Alternative number representations for robust analog-to-digital conversion

Özgür Yılmaz

University of British Columbia

May 29, 2008

Joint work with:

Theory: Ingrid Daubechies, Sinan Güntürk, Yang Wang

Implementation: Peter Vautour, Matt Yedlin

Analog-to-digital (A/D)conversion

Inherently analog signals: Speech, high quality audio, images, video, etc.
Objective: Represent an "analog signal" (takes its values in a continuous set) by finitely many bits=: 'quantization'

Analog-to-digital (A/D)conversion

Inherently analog signals: Speech, high quality audio, images, video, etc.
Objective: Represent an "analog signal" (takes its values in a continuous set) by finitely many bits=: 'quantization'

How is this done - a natural approach

Let $x \in [0,1]$, and $x_N :=$ N-bit truncation of the standard binary (base-2) representation of x,

$$x_N = \sum_{n=1}^{N} b_n 2^{-n}, \quad b_n \in \{0,1\}.$$

Then:

1. $|x - x_N| \leq 2^{-N}$
2. $(b_1, b_2, \ldots, b_N)$ provide an N-bit quantization of x with the *accuracy* of 2^{-N} (essentially optimal in rate-distortion sense).

Example ctd.

Next: can we compute the bits b_n on an analog circuit?

Successive approximation

Let $x_0 := 0$ and define $u_n := 2^n(x - x_n)$ for $n \geq 0$. Then

$$
\begin{aligned}
u_n &= 2u_{n-1} - b_n, \quad n = 1, 2, \ldots, \\
b_n &= \lfloor 2u_{n-1} \rfloor = \begin{cases} 1, & u_{n-1} \geq 1/2, \\ 0, & u_{n-1} < 1/2. \end{cases}
\end{aligned}
$$

Remarks

1. Note that $u_n = T(u_{n-1})$ where T is the doubling map.
2. The values of u_n and b_n above are macroscopic and bounded. So the successive approximation algorithm as above can be implemented on an analog circuit.
3. Given the optimality of the accuracy for a given bit budget, are we done?

Example ctd.

When designing an A/D converter (ADC), accuracy is not the only concern! In fact, truncated base-2 representations (:= "pulse code modulation" or PCM) are far from being the most popular choice of A/D conversion method.

Why not?

In practice, analog circuits are never precise:

- arithmetic errors, e.g., through nonlinearity,
- quantizer errors, e.g., threshold offset,
- thermal noise...

Therefore:

- All relations hold approximately, and all quantities are approximately equal to their theoretical values;
- in particular, in the case of the above described algorithm, only for a finite number of iterations, given that dynamics of an expanding map has "sensitive dependence on initial conditions".

More resilient algorithms to compute base-2 representations?

Question. Are there better, i.e., more resilient, algorithms than "successive approximation" for evaluating $b_n(x)$ for each x?

More resilient algorithms to compute base-2 representations?

Question. Are there better, i.e., more resilient, algorithms than "successive approximation" for evaluating $b_n(x)$ for each x?

Answer. The bits in the base-2 representations are essentially uniquely determined. Therefore, there is no way to recover from an erroneous bit computation:

- a 1 assignment for b_n when $x < x_{n-1} + 2^{-n}$ means an "overshoot" from which there is no way to "back up" later,
- a 0 assignment for b_n when $x > x_{n-1} + 2^{-n}$ implies a "fall-behind" from which there is no way to "catch up" later.

Example ctd. – conclusion

1. Any ADC based on base-2 expansions is bound to be not robust.
2. The fundamental problem with base-2 expansions is the lack of redundancy in these representations.
3. As this is a central problem in A/D conversion (as well as in D/A conversion), many alternative bit representations of numbers, as well as of signals, have been adopted or devised by circuit engineers, e.g., beta-representations and $\Sigma\Delta$ modulation.
4. Both "beta-encoding" and "$\Sigma\Delta$ modulation" produce redundant representations of $x \in [0, 1]$.

Rest of the talk

- introduce basic notation and terminology
- focus on a class of converters called Algorithmic Converters, and establish mathematical framework (including a formal definition of robustness)
- discuss accuracy characteristics of certain widely used algorithmic converters: PCM (truncated binary expansion), sigma-delta schemes (truncated Sturmian words), beta encoders (truncated beta representations)
- identify problems with these classes – robustness vs. accuracy

Rest of the talk

- introduce basic notation and terminology
- focus on a class of converters called Algorithmic Converters, and establish mathematical framework (including a formal definition of robustness)
- discuss accuracy characteristics of certain widely used algorithmic converters: PCM (truncated binary expansion), sigma-delta schemes (truncated Sturmian words), beta encoders (truncated beta representations)
- identify problems with these classes – robustness vs. accuracy
- introduce a novel algorithmic converter, the Golden Ratio Encoder, with superior characteristics – proof of stability, approximation rate, robustness...

Basic definitions – encoder and decoder maps

Let X be a compact normed space (the space of analog objects).
E_N is an N-bit encoder if

$$E_N : X \mapsto \{0,1\}^N.$$

Basic definitions – encoder and decoder maps

Let X be a compact normed space (the space of analog objects).
E_N is an N-bit encoder if

$$E_N : X \mapsto \{0,1\}^N.$$

A progressive family of encoders $(E_N)_1^\infty$ is generated by a single map $\psi : X \mapsto \{0,1\}^{\mathbb{N}}$ such that

$$E_N(x) = (\psi(x)_1, \ldots, \psi(x)_N).$$

Basic definitions – encoder and decoder maps

Let X be a compact normed space (the space of analog objects). E_N is an *N*-bit encoder if

$$E_N : X \mapsto \{0,1\}^N.$$

A progressive family of encoders $(E_N)_1^\infty$ is generated by a single map $\psi : X \mapsto \{0,1\}^{\mathbb{N}}$ such that

$$E_N(x) = (\psi(x)_1, \ldots, \psi(x)_N).$$

A map $D_N : \mathrm{Range}(E_N) \mapsto X$ is a decoder for E_N.
In general, $x \in X$ cannot be perfectly recovered from $E_N(x)$. That is, quantization is inherently lossy.

Basic definitions – distortion and accuracy

For a given decoder D_N for the encoder E_N, the distortion can be measured by

$$\delta_X(E_N, D_N) = \sup_{x \in X} \|x - D_N(E_N(x))\|.$$

We define the accuracy of E_N as

$$\alpha(E_N) = \inf_{D_N} \delta_X(E_N, D_N).$$

Above the choice of norm depends on the setting.

Basic definitions – distortion and accuracy

For a given decoder D_N for the encoder E_N, the distortion can be measured by

$$\delta_X(E_N, D_N) = \sup_{x \in X} \|x - D_N(E_N(x))\|.$$

We define the accuracy of E_N as

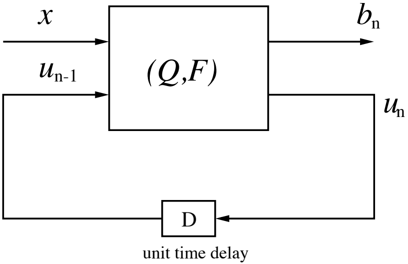
$$\alpha(E_N) = \inf_{D_N} \delta_X(E_N, D_N).$$

Above the choice of norm depends on the setting.
Remark.
When designing a progressive encoder family, one of the objectives:

$$\alpha(E_N) \to 0 \quad \text{as} \quad N \to \infty$$

as quickly as possible, e.g., exponential in N.

Algorithmic converters



$u_n \in \mathcal{U}$: state (continuous) of the circuit at time n
$x \in X$: the object to be quantized

$$Q : \mathcal{U} \times X \mapsto \{0, 1\} \qquad F : \mathcal{U} \times X \mapsto \mathcal{U}$$

The pair (Q, F) define a progressive family of encoders as follows:

$$b_n = Q(u_{n-1}, x) \qquad u_n = F(u_{n-1}, x).$$

The encoder E_N associated with (Q, F) is defined by

$$E_N(x) := (b_1, \ldots, b_N).$$

Algorithmic converters ctd.

Definition. Let $\psi^{Q,F}$ be the generator of the progressive family of encoders as defined above, i.e., for $x \in X$,

$$\psi^{Q,F}(x) := (b_1, b_2, \dots).$$

We say (Q, F) defines an algorithmic A/D converter if the map $\psi^{Q,F}$ is invertible on X.

Algorithmic converters ctd.

Definition. Let $\psi^{Q,F}$ be the generator of the progressive family of encoders as defined above, i.e., for $x \in X$,

$$\psi^{Q,F}(x) := (b_1, b_2, \dots).$$

We say (Q, F) defines an algorithmic A/D converter if the map $\psi^{Q,F}$ is invertible on X.

Remark. A large fraction of the ADCs used in practice, e.g., PCM (base-2), $\Sigma\Delta$ modulators, beta-encoders, are algorithmic converters. We will come back to this.

Algorithmic converters – robustness

Recall: Accuracy is not the only concern when evaluating the performance of an A/D converter!

Algorithmic converters – robustness

Recall: Accuracy is not the only concern when evaluating the performance of an A/D converter!

What else?

An ADC must be implemented, at least partly, on analog circuitry. Analog circuits are never precise.

In a typical implementation, the algorithmic converter functions are inaccurate:

$$(Q, F) \longleftrightarrow (\widetilde{Q}, \widetilde{F})$$

It is vital that the accuracy of the underlying algorithmic encoder is not drastically effected when such a change takes place.

Algorithmic converters – robustness

Quantify: Functions Q and F typically are compositions of elementary maps:

- Addition: $u \mapsto u + a$, $a \in \mathbb{R}$, $\quad (u, v) \mapsto u + v$.

Algorithmic converters – robustness

Quantify: Functions Q and F typically are compositions of elementary maps:

- Addition: $u \mapsto u + a$, $a \in \mathbb{R}$, $\quad (u, v) \mapsto u + v$.
- Multiplication: $u \mapsto bu$, $b \in \mathbb{R}$

Algorithmic converters – robustness

Quantify: Functions Q and F typically are compositions of elementary maps:

- Addition: $u \mapsto u + a$, $a \in \mathbb{R}$, $\quad (u, v) \mapsto u + v$.
- Multiplication: $u \mapsto bu$, $b \in \mathbb{R}$
- Decision element: $u \mapsto q_\tau(u) = \begin{cases} 0, & \text{if } u < \tau, \\ 1, & \text{if } u \geq \tau. \end{cases}$

Algorithmic converters – robustness

Quantify: Functions Q and F typically are compositions of elementary maps:

- Addition: $u \mapsto u + a$, $a \in \mathbb{R}$, $\quad (u, v) \mapsto u + v$.
- Multiplication: $u \mapsto bu$, $b \in \mathbb{R}$
- Decision element: $u \mapsto q_\tau(u) = \begin{cases} 0, & \text{if } u < \tau, \\ 1, & \text{if } u \geq \tau. \end{cases}$

Above, a, b, τ are parameters whose values are likely to vary within some tolerance.

Definition. Suppose $Q = Q_\lambda, F = F_\lambda$ where $\lambda \in \mathbb{R}^d$: parameters. Let E_N^λ be the associated algorithmic encoder. We say that E_N^λ is robust with respect to λ, if $\exists \epsilon > 0$ such that

$$\delta_X(E_N^\gamma, D_N^\lambda) \to 0 \quad \text{as} \quad N \to \infty \quad \text{whenever} \|\gamma - \lambda\| < \epsilon.$$

Examples

I. PCM (truncated binary) is an algorithmic converter.

Set $Q(u,x) = q_1(2u)$ and $F(u,x) = 2u - q_1(2u)$.

Encoder: (Successive approximation) For $x \in [0,1]$, initial state $u_0 = x$

$$b_n = q_1(2u_{n-1}) \qquad u_n = 2u_{n-1} - b_n, \quad n = 1, 2, \ldots$$

$E_N(x) = (b_1, \ldots, b_N) \rightarrow N$-bit trunc. binary exp. of x.

Generator: $\psi^{Q,F}(x) =$ bits in the binary expansion of x.

Decoder: $D_N(x) = 2^{-N-1} + \sum_{n=1}^{N} b_n 2^{-n}$.

Accuracy: $\alpha(E_N) = O(2^{-N})$ (optimal).

Examples - PCM ctd.

Let's investigate PCM in terms of its robustness properties.

Recall: $Q(u,x) = q_1(2x)$ and $F(u,x) = 2u - q_1(2x)$.

Important parameters: multiplication by 2 and threshold value $(= 1)$ of q_1.

Examples - PCM ctd.

Let's investigate PCM in terms of its robustness properties.

Recall: $Q(u,x) = q_1(2x)$ and $F(u,x) = 2u - q_1(2x)$.

Important parameters: multiplication by 2 and threshold value $(= 1)$ of q_1.

Imperfect implementation:

- multiply by $2 + \epsilon \Rightarrow |\epsilon/4| \leq \delta_X(\widetilde{E}_N, D_N)$,

Examples - PCM ctd.

Let's investigate PCM in terms of its robustness properties.

Recall: $Q(u,x) = q_1(2x)$ and $F(u,x) = 2u - q_1(2x)$.

Important parameters: multiplication by 2 and threshold value $(= 1)$ of q_1.

Imperfect implementation:

- multiply by $2 + \epsilon \Rightarrow |\epsilon/4| \leq \delta_X(\widetilde{E}_N, D_N)$,
- use q_τ with $|\tau - 1| \approx \epsilon \Rightarrow |\epsilon/2| \leq \delta_X(\widetilde{E}_N, D_N)$.

Examples - PCM ctd.

Let's investigate PCM in terms of its robustness properties.

Recall: $Q(u,x) = q_1(2x)$ and $F(u,x) = 2u - q_1(2x)$.

Important parameters: multiplication by 2 and threshold value $(= 1)$ of q_1.

Imperfect implementation:

- multiply by $2 + \epsilon \Rightarrow |\epsilon/4| \leq \delta_X(\widetilde{E}_N, D_N)$,
- use q_τ with $|\tau - 1| \approx \epsilon \Rightarrow |\epsilon/2| \leq \delta_X(\widetilde{E}_N, D_N)$.

PCM is not robust. To achieve the theoretical accuracy, one needs to implement a precise multiplier and a decision element with a precise "toggle point".

Digression – first-order ΣΔ modulation

Next, we review a family of ADCs which is popular in practice. Let $x \in [0,1]$. Define

$$T_x : u \mapsto \langle u + x \rangle.$$

Let $u_0 = \varphi \in [0,1)$ be arbitrary, and set

$$\left.\begin{cases} u_n = T_x^n(\varphi), \quad n = 1, 2, \ldots \\ b_n = \begin{cases} 0 & \text{if } u_{n-1} \in [0, 1-x), \\ 1 & \text{if } u_{n-1} \in [1-x, 1), \end{cases} \end{cases}\right\} \quad \text{1st-order } \Sigma\Delta$$

Digression – first-order $\Sigma\Delta$ modulation

Next, we review a family of ADCs which is popular in practice. Let $x \in [0,1]$. Define

$$T_x : u \mapsto \langle u + x \rangle.$$

Let $u_0 = \varphi \in [0,1)$ be arbitrary, and set

$$\left.\begin{cases} u_n = u_{n-1} + x - b_n, & n = 1, 2, \ldots \\ b_n = \lfloor u_{n-1} + x \rfloor & \end{cases}\right\} \quad \text{1st-order } \Sigma\Delta$$

Digression – first-order $\Sigma\Delta$ modulation

Next, we review a family of ADCs which is popular in practice.
Let $x \in [0, 1]$. Define

$$T_x : u \mapsto \langle u + x \rangle.$$

Let $u_0 = \varphi \in [0, 1)$ be arbitrary, and set

$$\left\{ \begin{array}{lcl} u_n & = & u_{n-1} + x - b_n, \quad n = 1, 2, \ldots \\ \\ b_n & = & \lfloor u_{n-1} + x \rfloor \end{array} \right\} \quad \text{1st-order } \Sigma\Delta$$

Remarks.

1. For irrational x, the above recursion produces Sturmian words. A first-order $\Sigma\Delta$ modulator encodes x by the associated N-bit truncated Sturmian word.
2. $\Sigma\Delta$ has been used for A/D conversion since 1960s.

Digression – first-order $\Sigma\Delta$ modulation

Remarks ctd.

3. **Encoding.** Let $E_N : x \mapsto (b_n)_1^N$
4. **Decoding.** Set $h_n = 1/N$, $n = 1, ..., N$, and define

$$D_N : (b_n)_1^N \mapsto \sum_{n=1}^{N} h_n b_n.$$

Then $|x - D_N(E_N(x))| \leq 1/N$.

5. One can improve this error bound by using a different reconstruction kernel $\tilde{h}$. In particular, Güntürk proved that

$$|x - \sum_{n=1}^{N} \tilde{h}_n b_n| \leq C_x N^{-2} \log^{2+\epsilon} N.$$

Proof uses machinery from discrepancy theory.

Digression – first-order $\Sigma\Delta$ modulation

Remarks ctd.

6. One can also obtain a lower bound for the approximation error:
   6.1 Consider the 1st-order $\Sigma\Delta$ scheme with $u_0 = 1/2$, let $b_{x,1/2}$ be the corresponding Sturmian word. Then for

$$A_{1/2}(N) := \{(b_{x,1/2})_1^N : x \in (0,1)\}, \quad \#A_{1/2} = \frac{3}{\pi^2} N^2 + O(N \log N).$$

   6.2 In fact, $(b_{x,1/2})_1^N = (b_{y,1/2})_1^N$ if x, y are between two consec. N-Farey points (Güntürk-Lagarias-Vaishampayan, cf. Mignosi). Then

$$C/N \leq \sup_{x \in (0,1)} |x - D_N^{opt}(b_{x,1/2})_1^N|$$

Digression – first-order ΣΔ modulation

Remarks ctd.

7. One can use 1st-order ΣΔ to quantize "varying input", e.g., samples of functions whose Fourier transform is compactly supported in $[-1/2, 1/2]$. Let $x_n = f(n/\lambda)$ where $\lambda > 1$ is the *oversampling factor*. With $u_0 = \varphi \in [0, 1)$, let, for $n = 1, 2, \ldots$,

$$u_n = T_{x_n}(u_{n-1}), \quad b_n = q_1(u_{n-1} + x_n),$$

One can also run this recursion backwards. Set $E(f) = (b_n)_{-\infty}^{\infty}$, and use the decoder D_ϕ

$$D_\phi : (b_n)_{-\infty}^{\infty} \mapsto (1/\lambda) \sum b_n \phi(\cdot - n/\lambda).$$

Here ϕ is an appropriate sampling kernel. Then we have (Daubechies-DeVore)

$$\|f - D_\phi(E(f))\|_\infty \leq \frac{1}{\lambda} \mathrm{Var}(\phi).$$

Digression – higher-order $\Sigma\Delta$ modulation

Rewrite the iteration for the 1st-order $\Sigma\Delta$:

$$u_n = u_{n-1} + x - \lfloor u_{n-1} + x \rfloor \rightsquigarrow (\Delta u)_n = x - q_1(u_{n-1} + x)$$

Generalize to kth-order: $u_{-k+1} = \cdots = u_0 = 0$, and

$$\left. \begin{cases} (\Delta^k u)_n & = x - b_n^k \\ b_n^k & = q_1(\rho(x, u_{n-1}, \ldots, u_{n-k+1})) \end{cases} \right\} k\text{th-order } \Sigma\Delta$$

Remarks

1. With an appropriate choice of $(h_n)_1^N$, one can show

$$|x - \sum_{n=1}^{N} h_n b_n^k| \leq CN^{-k}$$

if u_n remain bounded (unif. in N) (i.e., the scheme is stable).

Digression – higher-order $\Sigma\Delta$ modulation

Remarks ctd.

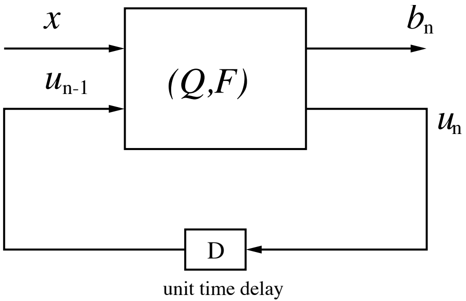
2. ρ is chosen to ensure stability (non-trivial). First infinite family of stable $\Sigma\Delta$ schemes of arbitrary order (not implementable in practice) was constructed by Daubechies and DeVore ($\sim$ 2000). For 2nd-order schemes, a wide family of rules ensure stability (OY-2002).
3. We can rewrite the recursion as

$$\mathbf{u}_n = \mathbf{L}_k \mathbf{u}_{n-1} + (x - q_1(\rho(\mathbf{u}, x)))\mathbf{1}$$

4. Error estimates can be improved (the piecewise affine system has tiling invariant sets)...
5. **Question.** Can we again count the number of possible N-words obtained via a kth-order $\Sigma\Delta$ scheme? A possible generalization of Sturmian shifts? ...

Back to examples of algorithmic converters

Recall



$u_n \in \mathcal{U}$: state (continuous) of the circuit at time n
$x \in X$: the object to be quantized

$$Q : \mathcal{U} \times X \mapsto \{0, 1\} \qquad F : \mathcal{U} \times X \mapsto \mathcal{U}$$

The pair (Q, F) define a progressive family of encoders as follows:

$$b_n = Q(u_{n-1}, x) \qquad u_n = F(u_{n-1}, x).$$

The encoder E_N associated with (Q, F) is defined by

$$E_N(x) := (b_1, \ldots, b_N).$$

Back to examples of algorithmic converters

II. First-order $\Sigma\Delta$ schemes are algorithmic converters.

Set $Q(u,x) = q_1(u+x)$ and $F(u,x) = u + x - q_1(u+x)$.

Encoder: For $x \in [0,1]$ and initial state $u_0 \in [0,1)$ arbitrary,

$$b_n = q_1(u_{n-1} + x), \qquad u_n = u_{n-1} + x - b_n, \quad n = 1, 2, \ldots$$

$E_N(x) = (b_1, \ldots, b_N) \rightarrow N$-bit $\Sigma\Delta$ encoding of x

Generator: $\psi^{Q,F}(x) = (b_1, b_2, \ldots)$.

Decoder: $D_N(x) = \dfrac{1}{N} \displaystyle\sum_{n=1}^{N} b_n$.

Accuracy: $\alpha(E_N) = O(1/N)$.

Examples – first-order $\Sigma\Delta$ ctd.

Robustness

Let's investigate 1st-order $\Sigma\Delta$ in terms of its robustness properties.
Recall: $Q(u,x) = q_1(u+x)$ and $F(u,x) = u + x - q_1(u+x)$.

Important parameters:

- Threshold value $(= 1)$ of q_1.
- No multiplier needed!

Examples – first-order $\Sigma\Delta$ ctd.

Robustness

Let's investigate 1st-order $\Sigma\Delta$ in terms of its robustness properties.
Recall: $Q(u,x) = q_1(u+x)$ and $F(u,x) = u + x - q_1(u+x)$.

Important parameters:

- Threshold value $(=1)$ of q_1.
- No multiplier needed!

Imperfect implementation:

- use q_τ with $|\tau - 1| \le \epsilon \Rightarrow \delta_X(\widetilde{E}_N, D_N) = O(1/N)$.

Examples – first-order $\Sigma\Delta$ ctd.

Robustness

Let's investigate 1st-order $\Sigma\Delta$ in terms of its robustness properties.
Recall: $Q(u,x) = q_1(u+x)$ and $F(u,x) = u + x - q_1(u+x)$.

Important parameters:

- Threshold value $(=1)$ of q_1.
- No multiplier needed!

Imperfect implementation:

- use q_τ with $|\tau - 1| \leq \epsilon \Rightarrow \delta_X(\widetilde{E}_N, D_N) = O(1/N)$.

First-order $\Sigma\Delta$ is robust! Main reason $\Sigma\Delta$ is popular...

Note. The accuracy of an N-bit 1st-order $\Sigma\Delta$ encoder is of $O(1/N)$, much worse than $O(2^{-N})$, accuracy of PCM.

Examples – kth-order $\Sigma\Delta$

III. kth-order $\Sigma\Delta$ schemes are algorithmic converters.

State space $\mathcal{U} \subset \mathbb{R}^k$.

Set $Q(\mathbf{u}, x) = q_1(\rho(\mathbf{u}, x))$ and $F(\mathbf{u}, x) = \mathbf{L}_k \mathbf{u} + x - q_1(\rho(\mathbf{u}, x))$.

$\rho : \mathcal{U} \times X \mapsto \mathbb{R}$ is called "quantization rule". (stability!)

$\mathbf{L}_k$ is the $k \times k$ lower triangular matrix of 1s.

Examples – kth-order $\Sigma\Delta$

III. kth-order $\Sigma\Delta$ schemes are algorithmic converters.

State space $\mathcal{U} \subset \mathbb{R}^k$.
Set $Q(\mathbf{u}, x) = q_1(\rho(\mathbf{u}, x))$ and $F(\mathbf{u}, x) = \mathbf{L}_k \mathbf{u} + x - q_1(\rho(\mathbf{u}, x))$.

$\rho : \mathcal{U} \times X \mapsto \mathbb{R}$ is called "quantization rule". (stability!)

$\mathbf{L}_k$ is the $k \times k$ lower triangular matrix of 1s.

Encoder: For $x \in [0, a]$, $a < 1$, initial state $\mathbf{u}_0 \in B \subset \mathbb{R}^k$ arbitrary,

$$b_n = Q(\mathbf{u}_n, x) \qquad \mathbf{u}_{n+1} = F(\mathbf{u}_n, x), \quad n = 1, 2, \ldots$$

$E_N(x) = (b_1, \ldots, b_N) \to$ N-bit $\Sigma\Delta$ encoding (order k) of x

Decoder: $D_N(x) = \sum_{n=1}^{N} h_n b_n$; h_n: approp. sampling kernel

Accuracy: $\alpha(E_N) = O(1/N^k)$.

Examples – kth-order $\Sigma\Delta$ ctd.

Robustness

Again, what about robustness of a kth-order $\Sigma\Delta$-scheme?

Important parameters:

- Threshold value $(= 1)$ of q_1, and
- multiplications and additions performed in the quantization rule ρ.

Examples – kth-order $\Sigma\Delta$ ctd.

Robustness

Again, what about robustness of a kth-order $\Sigma\Delta$-scheme?

Important parameters:

- Threshold value $(= 1)$ of q_1, and
- multiplications and additions performed in the quantization rule ρ.

kth-order $\Sigma\Delta$ with a wide family of quantization rules is robust! [Daubechies-DeVore, OY]

Note. The accuracy of an N-bit kth-order $\Sigma\Delta$ encoder is of $O(N^{-k})$, still much worse than $O(2^{-N})$, accuracy of PCM.

Examples – beta encoders

IV. Beta encoders (Daubechies et al.) are algorithmic converters.

Let $1 < \beta < 2$, and compute truncated cautious (not greedy, not lazy) beta representations of $x \in [0, 1)$.

Set $Q(u, x) = q_1(\beta u - \mu)$ and $F(u, x) = \beta u - q_1(\beta u - \mu)$.

Note that this corresponds to the recursion

$$u_n = \beta u_{n-1} - b_n, \quad b_n = \lfloor \beta u_{n-1} - \mu \rfloor$$

with $u_0 = x$

- $\mu = 0$: greedy selection,
- $\mu = (2 - \beta)/(\beta - 1)$: lazy selection
- $0 < \mu < (2 - \beta)/(\beta - 1)$: cautious selection.

Examples – beta encoders

IV. Beta encoders (Daubechies et al.) are algorithmic converters.

Let $1 < \beta < 2$, and compute truncated cautious (not greedy, not lazy) beta representations of $x \in [0, 1)$.

Set $Q(u, x) = q_1(\beta u - \mu)$ and $F(u, x) = \beta u - q_1(\beta u - \mu)$.

Encoder: For $x \in [0, 1)$ and initial state $u_0 = x$,

$$b_n = q_1(\beta u_{n-1} - \mu), \qquad u_n = \beta u_{n-1} - b_n, \quad n = 1, 2, \ldots$$

$E_N(x) = (b_1, \ldots, b_N) \rightarrow$ an N-bit trunc. β-rep. of x.

Decoder: $D_N(x) = \sum_{n=1}^{N} b_n \beta^{-n}$.

Accuracy: $\alpha(E_N) = O(\beta^{-N})$.

Examples – beta-encoders ctd.

Robustness

Recall: $Q(u,x) = q_{1+\mu}(\beta u)$ and $F(u,x) = \beta u - q_{1+\mu}(\beta u + \mu)$.

Important parameters: Threshold value $(= 1 + \mu)$ of q and multiplication by β

Examples – beta-encoders ctd.

Robustness

Recall: $Q(u,x) = q_{1+\mu}(\beta u)$ and $F(u,x) = \beta u - q_{1+\mu}(\beta u + \mu)$.

Important parameters: Threshold value $(= 1+\mu)$ of q and multiplication by β

Imperfect implementation:

- use q_τ with $|\tau - (1+\mu)| \le \epsilon \Rightarrow \delta_X(\widetilde{E}_N, D_N) = O(\beta^{-N})$.

Examples – beta-encoders ctd.

Robustness

Recall: $Q(u,x) = q_{1+\mu}(\beta u)$ and $F(u,x) = \beta u - q_{1+\mu}(\beta u + \mu)$.

Important parameters: Threshold value $(= 1 + \mu)$ of q and multiplication by β

Imperfect implementation:

- use q_τ with $|\tau - (1+\mu)| \leq \epsilon \Rightarrow \delta_X(\widetilde{E}_N, D_N) = O(\beta^{-N})$.
- multiply with $\beta + \epsilon$ at each multiplier $\Rightarrow C\epsilon \leq \delta_X(\widetilde{E}_N, D_N)$.

Examples – beta-encoders ctd.

Robustness

Recall: $Q(u,x) = q_{1+\mu}(\beta u)$ and $F(u,x) = \beta u - q_{1+\mu}(\beta u + \mu)$.

Important parameters: Threshold value $(= 1+\mu)$ of q and multiplication by β

Imperfect implementation:

- use q_τ with $|\tau - (1+\mu)| \leq \epsilon \Rightarrow \delta_X(\widetilde{E}_N, D_N) = O(\beta^{-N})$.
- multiply with $\beta + \epsilon$ at each multiplier $\Rightarrow C\epsilon \leq \delta_X(\widetilde{E}_N, D_N)$.
- The assumed value of β is different from the actual implemented value. Partial solution in [Daubechies-OY], still not satisfactory.

Examples – beta-encoders ctd.

Robustness

Recall: $Q(u,x) = q_{1+\mu}(\beta u)$ and $F(u,x) = \beta u - q_{1+\mu}(\beta u + \mu)$.

Important parameters: Threshold value $(= 1+\mu)$ of q and multiplication by β

Imperfect implementation:

- use q_τ with $|\tau - (1+\mu)| \leq \epsilon \Rightarrow \delta_X(\widetilde{E}_N, D_N) = O(\beta^{-N})$.
- multiply with $\beta + \epsilon$ at each multiplier $\Rightarrow C\epsilon \leq \delta_X(\widetilde{E}_N, D_N)$.
- The assumed value of β is different from the actual implemented value. Partial solution in [Daubechies-OY], still not satisfactory.

Beta encoders are robust wrt quantizer threshold value. They are not robust wrt multiplication by β.

Moral so far...

Encoders that enjoy superior accuracy properties (PCM, Beta) suffer from robustness issues.

Encoders that enjoy superior robustness properties ($\Sigma\Delta$) have inferior accuracy characteristics.

Next, we present a scheme with the best of both worlds!

The Golden Ratio Encoder (GRE)

Main idea. The above (classical) implementation of beta-encoders:

$$u_{n+1} = \beta u_n - b_n; \quad b_n = Q(u_n); \quad u_1 = x.$$

The characteristic polynomial: $p(y) = y - \beta$; choice of b_n ensures $|u_n|$ remain bounded –in this case, the scheme is stable.

The Golden Ratio Encoder (GRE)

Main idea. The above (classical) implementation of beta-encoders:

$$u_{n+1} = \beta u_n - b_n; \quad b_n = Q(u_n); \quad u_1 = x.$$

The characteristic polynomial: $p(y) = y - \beta$; choice of b_n ensures $|u_n|$ remain bounded –in this case, the scheme is stable.

Question. Is it possible to use more suitable difference equations and still obtain a β representation of $x \in [0, 1)$?

We want the characteristic polynomial to have integer coefficients (coefficients ± 1 are preferred), have one of its roots at $\beta \in (1, 2)$, and that $\exists (b_n)$ to keep the resulting system stable.

GRE

Consider

$$\boxed{u_{n+2} = u_{n+1} + u_n - b_n; \quad u_0 = x, \;\; u_1 = 0.}$$

The characteristic equation is $p(y) = y^2 - y - 1$ whose roots are

$$\phi = \frac{1+\sqrt{5}}{2} \;\; \text{(the golden ratio), and} \quad -\frac{1}{\phi}.$$

Using $\phi^2 = \phi + 1$, we obtain

$$D_N(b) = \sum_{n=0}^{N-1} b_n \phi^{-n} = x - \phi^{-N}(u_N + \phi u_{N+1}).$$

GRE

Consider

$$\boxed{u_{n+2} = u_{n+1} + u_n - b_n; \quad u_0 = x, \;\; u_1 = 0.}$$

The characteristic equation is $p(y) = y^2 - y - 1$ whose roots are

$$\phi = \frac{1+\sqrt{5}}{2} \;\; \text{(the golden ratio), and} \quad -\frac{1}{\phi}.$$

Using $\phi^2 = \phi + 1$, we obtain

$$D_N(b) = \sum_{n=0}^{N-1} b_n \phi^{-n} = x - \phi^{-N}(u_N + \phi u_{N+1}).$$

Proposition. If there is rule for choosing b_n such that $|u_n| \leq C$, then the above iteration produces a beta encoding of x with $\beta = \phi$ (hence the name). That is, for the corresp. encoder E_N,

$$|x - D_N(E_N(x))| = O(\phi^{-N}).$$

Next, we establish such rules...

Stability of GRE

Simplest stable GRE. Set

$$b_n = q_1(u_{n+1} + u_n) = \begin{cases} 0, & \text{if } u_{n+1} + u_n < 1, \\ 1, & \text{if } u_{n+1} + u_n \geq 1. \end{cases}$$

Proposition. For $x \in [0,1)$, if we run

$$u_{n+2} = u_{n+1} + u_n - b_n; \quad b_n = q_1(u_{n+1} + u_n); \quad u_0 = x; \quad u_1 = 0,$$

we have $0 \leq u_n \leq 1$ for every n.

Stability of GRE

Simplest stable GRE. Set

$$b_n = q_1(u_{n+1} + u_n) = \begin{cases} 0, & \text{if } u_{n+1} + u_n < 1, \\ 1, & \text{if } u_{n+1} + u_n \geq 1. \end{cases}$$

Proposition. For $x \in [0, 1)$, if we run

$$u_{n+2} = u_{n+1} + u_n - b_n; \quad b_n = q_1(u_{n+1} + u_n); \quad u_0 = x; \quad u_1 = 0,$$

we have $0 \leq u_n \leq 1$ for every n.

Remarks.

1. The corresponding GRE is stable, thus its accuracy is $O(\phi^{-N})$.
2. Not even one multiplication!

Stability of GRE

Simplest stable GRE. Set

$$b_n = q_1(u_{n+1} + u_n) = \begin{cases} 0, & \text{if } u_{n+1} + u_n < 1, \\ 1, & \text{if } u_{n+1} + u_n \geq 1. \end{cases}$$

Proposition. For $x \in [0, 1)$, if we run

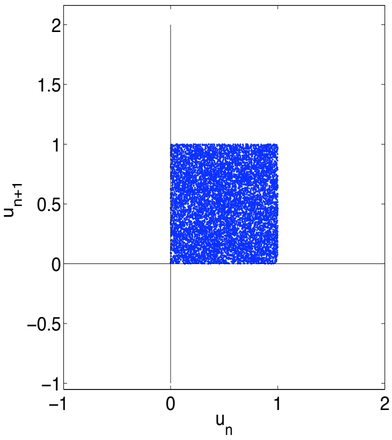
$$u_{n+2} = u_{n+1} + u_n - b_n; \quad b_n = q_1(u_{n+1} + u_n); \quad u_0 = x; \quad u_1 = 0,$$

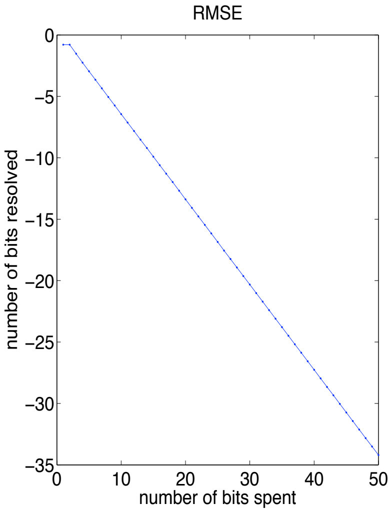
we have $0 \leq u_n \leq 1$ for every n.

Remarks.

1. The corresponding GRE is stable, thus its accuracy is $O(\phi^{-N})$.
2. Not even one multiplication!
3. Unfortunately, not robust wrt quantizer threshold: Replace q_1 with $q_{1+\epsilon} \Rightarrow$ unstable scheme! Need to do more work...

Numerical Experiment





Stability of GRE

Stable GREs with better robustness properties.

Describe the GRE iteration with a 2d-map. Define

$$T_Q : \begin{bmatrix} u \\ v \end{bmatrix} \mapsto \begin{bmatrix} 0 & 1 \\ 1 & 1 \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} - Q(u,v) \begin{bmatrix} 0 \\ 1 \end{bmatrix}.$$

Then, we can rewrite the recursion as

$$\begin{bmatrix} u_{n+1} \\ u_{n+2} \end{bmatrix} = T_Q \begin{bmatrix} u_n \\ u_{n+1} \end{bmatrix}.$$

Note: We now observe that GRE is an algorithmic converter.

Above, we used $Q(u,v) = q_1(u+v)$. We will now construct alternative Q for which the scheme is stable and robust.

Stability of GRE

Stable GREs with better robustness properties (ctd.)

Use $Q(u, v) = q_\tau(u + \gamma v) =: Q_\tau^\gamma(u, v)$ with $\gamma \neq 1$ and approp. τ.

Note. If implemented with Q_τ^γ, the parameters of concern regarding robustness are γ and τ.

Stability of GRE

Stable GREs with better robustness properties (ctd.)

Use $Q(u,v) = q_\tau(u + \gamma v) =: Q_\tau^\gamma(u,v)$ with $\gamma \neq 1$ and approp. τ.

Note. If implemented with Q_τ^γ, the parameters of concern regarding robustness are γ and τ.

Main Theorem. For every $1 < \gamma < 3$, there exists $\nu_1 < \nu_2$, and $\eta > 0$ such that GRE implemented with $Q_\tau^{\gamma'}$ is stable provided $|\gamma' - \gamma| < \eta$ and $\nu_1 < \tau < \nu_2$.

Corollary. The GRE implemented with Q_τ^γ is robust wrt γ and τ. In particular,

$$\delta_X(GRE_N^{\gamma',\tau}, D_N) = O(\phi^{-N})$$

whenever $|\gamma' - \gamma| < \eta$ and $\nu_1 < \tau < \nu_2$.

Stability of GRE

Stable GREs with better robustness properties (ctd.)

Use $Q(u,v) = q_\tau(u+\gamma v) =: Q_\tau^\gamma(u,v)$ with $\gamma \neq 1$ and approp. τ.

Note. If implemented with Q_τ^γ, the parameters of concern regarding robustness are γ and τ.

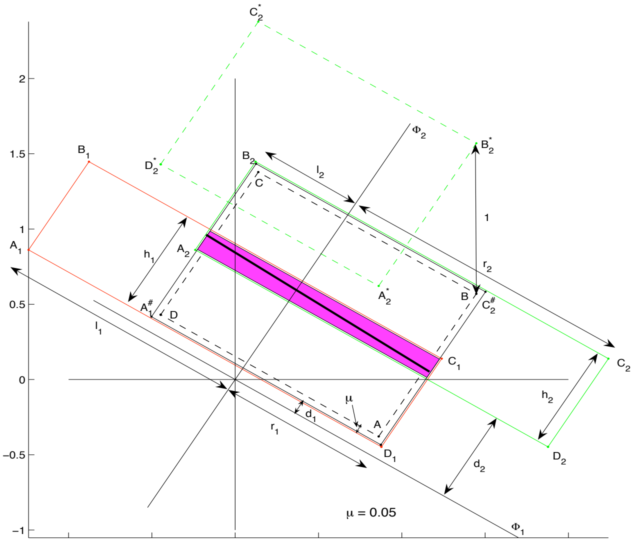
Main Theorem. For every $1 < \gamma < 3$, there exists $\nu_1 < \nu_2$, and $\eta > 0$ such that GRE implemented with $Q_\tau^{\gamma'}$ is stable provided $|\gamma' - \gamma| < \eta$ and $\nu_1 < \tau < \nu_2$.

Corollary. The GRE implemented with Q_τ^γ is robust wrt γ and τ. In particular,

$$\delta_X(GRE_N^{\gamma',\tau}, D_N) = O(\phi^{-N})$$

whenever $|\gamma' - \gamma| < \eta$ and $\nu_1 < \tau < \nu_2$.

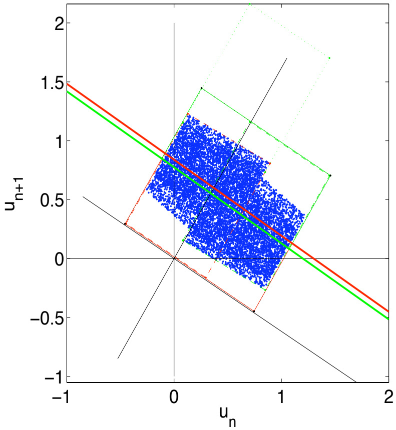
Sketch of the proof. See picture.

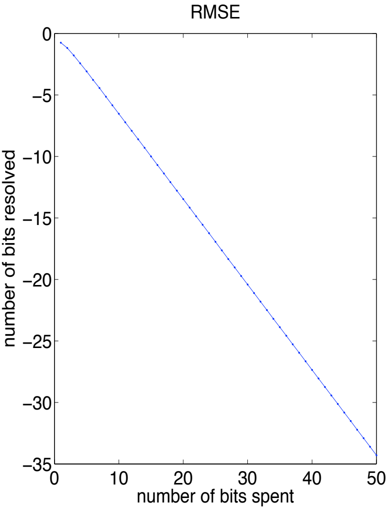


C_2^*
Φ_2
B_2^*
B_1
D_2^*
B_2
l_2
C
1
A_1
h_1
A_2
r_2
A_2^*
B
$C_2^\#$
$A_1^\#$
D
l_1
C_1
C_2
h_2
μ
d_1
A
r_1
D_1
D_2
d_2
$\mu = 0.05$
Φ_1
2
1.5
1
0.5
0
−0.5
−1
−1
−0.5
0
0.5
1
1.5
2

Numerical Experiment

$\nu_1 = 1.2, \nu_2 = 1.3, \gamma = 1.55.$





Highlights

1. GRE is an algorithmic A/D converter. Its implementation does not require any "precise multiplication" or "precise decision element".
2. GRE enjoys exponential accuracy.

Highlights

1. GRE is an algorithmic A/D converter. Its implementation does not require any "precise multiplication" or "precise decision element".
2. GRE enjoys exponential accuracy.
3. GRE is a "Nyquist-rate A/D converter", i.e., it quantizes each sample value independently (no memory). This makes GRE a good candidate for A/D conversion in settings where classical sampling theory does not apply, e.g., compressed sensing.

Highlights

1. GRE is an algorithmic A/D converter. Its implementation does not require any "precise multiplication" or "precise decision element".
2. GRE enjoys exponential accuracy.
3. GRE is a "Nyquist-rate A/D converter", i.e., it quantizes each sample value independently (no memory). This makes GRE a good candidate for A/D conversion in settings where classical sampling theory does not apply, e.g., compressed sensing.
4. GRE was implemented using the Fibonacci recursion. One can generalize and construct higher order "polynacci encoders" with $p(y) = y^k - y^{k-1} - \cdots - 1$ whose largest root $\beta_k \in (1, 2)$, all other roots inside the unit circle (thus like ϕ, β_k is a Pisot number). Moreover $\beta_k \to 2$.

Highlights

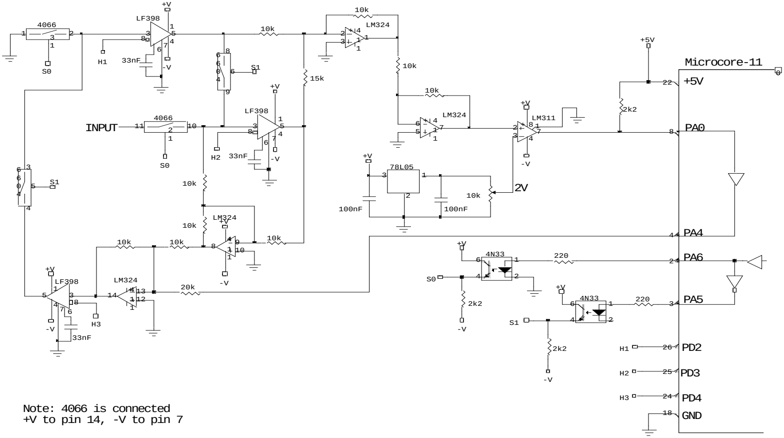
1. GRE is an algorithmic A/D converter. Its implementation does not require any "precise multiplication" or "precise decision element".
2. GRE enjoys exponential accuracy.
3. GRE is a "Nyquist-rate A/D converter", i.e., it quantizes each sample value independently (no memory). This makes GRE a good candidate for A/D conversion in settings where classical sampling theory does not apply, e.g., compressed sensing.
4. GRE was implemented using the Fibonacci recursion. One can generalize and construct higher order "polynacci encoders" with $p(y) = y^k - y^{k-1} - \cdots - 1$ whose largest root $\beta_k \in (1,2)$, all other roots inside the unit circle (thus like ϕ, β_k is a Pisot number). Moreover $\beta_k \to 2$.
5. Other technical issues, e.g., bias removal, requantization can be resolved.

Highlights

1. GRE is an algorithmic A/D converter. Its implementation does not require any "precise multiplication" or "precise decision element".
2. GRE enjoys exponential accuracy.
3. GRE is a "Nyquist-rate A/D converter", i.e., it quantizes each sample value independently (no memory). This makes GRE a good candidate for A/D conversion in settings where classical sampling theory does not apply, e.g., compressed sensing.
4. GRE was implemented using the Fibonacci recursion. One can generalize and construct higher order "polynacci encoders" with $p(y) = y^k - y^{k-1} - \cdots - 1$ whose largest root $\beta_k \in (1, 2)$, all other roots inside the unit circle (thus like ϕ, β_k is a Pisot number). Moreover $\beta_k \to 2$.
5. Other technical issues, e.g., bias removal, requantization can be resolved.
6. Finally, implementation...

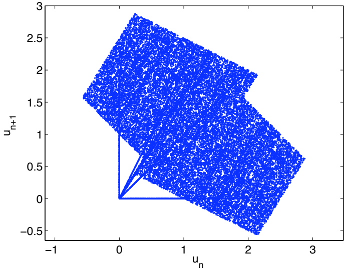
Does it work on analog hardware?

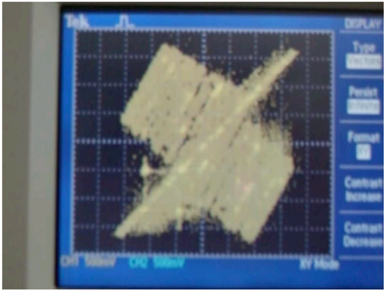
We implemented the GRE on a breadboard...



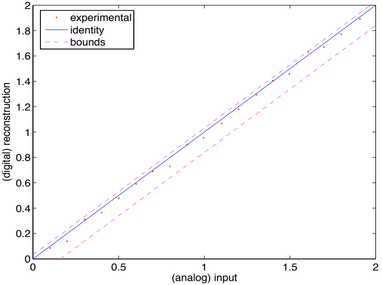
Hardware implementation

We plot u_{n+1} vs. u_n, computed theoretically (left) and measured from the circuit (right).

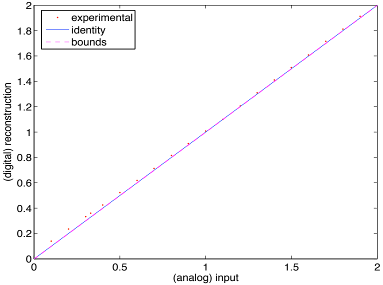




Performance of the circuit



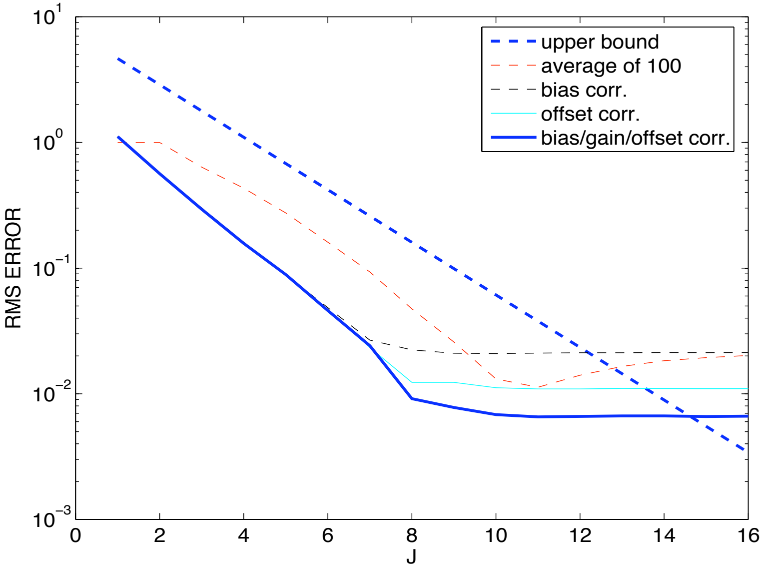
8-bit data



16-bit data

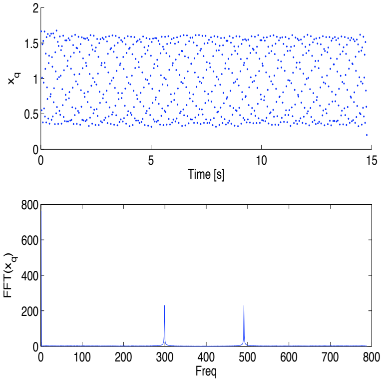
Performance of the circuit

The RMSE error vs. number of GRE bits (J).



Performance of the circuit

AC data.



16-GRE-bit reconstruction of a sinusoid $x(t) = (A - B)\sin(2\pi f t) + (A + B)/2$ with $f = 20.2Hz$, $A \approx 1.57$, and $B \approx 0.35$. The sampling rate was approx. 53.33 Hz.