EMPossible

Computational Science:
Computational Methods in Engineering

Calculating Discrete Convolutions

Property of the Fourier Transform and the DFT

If a convolution is Fourier transformed, it becomes a point-by-point multiplication in the Fourier domain.

$$\mathrm{FT}\left\{f(t)*g(t)\right\}=F(\omega)G(\omega)$$

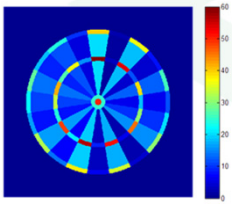
There is a similar property for the DFT

$$\mathrm{DFT}\left\{f(n)*g(n)\right\}=F(k)G(k)$$

In fact, this is a "circular convolution." To ensure this looks like an ordinary convolution, pad the function with enough zeros so that it is equal to or greater than the final convolution length output.

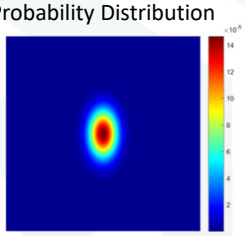
Convolutions for Statistics

Point Distribution



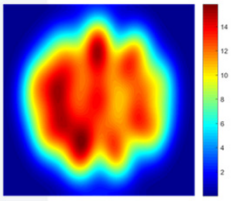
*

Probability Distribution



=

Score Expectation



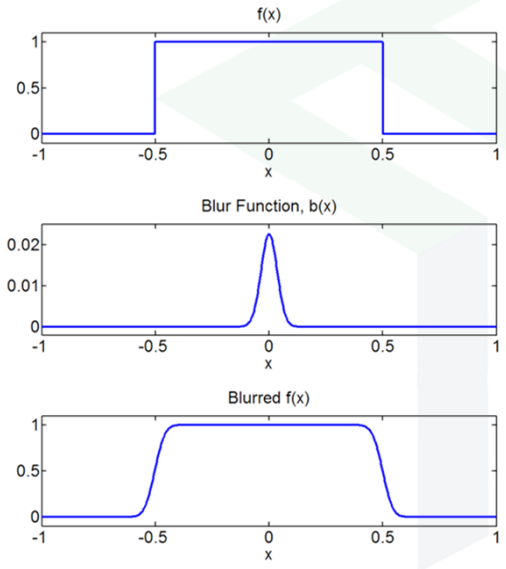
```
% ACCURACY
accx = 3.0 * centimeters;
accy = 5.0 * centimeters;

% THROWING PDF
PDF = (0.5*X/accx).^2 ...
    + (0.5*Y/accy).^2;
PDF = exp(-PDF);
PDF = PDF / sum(PDF(:));
```

```
% PERFORM CONVOLUTION USING FFT
S = fft2(ifftshift(PDF)).*fft2(DB);
S = real(ifft2(S));

% SHOW SCORE EXPECTATION
imagesc(xa,ya,S');
axis equal tight off;
colorbar;
```

EMPossible

Slide 3

Convolutions for Blurring and Smoothing Data

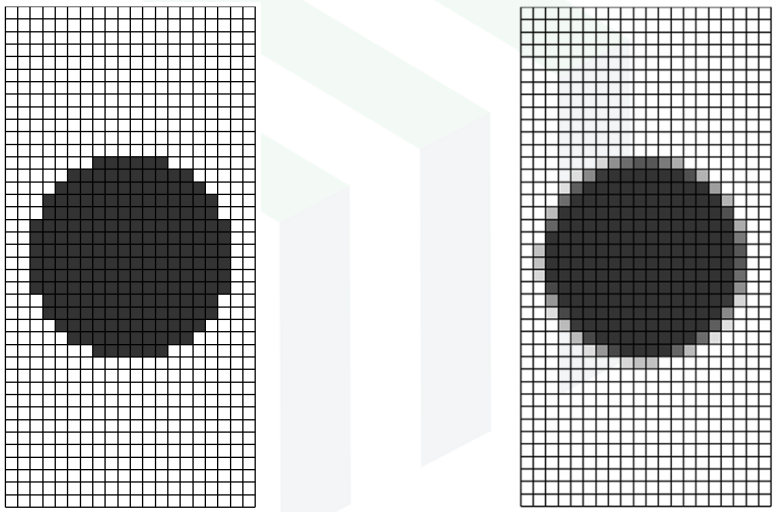



```
% FUNCTION
N  = 1000;
x  = linspace(-1,1,N);
f  = zeros(1,N);
n1 = round(0.25*N);
n2 = round(0.75*N);
f(n1:n2) = 1;

% BLUR FUNCTION
r = 0.05;
b = exp(-(x/r).^2);
b = b / sum(b(:));

% CONVOLVE
f2 = fft(f).*fft(ifftshift(b));
f2 = real(ifft(f2));
```

EMPossible

Slide 4

Blurring in Two Dimensions



EMPossible

Slide 5