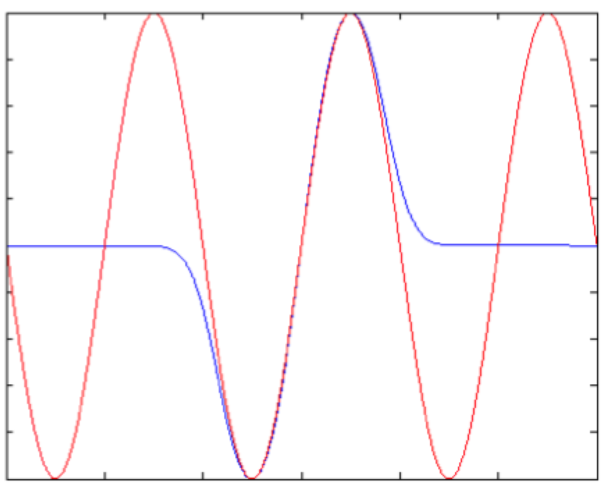
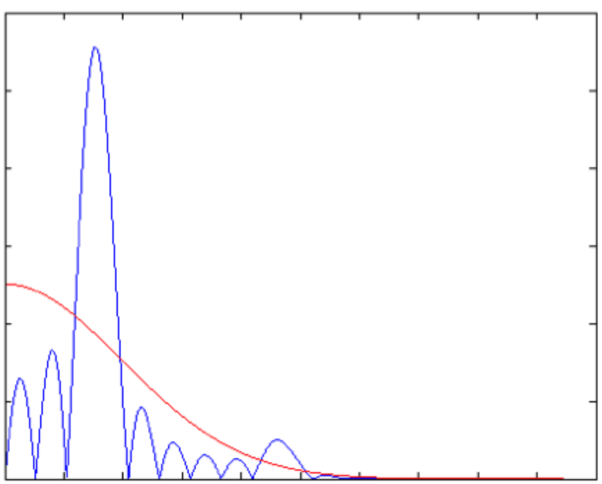
Spectral Methods for Neural Computation

Michael Lindsey
Boahen Lab Meeting
January 28, 2014





1. Outline

- What kinds of functions can be computed effectively with neurons?

1. Outline

- What kinds of functions can be computed effectively with neurons?

- Method for computing sinusoids

1. Outline

- What kinds of functions can be computed effectively with neurons?

- Method for computing sinusoids

- Robust to environmental changes

1. Outline

- What kinds of functions can be computed effectively with neurons?

- Method for computing sinusoids

- Robust to environmental changes

- Application: robot control

1. Outline

- What kinds of functions can be computed effectively with neurons?

- Method for computing sinusoids

- Robust to environmental changes

- Application: robot control

- Practical suggestions for neuromorphic engineering

1. Outline

- What kinds of functions can be computed effectively with neurons?

- Method for computing sinusoids

- Robust to environmental changes

- Application: robot control

- Practical suggestions for neuromorphic engineering

- (Analogous method for computing polynomials)

1. Outline

- What kinds of functions can be computed effectively with neurons?

- Method for computing sinusoids

- Robust to environmental changes

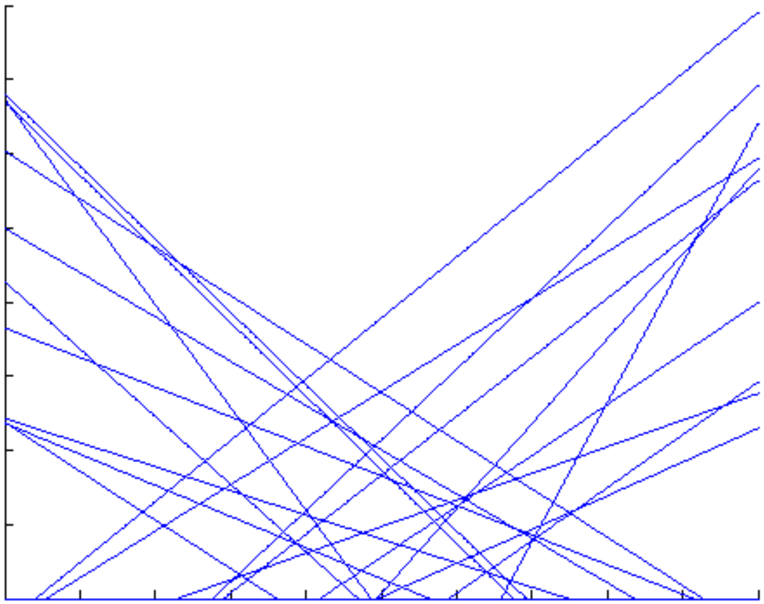
- Application: robot control

- Practical suggestions for neuromorphic engineering

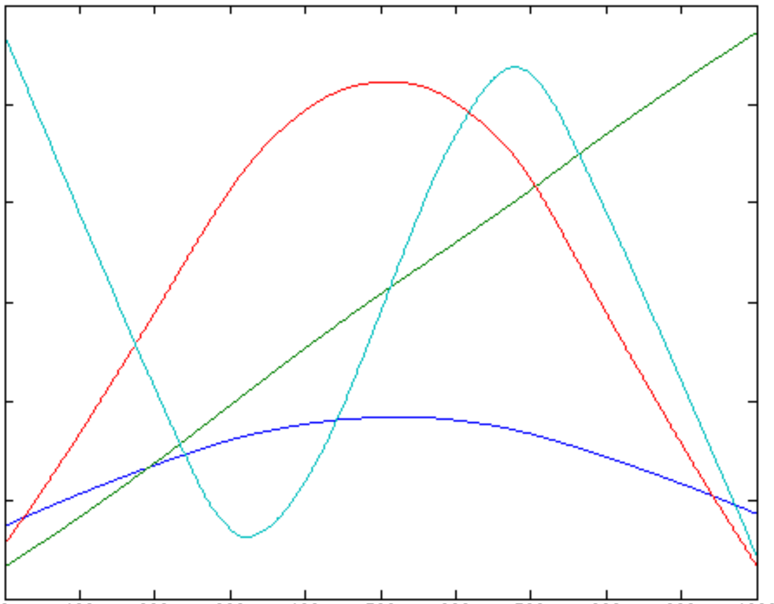
- (Analogous method for computing polynomials)

- (Application: numerical integration)

2. A Motivating Empirical Result

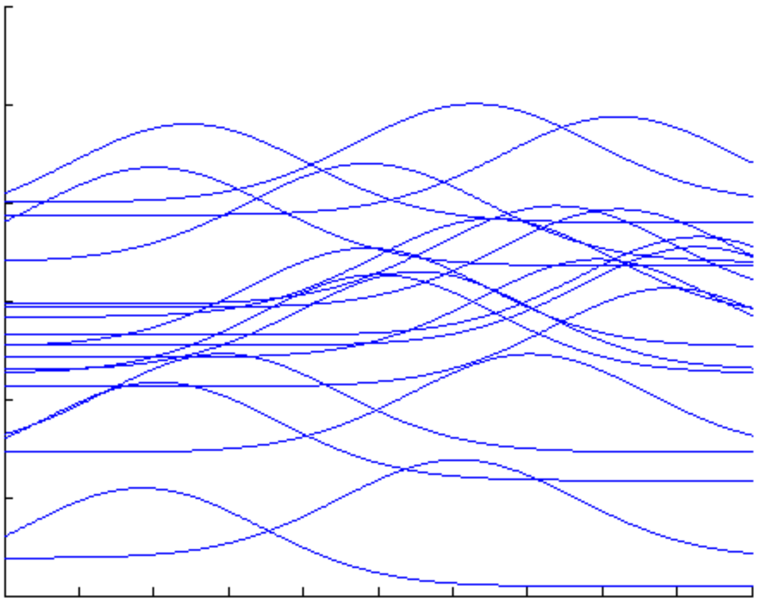


'Hinge' tuning curves

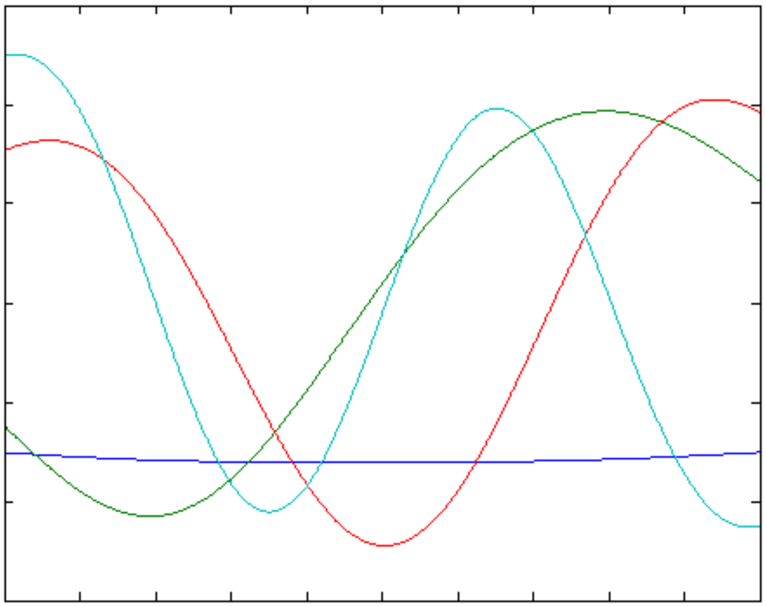


'Polynomial' basis

2. A Motivating Empirical Result



'Gaussian' tuning curves



'Fourier' basis

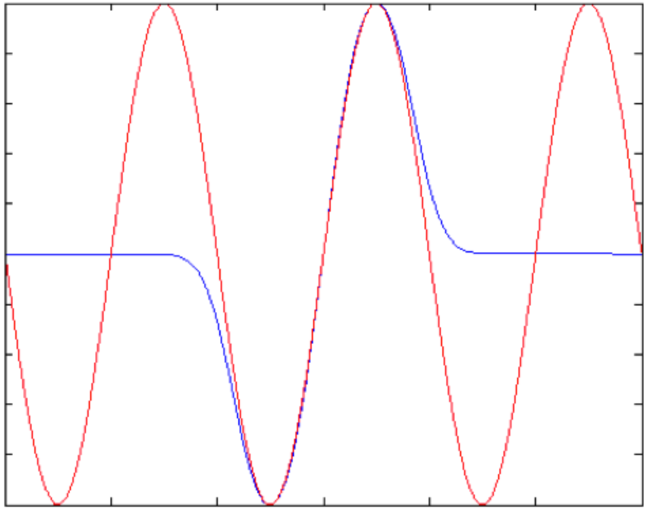
3. A Shot in the Dark

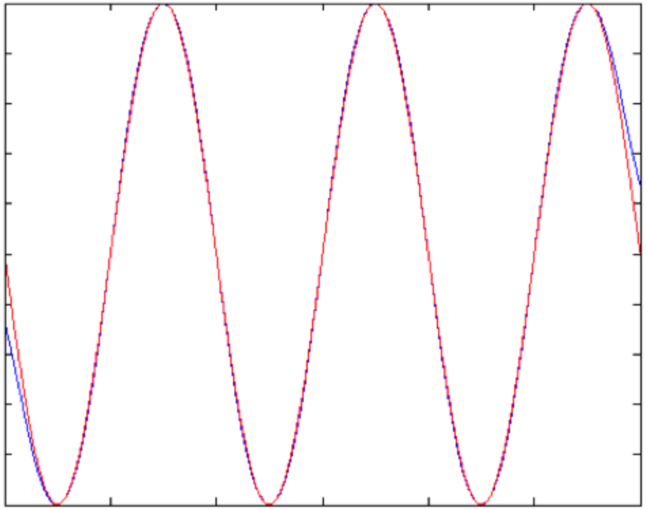
- Try adding up translated ($\pm$) Gaussian functions with extrema aligned with local extrema of sinusoid

3. A Shot in the Dark

- Try adding up translated (±) Gaussian functions with extrema aligned with local extrema of sinusoid

- Surprising result! But it's no accident...





4. The Fourier transform

- FT ($\mathcal{F}$): $\widehat{f}(\omega) = \mathcal{F}(f)(\omega) = \int_{\mathbb{R}} f(x)e^{-ix\omega}dx$

4. The Fourier transform

- FT ($\mathcal{F}$): $\widehat{f}(\omega) = \mathcal{F}(f)(\omega) = \int_{\mathbb{R}} f(x)e^{-ix\omega}dx$

- Inverse FT ($\mathcal{F}^{-1}$): $f(x) = \mathcal{F}^{-1}(\widehat{f})(x) = \frac{1}{2\pi}\int_{\mathbb{R}} \widehat{f}(\omega)e^{ix\omega}d\omega$

4. The Fourier transform

- FT ($\mathcal{F}$): $\widehat{f}(\omega) = \mathcal{F}(f)(\omega) = \int_{\mathbb{R}} f(x) e^{-ix\omega} dx$

- Inverse FT ($\mathcal{F}^{-1}$): $f(x) = \mathcal{F}^{-1}(\widehat{f})(x) = \frac{1}{2\pi} \int_{\mathbb{R}} \widehat{f}(\omega) e^{ix\omega} d\omega$

- Property 1: $\mathcal{F}$ and $\mathcal{F}^{-1}$ are linear operators

4. The Fourier transform

- FT ($\mathcal{F}$): $\widehat{f}(\omega) = \mathcal{F}(f)(\omega) = \int_{\mathbb{R}} f(x) e^{-ix\omega} dx$

- Inverse FT ($\mathcal{F}^{-1}$): $f(x) = \mathcal{F}^{-1}(\widehat{f})(x) = \frac{1}{2\pi} \int_{\mathbb{R}} \widehat{f}(\omega) e^{ix\omega} d\omega$

- Property 1: $\mathcal{F}$ and $\mathcal{F}^{-1}$ are linear operators

- Property 2 (translation): If $f_T(x) = f(x - T)$, then $\widehat{f_T}(\omega) = e^{-i\omega T} \widehat{f}(\omega)$

4. The Fourier transform

- FT ($\mathcal{F}$): $\widehat{f}(\omega) = \mathcal{F}(f)(\omega) = \int_{\mathbb{R}} f(x)e^{-ix\omega}dx$

- Inverse FT ($\mathcal{F}^{-1}$): $f(x) = \mathcal{F}^{-1}(\widehat{f})(x) = \frac{1}{2\pi}\int_{\mathbb{R}} \widehat{f}(\omega)e^{ix\omega}d\omega$

- Property 1: $\mathcal{F}$ and $\mathcal{F}^{-1}$ are linear operators

- Property 2 (translation): If $f_T(x) = f(x-T)$, then $\widehat{f_T}(\omega) = e^{-i\omega T}\widehat{f}(\omega)$

- We say that a function f is Schwartz if f is smooth (infinitely differentiable) and if f and all of its derivatives decay faster than any polynomial (e.g., the Gaussian function, any smooth function of compact support)

4. The Fourier transform

- FT ($\mathcal{F}$): $\widehat{f}(\omega) = \mathcal{F}(f)(\omega) = \int_{\mathbb{R}} f(x)e^{-ix\omega}dx$

- Inverse FT ($\mathcal{F}^{-1}$): $f(x) = \mathcal{F}^{-1}(\widehat{f})(x) = \frac{1}{2\pi}\int_{\mathbb{R}} \widehat{f}(\omega)e^{ix\omega}d\omega$

- Property 1: $\mathcal{F}$ and $\mathcal{F}^{-1}$ are linear operators

- Property 2 (translation): If $f_T(x) = f(x-T)$, then $\widehat{f_T}(\omega) = e^{-i\omega T}\widehat{f}(\omega)$

- We say that a function f is Schwartz if f is smooth (infinitely differentiable) and if f and all of its derivatives decay faster than any polynomial (e.g., the Gaussian function, any smooth function of compact support)

- Property 3: $\mathcal{F}$ and $\mathcal{F}^{-1}$ map Schwartz functions to Schwartz functions (in fact, FT of Gaussian is Gaussian)

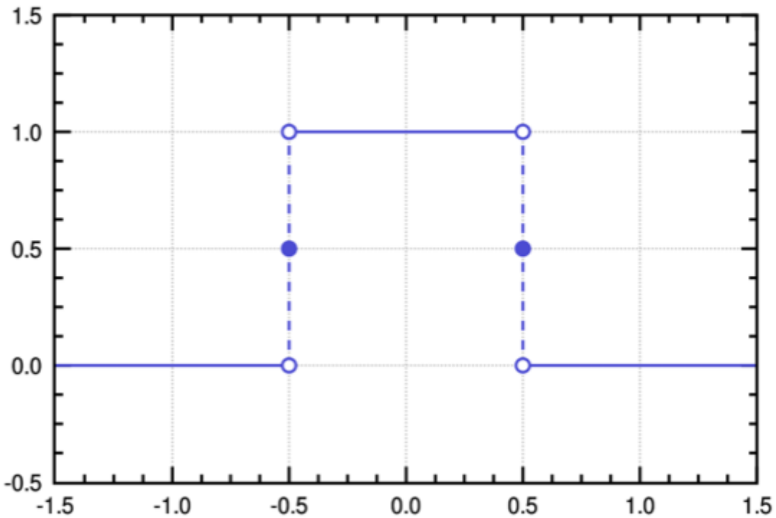
- Notionally: smoothness in spatial domain $\leftrightarrow$ decay in frequency domain (also, decay in spatial domain $\leftrightarrow$ smoothness in frequency domain)

- Notionally: smoothness in spatial domain $\leftrightarrow$ decay in frequency domain (also, decay in spatial domain $\leftrightarrow$ smoothness in frequency domain)

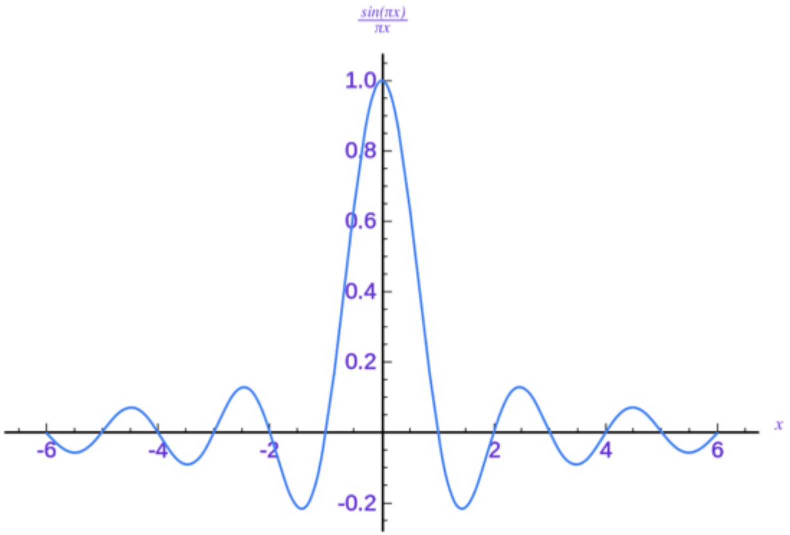
- Notionally: lack of smoothness (box function) $\leftrightarrow$ slow decay (sinc function)

- Notionally: smoothness in spatial domain $\leftrightarrow$ decay in frequency domain (also, decay in spatial domain $\leftrightarrow$ smoothness in frequency domain)

- Notionally: lack of smoothness (box function) $\leftrightarrow$ slow decay (sinc function)



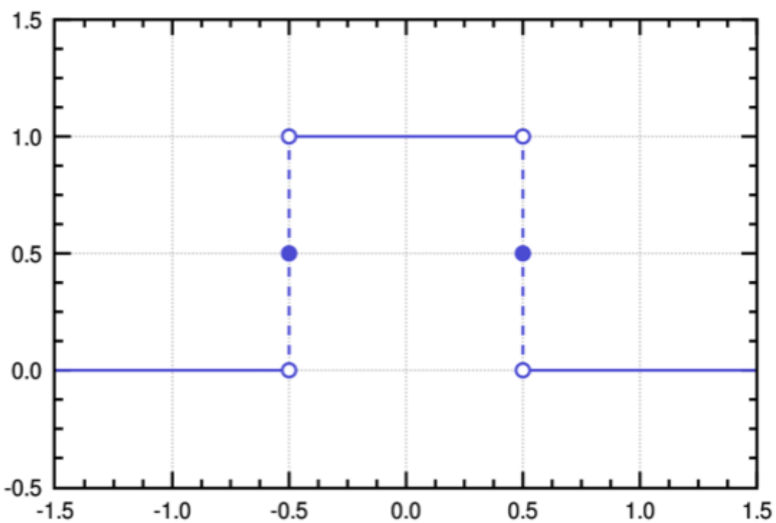
Discontinuity



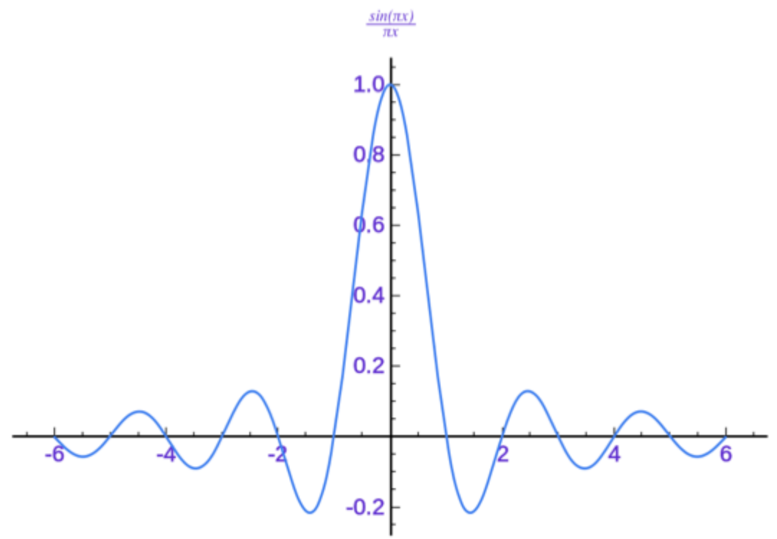
Slow decay

- Notionally: smoothness in spatial domain $\leftrightarrow$ decay in frequency domain (also, decay in spatial domain $\leftrightarrow$ smoothness in frequency domain)

- Notionally: lack of smoothness (box function) $\leftrightarrow$ slow decay (sinc function)



Discontinuity



Slow decay

-Property 4 (scaling): If $f_a(x) = f(\frac{x}{a})$, then $\widehat{f_a}(\omega) = |a|\widehat{f}(a\omega)$

5. Precise Statement for Constructing Sinusoids

- Let g be a Schwartz function. Let $x_k^{(+)} = 1 + 4k$, $x_k^{(-)} = -1 + 4k$. Let $g_k^{(+)}(x) = g(x - x_k^{(+)})$ and $g_k^{(-)}(x) = g(x - x_k^{(-)})$

5. Precise Statement for Constructing Sinusoids

- Let g be a Schwartz function. Let $x_k^{(+)} = 1 + 4k$, $x_k^{(-)} = -1 + 4k$. Let $g_k^{(+)}(x) = g(x - x_k^{(+)})$ and $g_k^{(-)}(x) = g(x - x_k^{(-)})$

- Let $f_N = \left(g_0^{(+)} - g_0^{(-)}\right) + \sum_{k=1}^{N} \left(g_k^{(+)} - g_k^{(-)} + g_{-k}^{(+)} - g_{-k}^{(-)}\right)$

5. Precise Statement for Constructing Sinusoids

- Let g be a Schwartz function. Let $x_k^{(+)} = 1 + 4k$, $x_k^{(-)} = -1 + 4k$. Let $g_k^{(+)}(x) = g(x - x_k^{(+)})$ and $g_k^{(-)}(x) = g(x - x_k^{(-)})$

- Let $f_N = \left(g_0^{(+)} - g_0^{(-)}\right) + \sum_{k=1}^{N} \left(g_k^{(+)} - g_k^{(-)} + g_{-k}^{(+)} - g_{-k}^{(-)}\right)$

- Then for all $x \in \mathbb{R}$, as $N \to \infty$,

$$f_N(x) \to \sum_{k=0}^{\infty} (-1)^k \left[a_k \sin\left(\left(\frac{\pi}{2} + k\pi\right)x\right) - b_k \cos\left(\left(\frac{\pi}{2} + k\pi\right)x\right)\right]$$

5. Precise Statement for Constructing Sinusoids

- Let g be a Schwartz function. Let $x_k^{(+)} = 1 + 4k$, $x_k^{(-)} = -1 + 4k$. Let $g_k^{(+)}(x) = g(x - x_k^{(+)})$ and $g_k^{(-)}(x) = g(x - x_k^{(-)})$
- Let $f_N = \left(g_0^{(+)} - g_0^{(-)}\right) + \sum_{k=1}^{N} \left(g_k^{(+)} - g_k^{(-)} + g_{-k}^{(+)} - g_{-k}^{(-)}\right)$
- Then for all $x \in \mathbb{R}$, as $N \to \infty$,

$$\begin{aligned} f_N(x) \to &\sum_{k=0}^{\infty}(-1)^k \left[a_k \sin\left(\left(\frac{\pi}{2} + k\pi\right)x\right) - b_k \cos\left(\left(\frac{\pi}{2} + k\pi\right)x\right)\right] \\ = \; &a_0 \sin\left(\frac{\pi}{2}x\right) - b_0 \cos\left(\frac{\pi}{2}x\right) + \dots \end{aligned}$$

5. Precise Statement for Constructing Sinusoids

- Let g be a Schwartz function. Let $x_k^{(+)} = 1 + 4k$, $x_k^{(-)} = -1 + 4k$. Let $g_k^{(+)}(x) = g(x - x_k^{(+)})$ and $g_k^{(-)}(x) = g(x - x_k^{(-)})$

- Let $f_N = \left(g_0^{(+)} - g_0^{(-)}\right) + \sum_{k=1}^{N} \left(g_k^{(+)} - g_k^{(-)} + g_{-k}^{(+)} - g_{-k}^{(-)}\right)$

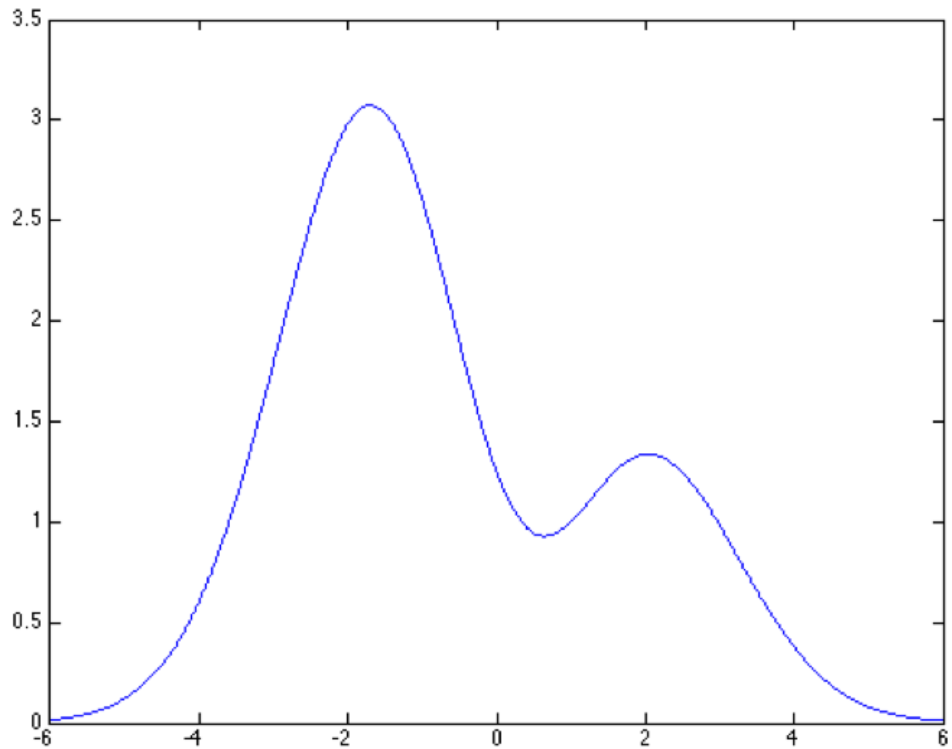
- Then for all $x \in \mathbb{R}$, as $N \to \infty$,

$$
\begin{aligned}
f_N(x) \to & \sum_{k=0}^{\infty}(-1)^k \left[a_k \sin\left(\left(\frac{\pi}{2} + k\pi\right)x\right) - b_k \cos\left(\left(\frac{\pi}{2} + k\pi\right)x\right)\right] \\
= & \ a_0 \sin\left(\frac{\pi}{2}x\right) - b_0 \cos\left(\frac{\pi}{2}x\right) + \ldots
\end{aligned}
$$

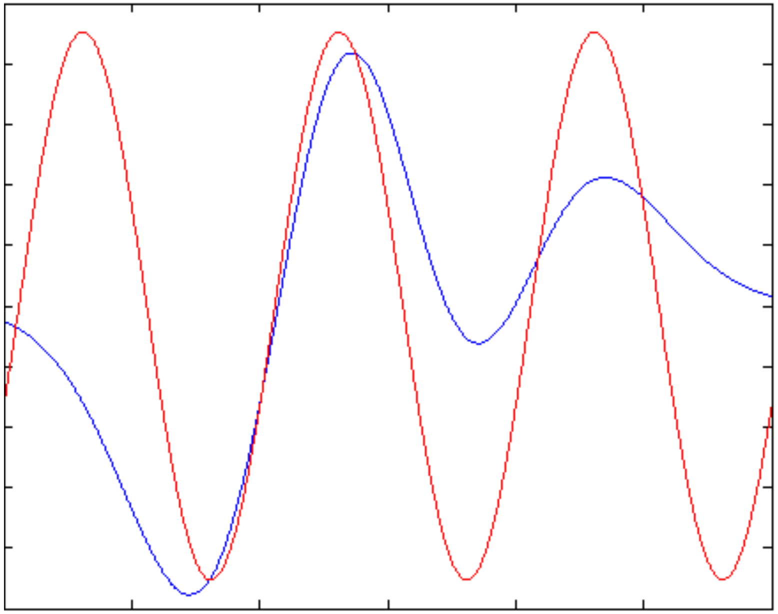
where $a_k = \Re\left(\widehat{g}\left(\frac{\pi}{2} + k\pi\right)\right)$, $b_k = \Im\left(\widehat{g}\left(\frac{\pi}{2} + k\pi\right)\right)$ for all k.

6. A Surprising Consequence

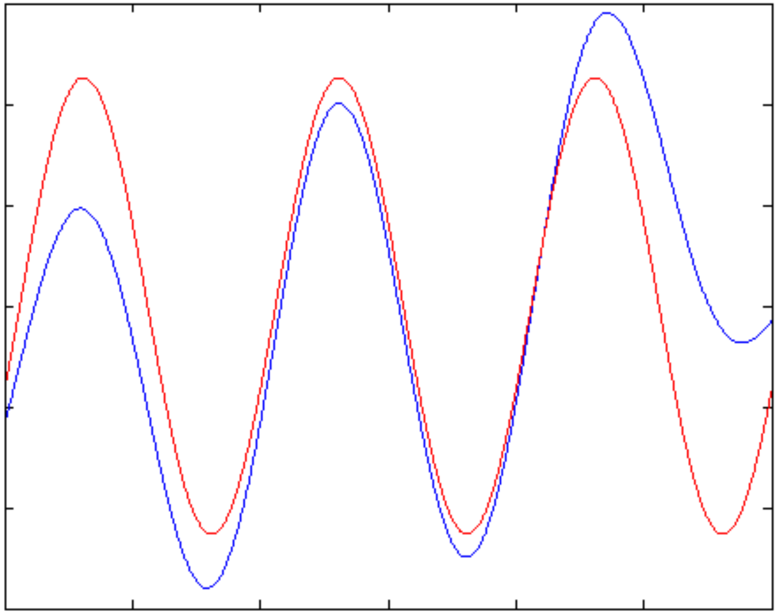
- We do not require that the tuning curve g have a single local extremum



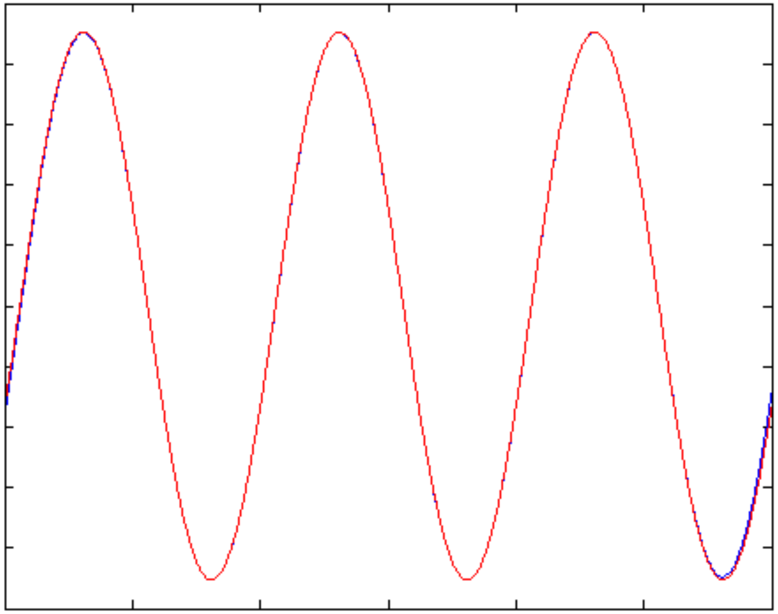
Weird tuning curve



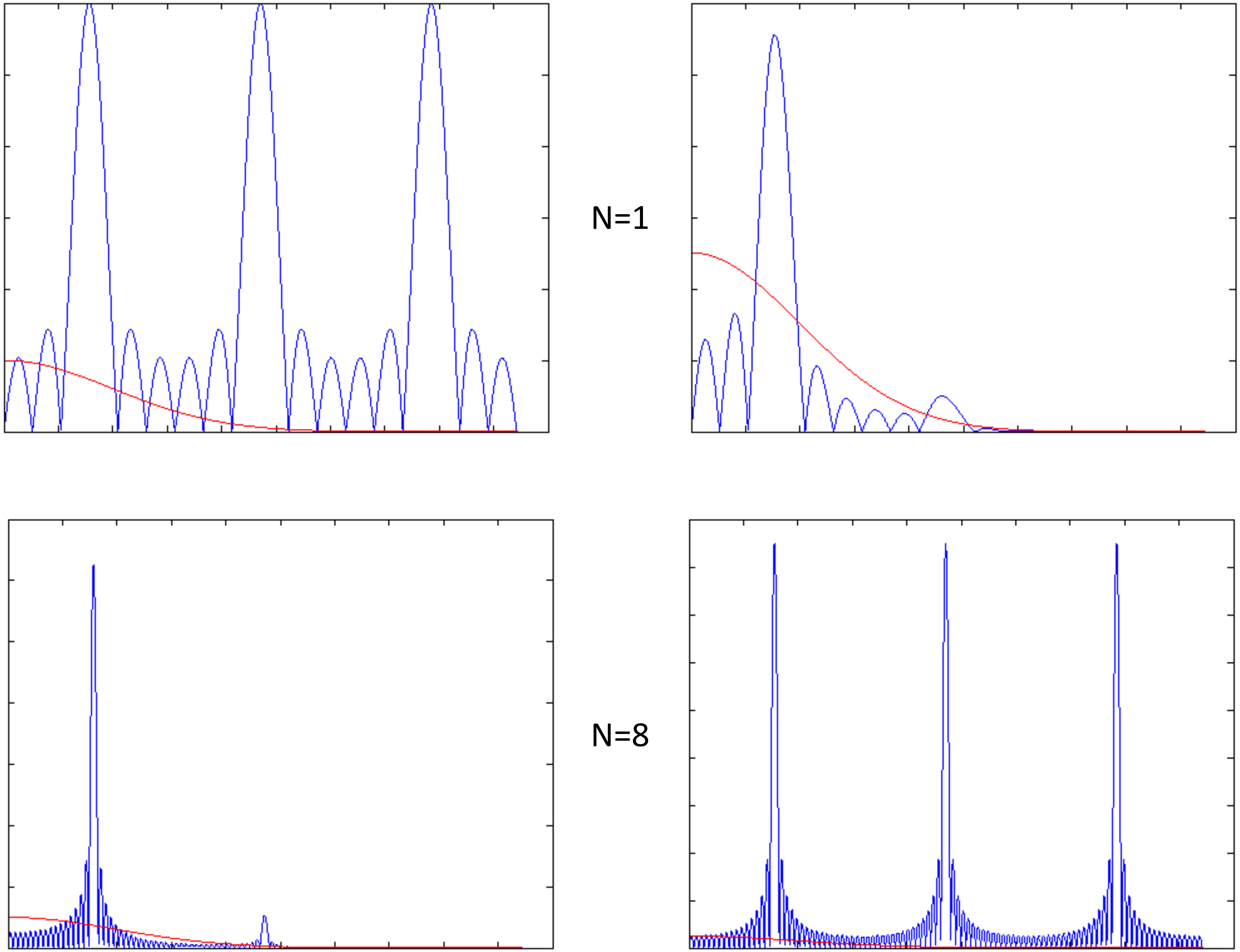
2 neurons



6 neurons



10 neurons


N=1
N=8

- Since g is Schwartz, $\widehat{g}$ is also Schwartz, so by a sufficiently large horizontal scaling of g, we can get $a_0 \gg a_k$ for all $k \geq 1$

- Since g is Schwartz, $\widehat{g}$ is also Schwartz, so by a sufficiently large horizontal scaling of g, we can get $a_0 \gg a_k$ for all $k \geq 1$

- In this case, $f_N(x) \approx a_0 \sin\left(\frac{\pi}{2}x\right)$ for N large enough.

- Since g is Schwartz, $\widehat{g}$ is also Schwartz, so by a sufficiently large horizontal scaling of g, we can get $a_0 \gg a_k$ for all $k \geq 1$

- In this case, $f_N(x) \approx a_0 \sin\left(\frac{\pi}{2}x\right)$ for N large enough.

- Can extend to the case where g is continuous and decays faster than x^{-1} (proof expresses g as a limit of Schwartz functions)

- Since g is Schwartz, $\widehat{g}$ is also Schwartz, so by a sufficiently large horizontal scaling of g, we can get $a_0 \gg a_k$ for all $k \geq 1$

- In this case, $f_N(x) \approx a_0 \sin\left(\frac{\pi}{2}x\right)$ for N large enough.

- Can extend to the case where g is continuous and decays faster than x^{-1} (proof expresses g as a limit of Schwartz functions)

- However, cannot guarantee that $a_0 \gg a_k$ for all $k \geq 1$. How to guarantee rapidly decaying Fourier transform?

7. Mollifying Pathological Tuning Curves

- We can mollify (smooth out) functions by convolving them with a smooth function of compact support

7. Mollifying Pathological Tuning Curves

- We can mollify (smooth out) functions by convolving them with a smooth function of compact support

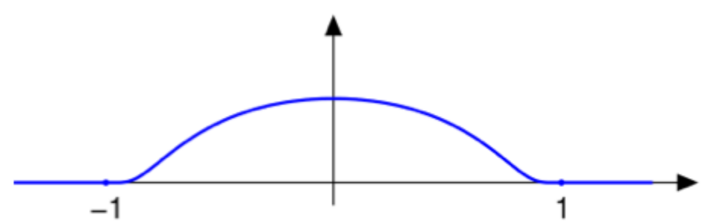
- This is like replacing the value of the function at each point with a smooth weighted average of the values at its neighboring points

7. Mollifying Pathological Tuning Curves

- We can mollify (smooth out) functions by convolving them with a smooth function of compact support

- This is like replacing the value of the function at each point with a smooth weighted average of the values at its neighboring points

- For example, take mollifier, $\varphi(x) = e^{\frac{-1}{1-|x|^2}} \mathbb{I}_{|x|<1}$

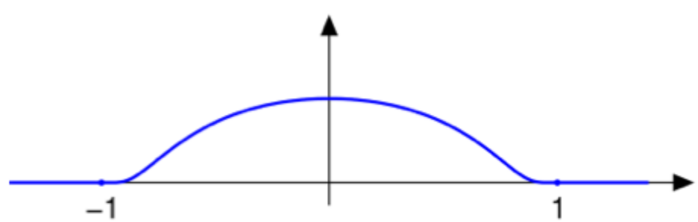


7. Mollifying Pathological Tuning Curves

- We can mollify (smooth out) functions by convolving them with a smooth function of compact support

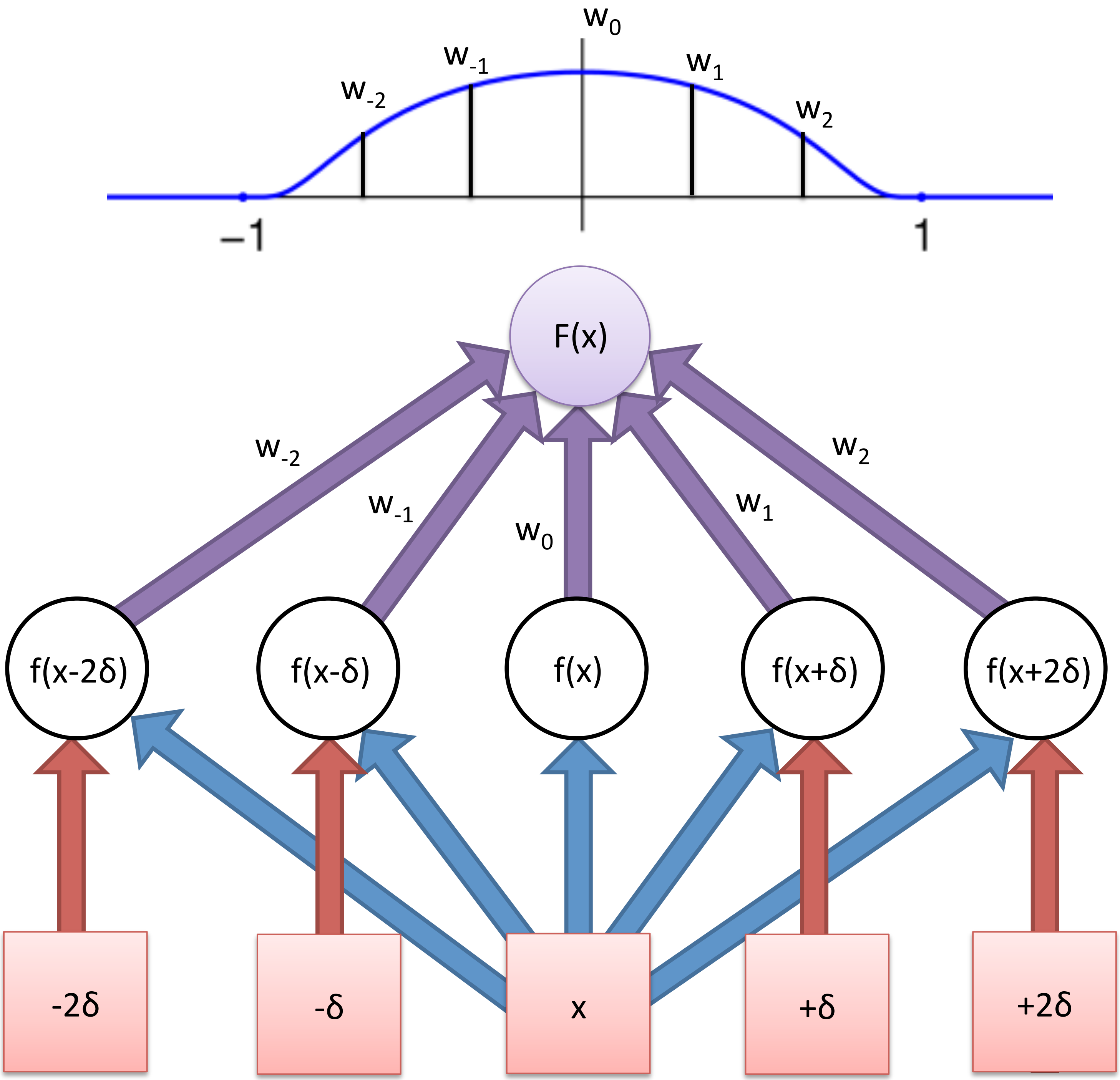
- This is like replacing the value of the function at each point with a smooth weighted average of the values at its neighboring points

- For example, take mollifier, $\varphi(x) = e^{\frac{-1}{1-|x|^2}} \mathbb{I}_{|x|<1}$



- A discrete mollification can be carried out by a simple neural network:

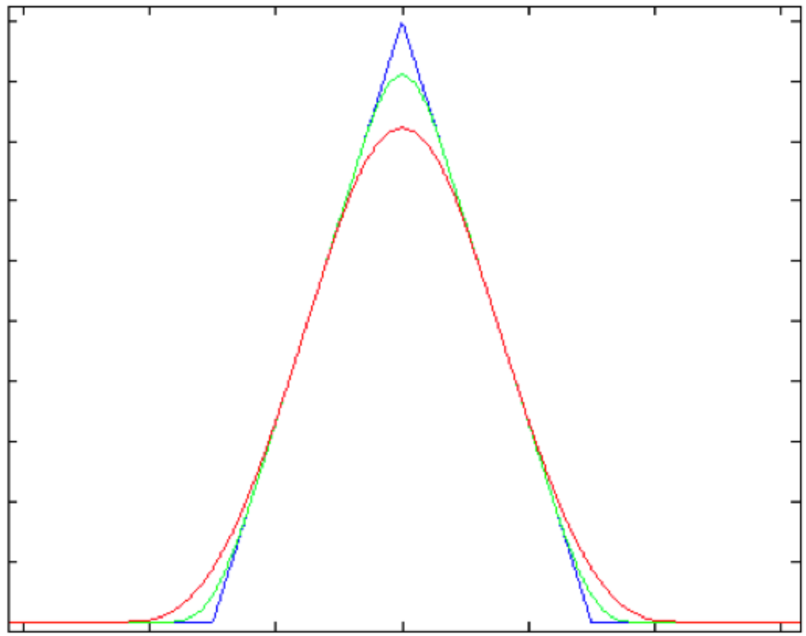
$$\widetilde{f}(x) = \left(\sum_{j=-n+1}^{n-1} \varphi\left(\frac{j}{n}\right)\right)^{-1} \sum_{j=-n+1}^{n-1} \varphi\left(\frac{j}{n}\right) f\left(x - j\delta\right)$$



w_0
w_{-1}
w_1
w_{-2}
w_2
−1
1
F(x)
w_{-2}
w_2
w_{-1}
w_1
w_0
f(x-2δ)
f(x-δ)
f(x)
f(x+δ)
f(x+2δ)
-2δ
-δ
x
+δ
+2δ

- We demonstrate this strategy on a nasty tuning curve (hat function)

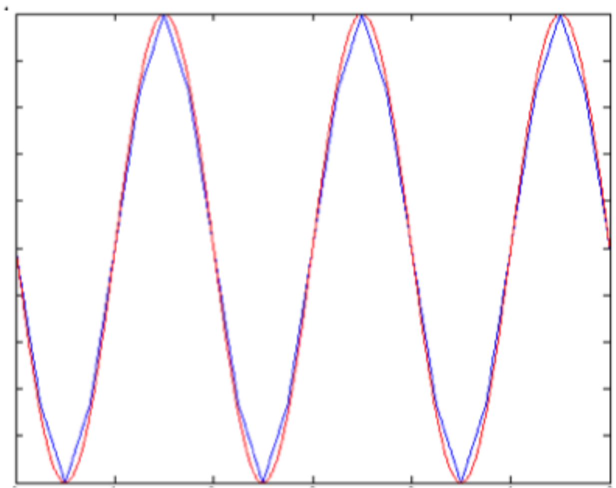
- We demonstrate this strategy on a nasty tuning curve (hat function)



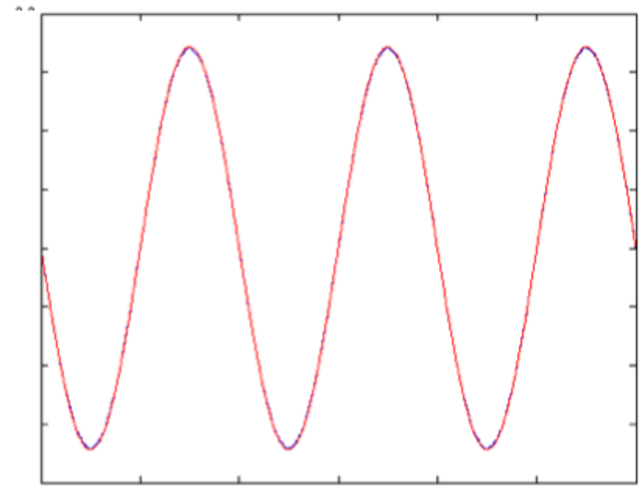
Mollified hat functions obtained from above procedure (with $\delta = 0.1$)

Blue: no mollification. Green: $n = 4$ (convex combination of 7 hat functions). Red: $n = 8$ (15 hat functions)

Approximation using no mollification (left), mollification with $\delta = 0.3$, $n = 4$ (right)

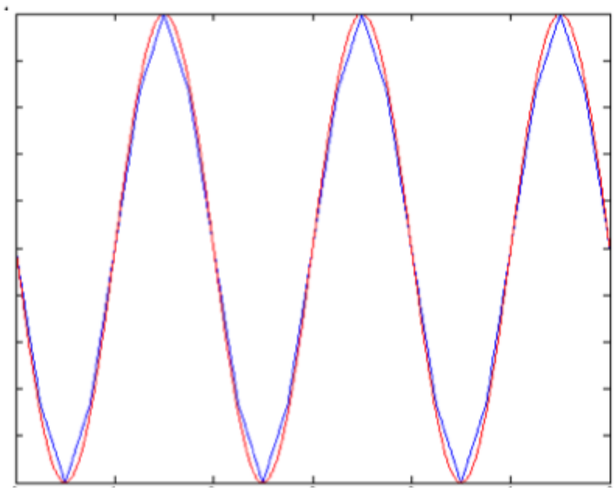


Without smoothing

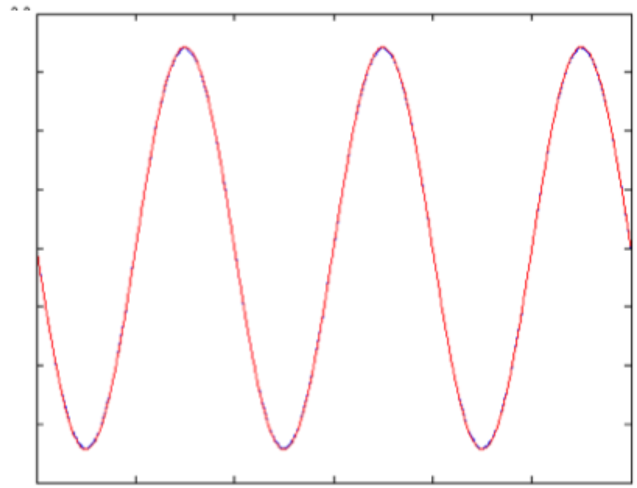


With smoothing

Approximation using no mollification (left), mollification with $\delta = 0.3$, $n = 4$ (right)



Without smoothing



With smoothing

n	0	4	8
L^2 error	1.3×10^{-3}	6.7×10^{-5}	2.9×10^{-5}
L^∞ error	0.0912	0.0065	0.0024

So to approximate one period of a sinusoid, we require about 14 hat-shaped tuning curves (as opposed to 2 Gaussian tuning curves)

- We know that this strategy will work in general because of the...

- We know that this strategy will work in general because of the...

Convolution theorem:

$$\mathcal{F}(f * g) = \mathcal{F}(f)\mathcal{F}(g)$$

- We know that this strategy will work in general because of the...

Convolution theorem:

$$\mathcal{F}(f * g) = \mathcal{F}(f)\mathcal{F}(g)$$

- Since a mollifier φ is Schwartz, $\mathcal{F}(\varphi)$ is Schwartz, and convolution with φ multiplies the frequency spectrum of our tuning curve by a rapidly decaying function

- We know that this strategy will work in general because of the...

Convolution theorem:

$$\mathcal{F}(f * g) = \mathcal{F}(f)\mathcal{F}(g)$$

- Since a mollifier φ is Schwartz, $\mathcal{F}(\varphi)$ is Schwartz, and convolution with φ multiplies the frequency spectrum of our tuning curve by a rapidly decaying function

- For a sufficiently wide mollifier, $\mathcal{F}(\varphi)$ is localized enough to make our approximation hold with negligible error

- We know that this strategy will work in general because of the...

Convolution theorem:

$$\mathcal{F}(f * g) = \mathcal{F}(f)\mathcal{F}(g)$$

- Since a mollifier φ is Schwartz, $\mathcal{F}(\varphi)$ is Schwartz, and convolution with φ multiplies the frequency spectrum of our tuning curve by a rapidly decaying function

- For a sufficiently wide mollifier, $\mathcal{F}(\varphi)$ is localized enough to make our approximation hold with negligible error

- We may need to choose sample spacing δ smaller for more irregular tuning curve shapes

8. What is the Optimal Tuning Curve?

8. What is the Optimal Tuning Curve?

- We want a tuning curve which is as localized as possible in both spatial and frequency domains

8. What is the Optimal Tuning Curve?

- We want a tuning curve which is as localized as possible in both spatial and frequency domains

- We can actually suggest an answer in a certain sense

8. What is the Optimal Tuning Curve?

- We want a tuning curve which is as localized as possible in both spatial and frequency domains

- We can actually suggest an answer in a certain sense (the Gaussian)

8. What is the Optimal Tuning Curve?

- We want a tuning curve which is as localized as possible in both spatial and frequency domains

- We can actually suggest an answer in a certain sense (the Gaussian)

- For $f \in L^2(\mathbb{R})$, let $P(t) = \frac{|f(t)|^2}{\|f\|_2^2}$ (so P is a pdf), and

$$\sigma^2(f) := \inf_{t_0} \int_{\mathbb{R}} (t - t_0)^2 P(t) dt,$$

so $\sigma(f)$ is the standard deviation of an RV with density P, $\frac{1}{\sigma(f)}$ measures the localization of f

8. What is the Optimal Tuning Curve?

- We want a tuning curve which is as localized as possible in both spatial and frequency domains

- We can actually suggest an answer in a certain sense (the Gaussian)

- For $f \in L^2(\mathbb{R})$, let $P(t) = \frac{|f(t)|^2}{\|f\|_2^2}$ (so P is a pdf), and

$$\sigma^2(f) := \inf_{t_0} \int_{\mathbb{R}} (t - t_0)^2 P(t) dt,$$

so $\sigma(f)$ is the standard deviation of an RV with density P, $\frac{1}{\sigma(f)}$ measures the localization of f

Weyl-Heisenberg Uncertainty Principle:

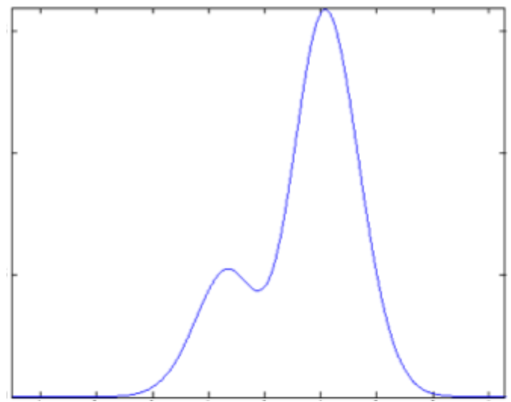
$\sigma(f)\sigma(\widehat{f}) \geq \frac{1}{2}$, with equality if and only if f is a Gaussian

Review

- We can build sinusoids from smooth, rapidly decaying tuning curves
- It's okay if the tuning curves have many peaks
- …but Gaussians are the best
- We can deal with non-smooth tuning curves
- Network structure itself encodes computation
- Robust to modification of tuning curve
- Sinusoids as basis

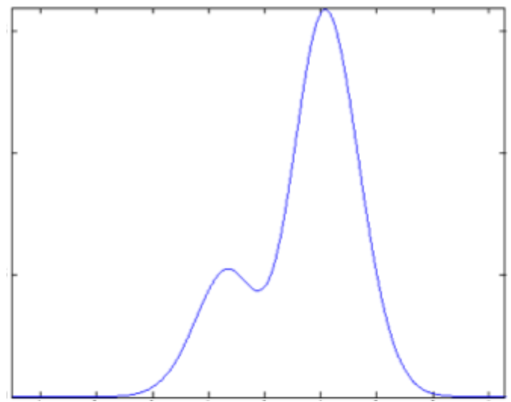
Application: Statistical Inference

- Take $g(x) = (2x^2 + 0.5)e^{-(x-0.32)^2}$



Application: Statistical Inference

- Take $g(x) = (2x^2 + 0.5)e^{-(x-0.32)^2}$

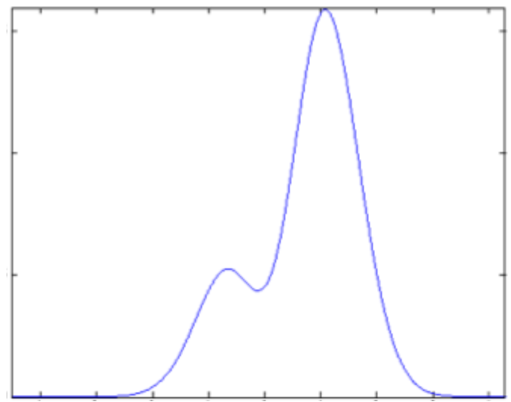


- We approximate the p-th moment of g by $\sum_{n=-3}^{3} n^p g(n)$

$$\sum_{n=-3}^{3} n^p g(n) \quad \int_{\infty}^{\infty} u^p g(u) du$$

Application: Statistical Inference

- Take $g(x) = (2x^2 + 0.5)e^{-(x-0.32)^2}$



- We approximate the p-th moment of g by $\sum_{n=-3}^{3} n^p g(n)$

	$\sum_{n=-3}^{3} n^p g(n)$	$\int_{\infty}^{\infty} u^p g(u) du$
$p = 0$	3.02	3.02
$p = 1$	2.11	2.10
$p = 2$	4.32	4.32
$p = 3$	5.24	5.30

9. Approximating Polynomials

9. Approximating Polynomials

Theorem. *Let g be a Schwartz function, and for all $n \in \mathbb{Z}$ let g_n be the function defined by $g_n(x) = g(x - n)$.*

9. Approximating Polynomials

Theorem. *Let g be a Schwartz function, and for all $n \in \mathbb{Z}$ let g_n be the function defined by $g_n(x) = g(x-n)$. Define*

$$f_N = \sum_{n=-N}^{N} n^p g_n,$$

where p is a non-negative even integer.

9. Approximating Polynomials

Theorem. *Let g be a Schwartz function, and for all $n \in \mathbb{Z}$ let g_n be the function defined by $g_n(x) = g(x-n)$. Define*

$$f_N = \sum_{n=-N}^{N} n^p g_n,$$

where p is a non-negative even integer. Then for all $x \in \mathbb{R}$,

$$f_N(x) \to \sum_{n=0}^{p} c_n(x) x^n$$

as $N \to \infty$

9. Approximating Polynomials

Theorem. *Let g be a Schwartz function, and for all $n \in \mathbb{Z}$ let g_n be the function defined by $g_n(x) = g(x-n)$. Define*

$$f_N = \sum_{n=-N}^{N} n^p g_n,$$

where p is a non-negative even integer. Then for all $x \in \mathbb{R}$,

$$f_N(x) \to \sum_{n=0}^{p} c_n(x) x^n$$

as $N \to \infty$, where

$$c_n(x) = i^{n-p} \binom{p}{n} \sum_{k \in \mathbb{Z}} \widehat{g}^{(p-n)}(2\pi k) e^{2\pi i k x}.$$

9. Approximating Polynomials

Theorem. *Let g be a Schwartz function, and for all $n \in \mathbb{Z}$ let g_n be the function defined by $g_n(x) = g(x-n)$. Define*

$$f_N = \sum_{n=-N}^{N} n^p g_n,$$

where p is a non-negative even integer. Then for all $x \in \mathbb{R}$,

$$f_N(x) \to \sum_{n=0}^{p} c_n(x) x^n$$

as $N \to \infty$, where

$$c_n(x) = i^{n-p} \binom{p}{n} \sum_{k \in \mathbb{Z}} \widehat{g}^{(p-n)}(2\pi k) e^{2\pi i k x}.$$

In particular, by modifying g with an appropriate horizontal scaling if necessary, we obtain the approximation (for large enough N) $f_N(x) \approx \sum_{n=0}^{p} c_n x^n$, where $c_n = \int u^{p-n} g(u) du$, so c_n are constants and f_N is approximately a polynomial.

Application: Robust Robot Control

- Robot control demands the computation of functions in joint positions $q_0, \ldots, q_n$

Application: Robust Robot Control

- Robot control demands the computation of functions in joint positions $q_0, \ldots, q_n$

- Generally these functions are products of functions q_i, $\sin(q_i)$, $\cos(q_i)$

Application: Robust Robot Control

- Robot control demands the computation of functions in joint positions $q_0, \ldots, q_n$

- Generally these functions are products of functions q_i, $\sin(q_i)$, $\cos(q_i)$

- Note that since we can square things, we can multiply things, due to the fact that $xy = \frac{1}{2}((x+y)^2 - x^2 - y^2)$

Application: Robust Robot Control

- Robot control demands the computation of functions in joint positions $q_0, \ldots, q_n$

- Generally these functions are products of functions q_i, $\sin(q_i)$, $\cos(q_i)$

- Note that since we can square things, we can multiply things, due to the fact that $xy = \frac{1}{2}((x+y)^2 - x^2 - y^2)$

- Thus we are equipped to do robot control using the above methods with explicit error bounds

Conclusions

- smoothness allows for discrete approach to continuous problems
- spectral intuition
- efficient, robust, general

Future work

- spike-based model
- heterogeneity
- time domain
- hardware-specific considerations