Solving the mystery of Magic Kernel Sharp

John P. Costella
770 5th St NW Apt 1207, Washington, DC 20001-2673, United States

(February 26, 2021)

Abstract

In this paper I derive the Fourier transforms of the Magic Kernel and the Magic Kernel Sharp kernel analytically. The remarkable results reveal that Magic Kernel Sharp is actually the third of a natural sequence of resizing kernels, the first two being nearest neighbor and linear interpolation, and explain why Magic Kernel Sharp does not share the fatal flaws of the popular Lanczos kernels—and is actually more computationally efficient. Finally, I explicitly construct the fourth, fifth, and sixth members of this sequence of kernels, the last of which has the same efficiency as the Lanczos-3 kernel, and show that their fidelity and efficiency should satisfy the requirements of even the most exacting precision use cases.

1. Introduction

The Magic Kernel Sharp algorithm has been used by Facebook to resize images since 2013, and by Instagram since 2015 [1]. To better understand why it provides such remarkably clear results, in 2015 I numerically analyzed its Fourier properties in depth [2], and found that it possesses high-order zeros at integral multiples of the sampling frequency, which highly suppress low-frequency aliasing artifacts. However, that numerical analysis did not fully explain the theoretical mystery of *how* or *why* it possesses such remarkable properties.

In recent weeks I realized that it would be challenging, but not impossible, to analytically derive the Fourier transform of the Magic Kernel itself in closed form. This paper contains the extremely surprising conclusion of that analysis: *the Magic Kernel is simply the convolution of the rectangular window function with itself, twice*, and hence its Fourier transform is simply $\text{sinc}^3 f$, which immediately explains its third-order zeros at integral values of f.

Armed with this insight, I realized that the rectangular window function can itself be thought of as the nearest neighbor kernel, with Fourier transform $\text{sinc}\, f$; and the convolution of the rectangular window function with itself—the "tent" function—is simply the linear interpolation kernel, with Fourier transform $\text{sinc}^2 f$. The first of these has discontinuities at its endpoints, and significant spectral energy outside the first Nyquist zone. The second is continuous everywhere, and has significantly less spectral energy outside the first Nyquist zone, but it still has discontinuous first derivative at its endpoints and midpoint. The third kernel in this fundamental sequence, the Magic Kernel, is continuous and has continuous first derivative everywhere, and has even less spectral energy outside the first Nyquist zone. However, unlike its two predecessors, its support in position space extends beyond the two neighboring sampling positions, which means that it will slightly blur the input if it is used as an interpolation kernel.

The correction for this blurring is the convolution of the Magic Kernel with a discrete kernel that undoes this blurring: the Sharp kernel. The combined Magic Kernel Sharp kernel represents the natural third member of this fundamental sequence of resizing kernels, with properties in both position space and Fourier space that make it arguably the best resizing kernel for practical purposes: the first and second approximations to the exact Sharp kernel have tiny support, requiring only three and six coefficients respectively, with the latter matching the exact kernel to nine bits of accuracy—more than enough precision for most practical applications.

Now, even though the observations above mean that it is (now) a two-line proof to obtain the Fourier transform of the Magic Kernel, in the following sections I still provide the outline for how I derived it analytically from nothing more than the position-space expression for it that I deduced back in 2011 [3], as that discussion leads naturally into a precise specification of the exact Sharp kernel and its Fourier transform.

I then derive the Fourier transforms of the Lanczos kernels, and use the results to explain the fundamental flaws possessed by these popular kernels, which do not similarly plague Magic Kernel Sharp. I also show that Magic Kernel Sharp is more computationally efficient than the comparable Lanczos-3 kernel.

Finally, I show that it is not too challenging to make use of the work herein deriving Magic Kernel Sharp to construct any arbitrary member of this sequence of resizing kernels. I explicitly construct the fourth, fifth, and sixth of these, the last of which has essentially the same computational efficiency as the Lanczos-3 kernel. I show that their properties are not only far superior to Lanczos-3, but indeed so sufficiently close to ideal that continuing on to even higher members of the sequence is not likely to be needed for any regular imaging purpose, although they may be of use in other high-precision fields.

2. Definitions

I use the standard Engineering definition of the composition of a position-space kernel $g(x)$ in terms of its frequency-space components $G(f)$:

$$g(x) \equiv \int_{-\infty}^{\infty} df\, G(f)\, e^{2\pi i f x}, \tag{1}$$

so that, making the usual Fourier theory assumptions about convergence that allow integrals to always be interchanged,

$$G(f) = \int_{-\infty}^{\infty} df\, g(x)\, e^{-2\pi i f x}, \tag{2}$$

where Eq. (2) can be taken to be the definition of the Fourier transform of $g(x)$,

$$G(f) \equiv \mathcal{F}\{g(x)\}, \tag{3}$$

and Eq. (1) the definition of the inverse Fourier transform of $G(f)$ back to $g(x)$,

$$g(x) \equiv \mathcal{F}^{-1}\{G(f)\}. \tag{4}$$

With these definitions,

$$\mathcal{F}\{x^n g(x)\} \equiv \left(\frac{-1}{2\pi i}\right)^n \frac{d^n}{df^n} G(f). \tag{5}$$

The transform of a product is the convolution of the transforms, and vice versa,

$$\mathcal{F}\{g(x)h(x)\} \equiv G(f) * H(f) \equiv (G*H)(f), \tag{6}$$

where the convolution operator $*$ is

$$(G*H)(f) \equiv \int_{-\infty}^{\infty} df'\, G(f')H(f-f'). \tag{7}$$

Finally, the transform of a translation brings in a phase factor:

$$\mathcal{F}\{g(x-a)\} \equiv e^{-2\pi i a f} G(f). \tag{8}$$

All kernels $g(x)$ in this paper are normalized:

$$\int_{-\infty}^{\infty} dx\, g(x) \equiv 1, \tag{9}$$

so that

$$G(0) \equiv 1. \tag{10}$$

The Fourier transform of a *discrete* kernel,

$$g_{\text{discrete}}(x) \equiv \sum_{n=-\infty}^{\infty} g_n\, \delta(x-n), \tag{11}$$

is periodic:

$$G_{\text{discrete}}(f-n) \equiv G_{\text{discrete}}(f) \quad \text{for all integers } n, \tag{12}$$

and is therefore uniquely defined by its values within the *first Nyquist zone*:

$$|f| \le \frac{1}{2}. \tag{13}$$

Convolutions for discrete kernels are sums, rather than integrals:

$$g_n * h_n \equiv (g*h)_n \equiv \sum_{n'=-\infty}^{\infty} g_{n'} h_{n-n'}. \tag{14}$$

I use the standard Engineering definition of the sinc function,

$$\operatorname{sinc} x \equiv \begin{cases} 1 & \text{if } x = 0, \\ \dfrac{\sin \pi x}{\pi x} & \text{otherwise,} \end{cases} \tag{15}$$

the rectangular window function,

$$\operatorname{rect} x \equiv \begin{cases} 1 & \text{if } |x| < \dfrac{1}{2}, \\ 0 & \text{otherwise,} \end{cases} \tag{16}$$

and the Dirac delta comb function,

$$\operatorname{comb} x \equiv \sum_{n=-\infty}^{\infty} \delta(x-n), \tag{17}$$

so that sinc and rect are a Fourier transform pair,

$$\mathcal{F}\{\operatorname{sinc} x\} = \operatorname{rect} f, \tag{18}$$

and comb is its own Fourier transform,

$$\mathcal{F}\{\operatorname{comb} x\} = \operatorname{comb} f. \tag{19}$$

Clearly, convolving with $\operatorname{rect} x$ is equivalent to integrating over a sliding window:

$$\operatorname{rect} x * g(x) \equiv \int_{x-1/2}^{x+1/2} dx' g(x'). \tag{20}$$

I will make use of the first two derivatives of the sinc function,

$$\operatorname{sinc}' x \equiv \frac{d}{dx} \operatorname{sinc} x = \begin{cases} 0 & \text{if } x = 0, \\ \dfrac{\cos \pi x - \operatorname{sinc} x}{x} & \text{otherwise} \end{cases} \tag{21}$$

and

$$\operatorname{sinc}'' x \equiv \frac{d^2}{dx^2} \operatorname{sinc} x = \begin{cases} -\dfrac{\pi^2}{3} & \text{if } x = 0, \\ -\pi^2 \operatorname{sinc} x - 2\dfrac{\operatorname{sinc}' x}{x} & \text{otherwise,} \end{cases} \tag{22}$$

and likewise its first two indefinite integrals

$$\operatorname{sinc}^{(-1)} x \equiv \int dx \operatorname{sinc} x = \frac{1}{\pi} \operatorname{Si} \pi x + c, \tag{23}$$

and

$$\operatorname{sinc}^{(-2)} x \equiv \iint d^2x \operatorname{sinc} x = \frac{1}{\pi^2}\Big(\pi x \operatorname{Si} \pi x + \cos \pi x\Big) + c, \tag{24}$$

where $\operatorname{Si} x$ is the sine integral,

$$\operatorname{Si} x \equiv \int_0^x dt \, \frac{\sin t}{t}. \tag{25}$$

3. The Magic Kernel Sharp kernel

The *Magic Kernel Sharp* kernel, which I shall henceforth denote $k_3(x)$, is defined as the convolution of the *Magic Kernel*, $m(x)$, and the *Sharp* kernel, $s(x)$:

$$k_3(x) \equiv m(x) * s(x), \tag{26}$$

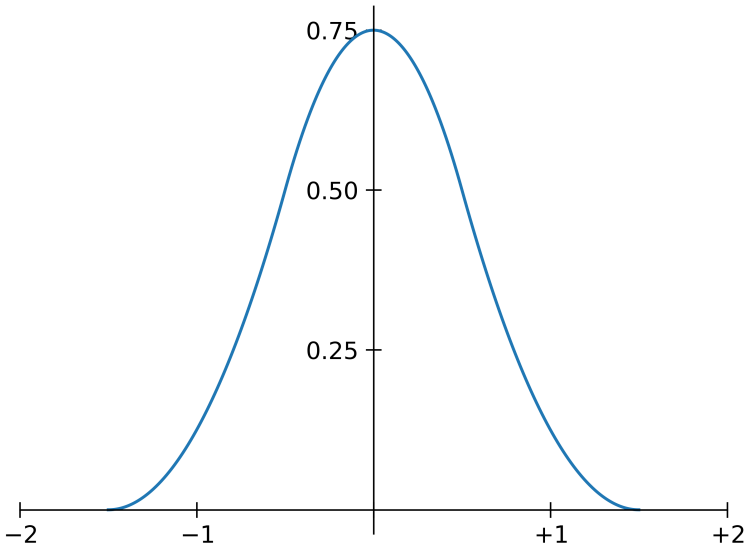


Figure 1: The Magic Kernel, $m(x)$.

where the Magic Kernel, $m(x)$, is a continuous kernel with a support of three units, which I originally deduced [3] to have the tri-parabolic functional form

$$m(x) \equiv \frac{1}{2}\Big(x+\frac{3}{2}\Big)^2 \text{rect}(x+1) + \Big(\frac{3}{4}-x^2\Big)\text{rect}\, x + \frac{1}{2}\Big(x-\frac{3}{2}\Big)^2 \text{rect}(x-1), \tag{27}$$

as shown in Fig. 1; and where the Sharp kernel, $s(x)$, is, in contrast, a *discrete* kernel,

$$s(x) \equiv \sum_{n=-\infty}^{\infty} s_n\, \delta(x-n), \tag{28}$$

where the coefficients s_n are defined so that the overall Magic Kernel Sharp kernel satisfies the *projection identity*,

$$k_3(n) \equiv \delta_{n0} \;\text{ for all integral } n, \tag{29}$$

an identity that is also satisfied by the sinc function.

Now, as I will show in Sec. 9, the solution to Eqs. (26), (27), (28), and (29) is a set of s_n that are nonzero for all n; *i.e.*, $s(x)$ has infinite support, and hence is not suitable for most practical purposes. However, there are two useful *approximations* to $s(x)$ that are manifestly practical. The first, *Sharp 2013* [4], denoted $s^{2013}(x)$, is defined to have only three nonzero coefficients s_n^{2013}:

$$s_n^{2013} \equiv \begin{cases} +\dfrac{3}{2} & \text{if } n = 0, \\ -\dfrac{1}{4} & \text{if } n = \pm 1, \\ 0 & \text{otherwise,} \end{cases} \tag{30}$$

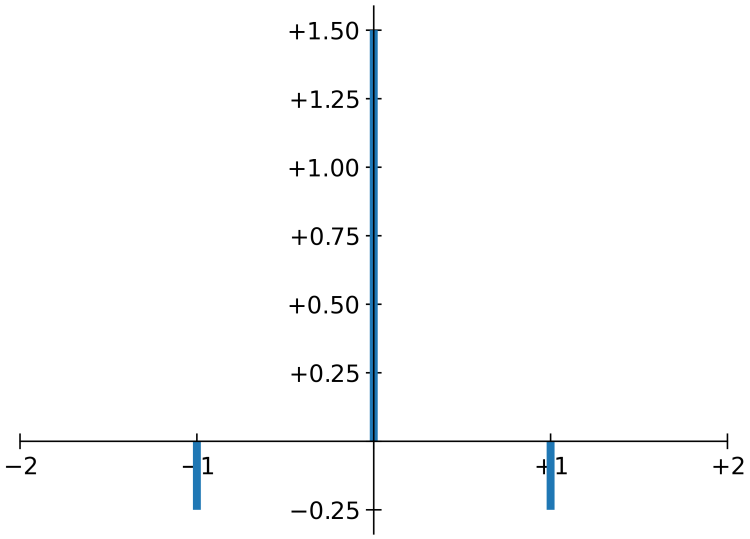


Figure 2: The Sharp 2013 kernel, $s^{2013}(x)$.

as shown in Fig. 2, and defines *Magic Kernel Sharp 2013*,

$$k_3^{2013}(x) \equiv m(x) * s^{2013}(x), \tag{31}$$

as shown in Fig. 3, which approximately satisfies the projection identity (29) to four bits of precision:

$$k_3^{2013}(n) = \begin{cases} +\dfrac{17}{16} & \text{if } n = 0, \\ -\dfrac{1}{32} & \text{if } n = \pm 2, \\ 0 & \text{otherwise.} \end{cases} \tag{32}$$

Magic Kernel Sharp 2013 is the kernel used for server-side resizing of images by Facebook and Instagram. The practical effect of only having the approximate satisfaction (32) of the projection identity (29) is a very slight sharpening of the output image, which is of little practical consequence—indeed, Instagram actually requires the addition of *more* sharpening [1].

The second approximation to $s(x)$, *Sharp 2021*, denoted $s^{2021}(x)$, is defined as the convolution of Sharp 2013, $s^{2013}(x)$, with an extra second-order *Correction 2021* kernel, $c^{2021}(x)$:

$$s^{2021}(x) \equiv s^{2013}(x) * c^{2021}(x), \tag{33}$$

where

$$c_n^{2021} \equiv \begin{cases} +\dfrac{17}{18} & \text{if } n = 0, \\ +\dfrac{1}{36} & \text{if } n = \pm 2, \\ 0 & \text{otherwise,} \end{cases} \tag{34}$$

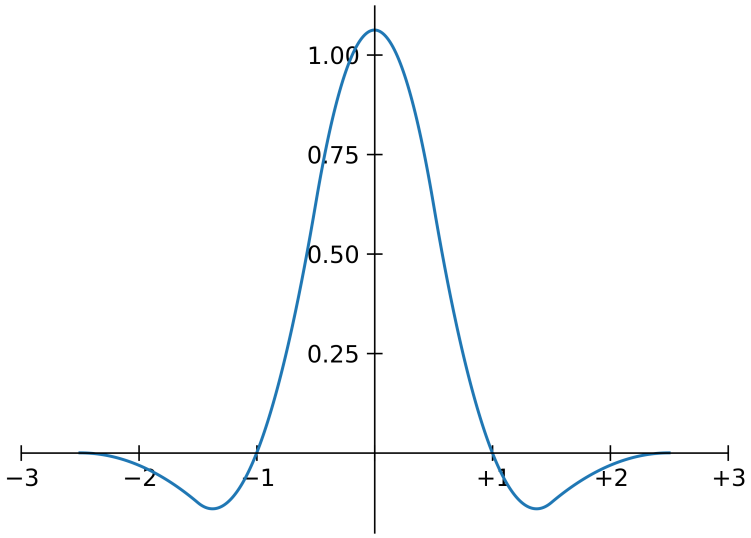


Figure 3: The Magic Kernel Sharp 2013 kernel, $k_3^{2013}(x)$.

as shown in Fig. 4, so that, performing the convolution, the coefficients of Sharp 2021 are

$$s_n^{2021} \equiv \begin{cases} +\dfrac{17}{12} & \text{if } n = 0, \\ -\dfrac{35}{144} & \text{if } n = \pm 1, \\ +\dfrac{1}{24} & \text{if } n = \pm 2, \\ -\dfrac{1}{144} & \text{if } n = \pm 3, \\ 0 & \text{otherwise,} \end{cases} \tag{35}$$

as shown in Fig. 5. This defines *Magic Kernel Sharp 2021*,

$$k_3^{2021}(x) \equiv m(x) * s^{2021}(x), \tag{36}$$

as shown in Fig. 6, which now approximately satisfies the identity (29) to nine bits of precision,

$$k_3^{2021}(n) = \begin{cases} +\dfrac{577}{576} & \text{if } n = 0, \\ -\dfrac{1}{1152} & \text{if } n = \pm 4, \\ 0 & \text{otherwise,} \end{cases} \tag{37}$$

which is more than sufficient for any imaging application that has a dynamic range of eight bits per channel. (Indeed, it is necessary to greatly zoom the vertical axis of Fig. 6, as shown in Fig. 7, to even barely discern this violation visually.)

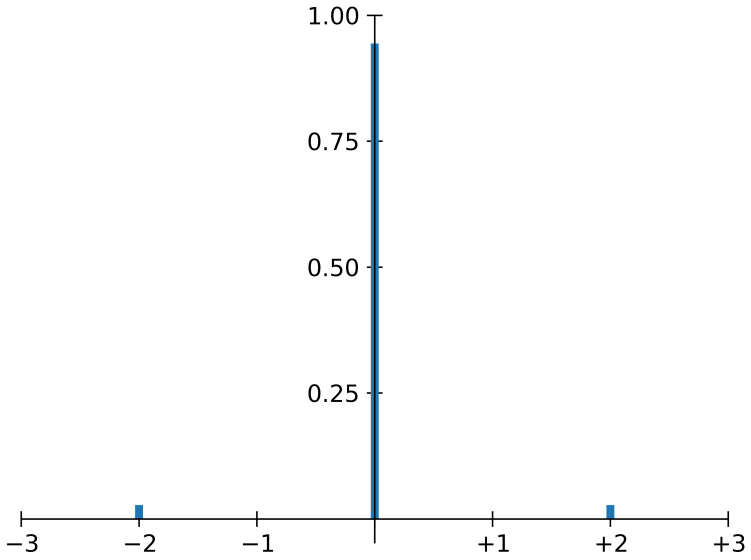


Figure 4: The extra Correction 2021 kernel, $c^{2021}(x)$, which is applied to the Sharp 2013 kernel, $s^{2021}(x)$, to create the Sharp 2021 kernel, $s^{2021}(x)$.

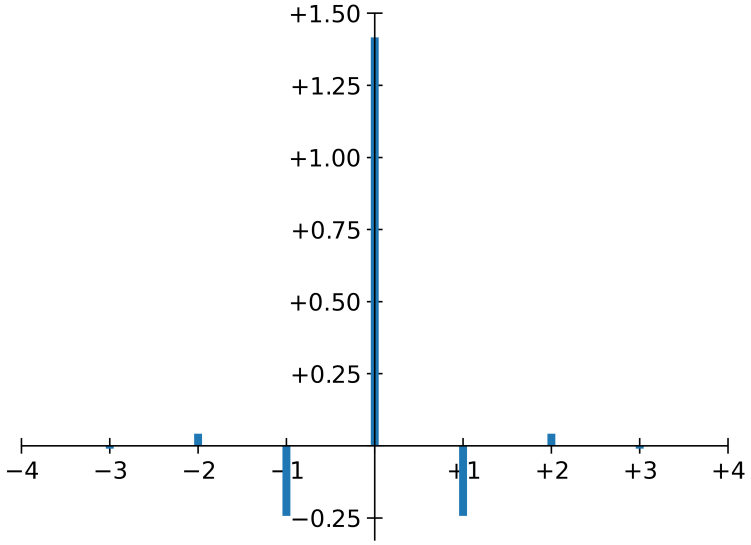


Figure 5: The Sharp 2021 kernel, $s^{2021}(x)$.

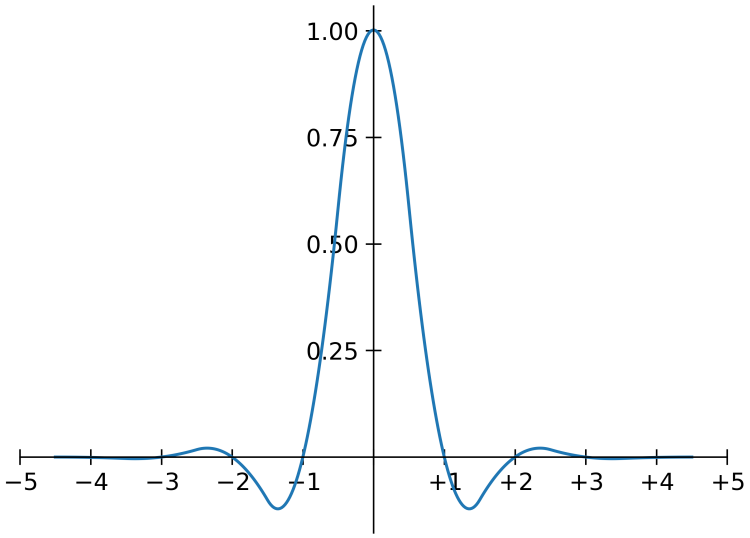


Figure 6: The Magic Kernel Sharp 2021 kernel, $k_3^{2021}(x)$.

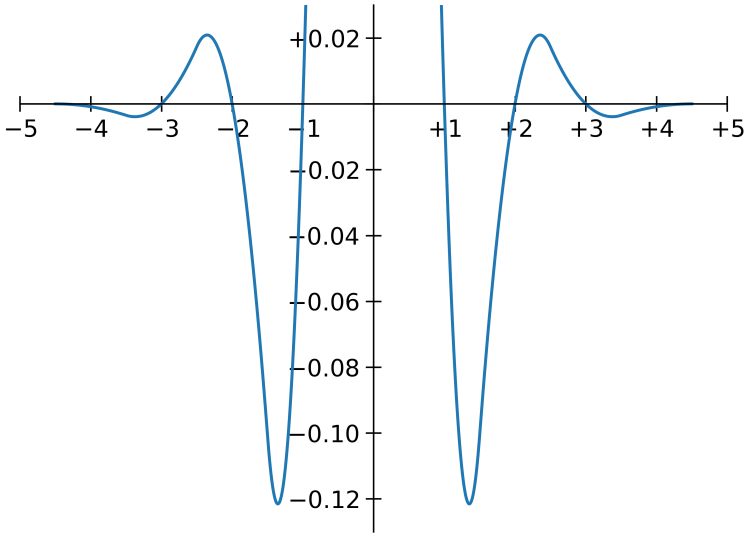


Figure 7: Vertical magnification of the graph of $k_3^{2021}(x)$ shown in Fig. 6, to show the $\sim$0.087% violation of the projection identity (29) at $x = \pm 4$, which is still almost too small to discern.

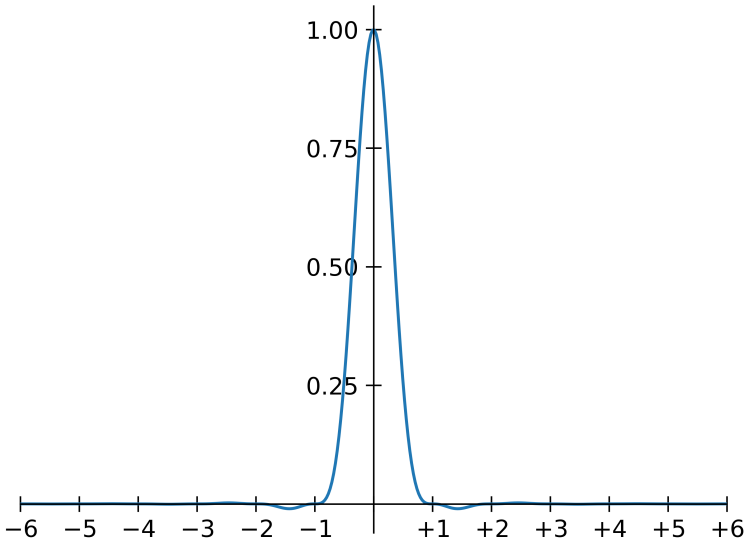


Figure 8: The Magic Kernel in frequency space, $M(f)$.

Magic Kernel Sharp 2021 is implemented in my ANSI C reference implementation [5], and should be used either when its full nine-bit precision is required, or when the slightly faster computational performance of the 2013 approximation is not critical.

4. The Fourier transform of the Magic Kernel

As noted in Sec. 1, it is easy to make use of the results of this section to simplify its derivation down to a simple two-line proof. However, it is worth noting that it is indeed possible (and I have the many pages of algebraic calculations on hand, if you feel the need to validate my pain) to explicitly make use of Eqs. (27), (5), (8), (18), (21), and (22), to ultimately obtain

$$M(f) = \operatorname{sinc}^3 f, \tag{38}$$

as shown in Fig. 8. In Fig. 9 I have zoomed the vertical axis so that we can see everything around its zeros. Note that the cube of the sinc function has not only given us our stationary points of inflection at all nonzero integral f, but it has also has reduced the magnitude of each first negative side-lobe to around 1% of the central lobe.

Similarly, in Fig. 10 I have zoomed the frequency axis. Clearly, $M(f)$ does not well approximate the square shape of the ideal rectangular window spectrum—indeed, like $m(x)$ itself, it actually resembles a Gaussian; this is why it slightly blurs the images that it is applied to.

Eqs. (18) and (6) now, of course, tell us that *the Magic Kernel $m(x)$ is itself just the convolution of* $\operatorname{rect} x$ *with itself, twice*:

$$m(x) = \operatorname{rect} x * \operatorname{rect} x * \operatorname{rect} x, \tag{39}$$

which, I am not ashamed to say, is the most surprising result that I have ever derived in my fifty-four years on this planet.

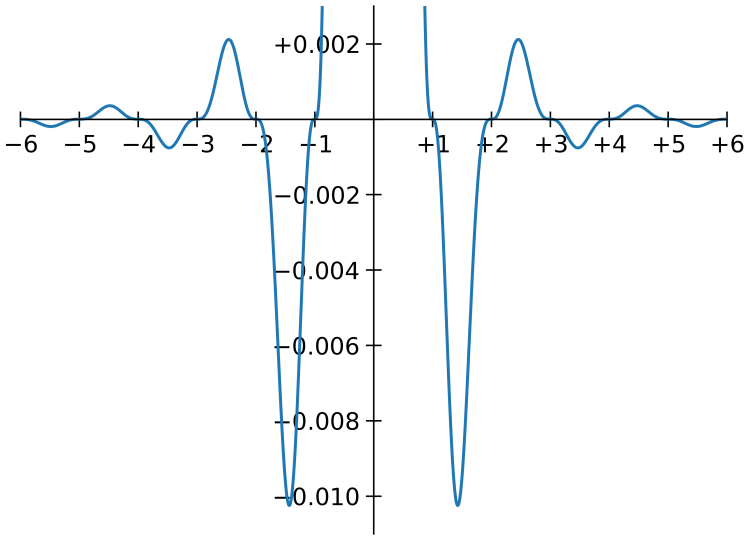


Figure 9: Vertical magnification of Fig. 8, showing the parts of $M(f)$ around its zeros.

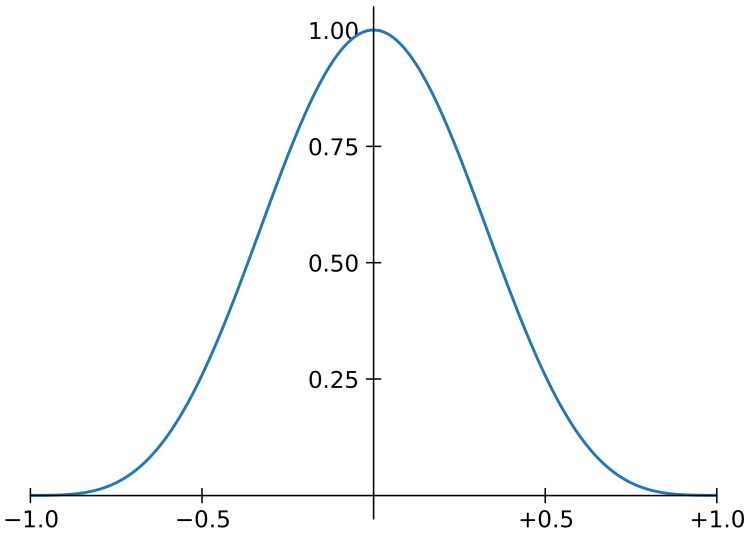


Figure 10: Horizontal magnification of Fig. 8, namely, the portion of the spectrum $M(f)$ of the Magic Kernel $m(x)$ that lies within the domain $-1 \leq f \leq +1$.

5. The Magic Kernel as the third in a fundamental sequence of kernels

As described in Sec. 1, the result (39) suggested to me that we could consider the Magic Kernel to be the third in a succession of kernels, each obtained by convolving rect x with itself the corresponding number of times. This sequence of kernels is shown graphically in Fig. 11.

Now, to best understand visually how each of the kernels performs in real life, it is useful to apply each of them as both an upsizing kernel and a downsizing kernel on suitably challenging test images.

For upsizing, aliasing artifacts are most clearly evident when a very small test image is upsized by a significant factor. I have therefore chosen the small test image shown in Fig. 12(b), originally derived from the image shown in Fig. 12(a) by downsizing by a factor of 16, and will use each of these kernels to upsize it back up by that factor of 16.

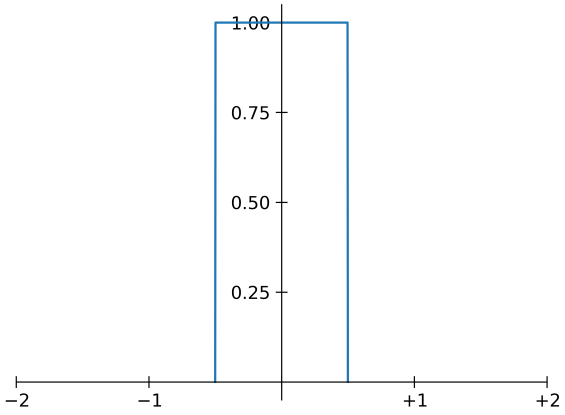
For downsizing, aliasing artifacts are most evident in the moiré patterns generated by bands in the original image whose frequencies are close to a multiple of the resampling frequency. I have therefore chosen the zone plate image of Fig. 13 for downsizing. As the bands near the circumference of the disk have a period of 16 pixels, and the inter-pixel spacing in the radial direction in the top-left (45°) corner is $\sqrt{2}$ times the inter-pixel spacing, I will downsize the image with each of the kernels by a factor of $16\sqrt{2}$.

The results of the upsizing tests are shown in Fig. 14, and the results of the downsizing tests are shown in Fig. 15.

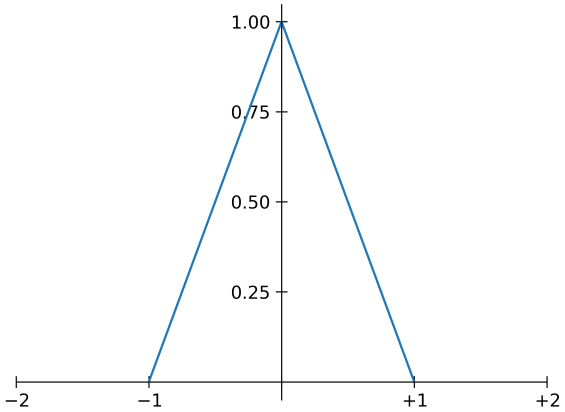
Now, the first in our sequence of kernels, $\text{rect}\, x \equiv k_1(x)$, shown in Fig. 11(a), can be interpreted as *the nearest neighbor interpolation kernel*: placing $k_1(x)$ at each pixel center simply allows that pixel value to "claim" its entire pixel "tile." The kernel $k_1(x)$ is discontinuous at its endpoints: we explicitly see the visual artifacts of that aliasing property when we look at the edges of the tiles that result when it is used as an image upsizing kernel, such as is shown in Fig. 14(a). Likewise, its poor aliasing performing for downsizing can be seen in the significant moiré patterns evident in Fig. 15(a).

The second kernel in this sequence, $\text{rect}\, x * \text{rect}\, x \equiv k_2(x)$, shown in Fig. 11(b), can be recognized as *the linear interpolation kernel* (leading to the *bilinear interpolation* kernel when it is applied separably for each dimension of a two-dimensional image): for upsizing, for example, placing $k_2(x)$ at each input pixel center means that that pixel's contribution to given any output pixel is unity if the center of the output pixel location maps exactly to that input pixel center—with no contributions from neighboring input pixels—and reduces linearly as this mapped output position moves across to either of its neighbors. This interpolation kernel is far superior to its predecessor $k_1(x)$, the nearest neighbor kernel, as *it is continuous at its endpoints*: there are none of the "sharp edges" that we see on the tiles created by $k_1(x)$. However, its *derivative* is still discontinuous at its endpoints and center point—namely, the positions of the original input pixels—and our visual systems *do* detect these discontinuities: they lead to the characteristic "star twinkle" artifacts of bilinearly upsampled images, such as can be seen in Fig. 14(b). For downsizing, Fig. 15(b) shows that the moiré patterns are much suppressed compared to the Fig. 15(a) of $k_1(x)$, but they are still present. (The artifacts at the right side of the top edge of Fig. 15(a) are due to the extension boundary conditions used for images, which are usually as good as it gets, visually, but which in this particular case are interacting with the diagonal bands of the test image.)

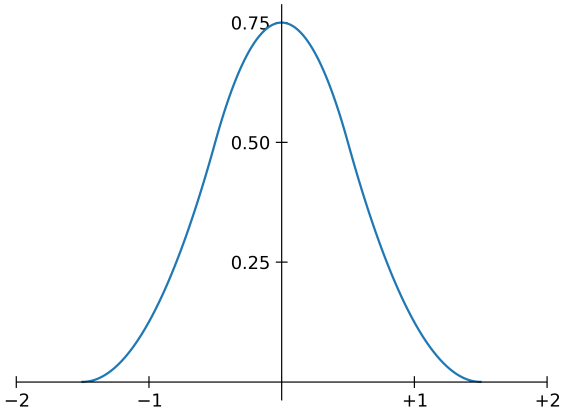
The third kernel in this sequence, $\text{rect}\, x * \text{rect}\, x * \text{rect}\, x \equiv m(x)$, shown in Fig. 11(c), is, herein



(a) $\operatorname{rect} x \equiv k_1(x)$



(b) $\operatorname{rect} x * \operatorname{rect} x \equiv k_2(x)$



(c) $\operatorname{rect} x * \operatorname{rect} x * \operatorname{rect} x \equiv m(x)$

Figure 11: The first three fundamental kernels obtained from $\operatorname{rect} x$, the rectangular window function: (a) the function itself; (b) convolved with itself; and (c) convolved with itself twice.

(a) Original 320×320 image

(b) Downsized to 20×20

Figure 12: Test image used for upsizing. (The image (b) shown here has been nearest-neighbor upsized so that its rendering is independent of your PDF viewer.)

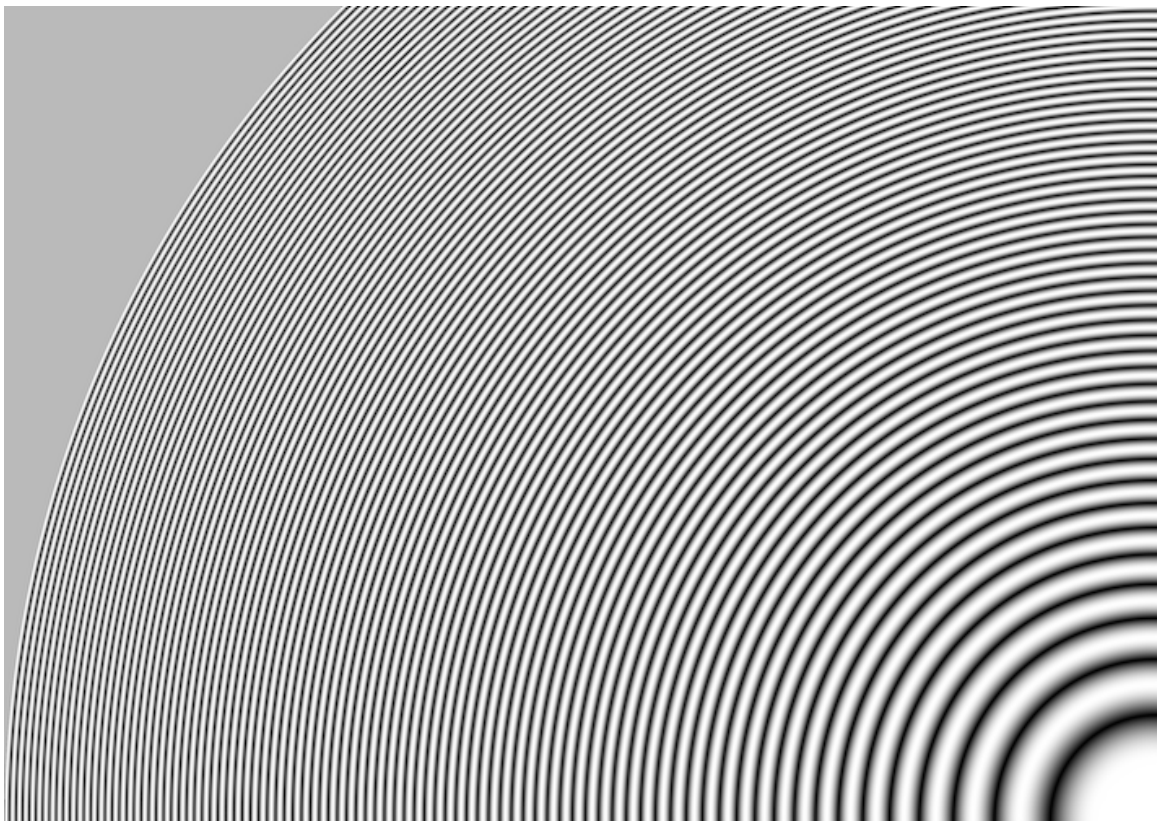
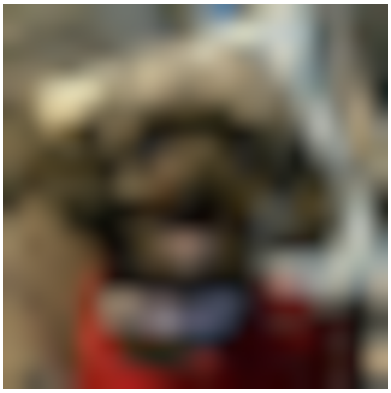


Figure 13: Test 2835×2004 image used for downsizing. (The image shown here has been downsized by a factor of 4, simply for file size efficiency for this PDF document.)

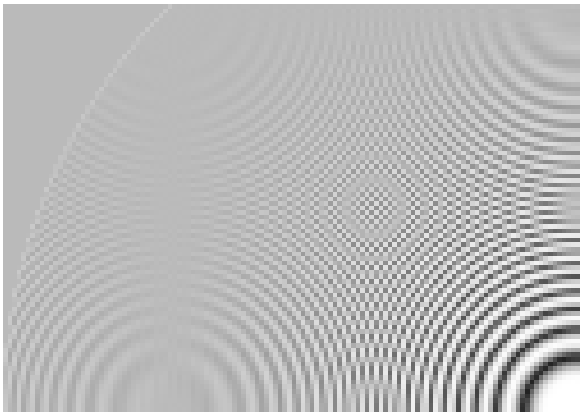
(a) Fig. 12(b) upsized with $k_1(x)$, the nearest neighbor kernel

(b) Fig. 12(b) upsized with $k_2(x)$, the (bi)linear interpolation kernel



(c) Fig. 12(b) upsized with $m(x)$, the Magic Kernel

Figure 14: Application of the first three fundamental kernels, as shown in Fig. 11, to the test image shown in Fig. 12(b), as upsizing kernels by a factor of 16.



(a) Fig. 13 downsized with $k_1(x)$, the nearest neighbor kernel

(b) Fig. 13 downsized with $k_2(x)$, the (bi)linear kernel

(c) Fig. 13 downsized with $m(x)$, the Magic Kernel

Figure 15: Application of the first three fundamental kernels, as shown in Fig. 11, to the test image shown in Fig. 13, as downsizing kernels by a factor of $16\sqrt{2}$.

and belatedly, recognized as the Magic Kernel. It is made up of three matching parabolas, so that it is both continuous and has continuous first derivative everywhere. This gives it essentially ideal visual smoothness properties, as is evident in Fig. 11(c): the "star twinkles" of $k_2(x)$ have been eliminated, and the image visually seems to have excellent rotational symmetry for pointlike features in the original image. (Discontinuities in the second derivative and higher generally do not seem to "stand out" to the human eye.) As a downsizing kernel, Fig. 15(c) shows that the Moiré aliasing artifacts have been suppressed even further, compared to those of $k_1(x)$ and $k_2(x)$.

An interesting question now arises: is Magic Kernel upsizing effectively "parabolic interpolation," in the same way that $k_2(x)$ upsizing is just linear interpolation? It is easy to see that it is not: interpolation involving $n \geq 3$ points is *not* performed by simply convolving with a kernel made up piecewise of polynomials of degree $n-1$; for example, *cubic interpolation* seeks to *fit a single cubic function* through four adjacent pixel values, rather than *convolve with a kernel that is piecewise cubic*. The former is what is used for *bicubic interpolation* for a two-dimensional image; whereas the latter is the self-evident generalization, by one more step, of the sequence that leads to the Magic Kernel—namely, $\operatorname{rect} x * \operatorname{rect} x * \operatorname{rect} x * \operatorname{rect} x$, which would be clearly made up of four matching cubics between $x = -2$ and $x = +2$ (and would have the Fourier transform $\operatorname{sinc}^4 f$).

Rather, Magic Kernel upsizing is effectively an interpolation *of* linear interpolation, which itself is in some sense an "interpolation" of nearest neighbor interpolation, so that Magic Kernel upsizing is in this sense "interpolation cubed," not parabolic interpolation.

6. The three fundamental kernels in frequency space

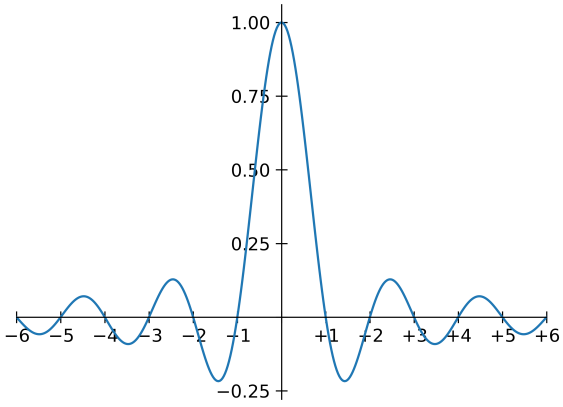
Now, the improved fidelity of the kernels as we progress through the sequence from $k_1(x)$ to $k_2(x)$ to $m(x)$ was described in the previous secion (and visualized in Fig. 14) in terms of the continuity of (or lack of) each kernel and its first derivative, in position space. But of course we can more clearly understand what is going on in terms of aliasing—and, in particular, its moiré pattern artifacts in Fig. 15—when we move back to frequency space.

By construction, the Fourier transforms of these three kernels are just the first three powers of $\operatorname{sinc} f$, as shown in Fig. 16.

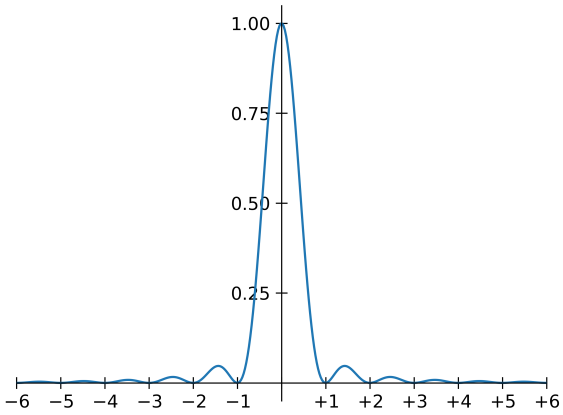
Fig. 16(a) shows that the Fourier spectrum of the nearest neighbor kernel $k_1(x)$ is a very poor approximation to the ideal rectangular window spectrum $\operatorname{rect} f$. Now, a useful measure of this fidelity is how much of the *spectral energy* (the squared magnitude in frequency space) lies outside the first Nyquist zone. A naive definition of this would measure all such energy outside the window $|f| \leq \frac{1}{2}$. But such a calculation would, for any practical kernel, be dominated by that part of the "wall" that lies just outside this zone, because any practical kernel will have a finite slope at these boundaries; we would basically be measuring just the "fall-off" of that slope. While this slope is an important metric, for many purposes we are really interested in the spectral energy that lies somewhere near the sampling frequency and beyond. So let us define the "leaked spectral energy," $E_{G(f)}$, of a kernel $G(f)$ as the total spectral energy outside, say, the band $|f| \leq 0.85$:

$$E_{G(f)} \equiv \int_{-\infty}^{-0.85} df\, |G(f)|^2 + \int_{0.85}^{\infty} df\, |G(f)|^2, \tag{40}$$

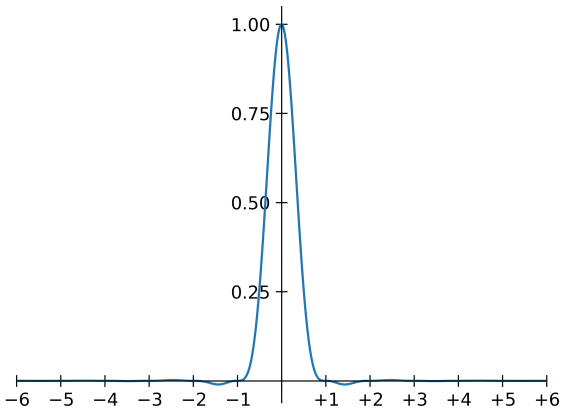
which I will, for convenience, refer to as the spectral energy "outside the first Nyquist zone."



(a) $K_1(f) = \mathrm{sinc} f$



(b) $K_2(f) = \mathrm{sinc}^2 f$



(c) $M(f) = \mathrm{sinc}^3 f$

Figure 16: The Fourier transforms of the first three fundamental kernels shown in Fig. 11.

With this definition, we find that $E_{K_1(f)}$ is about 10%, or about −10 dB. (I will henceforth use decibels for $E_{G(f)}$.)

On the plus side, $K_1(f)$ *does* possess simple zeros at all nonzero integers, which means that there is *some* suppression of aliasing artifacts folding over to low frequency: the moiré patterns in Fig. 15(a) are not, at least, at the *full* dynamic range of the input image bands—but they still show that $k_1(x)$ is a very poor resizing kernel.

Fig. 16(b) shows that the Fourier spectrum of the linear interpolation kernel $k_2(x)$ is much better: only −25.2 dB lies outside the first Nyquist zone (under the above definition), and it has *second*-order zeros at mutiples of the sampling frequency, which further improve its suppression of low-frequency aliasing artifacts: for upsizing, the "tile edges" have now been softened away; and for downsizing, the moiré patterns are now substantially suppressed.

Fig. 16(c) shows that the Magic Kernel $m(x)$ is yet another step up again in fidelity: just −39.0 dB of its spectral energy lies outside the first Nyquist zone, and its third-order zeros at all multiples of the sampling frequency provide an even better suppression of low-frequency foldover.

7. The exact Sharp kernel in frequency space

As noted in Sec. 3, the Magic Kernel $m(x)$ differs fundamentally from its two predecessors $k_1(x)$ and $k_2(x)$ in that it does not satisfy the *projection identity* (29): the second convolution has widened the support of $m(x)$ from the one unit possessed by $k_1(x)$, and the two units possessed by $k_2(x)$, to *three* units—which means that it encompasses the neighboring pixel locations at $x = \pm 1$, so that a same-size sampled application of it results not in the delta function (29), but rather the discrete *blur kernel*, $b(x)$, defined by its three nonzero coefficients

$$b_n = \begin{cases} \frac{3}{4} & \text{if } n = 0, \\ \frac{1}{8} & \text{if } n = \pm 1, \\ 0 & \text{otherwise.} \end{cases} \tag{41}$$

The ansatz that Magic Kernel Sharp (26) must satisfy (29) simply means that the discrete Sharp kernel s_n must "undo" the discrete blur kernel b_n:

$$s_n * b_n \equiv \delta_{n0}. \tag{42}$$

Now, the Fourier transform of b_n is trivial to compute:

$$B(f) = 1 - \tfrac{1}{2}\sin^2 \pi f, \tag{43}$$

as shown in Fig. 17. To "undo" this with $S(f)$, we simply require

$$S(f) \equiv \frac{1}{B(f)} = \frac{1}{1 - \frac{1}{2}\sin^2 \pi f}, \tag{44}$$

as shown in Fig. 18.

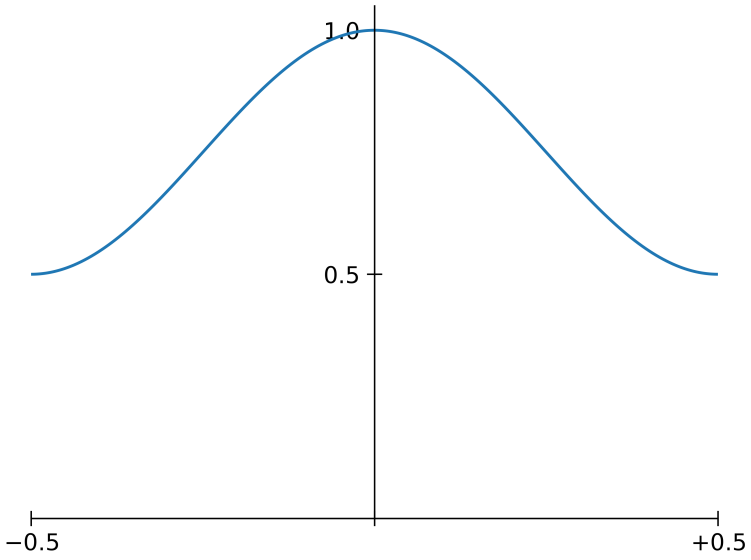


Figure 17: The Fourier transform $B(f)$ of the blur kernel b_n of the sampled Magic Kernel.

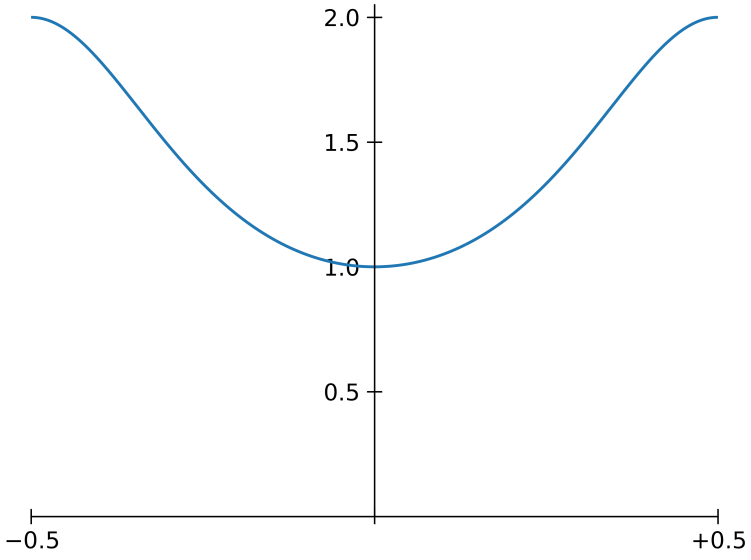


Figure 18: The Fourier transform $S(f) \equiv 1/B(f)$ of the Sharp kernel $s(x)$.

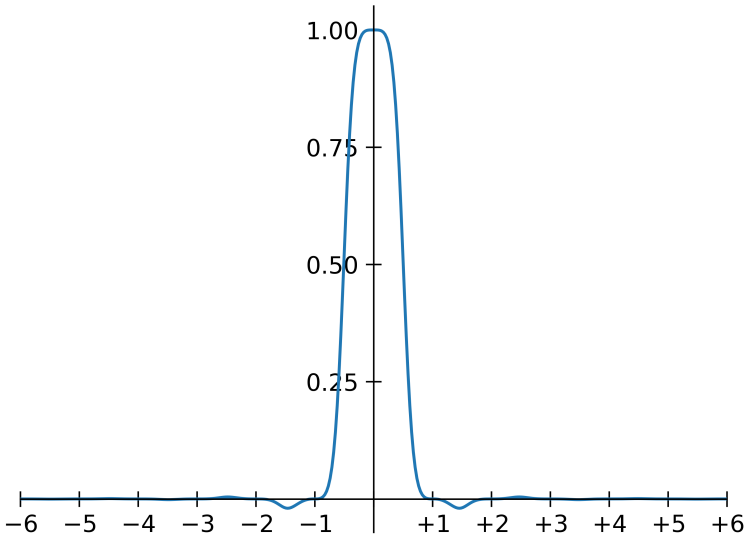


Figure 19: The Fourier transform $K_3(f)$ of the Magic Kernel Sharp kernel $k_3(x)$.

8. The Fourier transform of Magic Kernel Sharp

From Eqs. (26), (38), and (44), we find that the Fourier transform $K_3(f)$ of the Magic Kernel Sharp kernel $k_3(x)$ is

$$K_3(f) = \frac{\operatorname{sinc}^3 f}{1 - \frac{1}{2}\sin^2 \pi f}, \tag{45}$$

as shown in Fig. 19. Magnifying again the frequency axis, in Fig. 20, we can see that its spectrum is much "squarer" than that of the Magic Kernel, Fig. 10, *i.e.*, it is much more faithful to the shape of the ideal rectangular window spectrum. Its "flat top" might be surprising to see, but it is actually just a consequence of the projection identity ansatz (29), which can be rewritten

$$k_3(x) \operatorname{comb} x \equiv \delta(x), \tag{46}$$

which in frequency space is just

$$K_3(f) * \operatorname{comb} f \equiv 1, \tag{47}$$

or in other words *the aliased version of* $K_3(f)$ *must be unity for all* f. Now, since $M(f)$ falls off rapidly with $|f|$ because of the cube of the sinc function, and assuming *a priori* that the periodic $S(f)$ will be relatively well-behaved (as we know it ultimately is), then $K_3(f) \equiv M(f)\, S(f)$ will likewise fall off rapidly with $|f|$, and so there is little of it that will alias to low frequencies, and hence (47) stipulates that $K_3(f)$ must be close to unity for low frequencies, which is just the "flat top" that we see in Fig. 20.

Of course, this improved fidelity within the first Nyquist zone does bring with it a slight compensating penalty elsewhere: we now find that $E_{K_3(f)}$ is −35.9 dB, compared to the −39.0 dB of $M(f)$, due to the amplification of the small side lobes of $M(f)$ of Fig. 8 by the Sharp kernel $S(f)$ of Fig. 18 (as shown in Fig. 21, which is a vertical magnification of Fig. 19).

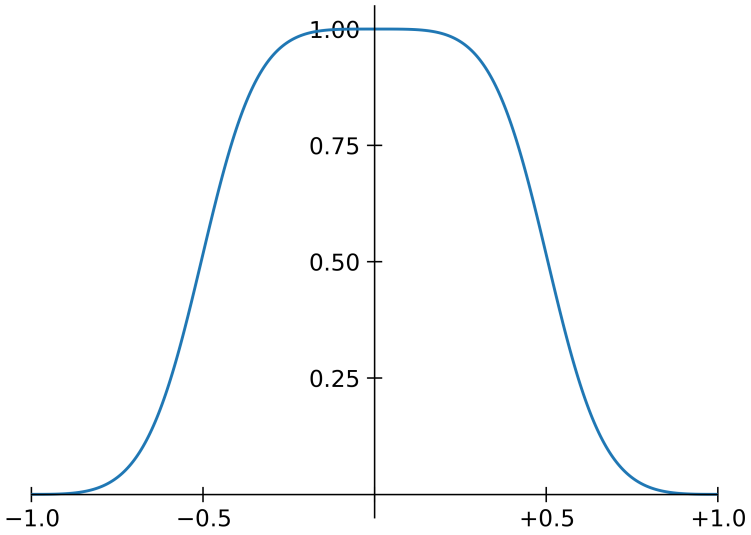


Figure 20: The Magic Kernel Sharp spectrum $K_3(f)$ for $-1 \leq f \leq +1$, namely, a horizontal magnification of the graph shown in Fig. 19. Note that this spectrum is much "squarer" than that of the Magic Kernel, $M(f)$, shown in Fig. 10.

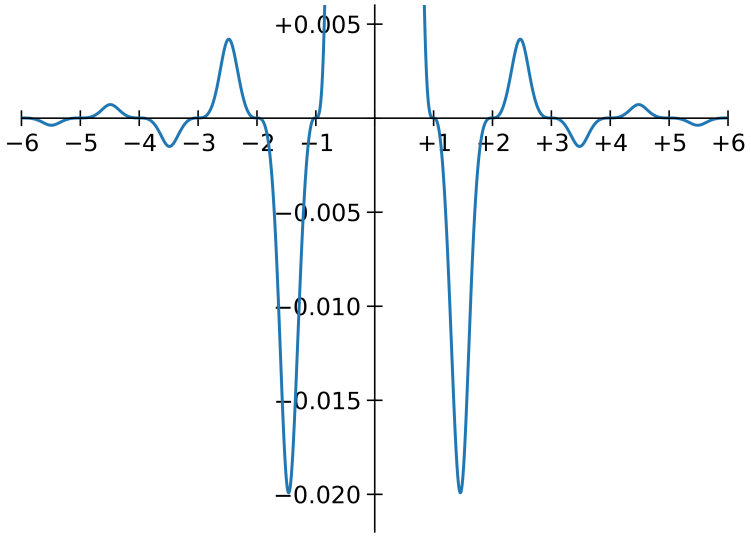


Figure 21: Vertical magnification of Fig. 19, more clearly showing the regions around the zeros of $K_3(f)$. Note the increased scale compared to Fig. 9.

n	s_n	s_n^{2013}	s_n^{2021}
0	+1.414 214	+1.500 000	+1.416 667
±1	−0.242 641	−0.250 000	−0.243 056
±2	+0.041 631	–	+0.041 667
±3	−0.007 143	–	−0.006 944
±4	+0.001 225	–	–
±5	−0.000 210	–	–

Table 1: The first six coefficients s_n of the exact Sharp kernel $s(x)$, together with those of its 2013 and 2021 approximations, $s^{2013}(x)$ and $s^{2021}(x)$.

Overall, however, this 3.1 dB penalty is definitely a price we want to pay: the squareness of the spectrum—and the consequent prevention of it blurring the input image unnecessarily—is of paramount importance for visual fidelity.

9. The exact Sharp kernel in position space

We are now in a position to derive the exact Sharp kernel in position space, by computing the inverse Fourier transform of the $S(f)$ of Eq. (44).

Firstly, let us rewrite (44) by using the Maclaurin series expansion of $1/(1-x)$ around $x = 0$:

$$S(f) = \sum_{n=0}^{\infty} \frac{1}{2^n} \sin^{2n}\pi f. \tag{48}$$

We can now make use of the trigonometric power formula [6]

$$\sin^{2n} f \equiv \frac{1}{2^{2n}}\binom{2n}{n} + \frac{(-1)^n}{2^{2n-1}} \sum_{k=0}^{n-1} (-1)^k \binom{2n}{k} \cos[2(n-k)f], \tag{49}$$

yielding

$$S(f) = \sum_{n=0}^{\infty} \left\{ \frac{1}{2^{3n}}\binom{2n}{n} + \frac{(-1)^n}{2^{3n-1}} \sum_{k=0}^{n-1} (-1)^k \binom{2n}{k} \cos[2\pi(n-k)f] \right\}. \tag{50}$$

This may look rather unedifying, but the only part that depends on f is the $\cos[2\pi(n-k)f]$, which is trivial to inverse Fourier transform:

$$\mathcal{F}^{-1}\Big\{\cos[2\pi(n-k)f]\Big\} = \frac{1}{2}\Big\{\delta(x-(n-k) + \delta(x+(n-k)\Big\}. \tag{51}$$

We therefore find that the position-space coefficients s_n of the exact Sharp kernel are given by

$$s_n = (-1)^n \sum_{m=|n|}^{\infty} \frac{1}{2^{3m}}\binom{2m}{m+|n|}. \tag{52}$$

The first six coefficients s_n are shown in Table 1, together with those of its 2013 and 2021 approximations. It can be seen that the 2021 approximations to the first three coefficients are

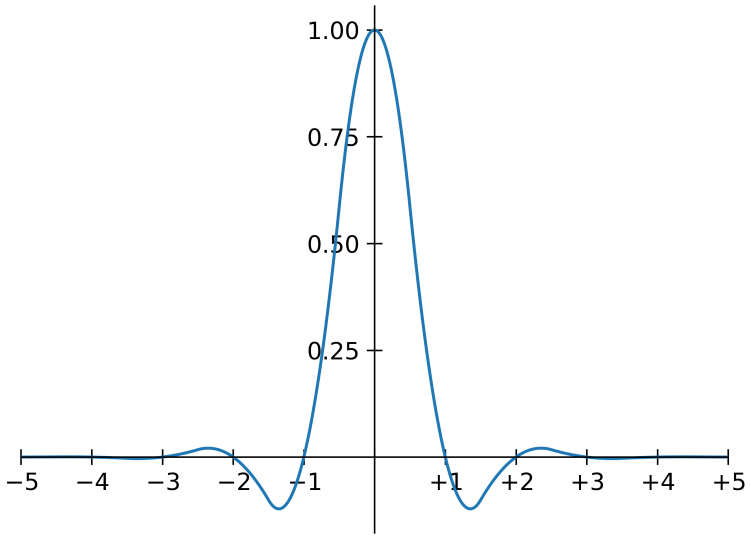


Figure 22: The exact Magic Kernel Sharp kernel, $k_3(x)$.

accurate to over nine bits of precision, which explains why (37) itself satisfies the projection identity (29) to nine bits of precision.

10. The exact Magic Kernel Sharp kernel in position space

Even though it is not in closed form, the expression (52) for the exact Sharp kernel in position space, $s(x)$, at least provides a *formal* specification of the exact Magic Kernel Sharp kernel in position space, $k_3(x)$, namely, Eq. (26).

We can, of course, also use these expressions to obtain a *numerical approximation* to the exact $k_3(x)$ to any desired level of accuracy. For example, the exact Magic Kernel Sharp kernel, $k_3(x)$, is shown in Fig. 22, which we can use to upsize the test image shown in Fig. 12(b), as shown in Fig. 23(c) of Fig. 23, and downsize the test image shown in Fig. 15, as shown in Fig. 24(c) of Fig. 24. (Note that the amplification of the side lobes of $K_3(f)$ compared to $M(f)$ does not seem to have caused too much regression in the suppression of the moiré patterns of $m(x)$, whereas it *has* significantly removed the blurring of the true image.)

11. The partition of unity property

When I originally deduced the form (27) of the Magic Kernel $m(x)$ in 2011 [3], what immediately amazed me was the fact that it "fits into itself," namely, if a copy of $m(x)$ is placed at every integral value of x, and then all of those copies are summed, the result is unity for all values of x:

$$\sum_{n=-\infty}^{\infty} m(x-n) = 1 \text{ for all } x. \tag{53}$$

In other words, $m(x)$ *tesselates the infinite horizontal strip between* $y = 0$ *and* $y = 1$, as represented graphically in Fig. 25. This is an extremely important property: it means that if $m(x)$

(a) Fig. 12(b) upsized with $k_2(x)$, the (bi)linear interpolation kernel

(b) Fig. 12(b) upsized with $m(x)$, the Magic Kernel

(c) Fig. 12(b) upsized with $k_3(x)$, Magic Kernel Sharp

Figure 23: Continuation of the sequence of example image upsizings shown in Fig. 14, where the final image (c) has been obtained using Magic Kernel Sharp.

(a) Fig. 15 downsized with $k_2(x)$, the (bi)linear kernel

(b) Fig. 15 downsized with $m(x)$, the Magic Kernel

(c) Fig. 15 downsized with $k_3(x)$, Magic Kernel Sharp

Figure 24: Continuation of the sequence of example image downsizings shown in Fig. 15, where the final image (c) has been obtained using Magic Kernel Sharp.

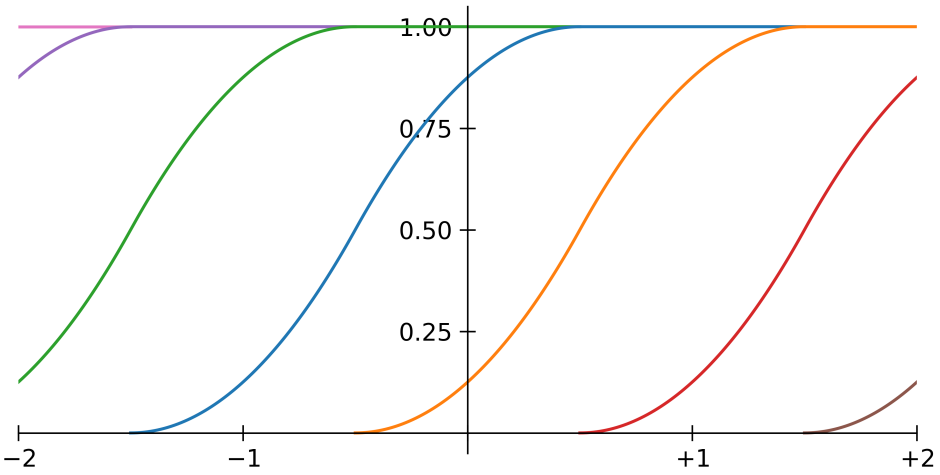


Figure 25: Graphical representation of the tesselation of the infinite horizontal strip between $y = 0$ and $y = 1$ represented by Eq. (53). The cumulative integral up to a given value of x up to each different copy of $m(x)$ is represented by a different color.

is applied as a kernel to a *constant input signal*, then the output will likewise be a constant for all x, and hence will generate a constant output signal for any resizing factor. In mathematical terminology, Eq. (53) says that $m(x)$ is *a partition of unity*:

$$\sum_{n=-\infty}^{\infty} g(x-n) = 1 \text{ for all } x. \tag{54}$$

Of course, the kernels $k_1(x)$ and $k_2(x)$ are *also* partitions of unity, and it is worth determining exactly which shared feature of each of these three kernels is responsible for this extremely desirable property. Now, Eq. (54) can be rewritten in the form

$$g(x) * \operatorname{comb} x = 1, \tag{55}$$

which, taking the Fourier transform of both sides, yields

$$G(f) \operatorname{comb} f = \delta(f). \tag{56}$$

The comb and delta functions here simply convert this into a *discrete* equation,

$$G(n) = \delta_{n0} \text{ for all integral } n. \tag{57}$$

This can be recognized as simply the *projection identity* (29), but now applied in *frequency* space, rather than position space. It is obviously satisfied by $K_1(f)$, $K_2(f)$, and $M(f)$—and $K_3(f)$ as well, for that matter—because the powers of $\operatorname{sinc} f$ in each of these spectrums provide the necessary zeros, and the consequence (10) of the normalization (9) of all kernels provides the required unity at $f = 0$.

We therefore see that all four of these kernels are partitions of unity, as they satisfy the projection identity in frequency space, whereas all but $m(x)$ satisfy the projection identity in

position space, which ensures no distortion of a sampled signal when the kernel is applied as an interpolation kernel.

12. The Fourier transforms of the Lanczos kernels

Because of their popularity, it is worth deriving the Fourier transforms of the *Lanczos kernels*,

$$\ell_a(x) \equiv \operatorname{sinc} x \ \operatorname{rect}\frac{x}{2a} \operatorname{sinc}\frac{x}{a}, \tag{58}$$

where a is a positive integer, typically 2 or 3. Clearly,

$$L_a(f) = 2a^2 \operatorname{rect} f * \operatorname{sinc} 2af * \operatorname{rect} af. \tag{59}$$

Consider the first convolution here, $\operatorname{rect} f * \operatorname{sinc} 2af$. From Eq. (20), this is just

$$\operatorname{rect} f * \operatorname{sinc} 2af = \int_{f-\frac{1}{2}}^{f+\frac{1}{2}} df' \operatorname{sinc} 2af', \tag{60}$$

in other words, the antiderivative of $\operatorname{sinc} 2af$, evaluated at $f-\frac{1}{2}$ and $f+\frac{1}{2}$:

$$2a \operatorname{rect} f * \operatorname{sinc} 2af = \operatorname{sinc}^{(-1)}2a(f+\tfrac{1}{2}) - \operatorname{sinc}^{(-1)}2a(f-\tfrac{1}{2}). \tag{61}$$

We can now similarly include the second convolution of (59)

$$\begin{aligned} 2a \operatorname{rect} f * \operatorname{sinc} 2af * \operatorname{rect} af &\equiv \int_{-\infty}^{\infty} df' \operatorname{rect} a(f-f')\left\{\operatorname{sinc}^{(-1)}2a(f'+\tfrac{1}{2}) - \operatorname{sinc}^{(-1)}2a(f'-\tfrac{1}{2})\right\}, \\ &= \int_{f-1/2a}^{f+1/2a} df' \left\{\operatorname{sinc}^{(-1)}2a(f'+\tfrac{1}{2}) - \operatorname{sinc}^{(-1)}2a(f'-\tfrac{1}{2})\right\}, \end{aligned} \tag{62}$$

to obtain the final expression for the Fourier transform $L_a(f)$ of the Lanczos-a kernel $\ell_a(x)$:

$$\begin{aligned} L_a(f) = \frac{1}{2}\Big\{&\operatorname{sinc}^{(-2)}(2af+a+1) - \operatorname{sinc}^{(-2)}(2af+a-1) \\ &- \operatorname{sinc}^{(-2)}(2af-a+1) + \operatorname{sinc}^{(-2)}(2af-a-1)\Big\}. \end{aligned} \tag{63}$$

Figs. 26 and 27 show graphically the Fourier transforms of the Lanczos-a kernels for $a = 2$ and $a = 3$. They look very good—similar to Fig. 19 for the spectrum $K_3(f)$ of Magic Kernel Sharp. Magnifying the frequency axis like we did for Figs. 10 and 20, in Fig. 28 and Fig. 29, we can see that $L_2(f)$ is almost as "square" as $K_3(f)$, and $L_3(f)$ is actually slightly "squarer" than $K_3(f)$.

These observations are backed up when we compute the amount of leaked spectral energy: it is −44.5 dB for Lanczos-2, and −53.3 dB for Lanczos-3. Recall that it was −35.9 dB for Magic Kernel Sharp, which is thus 8.6 dB worse than even Lanczos-2 on this measure, and 17.4 dB worse than Lanczos-3. These results are summarized in Table 2.

We can likewise use Lanczos-2 and Lanczos-3 as upsizing and downsizing kernels on our test images, and the results are shown in Figs. 30 and 31.

For upsizing, the quality of the results is now so good for all three kernels that it may be difficult to discern the differences visually from what is shown in Fig. 30. However, a careful

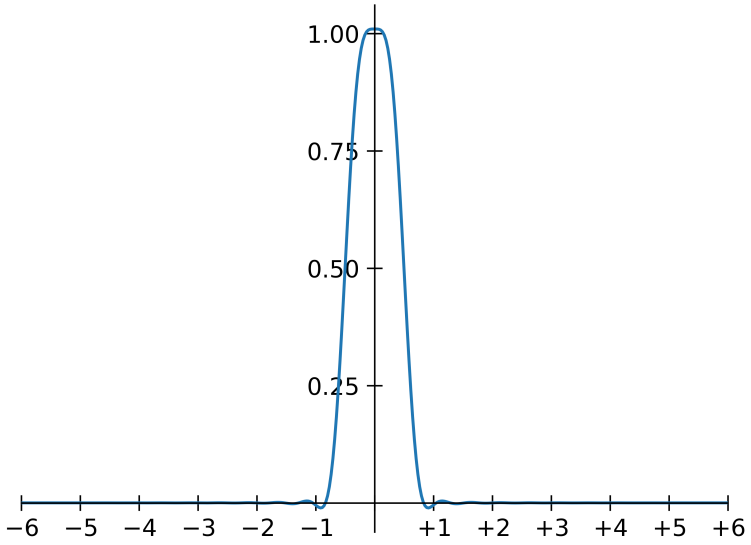


Figure 26: $L_2(f)$, the Fourier transform of the Lanczos-2 kernel $\ell_2(x)$.

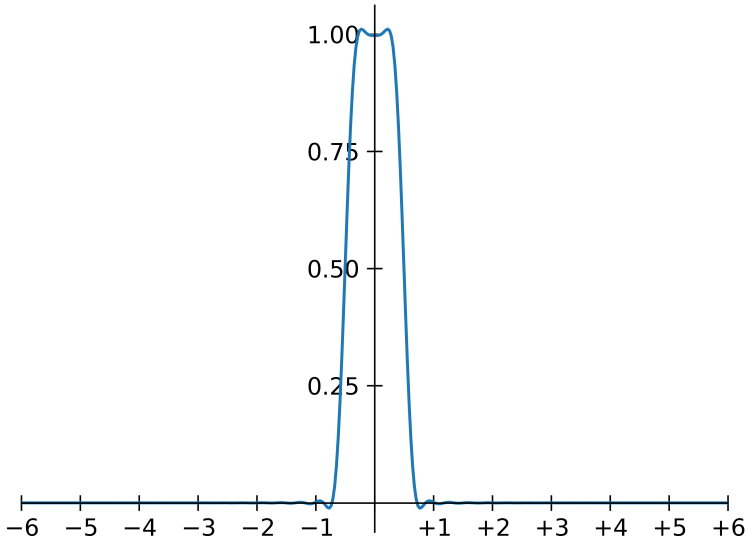


Figure 27: $L_3(f)$, the Fourier transform of the Lanczos-3 kernel $\ell_3(x)$.

Kernel	E
Nearest Neighbor	−10.0 dB
Linear Interpolation	−25.2 dB
Magic Kernel Sharp	−35.9 dB
Lanczos-2	−44.5 dB
Lanczos-3	−53.3 dB

Table 2: The leaked spectral energy for the five resizing kernels analyzed.

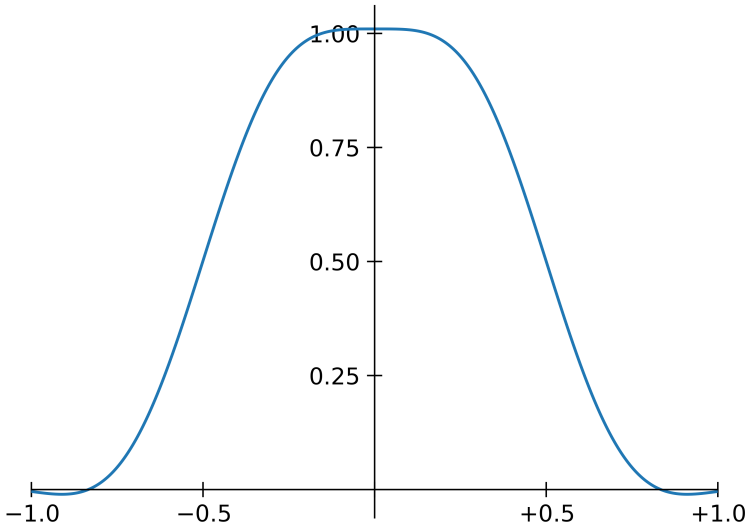


Figure 28: The portion of $L_2(f)$ within the domain $-1 \leq f \leq +1$.

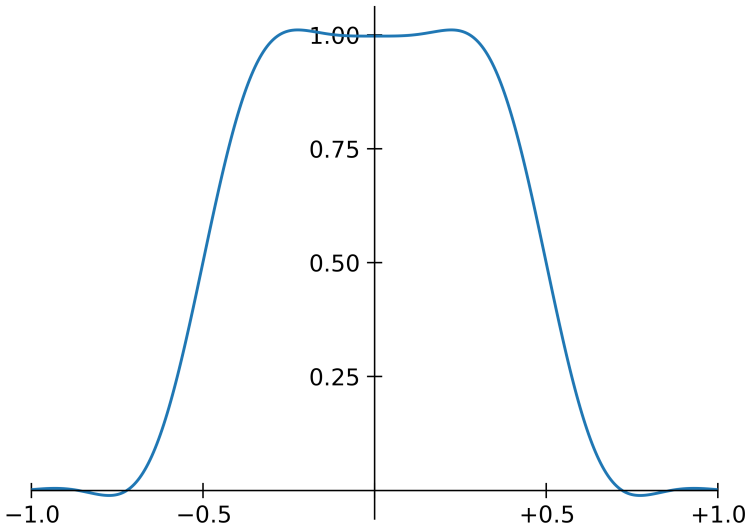


Figure 29: The portion of $L_3(f)$ within the domain $-1 \leq f \leq +1$.

(a) Fig. 12(b) upsized with $k_3(x)$, Magic Kernel Sharp

(b) Fig. 12(b) upsized with $\ell_2(x)$, the Lanczos-2 kernel

(c) Fig. 12(b) upsized with $\ell_3(x)$, the Lanczos-3 kernel

Figure 30: Comparison of using Magic Kernel Sharp, Lanczos-2, and Lanczos-3 as upsizing kernels on the test image of Fig. 12(b).

(a) Fig. 15 downsized with $k_3(x)$, Magic Kernel Sharp

(b) Fig. 15 downsized with $\ell_2(x)$, the Lanczos-2 kernel

(c) Fig. 15 downsized with $\ell_3(x)$, the Lanczos-3 kernel

Figure 31: Comparison of using Magic Kernel Sharp, Lanczos-2, and Lanczos-3 as downsizing kernels on the test image of Fig. 15.

examination shows that Lanczos-2 possesses some aliasing artifacts not shared by the other two. Performing a "flip test" between Magic Kernel Sharp and Lanczos-3, on the other hand, shows that both of them are extremely high quality, with no evidence of aliasing visually apparent on either. However, such a test reveals that the *shapes* in one are slightly distorted from those of the other. With such a complicated, real-life example image as the original Fig. 12(a), it is difficult to determine which upsizing is more faithful, and which is the distortion. I will return to this question shortly, with a more controlled experiment.

For downsizing, it may again be difficult to discern the differences visually from what is shown in Fig. 31. However, careful pairwise "flip tests" establish that Magic Kernel Sharp is superior to Lanczos-2, and Lanczos-3 is superior to Magic Kernel Sharp.

So does all of this mean that Lanczos-3 wins?

Not quite. But to understand why, it is first useful to briefly review what is actually going on when we resample an image, in frequency space. This is the subject of the next section.

13. A review of resampling

To understand most clearly the arguments that I will make in the next section, it is worth briefly reviewing how resampling works in terms of Fourier theory.

A sampled signal h_n in position space can be considered to be the set of Fourier *series* (not transform) coefficients corresponding to a continuous function $H_s(f)$ that is defined in *frequency* space *only* within the "box" of the first Nyquist zone $|f| < \frac{1}{2}$, where, from the axioms of Fourier series, $H_s(f)$ must have periodic boundary conditions at the Nyquist boundaries. To resample h_n to a different sampling grid that has $k > 0$ times as many samples per unit position as the original h_n, we first need to "reconstruct" our best estimate $\tilde{h}(x)$ of the original $h(x)$ from which h_n was obtained by sampling, for all of those values of x that are not on the original sampling positions $x = n$, making use only of the samples h_n.

The first step in doing this is to recognize that we don't actually have any information at all about all of these other values $h(x \neq n)$, which we can represent mathematically by setting the corresponding values in $\tilde{h}_0(x)$, our zeroth approximation to the true $h(x)$, to zero:

$$\tilde{h}_0(x \neq n) \equiv 0. \tag{64}$$

This corresponds to multiplying the true, original $h(x)$ by a Dirac delta comb:

$$\tilde{h}_0(x) \equiv h(x)\,\mathrm{comb}\,x \equiv \sum_{n=-\infty}^{\infty} h_n\,\delta(x-n). \tag{65}$$

Per Eqs. (11) and (12), this definition of $\tilde{h}_0(x)$ means that its Fourier transform, $\tilde{H}_0(f)$, is the convolution of $\mathrm{comb}\,f$ with our original sampled $H_s(f)$ that was defined only within the first Nyquist zone:

$$\tilde{H}_0(f) \equiv H_s(f) * \mathrm{comb}\,f \equiv \sum_{n=-\infty}^{\infty} H_s(f-n), \tag{66}$$

in other words, a copy of $H_s(f)$ is placed at every integral value $f = n$, and the results summed up. This is fine, because $H_s(f)$ was only defined within the first Nyquist zone anyway, so (66) just represents us "tesselating" frequency space with copies of it, as described by Eq. (12).

What we would now *like* to do is simply "comb out" a new sampled function $h'(x)$ from $\tilde{h}_0(x)$ at the new sampling rate k:

$$h'(x) \equiv \tilde{h}_0(x)\,\mathrm{comb}(kx+c), \tag{67}$$

where c simply defines the starting point of the new sampling grid relative to the old. However, inserting (65) shows that (67) makes no sense:

$$h'(x) = \sum_{n=-\infty}^{\infty} h_n\,\delta(x-n) \sum_{m=-\infty}^{\infty} \delta[k(x-m)+c]. \tag{68}$$

For example, if k and c were unrelated irrational numbers, then none of the delta functions in the second sum would be at any of the locations of the delta functions in the first, and so the result would *appear* to be zero. And even if k and c weren't so irrational, we would be multiplying two delta functions together, which is not defined except as an integrand. Likewise, Eq. (67) makes no sense in frequency space either:

$$H'(f) = \frac{e^{-2\pi i c f/k}}{k}\,\mathrm{comb}\,\frac{f}{k} * \tilde{H}_0(f), \tag{69}$$

because from the definition (17) this is just

$$H'(f) = e^{-2\pi i c f/k} \sum_{n=-\infty}^{\infty} \tilde{H}_0(f-kn), \tag{70}$$

which at every value of f sums up an infinite number of finite samples from $H_0(f)$, which does not converge. Both of these observations simply highlight the fact that the product (67) is mathematically ill-defined, as it stands.

The way out of this problem is the phrase "except as an integrand" above: we need to convolve $\tilde{h}_0(x)$ with *some* other interpolation kernel $g(x)$, before we can do anything with it:

$$\tilde{h}(x) \equiv g(x) * \tilde{h}_0(x). \tag{71}$$

Determining a good but practical interpolation kernel $g(x)$ is what this whole game is about.

Let us first consider *upsampling*, *i.e.*, $k > 1$. Here we have more samples than we started with, so we know, from the Shannon–Hartley theorem, that we do not need to lose any of the information contained in the h_n. Our only task is to come up with an interpolation kernel $g(x)$ that does not *corrupt* this information.

Now, Eq. (71) means that, in frequency space,

$$\tilde{H}(f) = G(f)\,\tilde{H}_0(f). \tag{72}$$

Let us denote the resampled signal by $\hat{h}(x)$. We know that, in frequency space, the resampling will place copies of $\tilde{H}(f)$ at every frequency kn, namely, the equivalent of (70),

$$\hat{H}(f) = \sum_{n=-\infty}^{\infty} \tilde{H}(f-kn), \tag{73}$$

where for simplicity I am setting $c = 0$. The actual samples $\hat{h}_n$ depend on that part of $\hat{H}(f)$ that lies within the *new* first Nyquist zone, $|f| < k/2$, which is larger than the original first Nyquist zone, $|f| < 1/2$. We therefore need $G(f)$ to do two things: firstly, to "mask off" anything in $\tilde{H}_0(f)$ outside the original first Nyquist zone but within the new first Nyquist zone, namely,

$$\frac{1}{2} \leq |f| \leq \frac{k}{2}; \tag{74}$$

and secondly, to prevent (73) from corrupting this new first Nyquist zone, to "mask off" anything in $\tilde{H}_0(f)$ within $k/2$ of any of the positions kn for nonzero n, namely,

$$k\Big(n-\frac{1}{2}\Big) \leq f \leq k\Big(n+\frac{1}{2}\Big). \tag{75}$$

A moment's thought shows that (74) and (75) together require that $G(f)$ "mask off" everything outside the original first Nyquist zone. As a concrete example, consider the case of $k = 1.2$. Eq. (74) requires that $G(f)$ mask off everything within $-0.6 \leq f \leq -0.5$, and $+0.5 \leq f \leq +0.6$; Eq. (75) for $n = 1$ requires that it mask off everything within $+0.6 \leq f \leq +1.8$; for $n = 2$ it requires $+1.8 \leq f \leq +3.0$; and so on.

If $G(f)$ masks off everything perfectly, then the net effect of upsampling is that the part of $H(f)$ that fell within the original first Nyquist zone will appear in the resampled signal, with the spectrum for all frequencies between the original Nyquist frequency and the new Nyquist frequency being zero. Effectively, the original "tesselated" copies $H_s(f)$ have now been "cut apart" again, and "slid" along the frequency axis, leaving gaps between them.

The case of *downsampling* is a little more straightforward. In this case, we know that any components of $H(f)$ above the Nyquist frequency cannot be represented faithfully on the new sampling grid, and so we simply need $G(f)$ to mask off anything outside the *new* first Nyquist zone. Resampling will then "tesselate" frequency space with this new "cropped" spectrum.

The explicit constructions above may seem long-winded, but they will be of use when we consider the properties of the Lanczos kernels.

14. The fundamental flaw with the Lanczos kernels

Let us now return from that pedagogical interlude, and look again at the Fourier transforms of the Lanczos-2 and Lanczos-3 kernels, shown in Figs. 26 and 27.

The first sign that something is amiss is the fact that, in Fig. 26, it doesn't look like $L_2(0)$ is exactly unity,

$$L_a(0) \neq 1, \tag{76}$$

as I assured will always be the case in Eq. (10). This implies a violation of the normalization condition (9). Indeed, the standard definition (58) of the Lanczos kernels ensures that $\ell_a(0) = 1$, but it doesn't at all take into account the multiplication of the sinc kernel—which has correct normalization—by the Lanczos sinc window, which actually destroys it. Admittedly, the violations of (9) by (76) are *numerically* small—just 1% for $\ell_2(x)$, and 0.3% for $\ell_3(x)$—but it portends more serious problems with the fundamental design of the Lanczos kernels themselves.

Indeed, let me provide another zoom of both Figs. 26 and 27, in Figs. 32 and 33. As I first

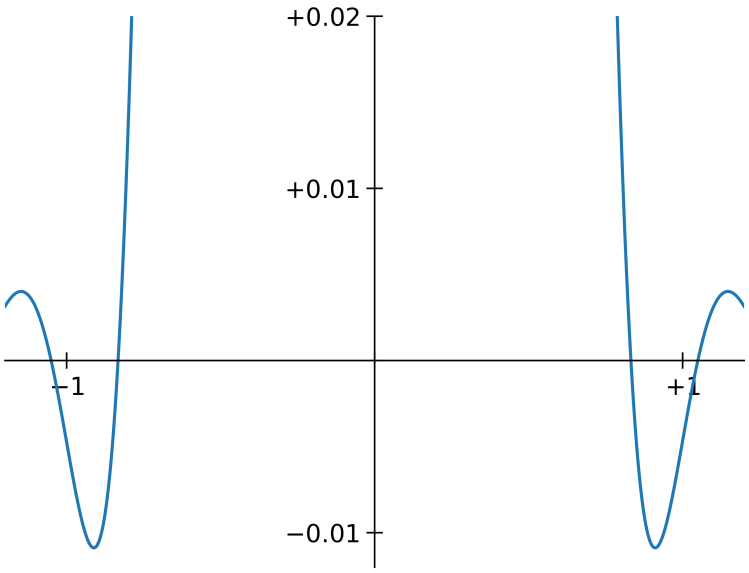


Figure 32: Another zoom of the $L_2(f)$ shown in Fig. 26. Note that there is no zero at $f = \pm 1$.

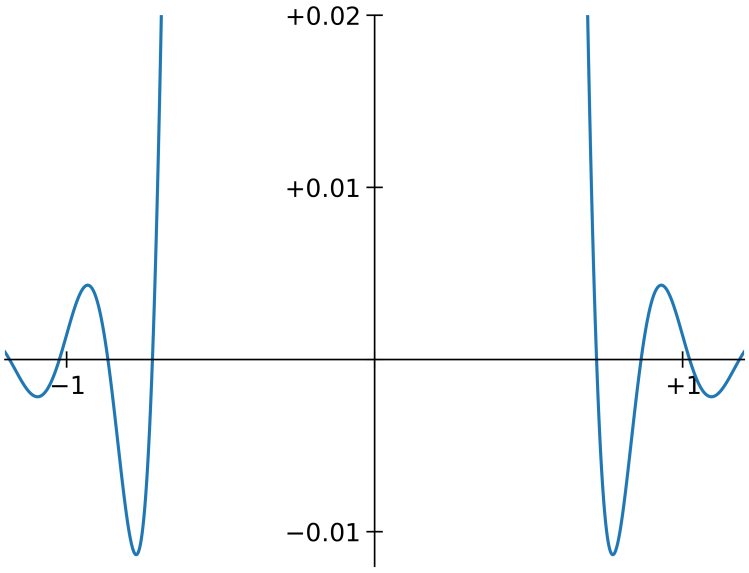


Figure 33: Another zoom of the $L_3(f)$ shown in Fig. 27. Note that there is no zero at $f = \pm 1$.

noted in 2015 [2], not only are all the zeros of $L_a(f)$ only *simple* zeros, they *moreover actually miss the integral values of* f. We therefore immediately know that the Lanczos kernels do not satisfy the *partition of unity* requirement of Eq. (57). This flaw is, of course, well-known, and puts the violation (76) into a more familiar context. What does *not* seem to be as well known, as far as I can tell, is what these properties around integral values of f actually imply for the fundamental fidelity of these kernels.

To investigate this further, consider first the case of upsizing. As described in the last section, in this case we simply want our kernel $G(f)$ to "mask off" everything outside the first Nyquist zone. Magic Kernel Sharp and the Lanczos kernels all do that, to a very good approximation. However, consider the ramifications of the small departures from ideality that they all possess: small weights of the alias copies of the true sampled spectrum $H_s(f)$ will survive the prefiltering process, and will appear as spurious additions to the final upsampled signal.

So far I have just described what is true for all practical approximations to the ideal rectangular window filter, and applies to all three of the kernels being discussed here. However, consider now the nature of the *signal*, $H(f)$, that we are processing. It is fair to say that the "overall structure" of an image—dogs, faces, buildings, plates of food, cats (can you tell yet that I work for Facebook?)—is contained in the *low-frequency* portion of $H(f)$; the high-frequency parts provide substructure, detail, textures, and so on. This might sound like a tautology, but in reality the "structure" of an image will generally be *coherent*, with this coherence being very visually meaningful, whereas the higher frequency parts will often be more decoherent and random, with its spectral energy providing important visual "weight," but for which the precise phase information is less critical. In most practical cases, this also means that most of the spectral energy of an image is concentrated around low frequencies (where "most" and "low" will vary for different types of images).

This practical consideration turns out to be critical for our current discussion. Focus on this coherent, low-frequency part of $H(f)$. Sampling places copies of this $H(f)$ at all nonzero integer values of f. *We had better filter out these copies particularly well*, because they are both strong and coherent. This means that *we want our kernel* $G(f)$ *to be particularly good at suppressing anything around nonzero integral values of* f.

The penny should by now have dropped. Magic Kernel Sharp possesses third-order zeros at all nonzero integral values, and these stationary points of inflection provide, by their very nature, significantly "wide" portions of frequency space around these integral values where $G(f)$ is close to zero (almost like a bathtub shape, except that one side turns down). For example, consider Fig. 34, which has the same scale as Figs. 32 and 33: this is going to do a much better job of eliminating the first aliased copies of the low-frequency $H(f)$ than what we see in either Fig. 32 or 33.

We can test this easily enough. Consider a Gaussian blob, with a radial standard deviation of, say, 3 pixels, so that its radial standard deviation in frequency space is $1/(2\pi\times 3) \approx 0.053$, which should have a good overlap with the lobes shown in Figs. 32 and 33. (The minima shown in Fig. 32 are at $|f| \approx 0.911$, and the maxima at $|f| \approx 1.147$; the first minima shown in Fig. 33 are at $|f| \approx 0.774$, the maxima at $|f| \approx 0.931$, and the second minima at $|f| \approx 1.094$.) Such a test image, and the results of upsizing it using Magic Kernel Sharp, Lanczos-2, and Lanczos-3, are shown in Fig. 35.

Whoa!

Both Lanczos upsizings are showing *pixelation*: something that we haven't seen since leaving

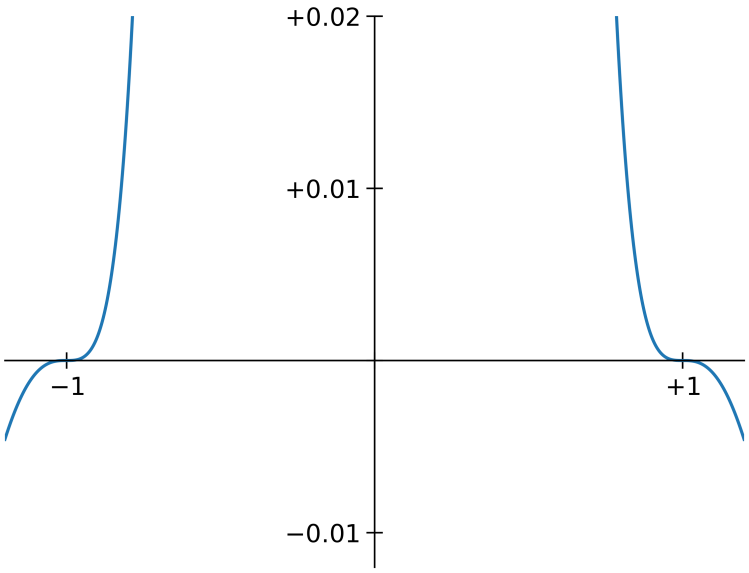


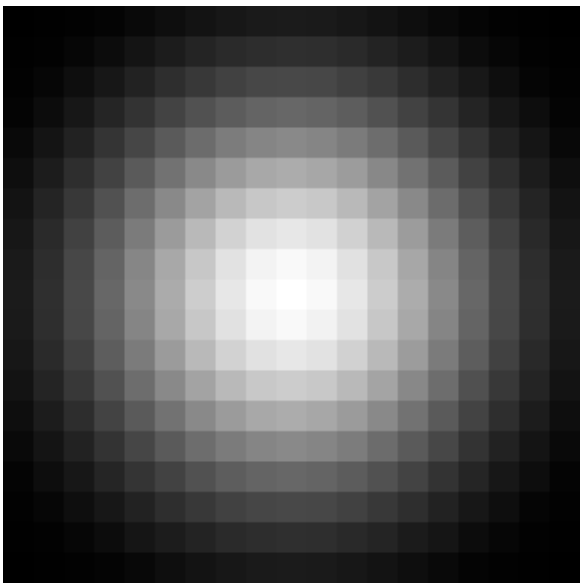
Figure 34: Zoom of the $K_3(f)$ shown in Fig. 19, to the same scale as Figs. 32 and 33. Note the third-order zero at $f = \pm 1$.

nearest neighbor resizing behind, in Fig. 14(a). Indeed, even the *linear* interpolation of Fig. 14(b) didn't show pixelation: its most glaring flaw was its "star twinkles." So how on Earth have we manage to regress all the way back to Page 15?

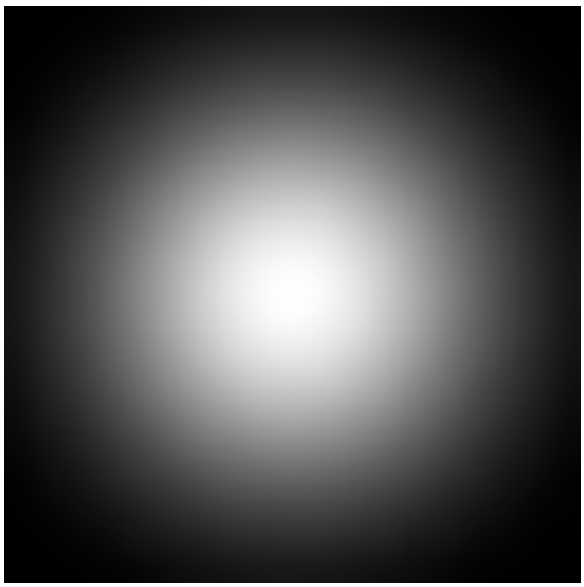
The answer is contained in our discussion above: the Lanczos kernels do not sufficiently suppress the aliased copies of the original spectrum $H(f)$ at all of the positions of the reciprocal sampling lattice in frequency space. When we then resample at a higher frequency, we are simply expanding our "square box" in frequency space, and "letting in" this lattice of aliasing artifacts. Now, by definition, each of these alias copies is centered on a lattice position that is a multiple of the original sampling frequency in each direction. This implies that, in position space, these aliasing artifacts will have a periodicity equal to that of the original sampling frequency—the lattice of artifacts in frequency space represents all the harmonics of these artifacts. And a general set of aliasing artifacts with periodicity in each direction equal to the original sampling frequency *is just pixelation*, as we see clearly in Fig. 35.

We can also understand why the Lanczos kernels have the same sort of pixelation problems as the nearest neighbor kernel: they all only have *simple zeros* at multiples of the sampling frequency. Even the linear kernel's second-order zeros are sufficient to suppress pixelation artifacts.

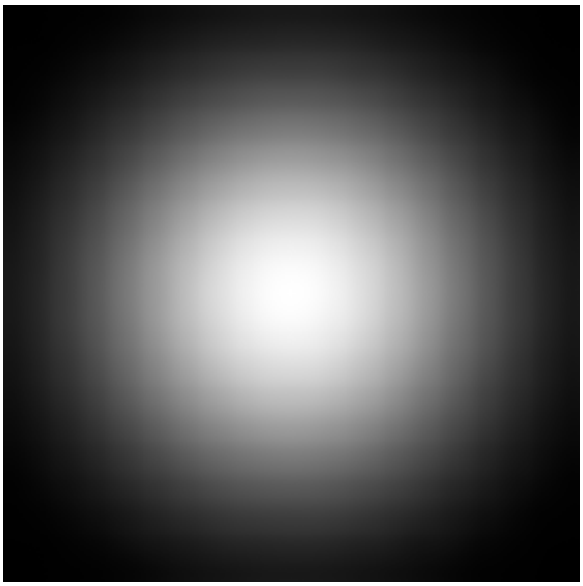
Clearly, the pixelation problem makes the Lanczos kernels manifestly unsuitable for upsizing applications: pixelation is such a visually objectionable property that it wasn't even necessary for me to try to defend it in the nearest neighbor upsampling of Fig. 14(a). Moreover, the admittance of aliasing artifacts based on the square, two-dimensional sampling lattice in reciprocal space *destroys rotational invariance*, as is again manifestly clear in Figs. 35(c) and 35(d). Even worse, it *creates subtle low-frequency distortions* in the output images. Indeed, we can now definitively resolve the mystery of which of Figs. 30(a) and 30(c) was the distortion: *it was the Lanczos-3 version, Fig. 30(c)*. (Reaching back to one of my previous jobs, as just one example, I know that any such subtle distortions in the field of *medical imaging* would be absolutely disastrous, with



(a) Original 19×19 image



(b) Upsized with Magic Kernel Sharp



(c) Upsized with Lanczos-2

(d) Upsized with Lanczos-3

Figure 35: Upsizing a Gaussian blob test image by a factor of 16.

(a) Flash on the left side of the vest (his right)

(b) Flash on the right side of the vest (his left)

Figure 36: Two frames of the "herringbone flashing" that plagued parts of a 4K HDR Netflix movie [7] that my wife Sally and I watched just the other night. In these two frames, the "flashing" appears on the two different sides of Chadwick Boseman's vest.

potentially serious safety, legal, and regulatory ramifications.)

Having thus deprecated the use of the Lanczos kernels for *upsizing*, are they still at least acceptable for use in *downsizing*?

Unfortunately not. In this case the problem is not the low-frequency structure of the signal $H(f)$, but rather its properties at *high* frequencies. Consider an image possessing a significant high-frequency component—say, a regular grid, or a periodic texture. Recall, from the review in the previous section, that if such a "spike" in frequency space lies outside the *new* first Nyquist zone under downsampling, then we need to first filter it out to prevent aliasing. This is always true. But consider the case of such a spike *lying near the new resampling frequency* (or a harmonic of it). Resampling will then *alias any surviving remnant of it to a low frequency.* This is particularly bad: as noted above, we make visual sense of the overall "structure" of an image from its low-frequency content, so any aliasing artifacts that manage to get into this critical part of frequency space can have devastating visual ramifications. For example, consider the low-frequency artifacts that currently seem to plague some 4K video content when the actor is wearing a "herringbone" fabric, such as is shown in Fig. 36. This "herringbone flashing" (as the actor moves) is so visually disturbing that you don't need a Ph.D. to be annoyed by it.

Now, as reviewed in the previous section, for downsampling we apply our "masking" kernel in the *new* sampling space, *i.e.*, with lower sampling frequency than the original, so that we discard those parts of $H(f)$ that lie outside the new first Nyquist zone, and hence cannot be represented on the new sampling space. We will therefore be at risk of these critical low-frequency aliasing artifacts *if our kernel does not provide sufficient suppression at its own sampling frequency* (and multiples of it).

Now, again, we know that Magic Kernel Sharp does provide such suppression, and the Lanczos kernels don't. The crucial question: is the magnitude of that lack of suppression sufficient to

create visual artifacts of practical consequence?

We can, again, test this easily enough. Consider as a test image a *vertical grating*, with the columns alternating black and white. This represents a spike right at the first Nyquist zone boundary, $|f| = 1/2$. If we downsize by a factor close to 2, then this spike will be near the sampling frequency of the new sampling space. Now, we noted that the maxima in Fig. 33 occur at $|f| \approx 0.931$, and this is not far from the minima shown in Fig. 32. If we choose to downsize our grating image by a factor of $k = 0.5/0.931 \approx 0.537$, then this spike will be located at $f \approx 0.931$ in the *new* sampling space. As this is well above the new Nyquist frequency $f = 0.5$, it *should* have been well and truly filtered out by our pre-filtering kernel; and since 0.931 is only 0.069 away from $f = 1$, we know that any surviving remnant will show up in our downsized image with a frequency of 0.069, *i.e.*, a period of around 14.5 pixels.

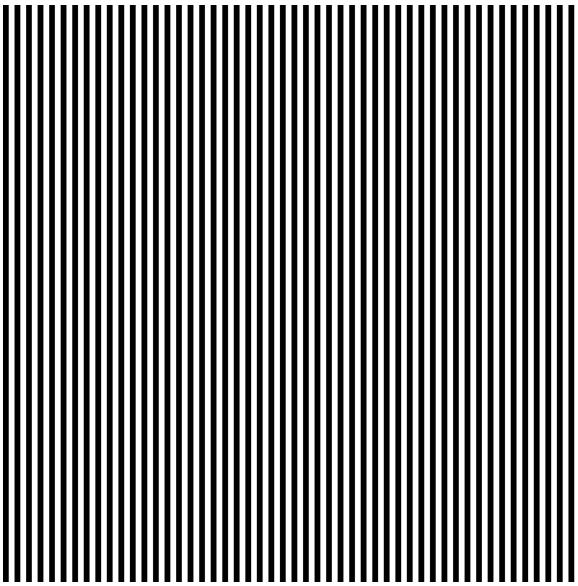
The results of this test are shown in Fig. 37. As suspected, both Lanczos kernels have allowed this high-frequency signal near the new sampling frequency to be aliased down to low frequencies without sufficient suppression, whereas Magic Kernel Sharp has sufficiently suppressed it. A more precise measurement shows that the peak-to-peak amplitude of the bands for Lanczos-2 is 2.1% of the full peak-to-peak amplitude of the original grating, and for Lanczos-3 it is 0.85%. These percentages are not large in linear space, but, as we have seen above, practical situations can arise when such bands are visible, particularly if digital signal processing is being performed as part of a processing pipeline, which may seek to enhance high frequencies (for the same reason that Instagram likes images to have extra "pop").

Of course, Magic Kernel Sharp is not perfect: a more precise measurement shows that, over each 14.5-pixel band, it produces single-pixel-wide positive and negative spikes (evidently from a curious sum of filtered harmonics) of amplitude ±0.14% of the full peak-to-peak amplitude of the original grating. It is not *a priori* obvious how to best compare these single-pixel spikes to the sinusoidal bands of the Lanczos filters; but, for want of a better metric, the mean absolute value of the latter is just the peak-to-peak amplitude divided by π, which is 0.67% for Lanczos-2 and 0.27% for Lanczos-3, and the mean absolute value for the Magic Kernel Sharp spikes is 0.019%. This suggests that, by this measure at least, Lanczos-2 is 35 times "worse" than Magic Kernel Sharp, and Lanczos-3 is 14 times "worse." In any case, we no longer find Magic Kernel Sharp straddling the performance gap between Lanczos-2 and Lanczos-3: there's now an order of magnitude difference.

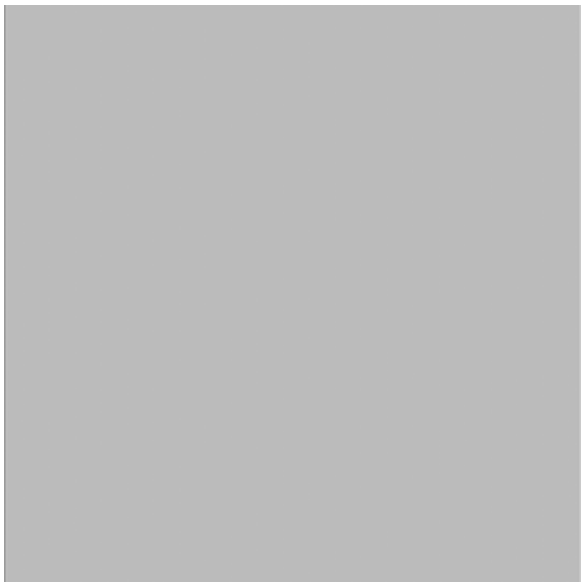
The crucial question now is: does this manifestation of the poor properties of the Lanczos kernels near harmonics of the sampling frequency rule it out for *downsizing* applications as well? I would argue that, given the outsized ramifications of this intrusion into low-frequency space (including, evidently, our Netflix viewing), it does.

We can now also reconsider the spectral energy leakage results that we summarized in Table 2. We did not distinguish between energy leaked to regions of frequency space around the harmonics of the sampling frequency, which we now see produce catastrophic low-energy aliasing artifacts, and all the rest of frequency space away from these harmonics, where aliasing artifacts will be of relatively high frequency, and will generally visually "average out" and not be as devastating as those at low frequency. So let us split $E_{G(f)}$ into a "harmonics" part, $E^{\text{harm}}_{G(f)}$, around the harmonics of the sampling frequency,

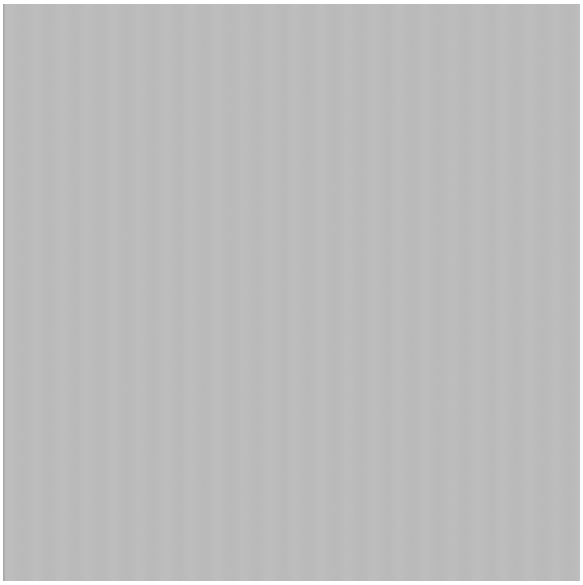
$$E^{\text{harm}}_{G(f)} \equiv \sum_{n \neq 0} \int_{n-0.15}^{n+0.15} df\, G(f), \tag{77}$$



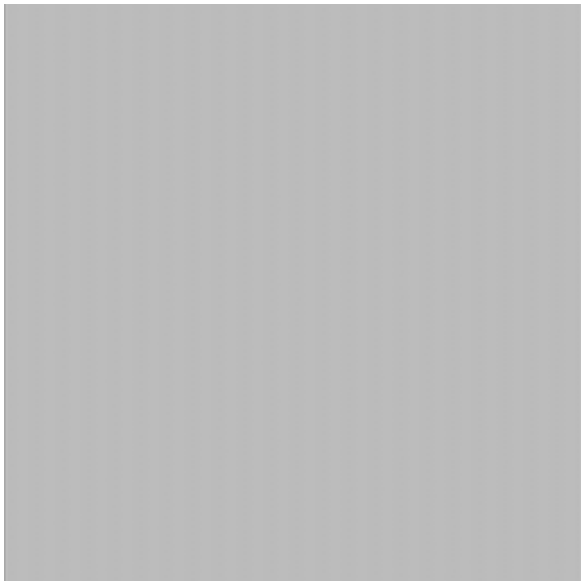
(a) 100×100 corner of original 600×600 image



(b) Downsized with Magic Kernel Sharp



(c) Downsized with Lanczos-2



(d) Downsized with Lanczos-3

Figure 37: Downsizing a test image of a vertical grating of alternating columns of black and white. Note that, to avoid aliasing issues with common PDF viewers, only the top-left 100×100 corner of the original 600×600 grating is shown in (a), and it is nearest-neighbor upsized to 500×500 to make its rendering here independent of your PDF viewer. All downsizings of the original 600×600 grating image are by a factor of 0.537 to a final size of 322×322.

Kernel	E_{harm}	E_{inter}
Nearest Neighbor	−21.4 dB	−10.3 dB
Linear Interpolation	−39.9 dB	−25.4 dB
Magic Kernel Sharp	−57.9 dB	−36.0 dB
Lanczos-2	−45.0 dB	−54.1 dB
Lanczos-3	−53.8 dB	−62.7 dB

Table 3: The harmonic and inter-harmonic leaked spectral energies for the five kernels.

and an "inter-harmonic" part, $E^{\text{inter}}_{G(f)}$, for all the rest:

$$E^{\text{inter}}_{G(f)} \equiv E_{G(f)} - E^{\text{harm}}_{G(f)}. \tag{78}$$

Computing now these two leaked spectral energies for each of the kernels we have analyzed, we obtain the results shown in Table 3. The harmonic column sheds light on why linear interpolation appears closer in visual fidelity to the Lanczos kernels than the metrics in Table 2 suggested, and why Magic Kernel Sharp significantly *outperforms* the Lanczos kernels on practical tests. Clearly, the Lanczos kernels have most of their leaked energy in the *harmonic* part—despite the inter-harmonic domain of frequency space being more than five times larger. This is exactly the wrong place for those flaws to land. In contrast, the other three kernels have most of their flaws *away* from the harmonics, effectively possessing *notch filters* at the harmonics of the sampling frequency—which is exactly what you want.

Of course, all practical decisions are a matter of trade-offs, and the only reason we are using these kernels at all is that we seek a hyperlocal solution in position space, for computational efficiency. If it were the case that Magic Kernel Sharp was nice theoretically, but much more expensive than the Lanczos kernels in practice, then the flaws in the latter might just be the price we have to pay. But, as I will now show, the reality is far different.

15. Computational efficiency of Magic Kernel Sharp

In addition to its natural avoidance of pixelation and low-frequency aliasing artifacts, Magic Kernel Sharp also has the distinct advantage over the Lanczos kernels that it is *more efficient*.

To see this, consider a general upsizing or downsizing operation for a D-dimensional signal with a kernel of support s, and let us denote a specific dimension by $d = 1, 2, \ldots, D$. Let us define the *resize factor*, r, to be the ratio of the larger dimension size to the smaller dimension size for each given dimension, *i.e.*, we always have $r > 1$. Let us denote by n_d the number of samples for dimension d for the smaller of the input and output spaces, and by $N \equiv n_1 n_2 \cdots n_D$ the total number of samples in this smaller space. For simplicity, let us disregard the rounding up or down of the (of course integral) numbers of input and output samples in each dimension, so that the larger number of samples for dimension d is taken to be $n_d\, r$, and the total number of samples in the larger space is $N r^D$. Let us also ignore all edge effects, where extension boundary conditions slightly reduce the number of operations required. As a final observation, note that the kernels considered in this paper are all separable, so we can perform the resizing sequentially through the dimensions.

Consider first the case of upsizing. We first resize in the first direction, $d = 1$. For each of the $n_2 n_3 \cdots n_D$ hyperplanes of the remaining dimensions, we resize the set of n_1 input samples of

dimension 1 into rn_1 output samples. Each of those output samples makes use of s of the input samples. The total number of operations is therefore $s \times rn_1 \times n_2 n_3 \cdots n_D \equiv rsN$.

We now upsize the second dimension, $d = 2$. This is just like what we did with the first dimension, except that there are now rd_1 input samples for dimension 1, rather than d_1 of them, because of our first upsizing. We therefore use $r^2 sN$ operations to upsize the second dimension.

We continue on until all of the dimensions are done. The total number of operations for an upsizing is therefore

$$sN \sum_{d=1}^{D} r^d. \tag{79}$$

Now, *any* resizing algorithm must use at least Nr^D operations, if it makes *some* use of each and every one of the input samples: for upsampling, there are Nr^D output samples, and they all need at least one operation to compute; for downsampling, there are Nr^D input samples, and we are assuming that all of them contribute in some way to at least one output sample. Let us therefore define a normalized relative *computational cost* factor, C_k, as the number of operations required by a given resizing kernel k divided by its minimum possible value Nr^D, so that, for upsizing, from Eq. (79) we have

$$C_k(r_k) = \frac{s_k}{r_k^D} \sum_{d=1}^{D} r_k^d, \tag{80}$$

where I am now also emphasizing that C_k is a function of r_k, the resizing factor that we are using kernel k to implement.

We now turn to downsizing. A moment's thought shows that this is just the time-reverse of upsizing, so that the total number of operations is the same result (79), when written in terms of what is now the output space number of samples N, and so the computational cost $C_k(r_k)$ is still given by Eq. (80).

Now consider same-size filtering in the smaller space, as is done for Magic Kernel Sharp, with the filtering kernel k having support s_k. We filter the first dimension, which requires $s_k N$ operations. The second dimension also requires $s_k N$ operations, because the filtering did not change the number of samples. And so on, for all D dimensions, so that overall we require $s_k ND$ operations, which of course is also just the same result (79), now with $r_k = 1$, and so Eq. (80) again holds.

Finally, if a kernel k is the composition of some number of other kernels k', then (80) simply becomes

$$C_k(r) = \sum_{k'} \frac{s_{k'}}{r_{k'}^D} \sum_{d=1}^{D} r_{k'}^d. \tag{81}$$

Note that the *order* of applying the kernels matters in operational terms, but there is no ordering needed in Eq. (81) provided that we suitably define the $r_{k'}$.

Let us now consider some practical cases for two-dimensional images, so that $D = 2$.

For Magic Kernel Sharp resizing, we can choose the first kernel, $k = 1$ to be (say) the Magic Kernel resizing kernel $m(x)$, with $s_1 = 3$ and $r_1 \equiv r$. For Magic Kernel Sharp 2013, $k = 2$ is the

Sharp 2013 kernel, $s^{2013}(x)$, with $s_2 = 3$ and $r_2 = 1$. For Sharp 2021, we add on the Correction 2021 kernel, $c^{2021}(x)$, as $d = 3$, with $s_3 = 3$ and $r_3 = 1$. Thus

$$C_{\text{Magic Kernel Sharp 2013}}(r) = 3 + \frac{3}{r} + \frac{3}{r^2}, \tag{82}$$

and

$$C_{\text{Magic Kernel Sharp 2021}}(r) = 3 + \frac{3}{r} + \frac{6}{r^2}. \tag{83}$$

For Lanczos-a resizing, we just have a single kernel $k = 1$, with support $s_1 = 2a$. Thus

$$C_{\text{Lanczos-}a}(r) = 2a + \frac{2a}{r}. \tag{84}$$

We now ask the question: for which values of r is Magic Kernel Sharp less computationally costly than Lanczos-a? Clearly, for large enough r, Magic Kernel Sharp is always more efficient, since its asymptotic value of 3 is always less than the $2a$ of Lanczos-a (since $a = 1$ is not a useful kernel). So where is the crossover point? Clearly, when

$$3 + \frac{3}{r} + \frac{s_s}{r^2} = 2a + \frac{2a}{r}. \tag{85}$$

where s_s is sum of the supports of the kernels used to construct the approximation of the Sharp kernel, *i.e.*, $s_s = 3$ for the 2013 version and $s_s = 6$ for the 2021 version. The solution to (85) is the rather inelegant

$$r_{\text{crossover}} = \frac{\sqrt{4a^2+4a(2s_s-3)+9-12s_s}}{2(2a-3)} - \frac{1}{2}. \tag{86}$$

However, it is simple enough to evaluate it for the four cases of interest to us.

For Lanczos-3, the crossover with Magic Kernel Sharp 2013 occurs at $r = (\sqrt{5}-1)/2 \approx 0.618$, and the crossover with Magic Kernel Sharp 2021 occurs at $r = 1$. However, since by definition $r > 1$, this tells us that *both versions of Magic Kernel Sharp are always more efficient than Lanczos-3*.

For Lanczos-2, the crossover with Magic Kernel Sharp 2013 occurs at $r = (\sqrt{13}-1)/2 \approx 1.303$, and the crossover with Magic Kernel Sharp 2021 occurs at $r = 2$. However, we know that Magic Kernel Sharp *always* has performance superior to Lanczos-2, so these two results are of little practical importance. (For the 2013 implementation of Magic Kernel Sharp at Facebook, Lanczos-2 would not have been implemented even if it had been possible to do so, because the values of r for which it would have been more efficient—less than around 1.3—are always avoided if at all possible—if there is a larger stored version of the given photo available—to avoid artifacts arising from multiple resizings.)

We thus see that Magic Kernel Sharp is always preferable to Lanczos-3 on computational efficiency grounds alone, and is more efficient than the inferior Lanczos-2 for most resizing factors.

16. More Cowbell

In the above I have shown why Magic Kernel Sharp—in either its 2013 or 2021 approximations, or even with an extra set of coefficients for even higher dynamic range applications—is a

manifestly practical, hyperlocal, high-quality resizing kernel. But even the exact Magic Kernel Sharp is not perfect: it still has some spectral energy outside the first Nyquist zone (although most of it is inter-harmonic, unlike Lanczos-3), and we do still see some subtle aliasing artifacts in the zone plate stress-test of Fig. 31(a).

For applications for which even higher fidelity may be required—perhaps astronomy, medical imaging, or signal processing—it is, as noted in Sec. 5, possible to extend the process further: to $k_4(x)$ or higher. The generalization to $k_4(x)$ would be particularly straightfoward: convolving the Magic Kernel $m(x)$ one extra time with rect x would produce the fourth kernel in the fundamental sequence, with support from $x = -2$ to $x = +2$. The discrete same-size blur kernel implied by this continuous kernel would still only have three nonzero coefficients, at $x = -1$, 0, and $+1$, and so its Fourier transform, and that of its inverse, the sharp kernel, should be obtainable using nothing more than the methods outlined in Secs. 7 and 9, simply with changed coefficients.

We know that the result of this construction, $k_4(x)$, would possess a Fourier transform $K_4(f)$ that is "squarer" again than the $K_3(f)$ of Magic Kernel Sharp, with less spectral energy outside the first Nyquist zone, from our discussion of the ramifications of the projection identity ansatz, namely, Eq. (47).

Now, how $K_4(f)$ would actually compare to the Lanczos-3 kernel $L_3(f)$ on these two measures would be an interesting question. But without answering it, let's turn this question on its head: We know that the computational cost of the Lanczos-3 kernel is manifestly acceptable for practical purposes, since it *is* used in practice. Shouldn't we compare it to the kernel in our fundamental sequence *that has the same computational cost*?

I would argue that this is a manifestly reasonable requirement. So which kernel $k_a(x)$ in our fundamental sequence satisfies this requirement?

To answer that, let us first expand our definitions and notation to more easily handle the case of arbitrary a. Let us define the *a-th Magic Kernel*, $m_a(x)$, to be the convolution of rect x with itself a times, which we can express formally as the recurrence relation

$$\begin{aligned} m_0(x) &\equiv \delta(x), \\ m_a(x) &\equiv \operatorname{rect} x * m_{a-1}(x). \end{aligned} \tag{87}$$

The a-th discrete *Blur Kernel*, $b_a(x)$, is defined to be the comb projeciton of $m_a(x)$:

$$b_a(x) \equiv m_a(x) \operatorname{comb} x. \tag{88}$$

The a-th discrete *Sharp Kernel*, $s_a(x)$, is defined to be the inverse of $b_a(x)$:

$$s_a(x) * b_a(x) \equiv \delta(x). \tag{89}$$

Finally the *a-th Magic Kernel Sharp kernel*, $k_a(x)$, is defined to be the composition of the a-th Magic Kernel and the a-th Sharp kernel:

$$k_a(x) \equiv m_a(x) * s_a(x). \tag{90}$$

Now, clearly, from the definition (87),

$$\begin{aligned} \operatorname{supp} m_0(x) &= 0, \\ \operatorname{supp} m_a(x) &= 1 + \operatorname{supp} m_{a-1}(x), \end{aligned} \tag{91}$$

and so

$$\operatorname{supp} m_a(x) = a. \tag{92}$$

The computational cost $C_{k_a(x)}$ of Magic Kernel Sharp a is now just the sum of the cost of Magic Kernel a,

$$C_{m_a(x)} = a + \frac{a}{r}, \tag{93}$$

and that of Sharp a,

$$C_{s_a(x)} = \frac{s_s}{r^2}, \tag{94}$$

namely,

$$C_{k_a(x)} = a + \frac{a}{r} + \frac{s_s}{r^2}. \tag{95}$$

In comparing this to the computational cost of the Lanczos-a kernel, namely, Eq. (84), the latter does not have any filtering in the smaller space, so it does not have a $1/r^2$ term. If we ignore this second-order correction, by ignoring the contribution (94) of the Sharp filtering, we find that, to this approximation, *Lanczos-a has the same computational cost as Magic Kernel Sharp 2a.*

Thus, if we want to perform an apples-to-apples comparison of Magic Kernel Sharp and Lanczos-3, then it is *Magic Kernel Sharp 6*, $k_6(x)$, that we should be comparing.

The *sixth* member of our sequence? That sounds daunting: it has taken plenty of work just to get to the *third* member. But it's actually not as onerous as it sounds.

Out first task is to construct Magic Kernel 6, $m_6(x)$. Now, how do we *actually* compute $m_a(x)$, from first principles? I didn't actually describe that in the previous sections, so let me do it now.

First, making use of the identity (20) on a function that itself is bounded by a rect function, a little thought yields the identity

$$\operatorname{rect} x * \operatorname{rect}(x-u)\, g(x) \equiv \operatorname{rect}\Big(x-u+\tfrac{1}{2}\Big)\int_{u-\frac{1}{2}}^{x+\frac{1}{2}} dx' g(x') + \operatorname{rect}\Big(x-u+\tfrac{1}{2}\Big)\int_{x-\frac{1}{2}}^{u+\frac{1}{2}} dx' g(x'), \tag{96}$$

which says that convolving $\operatorname{rect} x$ with a function that is rect-windowed around $x = u$ creates a new function that is a "split" sum of a new function that is rect-windowed around $x = u-\frac{1}{2}$ and another new function that is rect-windowed around $x = u+\frac{1}{2}$. Applying this to an arbitrary polynomial, we thus have

$$\begin{aligned} \operatorname{rect} x * \operatorname{rect}(x-u)\sum_n c_n x^n \;\equiv\; & \operatorname{rect}\Big(x-u+\tfrac{1}{2}\Big)\sum_n \frac{c_n}{n+1}\Big\{\Big(x+\tfrac{1}{2}\Big)^{n+1} - \Big(u-\tfrac{1}{2}\Big)^{n+1}\Big\} \\ & + \operatorname{rect}\Big(x-u-\tfrac{1}{2}\Big)\sum_n \frac{c_n}{n+1}\Big\{\Big(u+\tfrac{1}{2}\Big)^{n+1} - \Big(x-\tfrac{1}{2}\Big)^{n+1}\Big\}. \end{aligned} \tag{97}$$

We now start with Magic Kernel 1, the nearest neighbor kernel.

$$m_1(x) \equiv \operatorname{rect} x, \tag{98}$$

and apply (97) to it, where $u = 0$ and $c_0 = 1$, to obtain the formula for Magic Kernel 2, the linear interpolation kernel,

$$m_2(x) \equiv \text{rect}\, x * m_1(x) = \text{rect}\Big(x+\tfrac{1}{2}\Big)(x+1) + \text{rect}\Big(x-\tfrac{1}{2}\Big)(-x+1), \tag{99}$$

which is just the expected "tent" shape. Applying (97) to (99), in turn, yields

$$\begin{aligned} m_3(x) \equiv \text{rect}\, x * m_2(x) &= \text{rect}(x+1)\left\{\frac{1}{2}x^2 + \frac{3}{2}x + \frac{9}{8}\right\} \\ &+ \text{rect}\, x\left\{-x^2 + \frac{3}{4}\right\} \\ &+ \text{rect}(x-1)\left\{\frac{1}{2}x^2 - \frac{3}{2}x + \frac{9}{8}\right\}, \end{aligned} \tag{100}$$

which is just the Magic Kernel formula (27) that I originally deduced empirically ten years ago.

We now move into uncharted territory. Applying (97) another time, we obtain the expression for Magic Kernel 4, $m_4(x)$

$$\begin{aligned} m_4(x) &= \text{rect}\Big(x+\frac{3}{2}\Big)\left\{+\frac{1}{6}x^3 + x^2 + 2x + \frac{4}{3}\right\} \\ &+ \text{rect}\Big(x+\frac{1}{2}\Big)\left\{-\frac{1}{2}x^3 - x^2 + \frac{2}{3}\right\} \\ &+ \text{rect}\Big(x-\frac{1}{2}\Big)\left\{+\frac{1}{2}x^3 - x^2 + \frac{2}{3}\right\} \\ &+ \text{rect}\Big(x-\frac{3}{2}\Big)\left\{-\frac{1}{6}x^3 + x^2 - 2x + \frac{4}{3}\right\}. \end{aligned} \tag{101}$$

It is shown graphically in Fig. 38. Note that, as expected, it is slightly more spread out than Magic Kernel 3 in Fig. 1 (note that the scale is $-3 \le x \le +3$ for Fig. 39, whereas it was $-2 \le x \le +2$ for Fig. 1). It can be verified that $m_4(x)$ is everywhere continuous up to and including its *second* derivative, not just its first, as was true for $m_3(x) \equiv m(x)$.

Let us push on. Now, I did the algebraic calculations above by hand, but it is simple to program a computer algebra system such as Sage to do them for us. For example, the code

```
from collections import defaultdict
from pprint import pprint
var('x')

def recurse(f):
  n = defaultdict(int)
  for u, g in f.items():
    n[u - 1/2] += g.integral(x, u - 1/2, x + 1/2)
    n[u + 1/2] += g.integral(x, x - 1/2, u + 1/2)
  return dict(n)
```

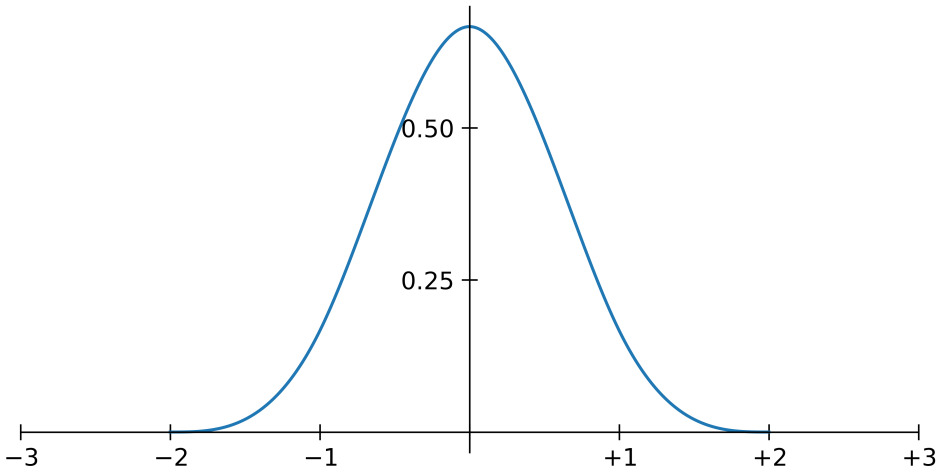


Figure 38: Magic Kernel 4, $m_4(x)$.

```
m = {1: {0: x^0}}
for a in range(2, 7):
  m[a] = recurse(m[a - 1])
pprint(m)
```

yields the output

```
{1: {0: 1},
 2: {-1/2: x + 1, 1/2: -x + 1},
 3: {-1: 1/2*x^2 + 3/2*x + 9/8, 0: -x^2 + 3/4, 1: 1/2*x^2 - 3/2*x + 9/8},
 4: {-3/2: 1/6*x^3 + x^2 + 2*x + 4/3,
     -1/2: -1/2*x^3 - x^2 + 2/3,
     1/2: 1/2*x^3 - x^2 + 2/3,
     3/2: -1/6*x^3 + x^2 - 2*x + 4/3},
 5: {-2: 1/24*x^4 + 5/12*x^3 + 25/16*x^2 + 125/48*x + 625/384,
     -1: -1/6*x^4 - 5/6*x^3 - 5/4*x^2 - 5/24*x + 55/96,
     0: 1/4*x^4 - 5/8*x^2 + 115/192,
     1: -1/6*x^4 + 5/6*x^3 - 5/4*x^2 + 5/24*x + 55/96,
     2: 1/24*x^4 - 5/12*x^3 + 25/16*x^2 - 125/48*x + 625/384},
 6: {-5/2: 1/120*x^5 + 1/8*x^4 + 3/4*x^3 + 9/4*x^2 + 27/8*x + 81/40,
     -3/2: -1/24*x^5 - 3/8*x^4 - 5/4*x^3 - 7/4*x^2 - 5/8*x + 17/40,
     -1/2: 1/12*x^5 + 1/4*x^4 - 1/2*x^2 + 11/20,
     1/2: -1/12*x^5 + 1/4*x^4 - 1/2*x^2 + 11/20,
     3/2: 1/24*x^5 - 3/8*x^4 + 5/4*x^3 - 7/4*x^2 + 5/8*x + 17/40,
     5/2: -1/120*x^5 + 1/8*x^4 - 3/4*x^3 + 9/4*x^2 - 27/8*x + 81/40}}
```

which is just the first six Magic Kernels that we want (and as many more as you might desire). Magic Kernel 5, $m_5(x)$, is shown in Fig. 39, and Magic Kernel 6, $m_6(x)$, is shown in Fig. 40, Again, they are getting slightly more spread out with each iteration. We can understand this on

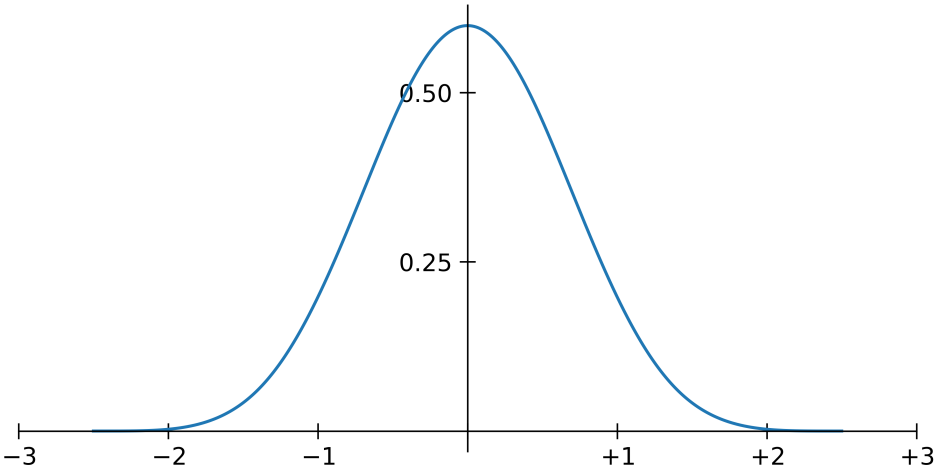


Figure 39: Magic Kernel 5, $m_5(x)$.

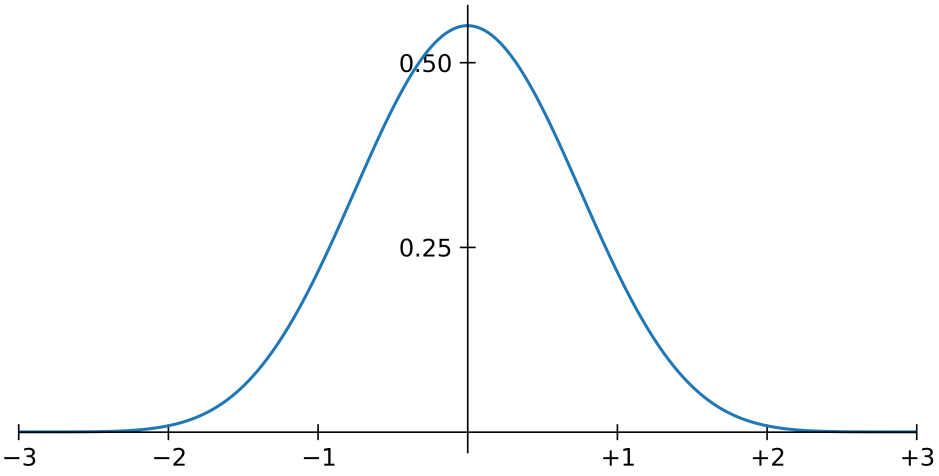


Figure 40: Magic Kernel 6, $m_6(x)$.

elementary grounds from the fact that the variance of rect x is $1/12$, and that each convolution with itself will add another $1/12$ to its variance, so that

$$\mathrm{Var}(m_a(x)) = \frac{a}{12}, \tag{102}$$

so that the standard deviation increases as the square root of a:

$$\mathrm{SD}(m_a(x)) = \sqrt{\frac{a}{12}}. \tag{103}$$

For example, I determined from first principles ten years ago that the standard deviation of the Magic Kernel is $1/2$, as (103) confirms; the standard deviation of Magic Kernel 6 has only increased to $1/\sqrt{2}$.

Now, as should be clear from the construction in Secs. 7 and 8, we can write down the frequency-space representations of $K_a(f)$ directly from the sampled values of $m_a(x)$:

$$K_a(f) \equiv \mathrm{sinc}^a f \left\{ m_a(0) + 2\sum_{n=1}^{N_a} m_a(n)\cos(2\pi n f) \right\}^{-1}, \tag{104}$$

where

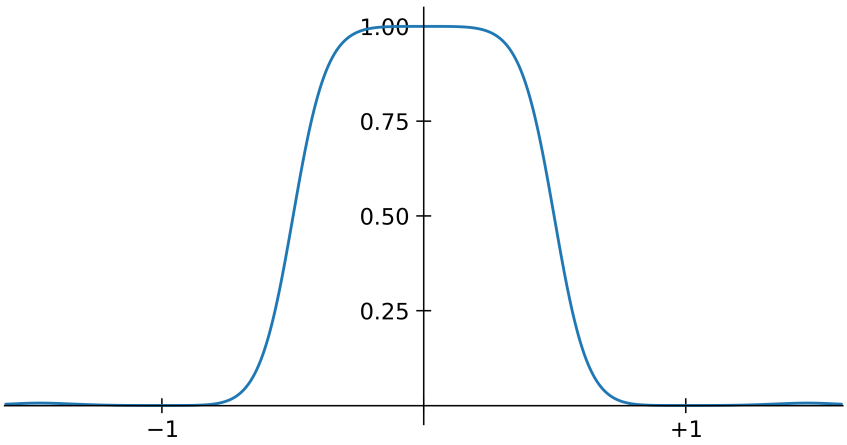
$$N_a \equiv \left\lfloor \frac{a-1}{2} \right\rfloor \tag{105}$$

is the largest sampled position within the support of $m_a(x)$, *i.e.*, $N_1 = N_2 = 0$, $N_3 = N_4 = 1$, and $N_5 = N_6 = 2$. From the expressions listed above, Sage dutifully provides the required $m_a(n)$:

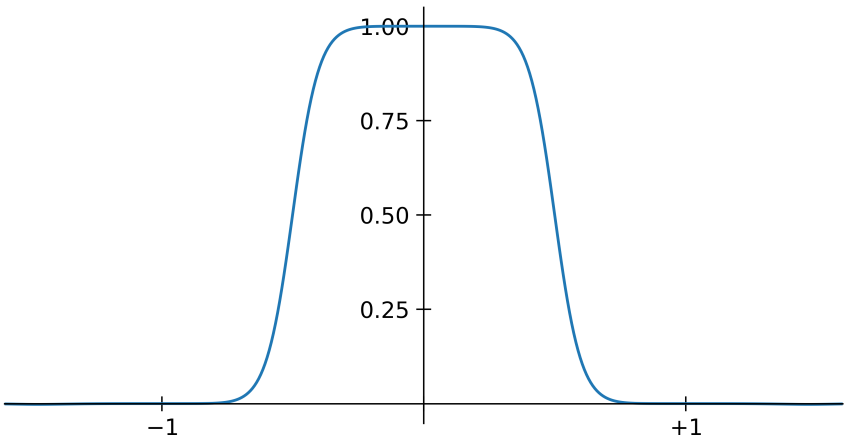
$$\begin{aligned}
m_1(0) &= 1, \\
m_2(0) &= 1, \\
m_3(0) &= \frac{3}{4}, \quad m_3(1) = \frac{1}{8}, \\
m_4(0) &= \frac{2}{3}, \quad m_4(1) = \frac{1}{6}, \\
m_5(0) &= \frac{115}{192}, \quad m_5(1) = \frac{19}{96}, \quad m_5(2) = \frac{1}{384}, \\
m_6(0) &= \frac{11}{20}, \quad m_6(1) = \frac{13}{60}, \quad m_6(2) = \frac{1}{120},
\end{aligned} \tag{106}$$

and the corresponding frequency space representations $K_4(f)$, $K_5(f)$, and $K_6(f)$ are shown in Fig. 41. As expected, they continue to approach the ideal square shape.

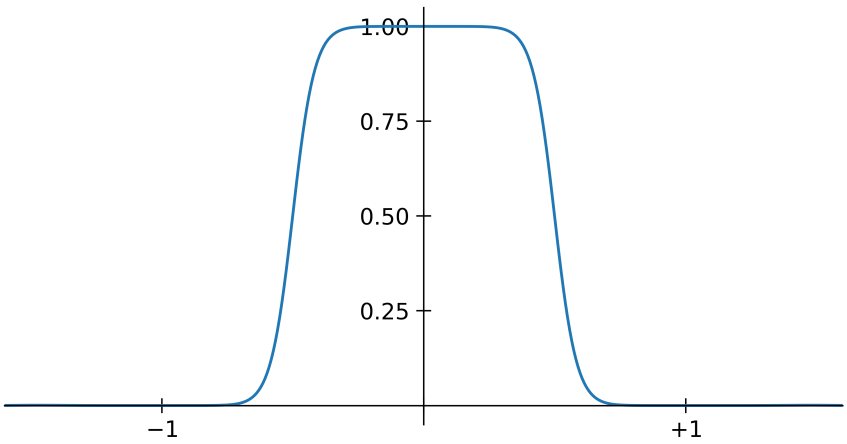
We can compute the leaked spectral energy for these three kernels, and add the results to Table 3, as shown in Table 4. We see that, remarkably, Magic Kernel Sharp 6 actually outperforms its same-sized competitor Lanczos-3 even for *inter*-harmonic leaked energy; but it is the 53.1 dB advantage for harmonic leaked energy that really stands out, and should put to rest any question about which of these two kernels should be used in practice.



(a) $K_4(f)$



(b) $K_5(f)$



(c) $K_6(f)$

Figure 41: Magic Kernel Sharp 4, 5, and 6 in frequency space.

Support	Kernel	E_{harm}	E_{inter}
1	Nearest Neighbor	−21.4 dB	−10.3 dB
2	Linear Interpolation	−39.9 dB	−25.4 dB
3	Magic Kernel Sharp 3	−57.9 dB	−36.0 dB
4	Lanczos-2	−45.0 dB	−54.1 dB
	Magic Kernel Sharp 4	−74.6 dB	−47.0 dB
5	Magic Kernel Sharp 5	−90.9 dB	−57.3 dB
6	Lanczos-3	−53.8 dB	−62.7 dB
	Magic Kernel Sharp 6	−106.9 dB	−67.6 dB

Table 4: The harmonic and inter-harmonic leaked spectral energies for all of the resizing kernels analyzed in this paper, ordered by increasing support in position space.

All that is now left for us to do is determine the functional forms of $k_4(x)$, $k_5(x)$, and $k_6(x)$ in *position* space, namely, by determining the coefficients of Sharp 4, 5, and 6. Now, while it is possible to compute these coefficients analytically as a power series expansion as I did in Sec. 9, it is of more practical use to understand how to derive the sequence of practical approximations to it, namely, the generalizations of Sharp 2013 and Sharp 2021. I didn't actually explain how I obtained those filters in the previous sections, so let me do that now.

The method is fundamentally just reversion of the coefficients of the discrete blur kernel implied by $m_a(x)$, which in Sec. 7 and above in (88) I denoted b_n for pedagogical simplicity, but which I will now refer to simply as $m_{a,n}$. For the v-th approximation $s_a^{(v)}(x)$, to $s_a(x)$, we seek to make all of the v values $k_{a,n}^{(v)} \equiv s_{a,n}^{(v)} * m_{a,n}$ for $1 \le |n| \le v$ vanish. If the remaining values $k_{a,|n|>v}^{(v)}$ are small enough, then the approximation v is a useful one, and the magnitudes of those residual terms tell us how much numerical accuracy we have with the approximation.

Now, there is a general method that works for any v for all Sharp a, and there is a simpler method that is recursive in v, more computationally efficient than the general method, that works only for Sharp 3 and Sharp 4. Let me first start with that latter simpler method.

The unique property shared by $m_3(x)$ and $m_4(x)$, of course, is that the blur kernels $m_{a,n}$ are "three-tap," having nonzero values only at $n = 0$ and ± 1. Our first approximation $s_{a,n}^{(1)}$ to $s_{a,n}$ can then be immediately constructed by simply subtracting two equally weighted copies of the $m_{a,n}$ from $m_{a,n}$ itself, one weighted copy shifted left one position and one shifted right:

$$k_{a,n}^{(1)} \equiv \frac{1}{1+2w_{a,1}}\Big(m_{a,n} + w_{a,1}m_{a,n+1} + w_{a,1}m_{a,n-1}\Big), \tag{107}$$

with the weight $w_{a,1}$ chosen so that the sum satisfies our requirement that

$$k_{a,-1}^{(1)} = k_{a,+1}^{(1)} = 0. \tag{108}$$

Eq. (107) is of course just the convolution of $m_{a,n}$ with the three-tap Sharp kernel $s_{a,n}^{(1)}$ with coefficients

$$s_{a,0}^{(1)} = \frac{1}{1+2w_{a,1}}, \qquad s_{a,\pm 1}^{(1)} = \frac{w_{a,1}}{1+2w_{a,1}}. \tag{109}$$

Clearly, (107) and (108) are solved if

$$m_{a,1} + w_{a,1}m_{a,0} = 0, \tag{110}$$

so that

$$w_{a,1} = -\frac{m_{a,1}}{m_{a,0}}, \tag{111}$$

and hence the coefficients Eq. (109) of the first approximation to Sharp a are just

$$s^{(1)}_{a,0} = \frac{m_{a,0}}{m_{a,0} - 2m_{a,1}}, \qquad s^{(1)}_{a,\pm 1} = \frac{-m_{a,1}}{m_{a,0} - 2m_{a,1}}. \tag{112}$$

For approximation 1 of Sharp 3, we have

$$s^{(1)}_{3,0} = +\frac{3}{2}, \qquad s^{(1)}_{3,\pm 1} = -\frac{1}{4}, \tag{113}$$

which is just what is stated in Eq. (30) for Sharp 2013, yielding the set of $k^{(1)}_{3,n}$,

$$k^{(1)}_{3,0} = +\frac{17}{16}, \qquad k^{(1)}_{3,\pm 2} = -\frac{1}{32}. \tag{114}$$

which is what is shown in Eq. (32). For Sharp 4, we find

$$s^{(1)}_{4,0} = +2, \qquad s^{(1)}_{4,\pm 1} = -\frac{1}{2}, \tag{115}$$

with resulting $k^{(1)}_{4,n}$ of

$$k^{(1)}_{4,0} = +\frac{7}{6}, \qquad k^{(1)}_{4,\pm 2} = -\frac{1}{12}. \tag{116}$$

Note that, whereas Eq. (114) for approximation 1 of Magic Kernel Sharp 3 represented a satisfaction of the projection identity (29) to 4 bits of accuracy, Eq. (116) only satisfies this identity to around 2.6 bits of accuracy. This is a consequence of the fact that Magic Kernel 4 is slightly wider than Magic Kernel 3—it has a standard deviation in position space of $1/\sqrt{3} \approx 0.58$ compared to 1/2 for Magic Kernel 3—and hence the $m_{4,\pm 1}$ are larger than the corresponding $m_{3,\pm 1}$: effectively, the method I am describing here converges less slowly for $a = 4$ than it does for $a = 3$. This is a general property of these approximations, for all a, and the greater consequent support required of the Sharp kernel, for any given level of numerical accuracy of identity (29) that we might demand, is the price we have to pay for the increased fidelity of the overall kernel that we obtain by going to larger a.

However, it is, at this point, worth emphasizing most strongly an absolutely critical point: *the antialiasing properties of Magic Kernel Sharp come completely from the Magic Kernel.* The Sharp step merely "flattens" the frequency response within as much of the first Nyquist zone as it can: without it, Magic Kernel a for $a \geq 3$ slightly blurs the image; if we include a first approximation to Sharp, then the result "overshoots" a little, and the resulting image is then slightly sharpened, compared to an ideal resizer. For many practical purposes, obtaining that flat frequency response within that portion of the first Nyquist zone is not a high priority at all, compared to the overwhelmingly important concern of avoiding aliasing artifacts (and particularly those that manage to invade the low frequency inner sanctum). For example, as noted in [1], Facebook

uses just Sharp 2013, the three-tap approximation in Eq. (113) above, and so all of the quarter-trillion photos on Facebook (a public number [8]) are slightly sharper than an ideal resizer would produce—and Instagram actually adds a little more sharpening to its 40 billion (another public number [9]). I am not personally aware of any of our billions of users complaining that their photos are slightly too crisp. *Aliasing artifacts*, on the other hand, are a completely different matter, and that is what I fixed back in early 2013 by building Magic Kernel Sharp. (JPEG compression artifacts are the third member of this trifecta, which can never be completely eliminated for all corner cases in any practical system that uses lossy compression.)

Keeping in mind these practical considerations, then, let us continue on and look at how to generate further ($v > 1$) approximations to Sharp 3 and Sharp 4. Let me first define a "zeroth" approximation $k^{(0)}_{a,n}$ to $k_{a,n}$ that is simply just the corresponding $m_{a,n}$ without any Sharp filtering at all:

$$k^{(0)}_{a,n} \equiv m_{a,n}, \tag{117}$$

so that

$$k^{(0)}_{3,0} = \frac{3}{4}, \qquad k^{(0)}_{3,\pm1} = \frac{1}{8}, \tag{118}$$

and

$$k^{(0)}_{4,0} = \frac{2}{3}, \qquad k^{(0)}_{4,\pm1} = \frac{1}{6}. \tag{119}$$

The fundamental observation that leads to a recursive solution is that the first approximations (114) and (116) to $k_{a,n}$ are of exactly the same form as the zeroth approximations (118) and (119), with the only difference being that the nonzero $n \neq 0$ coefficients are at $n = \pm2$ rather than $n = \pm1$. We can therefore use exactly the same procedure to eliminate these coefficients at $n = \pm2$, leaving even smaller coefficients at $n = \pm4$. Note that this means that *we can jump directly to* $v = 3$ with this method, since *three* terms are being annulled, not two. This is the *Correction kernel* step that I described in (33) and (34), with coefficients, say, $c^{(3)}_{a,n}$, that are the obvious generalization of (112):

$$c^{(3)}_{a,0} = \frac{k^{(1)}_{a,0}}{k^{(1)}_{a,0} - 2k^{(1)}_{a,2}}, \qquad c^{(3)}_{a,\pm2} = \frac{-k^{(1)}_{a,2}}{k^{(1)}_{a,0} - 2k^{(1)}_{a,2}}. \tag{120}$$

For this $v = 3$ Correction of Sharp 3, we have

$$c^{(3)}_{3,0} = \frac{17}{18}, \qquad c^{(3)}_{3,\pm2} = \frac{1}{36}, \tag{121}$$

which is just what is stated in Eq. (34) for Correction 2021. Applying these $c^{(3)}_{3,n}$ to $s^{(1)}_{3,n}$ gives us $s^{(3)}_{3,n}$, the third approximation to Sharp 3, $s_{3,n}$:

$$s^{(3)}_{3,0} = +\frac{17}{12}, \qquad s^{(3)}_{3,\pm1} = -\frac{35}{144}, \qquad s^{(3)}_{3,\pm2} = +\frac{1}{24}, \qquad s^{(3)}_{3,\pm3} = -\frac{1}{144}, \tag{122}$$

which is what is given in (35). This now yields $k^{(3)}_{3,n}$, the third approximation to Magic Kernel Sharp 3:

$$k^{(3)}_{3,0} = +\frac{577}{576}, \qquad k^{(3)}_{3,\pm4} = -\frac{1}{1152}, \tag{123}$$

namely, Magic Kernel Sharp 2021, which is what is shown in Eq. (37), and, as noted there, satisfies the projection identity (29) to nine bits of accuracy. Note that *it still represents a very slight sharpening of the original image*; we have not returned to the blurring of the unsharpened Magic Kernel.

For the corresponding $v = 3$ Correction to Magic Kernel Sharp 4, we find

$$c^{(3)}_{4,0} = \frac{7}{8}, \qquad c^{(3)}_{4,\pm 2} = \frac{1}{16}, \tag{124}$$

so that

$$s^{(3)}_{4,0} = +\frac{7}{4}, \qquad s^{(3)}_{4,\pm 1} = -\frac{15}{32}, \qquad s^{(3)}_{4,\pm 2} = +\frac{1}{8}, \qquad s^{(3)}_{4,\pm 3} = -\frac{1}{32}, \tag{125}$$

and

$$k^{(3)}_{4,0} = +\frac{97}{96}, \qquad k^{(3)}_{4,\pm 4} = -\frac{1}{192}, \tag{126}$$

which satisfies the projection identity to around 6.6 bits of accuracy. Again, we still have a small amount of residual sharpening—and again it is unlikely to be of any practical relevance.

We can continue on with this recursive process; the next approximation produced is $v = 7$, the one after that $v = 15$, and so on. For Magic Kernel Sharp 3 and 4 it is difficult to foresee any situation that would require it. For $v = 7$, Magic Kernel Sharp 3 now satisfies the projection identity to around 19 bits of accuracy, and Magic Kernel Sharp 4 to around 14 bits of accuracy. Again, these corrections are simply flattening slightly more of the frequency response within the first Nyquist zone, and are probably only of academic interest.

For Magic Kernel Sharp 5 and 6, we cannot use the above recursive method, because $m_{a,n}$ now has nonzero coefficients out to $n = \pm 2$, rather than the ± 1 we had for Magic Kernel Sharp 3 and 4:

$$m_{5,0} = \frac{115}{192}, \qquad m_{5,\pm 1} = \frac{19}{96}, \qquad m_{5,\pm 2} = \frac{1}{384}, \tag{127}$$

and

$$m_{6,0} = \frac{11}{20}, \qquad m_{6,\pm 1} = \frac{13}{60}, \qquad m_{6,\pm 2} = \frac{1}{120}, \tag{128}$$

and consequently if we tried to just subtract off two shifted copies of itself, the "tails," with two coefficients rather than one, would interfere with the zero we are trying to create at the opposite position, rather than just contribute to $n = 0$ as it did above.

We therefore need a more direct approach. Let us return to the definition that the v-th approximation makes the v coefficients $k^{(v)}_{a,n}$ for $1 \le |n| \le v$ vanish, by suitably tweaking the v coefficients of $s^{(v)}_{a,n}$ for $1 \le |n| \le v$ (where normalization then determines $s^{(v)}_{a,0}$). Now, ironically, to kickstart this process it is actually more instructive to first try to solve for the *infinite* number of coefficients $s_{a,n}$ of the exact solution, at least formally, which lets us write down an (infinite) set of simultaneous equations to solve. Once we have figured out the pattern of those equations—which is slightly complicated by the symmetry and normalization of the Sharp kernel (*i.e.*, the v independent $s_{a,n}$ actually creates a kernel with $2v+1$ nonzero coefficients), we can then truncate the set of equations to just the first v of them, and then solve those to obtain the $s^{(v)}_{a,n}$.

So let's do that. If one writes out the convolution explicitly between the infinite set of $s_{a,n}$ and the five $m_{a,n}$, and then rearranges them into matrix form, then the patterns become apparent. Let us define

$$A \equiv \begin{pmatrix} m_0 & m_1 & m_2 & \cdot & \cdot & \cdot & \cdot & \cdot & \cdots \\ m_1 & m_0 & m_1 & m_2 & \cdot & \cdot & \cdot & \cdot & \cdots \\ m_2 & m_1 & m_0 & m_1 & m_2 & \cdot & \cdot & \cdot & \cdots \\ \cdot & m_2 & m_1 & m_0 & m_1 & m_2 & \cdot & \cdot & \cdots \\ \cdot & \cdot & m_2 & m_1 & m_0 & m_1 & m_2 & \cdot & \cdots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots \end{pmatrix}, \tag{129}$$

$$Z \equiv \begin{pmatrix} m_2 & \cdot & \cdot & \cdot & \cdots \\ \cdot & \cdot & \cdot & \cdot & \cdots \\ \cdot & \cdot & \cdot & \cdot & \cdots \\ \cdot & \cdot & \cdot & \cdot & \cdots \\ \vdots & \vdots & \vdots & \vdots & \ddots \end{pmatrix}, \tag{130}$$

$$B \equiv \begin{pmatrix} m_1 \\ m_2 \\ \cdot \\ \cdot \\ \cdot \\ \vdots \end{pmatrix}, \tag{131}$$

$$R \equiv \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & \cdots \end{pmatrix}, \tag{132}$$

and

$$S \equiv \begin{pmatrix} s_1 \\ s_2 \\ s_3 \\ s_4 \\ s_5 \\ s_6 \\ s_7 \\ \vdots \end{pmatrix}. \tag{133}$$

One then finds that the simultaneous equations can be written down as

$$(A + Z - 2BR)S + B = 0. \tag{134}$$

The advantage of splitting out the matrices like this is that each has a simple pattern that can be easily programmatically coded up, and then matrices with any finite number v of rows and/or

columns can be instantiated, and then solved, using a system like Sage. Again, I did the first couple of these by hand, and then got Sage to do the rest for me.

So let us start with the first approximation, $v = 1$. Here we are specifying that the Sharp kernel $s^{(1)}_{a,n}$ should have just one side-tap $s^{(1)}_{a,\pm1}$, and we want to choose the coefficient of that side-tap so that $k^{(1)}_{a,\pm1} = 0$. The solution to (134) truncated to just one row and/or column is

$$s^{(1)}_{a,1} = \frac{-m_{a,1}}{m_{a,0} - 2m_{a,1} + m_{a,2}}. \tag{135}$$

We can see that this is just the same result as given in (112), except that now we have an extra contribution from $m_{a,2}$, which of course vanished for $a = 3$ and 4. For Sharp 5, we have

$$s^{(1)}_{5,1} = -\frac{76}{79} \approx -0.962, \tag{136}$$

yielding a first approximation to the Magic Kernel Sharp 5 kernel,

$$k^{(1)}_{5,0} = +\frac{20789}{15168} \approx +1.371, \qquad k^{(1)}_{5,\pm2} = -\frac{5545}{30336} \approx -0.183, \qquad k^{(1)}_{6,\pm2} = -\frac{19}{7584} \approx -0.003. \tag{137}$$

For Sharp 6, we have

$$s^{(1)}_{6,1} = -\frac{26}{15} \approx -1.733, \tag{138}$$

yielding a first approximation to the Magic Kernel Sharp 6 kernel,

$$k^{(1)}_{6,0} = +\frac{307}{180} \approx +1.706, \qquad k^{(1)}_{6,\pm2} = -\frac{203}{600} \approx -0.338, \qquad k^{(1)}_{6,\pm2} = -\frac{13}{900} \approx -0.014. \tag{139}$$

From this point I will list all coefficients only as their decimal approximations to three decimal places, since we are most interested in seeing when the approximation reaches eight bits of accuracy, namely, around 0.004. The exact fractions *are* of use in any practical implementation, and indeed I have used them in my reference ANSI C implementation [5]; I am more than happy to provide the coefficients directly (or, even easier, the Sage code) on request.

We can see that the amount of sharpening required for this first approximation continues to increase as we go to higher a: the $s^{(1)}_{a,\pm1}$ of -0.25 for $a = 3$ and -0.5 for $a = 4$ has increased to -0.962 for $a = 5$ and -1.730 for $a = 6$. This is just the aforementioned consequence of the Sharp kernel having to try to undo blurring effects that are getting slightly wider for each a: the width only goes up like $\sqrt{a}$, but still, by the time we get to $a = 5$ and $a = 6$, it is nonzero out to $n = \pm2$, and so we would not expect a single Sharp tap at $n = \pm1$ to be able to fully compensate. And this is indeed reflected in the increasing tails, the $k^{(1)}_{a,|n|\geq2}$ of $k^{(1)}_{a,n}$. Now, it turns out (for reasons I have yet to fathom) that the coefficients in these tails are always all of the same sign, so the sum of their absolute values is the same as the absolute value of their sum, and hence can be characterized by the deviation of $k^{(1)}_{a,0}$ from unity (which is just the negative of this sum, without taking absolute values, because of normalization). As we saw back in Sec. 3, $k^{(1)}_{3,0}$ is 6.2% high (*i.e.*, a residual sharpening by something like that amount). We now find $k^{(1)}_{4,0}$ to be 16.7% high, $k^{(1)}_{5,0}$ to be 37% high, and $k^{(1)}_{6,0}$ to be 71% high. What was a small amount of sharpening for Magic Kernel Sharp 2013, perfectly reasonable for practical use (indeed, closing in on a trillion

a	v	k_0	s_1	s_2	s_3	s_4	s_5	s_6	s_7
1	0	1.000	–	–	–	–	–	–	–
2	0	1.000	–	–	–	–	–	–	–
3	1	1.062	−0.250	–	–	–	–	–	–
	3	1.002	−0.243	+0.042	−0.007	–	–	–	–
	7	1.000	−0.243	+0.042	−0.007	+0.001	−0.000	+0.000	−0.000
4	1	1.167	−0.500	–	–	–	–	–	–
	3	1.010	−0.469	+0.125	−0.031	–	–	–	–
	7	1.000	−0.464	+0.124	−0.033	+0.009	−0.002	+0.001	−0.000
5	1	1.371	−0.962	–	–	–	–	–	–
	2	0.910	−0.736	+0.235	–	–	–	–	–
	3	1.037	−0.854	+0.304	−0.097	–	–	–	–
	4	0.987	−0.815	+0.294	−0.105	+0.033	–	–	–
	5	1.005	−0.830	+0.300	−0.108	+0.039	−0.012	–	–
6	1	1.706	−1.733	–	–	–	–	–	–
	2	0.845	−1.067	+0.385	–	–	–	–	–
	3	1.086	−1.424	+0.599	−0.214	–	–	–	–
	4	0.967	−1.276	+0.550	−0.230	+0.082	–	–	–
	5	1.015	−1.341	+0.581	−0.249	+0.104	−0.037	–	–
	6	0.994	−1.313	+0.570	−0.245	+0.105	−0.044	+0.016	–
	7	1.003	−1.325	+0.575	−0.248	+0.107	−0.046	+0.019	−0.007

Table 5: Central Magic Kernel Sharp coefficient $k_{a,0}^{(v)}$ resulting from applying the Sharp coefficients $s_{a,n}^{(v)}$ of the v-th approximation to Magic Kernel Sharp a. All coefficients are listed to three decimal places. Sufficient approximations v are included to achieve eight bits of accuracy. Sage code for generating the exact values is available from me on request.

examples), is less tenable for higher a (unless such "Magic Kernel Sharp+" is indeed desired, at the corresponding level).

Let us therefore push on to the second and higher approximations to the Sharp kernel, namely, $v \geq 2$. It is clearest to list all of the relevant coefficients in tabular form, which I do in Table 5. Laying them out like this, and reading down the columns, reminds us that there *is* an exact (infinite) set of coefficients $s_{a,n}$ for each exact Sharp a kernel. While it is possible to consider the successive approximations that we have derived above as self-contained kernels, in reality they are all just approximations to these six exact kernels—and it actually makes more sense to look at Table 5 holistically, to determine which coefficients to use for any particular application, rather than just to stick to any one particular approximation that went into generating the table. Indeed, when I built Magic Kernel Sharp for Facebook in 2013, I didn't actually use the coefficient -0.25 of Sharp 2013, but rather something that was very slightly smaller—because that is what our experiments on a large corpus of test images told us was the value that provided the most accurate results. Of course, I had no idea at that time that the *exact* value, according to the methods of this paper, was -0.243—indeed, I didn't even know that until a few days ago. The experimentally-determined value I used was somewhere between -0.25 and -0.243—which makes sense, overall, given that there was no s_2 in my simple three-tap sharpening filter, and hence the optimal s_1 needs to try to "carry the load" of the missing s_2.

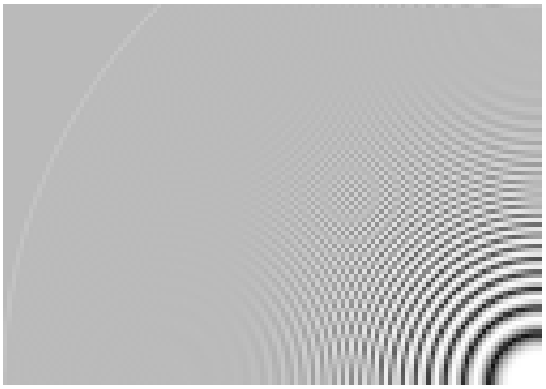
Examining Table 5 holistically, then, let us start with $a = 3$. As noted above, an extra application of the Correction kernel for $v = 7$ yields an improvement over $v = 3$ that is negligible if we only need eight bits of accuracy, and nor does it change and of the three s_n of $v = 3$. It is therefore sufficient to stick to $v = 3$, namely, Magic Kernel Sharp 2021, as described in Sec. 3.

Moving on to $a = 4$, it is clear that $v = 3$ does not quite give us our desired eight bits of accuracy, and so use of the extra recursive kernel to obtain $v = 7$ is justified. In other words, Magic Kernel Sharp 4 should recusrively use *two* Correction kernels on top of the original three-tap Sharp kernel, if full eight-bit accuracy is desired. The sum of the supports of the three components of the third approximation to Sharp 4 is just $3 + 3 + 3 = 9$. Of course, per the discussion above, if maximal flattening of the spectral response within the first Nyquist zone is not a priority, then $v = 2$ may be sufficient.

For $a = 5$, we have moved away from the recursive method, and are instead looking at a straight-out application of a multi-tap Sharp kernel. From the table, $v = 5$, resulting in a Sharp kernel with $2v + 1 = 11$ taps, gets us to eight-bit accuracy. If we have such a high-precision use case that it is worth moving up from $a = 4$ to $a = 5$, then it may be worth having a Sharp support kernel with such support; this decision may also depend on the resize factor r (large values of which make any Sharp support insignificantly costly, whereas r close to unity makes the computational cost more painful). Again, if flat spectral response is not a priority—say, if antialiasing fidelity is the reason for moving to Magic Kernel 5—then $v = 4$ or even $v = 3$ should be quite acceptable.

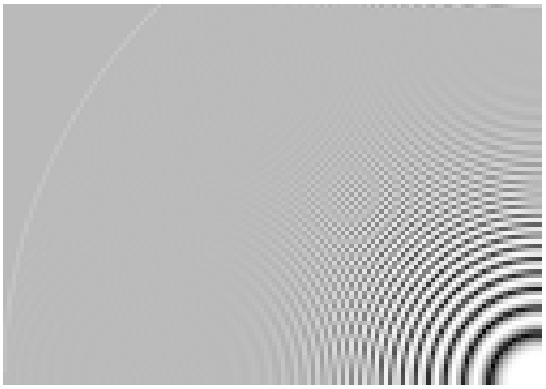
Finally, for $a = 6$, $v = 6$ provides 7.3 bits of accuracy, and $v = 7$ provides 8.5 bits. Whether this fidelity of flattening is required is again dependent on the application, and again anything down to $v = 3$ likely provides acceptable results for many purposes.

Of course, the proof of the pudding for Magic Kernel Sharp 4, 5, and 6 is how they perform in practice. The only image produced by Magic Kernel Sharp 3 that had visually discernible aliasing artifacts was Fig. 31(a), the downsizing of the maximal stress-test zone plate of Fig. 13. Indeed, all three resizing kernels in Fig. 31 showed artifacts; Lanczos-2 was inferior to Magic



(a) Downsized with Lanczos-2

(b) Downsized with Lanczos-3



(c) Downsized with Magic Kernel Sharp 3

(d) Downsized with Magic Kernel Sharp 4

(e) Downsized with Magic Kernel Sharp 5

(f) Downsized with Magic Kernel Sharp 6

Figure 42: Comparison of using Lanczos-2 and -3, and Magic Kernel Sharp 3, 4, 5, and 6 (with the full Sharp kernels provided in Table 5), as downsizing kernels on the test image of Fig. 15. The lower-left quadrant is the best illustration of aliasing artifacts, as the underlying image has too high a frequency to "show through" in this area, and hence does not provide a visual distraction with its own moiré effects, as is evident in the right halves of all six images. (The artifacts arising from using extension boundary conditions are again evident along the right-hand parts of the top edges of all images.)

Kernel Sharp 3, while Lanczos-3 was superior.

Applying, then, Magic Kernel Sharp 4, 5, and 6 to Fig. 13, and adding the results to our sequence of Fig. 31, we obtain the results shown in Fig. 42. As noted in the caption there, the lower-left quadrant provides the best proving ground for being able to visually discern aliasing artifacts. We see that Magic Kernel Sharp 4 matches Lanczos-3, and that Magic Kernel Sharp 5 and 6 exceed it, leaving no residual fringes in this area to within the eight bits of dynamic range of these images (even when examined pixel by pixel). Of course, this is what we expected from the analyses above, and particularly Table 4, but it always good to "see with our own eyes" what the formulas and spectra suggest should be the case.

Acknowledgments

This paper is dedicated to the memory of Mitzi [10], who suddenly, unexpectedly, and devastatingly left us shortly before I did this work.

Cropped parts of frames from [7] are used in Fig. 36 under Fair Use.

This paper describes personal hobby work that I have carried out since 2006 [1]. Any opinions expressed herein are mine alone.

References

[1] John P. Costella, "What is 'the magic kernel'?" http://johncostella.com/magic/#intro.

[2] John P. Costella, "2015: Mathematical properties of the Magic Kernel Sharp 2013 kernel," http://johncostella.com/magic/#fourier2015.

[3] John P. Costella, "2011: The Magic Kernel is born," http://johncostella.com/magic/#mk.

[4] John P. Costella, "2013: Magic Kernel Sharp," http://johncostella.com/magic/#mks.

[5] John P. Costella, "2021: Reference ANSI C implementation of Magic Kernel Sharp," http://johncostella.com/magic/#code.

[6] Wolfram, http://mathworld.wolfram.com/TrigonometricPowerFormulas.html.

[7] Netflix, *Ma Rainey's Black Bottom* (2020).

[8] Google, http://google.com/search?q=how+many+photos+are+there+on+facebook.

[9] Google, http://google.com/search?q=how+many+photos+are+there+on+instagram.

[10] Mitzi, http://facebook.com/mitzi.the.dog.