All of the previous algos we discussed were inefficient. Yes, they are sample efficient but not computationally; we take considerably fewer samples than the full signal but then do lots of FFTs! All to return only k items! Very inefficient (unless your application is really only constrained by sampling time/powr space/...).

Let's look at architecture of oSaMP to understand the bottlenecks and what we should change:

1. \( r = y - \Phi x^{(t)} = \Phi (x - x^{(t)}) \) = sample from residual signal or sample from current approx \( x^{(t)} \) and subtract from measurements.

\[ \text{Don't compute } \Phi x^{(t)} \text{ explicitly!} \]

2. \( w^{(t+1)} = \arg \min_{z \in \mathbb{C}^n} \| y - \Phi z \|_2 \quad \text{subject to} \quad z \in \Lambda^{(t+1)} \)

\[ \Theta(k\log n) = m \]

\[ \Lambda^{(t+1)} \]

or knowing \( \Lambda^{(t+1)} = 3k \)-term supp set, estimate the value of the entries of \( z \) on that support set.

3. \( \text{Supp}_{2k} (\hat{r}) \)

Given samples (of residual), how to find/identify top 2k entries without computing \( \hat{r} \). 

These are our two big bottlenecks.
This is actually revisionist history! SFT algs have been around for a while, just in slightly different form.

1991 Kushilevitz & Mansour
1989/1993 Goldreich & Levin (crypt/ades)

Hadamard transform
(Fourier analysis on Boolean cube)

1992 Kushilevitz
2002 GGIMNS (streaming/sublinear alg)
2003 Akavia, Goldwasser, Safra (crypto)

For various flavors of $\mathbb{Z}_n$ for various flavors of $n$
($n$ prime, $n = 2^l$, $n$ prime, $n = 2^l$, $n$ prime, $n = 2^l$

Gilbert, Muthukrishnan, Strauss

$\text{running time } (\text{+ # samples}) = O(k \log(n))$

Randomized algs with constant prob. of error (i.e., per signal)

2010 onwards LOTS OF WORK!
big flurry of algs, implementations, and hardware.
Detic constructions, too.

See 2015 SP Mag survey, with
most prominent one is HIKP 2012, by Hassanieh, Indyk, Katabi, Price.

Let's outline basic components & techniques (that don't suffer from the bottlenecks).

1. To get intuition, let's assume spectrum of $x$ (on $\mathbb{Z}_n$) consists of a

   SINGLE non-zero freq.

   - two samples are sufficient to get position and value/coeff.
     of freq.
   - $k \log(n)$ samples suffice if tone + noise.

2. If we have ... is ... then in $k$-fold random subspace of Fourier space

i.e., build a filterbank
1. Single freq. recovery: suppose freq. $w$ is fixed with coeff. $x_w \in \mathbb{C}$. 

$$x_j = x_w \frac{1}{\sqrt{n}} e^{j\frac{\pi}{n}} \quad j \in \mathbb{Z}_n$$

a. calculate $w$ by choosing $j, j+1 \in \mathbb{Z}_n$ and calculating

$$\text{phase}\left(\frac{x_{j+1}}{x_j}\right) = \frac{\alpha_w \sqrt{n}}{x_w \sqrt{n}} e.\frac{\pi \sin(1/n)}{\pi \sin(1/n)} = e$$

$$= \text{phase}\left(\cos\left(\frac{\pi w}{n}\right) + i \sin\left(\frac{\pi w}{n}\right)\right)$$

b. once you know $w$, calculate $\alpha_w$ by computing

$$x_j \sqrt{n} e^{-j\frac{\pi}{n}} = \alpha_w \cdot 1$$

---

This doesn't work very well if $x$ is noisy. Have to perform binary search to zoom in on $w$. Let's suppose $n=8$.

---

Diagram showing the unit circle in $\mathbb{C}$ with points labeled for division into 2 halves and testing if closer to $-1$ or $1$.
\[ |x_j| \cdot |1 - e^{2\pi i \omega/b}| = |i \cdot x_j - x_{j+1}| < |x_j + x_{j+1}| = |x_j| \cdot |1 + e^{2\pi i \omega/b}| \]

\[ |x_j| \cdot |1 - e^{\pi i \omega/b}| = |x_j - x_{j+1}| < |x_j + x_{j+1}| = |x_j| \cdot |1 + e^{\pi i \omega/b}| \]

\[ |1 + e^{\pi i \omega/b}| < |1 + e^{2\pi i \omega/b}| \quad \text{(i.e., } e^{\pi i \omega/b} \text{ is closer)} \]

If \( e^{\pi i \omega/b} \) is closer to +1, then we will see
\[ |x_j - x_{j+1}| < |x_j + x_{j+1}| \quad \text{[and similarly for]} \]
\[ e^{\pi i \omega/b} \quad \text{is closer to +1, then we have } \{1, 0, i\} \text{ and } \{0, 1, -i\} \]
\[ \text{[similarly for } e^{\pi i \omega/b} \text{ and } e^{3\pi i \omega/b} \text{ and } e^{2\pi i \omega/b} \text{]} \]

Let's suppose we learn \( \omega \in \{1, 3, 5, 6\} \), then we can simplify our problem

Define \( x_j' = e^{-2\pi i \cdot 4 \cdot (2j/\omega)} \cdot x_j \)

\[ = e^{-2\pi i \cdot (2j/\omega)} \cdot \frac{\omega}{\sqrt{n}} e^{2\pi i \omega/b} \]

\[ = \frac{\omega}{\sqrt{n}} e^{2\pi i (\omega - 4)/4} \quad \text{for } j \in \mathbb{Z}_{n/2} = \mathbb{Z}_4. \]

- We shifted/rotated domain into first quadrant by multiplying by \( e^{-2\pi i \cdot 2j/\omega} \).
- Then discarded odd entries.

and iterate...

This is in the Gillard- Strauss- Tropp 2008 survey, albeit expressed slightly differently.
Let's do an example. Suppose $n = 2 \cdot 5 \cdot 7 = 70$ [a smooth # = product of lots of small primes]

and suppose

$$x_j = \alpha \cdot \frac{1}{\sqrt{n}} \cdot e^{2\pi j \omega / n}$$

$x$ is $1$-tone with freq. $\omega \in \mathbb{Z}/n$.

for $j \in \mathbb{Z}/n$

Let's form a short vector $a \in \mathbb{C}^2$ by sampling $x$ at $j = 0, n/2$.

$$a_0 = x_0 = \alpha \cdot \frac{1}{\sqrt{n}}$$

$$a_1 = x_{n/2} = \alpha \cdot \frac{1}{\sqrt{n}} e^{2\pi \cdot 1 \cdot \omega / n} = \alpha \cdot \frac{1}{\sqrt{n}} (-1)$$

and compute the Fourier transform of $a$ ($\hat{a} \in \mathbb{C}^2$):

$$\hat{a}_0 = \alpha \cdot \frac{1}{\sqrt{n}} \left( \frac{1 + (-1)^w}{\sqrt{2}} \right)$$

$w$ is an integer and

it's either EVEN or ODD

$\Rightarrow$ only one of $\hat{a}_0 \neq \hat{a}_1$ will be non-zero

if $\hat{a}_0 \neq 0$, then $\omega \equiv 0 \mod 2$

if $\hat{a}_1 \neq 0$, then $\omega \equiv 1 \mod 2$

Can learn $\omega \mod 5$ and $7$ by sampling $x$ at intervals $n/5$, $n/7$ and then computing shorter FFTs. [aliased FFTs can do in parallel.]

Then, from moduli, can reconstruct $\omega$.

Thus (CRT)

Any integer $x$ is uniquely specified mod $n$ by its remainders
mod $m$ rel. prime into $p_1, p_2, ..., p_m$ as long as $\prod_{i=1}^{m} p_i \geq n$.

How many samples? How fast?

$$2 + 5 + 7 = 14$$

FFT FFT FFT ... and then FFTs. 

The algorithm is a small linear system

size = # prime factors.

$\sim O(n)$
Filtering to Isolate Freqs.

It's important to view our CRT argument as a sampling from a filtered version of our signal in order to figure out how to handle signals with more than 1 freq.

- the sampling at 0, n/2
  - "filters" the signal into 2 freq. "bins"—
    - those that are EVEN \( \equiv \) 0 \( \mod \) 2
      - Similarly for the other factors bins for different conj. classes \( \mod \) 5, 7.

- ex:
  - 3 bins for 0, 1, 2 \( \mod \) 3.
  - 2 freqs in spectrum.

If signal has 2 freqs, then the freqs can't be equal \( \mod \) 3

(CRT says they can't collide if you take enough primes)

More generally: need \( \mathbb{F}_k^3 \)

- bins to isolate the k freqs.
- \( \Rightarrow \) short FFTs
  - give enough info to get all k freqs.

Lots of research on good near-perfect bandpass filters
- \( \text{e.g. sinc} \times \text{Gaussian} \)

[need a short \# coeff on time...]

diff. sublinear algos. have used
diff. filters: Gaussians, Delphi-Cholstep,
indicators, spike trains, ...

Q: How do we handle freqs. that are close together?

Those bins are del'ic!

but sample closely...
Need:
1. Permute spectrum efficiently → use pairwise indep.
   random permutations
2. Sample on time side from the
   permuted spectrum → use 2 invariance props.
   of the FT.

\[ a_j = x ej \quad \leftrightarrow \quad \hat{a}_j = \hat{x}_{c'}j \quad \text{assuming } c' \text{ exists mod n} \]

b) Translation

\[ a_j = e^{2\pi ibj/n} \quad \leftrightarrow \quad \hat{a}_j = \hat{x}_j \cdot b \]

\[ \Rightarrow \text{if } we \ \text{choose } b, c \text{ indep., uniformly at random}, \ (c \in \mathbb{Z}_n^*), \ \text{then} \]

\[ a_j = e^{2\pi ibj/n} \cdot x ej \]

has a permuted spectrum \[ \hat{a}_j = \hat{x}_{c'}j - b \]; i.e.,

\[ \hat{x}_w \Rightarrow \hat{a}_w \text{; then } \hat{a}_{(wc+b) \text{mod n}} = \lambda w \]

we permute the spectrum

You check: if \( b, c \) indep., uniformly random (\( c \in \mathbb{Z}_n^* \)), then

\[ w \mapsto wc + b \] is a random permutation

2. It's actually a pairwise indep. random permutation

\[ cw_1 + b = s_1 \]
\[ cw_2 + b = s_2 \]

what's the probability of collision?
Put all these pieces together in an iterative procedure:

```
while not done{
    IDENTIFY freqs. with big Fcoeffs
    ESTIMATE Fcoeffs of those identified freqs.
    SUBTRACT (samples of) current approx. from
        (samples of) orig. signal. (i.e., re-measure)
}
```

There are some implementations:

- MAFIT (on coursera.org)
- OPPET V1,2,3 (on MIT webpage)
- ETH

[GST 200B has full pseudocode that is easy to implement!]

[Note: caveats!]

[HIKP1,2 also have pseudocode]
SFT \((x, k)\)  

Input: \(x\) has length \(n = 2^l\)  
\(k\) # of desired freqs.  
Gen. samples FIRST and input those.  

Output: \(\Lambda = \{(w, \alpha_w)\}\) = list containing \(O(k)\) (freq, coeff) pairs.

\[ K \leftarrow 8k \rightarrow \text{initialize.} \]
\[ \Lambda \leftarrow \emptyset \]

for \(j = 1\) to \(5\) \(\}

\[ \Omega \leftarrow \text{identification (} \kappa, \Lambda, K \) \]
\[ c \leftarrow \text{estimation (} \kappa, \Lambda, \Omega \) \]

for each \(w \in \Omega\) \{  
  if \((w, \alpha_w) \in \Lambda\), then replace with \((w, \alpha_w + c_w)\) in \(\Lambda\)
  else add new pair \((w, c_w)\) to \(\Lambda\).
\}

Return \(K\) pairs in \(\Lambda\) with largest (abs. val.) coeff.

Return \(\Lambda\) or prune to top \(k\) pairs.

**Sample-Residual \((x, \Lambda, t, \sigma, K)\)**

for \(k = 1\) to \(K\) \{  
  \[u_k \leftarrow x(t + \sigma(k-1) \mod n)\]  
  sample arith. prog. from signal  
  \[v_k \leftarrow \sum_{(w, \alpha_w) \in \Lambda} (\alpha_w e^{\text{arith}(n) t + \text{nui}(k-1)})\]  
  non-uniform FFT
\}

Return \(u - v\rightarrow\) residual.
Identification \((K, A, \Lambda)\)

\(\text{reps} \leftarrow s\)  
\(\omega_k \leftarrow 0 \text{ for } k=1, 2, \ldots, K\)  
\(\text{initialize}\)

\(\sigma \sim \text{Unif}\{1, 3, 5, \ldots, n-1\}\)  
\(\text{random odd dilate factor (needs to be odd so it's invert. mod } 2^j\} \)

\(\text{for } b=0 \text{ to } \lfloor \log(n/2) \rfloor \)  
\(\text{loop from LSB to MSB}\)
\(\text{vote}_k \leftarrow 0 \text{ for } k=1, 2, \ldots, K\)

\(\text{for } j=1 \text{ to } \text{reps} \)  
\(\text{Draw } t \sim \text{Unif}\{0, 1, \ldots, n-1\}\)  
\(\text{random sample pt.}\)

\(u \leftarrow \text{Sample-Shattering}(\alpha, \Lambda, t, \sigma; K)\)
\(v \leftarrow \text{Sample-Shattering}(\alpha, \Lambda, t+\Lambda/2^{j+1}, \sigma; K)\)

\(\text{for } k=1 \text{ to } K\)  
\(E_0 \leftarrow u_k + e^{-\pi i \omega_k/2} v_k\)
\(E_1 \leftarrow u_k - e^{-\pi i \omega_k/2} v_k\)

\(\text{if } |E_1| > |E_0| \)  
\(\text{then } \text{vote}_k \leftarrow \text{vote}_k + 1\)

apply bit test to demod. sig.

\(\text{for } k=1 \text{ to } K\)  
\(\text{if } \text{vote}_k > \text{reps}/2 \)  
\(\omega_k \leftarrow \omega_k + 2^j\)  
\(\text{majority vote for bit value}\)

\(\text{Return unique}(\omega_1, \omega_2, \ldots, \omega_K)\)  
\(\text{remove duplicate fres}\)
\textbf{Sample-shattering}(x, \Lambda, t, \sigma, K)
\begin{align*}
Z & \leftarrow \text{Sample-residual}(x, \Lambda, t, \sigma, K) \\
Z & \leftarrow \text{FFT}(Z) \\
\text{return}\ (Z)
\end{align*}

\textbf{Estimation}(x, \Lambda, \Omega)
\begin{align*}
\text{reps} & \leftarrow 5 \\
\text{for} \ j = 1 \ \text{to} \ \text{reps} \ \text{do} \\
& \text{Draw} \ \sigma \sim \text{Unif}.\{1, 3, 5, \ldots, n-1\} \\
& t \sim \text{Unif}.\{10, 12, \ldots, n-1\} \\
Z & \leftarrow \text{sample-residual}(x, \Lambda, t, \sigma, K) \\
\text{for} \ l = 1 \ \text{to} \ |\Omega| \ \text{do} \\
& c_{\varepsilon}(j) = \sum_{k=1}^{K} a_k e^{\text{ }\pi i \left(\sigma_0 \zeta^k (k-1)/n\right)} \ \text{in parallel, FFT} \\
& c_{\varepsilon}(j) \leftarrow \left(\frac{1}{n}\right)^4 \frac{\text{demodulate and scale the est.}}{c_{\varepsilon}(j)} \\
\text{end for} \\
\varepsilon_{\text{est}}(j) \leftarrow \text{Median}\{c_{\varepsilon}(j) \mid j = 1, \ldots, \text{reps}\} \ \text{for} \ l = 1, 2, \ldots, |\Omega| \\
\text{return} \ c_1, c_2, \ldots, c_{|\Omega|} \ \text{helps with robustness.}