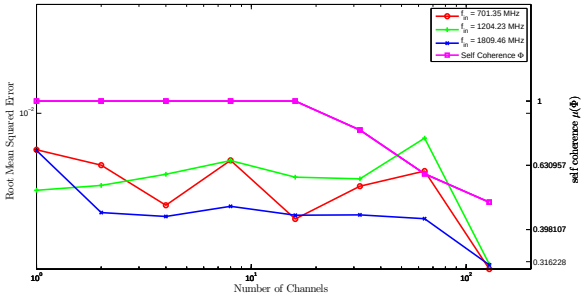


Figure 2.20: The RMSE error as a function of the number of channels (but fixed overall data rate at 400 MHz) for a signal with SNR = 0 dB, using Φ from the Bernoulli model

To study the effect of these considerations, simulations were run using Φ matrices representing systems of various channels. Since we do not have a working Simulink model of an arbitrary channel system, the Φ matrices were chosen with signed Bernoulli entries, but in an otherwise realistic fashion (the PRBS repeated, only whole measurements were kept, and stagger used for some tests).

Figure 2.19 shows the median results of 200 random simulations per parameter. In order to collect statistics, we used the ideal matched filter reconstruction because it is extremely fast and has no parameters. The signal was a pulse of length 100 ns and the phase and frequency were chosen uniformly at random, with $N = 1024$. The matched filter must calculate normalizing factors for every Φ matrix, which is relatively slow, so for every parameter, two Φ matrices were generated and then 100 random signals were used with every Φ matrix. This version of the matched filter searches only over frequency, phase, and amplitude, and used *a priori* knowledge of the location and size of the pulse. Noise was added to the measurements *after* the measurement operation: $b = \Phi x + z$.

The results are shown in Figure 2.19. We consider several levels of noise to get a full picture of how the system performs. We tested both centered pulses and pulses offset to the side of the interval, since our hypothesis is that zero columns in the Φ matrix, which always occur at the edges, will affect these offset pulses. Indeed, this is what we observe. Consider the four plots on the right. The 40 channel system is amazingly robust to noise when we measure a centered pulse, since for a success rate of 50%, it can withstand noise up to $\sigma = 20$, whereas the 8 channel receiver can only withstand up to $\sigma = 10$. Yet on the *offset* pulse, the 8 channel receiver outperforms the 40 channel receiver since the 40 channel Φ matrix has more incomplete measurements that must be discarded. Looking at the left four plots shows that, depending slightly on the noise level, the best design is 16 or 20 channels. The plots are not monotonic, which may seem surprising, but this is an artifact due to using any fixed N, since some large numbers divide into this with smaller remainder than do smaller numbers. For $N = 2048$ or $N = 4096$, the maximum would be in a slightly different location. As computing power increases and it is easier to perform reconstructions with extremely large N, it would become increasingly beneficial to use large channels system.

Figure 2.20 show results using a similar Bernoulli matrix and using full ℓ_1 reconstruction (with reweighting), so this is a more realistic scenario. Due to the increased computational complexity, only three signals (low, medium, and high frequency) are used, and only one trial is performed per channel number. Because of the full reconstruction, we are able to examine the root-mean-squared error (RMSE). Also plotted is the self-coherence $\mu(\Phi)$ (that is, the maximum absolute value of a non-diagonal entry of $\Phi^T\Phi$), which decreases as the number of channels increases. A small self-coherence is a sign of a useful compressed sensing matrix, so this corroborates the intuition that a large number of channels is good for theoretical purpose. From these limited trials, which did not test an offset pulse like in the earlier figure, we only see benefits of an increased number of channels when the channels approach 100, which is impractical.

From a system design point-of-view, more channels are more difficult to design, since it becomes important to isolate each channel to prevent cross-talk, and it also requires considerable power for the clock distribution system.

Because of this hardware complexity, 8 or 12 channels is a practical limit. The simulations

suggest that increasing channels up to a point, probably around 20, may significantly improve the performance of the system. However, we choose 8 channels for the final design because of prohibitive design complexity for 12 or more channels. The Northrop Grumman design uses 4 channels, and the simulations suggest it will be more sensitive to noise than the Caltech version, but it will benefit from simpler design.

2.5 Chip sequence

Perhaps the most important design choice is the chip sequence, which we refer to synonymously as the PRBS (pseudo-random bit sequence). The RMPI chip sequence is periodic, but for the sake of analysis, we will consider both periodic and infinite chip sequences. We may abuse notation and write $c(t)$ for the time domain chip sequence, but also use $\{c_n\} = \{c_0, c_1, \ldots, c_{N_{\text{chip}}-1}\}, c_n \in \{-1, 1\}$ to be the pseudo-random bit sequence (PRBS) that defines $c(t)$. Specifically,

$$c(t) = \sum_{n=-\infty}^{\infty} c_n g(t - n\Delta T) \tag{2.5.1}$$

where g is the rectangular window that is 1 inside $[-\Delta T/2, \Delta T/2]$ and 0 elsewhere. When $N_{\text{chip}} < \infty$, we implicitly use $c_n = c_m$ whenever $n \equiv m \mod N_{\text{chip}}$. There are two time periods associated with the chip sequence: ΔT is the Nyquist period, and is the duration of the bit. The RMPI operates at 5 GHz and so ΔT is 200 ps. Throughout this thesis, this rate if fixed. The other time period is $T_{\text{chip}} \triangleq \Delta T N_{\text{chip}}$, which is the periodicity of the chip sequence. In the final RMPI design, $N_{\text{chip}} = 128$, but in this chapter will will consider varying this. For a summary of notation, see Table 2.1.

In the Caltech RMPI, the chipping bits $\{c_k\}$ are loaded into a memory unit, and thus can be arbitrarily chosen. We will assume they are randomly and independently chosen, with equal probability of $+1$ and -1. The Northrop Grumman RMPI does not allow arbitrary chip sequences and instead uses a Gold code, which is a mixture of simple linear feedback shift registers (LFSR).

For the rectangular window g, an elementary calculation gives

$$\mathcal{F}g(f) \triangleq G(f) = \frac{\sin \pi f \Delta T}{\pi f} = \Delta T \operatorname{sinc}(f \Delta T) \tag{2.5.2}$$

where $\operatorname{sinc}(x) \triangleq \sin(\pi x)/(\pi x)$ at $x \neq 0$ and $\operatorname{sinc}(0) = 1$. It would be interesting to consider alternatives, such as smoother windows, so that the spectrum is more concentrated in the $[-f_s, f_s]$ band, but this may be difficult to implement in hardware and was not considered in our studies.

2.5.1 Spectral properties of the chip sequence

2.5.1.1 Infinite period

We wish to find the power-spectral density of a random chip sequence of infinite length. A chip sequence is not a stationary process, since if times t_1 and t_2 both lie inside the same ΔT interval, then the correlation is exactly 1, and if they live in different ΔT intervals, the correlation is 0, and hence there is some absolute time dependence. Signals with this discrete structure are a type of cyclostationary process.

The autocorrelation function of a random signal is defined

$$r_{tt}(t,\tau) = \mathbb{E}\, x(t)\bar{x}(t-\tau) \tag{2.5.3}$$

and if the signal is wide-sense stationary, then so is the autocorrelation and we have $r_{tt}(t,\tau) = r_{tt}(\tau)$. In this case, the Wiener-Khinchin theorem shows how to calculate the power spectral density (PSD) Ψ:

$$\Psi(f) = \mathcal{F} r_{tt}(\tau). \tag{2.5.4}$$

The theorem is useful because stationary signals are not necessary square-integrable, and so care must be taken when using the Fourier transform. When x is square-integrable, then the direct formula for the PSD is:

$$\Psi(f) = \lim_{T\to\infty} \frac{\mathbb{E}\,|\hat{x}(f)|^2}{T} \tag{2.5.5}$$

where $\hat{x}$ is understood to be the Fourier transform of x restricted to the domain $[-T/2, T/2]$.

Let g be a boxcar signal; that is, it is zero everywhere except it is 1 on the interval $[0, \Delta T]$. Then a chip sequence $c(t)$ with infinite period can be viewed as a superposition of shifted versions of $\pm g$. Neither the shifting nor the ± 1 affects $|G(f)|$, so intuitively the PSD of an infinite-period chip sequence is just $|G(f)|^2/T$ using (2.5.5). It turns out this is correct, and we can make it rigorous as follows, using a subtle trick (see, e.g., [BM90]) to make $c(t)$ stationary. This is also known as the time-average autocorrelation function [Pro01].

The autocorrelation of $c(t)$ is

$$r_{tt}(t,\tau) = \mathbb{E}\, c(t)c(t+\tau) = \sum_{j=-\infty}^{\infty}\sum_{k=-\infty}^{\infty} \mathbb{E}(c_j c_k) g(t-k\Delta T) g(t+\tau-j\Delta T) \tag{2.5.6}$$

$$= \sum_{k=-\infty}^{\infty} g(t-k\Delta T) g(t+\tau-k\Delta T) \tag{2.5.7}$$

due to the independence of c_k and c_j. This is not stationary because it depends on τ *and* t.

The insight is that a time shift of $c(t)$ will have the same PSD (since it only changes the phase of the Fourier transform), so we'll consider a random shift of the chip sequence by an amount

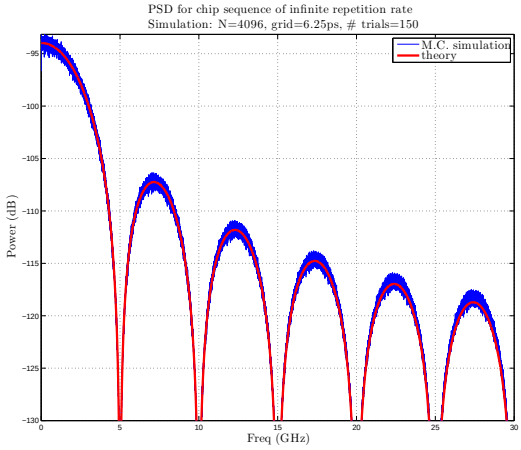


Figure 2.21: PSD Ψ of a chip sequence with infinite repetition rate; see (2.5.9). This plot was generated by averaging many sample realizations of very long chip sequences. The chip sequence is modulated at $f_s = 5$ GHz.

$\phi \sim U[0, \Delta T]$. The expectation is now taken over c_i and over ϕ.

Taking the expectation over ϕ is just the integral from $[0, \Delta T]$ divided by ΔT. Each term k in the sum is an integral from $[k\Delta T, (k+1)\Delta T]$, and thus the t dependence integrates out:

$$r_{tt}(t, \tau) = r_{tt}(\tau) = \frac{1}{\Delta T} \int_{-\infty}^{\infty} g(t) g(t+\tau) dt.$$

We are now in position to use the Wiener-Khinchin theorem (2.5.4), and take the Fourier transform with respect to τ (we can change the order of integration using Fubini's theorem since the integrands are well-behaved). Using $\mathcal{F} g(t+\tau)(f) = e^{i2\pi ft} G(f)$, we have

$$\begin{aligned} \Psi(f) &= \frac{1}{\Delta T} \int_{-\infty}^{\infty} g(t) e^{i2\pi ft} G(f) dt \\ &= \frac{G(f)}{\Delta T} \int_{-\infty}^{\infty} g(t) e^{i2\pi ft} dt \\ &= \frac{G(f)}{\Delta T} \bar{G}(f) = |G(f)|^2 / \Delta T \end{aligned} \tag{2.5.8}$$

which agrees with the intuition. In our system, g is a rectangular window, so using (2.5.2),

$$\Psi(f) = \frac{1}{f_s} \left(\frac{\sin \pi f / f_s}{\pi f / f_s} \right)^2. \tag{2.5.9}$$

See Figure 2.21 for a plot. Most of the power falls inside the first lobe from $[-f_s, +f_s]$.

2.5.1.2 Finite period

The actual RMPI design repeats the PRBS chip sequence every N_{chip}. Since $c(t)$ is then periodic, it has a Fourier series representation, which we now derive.

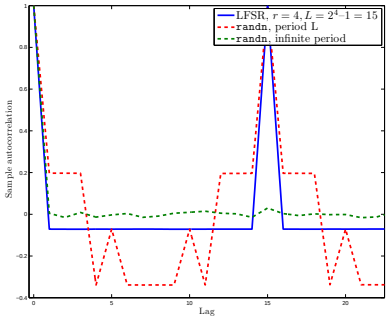


Figure 2.22: Sample autocorrelations of some PRBS sequences. The LFSR is of maximal length $L = 2^r - 1$. For lags greater than 1 and less than L, the autocorrelation of the LFSR is $-\frac{1}{L}$, whereas it is 0 in this region for a random sequence. The `randn` sequences are the signs of a pseudo-random sequence generated with Matlab's default `randn` routine which approximates a Bernoulli random variable very well.

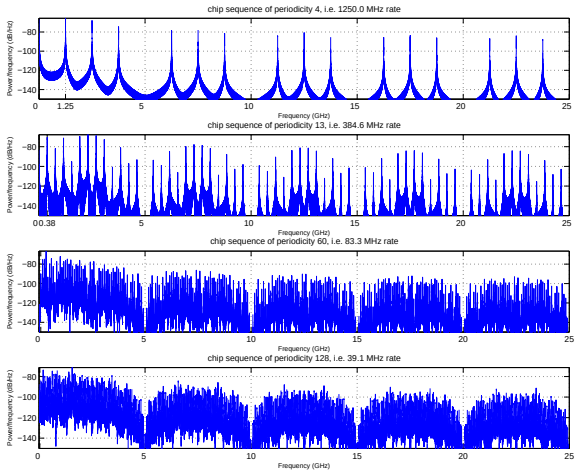


Figure 2.23: Spectrum of finite length PRBS. The longer the periodicity in the PRBS, the finer the grid spacing, which in general is advantageous. Each row is generated by a different N_{chip}. Contrast this to the smooth spectrum of a chip sequence of infinite length in Figure 2.21.

Let N_{chip} be the repetition length of the chipping sequence ($T_{\text{chip}} = N_{\text{chip}}\Delta T$; see Table 2.1 on page 45). Then the Fourier series is

$$\begin{aligned}
\hat{c}[k] &\triangleq \frac{1}{T_{\text{chip}}} \int_0^{T_{\text{chip}}} c(t) e^{-i2\pi kt/T} dt \\
&= \frac{1}{T} \sum_{n=0}^{N_{\text{chip}}-1} \int_{n\Delta T}^{(n+1)\Delta T} c_n e^{-i2\pi kt/T} dt \\
&= \frac{1}{T} \sum_{n=0}^{N_{\text{chip}}-1} c_n \frac{T}{-i2\pi k} \left(e^{-i2\pi k(n+1)/N_{\text{chip}}} - e^{-i2\pi k(n)/N_{\text{chip}}} \right) \\
&= \frac{\sin(\pi k/N_{\text{chip}})}{\pi k} \sum_{n=0}^{N_{\text{chip}}-1} c_n e^{-i2\pi k(n+\frac{1}{2})/N_{\text{chip}}}.
\end{aligned}$$

With the convention that $\sin(0)/0 = 1$, this holds for any integer k. Note that this would be periodic

in N_{chip} if it were not for the sinc term. For nonzero k,

$$\hat{c}[k+\tau N_{\text{chip}}] = \frac{\sin(\pi k/N_{\text{chip}}+\pi\tau)}{\pi(k+\tau N)} \sum_{n=0}^{N_{\text{chip}}-1} c_n e^{-i2\pi k(n+\frac{1}{2})/N_{\text{chip}}}(-1)^{\tau} = \frac{k}{k+\tau N_{\text{chip}}}\hat{c}[k] \quad (2.5.10)$$

and for zero k, $\hat{c}[\tau N_{\text{chip}}] = 0$ if $\tau \neq 0$.

To get the Fourier transform from the Fourier series, we haven

$$\hat{c}(f) = \sum_{k=-\infty}^{\infty} \hat{c}[k]\delta(f - \frac{k}{T_{\text{chip}}}) \quad (2.5.11)$$

which has discrete spacing T_{chip}. It is almost periodic in $N_{\text{chip}}/T_{\text{chip}} = f_s$, except for the decay of $k/(k+\tau N_{\text{chip}})$ from equation (2.5.10).

To find the power-spectral density of a finite-length sequence, we revisit (2.5.6). The double sum no longer collapses to $j = k$ but rather $j = k + j'N_{\text{chip}}$ for any integer j'. Otherwise, the same reasoning holds, and the equivalent of (2.5.8) is

$$\begin{aligned}\Psi(f) &= \frac{|G(f)|^2}{\Delta T} \sum_{j'=\infty}^{\infty} e^{-i2\pi f j' T_{\text{chip}}} \\ &= \frac{|G(f)|^2}{\Delta T}\frac{1}{T_{\text{chip}}} \sum_{k=-\infty}^{\infty} \delta(f - \frac{k}{T_{\text{chip}}}) = \frac{1}{N_{\text{chip}}}\left(\frac{\sin \pi f/f_s}{\pi f/f_s}\right)^2 \sum_{k=-\infty}^{\infty} \delta(f - \frac{k}{T_{\text{chip}}})\end{aligned} \quad (2.5.12)$$

where we used the Poisson summation formula [Mal08]:

$$\sum_{k=-\infty}^{\infty} e^{-i2\pi kt/T} = T \sum_{k'=-\infty}^{\infty} \delta(t - k'T).$$

If the PRBS is generated via a linear-feedback shift register (LFSR), the statistics change slightly, due to the extra correlation. In particular, outputs c_n of a maximal LFSR with r taps, where $L = N_{\text{chip}} = 2^r - 1$, have exactly $(L+1)/2$ values of $+1$ and $(L-1)/2$ values of -1. Furthermore, half the runs of consecutive $+1$ or -1 are of length one, one-fourth are of length two, one-eighth are of length three, etc. [PSM82]. The effect of this is seen in Figure 2.22; in contrast to a random sequence, the autocorrelation is not 0 when $\tau > \Delta T$. The properties of LFSR are relevant since the Northrop Grumman design uses Gold codes, which are based on LFSR output. Properties of LFSR sequences have been much studied since they are used in spread spectrum communication and can also be converted into 2D masks for use in coded aperture interferometry. For reference we give the one-sided PSD of a LFSR sequence with length N_{chip} [PSM82]:

$$\Psi(f) = \frac{N_{\text{chip}}+1}{{N_{\text{chip}}}^2}\Big(\frac{\sin \pi f/f_s}{\pi f/f_s}\Big)^2 \sum_{k=-\infty}^{\infty} \delta(f - \frac{k}{T_{\text{chip}}}), \quad f > 0 \quad (2.5.13)$$

and $\Psi(0) = \frac{1}{{N_{\text{chip}}}^2}\delta(0)$. This is quite similar to (2.5.12).

2.5.2 Chip design considerations

For this section, we will suppose that the PRBS coefficients are chosen randomly from ± 1, and that we use a square window g. A recent paper [HBC11] suggests using run-length-limited codes as an alternative to random coefficients, but this is beyond the scope of our work. Given our decisions, there are two major questions left unresolved:

1. At what *rate* should the chipping sequence be modulated? Until now, we have implicitly assumed that the width of the window g was $\Delta T \triangleq 1/f_s$, since this is what the final design uses. In this section, we will use the variable f_{chip} to represent the rate of the chipping sequence, so that g has width $1/f_{\text{chip}}$.

2. How often can the PRBS *repeat*? That is, how small can the periodicity N_{chip} be?

Recalling the guiding principle of the RMPI design, we want all signals in the model to generate measurements of approximately the same energy. For a probabilistic analysis, we might look for a uniform bound over all input tones using the chip Fourier series, similar to the analysis in [ERW11] for the NUS. A simpler approach is to consider just the chip PSD, which gives an idea of "average case" behavior.

Regarding the finite-period chip sequence PSD (2.5.12), there are two salient issues. The first is the sinc window, and this will be the subject of §2.5.2.1 which discusses f_{chip}. The second feature is the discrete spacing, which is caused by $N_{\text{chip}} < \infty$, and this is the subject of §2.5.2.2.

Both sections will consider the power output when faced with an input signal of the form $x(t) = Ae^{2\pi i f_{\text{in}} t}$. If $H_{\text{sys}}(f)$ is the transfer function of the system (discussed in §2.6), then the output power is

$$P_{\text{out}} = \int_{-\infty}^{\infty} |A|^2 |H_{\text{sys}}(f)|^2 \Psi(f - f_{\text{in}}) df.$$

If η_0 is the noise output power (also a function of H), then a reasonable prerequisite for the signal post-processing to recover x is that

$$P_{\text{out}} \geq \eta_0 \text{SNR}_{\text{min}}$$

for some value of SNR_{min}.

Before discussing f_{chip} and N_{chip} in detail, we note a few complications which arise. First, the power output depends on both Ψ and H_{sys}, so the chip sequence and the integrator cannot be considered independently. In future work we hope to make a combined analysis, but for now we treat them independently. The second issue is that the length of N_{chip} is not the only consideration, but that the relation of N_{chip} to N_{int} is significant. For example, if N_{chip} and N_{int} are relatively prime, the system matrix Φ needs $N_{\text{min}} = N_{\text{chip}} N_{\text{int}}$ columns to characterize it, which means the

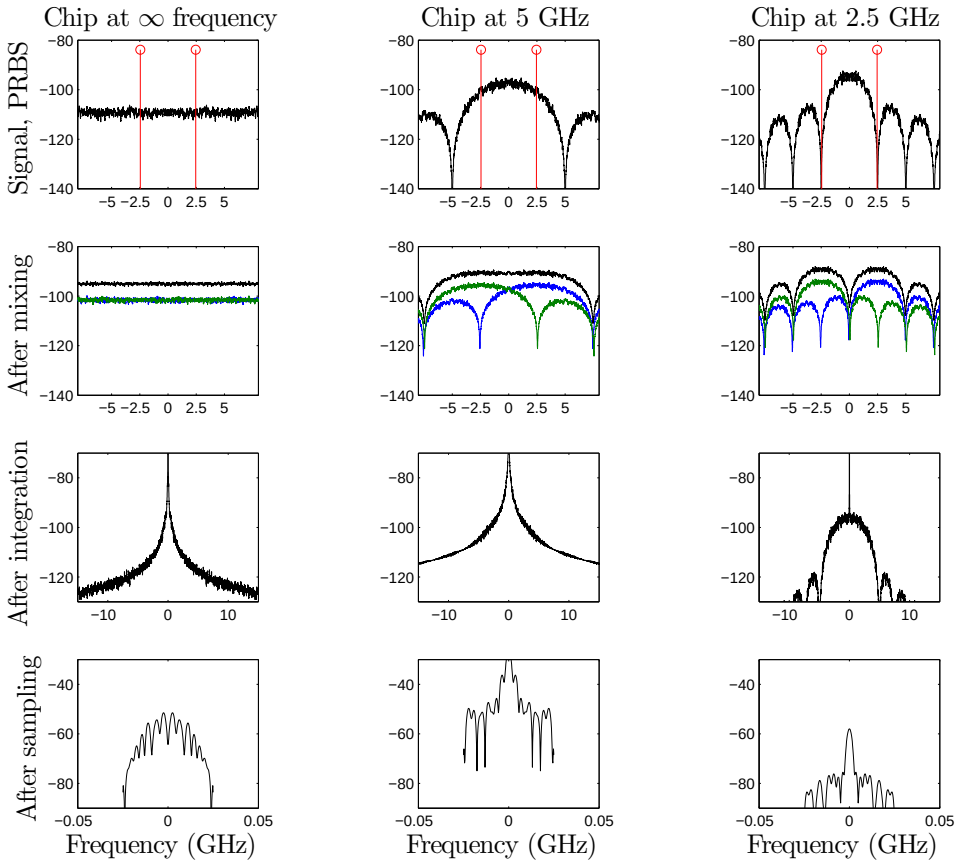


Figure 2.24: Effect of chipping rate. See also the companion Figure 2.25. Note that chipping at an infinite frequency is not actually desirable.

calibration procedure is much longer. In general, Φ has $N_{\text{min}} = \text{LCM}(N_{\text{chip}}, N_{\text{int}})$ columns. If $N_{\text{int}} = 100$, there is a very significant difference between $N_{\text{chip}} = 1000$ and $N_{\text{chip}} = 1001$. For the calibration reason, it is convenient to keep $\text{LCM}(N_{\text{chip}}, N_{\text{int}}) \lesssim 4000$. But as we will see below, it can also be quite bad for reconstruction behavior if N_{chip} is a low multiple of N_{int} or vice-versa.

2.5.2.1 Chip sequence rate

The PSD Ψ of the chip sequence is proportional to $\frac{\sin(\pi f/f_{\text{chip}})}{\pi f/f_{\text{chip}}}$, which is zero when $f = f_{\text{chip}}$. If an input has frequency f_{in}, the signal, after mixing with the PRBS, is just a convolution of the sinc with f_{in} and $-f_{\text{in}}$. Because the integration step will act as a low-pass filter, this means the output power is determined by the integral of Ψ in the regions around $-f_{\text{in}}$ and f_{in}. Thus, if $f_{\text{in}} = f_{\text{chip}}$, the output power will be very small.

Figure 2.24 shows this effect. Each column shows a different PRBS: the left column uses a PRBS that operates at such an extremely high frequency $f_{\text{chip}} \gg f_s$ that it has a flat spectrum, the middle column shows $f_{\text{chip}} = f_s$, and the right column shows $f_{\text{chip}} = f_s/2$. The red line in the top row represents an input tone at frequency $f_{\text{in}} \sim f_s/2$. The second row shows the result after

mixing with the PRBS (the blue and green lines show the convolution of the PRBS with f_{in} and $-f_{\text{in}}$, respectively). The third and fourth lines show the system after integration, which attenuates the high frequencies, and sampling, which folds everything into $[-f_{\text{ADC}}/2, f_{\text{ADC}}/2]$ (using $f_{\text{ADC}} = 50$ MHz). By examination of the bottom row (which has the same y-axis scale for all columns), it's clear that the power in the $f_{\text{chip}} = f_s$ variant is much greater than in the other cases.

Figure 2.25 shows the output power as a function of the input tone frequency f_{in}, averaging over many random trials. For $f_{\text{chip}} = f_s$, low frequency tones have only slightly higher energy than high frequency tones. For $f_{\text{chip}} = f_s/2$, the output power vanishes as f_{in} approaches $f_s/2$ since f_{in} falls into a zero of the sinc function. With the infinite frequency PRBS, the power is constant as a function of f_{in}. However, the absolute value of power is lower than in the $f_{\text{chip}} = f_s$ case, since the energy of the PRBS was spread out among all frequencies, whereas for $f_{\text{chip}} = f_s$ the energy is concentrated in the first lobe of the sinc function. Thus $f_{\text{chip}} \gg f_s$ is actually not desirable. The ideal PRBS would have a PSD that is flat inside $[-f_s, +f_s]$ and 0 elsewhere.

If we restrict ourselves to using a pseudo-random ± 1 sequence, then $f_{\text{chip}} = f_s$ is close to the optimal rate. If $f_{\text{chip}} \ll f_s$, then a low-frequency signal will generate large measurements and a high-frequency signal will generate tiny measurements, thus failing the principle of democracy. If $f_{\text{chip}} \gg f_s$, then too much PRBS power is out-of-band and wasted, and measurements of all signals will be too small and thus affected greatly by noise.

2.5.2.2 Chip sequence period

For hardware calibration and practical design reasons, the chip sequence repeats every N_{chip} terms, which is T_{chip} seconds where $T_{\text{chip}} = N_{\text{chip}}/f_{\text{chip}}$ (and from henceforth, we will fix $f_{\text{chip}} = f_s$). The PSD of the chip sequence has a discrete spectrum that is only nonzero at multiples of $1/T_{\text{chip}}$. The considerations here are exactly the same as in the previous section. If an input signal has carrier frequency f_{in}, then because the integration acts as a low-pass filter, the dominant contribution to the

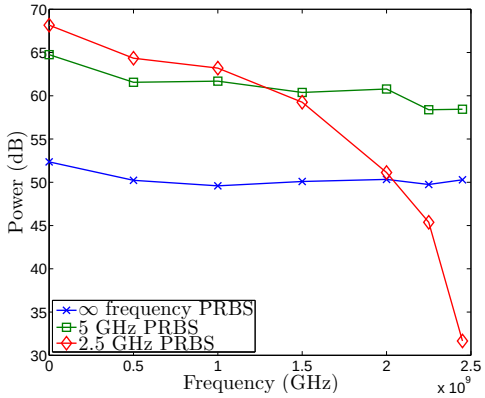


Figure 2.25: Using the same setup as Figure 2.24, and plotting power as a function of f_{in}

output comes from integrating Ψ in the regions near $-f_{\text{in}}$ and $+f_{\text{in}}$. If f_{in} is a multiple of $1/T_{\text{chip}}$, then $\Psi(f_{\text{in}})$ is nonzero. If f_{in} is not a multiple, then the attenuation effect of the the integrator H will be more significant.

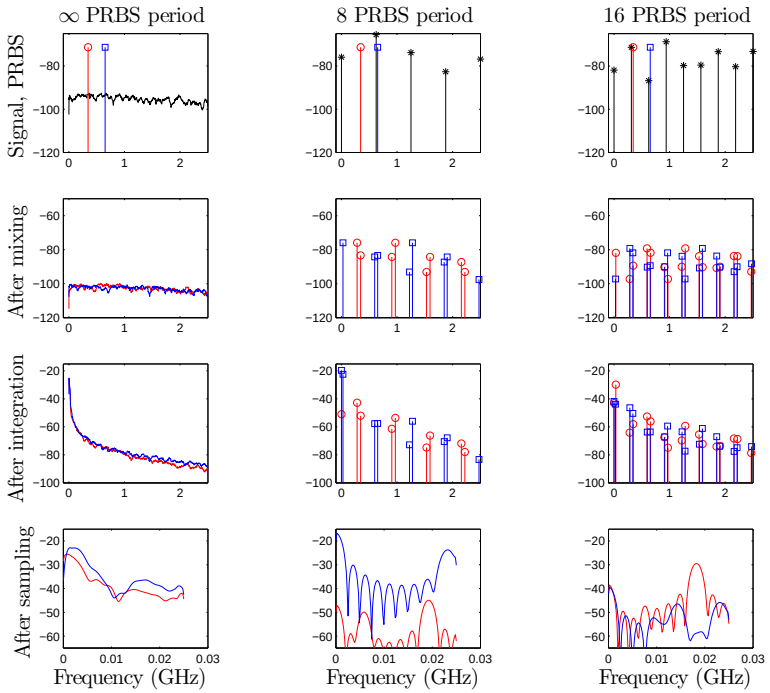


Figure 2.26: Depiction of problem associated with short chip periods N_{chip}. Using $N = 2048$. Shown are two possible inputs, one in red and one in blue. The left column shows an infinite-period chip sequence, while the middle and right columns show finite length chip sequences. The chip sequence with length $N_{\text{chip}} = 8$ works very well for signals (like the blue signal) with carrier frequencies that are lucky enough to be near one of its harmonics (harmonics spaced every $1/T_{\text{chip}} = 625$ MHz). For signals that are not near the carrier frequency, like the red signal, the power of the output is much less. For the left column, the energy of the ADC samples in the blue tone were .69 that of the red tone. For the rightmost column, the blue samples were 2.8 times larger in energy than the red samples. The center column shows the real disaster: the energy of the samples of the blue tone are 91 times larger that of the red tone. See Figure 2.29 for validation that this hurts reconstruction.

Figure 2.26 demonstrates this problem. Shown are three versions of the PRBS, with $N_{\text{chip}} = \{\infty, 8, 16\}$. In practice, we will consider $N_{\text{chip}} \simeq 100$, but we use small values of N_{chip} in the plot since they demonstrate the effect more clearly. The red and blue lines represent two input frequencies f_{in} and f'_{in}, chosen so that both of them lie on the $1/T_{\text{chip}}$ grid when $N_{\text{chip}} = 16$, but so that only f'_{in} lies on the $1/T_{\text{chip}}$ grid when $N_{\text{chip}} = 8$. After mixing, the power of both signals is the same. However, the integration severely reduces the power of the red signal, since its components are farther from DC.

This problem cannot be eliminated by changing the frequency response of the integrator, because we argued previously that we need the integrator to show decay: if it has a flat response, then it is really just sampling from a single point in time (i.e., $h(t) = \delta(t)$). However, the exact shape of H determines the how bad the problem is. The design of the integrator is considered in §2.6 independently of N_{chip}, but we hope to simultaneously analyze the two parameters in future work.

Recall that when choosing f_{chip}, we argued that $f_{\text{chip}} = \infty$ is not actually desirable (nor is it implementable). However, $N_{\text{chip}} = \infty$ *is* desirable from a theoretical perspective, as Figure 2.26 suggests. Unfortunately this would make calibration impossible, so we do not consider this option. The Caltech design uses a shift register which consumes power proportional to N_{chip}. The Northrop Grumman design is based on LFSR which generates $N_{\text{chip}} = 2^r - 1$ where r is the number of taps, and consumes power approximately proportional to r. Thus the LFSR variant might be better for future work since it allows large N_{chip}, although this would require more work for calibration.

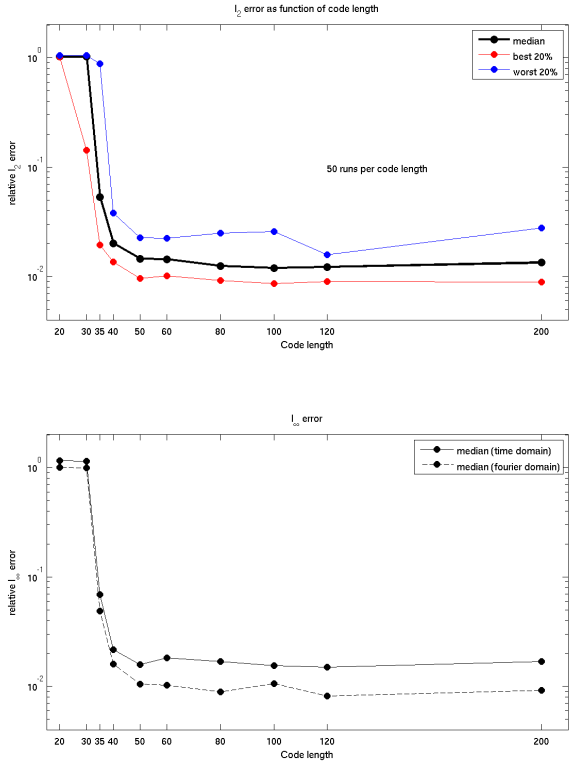


Figure 2.27: Initial codelength N_{chip} tests from 2008. We chose 128 to be safe.

Other than just the length of N_{chip}, the relation of N_{chip} to the integration period N_{int} is important. This does not affect the power, but rather the identifiability. If N_{chip} is a multiple of N_{int}, then after sampling, the result will be a discrete spectrum, and it may be possible for two distinct signals with carrier frequencies f_{in} and $f'_{\text{in}} = f_{\text{in}} + f_{\text{ADC}}$ to generate the same measurements. In general, if N_{chip} and N_{int} have a large GCD, then the outputs of the RMPI are more similar to each other, so the system is less robust.

To verify all of the above reasoning, we present results from several numerical tests. Figure 2.27 shows initial tests from August 2008 that were used to inform the initial version 1 design. This test and all other tests were using $N_{\text{int}} = 100$ (i.e., $f_{\text{ADC}} = 50$ MHz) unless otherwise specified. The

results from this first test suggested that performance was catastrophically bad if $N_{\text{chip}} < 40$, and inspired the choice $N_{\text{chip}} = 128$ in order to be a safe margin away from this failure point.

Figure 2.28 shows the spectrum of a single multi-tone input, and how it aligns with the PRBS grid. Using a realistic model of the system and full ℓ_1 reconstruction, we found that the tones near $1/T_{\text{chip}}$ gridpoints were recoverable, while those far from the gridpoints were not recoverable.

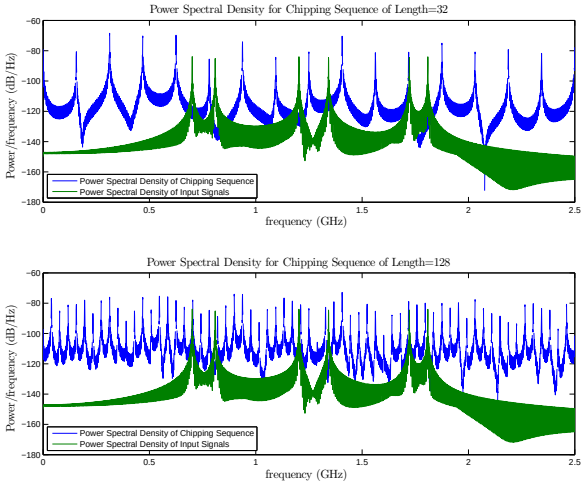


Figure 2.28: The PSD of chipping sequences (blue) and pure tone inputs (green). For a fixed integrator bandwidth B, with PRBS of length $N_{\text{chip}} = 32$ (top plot), the tones at 700, 1200, and 1800 MHz did not reconstruct successfully because they have large frequency separation from the nearest PRBS frequency $1/T_{\text{chip}} = 156.25$ MHz. Tones at 800 and 1750 MHz, close to a PRBS frequency, were recoverable. In the bottom plot with a 128 bit sequence, the maximum frequency separation was far less ($1/T_{\text{chip}} = 39.06$ MHz), and reconstruction was successful for all tones.

Next, we present evidence from some matched filter reconstructions, using a similar testing procedure as that described in the chip rate section. Because of the speed of reconstruction, we are able to generate meaningful statistics, and also have the benefit that the matched filter reconstruction relies on no parameters. The data were collected using an ideal Bernoulli matrix Φ, so integration was approximated by a discrete sum; for the sake of PRBS length, this approximation is most likely not significant.

Figure 2.29 shows the error of the frequency estimate as a function of $(f_{\text{in}} - 1/T_{\text{chip}}) \mod 1/T_{\text{chip}}$. A value of 0 on the x-axis indicates that the input frequency f_{in} was exactly on a grid point. The different plots show various values of N_{chip}. Each point represents the result of one independent test. Noise was included in the measurements; these data are a subset of the same data plotted differently in Figure 2.30. In Figure 2.29, we see a clear dependence between low error and $f_{\text{in}} \simeq 1/T_{\text{chip}} \mod 1/T_{\text{chip}}$, which confirms the reasoning above.

Figure 2.30 plots frequency error (and empirical frequency recovery "success" rates) as a function of of N_{chip} for various amounts of additive noise. As expected, recovery is more likely for large N_{chip}, but the effect is small after about $N_{\text{chip}} = 100$, so the choice of $N_{\text{chip}} = 128$ is about optimal. We also include results of the same simulations but using a matrix with staggered rows, and this improves recovery, especially for small N_{chip}. The simulations show periods N_{chip} of various lengths, including

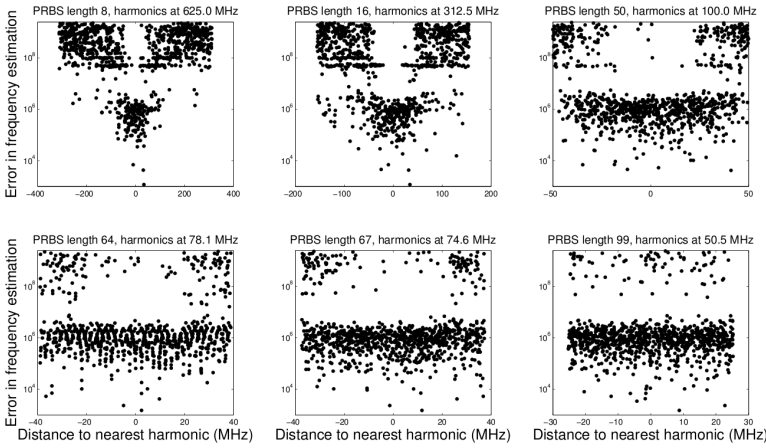


Figure 2.29: Explains why we want long N_{chip}. Error increases the farther we are from a harmonic. See Figure 2.26 for a cartoon depiction. See also Figure 2.30.

some chosen to be very close to each other but with different properties related to $N_{\text{int}} = 100$; for example, we included $N_{\text{chip}} = \{99, 100, 101\}$. The plots show very clearly that the behavior is different for even these small changes, but this behavior is not consistent over all noise levels and we find the results inconclusive and in need of further study.

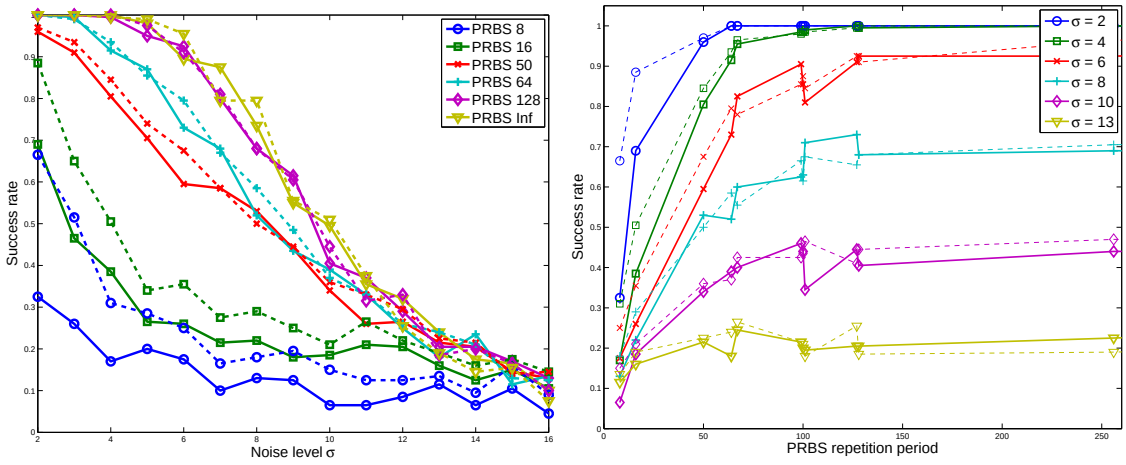


Figure 2.30: Tested PRBS sequences of period $N_{\text{chip}} = \{8, 16, 50, 64, 67, 99, 100, 101, 127, 128, 256, 512, \infty\}$ (∞ refers to length N, which was $N = 1024$ for this test). Other simulations parameters were the same as in Figure 2.19. Dashed lines for a matrix with staggered channels; this typically improves results, but not by too much. The seemingly bizarre choices of PRBS lengths were chosen because it is not only relevant what their length is, but also the least common multiple of the PRBS length N_{chip} and the ADC rate N_{int}. The ADC samples every $N_{\text{chip}} = 100$ Nyquist samples.

2.5.2.3 Case study: test of NG chip sequence

In the Caltech RMPI, the PRBS for each channel is loaded from a 128-bit shift register, so this can be programmed to an arbitrary sequence. However, this adds significant power cost to the design. Since this is the first working RMPI chip, it was considered worth the sacrifice in order to allow the extra flexibility: for example, it is possible to implement the run-length-limited ideas.

The Northrop Grumman chip saves power by avoiding shift registers, and instead generates the PRBS sequence from a LFSR. Initially, the direct outputs of a LFSRs were used, but after this author's analysis in January 2009, it was identified that this caused significant performance degradation, partially due to the occasional very long runs of $+1$ or -1. For a system with many channels, this is not an issue, but the NG chip only has 4 channels. In addition, the sequences sent to each channel were correlated, which added to the problems. Offsetting (aka staggering) the channels helps the situation, but not enough to match the performance of the Caltech RMPI.

Following the analysis, the PRBS was modified so that it was generated by a Gold code, which can be created by mixing the outputs of LFSR, so the additional hardware complication was minimal. The output of each Gold code repeated every 63 periods, but only the first 52 periods were used. Thus each of the four channels used a PRBS with length $N_{\text{chip}} = 52$. Because the NG design uses four channels, the ADC rate is greater than in the Caltech design, and operates just below 100 MHz, so that $N_{\text{int}} = 52$. Thus $N_{\text{chip}} = N_{\text{int}}$.

This design underperformed, so we analyzed the system. An empirical test was conducted as follows, which consisted of 100 independent trials. For each trial, a carrier frequency was picked uniformly at random from between 300 MHz and 2.5 GHz, which was modulated by a simple trapezoidal pulse that spanned almost all the signal; the Nyquist length was $N = 1024$ so this was about 200 ns. To be realistic, synthetic noise was added both to the signal and to the measurements, calculated to give 60 dB SNR separately.

Reconstruction was done via two methods. The first method used the over-complete Gabor dictionary with about 8 reweightings using the NESTA algorithm. The second method was more complicated, and consisted of first reconstructing the signal with a DCT orthobasis and reweighting a few times, followed by using the over-complete Gabor dictionary with reweightings. The second method almost always worked better. These techniques will be discussed in §2.7.

Reconstructions were declared either a success or a failure. To fail, two criteria had to be met (in practice, criterion 1 was almost never met without criterion 2 also being met). Both criteria were based on the frequency only, using Welch's method to estimate the PSD. The criteria were

1. Failure Criterion 1: the peak of the spectrum was not close to the actual carrier frequency. The allowable tolerance was set to the the width of the main lobe at -20 dB from the peak.

2. Failure Criterion 2: the side-lobes were too large. Side-lobes were required to be 20 dB lower

Design	Failure rates per 100 tests Method 1	Method 2
$N_{\text{chip}} = 52$, stagger 1	46	33
$N_{\text{chip}} = 52$, stagger 2	44	31
$N_{\text{chip}} = 52$, stagger 10	48	35
$N_{\text{chip}} = 52$, seed 1	57	43
$N_{\text{chip}} = 52$, seed 2	54	39
$N_{\text{chip}} = 52$, seed 3	50	35
$N_{\text{chip}} = 51$, seed 1	25	13
$N_{\text{chip}} = 104$, seed 1	13	7
$N_{\text{chip}} = 104$, seed 2	20	7
$N_{\text{chip}} = 104$, seed 3	20	9
$N_{\text{chip}} = 3276$	0	0
$N_{\text{chip}} = 312$, random signed Bernoulli	0	0
Entire channel a Gaussian matrix	0	0
Caltech design, $N_{\text{chip}} = 128$	0	0

Figure 2.31: June 2009 tests of the Northrop Grumman chipping sequence. The first half of the table show various NG designs; $N_{\text{int}} = 52$. When $N_{\text{chip}} = N_{\text{int}}$ or N_{chip} is short, the performance suffers. The NG design was modified to use $N_{\text{chip}} = 3276$ which results in much better performance. The second half of the table show some other models for comparison; the Caltech design seems to work well.

than the main peak.

By inspection, most signals that passed these frequency criteria also looked good in the time domain. The results are presented in Table 2.31. The LFSR depends on an initial state, referred to as a "seed," and the test was run using various seeds in order to determine if the poor performance was due to an unlucky choice of the seed.

This alignment of $N_{\text{chip}} = N_{\text{int}}$ proved to be disastrous, and shifting the channels relative to one-another did nothing to improve the results, nor did changing the seeds of the Gold codes. These results are reported in the first 6 rows of the table.

Initially attempts tried two fixes that yielded only modest improvement. The first fix is to take 51 samples of the Gold code to use for the periodic PRBS, with the idea that the mis-alignment $N_{\text{chip}} \neq N_{\text{int}}$ will improve recovery. This appears to have helped (see Table 2.31). A similar modification is to take 104 samples from the Gold code (which means the last 41 sample are duplicates of the first 41), and this also helps. However, neither modification matches the performance of the Caltech design or other baseline tests shown at the bottom of the table.

In light of these results that show poor performance compared to the Caltech Φ matrix, the NG team found a way to devise a PRBS of length $N_{\text{chip}} = 52 \times 63 = 3276$ by adjusting the LFSR inputs to the Gold code. According to the above criteria, this adjustment is extremely beneficial and results in no errors per 100 tests.

Note that in order to fully characterize the system, Φ must have 3276 rows. For the Caltech system, $N_{\text{chip}} = 128$ and $N_{\text{int}} = 100$, so their LCM is 3200; thus Φ has 3200 rows, so calibration of the Caltech and NG systems are of about the same difficulty.

2.6 Integration

Integration can be thought of in the time domain, or as a filter in the frequency domain, with a $H(s) = 1/s$ transfer function. If we think of integration as a filter, then we quickly realize that $H(s) = 1/s$ is not realizable, and so we will use approximate-integrators. The common approximations are that the integrator is either a single-pole system or a double-pole system. For any of these cases, we can write down the exact formula that represents (non-ideal) integration, as described in the modeling section. This section concerns the constraints on the poles of the integrator.

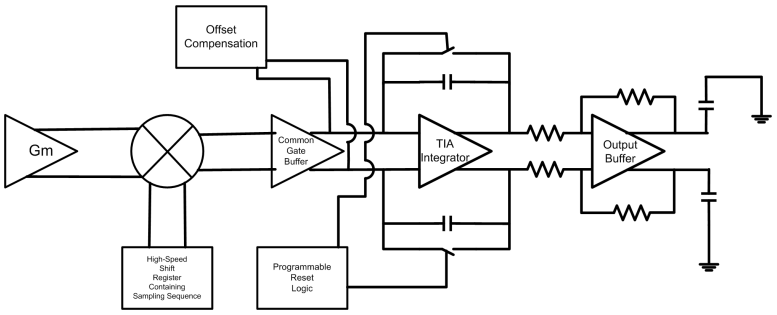


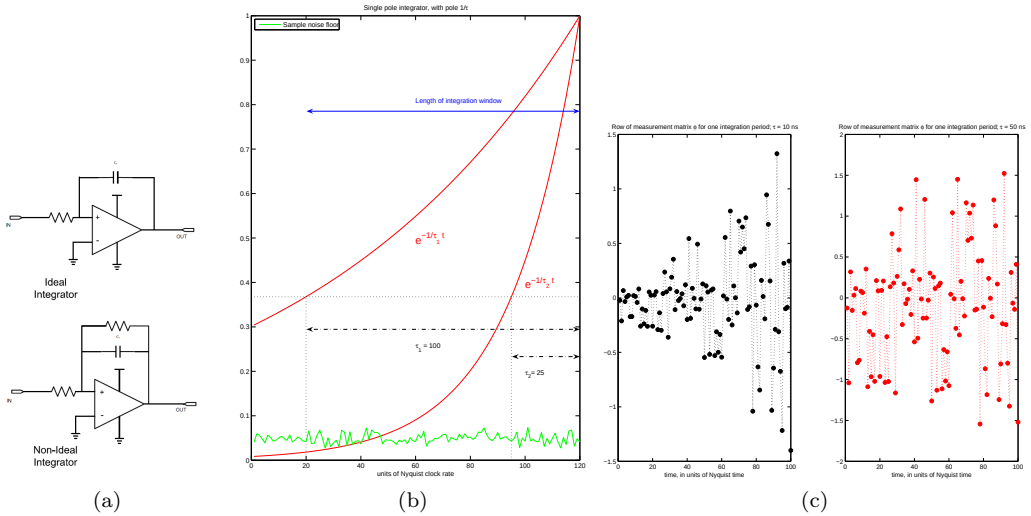
Figure 2.32: A more detailed diagram of the blocks in a single channel (after the LNA). See also Figures 2.11 and 2.12.

At low frequencies, the system is dominated by the smallest pole, also called the "inner pole". The transfer function of a single-pole system, with pole a, is $H(s) = 1/(s + a)$, and the impulse response is $h(t) = e^{-at}$. If the pole is large, then h decays quickly in time. See Figure 2.33 for a visualization.

2.6.1 General constraints

The ideal RMPI uses an exact integrator, so $H(s) = 1/s$. A first-order low-pass filter has attenuation of 20 dB per decade, which means exactly that the attenuation at high frequencies obeys $1/s$, so an integrator is quite similar to a low-pass filter at high frequencies. In papers such as [TLD+10], it is mentioned that an exact integrator is not necessary, and any similar low-pass filter works: "It suffices to perform low-pass filtering before the samples are taken." The actual story is a bit complicated, since the performance depends heavily on the bandwidth of the low-pass filter. We present results from a preliminary Northrop Grumman design that tried to approximate a band-pass filter (see Figure 2.34), and see that it has bad performance. Typical results are shown in Figure 2.35.

Consider an ideal band-pass filter with large bandwidth, such as from $[-f_s/2, f_s/2]$. That is, $H(s) = 1, |s| \leq f_s/2$ (or any constant) and $H(s) = 0$ otherwise. We can show how this system will perform badly. Consider two inputs x_1 and x_2 with frequencies f_1 and f_2. After mixing with the chip sequence $c(t)$, the spectrum of the signal is just a convolution of $\hat{c}(f)$ with f_1 or f_2. Since H has a flat response, the Fourier transform of the first signal $\mathcal{F}(x_1 c)$ and the second signal $\mathcal{F}(x_2 c)$



(a) (b) (c)

Figure 2.33: Plot (a) a block diagram of a realistic integrator, showing the parasitic resistance. It is impossible to make an integrator without a pole a. Plot (b) shows $h(T_{\text{int}} - t)$ where h is the impulse response $h(t) = e^{t/\tau}$ and $\tau = 1/a$ is the time constant. If the time constant is too small, then the correlation with the signal only occurs for the time right before the sampling, which is bad for sampling radar pulses. See also Figure 2.8. The right plot (c) shows the rows of a Φ matrix from the single pole model, with $\tau = 10$ ns (left) and $\tau = 50$ ns (right). The version 1 RMPI design has $\tau = 22$ ns, and the version 2 RMPI design has $\tau = 3333$ ns.

only differ in the region $[f_s/2 - \Delta f, f_s/2 + \Delta f]$ (and the negative frequency counterpart) where $\Delta f = |f_1 - f_2|$. If $f_1 \simeq f_2$, then the difference in the measurements of the system are determined by integrating the integral of the chip sequence Fourier transform over $[f_s/2 - \Delta f, f_s/2 + \Delta f]$. So for similar frequencies, the measurements will not be significantly distinct, and therefore reconstruction by any method is not stable.

This agrees with the time-domain intuition. The larger the flat response of the filter $H(s)$, the more similar the system is to being a delta function in time. As we've argued, a delta function in time is not good for compressed measurements since it is reasonably coherent with radar pulses.

2.6.1.1 Northrop Grumman integrator design

To summarize the above argument, if $H(s) = 1$, then $H(if_1)$ and $H(if_2)$ are indistinguishable. If $H(s)$ is a bandpass filter with large band, then $H(if_1)$ and $H(if_2)$ are distinguishable, but not robustly so. A much better frequency response is $H(s) = 1/s$ since then $H(if_1)$ and $H(if_2)$ are distinct for even small shifts $|f_2 - f_1|$.

Shifting. If the integrator has a large pole, then $h(t)$ is narrowly concentrated in time. See Figure 2.33 for a depiction. The first measurement from the system is $\int_0^{T_{\text{int}}} x(t)c(t)h(T_{\text{int}} - t)dt$, so this heavily weights $x(T_{\text{int}})$ and earlier, and ignores $x(0)$. Subsequent measurements will weight regions just before $x(nT_{\text{int}})$. Thus parts of the signal x are almost completely ignored by the measurements. An obvious fix is to shift/stagger each channel by a unique amount. For example,

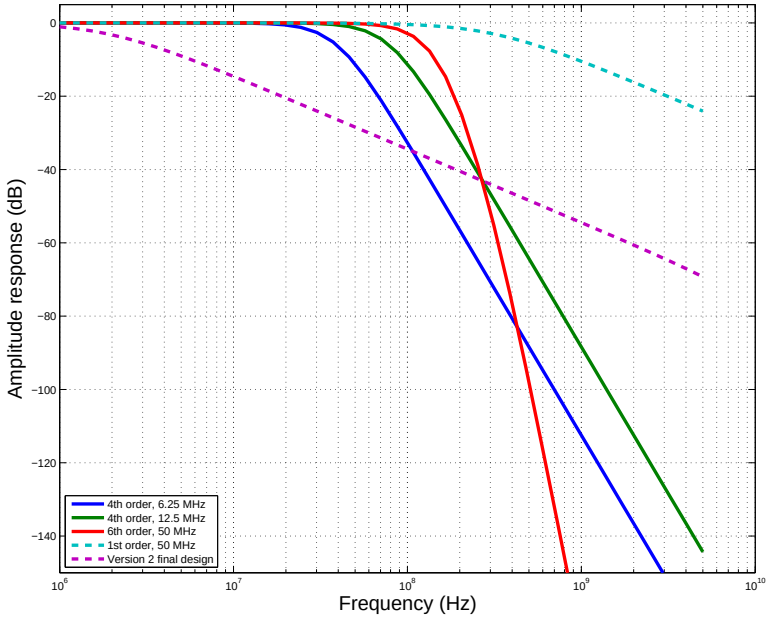


Figure 2.34: Bode plots of preliminary NG integrator designs (solid lines), proposed modification (50 MHz single pole), and the Caltech ver. 2 final design (302 KHz pole; see Table 2.4). The integration filter should have its first pole as close to DC as possible so there is a "memory" effect. These NG designs performed poorly (see Figure 2.35) because they are too similar to an ideal low-pass filter.

channel one integrates from $[0, T_{\text{int}}]$, channel two integrates from $[T_{\text{int}}/8, 9/8T_{\text{int}}]$, etc.

Consider the preliminary Northrop Grumman designs shown in Figure 2.34. We also include a slightly improved version with its lowest pole at 50 MHz pole. The top of Figure 2.35 shows that even for the improved 50 MHz version, we fail to reconstruct a pulse; using a Bernoulli matrix, or using the Caltech design, the same pulse was reconstructed, so this is an indication that the integrator is underperforming. The bottom of the plot shows the reconstruction when using the same design but with shifted channels. Performance is much better. To systematically test the effect of shifting, we use the single-pole model, and vary the time constant $\tau = 1/a$ and vary the amount of shifting. We use $N_{\text{int}} = 100$, and consider offsetting each channel from one another by $0, 12, 25$, or 50. Since there are 8 channels, the offset of 12 per channel makes the most sense, since then no two channels align. Figure 2.37 shows the results. For large time constants, the error is small regardless of shifting. For a time constant of $12.5\Delta T$, the error is very large (relative error is about 1) without using shifts. Using any of the nonzero shifts, the relative error for this time constant goes down to between .02 and .1. Interestingly, the exact amount of shift does not seem to make a big difference. Because of these results, we highly recommend staggering the channels. The only reason not to stagger channels is for ease of calibration. The current version 2 design can be programmed to shift by an arbitrary amount.

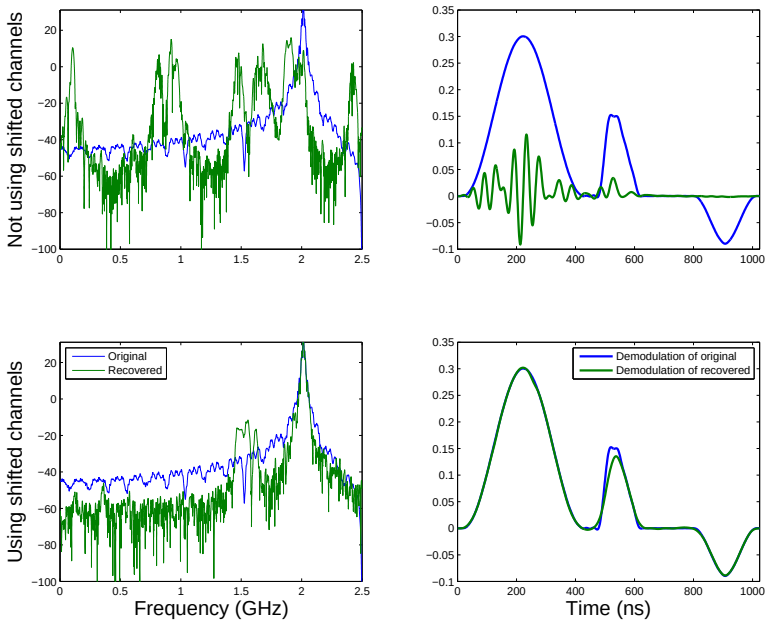


Figure 2.35: Simulation using a preliminary Northrop Grumman design, using first-order filter 50 MHz pole and 4 channel at 100 MHz each (see Figure 2.34). Some of the initial NG designs were too similar to band-pass filters, rather than integrators, and lacked the memory necessary to encode pulses. This was the best model tested (February 2009). The top row is with all 4 channels aligned; the bottom row has shifted/offset each channel by a quarter of the integration period. This shifting is extremely helpful.

2.6.1.2 Multipole systems

From a circuit design point-of-view, creating a single-pole integrator with a very small pole a is difficult. In version 1 of the RMPI chip, the lowest achievable pole was about 44 MHz. If we allow a multi-pole system, then it is much easier to push the first pole closer to 0. By doing this, the version 2 RMPI chip has its first pole at 302 kHz, and a second pole at about 300 MHz. With such a small first pole, the impulse response $h(t)$ is very flat over the integration time $[0, T_{\text{int}}]$. Figure 2.38 shows matched filter tests of an exact 300 kHz integrator, and compares recovery to that of a Bernoulli ± 1 matrix. The results are averaged over 400 samples, and the matched filter requires no parameters, so this test is quite reliable. The single-pole model without stagger suffers slightly at high noise level, but not by a significant amount, since the time constant $1/(300\text{ kHz})$ is very long. This test adds noise of the form $\Phi(x) + z$, instead of $\Phi(x + z)$, since for the latter model there is no discernible difference in performance. The variance of z was adjusted to account for the gain of the system. With the former noise model, the additive noise z will make it slightly more difficult to recover x at the beginning of time periods. We conclude that the 300 kHz pole has little effect on the system, and is much better than the 44 MHz pole in the version 1 design.

But does the second pole near 300 MHz have an effect? Figure 2.39 shows several tests on different types of integrators. The left column shows a version of Φ generated from a Simulink model that incorporated a realistic version of the integrator, with several zeros and poles. It performs similarly

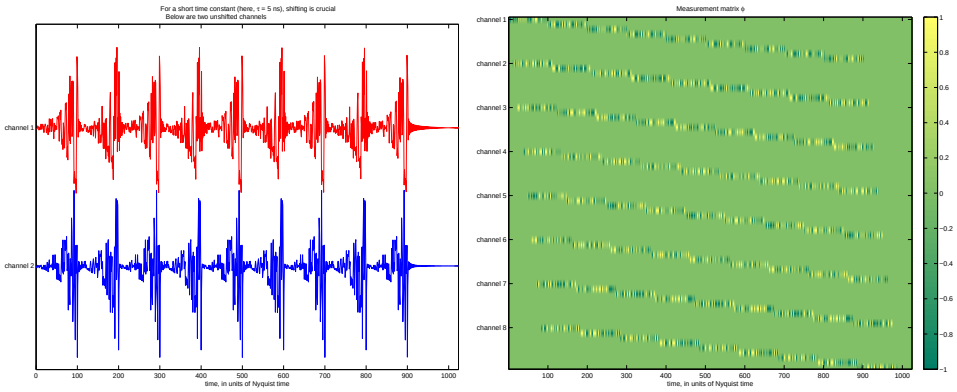


Figure 2.36: Figure (a) shows copies of $h(nT_{\text{int}} - t)$ for $n = 1, \ldots, 9$ overlaid, for $h(t) = e^{-at}$ where a is large. The top data (in red) are for channel one, and the bottom data (in blue) are for channel two. We see that without shifts, these align. For times $t = [0, 20\Delta T]$, $x(t)$ is barely measured. This can be fixed by staggering the channels. Figures (b) shows the matrix Φ of a staggered system.

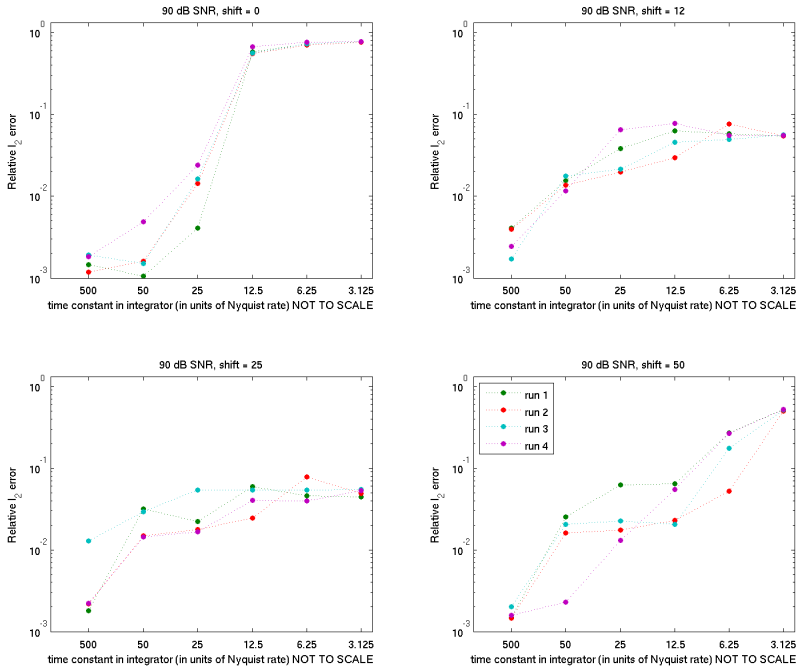


Figure 2.37: The results of four independent reconstructions using a single-pole model. The four plots show four different values of the channel stagger. In the 0 offset model, all channels aligned; in the 12 offset model, none of the channels aligned; in the 25 and 50 offset models, 2 and 4 channels aligned, respectively. For small values of the time constant, shifting by any amount is helpful.

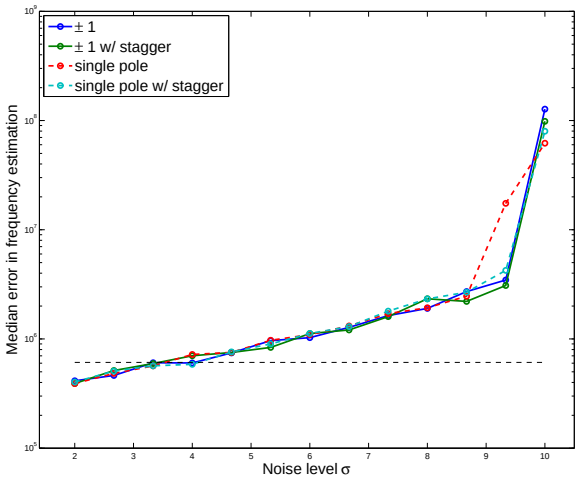


Figure 2.38: The ±1 model compared to the single-pole model (with 300 KHz pole). This compares performance, not modeling error. The single-pole model suffers a bit, due to the decay of the time domain transfer function $h(t)$. Shifting the channels helps this. $N = 2048$, and simulations used 2 realizations of each Φ matrix and then 200 sample noise vectors and carrier frequencies; x was a smooth pulse of length 100 ns. The dashed horizontal line shows the frequency resolution of the grid search.

to the other Φ models, so it seems that it has little effect.

Let the second pole be at location a_2. The transfer function of a two-pole system is

$$H(s) = \frac{1}{(s+a)(s+a_2)}$$

so that $H(s)$ decays like $1/s^2$ for frequencies much larger than a_2. To reason about the effects of the second pole, we simplify and consider

$$H(s) = \begin{cases} \frac{1}{a_2(s+a)} & s < a_2 \\ 0 & s \geq a_2 \end{cases}.$$

If a_2 is in the range 100 MHz to 1 GHz, this is quite similar to the scaled single-pole integrator $H(s) = \frac{1}{a_2(s+a)}$. The main limit is that if the PRBS has a short repetition period T_{chip}, then we must have $a_2 \gg 1/T_{\text{chip}}$, otherwise signals with frequency that is far from a harmonic of $1/T_{\text{chip}}$ will generate very small measurements. For the current chip design, $T_{\text{chip}} = 128\Delta T$, so $1/T_{\text{chip}} =$ 39 MHz, and hence a practical lower-limit of a_2 is about 100 MHz, since otherwise all but one or two of the spikes in the spectrum of $x(t)c(t)$ will extremely attenuated. With only one or two of Fourier series of c in the passband, it is much more likely for two inputs to generate similar measurements, and hence the system is less robust. The small bandwidth would also exacerbate the power difference between inputs that are exact harmonics of $1/T_{\text{chip}}$ and those that aren't.

To summarize the findings, we can view the integrator as a low-pass filter in the sense that we do not need frequencies above about 100 MHz, but the passband behavior of the filter must not be

Model	Zero	First pole	Second pole	Third pole
NG InP	NA	12 MHz	NA	NA
Caltech ver. 1	NA	44 MHz	NA	NA
Caltech ver. 2	360 MHz	302 KHz	361 MHz	1.696 GHz

Table 2.4: Poles and zeros of the integrating filter for various design. Poles are calculated at 3 dB cutoff from magnitude response. See also Figure 2.40.

flat! The exact poles and zeros of the RMPI are shown in Table 2.4, and the extracted frequency response of the integrator, taken from SPICE simulations, is shown in Figure 2.40.

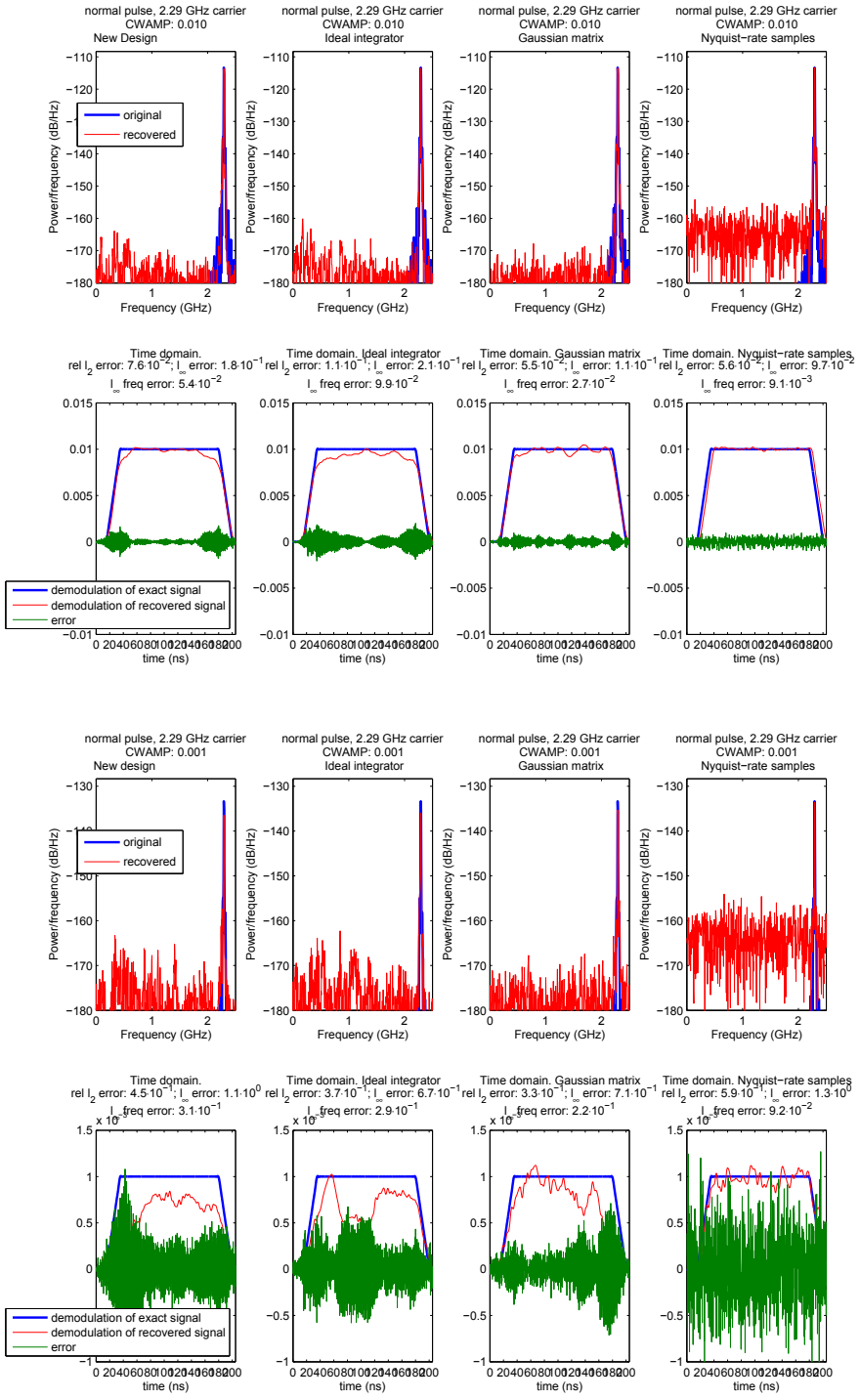


Figure 2.39: Does the new integrator suffer compared to, say, a Gaussian matrix? Not much. This test used a pessimistic noise level, and then ran three different amplitudes. For each test, 4 types of Φ matrix were used: a realistic version from the calibration, a single-pole model, a matrix with iid $\mathcal{N}(0,1)$ entries, and a Nyquist sampled version ($\Phi = I$). There is no significant difference. Only the two smallest signals are shown, since the large amplitude signal had visually perfect reconstruction.

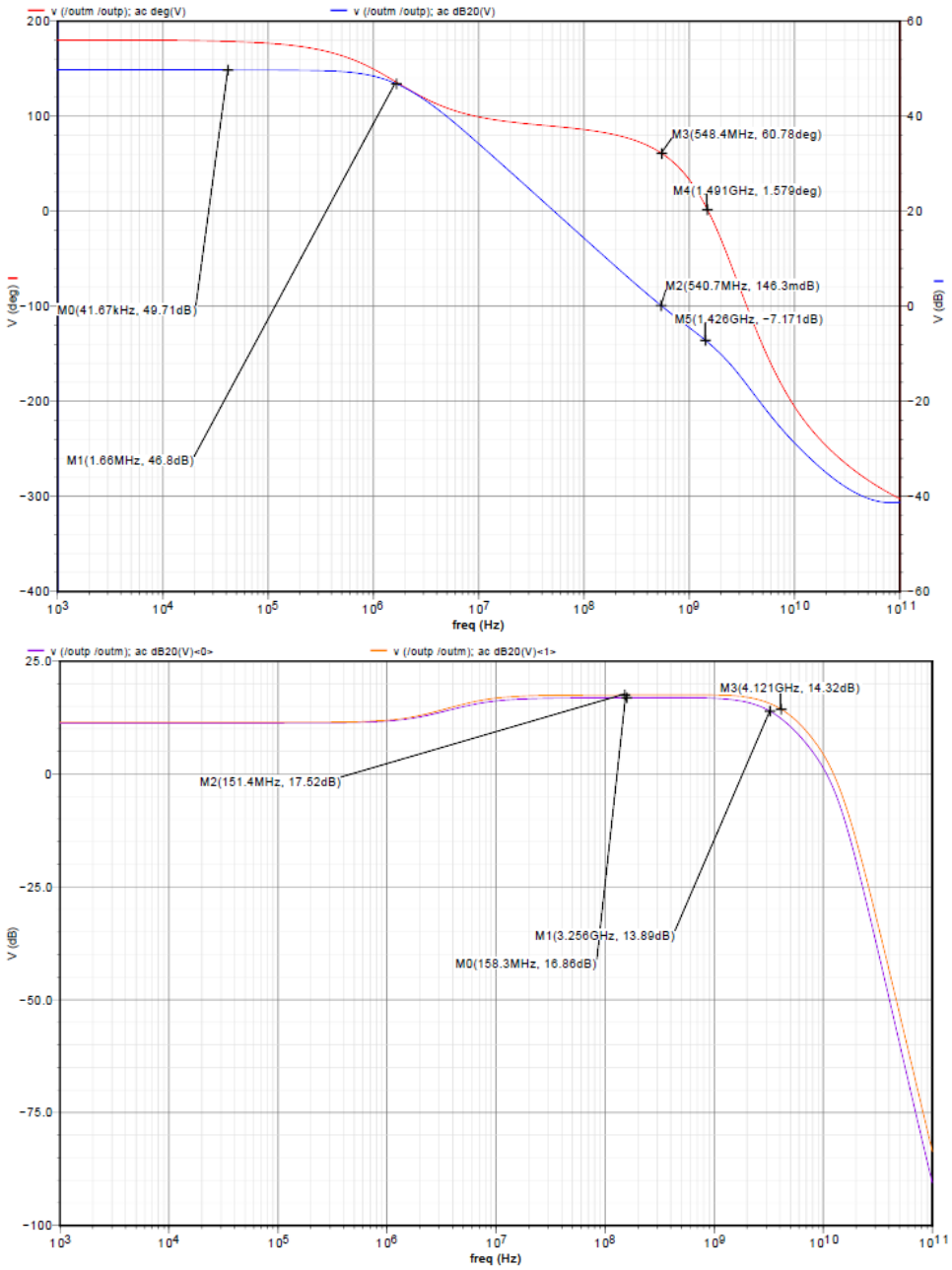


Figure 2.40: Top: integrator frequency response. Bottom: LNA frequency response. Blue curves are gain (scale on right axis), red curves are frequency response (scale on left axis). Results are from SPICE-extracted simulations. See Table 2.4 for values of poles and zeros. Other major blocks were modeled as well, and the resulting design parameters used to inform the Simulink model.

2.7 Recovery

This section covers the reconstruction of a pulse from its samples, assuming that a linear model Φ of the system is available. One of the key computations is solving an ℓ_1 minimization problem. This was originally solved using a version of l1Magic [CR07b] modified to allow the analysis formulation. Subsequently, we developed the algorithm NESTA (see Chapter 3) which is much faster. Detailed considerations of ℓ_1 solvers are discussed in Chapter 3 and 4. This discussions below assumes the existence of a ℓ_1 solver.

To give an idea of the computational time involved, a very high accuracy solve for $N = 1024$ with NESTA as the ℓ_1 solver, using several reweighting steps, takes 39 seconds on an i7 quad-core computer at 1.6 GHz. For lower accuracy computations, recovery time is closer to 10 seconds. Of the 39 seconds, roughly 65% of the time is spent computing the Gabor frame code. This code spends about half its time with $\mathcal{O}(N)$ re-arrangements, and the other half with $\mathcal{O}(N \log N)$ FFT calls. Most of this code is multi-threaded, and a reasonable estimate is that it would take 150 seconds to run the entire recovery on a single core (when desiring extremely high accuracy). This same i7 machine reaches about 1.8 Gflop/s per core on the FFT using the LINPACK Benchmark [DBMS84], so the computation requires roughly 270 Gflop. A medium-precision reconstruction requires about 100 Gflop. As of 2011, good GPU cards run about 500 to 1000 Gflop/s, so even with a special GPU implementation, and assuming no major overhead losses, the computation takes .1 to .2 seconds. Since $N = 1024$ represents $T = 204$ ns, this is a factor of at least $5 \cdot 10^5$ away from real-time recovery. More work is needed to bring this down to a reasonable level. Matched filtering will likely be key. One might imagine a hybrid system that uses crude methods for pulse detection and high-accuracy ℓ_1 solves only when a pulse is detected.

Practical aspects of signal recovery for RMPI have not been discussed in a single paper, but the ideas presented here draw upon many sources. Some ideas were discovered independently of known results. For the matched filter, see related ideas in [DDW$^+$07, DB10], as well as the recent paper [ERW11]. There is also recent unpublished work by Michael Wakin on using a notched filter to project out interfering signals; this relies on a result from [DW10].

2.7.1 Matched filter

An alternative to ℓ_1 minimization is the compressive matched filter, which is discussed in this section. If there is much prior knowledge about the signal—for example, the exact shape of the pulse envelope is known—then the matched filter is a very powerful technique, for in this case it is robust, fast, and optimal. In general, the pulse envelope is not known well, and in this case ℓ_1-based recovery methods are preferred. For the purposes of testing the system, the matched filter is used since it represents the best recovery possible, and so exposes the limits of the system. Because it is fast, it

allows hundreds of simulations in a short amount of time.

Consider our discrete model $b = \Phi x_0 + z$. Assume x belongs to some parameterized set $\mathcal{X}$; for example, $\mathcal{X}$ could be the set of Fourier modes on a grid of length N. We will actually allow the set $\{ax : x \in \mathcal{X}, a \in \mathbb{C}\}$ where c contains amplitude and phase information. Both x and c may be complex, but we assume all other variables are real.

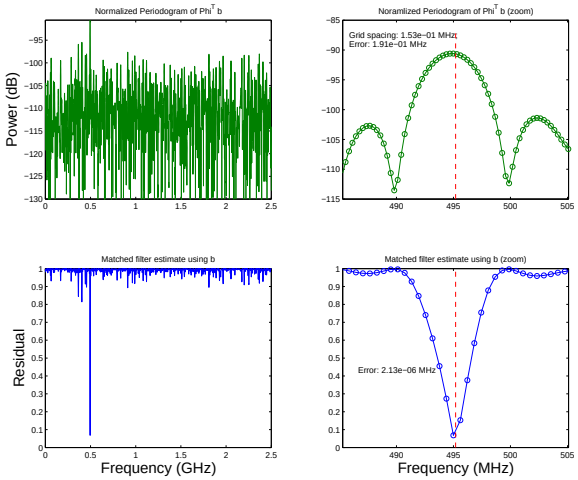


Figure 2.41: Matched filter estimation of an off-grid pure tone. The periodogram estimates (top row) are biased, even after accounting for the norms of the FFT of Φ.

Theorem 2.7.1 (Matched filter in "compressed domain")**.** *If $z \sim \mathcal{N}(0, \Sigma)$ and $a_0 x_0 \in \mathcal{X}$, then the maximum likelihood estimator x_{MLE} is given by the matched filter estimate*

$$x_{MF} = \underset{x \in \mathcal{X}, a \in \mathbb{C}}{\operatorname{argmin}} \|b - \Phi(ax)\|^2_{\Sigma^{-1}}.$$

Proof. Trivial, since the log-likelihood function is

$$\ell(\hat{x}) \equiv \log \mathbb{P}(b | x = \hat{x}) = \log \mathbb{P}(z = b - \Phi\hat{x}) = \text{const} - \langle b - \Phi\hat{x}, \Sigma^{-1}(b - \Phi\hat{x}) \rangle.$$

□

We'll specialize this to the following parametric model. Let x be a real-valued tone which we observe for $T = N/f_s$ seconds, with arbitrary phase and frequency (the frequency can be off-grid, but still within $[0, f_s/2]$), and windowed by a *known* window w (with $W = \text{diag}(w)$). Let $f_\gamma(n) = e^{i\omega_\gamma n/f_s}$ be a complex exponential and $F_\gamma = [f_\gamma, \bar{f}_\gamma]$ be a $N \times 2$ matrix. For the moment, let γ be fixed. The reason we use f_γ and its conjugate (i.e., the negative frequency) $\bar{f}_\gamma$ is that the measurements b are real valued. This approach is equivalent to minimizing $\|b - \text{Re}(\Phi W a f_\gamma)\|$ over $a \in \mathbb{C}$.

We can solve for the coefficient $a \in \mathbb{C}^2$ easily (c has two entries, but it will turn out to be

conjugate symmetric, so it only has one complex-valued degree-of-freedom):

$$a_\gamma \equiv \underset{a\in\mathbb{C}^2}{\operatorname{argmin}} \|b - \Phi W F_\gamma a\|^2_{\Sigma^{-1}} = (F_\gamma^* W \Phi^T \Sigma^{-1} \Phi W F_\gamma)^{-1} F_\gamma^* W \Phi^T \Sigma^{-1} b, \tag{2.7.1}$$

$$\min_{a\in\mathbb{C}^2} \|b - \Phi W F_\gamma a\|^2_{\Sigma^{-1}} = \|b\|^2_{\Sigma^{-1}} - c^*(F_\gamma^* W \Phi^T \Sigma^{-1} b). \tag{2.7.2}$$

In practice, we choose f_γ so that ω_γ is on a (possibly over-sampled) grid. To minimize over all γ on this grid, we compute $\tilde{b} = W\Phi^T\Sigma^{-1}b$ once, and then compute $F_\gamma^*\tilde{b}$ for all frequencies simultaneously using a (possibly over-sampled) FFT. For every γ, the 2×2 matrix $(F_\gamma^* W \Phi^T \Sigma^{-1} \Phi W F_\gamma)^{-1}$ must be computed, but this is independent of b, so it can be computed once and stored. So amortizing this cost, we can find the best frequency ω_γ up to a fixed frequency resolution ϵ by computing over-sampled FFTs with complexity proportional to $\mathcal{O}((N2^{-\epsilon})\log(N)/\epsilon)$. Σ^{-1} is computed via a pseudo-inverse or regularized inverse, in order to make the process more robust.

For even higher accuracy in the frequency, a one-dimensional optimization using Matlab's `fminbnd` is performed in the region near the peak frequency found by the FFT method. This function does not use the FFT, but is efficient since it typically only requires a handful of calls; it is also possible to use the Goertzel algorithm, but this would probably not offer an advantage. See Figure 2.41 for a plot of the minimal values of the functional in a region near a minima. In special cases, this function is convex [ERW11] within small regions near the minimum. For our case, empirical results suggest that it is also convex in a reasonably wide interval around its minimum. For very noisy signals, the high accuracy search is not performed, since this level of precision is generally beyond the accuracy level of the estimation.

Note that with the matched filter, there is less difference between input signals x that are "on-grid" and "off-grid". The spectral leakage which hurts ℓ_1-based recovery is no longer an issue, but the sacrifice is the simpler signal model.

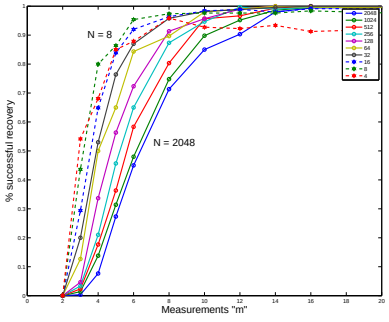


Figure 2.42: Rate of successful recovery via matched filter as function of M and N. "Success" is defined as frequency error less than .1 KHz. Input is tone of random phase and frequency. Matrix Φ is chosen iid signed Bernoulli ± 1. Recovery shows a strong dependence on M, and a weak dependence on N, as expected.

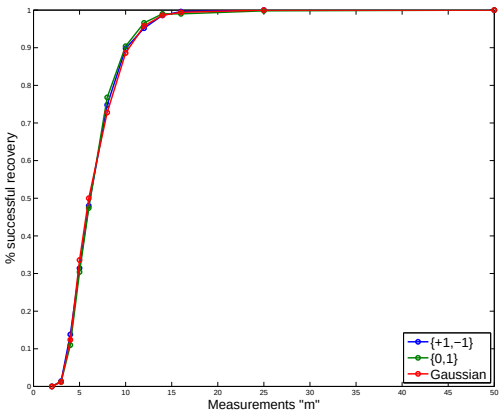
Figure 2.42 shows recovery of a randomly chosen sinusoid from M noiseless compressed measurements, using a $M \times N$ matrix Φ with each entry iid signed Bernoulli. The signal has three

parameters: amplitude, frequency, and phase, so typically $M \geq 3$ measurements are needed to have 100% recovery. Note that recovery rates are lower when N is large, since this enlarges the size of the ambient vector space. Because the frequency is chosen uniformly at random from $[0, f_s/2]$, this is an infinite-dimensional problem and more closely resembles Logan's phenomena [Log65, DS89] than compressed sensing. To see how we can recover from so few measurements, consider the case when the amplitude is known and we have two time samples, one at t_1 and one at $t_1 + \Delta T$, so $M = N = 2$. Let the signal be $\sin(\omega t + \theta)$, and the measurements are $b_1 = \sin(\omega t_1 + \theta)$ and $b_2 = \sin(\omega t_1 + \omega \Delta T + \theta)$. Then ω and θ can be determined by solving a 2×2 system of equations using the data $\sin^{-1} b_1$ and $\sin^{-1} b_2$; this is not always solvable since $|\sin^{-1} b_1 - \sin^{-1} b_2| < \pi$, but if the signal is band-limited then $\omega \Delta T < \pi$ and there is a unique solution.

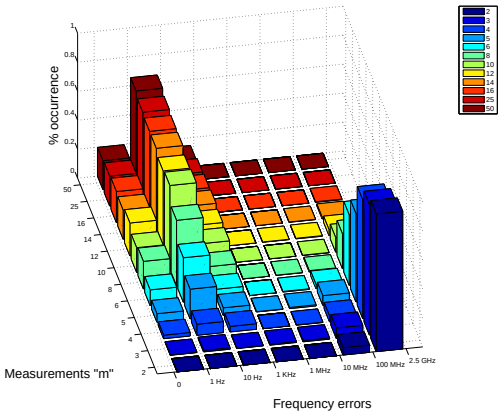
For the special case when Φ is a subset of the identity, and the frequency is on-grid, then we can be more specific. This is now a finite-dimensional problem. If N is prime, then Theorem 1.1 in [CRT06] guarantees that the the matched filter will recover all signals composed of $K \leq M/4$ "on-grid" tones (the extra factor of 2 is because in the change-of-variables from time to frequency, a signal with K tones has $2K$ nonzeros in the Fourier transform; think of this as the price to pay for unknown phase). Furthermore, if $K > M/4$, then one can construct an un-recoverable signal.

In general, now let N be arbitrary, and again let the signal x have K "on-grid" tones so that it is exactly $2K$ sparse in the Fourier basis F, $x = F\alpha$ with $\|\alpha\|_0 = 2K$. If Φ is a $M \times N$ matrix with iid Gaussian entries, then ℓ_0 recovery will successfully recover x whenever all $M \times 4K$ submatrices of ΦF have only trivial nullspace. Thus typically $M \geq 4K$ is a necessary condition to guarantee recovery. The results in Figure 2.43, which have $K = 1$, show that recovery with $M < 4$ is very rare. When $K = 1$, the matched filter recovery is equivalent to ℓ_0 minimization.

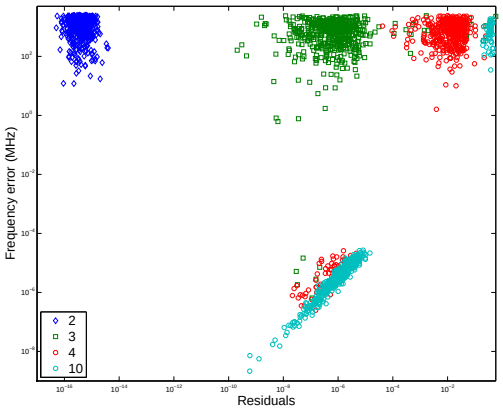
Figure 2.43 shows the results of some matched filter experiments which give an idea of the limits of recovery. The matrix is $\Phi \in \mathbb{R}^{M \times N}$ with $N = 1024$, and the results are a function of M; we do not yet impose a block structure on Φ or use a windowed signal. For comparison, the RMPI has $M = 80$ when $N = 1024$, it is well-above the threshold of recovery. The input signal is a pure sinusoid with random phase and (off-grid) frequency, and no noise. The matched filter is extremely computationally efficient when multiple trials use the same Φ matrix, so for each setting, 5 different Φ matrices were generated, and for each matrix 100 trials were carried out.



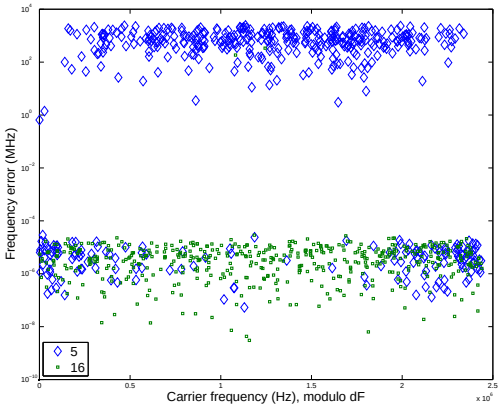
(a) Rate of successful recovery, N fixed. "Success" is defined as frequency error less than .1 KHz.



(b) Histogram of size of frequency errors



(c) Residual of matched filter



(d) Effect of off-grid frequencies

Figure 2.43: (a) uses data from three different Φ ensembles: either each entry $\Phi_{i,j}$ is chosen from $\{+1, -1\}$ uniformly, or from $\{0, 1\}$ uniformly, or from $(-\infty, +\infty)$ with a normal distribution. In all subsequent subfigures, we only show data from the $\{+1, -1\}$ model since the results are not significantly different among the ensembles. In (a), we confirm our intuition that more samples (larger M) improves the chance of recovery. In (b), a histogram of the type of frequency errors is shown for each value of M. For large M, most errors are small, while for small M (e.g., $M = 5$), errors are either small or catastrophic. Subfigure (c) shows the relationship of the matched filter residual (2.7.2) to the frequency error. For large M, there is a clear relationship, likely due to how well the convex optimization solver worked. For $M = 2$, the error was always catastrophic, but the residuals were small since the matched filter is an exactly-determined problem. For $M = 4$, only the non-catastrophic errors show a relationship between error and residual. Subfigure (d) plots the error as a function of how far the carrier signal frequency is from a "grid" frequency $dF = f_s/N_{\text{chip}} = 4.88/2$ MHz ($N_{\text{chip}} = N$ in this example; that is, the PRBS did not repeat). There is no discernible relationship for large M, which suggests that the matched filter recovery works equally well for tones that are on- and off-grid. For small M, there is a non-linear correlation: if the frequency was close to a grid-frequency, then a disaster was less likely to happen. But conditioned on whether a disaster occurs or not, the nearness to a grid frequency does not affect accuracy.

The matched filter can be used on real data as well, but in this case the exact pulse window is not known. The pulse window may be approximated by a rectangle, but a parametric search over the start and end of the rectangle must be performed; for speed considerations, this search may be done by a power analysis in the compressed domain. If more than one pulse is present, the matched filter proceeds in a greedy fashion by identifying the dominant pulse, then subtracting off

the pulse and repeating the process. This subtraction may not be accurate if the pulse window is not exactly rectangular, so we expect l1 recovery to perform better than matched filtering in this case. In general, if the window is unknown, matched filtering does not return an accurate estimate of the signal, but it can often return accurate pulse descriptor words (PDW) such as the frequency, phase, and time-of-arrival.

2.7.2 ℓ_1 recovery

Once data b have been taken, there are many options to recover an estimate $\hat{x}$ of the original signal x_0. This section reviews the basic techniques, plus many tricks that are essential in practice.

2.7.2.1 Analysis versus synthesis

Let $\mathcal{D}^*$ be the transform that induces sparsity in x, which is discussed in the subsequent section. For now, assume it is fixed. The two variants of ℓ_1 minimization are *synthesis*:

$$\underset{\alpha}{\text{minimize}} \ \|\alpha\|_1 \quad \text{such that} \quad \|\Phi\mathcal{D}\alpha - b\|_2 \leq \varepsilon, \tag{2.7.3}$$

and *analysis*:

$$\underset{x}{\text{minimize}} \ \|\mathcal{D}^* x\|_1 \quad \text{such that} \quad \|\Phi x - b\|_2 \leq \varepsilon. \tag{2.7.4}$$

The dictionary $\mathcal{D}$ is assumed to be a tight frame, so that $\mathcal{D}\mathcal{D}^* x = x$. If $\mathcal{D}$ is a basis, then the two programs are equivalent by a change-of-variables, since $\mathcal{D}^*\mathcal{D}\alpha = \alpha$. In general, $\mathcal{D}$ is an overcomplete dictionary, and the two programs are not equivalent. That is, if $x = \mathcal{D}\alpha$, it is not necessarily true that $\alpha = \mathcal{D}^* x$. The differences will be further explored in §3.6.1 and §4.7.5. Since $\mathcal{D}$ is overcomplete (say, $N \times D$ and $N < D$), x is N-dimensional and α is D-dimensional. The analysis problem has fewer variables, but the same number of constraints, so it is effectively more constrained.

As a simple example of the differences, let $\mathcal{D} = (I, I)$, and $x = e_1$ where e_1 is the first unit vector. By analyzing x, we have $\mathcal{D}^* x = [e_1; e_1]$ (in Matlab notation) which is 2-sparse. Via synthesis, we have $x = \mathcal{D}\alpha_1 = \mathcal{D}\alpha_2$ for $\alpha_1 = [e_1; 0]$ and $\alpha_2 = [0; e_1]$, and even $x = \mathcal{D}(t\alpha_1 + (1-t)\alpha_2)$ for any t. The synthesis coefficients *can* be sparser, but not they are unique and less stable

Some preliminary results about the differences appear in Chapter 5 of a recent PhD thesis [Ran09], where it is suggested that analysis often recovers signals more effectively. The Phase 0 report on the DARPA project reaches the same conclusion. The work in [EMR07] reports that analysis performs better than synthesis when using standard test images, and [CWB08] reports similar results on pulse signals. That work also notes that when reweighting, the differences are exacerbated. More experiments appear in [SED04, SED05]. Recent theoretical work [CENR11] has shown that analysis enjoys similar theoretical properties as the synthesis formulation does.

We propose an additional explanation, based on the observation that reweighting exacerbates the difference between synthesis and analysis. Consider the case of a dictionary that is heavily redundant. To simplify, we take this to the extreme, and assume that there are two atoms ϕ_1 and ϕ_2 which are identical; write α_1 and α_2 for their respective coefficients.

In synthesis, since α is the object of interest, it is intended to be sparse, so even if there is energy in the ϕ_1 component, it is likely that either α_1 is zero or α_2 is zero. Without loss of generality, let $\alpha_1 > 0$ and $\alpha_2 = 0$. If ℓ_1 minimization were performed on a slightly perturbed dataset $\tilde{b} \simeq b$, perhaps $\tilde{\alpha}_1 = 0$ and $\tilde{\alpha}_2 > 0$. But once reweighting is applied, the symmetry is broken. If $\alpha_1 > 0$ and $\alpha_2 = 0$, then the weights obey $w_1 < w_2$, so at all future reweighting steps, it is nearly guaranteed to keep $\alpha_1 > 0$ and $\alpha_2 = 0$. In a sense, the support has been locked-in. In a realistic case, ϕ_1 and ϕ_2 are not identical, so their coefficients are not interchangeable. If α_1 and α_2 are incorrectly identified after the first round, then reweighting will not fix this, and α becomes trapped in a local minimum. The opposite problem might happen too: reweighting is used to force some coefficients to zero, but if there are nearly interchangeable coefficients, forcing one coefficient to zero will just cause another similar coefficient to grow larger.

In contrast, analysis is much more uniform. If $\phi_1 \simeq \phi_2$, then α_1 is guaranteed to be close to α_2. Analysis can better exploit the reweighting technique. The downside of analysis is that extreme sparsity is not as possible, but this may be worth it for the trade offs.

To determine which formulation to use for the RMPI recovery, realistic numerical simulations were performed. A few of the results are shown in Figure 2.44. The results were overwhelmingly in favor of analysis, so all subsequent experiments have used analysis. However, these simulations were in the early phase of the project, and it would be useful to re-run the comparison tests using the better calibrated Φ matrices.

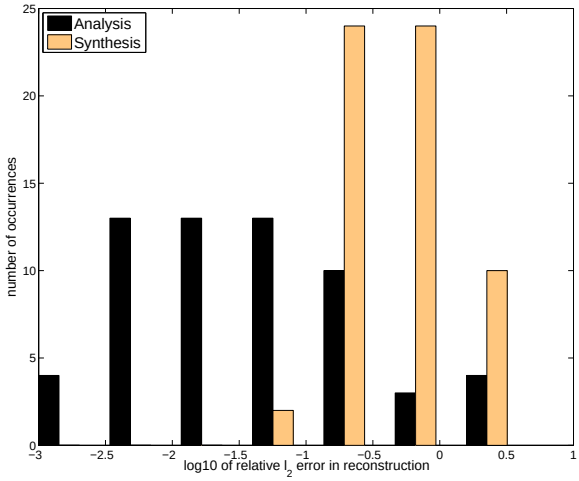


Figure 2.44: Performance of analysis formulation and synthesis formulation. Original tests from 2008 using l1Magic. Repeated over independent runs (noise varied) and 4 signal models: narrow and wide pulses, and low and high frequencies.

2.7.2.2 Dictionary choice

A general purpose reference for this section is [Mal08], but we don't cite specifics since most of this material is well known. Gabor frames go back to at least Gabor in the 1940s, who was inspired by musical scores. This type of harmonic analysis has been well-studied in many communities due to its widespread usefulness and the interest in wavelets beginning in the 1980s.

Dictionaries. A dictionary $\mathcal{D} \in \mathbb{R}^{N \times D}$ of D unit-norm atoms ϕ_d is chosen to represent the radar signal x: $x = \mathcal{D}\alpha = \sum_{d=1}^{D} \alpha_d \phi_d$ for some coefficients α. It is also possible to consider complex-valued dictionaries, but since x is real-valued, it is always possible to convert a length D complex coefficient vector α to a length $2D$ real-valued vector, so we assume wlog that α is real. The "analysis" operation is $\mathcal{D}^* x$, and the "synthesis" operation is $\mathcal{D}\alpha$. The statements below are tailored to finite dimensions, but we remark that most of these hold in any separable Hilbert space. In Hilbert space, a dictionary must have a dense span; in finite dimensional spaces, all we require is that $\mathcal{D}$ has full row-rank. If it doesn't have full column-rank, we say $\mathcal{D}$ is redundant or over-complete.

We are typically interested in over-complete dictionaries, so $D > N$, and thus it is not possible for $\mathcal{D}^*\mathcal{D} = I_D$, although it is possible for $\mathcal{D}\mathcal{D}^* = I_N$. The dictionary is a *frame* if $\mathcal{D}\mathcal{D}^*$ is non-singular, and a *tight frame* if $\mathcal{D}\mathcal{D}^*$ is a scaled identity aI_N, where we call a the frame bound. Thus a frame has a pseudo-inverse $\mathcal{D}^\dagger = \mathcal{D}^*(\mathcal{D}\mathcal{D}^*)^{-1}$, and for a tight frame, the pseudo-inverse is just a scaled version of the adjoint. The dual analysis operation is $\mathcal{D}^\dagger x$, and the dual synthesis operation is $(\mathcal{D}^\dagger)^* \alpha$. For tight frames, the analysis and dual analysis, and synthesis and dual synthesis, are the same, up to constants.

If $\mathcal{D}_1$ and $\mathcal{D}_2$ are both tight frames, then via block matrix multiplication, it is clear that $\mathcal{D} = [\mathcal{D}_1, \mathcal{D}_2]$ is also a tight frame.

There are three considerations when choosing a dictionary $\mathcal{D}$:

1. $\mathcal{D}$ should approximate the signal well, meaning that for every signal x, it can be represented $x = \mathcal{D}\alpha$ for some α that only has a few large coefficients.

2. The atoms ϕ_i of $\mathcal{D}$ should be as incoherent as possible; in other words, D should not be unnecessarily large.

3. All analysis and synthesis operations should be computationally efficient.

The signal model is that x is a radar pulse, which means it is localized in both time and frequency, so we seek a dictionary that approximates these signals well. For this reason, the Fourier basis is a poor choice, since all elements have full time support. However, due to the computation requirements, most realistic choices of dictionaries are based on the Fourier basis so that they can take advantage of FFT routines. Another class of dictionaries is based on wavelets, since there is a fast wavelet transform.

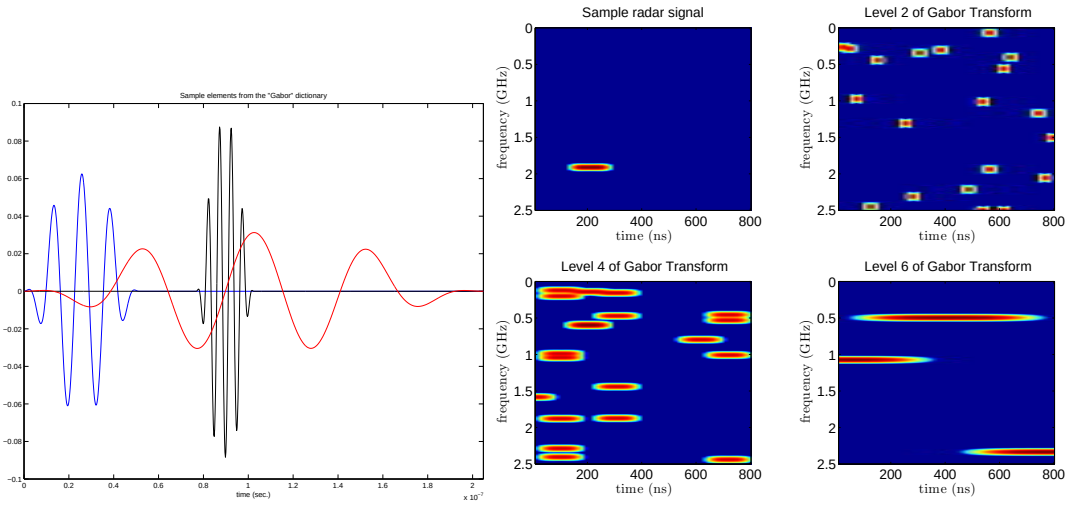


Figure 2.45: Sample elements from a multiscale Gabor frame. Left: time domain. Right: time-frequency plane

Gabor Frames. We consider Fourier-based dictionaries. Two major classes are the Gabor dictionaries and the lapped Fourier/DCT dictionaries (such as the modified discrete cosine transform used in MP3 compression). The two classes share similarities, and while we choose to present the dictionary in a Gabor framework, we have not lost much generality.

Let F be the $N \times N$ finite Fourier basis, with N columns f_d, where

$$f_d(n) = e^{i2\pi dn/N}.$$

Let g be a window function (e.g., Gaussian, sine), and write $G = \text{diag}(g)$. Define $g_m[n] = g[n - mM]$ to be a shifted version for some constant M, and similarly for G_m. Then a Gabor dictionary is

$$\mathcal{D} = [(FG_0), (FG_1), (FG_2), \ldots, (FG_{N/M-1})]$$

where we assume $M|N$.

The most useful Gabor dictionaries are those that are tight frames. To be a tight frame, we need

$$I_N = \mathcal{D}\mathcal{D}^* = F^*(G_1^2 + \ldots G_L^2)F.$$

In other words, g^2 must be a partition of unity when shifted by M. We consider only windows g such that g^2 is periodic in N.

The dictionary used for the RMPI relies on an iterated sine window:

$$g(x) = \sin(\pi/4(1 + cos(\pi x))), \quad x \in (-1, 1]$$

(implicitly, define $g(x) = 0$ outside $(-1, 1]$), and the discrete version is

$$g[n] = g(-1 + 2n/N), \quad n = 1, \ldots, N.$$

Using simple properties of sine and cosines, we see that $g(x)^2 + g(x+1)^2 = 1$, so this is a tight frame.

The trivial window $g_{\mathrm{F}} \equiv 1$ is valid ($M = N$), as is the window g_{T} which is only nonzero for a single discrete point. These can be viewed in the time-frequency plane as extreme types of tiling. Viewing the time-frequency (T-F) plane with time on the horizontal access and frequency on the vertical access (as in Figure 2.45), where as g_{T} corresponds to atoms that are very skinny (and also short). We would like a Gabor frame that tiles the T-F plane with atoms of compact support (so they approximate radar pulses well) and minimal overlap (to minimize coherence among the atoms). What can we expect? In the best case, we want exactly no overlap, which means that $\mathcal{D}$ is not only a dictionary but also a basis.

Unfortunately, the uncertainty principle prevents this, as quantified in Balian-Low theorem which states that no windowed Fourier *basis* exists if the window g is smooth and of compact support (clearly, if we allow g_{F}, then the $\mathcal{D} = F$ which is a basis). If we want well-localized T-F atoms, we must accept overlap, which is why $\mathcal{D}$ is usually over-complete.

We note that Mallat (§12.3.3 [Mal08]) describes a Gabor dictionary based on a discrete Gaussian window which is *nearly* a tight frame; the advantage of that particular construction is that it is translation-invariant and multi-scale. It may merit further consideration for the RMPI, but so far we stick to tight frames.

Multiscale Gabor Frames. Exploiting the property that concatenating two tight frames $\mathcal{D} = [\mathcal{D}_1 \mathcal{D}_2]$ gives a new tight frame, it is easy to build up a multi-scale Gabor frame.

Consider the window function g described previously, where $g(x)^2 + g(x+1)^2 = 1$ for $x \in (-1, 0]$. Let $g^{(2)}(x) \triangleq g(2x \mod (-1, 1])$, then $g^{(2)}(x)^2 + g^{(2)}(x + 1/4)^2 + g^{(2)}(x + 1/2) + g^{(2)}(x + 3/4) = 1$, so the dictionary defined with F and $G^{(2)}$ is also a tight Gabor frame. We can add this to the Gabor frame that used the original window, and the result is a new tight Gabor frame that has window functions at two levels. Clearly this can be repeated, so that windows of various widths are included in the dictionary.

A final twist on the windows is to introduce shifted versions. Since $g(x)^2 + g(x+1)^2 = 1$ and g^2 is periodic with period 2; it is also true that $g(x-1/2)^2 + g(x+1/2)^2 = 1$. So in addition to windows G of different widths, we can also include windows that are shifted. In practice, we typically include about 2 levels of narrow windows, and one shifted variant of each narrow window.

Frequency scalings. Previously, attention was restricted to the $N \times N$ Fourier basis F, which was windowed by some diagonal window matrix G, where the Fourier atoms are $f_d(n) = e^{i2\pi dn/N}$ for $d = 1, N$. It's also possible to use a zero-padded or sub-sampled Fourier frame. The $2\times$ zero-padded Fourier frame consists of atoms

$$f_d(n) = e^{i2\pi dn/(2N)} \quad \text{for} \quad d = 1, \ldots, 2N$$

so it has half the spacing in frequency. Likewise, a $2\times$ sub-sampled Fourier set consists of

$$f_d(n) = e^{i2\pi 2dn/N} \quad \text{for} \quad d = 1, \ldots, N/2$$

which, by itself, does not span the space.

The multiscale Gabor frame that we use in practice will adjust F to be oversampled when using a wide scaling of the window, and a sub-sampled Fourier set when using a narrow scaling of the window. A basic theorem by Daubechies (see [Mal08]) shows that the frame bounds (of a single-scale Gabor frame), and hence the redundancy, are directly related to the product of the frequency scaling and the time scaling.

In general, the parameters of the Gabor frame are picked via trial-and-error by testing which frames give the best reconstruction results. This also depends on the length N used. Because test signals are generated with a known window, it would be easy to overfit the parameters of the Gabor frame to work very well for certain test signals, and hence we do not try to fine tune the parameters. Real radar signals are generally "supposed" to be trapezoidal, but they are generated by waveform generators with limited bandwidth, which has the effect of smoothing the trapezoidal window. This is beneficial to recovery since the signal will have fewer sharp edges and hence a sparser frequency representation.

For a typical reconstruction, the Gabor frame is between $8\times$ and $64\times$ overcomplete, and usually between $12\times$ and $20\times$ overcomplete. The signal length N is between 2^{10} and 2^{14} depending on what calibration matrix is available and how much computational time can be spent. Hence for $N = 2^{14}$, the dimension D can be as large as $2^{20} \approx 10^6$. This was one of the main motivating factors for developing the rapid algorithms described in subsequent chapters. For example, NESTA solves a problem with size $N = 2^{12}$ in about 1 minute.

Given a dictionary $\mathcal{D}$, we solve either the ℓ_1 analysis problem which uses weighted norm $\|\mathcal{D}^* x\|_1$ and measurement matrix $A = \Phi$, or the ℓ_1 synthesis problem with norm $\|\alpha\|_1$ and measurement matrix $A = \Phi\mathcal{D}$. The conclusion of this thesis discusses possible improvements over using a Gabor frame by itself; model-based ideas will probably be useful in future versions.

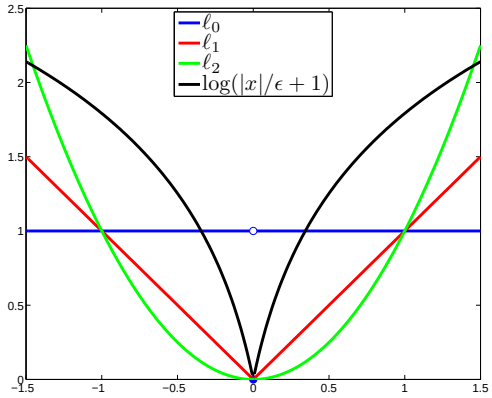


Figure 2.46: The ℓ_0 quasi-norm and various approximations to it. There are two features we wish to capture: a sharp cusp at 0, which will force small values to exactly 0, and a slow growth at infinity, so that large values are not excessively penalized. The ℓ_1 function is in some senses the best possible convex choice.

2.7.2.3 Reweighting

Consider the generic ℓ_1 synthesis problem

$$\min_x \|x\|_1 \quad \text{such that} \quad \|Ax - b\|_2 \leq \varepsilon.$$

The following discussion will also apply to ℓ_1 analysis, but we use the synthesis form for simplicity of exposition. In both cases, the reweighting vector will apply to an object in the coefficient domain (e.g. of size D) and not in the signal domain, but we use x to represent the object to be general.

The assumption is that x is sparse, and thus what we really would like is the solution to the problem

$$\min_x \|x\|_0 \quad \text{such that} \quad \|Ax - b\|_2 \leq \varepsilon$$

where $\|x\|_0$ is the quasi-norm which is the number of nonzero entries in x.

The reweighting approach is to solve a series of weighted ℓ_1 problems

$$\min_x \|Wx\|_1 \quad \text{such that} \quad \|Ax - b\|_2 \leq \varepsilon$$

where $W = \text{diag}(w)$ for a weight vector

$$w_i = \frac{1}{|x_i^{\text{old}}| + \delta} \tag{2.7.5}$$

and x^{old} is a guess of the solution which is updated after every ℓ_1 solve. If $x^{\text{old}} = x_0^\star$ where $x_0^\star$ is the solution to the ℓ_0 problem, then clearly in the neighborhood of $x_0^\star$ the weighted norm approaches the value of the ℓ_0 quasi-norm as $\delta \to 0$. This scaled problem can always be solved by existing ℓ_1

algorithms since even if the algorithm doesn't explicitly handle scalings, a change of variables can be made.

The idea of reweighting is to better approximate the $\|x\|_0$ norm. We follow the approach of [CWB08], which motivates this approach using a log-based approximation mentioned in Fazel's thesis in 2002 [Faz02] if not earlier. The idea is also very similar to the idea of iteratively re-weighted least-squares (IRLS); see [DDFG10] for a modern perspective on this old algorithm. Indeed, IRLS has moved beyond its original use as just a robust replacement to least-squares, and is now used to approximate ℓ_1 norms or even ℓ_p ($p < 1$) norms.

There are variants of this idea in the literature; see [CWB08] and [Cev09] for further references. We briefly note FOCUSS [GGB95] which is a special form of IRLS, and the work of Chartrand and collaborators on both algorithms and theoretical signal recovery guarantees for ℓ_p minimization for $p < 1$ (see, e.g., [Cha07, CY08]). In general, the weight vectors can be chosen in many ways and regularization parameters δ can be chose in many fashions; e.g.,

$$w_i = \frac{1}{|x_i^{\text{old}}|^2 + \delta^2}$$

and the reweighted problems can be of ℓ_1 or of ℓ_2 type. It is our opinion that the differences among these schemes is probably not very great, and so we have restricted attention to weights of the form (2.7.5). Other variants, such as iterative-support detection [WY10], set the weights of all nonzero coefficients to zero. This approach would be interesting to try in future work.

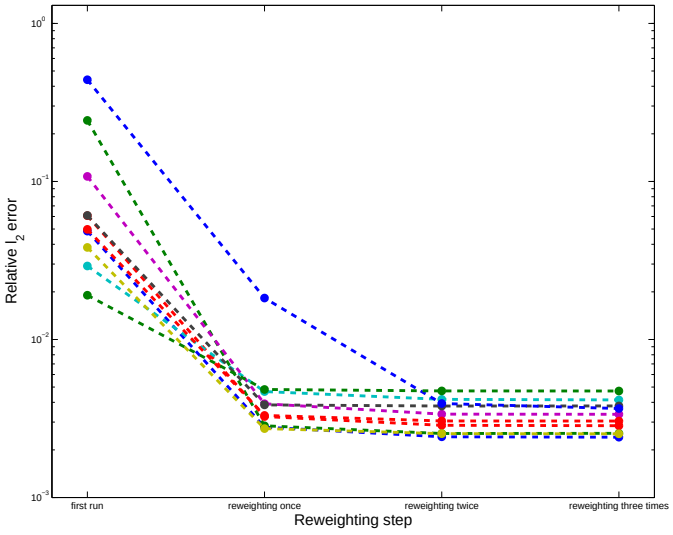


Figure 2.47: Early reweighting results from 2008, using l1-Magic [CR07b]. Shown are 10 independent runs (each color is an independent run). Signal is a pulse with 2.21 GHz carrier frequency. Reweighting is clearly beneficial.

Log approximation. The approach we take to motivate the weights is to approximate the $\|x\|_0$ by $\log(|x|+\delta)$ for some small δ. See Figure 2.46 for the graph of this log function, and the comparison

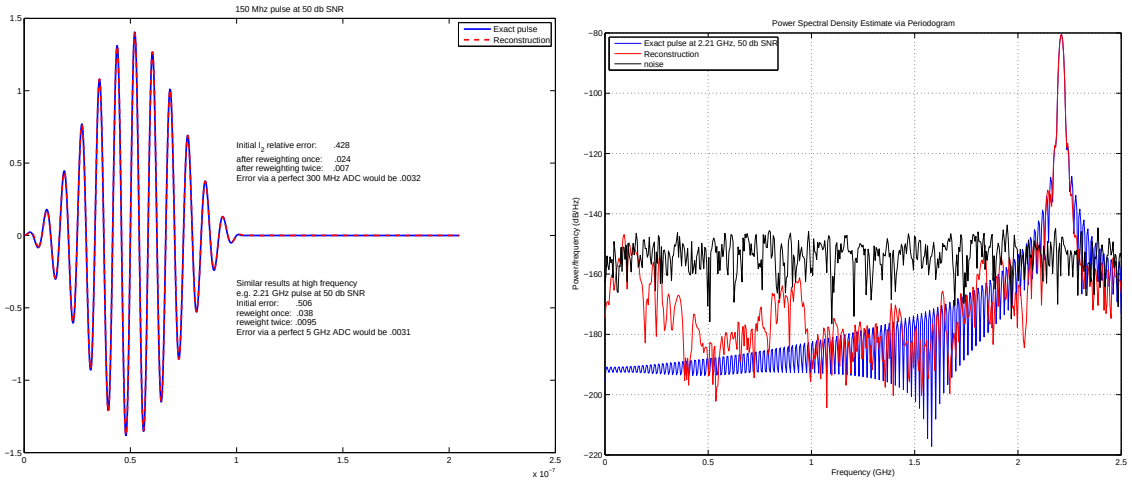


Figure 2.48: Example of a reconstruction that used two rounds of reweighting, which were very beneficial. In the frequency domain, the input noise is shown in black (but not shown in the time domain picture, for clarity). The recovered solution is less noisy than equivalent Nyquist-rate samples.

to the standard ℓ_1 norm (in this 1D graph, just the absolute value function). The log function is not convex, and in fact it is concave if restricted to the positive orthant. The reweighted ℓ_1 approach is a gradient descent algorithm to minimize this concave function; so far, there are no known results about the convergence (to a global minimizer) of this algorithm. However, [Nee09] has basically shown that in terms of recovering sparse vectors, the reweighting procedure will not hurt.

Consider $f(u) = \log(u + \delta)$ for $u > 0$ (for example, introduce the extra variable u and constraint $|x| \leq u$). This is concave, and hence it is below its tangent; we say that the tangent line "majorizes" f. Thus taking a gradient descent step will result in a new point with smaller value f. This is part of the MM framework [HL04] (which in this case, stands for "majorize minimize"). The gradient of f a point u_0 is $\nabla f = 1/(u_0 + \delta)$, so the linearization at u_0 is

$$f(u_0) + u/(u_0 + \delta)$$

and minimizing this (ignoring the constant terms) is equivalent to solving a reweighted ℓ_1 problem with weights $w = 1/(u_0 + \delta)$ with the constraints $|x| \leq u$.

Because this is non-convex, the starting point is very crucial, and we have observed in practice (see §2.5.2.3) that using a DCT basis, prior to using a Gabor frame, to get a suitable starting value generally works better. Furthermore, unlike the log model above, we adaptively choose δ every round instead of fixing it. This works better in practice but makes convergence results even trickier.

The regularization parameter $\delta > 0$ is necessary since otherwise the weights become infinite for any zero coefficients. If $\delta \approx 0$, then the weights might be huge for nearly zero coefficients, and the method will be less stable and take a long time to converge. As δ becomes larger, fewer iterations are needed. If c is the smallest magnitude of the true solution's nonzero entries, then the idea is to

pick $\delta \approx c$, though of course c is unknown.

The RMPI experiments have used two rules for picking δ; both rely on a new percentage parameter ρ which is typically set to .95 or in the range $[.8, .99]$. Let $u = |x|$ be the current estimate of $|x|$ from the previous iteration, and assume wlog that u is sorted from large to small. Let D be the dimension of x. The rules are

1. Let K be the nearest integer to ρD. Pick δ such that $u_K \leq \delta < u_{K+1}$. The idea is that we want a sparse approximation, and we use ρD so that this scales automatically if the size of the dictionary is changed.

2. For any δ, let T_δ be the set of entries of u that have magnitude greater than δ. Pick δ to be the smallest number such that
$$\sum_{i \in T_\delta} |x_i|^2 \geq \rho \sum_{i=1}^{D} |x_i|^2.$$
The idea is that we choose the threshold so that ρ percent of the signal's energy is explained by coefficients of this size or greater. This is computationally efficient to carry out by using a cumulative sum of sorted coefficients.

Both rules depend on ρ, and both perform about the same. Recent experiments have been using rule 2. There are some signals that are well recovered only when ρ is tweaked, but this value of ρ is signal dependent, so we keep ρ relatively constant to avoid overfitting to specific signals.

The reweighting steps are performed for a fixed number of iterations (usually between 2 and 5) or until the new iterates are sufficiently close to previous iterates.

Improvements. The basic premise of recent model-based CS theory [BCDH10] is that realistic signals have much more structure than just sparsity in a basis. For example, the coefficients of a multi-level transform of a signal will not be independent. If energy is present in a small scale coefficient, then some energy is likely present in a "parent" coefficient.

The works of [BCDH10, Cev09, JMOB10] have proposed using special thresholding or proximal method algorithms to enforce special structure. The structure is usually of a hierarchical form. For example, in a multi-scale transform, a tree can be drawn of parent-child windows. The structure assumption is that if a child coefficient is nonzero, then so is the parent node. Adding this additional assumption usually improves recovery, since it is restricting the feasible set to be smaller. The RIP bound can be changed to an RIP over the model set, which should produce better RIP constants.

To stay within the ℓ_1 framework, we suggest using a tree structure of multi-level transforms and then choosing weights to reflect this. For example, instead of reweighting each entry by

$$w_i = \frac{1}{u_i + \delta}$$

we reweight by

$$w_i = \frac{1}{\tilde{u}_i + \delta}, \quad \tilde{u}_i \triangleq \max_{(j \text{ is a child of } i)} u_j$$

(with the convention that a node is a child of itself). A similar idea has been independently suggested in [DWB08]. Unfortunately we have not yet tested these ideas, since it adds an extra layer of complexity and parameter choice on top of the already numerous parameters for selecting a dictionary.

Even if not using a tree structure to determine weights, the multiscale nature of the Gabor dictionary should be taken into account. For example, the coefficients at each level of the transform should have a specific regularization parameter δ_{level}. This has not been implemented yet and is expected to help. In general, we note that there is much existing knowledge about *denoising* via wavelet thresholding, and it is our opinion that *reweighting* techniques can benefit. For example, cycle spinning (averaging over various shifts to make the transform translation invariant) and group thresholding (using local neighborhood information to inform thresholding/reweighting) are well-known in the denoising literature (e.g., [DJ94, Nas08]), and it seems likely that these techniques would improve reweighting performance.

2.7.2.4 Debiasing

Solving an ℓ_1 minimization with inequality constraints problem generally introduces bias into the solution. Consider the generic problem

$$\min_x \|x\|_1 \quad \text{such that} \quad \|Ax - b\|_2 \leq \varepsilon.$$

By duality, if A has full row-rank, there is some scalar $\lambda > 0$ such that this is equivalent to solving

$$\min_x \lambda\|x\|_1 + \frac{1}{2}\|Ax - b\|_2.$$

To analyze this, take a very easy case: $A = I$. Let x_0 be the true signal we wish to estimate, and $b = x_0 + z$ be the measurements where z is any zero-mean random variable. Since A is the identity, the solution is known in closed-form (this is just the proximity operator of ℓ_1, discussed in detail in Chapter 4):

$$\hat{x} = \text{sgn}(b) \cdot [\,|b| - \lambda]_+$$

where $[\cdot]_+$ means the positive part. For the positive entries $i \in \hat{T_+}$ of $\hat{x}$, this is

$$\hat{x}_i = b_i - \lambda = (x_0 + z)_i - \lambda.$$

Regardless of the dimensions of the problem, these entries are biased by the amount λ.

For general matrices A, the estimates are biased as well. For a sparse recovered vector $\hat{x}$, the fix is quite simple. Set $\hat{T}$ to be the support of $\hat{x}$, and then solve the over-determined least-squares problem

$$\min_{x:x|_{\hat{T}^c}=0} \|Ax - b\|_2$$

to find the best vector, with support fixed to $\hat{T}$, that explains the data. If the support $\hat{T}$ is equal to (or an unbiased estimator of) the true support T of the original signal x_0, then since the least squares problem is unbiased, this procedure removes the bias from the solution.

Unfortunately, when x_0 is not exactly sparse but rather just compressible, the support $\hat{T}$ is generally large, and even if it is smaller than N so that the least-squares problem has a unique solution, the debiased version fails to estimate the original signal well. For recovering radar pulses in the Gabor frame, debiasing was tried many times, but it never helped, and thus it is not used. As suggested by the $A = I$ example, coefficients are biased by an amount unrelated to their magnitude, so that small coefficients have more bias relative to their size, yet clearly small coefficients are very sensitive to errors in determining $\hat{T}$. We leave this issue for further study.

Fortunately, empirical evidence is that reweighted techniques remove bias, and since reweighting is used heavily in the reconstructions, we do not worry much about bias. One simple reweighting technique, called the *adaptive LASSO* [Zou06] in the statistics community, has been analyzed. In particular, [Zou06] proved a conjecture from 2001 that the LASSO can be inconsistent, and proved a celebrated result that the adaptive LASSO is consistent. The adaptive LASSO is basically one step of reweighted ℓ_1 minimization. The weights are chosen inversely proportional to some power of the absolute value of coefficients of any consistent estimator (such as the least-squares estimator); it is closely related to the non-negative garotte introduced in 1995 [Bre95], as well as more recent two-step procedures such as [KXAH10]. Intuitively, reweighted ℓ_1 should perform even better. The fact that the adaptive LASSO is consistent does not prove it is unbiased, but suggests that reweighting is useful.

2.7.2.5 Non-linearity correction

Although there is current work such as [Blu11] aimed at extending compressed sensing to work with non-linear observations, the theory and algorithms for *directly* handling non-linearities is in its infancy. In this section we review an indirect method of partially compensating for non-linearities in the data collection.

So far, the working assumption has been that data are collected

$$b = \Phi x$$

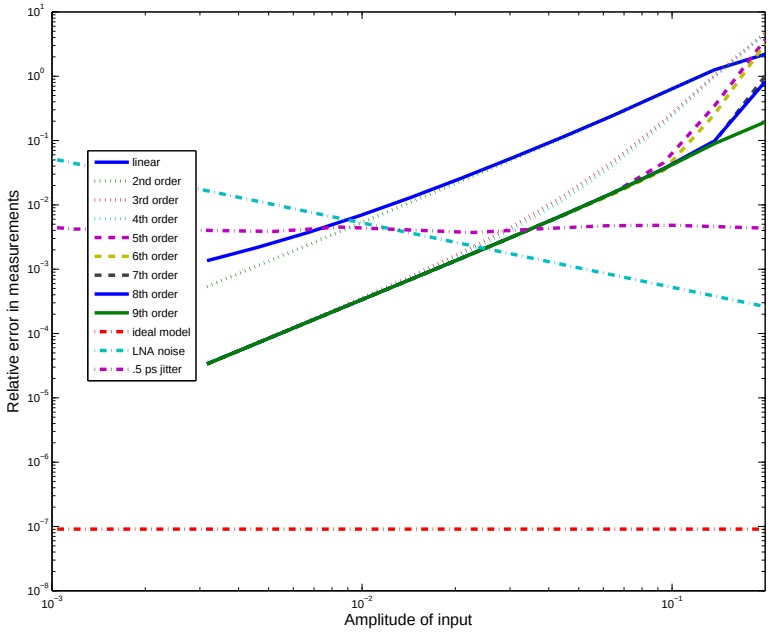


Figure 2.49: Using a non-linear Simulink model, trying to compensate for known non-linearity with a polynomial model by adjusting the predictions from just Φx to $\Phi p(x)$; note that we should not expect perfect agreement since $p(x)$ is calculated on the Nyquist grid, whereas non-linearities will induce harmonics outside this band. Input was 700 MHz tone. Also shown are errors due to jitter and noise (and the ideal case), for comparison. The higher terms of the polynomial only affect large amplitudes, as expected.

which we call the linear model (for conciseness, we don't write down the noise term). In reality, the circuit is non-linear, so a more appropriate model is

$$b = b_0 + \Phi_1 x + \Phi_2 x^2 + \dots.$$

For example, the non-linearities in the mixer and integrator will make such a model. This model is difficult to deal with since the matrices Φ_i are unknown and would be difficult to measure via calibration.

Fortunately, the dominant non-linearity is in the LNA front-end; since this occurs before the mixing, the LNA portion of the system is approximately separable over time in the incoming signal, like a diagonal matrix. This leads to the following model:

$$b = \Phi p(x), \quad p(x) = a_0 + a_1 x + a_2 x^2 + a_2 x^3 \dots \tag{2.7.6}$$

which is much more tractable since the polynomial now has scalar coefficients. Like many RF designs, the dominant terms in the polynomial are a_1 and a_3. For small amplitudes, $p(x) \approx a_1 x$ so the effects of higher-order terms are not significant. But the growth of x^3 outpaces that of x so at some point the third-order term takes effect, and this is the effective upper limit on the system. The point at which this happens can be measured via several standard conventions, such as IIP3. See

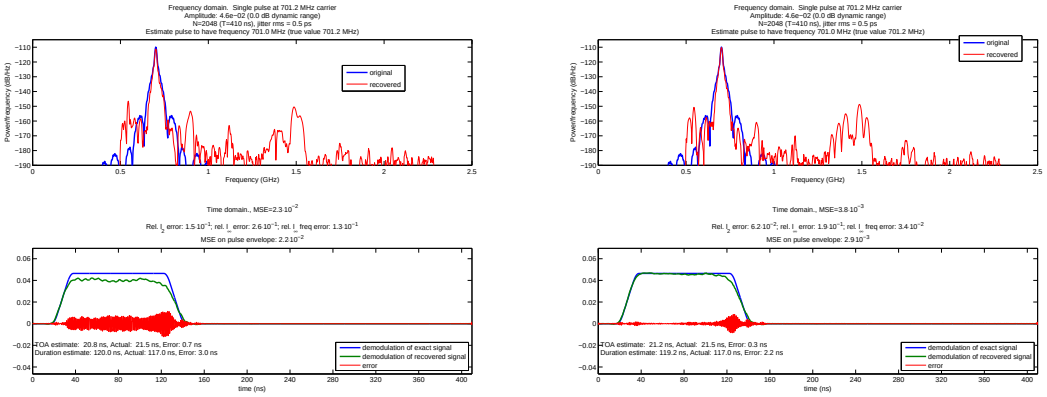


Figure 2.50: (Left) Before non-linearity correction. Notice the gain compression. (Right) After non-linearity correction. See also Figure 2.51.

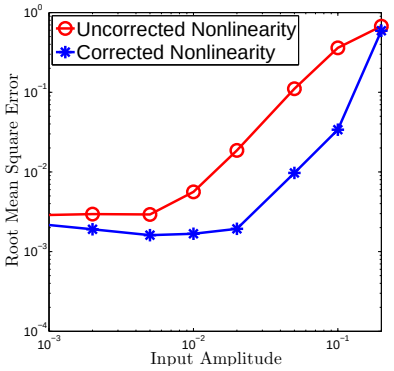


Figure 2.51: Error of reconstruction as a function of input amplitude A, with and without non-linearity correction. Input was a single signal. At very large amplitudes, no post-processing technique can help.

Figure 2.49 for effects of various terms of the polynomial as a function of amplitude.

For now, assume p is known (that is, its coefficients and degree). Is it possible to correct for this effect in post-processing? The answer is yes; see Figure 2.50. However, the technique we present works poorly on high frequency signals. We hypothesize that this is due to the discrete representation of the signal in post-processing. Dealing with a vector x of length n (where Φ has n columns) implicitly assumes that x is band-limited to maximum frequency $f_s/2$. However, non-linearities introduce harmonics, so a high frequency signal with large amplitude will have a component past $f_s/2$. When working with the discrete representation, this is aliased into lower frequencies, and is harder to correct for. It is possible that this problem can be fixed by allowing longer discrete vectors, so that frequencies can go up to f_s or even higher.

We introduce the basic technique. This is a classic example of what applied mathematicians and physicists do best: linearization.

Start with the assumption that $b = \Phi x$ and solve the standard ℓ_1 minimization problem to find

a guess x_1. Now, linearize p about x_1:

$$p(x_1 + h) = p(x_1) + p'(x_1)h + \mathcal{O}(h^2)$$

and update the model to be

$$b = \Phi p(x) + (\Phi p'(x))h, \quad \text{i.e.} \quad \tilde{b} = b - \Phi p(x),\ \tilde{\Phi} = \Phi p'(x).$$

This model is linear in h, so the ℓ_1 minimization with $\tilde{b}$ and $\tilde{\Phi}$ is solved for h. Then the variable is updated

$$x_2 \leftarrow x_1 + h$$

and the process is repeated by linearizing p around x_2, etc.

This technique was tested on synthetic data from Simulink simulations, where we know exactly the non-linearity that Simulink uses (these values taken from SPICE simulations), and it works well on low and medium frequencies. The entire polynomial, up to 9th order, is known; to test robustness, the correction was tried using different number of terms. In reality, it would only be reasonable to correct for the 2nd- and 3rd-order terms, since high order terms would be impossible to estimate reliably. Reconstructions used either 3 or 6 terms in the polynomial. These Simulink tests do show some robustness, since even though the 9th-order polynomial is fully known, the measurements are not exactly in the form of (2.7.6) since the transconductor and common-gate polynomials are applied *after* mixing, so the polynomial is not separable.

A final note: if the harmonics fall out-of-band and are caught by a low-pass filter, this appears to solve the non-linearity problem. However, the effect of the non-linearity is still there, and it is now more appropriate to look at the intermodulation effect: when two tones are passed in the receiver, the non-linearity causes harmonics at the sum and difference of the two frequencies. Furthermore, in addition to distortions of large signals, non-linearities cause "desensitization" and blocking of small input signals that are concurrent. If there is significant noise after this point, there is not much that post-processing can do to recover the small signal. Hence, ultimately non-linearities remain a circuit design issue, and can only be partially corrected in software.

Unknown coefficients. The case when p is not known is difficult, and was not tested numerically. The SPICE model can help give estimates of p, but these will never match the true values of the circuit. We did not yet test the robustness of the non-linearity correction to inaccurately estimated coefficients.

It may be possible to use the EM or MM algorithm [DLR77, HL04] and iterate between non-linearity correction for a fixed p, and then updating the estimate of p. This would be similar to the

dictionary learning techniques that have been used quite successfully in image processing. We leave this issue for further study. Another variation left for further study is the possibility of allowing for a polynomial with *matrix* coefficients Φ_i, since this would model non-linearities other than just the LNA non-linearities, but would require a novel calibration scheme.

2.7.2.6 Windowing

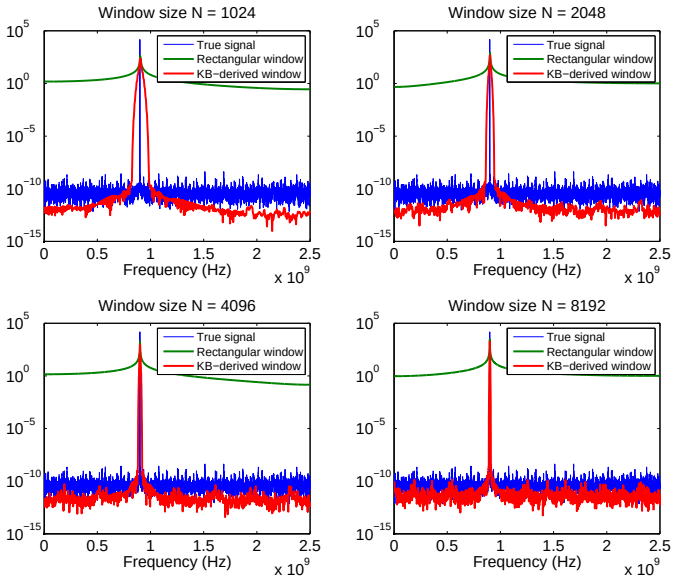


Figure 2.52: The importance of windowing. For an arbitrary off-grid tone, a finite set of samples will experience spectral spreading; this is due to the Fourier transform of the boxcar function, which has very slow decay due to the sharp corners. Using a smooth window introduces some loss of time resolution but offers much greater frequency resolution, and is usually worth the loss of time resolution. For any type of window, the larger the number of samples N, the less spectral spreading; this is a reason why fast methods for large-scale optimization, such as those developed in Chapters 3 and 4, are essential in signal processing. This figure shows the same signal but with different lengths N chosen for processing, with small N in the upper left and large N in the lower right.

The simulations and analysis in this chapter assume that a pulse x is located in the middle of a known time interval, so boundary effects are not significant. In a deployed system, obviously this assumption is invalid. The simplest solution is to increase the chunk of time under analysis. This increases the chances that a signal is in the middle of the interval, and it also reduces the frequency spreading that would be induced by windowing a pure tone by a rectangular window. However, due to processing limitations (since reconstruction algorithms are so far more than linear complexity), there is an upper limit on the size N of the discrete signal. We also would like to seamlessly combine signal estimates from different blocks.

The standard solution to this problem is to introduce a smooth window, and then overlap adjacent chunks of size N, e.g., a 50% overlap. There are many choices of windows, each with a unique time-frequency resolution trade off, and this has been well-studied for many years. It is also convenient to choose windows that form a partition-of-unity (also known as the Princen-Bradley condition), such

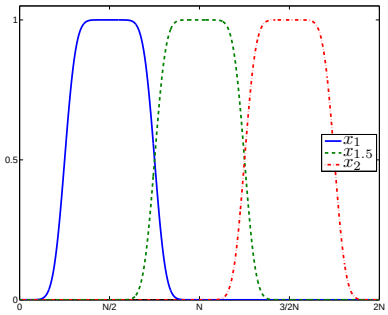


Figure 2.53: Overlapping windows which form a partition of unity, for blocks of length N. Using a Kaiser-Bessel derived window, with parameter $\alpha = 8$

as the windows previously mentioned in the Gabor dictionary section; see Figure 2.53. This is the technique of a modified discrete cosine transform (MDCT) used for MP3 compression, to give just one example. For a compressed sensing device that takes non-uniform time-samples, like the NUS, windowing can be applied "retroactively". For NUS, samples are taken

$$b = Sx$$

where S is a $M \times N$ matrix that is effectively diagonal, since it is just a $N \times N$ identity matrix with $N - M$ of the rows removed. Suppose w is a window and let $W = \mathrm{diag}(w)$ be the matrix version. We desire

$$\tilde{b} = SWx.$$

Because S is so simple and $SS^T = I_M$, it effectively commutes with W; specifically,

$$\tilde{b} = (SWS^T)Sx = \tilde{W}b, \quad \tilde{W} = (SWS^T)$$

so it can be recovered *ex post facto*.

The case of RMPI is unfortunately more difficult, since the measurements Φ have a much more complicated structure than that of S. Again, we seek

$$\tilde{b} = \Phi W x.$$

This time, it is not possible to write $\tilde{b}$ in terms of $b = \Phi x$.

One possibility to overcome this is an iterative scheme, which we suggest here and may be novel. Start with an estimate x_1 of x, obtained via ℓ_1 minimization without windowing. Then update the observations

$$b_1 \leftarrow b - \Phi((W - I)x_1).$$

If the estimate x_1 is very accurate, then $b_1 \simeq \tilde{b}$. Form a new estimate x_2 by re-solving the ℓ_1

minimization using b_1. This process is then repeated until convergence. For best results, each stage would use many consecutive windows to produce estimates of x in a particular (probably overlapped) time block, and then combine them to make a global estimate of x before proceeding with the next iteration. This idea has not yet been tested numerically.

The authors of [TLD+10] suggest adding a time-varying channel and doubling the number of channels in order to window the signal in hardware, before the integration step. This is a serious design change, and reflects the difficulty of the windowing problem. It seems likely that a software solution can obviate such a step; as shown by matched filter, it does not appear to be a fundamental problem with the manner in which the data were collected.

It is also possible that when restricted to radar pulse, and hence using the Gabor dictionary, that windowing of overlapping blocks is not necessary, due to the the localized time support of the Gabor atoms themselves. More study is clearly needed.

2.7.2.7 Further improvements

These improvements have not yet been tried due to time constraints, but we expect that some of them may have beneficial effect.

Modeling noise. The experiments presented have always used the constraints

$$\|\Phi x - b\|_2 \leq \varepsilon.$$

However, most thermal noise is injected before measurements. A complete noise model (ignoring jitter) is

$$b = \Phi(x + z_1) + z_2$$

where z_1 likely dominates z_2, and both are iid. This can be modeled using a single z_2 term which incorporates Φz_1. If both z_1 and z_2 are i.i.d. Gaussian with variances σ_1^2 and σ_2^2, respectively, then modeling both together is equivalent to a new z_2 term with covariance matrix

$$\Sigma = \sigma_1^2 \Phi\Phi^T + \sigma_2^2 I. \tag{2.7.7}$$

With no other priors on the signal, the maximum likelihood estimate of x is

$$\hat{x} = \underset{x}{\operatorname{argmin}} \langle \Phi x - b, \Sigma^{-1}(\Phi x - b)\rangle.$$

This inner product can be concisely written as $\|\Phi x - b\|^2_{\Sigma^{-1}}$.

Of course we have *a priori* knowledge that the signal x is sparse, so we use ℓ_1 minimization. It is then only natural to solve the ℓ_1 problem with a weighted ℓ_2 norm for the constraints. For the

analysis problem, this is

$$\underset{x}{\text{minimize}} \ \|\mathcal{D}^* x\|_1 \quad \text{such that} \quad \|\Phi x - b\|_{\Sigma^{-1}} \leq \varepsilon.$$

This can be solved by existing ℓ_1 algorithms without modification by updating $\Phi \leftarrow \Sigma^{-1/2}\Phi$ and $b \leftarrow \Sigma^{-1/2} b$. The estimate of Σ^{-1} should be performed by pseudo-inverse, or σ^2 should be artificially enlarged, in order to keep it stable.

A final remark is that the parameter ε is an issue. In theory, it can be estimated from the chip during a calibration phase when it is known that there is no input. The variance of iid white noise can be estimated with the sample variance; if it is suspected that other factors are involved, then a more robust measure is mean-absolute-deviation. To estimate the covariance matrix of correlated white noise, the situation is more involved, but there are existing techniques. The maximum likelihood estimate is just the sample covariance, but this could be improved by imposing structure on the covariance matrix, e.g., of the form (2.7.7). This can be solved with software such as TFOCS, discussed in Chapter 4. In the simulations in this thesis, ε was either estimated manually or based off of $\|b\|_\infty$. Performance does depend on ε but it is not overly sensitive. The only time an accurate value of ε is crucial is when one is trying to recover two concurrent tones that are widely different in dynamic range.

Jitter. Jitter has been treated like additive white noise, but this is not true at all since it is signal dependent. Preliminary work by Jerry Fudge's team at L-3 attempts to take this into account, but details are not public and it may not be applicable to our design.

Here, we mention another recent technique. Since jitter is part of the sampling, it has a non-linear effect on the measurements. For this reason, the likelihood function is not known in closed form. The authors in [WG11] propose numerical evaluation of the likelihood function via quadrature rules, and then use the EM algorithm ([DLR77]; see also [HL04]) to maximize the likelihood function, and report improvements. Their technique deals with *sampling* jitter. As mentioned earlier, sampling jitter is not a major concern for the system, but the *mixer* jitter is, and perhaps a similar EM algorithm could improve the system with respect to the mixer jitter.

Block norms. The reconstruction algorithm processes one chunk of length N (or time T) at a time. If the input signal x possesses structure that lasts longer than T, then this should be exploited. For example, if x is a pure tone, it will look nearly identical in every window, and hence the multiple observations should be combined in order to improve its estimate. Radar pulses typically repeat many times a second, so they also posses this structure. Of course there is a limit to how much batch processing can be done (imposed by computational resources; and by perhaps requiring a finite time-lag for real-time processing, where future observations are not available).

A simple way to take advantage of this structure is to use a block norm. We write $\|X\|_{(1,p)}$ to denote the sum of the p-norms of the rows of a $N \times L$ matrix:

$$\|X\|_{(1,p)} \triangleq \sum_{\text{rows } i=1,\ldots,N} \Big(\sum_{\text{columns } j=1,\ldots,L} |X_{i,j}|^p \Big)^{1/p}. \tag{2.7.8}$$

This is in contrast to the norm $\|A\|_{q\to p} \triangleq \sup_{z\neq 0} \|Az\|_p / \|z\|_q$. The $\|\cdot\|_{(1,2)}$ norm is also know as the *row-norm* of a matrix.

Instead of minimizing the ℓ_1 norm of a vector x from time period 1, we minimize the $\|\cdot\|_{(1,2)}$ or $\|\cdot\|_{(1,\infty)}$ norm of the matrix X where each column of X is from a different time period; this technique could also be combined with the windowing technique suggested earlier, in which case the columns of X actually come from time periods that might overlap. This is sometimes referred to as "multi-channel" processing ("channels" here are not the same notion as channels of the RMPI). Modern software such as TFOCS (Chapter 4) can handle these norms. This is discussed in the literature; for example, in [BKR09], which refers to the signal model as a "fusion frame".

2.8 Results

Using the Simulink model with non-idealities, it is possible to get an accurate idea of the limits of the system. This section discusses the robustness of the system to non-idealities, and the expected performance in the presence of a realistic level of the non-idealities.

To measure error against a known input x_0, we use the (relative) RMSE criteria already mentioned in (2.4.1)

$$\text{RMSE}(x) \triangleq \frac{\|x - x_0\|_2}{\|x_0\|_2}$$

and the signal-to-noise-and-distortion ratio SNDR (often referred to as just SNR)

$$\text{SNDR}_{\text{dB}} = 10^{-20\,\text{RMSE}(x)} \tag{2.8.1}$$

as well as the effective number of bits

$$\text{ENOB} = \frac{\text{SNDR}_{\text{dB}} - 1.76}{6.02}. \tag{2.8.2}$$

The maximum allowed input amplitude of the RMPI is 0.1 V, and this is referred to as "full-scale". The terminology dBFS reflects size relative to full-scale. For example, if a pulse is -20 dBFS, this means the amplitude is .01 V.

2.8.1 Non-idealities

The RMPI is affected by many non-idealities. A non-ideality is anything that our model does not account for. We do not discuss errors in creating the model, since those have been discussed in the calibration section. The non-idealities we consider are

- Thermal noise.
- Clock jitter at the mixers and ADCs.
- Quantization.
- Cross-talk between signal paths.
- Clipping.
- Non-linearities. (This was also previously discussed in §2.7.2.5.)

Of these, the thermal noise, jitter, and non-linearities have the biggest effect. Quantization, cross-talk, and clipping are generally not a dominant source of error. The section concludes with §2.8.1.6, which discusses all the non-idealities together.

2.8.1.1 Noise

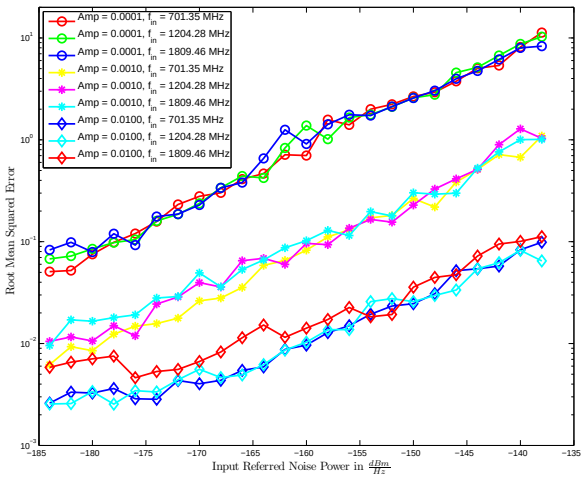


Figure 2.54: Error of reconstruction as a function of input-referred noise power η_{in}, shown for various input amplitudes A (circles, plus, and diamonds) and carrier frequencies f_{in}

Thermal noise is a major issue. Because it has a constant value regardless of the input, it affects small inputs and limits the lower range of signals that the system can reliably distinguish. In the Simulink model, three of the blocks have thermal noise, but by far the dominant source is at the LNA. For the LNA block, we treat input referred noise, which takes into account noise from different

stages of the block. In Simulink, this is modeled as iid Gaussian noise using an appropriately fine time-scale.

Figure 2.54 shows the results of different levels of LNA noise. Noise power η_{in} is measured in dBm (dB per relative to milliwatts), for 1 Hz bandwidth. The total noise power across a different bandwidth B is $\eta_{\text{in}} + 10\log_{10}(B)$.

The figure plots signals for 3 levels of amplitude and 3 frequencies (low, medium, and high). The frequency dependence is quite small. As a function of noise power, the trend in RMSE is linear. A pessimistic estimate of the noise in the CMOS chip is -164 dBm. At this level, we have just barely acceptable performance for a single input (RMSE$\simeq 0.4$) at $A = 10^{-4}$ V; the max amplitude possible is about 10^{-1} V. For an exact RMSE cutoff of .33, the single-pulse dynamic range of the system is 55 dB or equivalently 9 ENOB.

2.8.1.2 Jitter

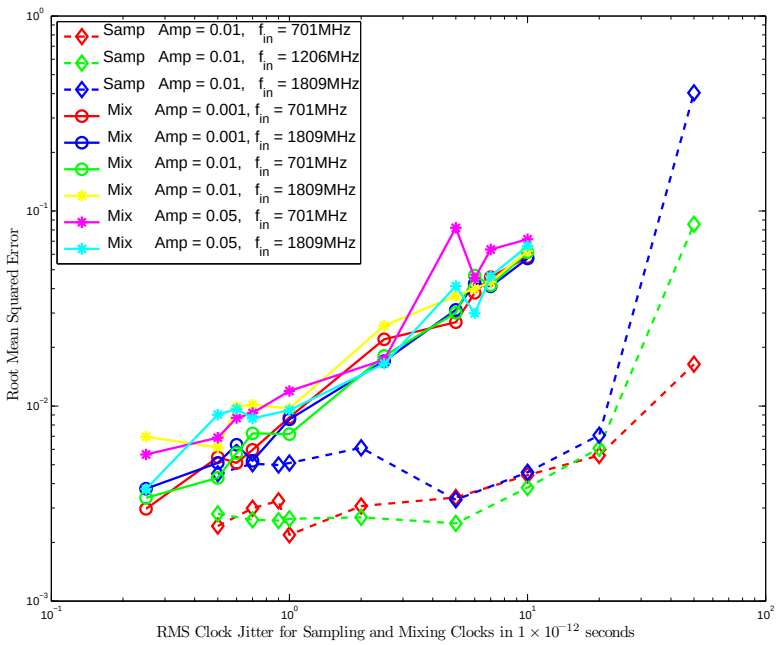


Figure 2.55: Error of reconstruction as a function of rms jitter σ. The diamonds (dashed lines) show error for *sampling* jitter at the ADC, while the circles and pluses show error for jitter at the *mixer*.

The RMPI runs off a clock which operates at f_s, and this clock determines the output of the shift registers that define the PRBS. The actual rise and fall time of the clock signal do not exactly lie on the f_s grid but are jittered slightly forward or backward in time. Let e_n be the error in time between $n\Delta T$ (when the clock *should* change) and when the clock actually changes. Similarly to thermal noise, this can be viewed as a stochastic quantity. The most commonly used figure-of-merit is "rms jitter" σ which is the standard deviation of e_n, since typically $\mathbb{E}\, e_n = 0$. In simulation, e_n is treated as an iid Gaussian variable.

The distinguishing feature of jitter is that it induces an error proportional to both the amplitude and frequency of the signal. If an input is $x(t) = A\sin(\omega t)$, then the difference between the ideal sample at $x(t)$ and a jittered sample at $x(t+e)$ is roughly $A\omega e$. Hence jitter is a non-linear effect, and is more significant at high frequencies.

Jitter also plays a role in the sampling of the ADCs. This sampling jitter is less important because it is after the integration and the signal is dominated by low frequency components; in contrast, mixing jitter is more significant because (a) it affects the PRBS which is very high frequency, so a small time error corresponds to a larger amplitude error, and (b) due to the high rate, the jitter affects every Nyquist sample, whereas sampling jitter only affects every $N_{\text{int}} = 100$ Nyquist samples. In the version 2 design, the CMOS mixer has 0.5 ps *mixing* jitter. The 12-bit ADC samplers have 2 ps sampling jitter but their bandwidth is only 300 MHz.

Figure 2.55 plots the RMSE of the reconstruction for pulses of low, medium, and high frequencies, as a function of σ. The ADC sampling jitter only has an effect for $\sigma > 20$ ps, so the this is not expected to be a limiting factor. For any $\sigma > .1$ ps, the mixing jitter has a negative effect on the reconstruction. Because of this, the RMPI design was thoroughly engineered to keep jitter as minimal as possible, and the rms jitter of 0.5 ps is state-of-the-art. At this value, jitter has a negative effect, but thermal noise and non-linearities have a larger negative effect, as will be discussed in §2.8.1.6.

2.8.1.3 Quantization

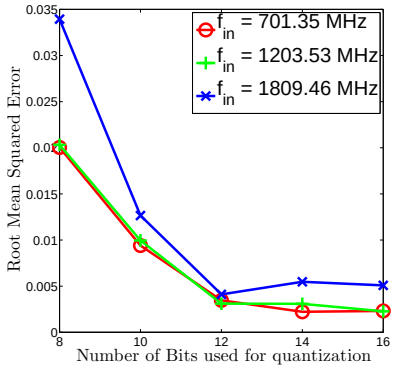


Figure 2.56: Error of reconstruction versus number of bits of quantization for amplitude $A = 0.001V$ (-40 dBFS)

The ADC samplers have finite resolution and quantize the output to only B bits. An ADC can be configured to an expected range of input which should closely match the maximum output of the system. Having set the scale of the ADC to match the largest expected values of our system, Figure 2.56 shows the effect of this quantization on a small signal input. For $B \geq 12$ bits, there is no appreciable effect of the quantization. For even smaller signals, it may be necessary to use 14 bits, assuming that thermal noise doesn't dominate the signal.

If the system is upgraded to handle even more dynamic range, than 16 or more bits may be

necessary, which should not be an issue since there are commercial 50 MHz ADC with large ENOB. An alternative would be to implement a dynamic system that changes the scaling to the ADC, but this may be difficult. The version 2 RMPI uses 12 bit ADCs, but these are off-chip and so could be upgraded easily.

2.8.1.4 Cross-talk

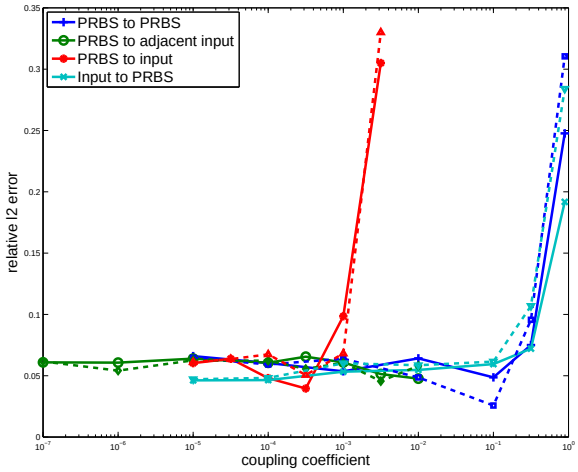


Figure 2.57: Results of the cross-talk simulations. Solid line is a 700 MHz input; dashed line is a 2.1 GHz input. Both inputs were pulses 100 ns long and amplitude .01 V (-20 dBFS).

Cross-talk is the interference caused by signals when electronic components are in close proximity. The dominant sources of cross talk are shown in Figure 2.58. To minimize cross-talk, the chip was designed with several safeguards: each channel's analog and digital supplies are not connected to one another on chip; clock distribution supplies and references are completely independent; the analog and digital grounds are isolated from one another; extensive use is made of guard rings around all blocks, subblocks, and channels; and each block is triple-welled. The worst-case coupling levels reported in the literature for mixers of this class is 10^{-2} [CMA09]. According to the simulations in Figure 2.57, this is right at the borderline level for inducing failure due to PRBS-to-input cross-talk. However, this worst-case result was in a design that didn't use triple wells or substantial guard ring structures, and we expect our coupling levels to be smaller. Since this was a worst-case value, it is not expected that cross-talk is a significant limitation in the design.

2.8.1.5 Clipping

Large amplitude signals run the risk of railing the integrator, since the integration is only designed for a limited voltage range, $\pm v_{max}$. To analyze how likely clipping is to occur, we assume Φ is the Bernoulli ± 1 model, and work in discrete space. What is the worst-type of input signal? It is a signal x_n that exactly mimics the discrete chipping sequence $c_n \subset \{-1, 1\}^N$. However this is extremely

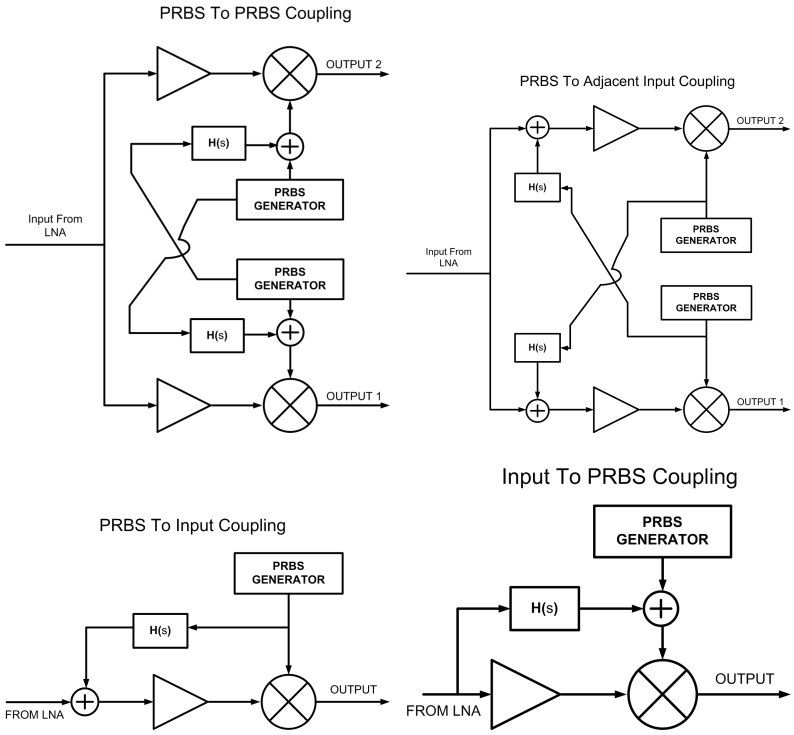


Figure 2.58: The types of cross-talk mechanisms considered in the testing show in Figure 2.57. The $H(s)$ block represents the unwanted cross-talk that might occur.

unlikely. If the chipping sequence c_n is random, then a "typical worst-case" input is actually any arbitrary signal with amplitude A_{max} and a reasonably slow frequency($f \lesssim 1$ GHz, so that the signal is approximately constant over a Nyquist sample), so without loss of generality we assume x_n is a DC signal with amplitude A_{max}. Let $S_k = \sum_{n=1}^{k} c_n$. Thus the question of clipping is: over a fixed interval $T = N\Delta T$, what is the probability that at some (possibly intermediate) time $k \leq N$, $|S_k| \geq \frac{v_{max} f_s}{A_{max}} \equiv v$? We'll break this question into three parts.

First, consider the sum at the *end* ($k = N$) of the time in question, $p \equiv \mathbb{P}(S_N \geq v)$. The sum S_N is just a shifted and scaled sum of Bernoulli random variables, so S_N is a shifted and scaled binomial random variable, which has a known CDF. For large N, this is excellently approximated by the CDF of a Gaussian variable with $\mu = N/2$ and $\sigma = \sqrt{N/4}$.

At first glance, finding $\mathbb{P}(S_k \geq v,\, k \leq N)$ appears difficult. Fortunately there is a well-known clever trick (thanks to Yaniv Plan for pointing this out). Consider an intermediate sum $S_k = v$, and look at the set of all possible random walks of length $N - k$ that start at this point. Then the expected value at time N of these extensions is still v since a random walk has mean 0. In fact, it has median 0 as well, and half of the extensions will end up with value greater than v, and half will end up with value less than v (note: this reasoning requires that $N - k$ is odd, so that none of the extensions can end up with value exactly v).

Thus if we record the number of sums S_N which exceed v, this accounts for half of the sums

which had reached v at an earlier time. Thus $\mathbb{P}(S_k \geq v,\, k \leq N) = 2p$. This reasoning is a little imprecise since it actually depends on whether v (assumed to be an integer for the moment) has the same parity as N, but for large N and values of v that are not too close to n, the error is small. See Figure 2.59 for experimental evidence.

As the final step we account for negative amplitudes, so $p_{clip} = \mathbb{P}(|S_k| \geq v,\, k \leq n) = 4p$.

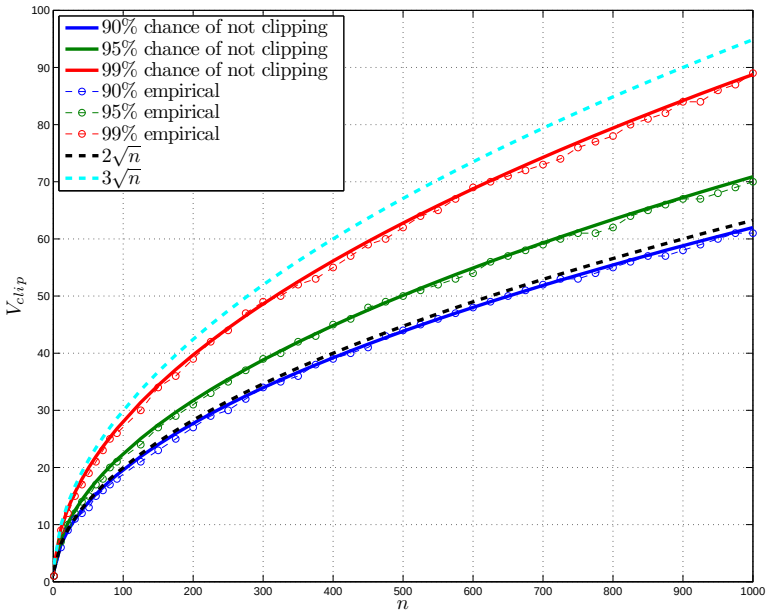


Figure 2.59: Simulation to verify the chance of a random walk clipping. The solid lines are the predictions from using the inverse CDF of a normal distribution, and the circles are the empirical values from 10,000 Monte Carlo simulations, which show excellent agreement, thus validation the $2p$ argument. Furthermore, the lines are bounded between $2\sqrt{n}$ and $3\sqrt{n}$, which provides a useful heuristic. To use the chart: suppose $v_{clip} = 70$ and we want to be 99% sure that a maximum amplitude signal will not clip. Then we must reset the integrator every $n = 600$ Nyquist units. (Note: for a practical calculation, v_{clip} needs to be properly scaled; see (2.8.3).)

Let us calculate the scaling for the version 2 RMPI chip. Suppose the input is a low-frequency sine wave $x(t)$ of amplitude A. If the mixing sequence c is random then almost all inputs look the same, and what really matters is the absolute integral $\int |x(t)|$, so this sine wave input can be modeled as a DC input with amplitude $2A/\pi$. The max acceptable amplitude input is 0.1 V_{pp}, and the LNA gain is 18 dB (this is actually variable; it can be set from 12 to 18 dB), so the input voltage is $v_{in} = 10^{18/20} 0.1 \times 2/\pi \simeq .8 \times (2/\pi)$ V. A transconductor of 20 mS (mS $= 10^{-3}$ Amps/volt) converts the voltage signal into a current signal. The current is attenuated by a factor of 2 in the mixing stage, and the path is differential so we (temporarily) attenuate by another factor of 2 to account for this. Thus the input current to the integrator is $I_{in} = (2/\pi)(.8 \text{ V})(20 \cdot 10^{-3} \text{ Amps/V})/4 = (2/\pi)4 \cdot 10^{-3}$ Amps. To find the output voltage of the integrator (which is approximately just a capacitor), we use the capacitor equation:

$$v_{out}(\tau) = \frac{1}{C} \int_0^\tau I_{in}(t) dt$$

where the capacitance is $C = 40$ pF. For a DC input and discrete chipping sequence, the integral becomes a sum, with time step $\Delta T = \frac{1}{5\text{ GHz}}$. The clipping voltage is $v_{clip} = 4v_{out}$ because we now combine the two differential paths and also consider peak-to-peak voltage. The supply voltage to the chip is 1.5 V, so the peak-to-peak clipping limit is $v_{clip} = 3$ V. Combining all these scalings, we can write down an equation:

$$\begin{aligned} 3\text{V} = v_{clip} = 4v_{out} &= \frac{4}{C}\sum_{k=0}^{n-1} c_k I_{in} \Delta T \\ &= \left(\sum_{k=0}^{n-1} c_k\right) \frac{4}{40 \cdot 10^{-12}} \frac{1}{5 \cdot 10^9} (2/\pi) 4 \cdot 10^{-3} \\ &= \left(\sum_{k=0}^{n-1} c_k\right) \frac{4}{25\pi} \end{aligned} \tag{2.8.3}$$

which means the system will clip with

$$\left|\sum_{k=0}^{n-1} c_k\right| = \frac{75\pi}{4} = 58.9. \tag{2.8.4}$$

The integration period T_{ADC} is 100 Nyquist samples, so $n = 100$ per sample, and thus it is *possible* to clip in a single integration period, which is catastrophic. However, it is extremely *unlikely* to clip in a single sample (if the assumptions hold; to be more rigorous, we would need to impose a prior on the input signals). Using Figure 2.59, then there's a 99% chance the signal will not clip until $n \gtrsim 420$, and a 90% chance the signal will not clip until $n \gtrsim 850$. Thus with a max amplitude signal, and the highest gain setting of the LNA, clipping should not be an issue if we restrict to 4 to 9 integration windows T_{ADC}. If a signal has half the max amplitude, then we need the random walk to reach 118, which is extremely unlikely to occur if $n \leq 1000$.

If the overall experiment is significantly longer than $1000\Delta T$ (which it usually is, even though we typically analyze chunks of size 1024 or so), then for a pure tone input, the system will eventually rail (in fact, if we run the system forever, this occurs with probability 1), so we would periodically reset the integrators; furthermore, the fact that the PRBS sequence c is in periodic would be significant. Such considerations are beyond the scope of this work. For inputs that are pulses at some repetition rate, the system will periodically have no input, and the natural discharge of the capacitors will prevent railing.

The system can be told to reset at will, but this involves the loss (or at least corruption) of a sample. Since we do not expect railing to be a problem for short signals, we have not simulated the effects of resetting when running worst-case simulations, but this issue would be important for future chips that operate robustly. Possible alternative designs may be beneficial, such as having two samplers that alternate, so each capacitor can be reset every integration period. The Northrop

Grumman InP chip uses a system like this. The downside of this design is slightly more complexity and power consumption.

2.8.1.6 Combining non-idealities

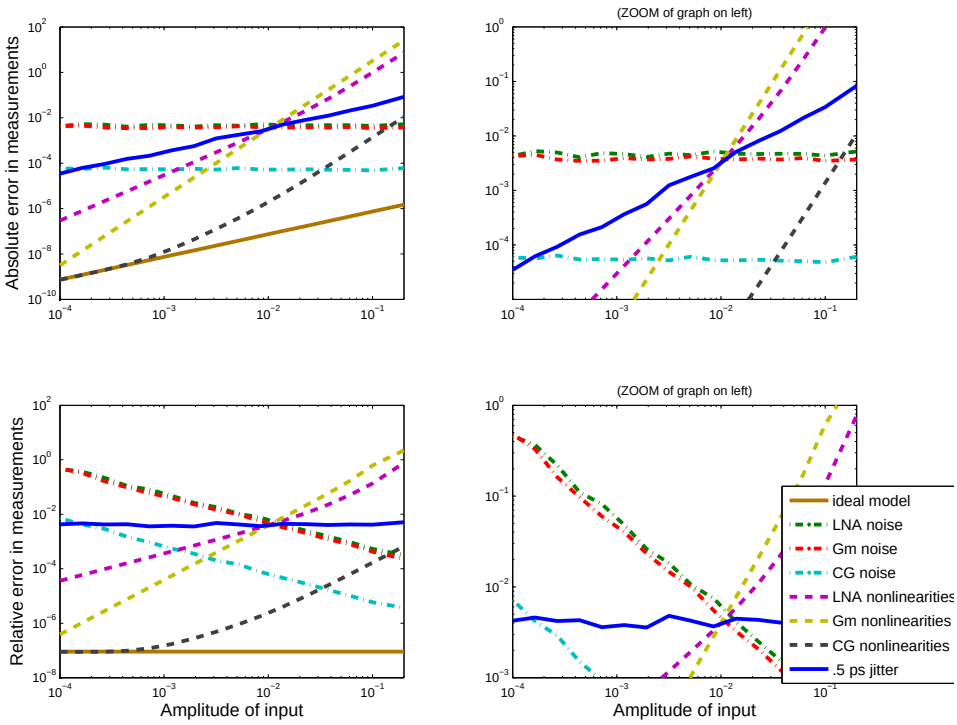


Figure 2.60: Various non-idealities. The four plots show the same data but in different ways: the top row shows error in absolute terms, while the bottom row shows error in relative term. The right column is a zoom of the left column. The "ideal" model uses a Φ matrix generated from a (noiseless) Simulink calibration. The other measurements are from the Simulink model with various non-idealities turned on. This shows at what amplitudes each non-linearity has an effect. The thermal noise has an absolute effect (constant on the top row), while jitter has a constant relative effect (constant on bottom row) since it induces error proportional to the amplitude of the input. See also Figure 2.61.

The major non-idealities affect the system in different ways.

- Thermal noise is constant, so it affects *small* signals.
- Jitter error is proportional to the amplitude of a signal, so it affects *all* signals equally.
- Non-linear effects are only an issue when the amplitude is *large*.

Jitter and non-linearities make two-signal recovery difficult. If one signal is much smaller than the other signal, the smaller signal still sees the large jitter errors induced by the large signal. The small signal is also desensitized or even blocked due to the non-linear effects of the large signal [Raz97]. For this reason, low-jitter and a high linearity range are extremely crucial for good performance.

For the simpler case of just one signal recovery, which of these non-idealities is limiting? To test this, we calibrate an accurate model Φ using Simulink to provide the black-box system S. The

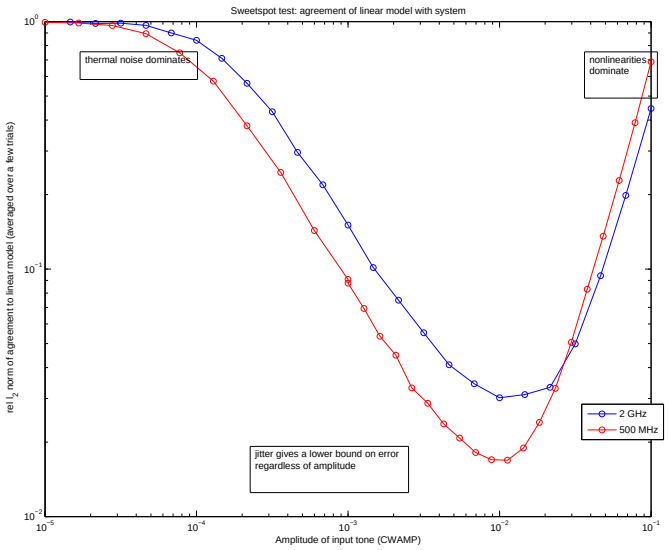


Figure 2.61: "Sweetspot test." The amplitude 10^{-2} (-20 dBFS) has the least measurement error when we include the non-linearities. See also Figure 2.60.

agreement $\|\Phi x - S(x)\|_2 / \|S(x)\|_2$ is very good, typically on the order of 10^{-7} (see the brown line in Figure 2.60). It is not exactly 0 because S is based on ODE solvers, whereas Φx assumes that x is band-limited and is computed as a finite-precision matrix-vector multiply, so both methods have a small amount of error. We now consider $S_{\text{non-ideal}}$, which is the Simulink model using a non-ideality. Figure 2.60 plots

$$\|\Phi x - S_{\text{non-ideal}}(x)\|_2 \quad \text{(top plots)}, \quad \text{and} \quad \|\Phi x - S_{\text{non-ideal}}(x)\|_2 / \|S(x)\|_2 \quad \text{(bottom plots)}.$$

The $S_{\text{non-ideal}}$ model used exactly one type of non-ideality for each line shown in the figure, which gives an idea of the relative importance of each non-ideality.

The top row shows the absolute error, as a function of input amplitude (recall that 0.1 is full-scale input). Thermal noise is independent of amplitude, so it shows up as a flat line on this plot. The thermal noise at the LNA and transconductor (Gm) are about equal; the noise at the common gate filter is not significant. Non-linear effects from the LNA and Gm play a role at inputs of -20 dBFS and higher.

The bottom row shows relative error, and hence the error due to jitter is a flat line. Jitter was tested at several RMS values, and we only show the $\sigma = 0.5$ ps. For larger values of RMS jitter, the jitter will become a dominant source of error, but at this value it is not a limiting factor (though with non-linearity correction, it becomes a slightly limiting factor for a small range of amplitudes). However, since jitter affects two-pulse recovery, there is no "safe" value of σ. The value $\sigma = 0.5$ ps is chosen because it is the lowest practical value that we can design.

If we allow $S_{\text{non-ideal}}$ to use *all* non-idealities at once, then we get the plot in Figure 2.61. The figure shows relative measurement agreement. Two different signals are shown, one at low frequency and one at high frequency, since jitter affects the high frequency signal more. The plot shows that the amplitude 0.01 (which is -20 dBFS) has the least error, since this is large enough to ignore thermal noise but also small enough not to have significant non-linearity problems. We refer to this amplitude as the "sweet spot" of the system. This also indicates that the system has a maximum dynamic range of about 60 dB, since for signals greater than 0 dBFS or smaller than -60 dBFS the measurements have almost 100% relative error, and thus it is unreasonable to expect any recovery algorithm to recover the input. This is not an exact rule, since the reconstruction is non-linear and can denoise in some cases, but it is a reasonable rule-of-thumb.

2.8.2 Simulation results

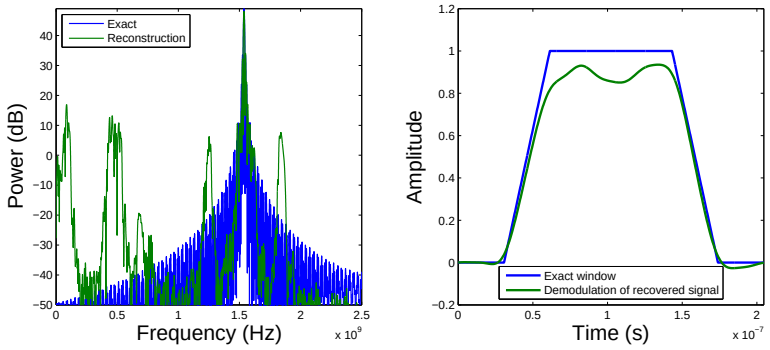


Figure 2.62: Reconstruction using data from SPICE simulation for version 1 design. The matrix Φ was modeled as an ideal single-pole integrator with the pole at 44.05 MHz. Agreement is crude, but not bad considering the non-calibrated Φ matrix. Error in carrier frequency estimation is 0.1 MHz.

The results here are mainly from detailed Simulink simulations, using accurate levels of non-idealities. Less systematic results show that we can recover from SPICE simulations as well.

Figure 2.62 shows a reconstruction from SPICE data from the version 1 model in December 2008. The Φ matrix uses ± 1 entries and is not calibrated, so the performance is quite remarkable given

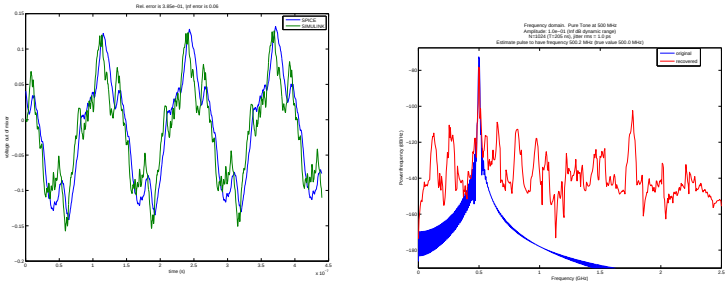


Figure 2.63: May 2010, SPICE simulation, using a Simulink calibrated Φ matrix to recover. Left is the integrator output (before ADC sampling); right is a reconstruction. Input was a pure tone at 500 MHz.

the lack of calibration. Figure 2.63 shows results from the version 2 model. The Φ matrix is from a calibrated Simulink model. Results would be even better if Φ were calibrated from a SPICE model, but this is time-prohibitive.

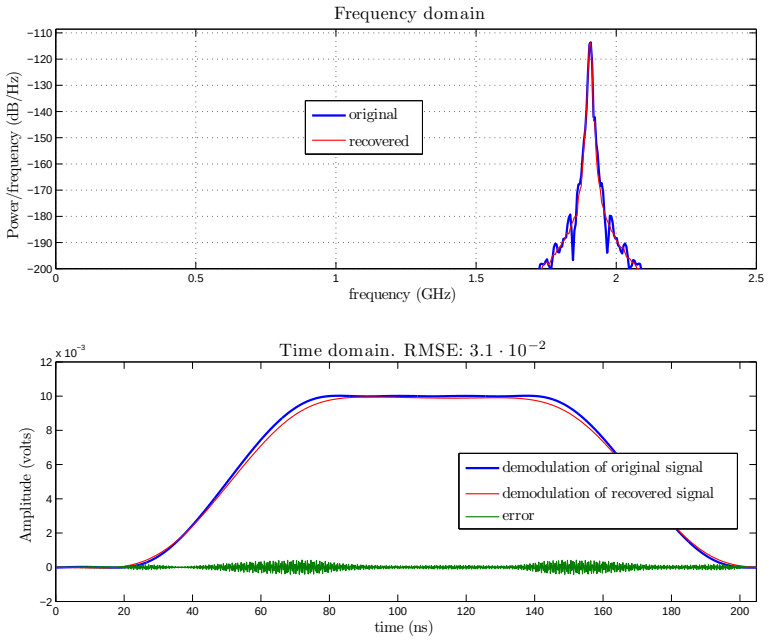


Figure 2.64: An input test signal and its reconstruction. Carrier frequency is $f_{in} = 1907.2$ MHz. Simulation (via Simulink) includes all non-idealities, e.g., jitter (rms .5 ps), noise, non-linearities. Amplitude is 10^{-2} (-20 dBFS), and observation time is $T = 205$ ns ($N = 1024$). The recovered signal estimates the correct frequency to within 0.1 MHz. The relative error $\|\hat{x} - x\|_2/\|x\|_2$ is $3.1 \cdot 10^{-2}$. The time-of-arrival estimate (defined as when the signal first reaches 10% of its max amplitude) was accurate to 1.2 ns, and the duration estimate was accurate to 2.6 ns.

Figure 2.64 shows a typical recovery from noisy Simulink data, using a calibrated Φ. Because of the calibration, the recovery is excellent.

2.8.2.1 Single pulse

To test the recovery of a single pulse, there are three dominant parameters: the amplitude, the frequency, and the pulse duration.

Frequency. The effective instantaneous band-width (EIBW) of the system is a measure of the range of frequencies that can be accurately recovered. Figure 2.65 shows the ENOB of recovery on a 200 ns pulse, as a function of carrier frequency. The EIBW is almost the full 2.5 GHz, though not quite since the pulse window induces spectral spreading, so carriers at 2.45 GHz will cause much of the narrowband content to appear at greater than 2.5 GHz. The hardware is not designed for more then 2.5 GHz, and the recovery algorithms use a Nyquist grid so the post-processing is not suitable for signals past 2.5 GHz either.

The hardware version 2 RMPI has been designed with less bandwidth than in the simulated

versions. In particular, the frequency response is very small below 300 MHz, and a more accurate estimate of the EIBW is 2 to 2.2 GHz.

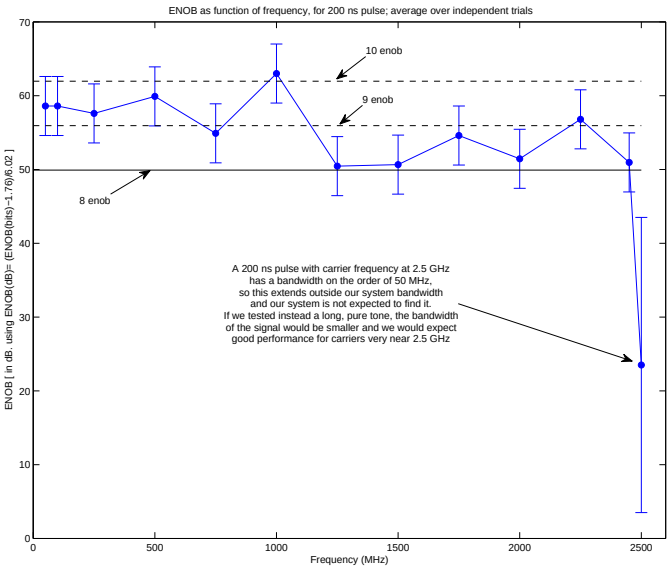


Figure 2.65: ENOB as a function of frequency for 200 ns pulses. For all frequencies, the amplitude is at the sweet-spot amplitude.

Single-pulse dynamic range. To consider the limits of the system, we test the largest and smallest signals it can handle. Recovery is better if the pulses are longer. Figures 2.66 and 2.67 show relative MSE as a function of amplitude and pulse duration, at carrier frequencies 700 MHz and 2100 MHz, respectively. The pulse has a 25 ns rise and fall time, and measurements are taken from a Simulink model with realistic values for all the non-idealities. The recovery software was set to assume a bandwidth from 500 MHz to 2500 MHz (this slightly helps recovery; the restriction to this bandwidth is imposed by adding extra rows to Φ that correspond to entries from the DFT matrix, and adding corresponding 0 samples).

For durations of less than 100 ns, recovery is poor since only a few samples are taken of the system. We focus on 200 ns pulses since this is the shortest reasonable radar pulse our system is designed to recover. At 700 MHz, Figure 2.66 shows that the MSE is less than 10% for amplitudes between 0 and -55 dBFS. Thus using MSE < 0.1 as the cutoff, the dynamic range is 55 dB. Converting from dB to ENOB, this is 8.8 ENOB. For 100 ns pulses, the dynamic range is 50 dB, or about 8.0 ENOB.

At higher frequencies, such as shown in Figure 2.67, results are slightly worse and ENOB is only 8.0 for 200 ns pulses. The frequency- and duration-dependent results may depend slightly on the parameters used for reconstruction, e.g., the parameters of the Gabor dictionary.

Fixing the duration at 200 ns, Figure 2.68 shows the effect of frequency. For each frequency point, the actual frequency is chosen randomly in a small interval, so that on-grid effects are not

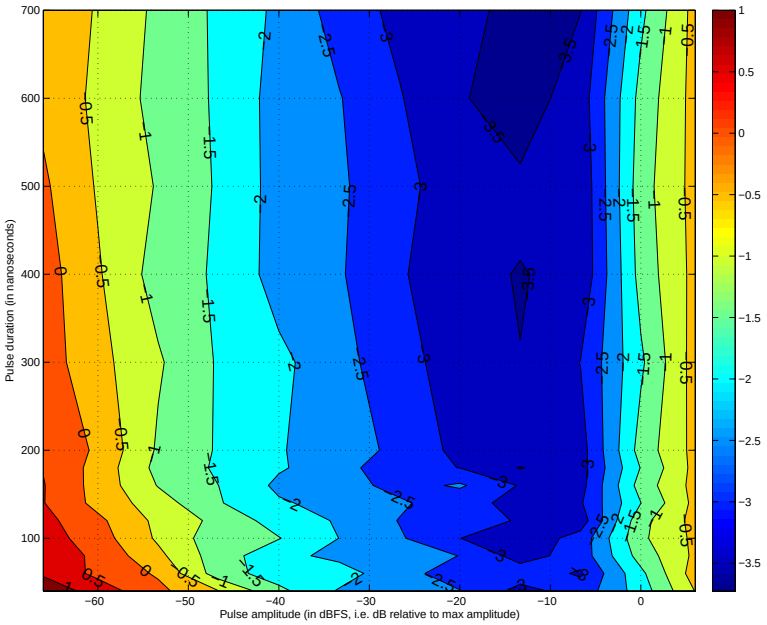


Figure 2.66: Plotting relative MSE (elevation) as function of pulse duration (vertical axis) and amplitude (horizontal axis). Contours shown $\log_{10}$ scale. 701.4 MHz carrier. Results improve for longer duration, as expected.

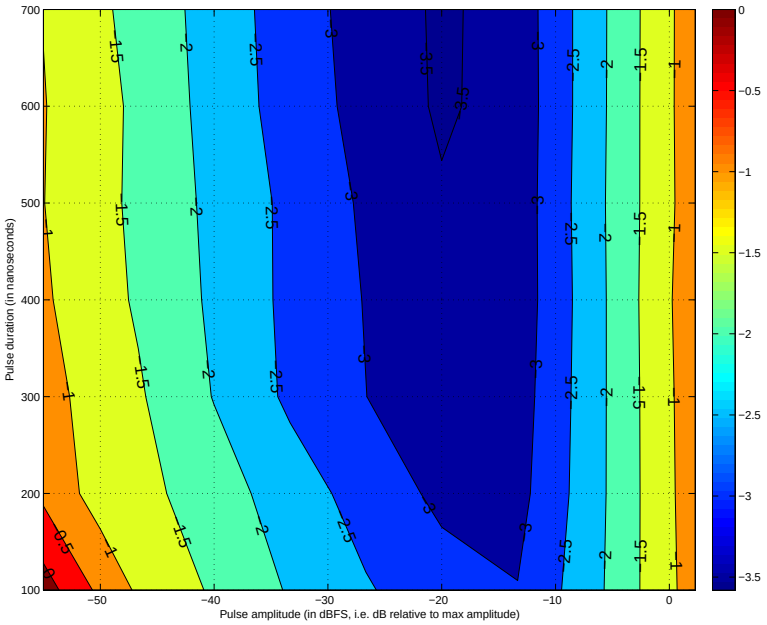


Figure 2.67: Same plot as Figure 2.66 except with 2105.1 MHz carrier instead of 700 MHz carrier (chosen so that it is not on-grid), and showing the median results over 10 independent runs. Results are slightly poorer at this high frequency.

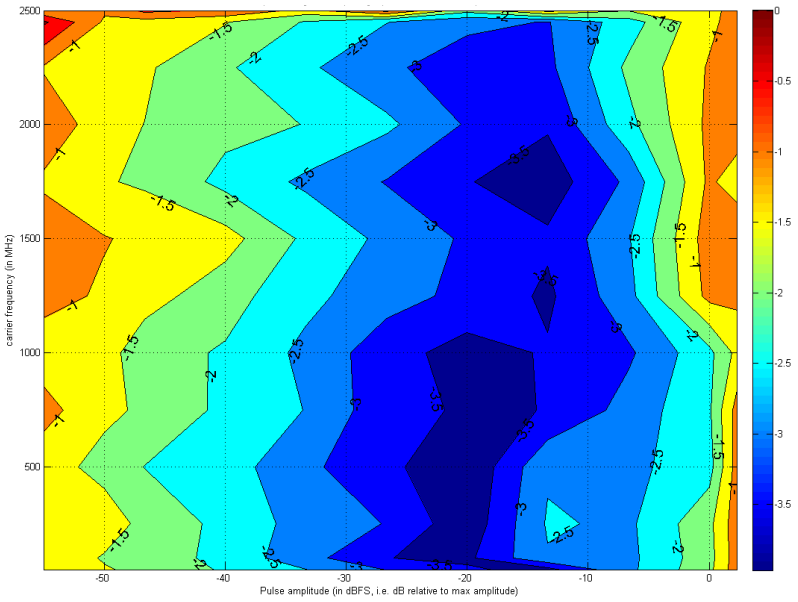


Figure 2.68: From realistic Simulink simulations in spring 2010. Error in relative MSE as a function of frequency. Recovery as a function of frequency and amplitude, for a 200 ns pulse.

significant. Phase is random as well, and pulse rise and fall time are 25 ns. Recovery is generally better at lower frequencies, which is not surprising since the PRBS spectrum is slightly larger at lower frequencies, so low frequency signals are affected less by noise; furthermore, clock jitter at the mixer is more significant for high frequency tones.

2.8.2.2 Two pulses

We consider two types of two-pulse tests: keeping both pulses at the same amplitude, or fixing one amplitude and varying the other pulse amplitude. For both types of tests, the pulses overlap 80% or more.

Equal amplitude. For both pulses with equal amplitudes, the results are quite similar to single-pulse tests. With 200 ns pulses, the relative MSE is less than 10% for amplitudes in the range -6 dBFS to -54 dBFS. The upper limit is slightly worse than the single-pulse case since there is more energy in two pulses, so non-linearity is a larger effect. Using non-linearity correction, the upper limit can be pushed to -3 dBFS, so there is 51 dB dynamic range.

The basic conclusion is that two pulses of the *same* amplitude are not more difficult to recover than a single pulse. In this case, sparsity is not a limiting factor, rather non-idealities are.

The Figure 2.69 shows a typical recovery. This is an extreme case using two very short pulses. The amplitude of both is -20 dBFS which is optimal according to the "sweet spot" calculations in Figure 2.61

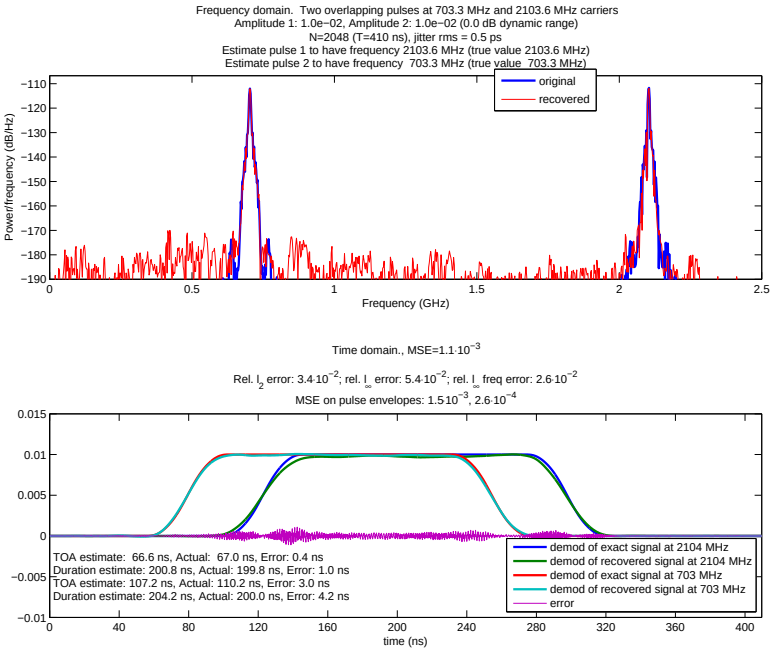


Figure 2.69: Two pulses in realistic simulation, each 80 ns long with 80% overlap. Note that the pulse codewords (carrier frequency, time of arrival, and duration) are estimated very accurately.

Different amplitudes. If two signals overlap in time and have very different amplitudes, recovery of the small tone is quite difficult. Using noisy Simulink simulations, we estimate that the system can recovery long pulses with up to 30 dB difference in amplitude. Figure 2.70 shows an example reconstruction of two pulses with 10 dB difference in amplitude. Clearly, one limiting factor is that large pulses induce much jitter error, which will swamp small signals.

Figure 2.71 shows two concurrent 200 ns pulses that have been recovered from *noiseless* measurements, using a model of the 300 kHz single-pole integrator. The pulses are 40 dB different in amplitude; any greater difference and the small pulse is not recoverable (for this particular instance; with a different input, recovery beyond 40 dB is sometimes possible). Because this is noiseless, it suggest a limit on the recovery procedures of the system. It is possible that better choices of a dictionary or new recovery techniques may improve the dynamic range considerably; for example, slight changes in the Gabor dictionary increase the maximum dynamic range from 30 to 40 dB, so it is reasonable that further improvements may push this to 50 or 60 db. The dynamic range is also affected by variables such as how close the carrier frequency of the two signals are and the chip repetition rate N_{chip}.

Hopefully future results will improve this pulse-on-pulse dynamic range. We note that this problem is quite difficult, and modern radar systems generally ignore any pulses that arrive concurrently. Other issues, such as jitter caused by large signals, are applicable to any ADC system that views two pulses.

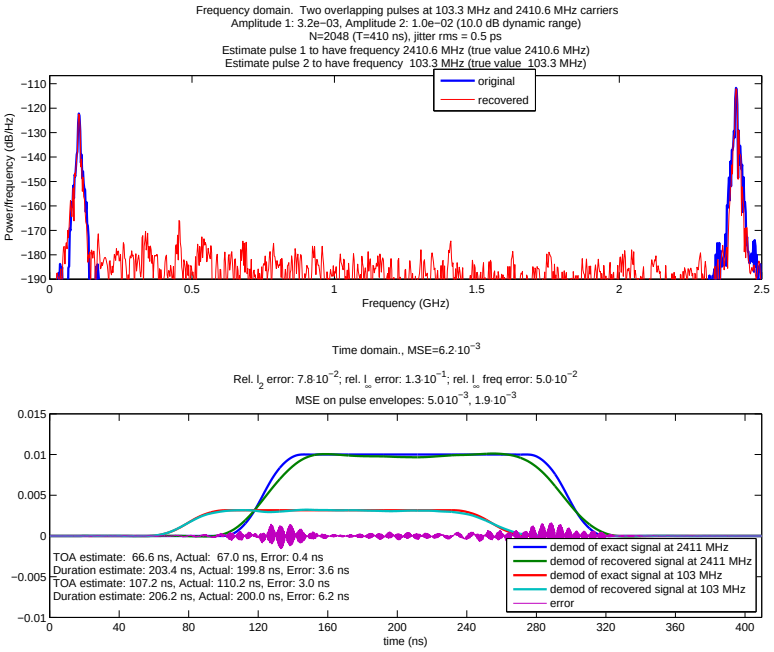


Figure 2.70: Like Figure 2.69 but with pulses of different amplitudes, which is much more difficult

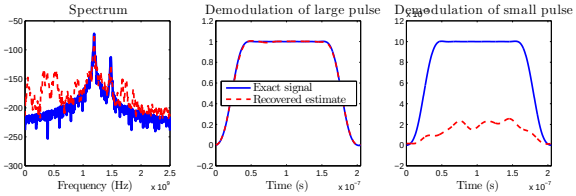


Figure 2.71: Two overlapping 200 ns pulses with 40 dB difference in magnitude. Noiseless setting. With any more dynamic range, recovery is poor. The range is better if we consider longer pulses.

2.8.2.3 Comparison

The single pulse ENOB for the RMPI varies between 8 and 8.8 ENOB. The version in simulation has 2.5 GHz EIBW, and consumes $p = 3.6$ watts including the ADCs (but not including the FPGA or computer). Using Walden's figure-of-merit (FOM) [Wal99], and with 8 ENOB,

$$F = \frac{2^{\text{ENOB}}\text{EIBW}}{p} = 1.76 \cdot 10^{11}.$$

The best ADCs with 1 GHz EIBW have only 7 ENOB and consume about 3 watts, according to [LRRB05]. Thus the RMPI has improved bandwidth, improved ENOB, and about the same power consumption. Many aspects of the RMPI are not optimized for maximizing the FOM, since we placed extra value on reconfigurability. For example, the shift-registers for the PRBS are programmable, but require high amounts of power. In short, the RMPI is not a tweaked system but rather a prototype, and therefore the fact that it already exceeds the state-of-the-art is encouraging.

A channelized conventional receiver can have a higher FOM than the RMPI, but the FOM is

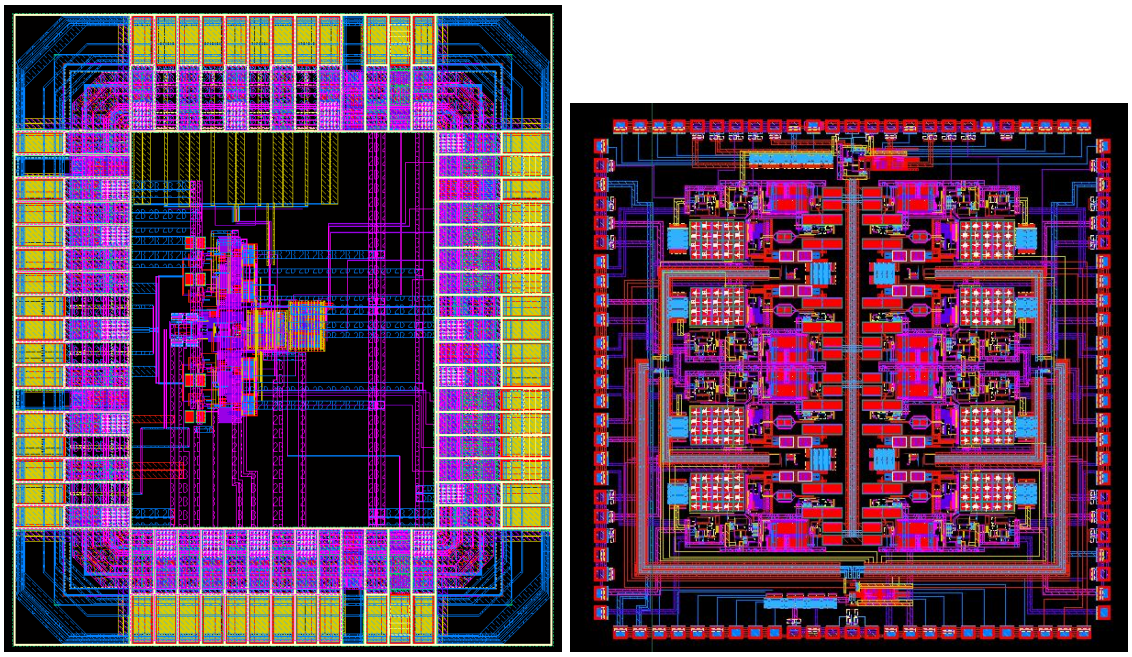


Figure 2.72: Layouts of the version 1 (left) and version 2 (right) Caltech CMOS RMPI chips

not the only meaningful measure. For example, a channelized receiver covering 2.5 GHz EIBW will consume 50 to 200 watts, in contrast to the RMPI which consumes 3.6 watts for the same EIBW. For a low-power environment, if computations can be done off-line, then the RMPI system is quite advantageous.

2.8.3 Hardware

2.8.3.1 NG InP version

The Northrop Grumman indium-phosphide chip has been returned from fabrication and, as of March 2011, is in the process of being tested and calibrated. This design uses 4 channels, each sampling at 100 MHz, as opposed to the 8 channel 50 MHz design of the Caltech CMOS design. The chief designer was Dr. Eric Nakamura from Northrop Grumman Space Technologies. The Caltech team served to consult on their design, and shared hardware ideas as well as higher-level ideas. In particular, we ran an analysis of their initial integrator filters and made improvements. We also found that their LFSR PRBS sequences were under-performing, which motivated using Gold codes to get better statistical properties in the PRBS. Another test (see §2.5.2.3) found that the Gold codes needed to be of a minimal length in order to guarantee good reconstructions.

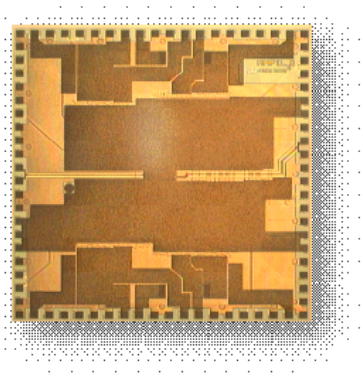


Figure 2.73: Die photo of the NG InP 4-channel RMPI

2.8.3.2 Version 1

The Caltech MICS lab, run by Professor Azita Emami and with head designer Juhwan Yoo, designed the Caltech CMOS chip. From 2008 to 2009, a "version 1" chip was designed and fabricated (see Figures 2.72 and 1.1), arriving back for testing in Spring 2009. Unfortunately, the IBM design toolkit failed to catch a short circuit in the bypass capacitors, and the chip was manufactured with a short. To get around this, each chip was manually cut to stop the short, but this prevented the circuit from working exactly as intended (the bias points were affected) and required large amounts of power to run the circuit (5 watts); overheating was a major issue, and this changed the chips' performance significantly.

This issue prevented detailed performance analysis (although some components, like the LNA and the digital logic system, were verified as working correctly). Without the shorts, the expected power draw was 700 mW, the dynamic range of the chip itself was 45 to 50 dB, operating from 0.1 to 3 GHz. The die area of the chip was 2 mm $\times$ 2 mm. Signals were successfully run through the analog signal path, but at low gain, and no reconstructions were made.

Nonetheless, the chip was still useful. The data acquisition hardware was built, including the chip's PC board and the data-acquisition FPGA. The design and failures of the chip informed the design of the "version 2" chip. Furthermore, using a detailed SPICE simulation, it was demonstrated that the version 1 chip correctly recovers a pulse; see Figure 2.62.

2.8.3.3 Version 2

The second chip (see Figure 2.72) was designed in 2009 and 2010, and arrived back from fabrication in fall 2010. All major components were upgraded. One of the biggest changes was the design of the integrating filter. Simulations predicted that a second-order filter would work, as long as the second pole was not too close to the first, allowing a few hundred MHz of first-order behavior. Allowing this second pole means that the first pole can be situated much closer to DC, and this has an enormous

beneficial effect. The version 1 integrator's first pole was at 44 MHz, while the version 2 integrator's first pole is 300 KHz; see Table 2.4.

Like version 1, the chip was fabricated in IBM's 90 nm CMOS9SF `06_02_00_LB` process. It consumes 830 mW of power (see Table 2.5 for a breakdown), and has a bandwidth of 2.5 GHz. It has a full integrated RF and baseband signal path, so it functions as both a receiver and an ADC. All controls are fully digitally programmable, so the chip sequence can be controlled by the FPGA, which is in turn controlled by a standard computer. Each channel's supply domain is completely isolated. An extra ninth "test" channel was built into the chip, with internal nodes routed out to pads to allow testing. All block biases included programmable power up and down circuitry, allowing for considerable testing capabilities including aggressor/victim cross-talk tests. The test channel also incorporated a replica clock distribution, and digital shift register nodes routed out to pads; this allows characterization of noise and jitter.

The version 2 hardware is currently being calibrated, so the RMPI has not yet been used to recover real pulses, but this is only a few days or weeks away, and will be reported in upcoming papers.

Below we describe some specific details. This section was written primarily by Juhwan Yoo and Azita Emami. Dynamic range is described as a function of the measurements, and is not the same as the dynamic range achieved via reconstruction. Detailed power consumption figures are in Table 2.5.

Due to the large bandwidth requirement of the RMPI receiver, the noise floor is relatively high, and this effect cannot be mitigated due to the fundamental nature of its source, thus limiting receiver sensitivity. With respect to non-linearity, the RMPI is similarly constrained by the required speed of operation due to the fact that the process technology that is required to achieve the necessary broadband operation places a hard constraint on the supply voltages which can be used to power the receiver. Thus, as in the case of the version 1 design, when using a voltage-mode signal propagation methodology, nonlinearity plays a significant role due to an analog design's fundamental reliance on operating receivers around linearized operating points. These two constraints limited the A2I version 1 design to approximately 40–45 dB of dynamic range when operating the signal path with 1.8 V and 2.5 V supplies.

In the version 2 design, in order to mitigate the effects of non-linearity induced by large voltage amplitude operation, a current domain approach is used. Most of the RF signal processing path is designed to handle very high currents while maintaining extremely low-impedance at each node, thus limiting the voltage swing at each node. The current is then converted back to a voltage signal in the integrating stage using transimpedance amplifier-based integrators with very high-loop-gain feedback. The high-loop gain, along with the use of current-mode propagation of the RF signal, mitigated the effects of non-linearity. This allowed the achievement of dynamic range of approximately 55 dB.

Component	Power consumption per units	Quantity	Total Power
LNA/Balun	24 mW	1	24 mW
Transconductor (Gm)	18 mW	8	144 mW
Common Gate TIA	3 mW	8	24 mW
Integrator	12 mW	8	96 mW
Mixer	0.3 mW	8	2.4 mW
Output buffers	30 mW	8	240 mW
Shift registers	30 mW	8	240 mW
Clock distribution	NA	NA	60 mW
ADC (12 bit AD9235)	350 mW (worst-case conditions)	8	2.8 W
Total w/o ADCs			830 mW
Total w/ ADCs			3.63 W

Table 2.5: Power consumption at 1.5 V

The clock distribution is a vital part of the RMPI system. This is due to the fact that it is used to clock the high-speed shift registers which store the PRBS sequence with which the input signal is mixed. The Simulink simulations conducted indicate that the system dynamic range is particularly sensitive to the jitter performance of the system clock. Thus the clock distribution uses a low amplitude sinusoid as opposed to an inverter-based clock-tree, and then performs a CML-to-CMOS level conversion locally at each channel thus achieving very minimal duty-cycle distortion as well as low jitter. The analog distribution is designed to utilize current-mode open drain transmitters and transimpedance amplifier-based receivers with 50 Ω matched transmission lines. Several repeaters are inserted to maintain signal integrity and a H-tree type symmetric clock distribution is used.

2.9 Recommendations

We summarize the important findings; for future directions, see Chapter 5.

Integrator. The integrator should have a very low first pole, at least under 50 MHz, but preferably under 1 MHz, as we have done with the version 2 design. The outer poles are less important, and we set ours at 300 MHz, but it may be possible to bring this down even lower to about 100 MHz.

Channels. We have found that more channels, and using stagger, is beneficial. But too many channels complicates the design and reconstruction (more lost samples due to boundary effects). The NG design uses 4, and the Caltech design uses 8. We recommend between 8 and 12 channels as a good trade off.

Chip sequence. This is quite important. Many errors can be traced to chip sequences with short repetition times N_{chip}. It appears that periods greater than about 60 work, and ours is set

at length 128 to be conservative. However, it may be wise to move this even higher, e.g., 256 or 512. One issue which is not fully determined is how N_{chip} and the integrator period N_{int} should be related; specifically, should they be co-prime? It seems that co-prime numbers will give the best reconstruction problems. The difficulty with this is that the Φ matrix is not fully characterized until it has $N_{\text{min}} = N_{\text{chip}}N_{\text{int}}$ columns, since there are no repetitions; hence, calibration is much slower. For the current design, $N_{\text{chip}} = 128$ and $N_{\text{int}} = 100$ so the system is fully characterized by $N = 3200$ columns, which is reasonable. We strongly advise against setting one of N_{chip} or N_{int} to be a multiple of the other.

Reset. Reset is not too much of an issue according to the analysis. However, a design that alternates between integrators and allows a reset every period may be worth the power, especially for a robust real-time system. In our academic setting, this appears to be unnecessary.

Mixer. Mixer jitter is significant. It took much redesign to get the design to have less than 0.5 ps rms jitter. We advise other systems to also reduce rms jitter to under 1 ps, since otherwise it will kill dynamic range when a large and small signal are both present.

ADC. If the ADCs have more than about 12 bits, there is no noticeable effect for even small signals, assuming that the ADCs are calibrated to have full range with large signals. The RMPI uses 12-bit ADC but these can be switched out for 14-bit ADCs if necessary. We recommend implementing a sample-and-hold block (on-chip) before the ADC (off-chip) so that the ADC does not need a highly accurate clock.

Recovery. The matched filter techniques are quite promising, since they are fast and only attempt to estimate a few parameters. For recovery of single tones, they are recommended. For complex signal environments, we recommend using reweighted ℓ_1 analysis with a very redundant Gabor dictionary.