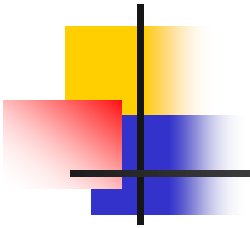
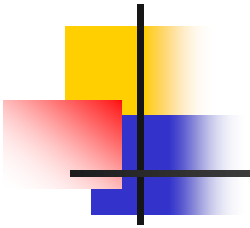


Image Deblurring

- Introduction
- Inverse filtering
  - Suffer from noise amplification
- Wiener filtering
  - Tradeoff between image recovery and noise suppression
- Iterative deblurring*
  - Landweber algorithm



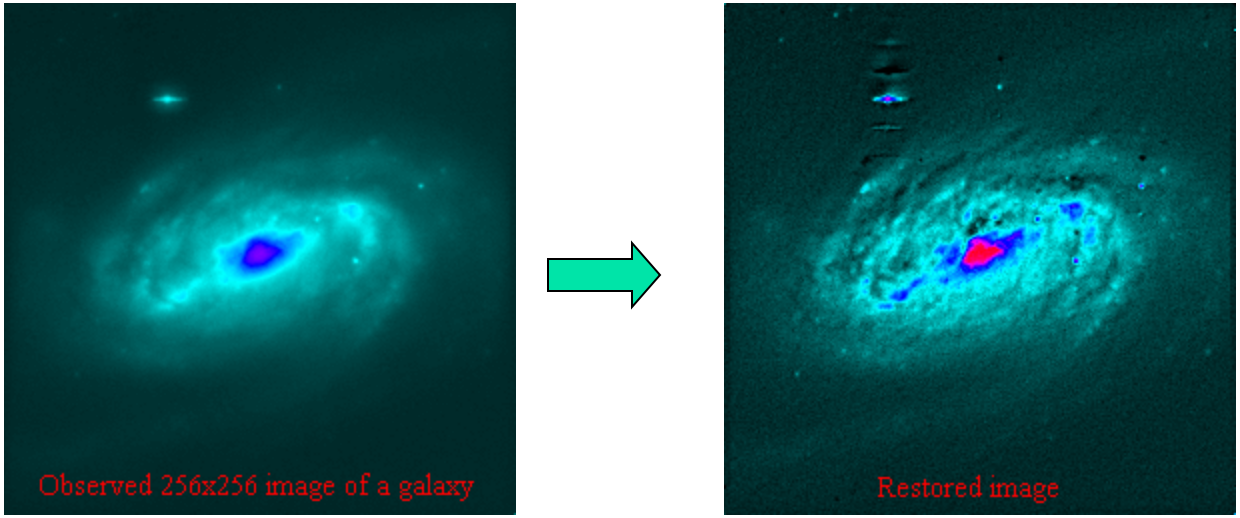
Introduction

- Where does blur come from?
  - Optical blur: camera is out-of-focus
  - Motion blur: camera or object is moving
- Why do we need deblurring?
  - Visually annoying
  - Wrong target for compression
  - Bad for analysis
  - Numerous applications

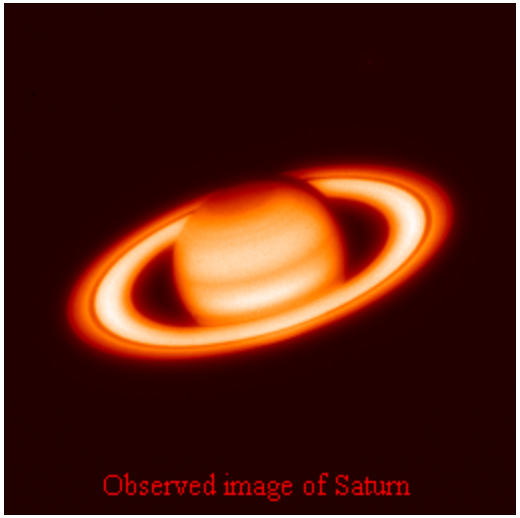
Application (I): Astronomical Imaging

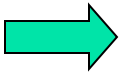
- The Story of Hubble Space Telescope (HST)
  - HST Cost at Launch (1990): $1.5 billion
  - Main mirror imperfections due to human errors
  - Got repaired in 1993

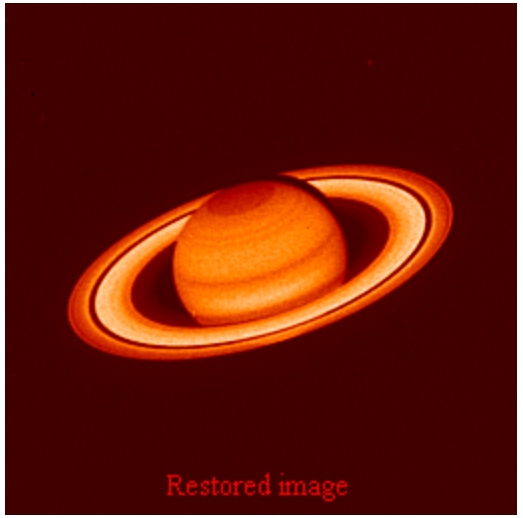
Restoration of HST Images



Another Example







The Real (Optical) Solution

Before the repair

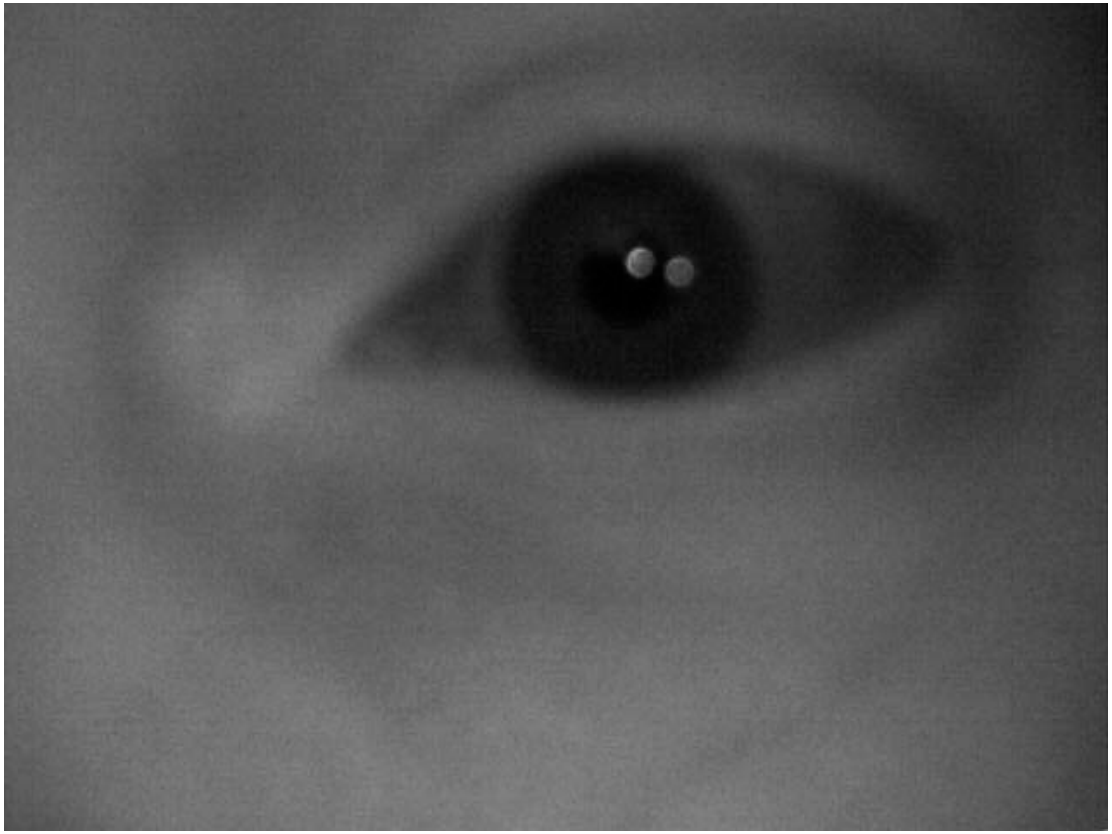
After the repair

Application (II): Law Enforcement

Motion-blurred license plate image

Restoration Example

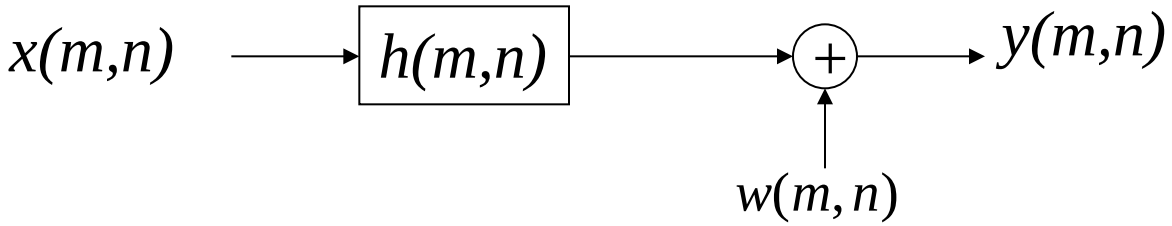
Application into Biometrics



out-of-focus iris image

Modeling Blurring Process

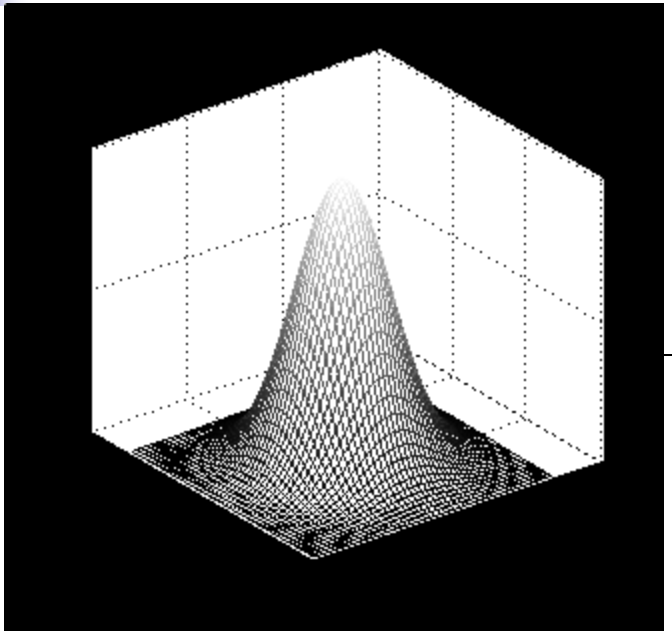
- Linear degradation model



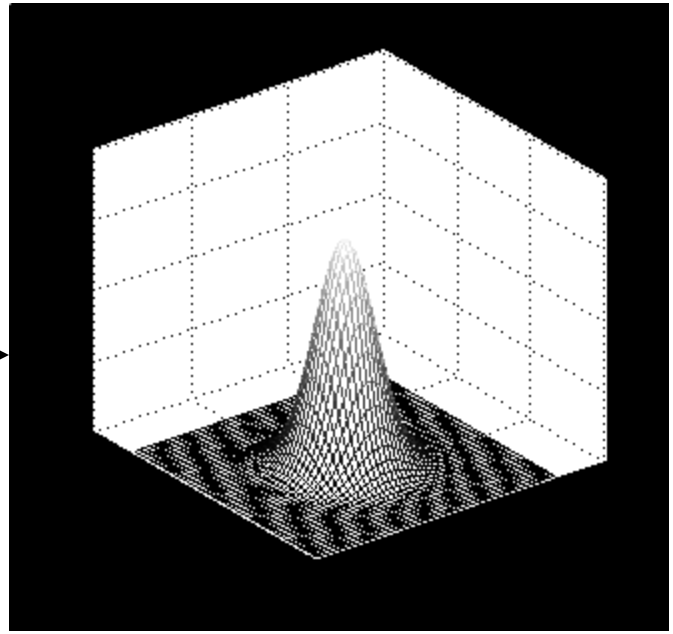
$h(m,n)$ blurring filter

$w(m,n) \sim N(0,\sigma_w^2)$ additive white Gaussian noise

Blurring Filter Example



FT

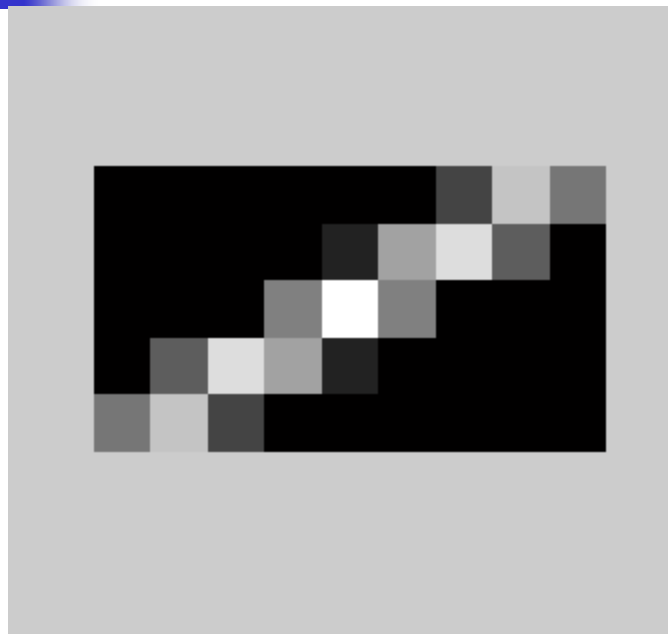


$$h(m,n) = \exp(-\frac{m^2+n^2}{2\sigma^2})$$

$$H(w_1,w_2) = \exp(-\frac{w_1^2+w_2^2}{2\sigma^2})$$

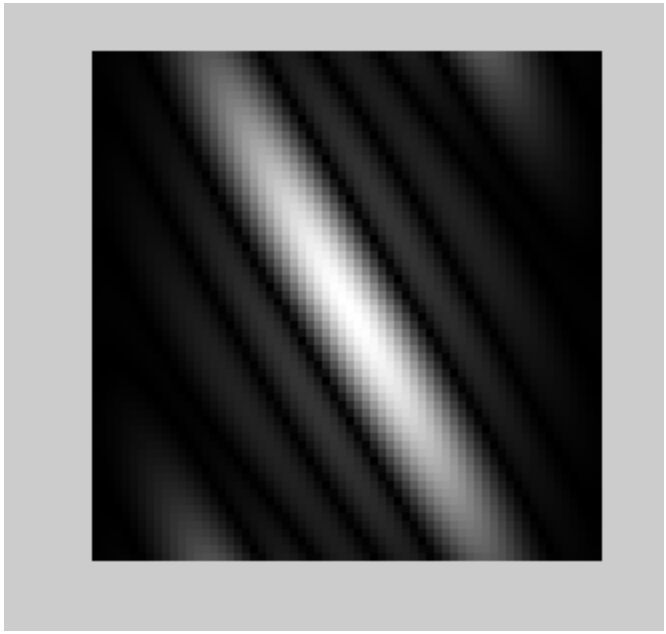
Gaussian filter can be used to approximate out-of-focus blur

Blurring Filter Example (Con't)



$h(m,n)$

FT



$H(w_1, w_2)$

MATLAB code: h=FSPECIAL('motion',9,30);

Motion blurring can be approximated by 1D low-pass filter along the moving direction

Image Example

x(m,n)

BSNR=40dB

BSNR=10dB

h(m,n): 1D horizontal motion blurring [1 1 1 1 1 1 1]/7

Blind vs. Nonblind Deblurring

- Blind deblurring (deconvolution): blurring kernel h(m,n) is unknown
- Nonblind deconvolution:

  blurring kernel h(m,n) is known

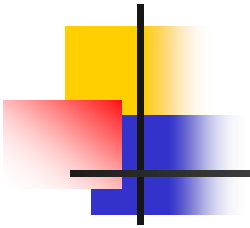
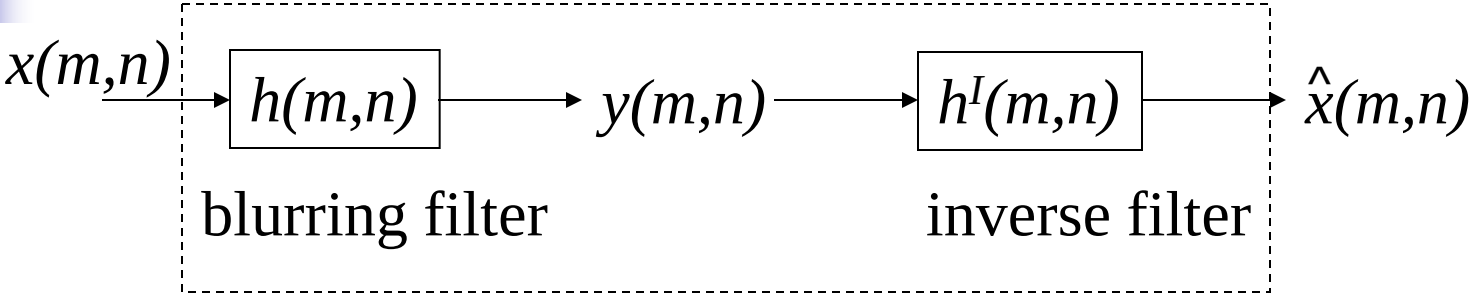


Image Deblurring

- Introduction
- Inverse filtering
  - Suffer from noise amplification
- Wiener filtering
  - Tradeoff between image recovery and noise suppression
- Iterative deblurring*
  - Landweber algorithm

Inverse Filter



To compensate the blurring, we require

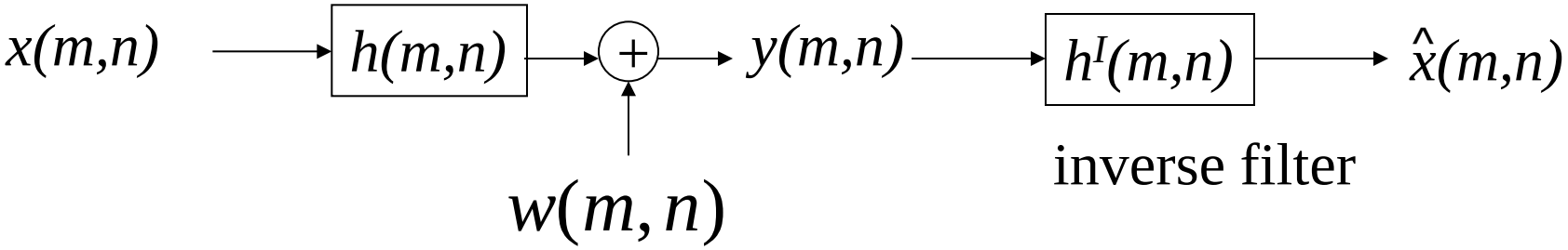
$$h_{combi}(m,n) = h(m,n) \otimes h^{I}(m,n) =$$

$$\sum_{k=-\infty}^{\infty} \sum_{l=-\infty}^{\infty} h^{I}(m-k, n-l) h(k,l) = \delta(m,n), \forall (m,n)$$

$$\Downarrow$$

$$\boxed{H^{I}(w_1, w_2) = \frac{1}{H(w_1, w_2)}}$$

Inverse Filtering (Con't)



Spatial:

$$\hat{x}(m,n) = y(m,n) \otimes h^I(m,n) = (x(m,n) \otimes h(m,n) + w(m,n)) \otimes h^I(m,n)$$

Frequency:

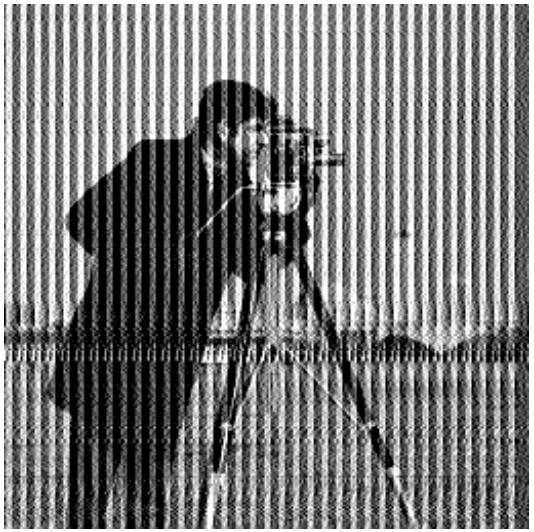
$$\hat{X}(w_1,w_2) = Y(w_1,w_2)H^I(w_1,w_2) = \frac{X(w_1,w_2)H(w_1,w_2)+W(w_1,w_2)}{H(w_1,w_2)}$$

$$= X(w_1,w_2) + \frac{W(w_1,w_2)}{H(w_1,w_2)}$$

amplified noise

Image Example

motion blurred image
at BSNR of 40dB



deblurred image after
inverse filtering

Q: Why does the amplified noise look so bad?

A: zeros in $H(w_1,w_2)$ correspond to poles in $H^I(w_1,w_2)$

Pseudo-inverse Filter

Basic idea:

To handle zeros in $H(w_1,w_2)$, we treat them separately when performing the inverse filtering

$$H^{-}(w_1,w_2)=\begin{cases}\dfrac{1}{H(w_1,w_2)} & |H(w_1,w_2)|>\delta \\ 0 & |H(w_1,w_2)|\le\delta\end{cases}$$

Image Example

motion blurred image
at BSNR of 40dB

deblurred image after
Pseudo-inverse filtering
(δ=0.1)

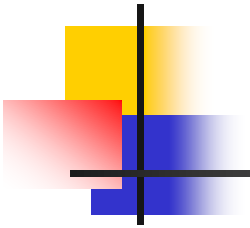


Image Deblurring

- Introduction
- Inverse filtering
  - Suffer from noise amplification
- Wiener filtering
  - Tradeoff between image recovery and noise suppression
- Iterative deblurring*
  - Landweber algorithm

Norbert Wiener (1894-1964)

Wiener Filtering

Also called Minimum Mean Square Error (MMSE) or Least-Square (LS) filtering

$$H_{mmse}(w_1, w_2) = \frac{H^*(w_1, w_2)}{|H(w_1, w_2)|^2 + K}$$

constant

Example choice of K:

$$K = \frac{\sigma_w^2}{\sigma_z^2}$$

σ_w^2 — noise energy

σ_z^2 — signal energy

K=0 $\rightarrow$ inverse filtering

Image Example

motion blurred image
at BSNR of 40dB

deblurred image after
wiener filtering
(K=0.01)

Image Example (Con't)

K=0.1

K=0.01

K=0.001

Constrained Least Square Filtering

Similar to Wiener but a different way of balancing the tradeoff between

$$H_{mmse}(w_1, w_2) = \frac{H^*(w_1, w_2)}{|H(w_1, w_2)|^2 + \gamma |C(w_1, w_2)|^2}$$

Example choice of C:

$$C(m,n) = \begin{bmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{bmatrix}$$

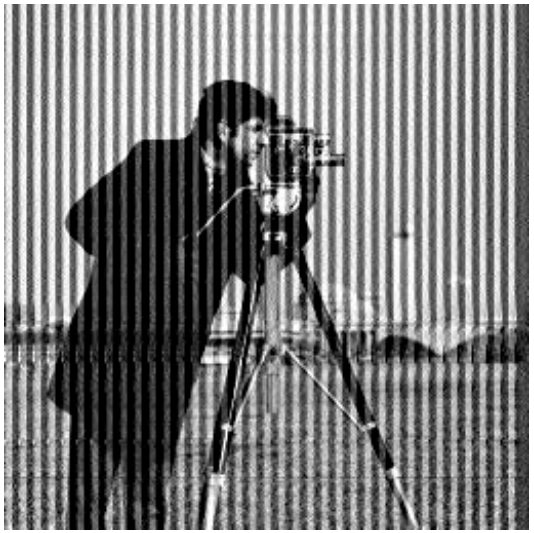
Laplacian operator

γ=0 $\rightarrow$ inverse filtering

Image Example

γ=0.1

γ =0.01



γ =0.001

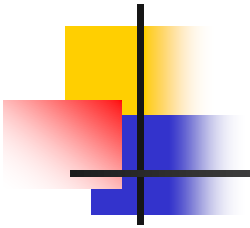


Image Deblurring

- Introduction
- Inverse filtering
  - Suffer from noise amplification
- Wiener filtering
  - Tradeoff between image recovery and noise suppression
- Iterative deblurring*
  - Landweber algorithm

Method of Successive Substitution

- A powerful technique for finding the roots of any function f(x)
- Basic idea
  - Rewrite f(x)=0 into an equivalent equation x=g(x) (x is called fixed point of g(x))
  - Successive substitution: $x_{i+1}=g(x_i)$
  - Under certain condition, the iteration will converge to the desired solution

Numerical Example

$$f(x) = x^2 - 3x + 2$$

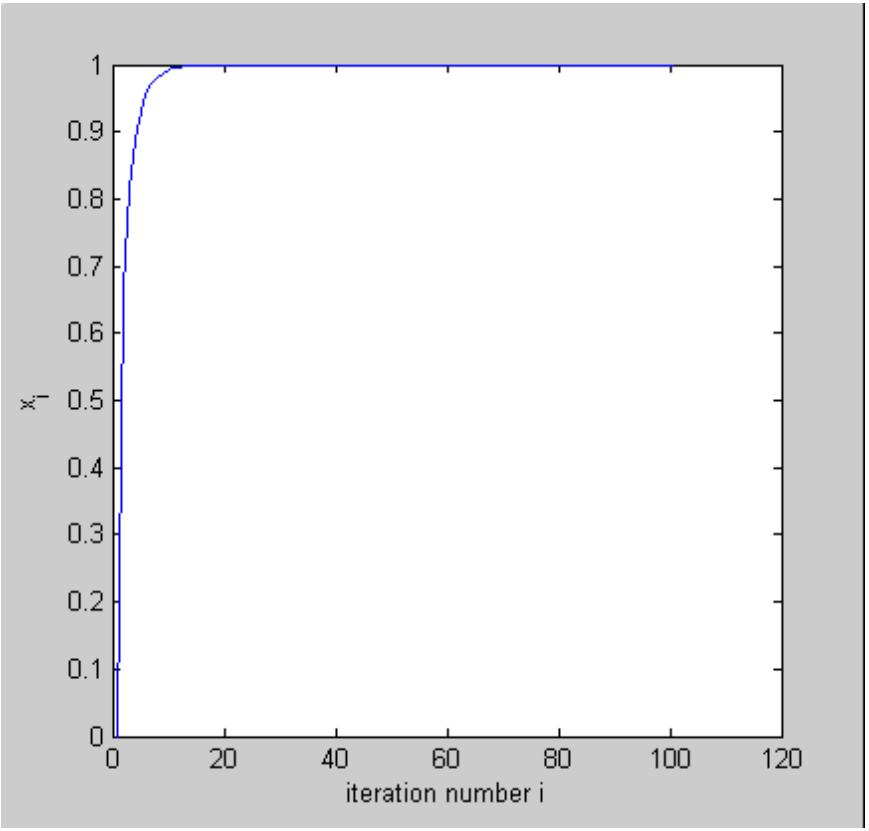
$$\downarrow$$

Two roots: $x_1 = 1, x_2 = 2$

$$f(x) = x^2 - 3x + 2 = 0 \Rightarrow x = \frac{x^2 + 2}{3} = g(x)$$

successive substitution: $x_{i+1} = \frac{x_i^2 + 2}{3}$

Numerical Example (Con't)



Note that iteration quickly converges to x=1

Landweber Iteration

Linear blurring $Y(w_1, w_2) = H(w_1, w_2)X(w_1, w_2)$

$\downarrow$

We want to find the root of $f(X) = Y - HX$

$\downarrow$

$$f(X) = 0 \Rightarrow X = X + \beta f(X) = X + \beta(Y - HX) = g(X)$$

β relaxation parameter – controls convergence property

Successive substitution:

$$X_0 = 0$$
$$X_{n+1} = X_n + \beta(Y - HX_n)$$

Advantages of Landweber Iteration

- No inverse operation (e.g., division) is involved
- We can stop the iteration in the middle way to avoid noise amplification
- It facilitates the incorporation of a priori knowledge about the signal (X) into solution algorithm