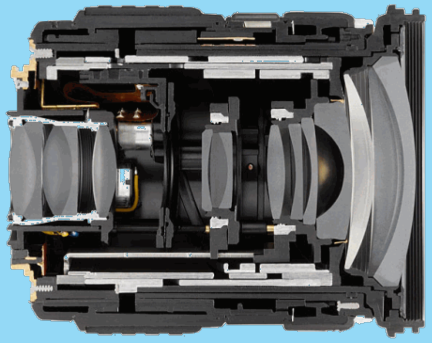
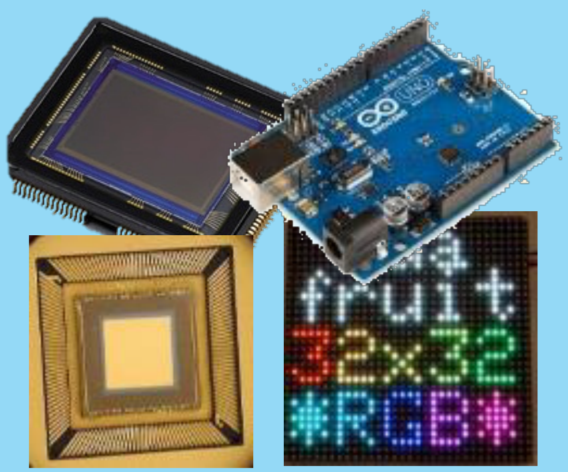


Optics

+



Sensors
&
devices

+

Signal
processing
&
algorithms

COMPUTATIONAL METHODS FOR IMAGING (AND VISION)

LECTURE 20: ITERATIVE METHODS

PROF. JOHN MURRAY-BRUCE

WHERE ARE WE

WE ARE HERE!

10	15-Mar-21	Forward Models and Inverse Problems	Linear Inversion - Inverse problems - Deconvolution and Denoising	IIP 4, Appendix E	**HW 4**	
	17-Mar-21		Intro to Regularized Inversion I - Tikhonov	IIP 5, Appendix E		
11	22-Mar-21	Regularization	Intro to Regularized Inversion II - Iterative methods - Steepest descent	IIP 6		
	24-Mar-21		Statistical methods I - ML estimation - Bayesian estimation	IIP 7.1 - 7.5		
12	29-Mar-21	Forward models and Inverse Problems II	LSV imaging systems: Forward problem - SVD - Inversion	IIP 8.1, 9, 10		
	31-Mar-21	Non-linear Regularization	Beyond L_2-regularization - Sparsity (l_0- and l_1-priors) - TV prior	SMIV 1.1 - 1.5 Papers & Handout		**HW 4**
13	5-Apr-21		Algorithms overview - ISTA/FISTA - ADMM	Papers & Handout		
	7-Apr-21	Introductory Optics	Geometrical/Ray Optics - Rays & pinhole cameras - Lenless imaging and Coded apertures	IIP 8.2, 8.3, 9.5 Papers & Handout		
14	12-Apr-21 14-Apr-21	**Spring Break (no classes)**				
15	19-Apr-21	Applications of Comp. Imaging	Looking around corners (NLOS imaging)	Papers & Handout		
	21-Apr-21		Compressive Imaging and Imaging from few photons	Papers & Handout		
16	26-Apr-21 28-Apr-21	**Group Presentations (Teams)**				
17	3-May-21	***no class**				
	5-May-21	**Final Exam: 12:30 PM - 2:30 PM**				

OUTLINE

- Fixed point equations
  - Fixed-point iterations (Landweber method)
- Iterative regularization methods
  - Gradient methods (steepest descent and conjugate gradient)

LEARNING GOALS

- Recognize regularization as a cure for ill-conditioning
- Understand gradient methods and reasons why they are efficient
- Code for steepest descent and conjugate gradient methods

READING

- IIP Chapter 6

RECAP

MOTION BLUR

PSF

$$\mathbf{g} = \mathbf{A}\mathbf{f} + \mathbf{n} \qquad \mathrm{cond}(\mathbf{A}) = 815$$

f

g

$A^{-1}g$

Noise dominates at high frequencies

LEAST SQUARES PROBLEM
MOORE-PENROSE REVISITED

- The least square problem:

$$\boxed{\min_{\mathbf{f}} \|\mathbf{g} - \mathbf{A}\mathbf{f}\|_2^2}$$

  - Solution: the Moore-Penrose pseudo-inverse

$$\mathbf{A}^{\dagger} = (\mathbf{A}^H\mathbf{A})^{-1}\mathbf{A}^H$$

- However, the Moore-Penrose pseudo-inverse still gives unsatisfactory imaging results!

TIKHONOV REGULARIZATION

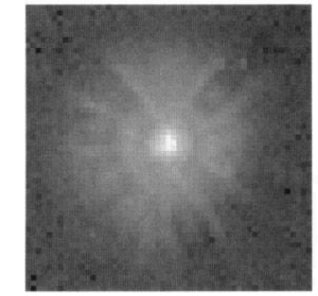
- **Tikhonov regularized inversion:** solves the minimization problem

$$\min_{\mathbf{f}} \overbrace{\boxed{\|\mathbf{g}-\mathbf{A}\mathbf{f}\|_2^2}}^{\text{DATA FIDELITY TERM}} + \overbrace{\boxed{\underbrace{\mu}_{\text{REGULARIZATION PARAMETER}} \|\mathbf{f}\|_2^2}}^{\text{REGULARIZATION TERM}}$$

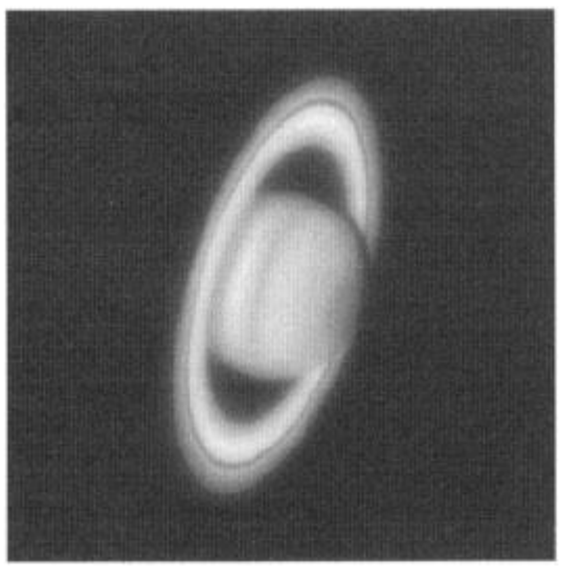
- **Solution:** $\mathbf{f}_\mu = (\mathbf{A}^H\mathbf{A} + \mu\mathbf{I})^{-1}\mathbf{A}^H\mathbf{g}$

- Notice that this solution is a function of μ:
  - μ controls the balance between the data fidelity and the regularization terms
  - Large μ gives highly regularized solution compared to small μ
  - When $\mu = 0$ get the usual pseudo-inverse

EXAMPLE

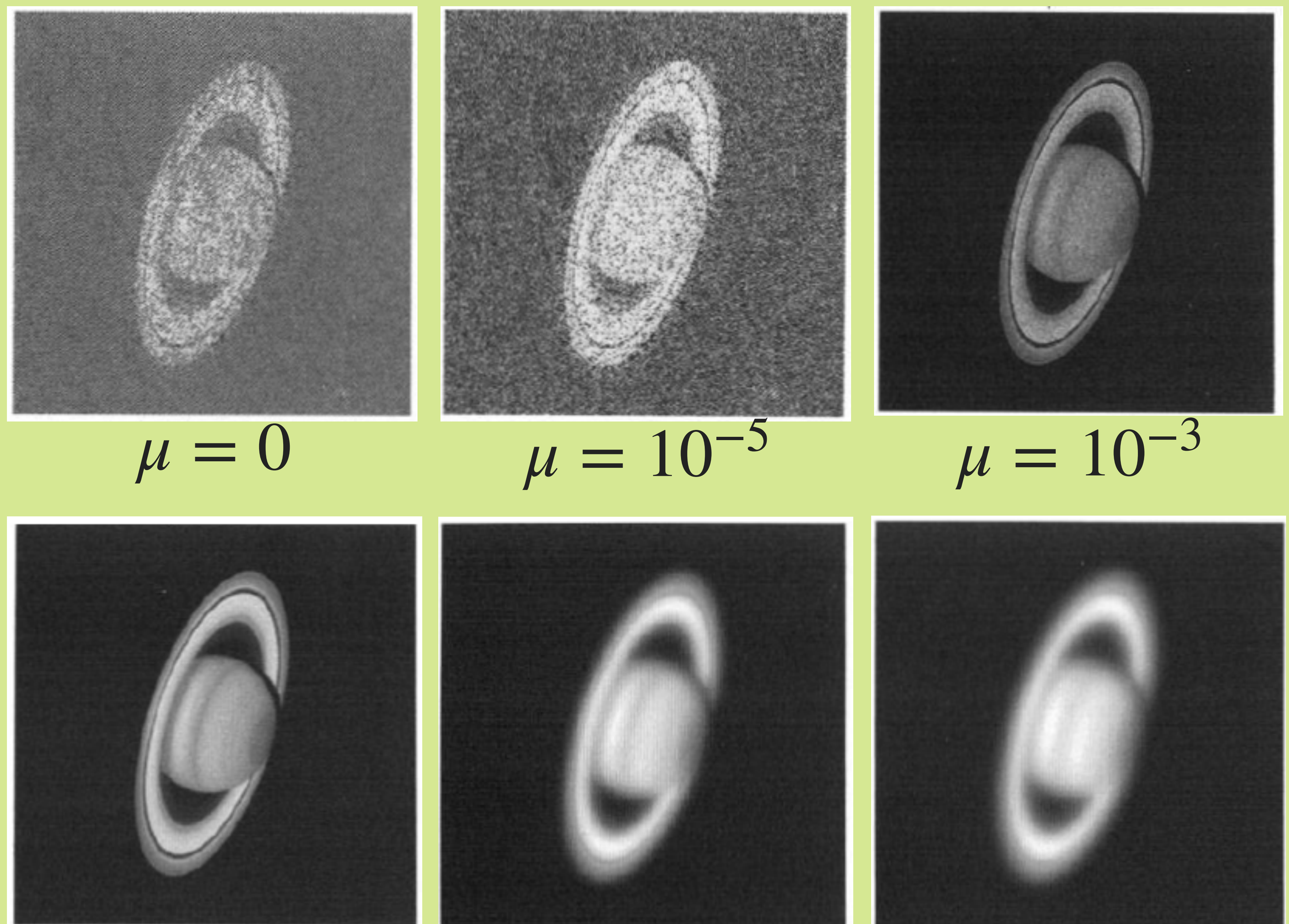


PSF



Ground truth

Tikhonov regularized inverses



ITERATIVE REGULARIZATION METHODS FOR INVERSION

SOLVING LINEAR EQUATIONS

- Several methods have been introduced to solve linear system of equations:

$$Ax = y$$

  - **Closed form** methods, which explicitly compute the (psuedo-)inverse of the matrix $\mathbf{A}$. Can be prohibitive for large scale linear systems
  - **Iterative** methods

- Key feature of **iterative methods** is that the number of iterations plays the role of the regularization parameter:

  - The iterates often approach the object **(solution)** and then move away.

  - Thus, there is an optimal number of iterations k_{opt}.

FIXED-POINT METHODS

FIXED-POINT ITERATIONS

AN OVERVIEW

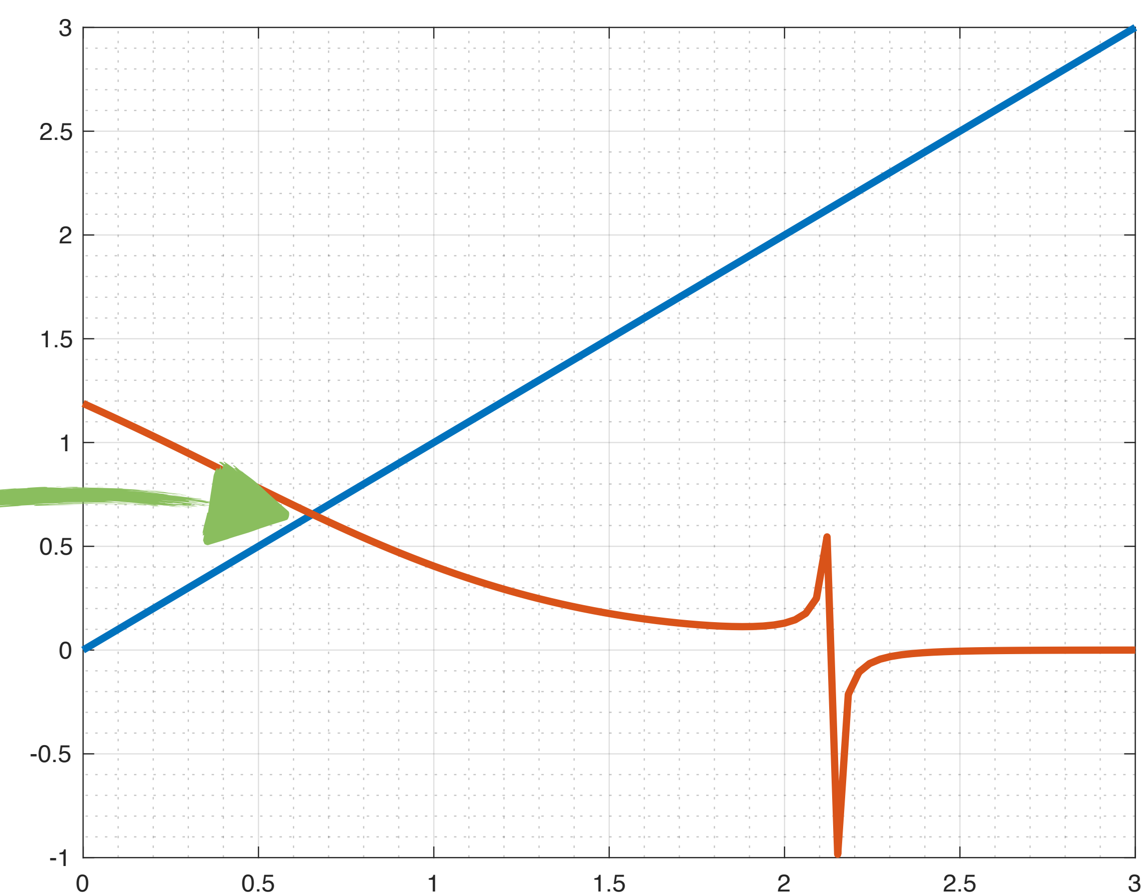
We want to find the solution to the equation $\dfrac{e^{-x^2}}{x\sin(x+1)} - 1 = 0$

Rearranging gives $x = \dfrac{e^{-x^2}}{\sin(x+1)}$

$$f(x) = x$$

$$g(x) = \frac{e^{-x^2}}{\sin(x+1)}$$

Fixed point of $g(x)$



FIXED-POINT ITERATIONS
A REVIEW

Natural approach to solve $x = \dfrac{e^{-x^2}}{\sin(x+1)}$ for x (under some conditions) is to use the **fixed-point method.**

$$f(x) = x$$

$$g(x) = \frac{e^{-x^2}}{\sin(x+1)}$$

Algorithm: Fixed-point iteration

For $k = 1 \ldots K$ do:

$$x_{k+1} = g(x_k)$$

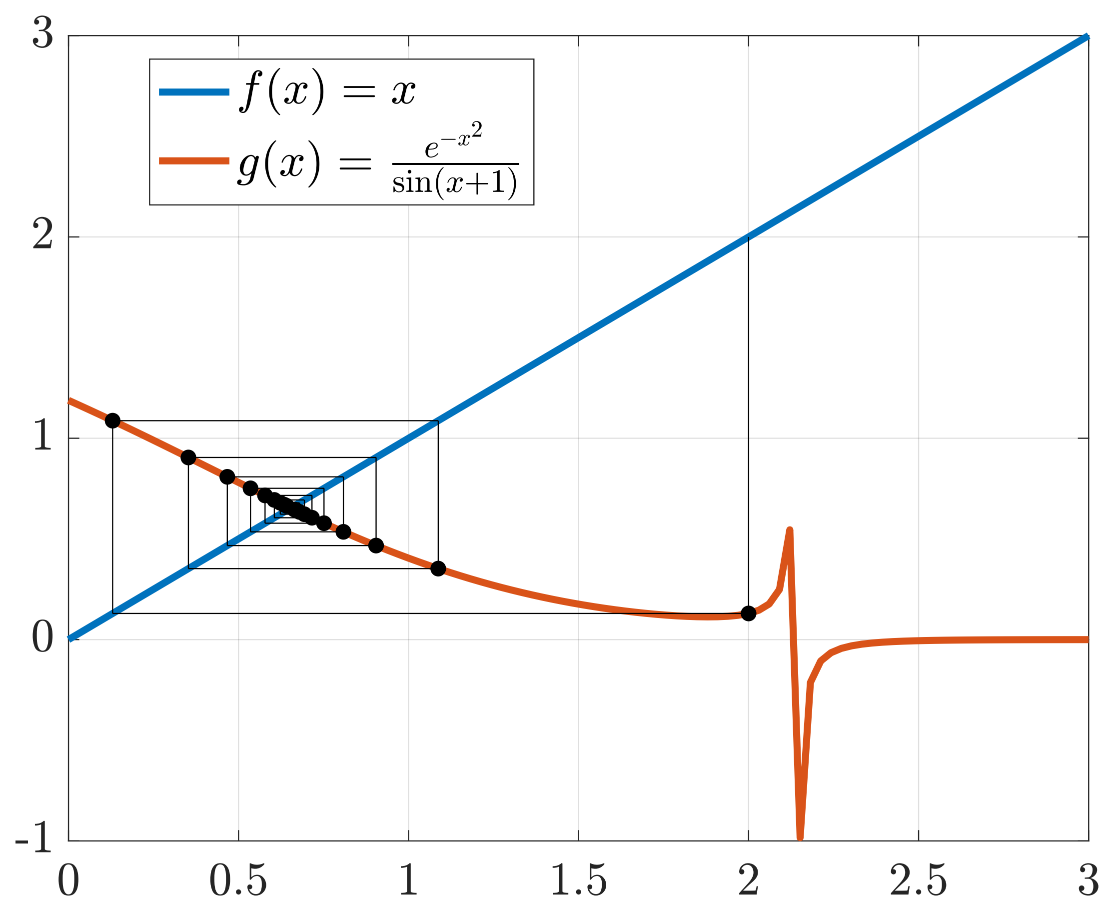
FIXED-POINT ITERATIONS
A REVIEW

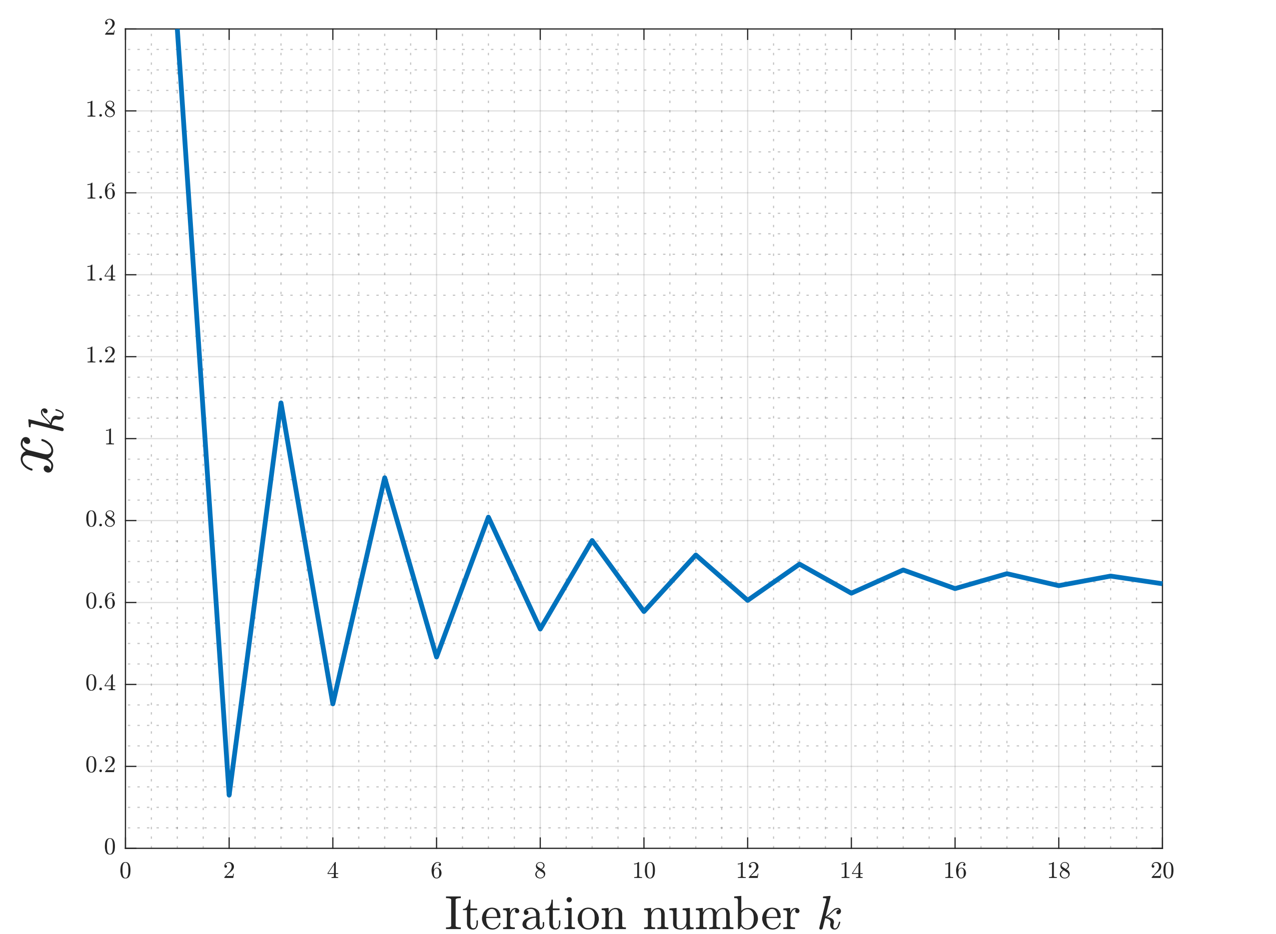
Algorithm: Fixed-point iteration

For $k = 1 \dots K$ do:

$$x_{k+1} = g(x_k)$$

$$x_{k+1} = \frac{e^{-x_k^2}}{\sin(x_k + 1)}$$





FIXED-POINT METHODS
SUCCESSIVE APPROXIMATION

- The equations we want to solve (invert) look like: $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{y}$
- Define an operator: $T(\boldsymbol{x}) = \boldsymbol{x} + \lambda(\boldsymbol{y} - \boldsymbol{A}\boldsymbol{x})$
- The so-called **fixed-point** of $T(\boldsymbol{x})$ is a solution to $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{y}$, and vice versa. Hence $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{y}$ is equivalent to: $\boldsymbol{x} = T(\boldsymbol{x})$
  - **Intuition:** putting $T(\boldsymbol{x}) = \boldsymbol{x} + \lambda(\boldsymbol{y} - \boldsymbol{A}\boldsymbol{x})$ into $\boldsymbol{x} = T(\boldsymbol{x})$ gives $\boldsymbol{x} = \boldsymbol{x} + \lambda(\boldsymbol{y} - \boldsymbol{A}\boldsymbol{x})$
- Classic approach for solving: equations of the form $\boldsymbol{x} = T(\boldsymbol{x})$ is to used fixed-point iterations: i.e. do

$$\boldsymbol{x}_{k+1} = T(\boldsymbol{x}_k)$$

FIXED-POINT METHODS
SUCCESSIVE APPROXIMATION

- Classic approach for solving: equations of the form $\boldsymbol{x} = T(\boldsymbol{x})$ is to used fixed-point iterations: i.e. do

$$\boldsymbol{x}_{k+1} = T(\boldsymbol{x}_k)$$

Hence, $\boldsymbol{x}_{k+1} = \boldsymbol{x}_k + \lambda(\boldsymbol{y} - \boldsymbol{A}\boldsymbol{x}_k)$

$$= \lambda \boldsymbol{y} + (I - \lambda \boldsymbol{A})\,\boldsymbol{x}_k$$

  - **BUT! Will this converge always?**
- The answer can be obtained in the **Fourier domain**

FIXED-POINT METHODS
SUCCESSIVE APPROXIMATION

- Iteration scheme:
$$\boldsymbol{x}_{k+1} = \lambda \boldsymbol{y} + (I - \lambda \boldsymbol{A})\,\boldsymbol{x}_k$$
- Taking FT of both sides of the iteration scheme:
$$\boldsymbol{X}_{k+1}(\omega) = \lambda \boldsymbol{Y}(\omega) + \left(1 - \lambda \boldsymbol{H}(\omega)\right)\ \boldsymbol{X}_k(\omega)$$
$$\boldsymbol{X}_k(\omega) = \left(1 - \lambda \boldsymbol{H}(\omega)\right)^k \boldsymbol{X}_0(\omega) + \lambda \left[1 - \left(1 - \lambda \boldsymbol{H}(\omega)\right)^k\right] \frac{\boldsymbol{Y}(\omega)}{\boldsymbol{H}(\omega)}$$
- For frequencies when $\boldsymbol{H}(\omega) \neq 0$, then the iteration will converge iff:
$$|\,1 - \lambda \boldsymbol{H}(\omega)\,| < 1,$$
- This is in reality a very restrictive condition.

FIXED-POINT METHODS
ALTERNATIVE APPROACH: LANDWEBER

- Consider the equivalent system: $\boldsymbol{A}^*\boldsymbol{A}\boldsymbol{x} = \boldsymbol{A}^*\boldsymbol{y}$
- **Iteration scheme:**

$$\boldsymbol{x}_{k+1} = \lambda \boldsymbol{A}^*\boldsymbol{y} + (I - \lambda \boldsymbol{A}^*\boldsymbol{A})\,\boldsymbol{x}_k$$

- By repeating the same steps as before, it follows that:

$$X_k(\omega) = \left(1 - \lambda\,|\boldsymbol{H}(\omega)|^2\right)^k X_0(\omega) + \lambda\left[1 - \left(1 - \lambda\,|\boldsymbol{H}(\omega)|^2\right)^k\right]\frac{\boldsymbol{Y}(\omega)}{\boldsymbol{H}(\omega)}$$

- For frequencies when $\boldsymbol{H}(\omega) \neq 0$, then the iteration will converge iff:

$$-1 < 1 - \lambda\,|\boldsymbol{H}(\omega)|^2 < 1,$$

- Or equivalently:

$$0 < \lambda < \frac{2}{|\boldsymbol{H}(\omega)|^2}$$

- Or: $\quad 0 < \lambda < \dfrac{2}{\boldsymbol{H}_{\max}^2}$ (where $\boldsymbol{H}_{\max}$ is the maximum value of $|\boldsymbol{H}(\omega)|$).

FIXED-POINT METHODS

LANDWEBER

Algorithm: Landweber iteration

Input: A, y, λ

Initialize $\boldsymbol{x}_0$

For $k = 1 \ldots K$ do:

$$\boldsymbol{x}_{k+1} = \lambda \boldsymbol{A}^* \boldsymbol{y} + (1 - \lambda \boldsymbol{A}^* \boldsymbol{A}) \boldsymbol{x}_k$$

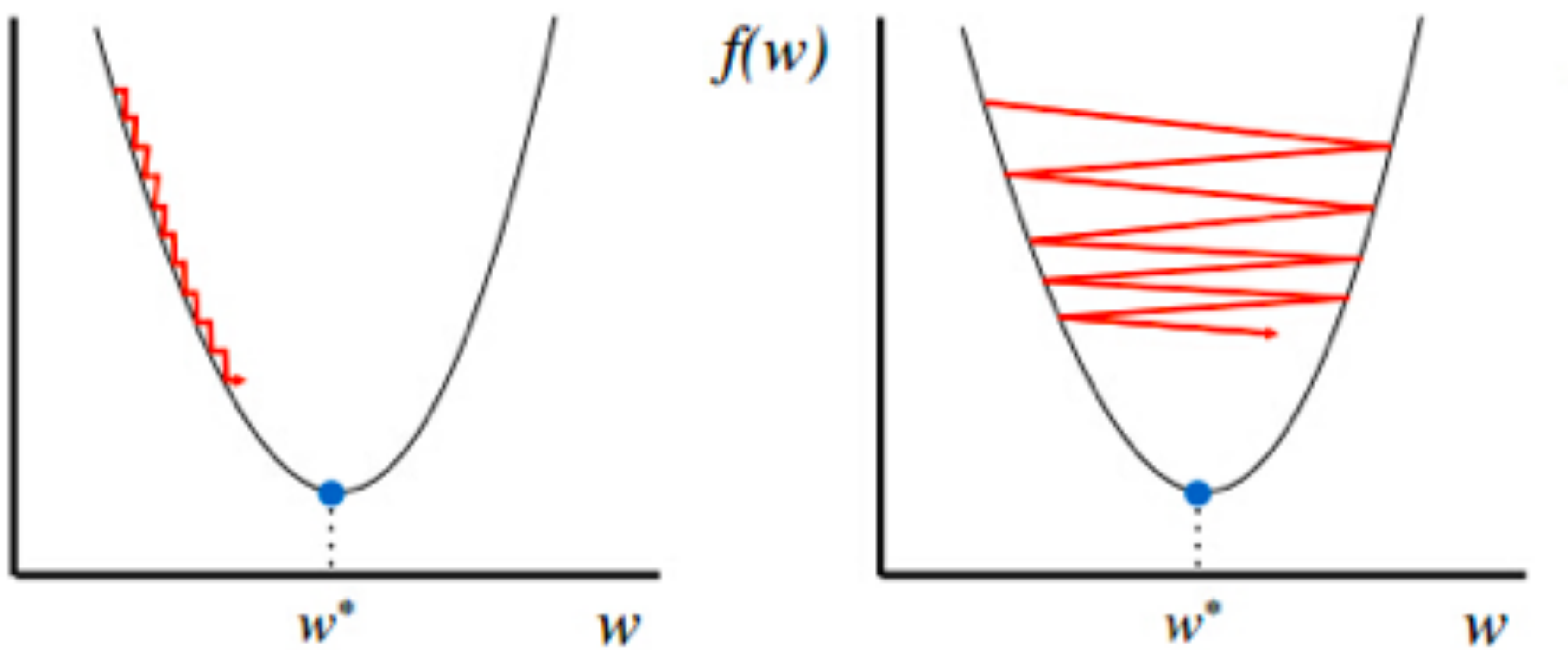
Return: $\boldsymbol{x}_{\text{est}} = \boldsymbol{x}_k$

Remember that the choice of λ should satisfy: $0 < \lambda < \dfrac{2}{\boldsymbol{H}_{\max}^2}$ (can just start small)

Also, x_0 can be initialized randomly or with zeros.

SELECTING THE STEP-SIZE (LEARNING RATE) λ

- **Choose λ too small:** takes long to converge
- **Choose λ too large:** overshoots the optima and may even diverge



Landweber method belongs to a class of Gradient Methods

STEEPEST DESCENT

CONJUGATE GRADIENT METHOD

GRADIENT METHODS

GRADIENT METHODS

Key idea: at each step of the iteration, the new approximation is obtained by updating the old one, in the **direction of the gradient** of the **error functional.**

STEEPEST DESCENT

CONJUGATE GRADIENT METHOD

GRADIENT METHODS

GRADIENT METHODS

STEEPEST DESCENT

- Real systems are imperfect and so **measurements are often noisy**.
- **Inverse problem:** To solve for $\boldsymbol{x}$ given the linear system

$$\boldsymbol{y} = \boldsymbol{A}\boldsymbol{x} + \boldsymbol{n},$$

where $\boldsymbol{n}$ is (unknown) noisy perturbations in the measurements.

- **Error functional:**

$$\mathscr{E}(\boldsymbol{x}) = \frac{1}{2}\|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{y}\|_2^2$$

- **Goal:** find the $\boldsymbol{x}$ that minimizes this error functional

$$\boldsymbol{x}_{\text{est}} = \arg\min_{\boldsymbol{x}} \frac{1}{2}\|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{y}\|_2^2$$

GRADIENT METHODS
STEEPEST DESCENT

- **Iterative scheme:** the update to be in the direction of the negative gradient of $\mathscr{E}(\boldsymbol{x})$:

$$\boldsymbol{x}_{k+1} = \boldsymbol{x}_k - \lambda \cdot \nabla_{\boldsymbol{x}} \left(\mathscr{E}(\boldsymbol{x}) \right) \Big|_{\boldsymbol{x}_k}$$

- Note that the gradient of the error functional is:

$$\nabla_{\boldsymbol{x}} \left(\mathscr{E}(\boldsymbol{x}) \right) \Big|_{\boldsymbol{x}_k} = \boldsymbol{A}^*(\boldsymbol{A}\boldsymbol{x}_k - \boldsymbol{y})$$

- One can find the optimal value for the step-size λ:

$$\lambda = \lambda_k = \frac{\|\boldsymbol{r}_k\|^2}{\|\boldsymbol{A}\boldsymbol{r}_k\|^2}, \text{ where } \boldsymbol{r}_k = \boldsymbol{A}^*(\boldsymbol{A}\boldsymbol{x}_k - \boldsymbol{y})$$

GRADIENT METHOD

STEEPEST DESCENT

Algorithm: Steepest descent

Input: $A, \boldsymbol{y}$

Initialize $\boldsymbol{x}_0$

For $k = 0 \ldots K-1$ do:

$$\boldsymbol{r}_k = \boldsymbol{A}^*(\boldsymbol{A}\boldsymbol{x}_k - \boldsymbol{y})$$

$$\lambda_k = \frac{\|\boldsymbol{r}_k\|^2}{\|\boldsymbol{A}\boldsymbol{r}_k\|^2}$$

$$\boldsymbol{x}_{k+1} = \boldsymbol{x}_k - \lambda_k \boldsymbol{r}_k$$

Return: $\boldsymbol{x}_{\text{est}} = \boldsymbol{x}_K$

Observe that the search directions are built from the residual vectors $\boldsymbol{r}_k$

STEEPEST DESCENT

CONJUGATE GRADIENT METHOD

GRADIENT METHODS

GRADIENT METHODS
CONJUGATE GRADIENT METHOD

- **Steepest descent:** the search directions are built from residual vectors $\boldsymbol{r}_k$
  - Next residual is the current one plus an update: $\boldsymbol{r}_{k+1} = \boldsymbol{r}_k - \alpha_k \boldsymbol{A}^*\boldsymbol{A}\boldsymbol{d}_k$
  - The solution is updated as: $\boldsymbol{x}_{k+1} = \boldsymbol{x}_k + \alpha_k \boldsymbol{d}_k$
- **Conjugate gradient:** a better update direction $\boldsymbol{d}_k$ is:
  - Each new **residual** is **orthogonal** to all the previous residuals and **search directions**;
  - Each new **search direction** is constructed (from the residual) to be A**-orthogonal** to all the **previous residuals and search directions**.
  - $\alpha_k = \dfrac{\|\boldsymbol{r}_k\|^2}{\langle \boldsymbol{r}_k, \boldsymbol{A}^*\boldsymbol{A}\boldsymbol{d}_k\rangle}$
  - Update $\boldsymbol{d}_{k+1} = \boldsymbol{r}_{k+1} - \beta_k \boldsymbol{d}_k$, where $\beta_k = \|\boldsymbol{r}_{k+1}\|^2 / \|\boldsymbol{r}_k\|^2$

Algorithm: Steepest descent

Input: A, y

Initialize $\boldsymbol{x}_0$

For $k = 0 \ldots K-1$ do:

$$\boldsymbol{r}_k = A^*(A\boldsymbol{x}_k - y)$$

$$\lambda_k = \frac{\|\boldsymbol{r}_k\|^2}{\|A\boldsymbol{r}_k\|^2}$$

$$\boldsymbol{x}_{k+1} = \boldsymbol{x}_k - \lambda_k \boldsymbol{r}_k$$

Return: $\boldsymbol{x}_{\text{est}} = \boldsymbol{x}_K$

GRADIENT METHODS

CONJUGATE GRADIENT METHOD (CGM)

- **Conjugate gradient:** a better update direction d_k is such that
  - Each new **residual** is **orthogonal** to all the previous residuals and **search directions**;
  - Each new **search direction** is constructed (from the residual) to be A**-orthogonal** to all the **previous residuals and search directions.**
  - $\alpha_k = \dfrac{||r_k||^2}{\langle r_k, A^*Ad_k \rangle}$
  - Update $d_{k+1} = r_{k+1} - \beta_k d_k$, where $\beta_k = ||r_{k+1}||^2/||r_k||^2$

Due to these factors, CGM is a faster algorithm that converges in steps upper bounded by the number.

GRADIENT METHOD

CONJUGATE GRADIENT METHOD

Will find the exact solution in a finite number of iterations, which is at most the size of the matrix A

Algorithm: Conjugate gradient

Input: $A, \boldsymbol{y}$

Initialize $\boldsymbol{x}_0$

Set $\boldsymbol{p}_0 = \boldsymbol{r}_0 = A^*\boldsymbol{y}$

For $k = 0 \ldots K-1$ do:

$$\alpha_k = \frac{\|\boldsymbol{r}_k\|^2}{\langle \boldsymbol{r}_k, A^*A\boldsymbol{p}_k \rangle}$$

$$\boldsymbol{x}_{k+1} = \boldsymbol{x}_k + \alpha_k \boldsymbol{p}_k$$

$$\boldsymbol{r}_{k+1} = \boldsymbol{r}_k - \alpha_k A^*A\boldsymbol{p}_k$$

$$\beta_{k+1} = \frac{\|\boldsymbol{r}_{k+1}\|^2}{\|\boldsymbol{r}_k\|^2}$$

$$\boldsymbol{p}_{k+1} = \boldsymbol{r}_{k+1} + \beta_{k+1}\boldsymbol{p}_k$$

Return: $\boldsymbol{x}_{\text{est}} = \boldsymbol{x}_K$

ITERATIVE REGULARIZATION METHODS

MATLAB PRACTICE 10

ITERATIVE METHODS

- Fixed point-iterations
- Gradient methods
    - Steepest descent method
    - Conjugate gradient method (time permitting)
- Compare execution speed

WHAT WE COVERED TODAY

- **Iterative methods**
  - Fixed-point method
  - Steepest descent
  - Conjugate gradient method

TILL NEXT TIME

CURING ILL-CONDITIONING WITH REGULARIZATION (CONTINUED)