Computational inverse problems

Nuutti Hyvönen and Juha-Pekka Puska

nuutti.hyvonen@aalto.fi, juha-pekka.puska@aalto.fi

Second lecture

2.2 Truncated singular value decomposition (cont.)

Summary of the previous lecture

The problem: Find $x \in H_1$ that satisfies the equation

$$Ax = y,$$

where $y \in H_2$ is given and $A : H_1 \to H_2$ is a compact linear operator.

Singular value decomposition (SVD):

$$Ax = \sum_n \lambda_n \langle x, v_n \rangle u_n \qquad \text{for all } x \in H_1.$$

The solutions: If solutions exist, they are of the form

$$x = x_0 + \sum_n \frac{1}{\lambda_n} \langle y, u_n \rangle v_n,$$

where $x_0 \in \mathrm{Ker}(A)$.

Solvability conditions: There exists a solution *if and only if*

$$y = Py \qquad \text{and} \qquad \sum_n \frac{1}{\lambda_n^2} |\langle y, u_n \rangle|^2 < \infty,$$

where P is an orthogonal projection onto $\overline{\mathrm{Ran}(A)} = \overline{\mathrm{span}\{u_n\}}$.

The natural way to circumvent problems with the first solvability condition is to consider the projected equation

$$Ax = PAx = Py$$

instead of $Ax = y$. However, this does not help with the second condition since there is no guarantee that

$$\sum_n \frac{1}{\lambda_n^2} |\langle Py, u_n \rangle|^2 < \infty$$

for a general $y \in H_2$, if $\mathrm{rank}(A) = \infty$, i.e., if $\mathrm{Ran}(A)$ is infinite-dimensional.

Truncated singular value decomposition (TSVD)

Let us define a family of finite-dimensional orthogonal projections by

$$P_k : H_2 \to \mathrm{span}\{u_1, \ldots, u_k\}, \quad y \mapsto \sum_{n=1}^{k} \langle y, u_n \rangle u_n.$$

Due to the orthogonality of $\{u_n\}$,

$$P(P_k y) \;=\; \sum_n \langle P_k y, u_n \rangle u_n \;=\; \sum_{n=1}^{k} \langle y, u_n \rangle u_n \;=\; P_k y,$$

and, moreover,

$$\sum_n \frac{1}{\lambda_n^2} |\langle P_k y, u_n \rangle|^2 \;=\; \sum_{n=1}^{k} \frac{1}{\lambda_n^2} |\langle y, u_n \rangle|^2 < \infty.$$

(Note that one must choose $k \leq \mathrm{rank}(A)$ if the latter is finite.)

In consequence, the problem

$$Ax \;=\; P_k y. \tag{4}$$

satisfies the solvability conditions (3). The corresponding solutions are given by

$$x \;=\; x_0 + \sum_n \frac{1}{\lambda_n} \langle P_k y, u_n \rangle v_n \;=\; x_0 + \sum_{n=1}^{k} \frac{1}{\lambda_n} \langle y, u_n \rangle v_n \in H_1.$$

By the *truncated SVD solution* of $Ax = y$ for a given $k \geq 1$, we mean the $x_k \in H_1$ that satisfies (4) and is orthogonal to the subspace $\mathrm{Ker}(A)$. Since $\{v_n\}$ span $(\mathrm{Ker}(A))^{\perp}$, it easily follows that such x_k is *unique*, has the *smallest norm* of the solutions to (4), and is given by

$$x_k \;=\; \sum_{n=1}^{k} \frac{1}{\lambda_n} \langle y, u_n \rangle v_n.$$

The special case: $H_1 = \mathbb{R}^n$ and $H_2 = \mathbb{R}^m$

Let $H_1 = \mathbb{R}^n$ and $H_2 = \mathbb{R}^m$, which means that

$$Ax = y$$

is a matrix equation or, in other words, a system of linear equations. In particular, $A \in \mathbb{R}^{m \times n}$.

Since all operators of finite rank, i.e., with finite-dimensional range, are compact, we have the representation

$$Ax = \sum_{j=1}^{p} \lambda_j (x^{\mathrm{T}} v_j) u_j = \sum_{j=1}^{p} \lambda_j (u_j v_j^{\mathrm{T}}) x, \qquad p \leq \min\{n, m\},$$

where $\{v_j\}_{j=1}^{p} \subset \mathbb{R}^n$ and $\{u_j\}_{j=1}^{p} \subset \mathbb{R}^m$ are sets of orthonormal vectors and $\{\lambda_j\}_{j=1}^{p}$ are positive numbers such that $\lambda_j \geq \lambda_{j+1}$. (Note that $p = \mathrm{rank}(A)$.)

How can one write this decomposition in a neat matrix form?

Let us introduce, e.g., by Gram–Schmidt process, complementary sets of orthonormal vectors $\{v_j\}_{j=p+1}^n$ and $\{u_j\}_{j=p+1}^m$, such that the completed systems $\{v_j\}_{j=1}^n$ and $\{u_j\}_{j=1}^m$ are orthonormal basis for $\mathbb{R}^n$ and $\mathbb{R}^m$. Moreover, we set $\lambda_j = 0$ for $j = p+1, \ldots, \min\{n, m\}$.

Next, we define three auxiliary matrices:

$$\begin{aligned} V &= [v_1, \ldots, v_n] \in \mathbb{R}^{n\times n}, \\ U &= [u_1, \ldots, u_m] \in \mathbb{R}^{m\times m}, \\ \Lambda &= \mathrm{diag}(\lambda_1, \ldots, \lambda_{\min\{n,m\}}) \in \mathbb{R}^{m\times n}. \end{aligned}$$

Here, $\Lambda \in \mathbb{R}^{m\times n}$ is a diagonal matrix with the elements $\lambda_1, \ldots, \lambda_{\min\{n,m\}}$ on its diagonal; if $m > n$ (resp. $n > m$), there are $m - n$ empty rows (resp. $n - m$ empty columns) at the bottom of Λ (resp. at the right end of Λ). Note that due to the orthonormality of $\{v_j\}$ and $\{u_j\}$, the matrices V and U are orthogonal:

$$V^{\mathrm{T}}V = VV^{\mathrm{T}} = I \qquad \text{and} \qquad U^{\mathrm{T}}U = UU^{\mathrm{T}} = I.$$

A simple computation shows that

$$U\Lambda V^{\mathrm{T}}x \;=\; \sum_{j=1}^{p} \lambda_j u_j (v_j^{\mathrm{T}} x) \;=\; Ax$$

for all $x \in \mathbb{R}^n$. Hence, we have the decomposition

$$A \;=\; U\Lambda V^{\mathrm{T}}.$$

This is what we call the SVD in the case of matrices in $\mathbb{R}^{m\times n}$. In particular, this is how Matlab understands the SVD.

Note, in particular, that the singular values $\{\lambda_j\}_{j=1}^{\min\{n,m\}}$ are just *non-negative* — earlier they were assumed to be positive —, and

$$\begin{aligned}
\mathrm{Ran}(A) &= \mathrm{span}\{u_j \mid 1 \leq j \leq p\}, \\
\mathrm{Ker}(A) &= \mathrm{span}\{v_j \mid p+1 \leq j \leq n\}, \\
(\mathrm{Ran}(A))^{\perp} &= \mathrm{span}\{u_j \mid p+1 \leq j \leq m\}, \\
(\mathrm{Ker}(A))^{\perp} &= \mathrm{span}\{v_j \mid 1 \leq j \leq p\}.
\end{aligned}$$

Truncated SVD for a matrix $A \in \mathbb{R}^{m\times n}$

The truncated SVD solution, i.e., the solution of

$$Ax \;=\; P_k y \quad \text{and} \quad x \perp \mathrm{Ker}(A), \qquad\qquad 1 \le k \le p,$$

where $P_k \to \mathrm{span}\{u_1, \ldots, u_k\}$ is an orthogonal projection, is given in the matrix framework by

$$x_k \;=\; \sum_{j=1}^{k} \frac{1}{\lambda_j} \langle y, u_j \rangle v_j \;=\; \sum_{j=1}^{k} \frac{1}{\lambda_j} v_j (u_j^{\mathrm{T}} y) \;=\; V \Lambda_k^{\dagger} U^{\mathrm{T}} y.$$

Here, $\Lambda_k^{\dagger} \in \mathbb{R}^{n\times m}$ is a diagonal matrix, with $\min\{m, n\}$ number of non-negative elements $1/\lambda_1, \ldots, 1/\lambda_k, 0, \ldots, 0$ on its diagonal.

For the largest possible cut-off $k = p$, the matrix

$$A^\dagger \;:=\; A_p^\dagger \;=\; V\Lambda_p^\dagger U^{\mathrm{T}} \;=:\; V\Lambda^\dagger U^{\mathrm{T}}$$

is called the *Moore–Penrose pseudoinverse*. It follows from the above material that $x^\dagger = A^\dagger y$ is the solution of the projected equation

$$Ax \;=\; P_p y \;=\; Py,$$

where $P : \mathbb{R}^m \to \mathbb{R}^m$ is, once again, the orthogonal projection onto $\mathrm{Ran}(A)$. However, since the smallest non-zero singular value λ_p is typically extremely small in inverse problems, the use of pseudoinverse is often very sensitive to inaccuracies in the data y.

An example: Heat distribution in a rod (revisited)

Recall once again the heat equation

$$\begin{aligned} u_t &= u_{xx} && \text{in } (0,\pi)\times\mathbb{R}_+, \\ u_x(0,\cdot) &= u_x(\pi,\cdot) = 0 && \text{on } \mathbb{R}_+, \\ u(\cdot,0) &= f && \text{on } (0,\pi). \end{aligned}$$

Our plan is to discretize the dependence on the spatial variable x, and then investigate the properties of the corresponding inverse problem numerically.

To begin with, we introduce the step size $h = \pi/100$ and the grid points $x_j = jh$, $j = 0, \ldots, 100$. Furthermore, we denote $U_j(t) = u(x_j, t)$.

We approximate the second derivative of u with respect to x at the point (x_j, t) by the difference rule:

$$u_{xx}(x_j, t) \approx \frac{1}{h^2}\left(U_{j-1}(t) - 2\,U_j(t) + U_{j+1}(t)\right), \qquad 1 \le j \le 99.$$

Furthermore, we discretize the boundary conditions by requiring that

$$u_x(0, t) \approx \frac{1}{h}\left(U_1(t) - U_0(t)\right) = 0 = \frac{1}{h}\left(U_{100}(t) - U_{99}(t)\right) \approx u_x(\pi, t).$$

By solving this for $U_0(t)$ and $U_{100}(t)$ and substituting into the preceding difference rule, we obtain altogether that

$$\begin{aligned}
u_{xx}(x_1, t) &\approx \frac{1}{h^2}\left(-U_1(t) + U_2(t)\right), \\
u_{xx}(x_j, t) &\approx \frac{1}{h^2}\left(U_{j-1}(t) - 2\,U_j(t) + U_{j+1}(t)\right), \qquad 2 \le j \le 98, \\
u_{xx}(x_{99}, t) &\approx \frac{1}{h^2}\left(U_{98}(t) - U_{99}(t)\right).
\end{aligned}$$

Denoting $U(t) = (U_1(t), \ldots, U_{99}(t))^{\mathrm{T}}$ and $F = (f(x_1), \ldots, f(x_{99}))^{\mathrm{T}}$ and plugging the above approximations into the heat equation, we end up with a set of ordinary differential equations:

$$\begin{aligned} U'(t) &= B\,U(t), \qquad t \in \mathbb{R}_+, \\ U(0) &= F, \end{aligned}$$

where $B \in \mathbb{R}^{99\times 99}$ is a certain tridiagonal matrix (see next slide).

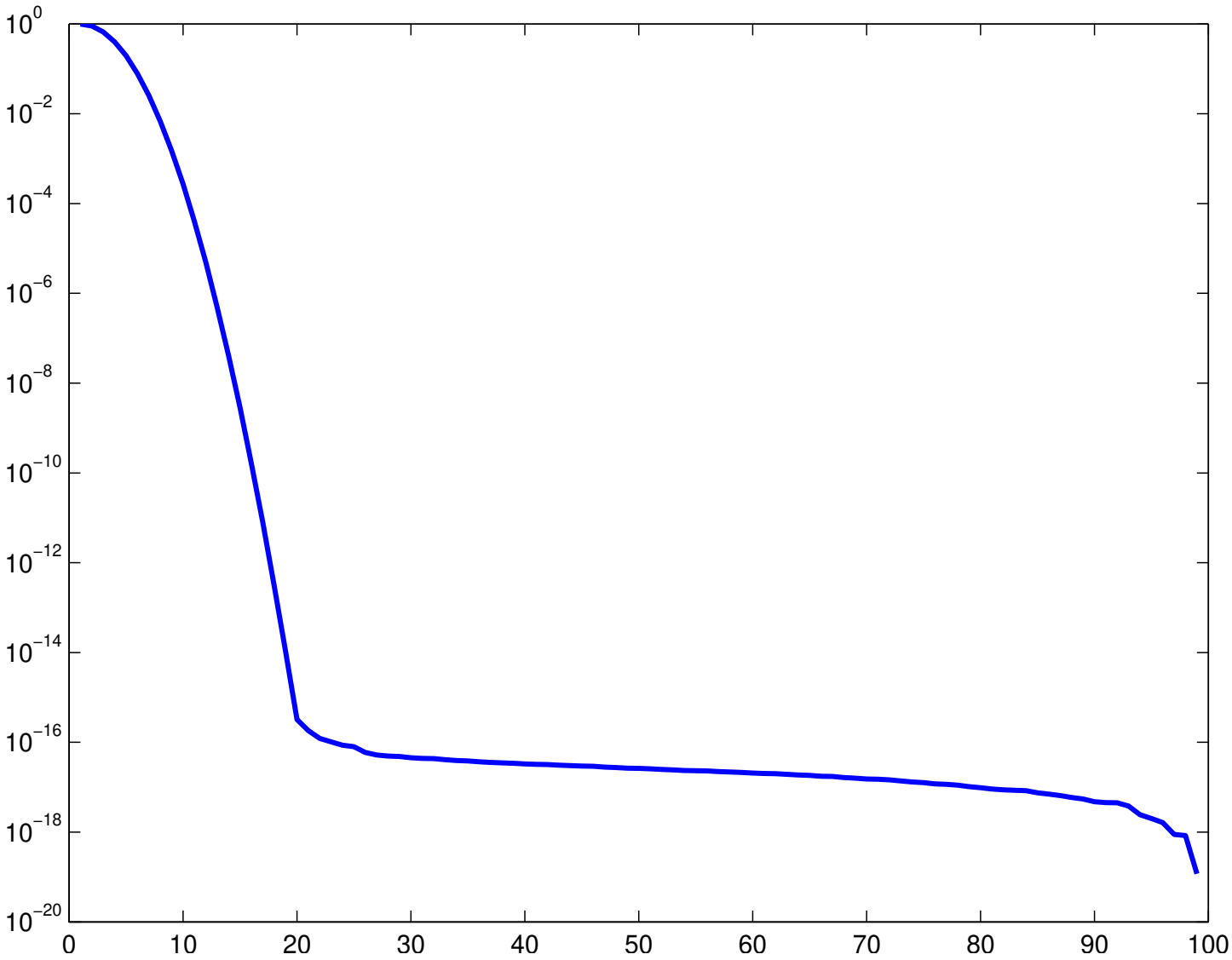
The forward solution corresponding to this space-discretized problem can be given with the help of the matrix exponent function as

$$U(T) = AF,$$

where $A = A(T) = e^{TB}$ and $T > 0$.

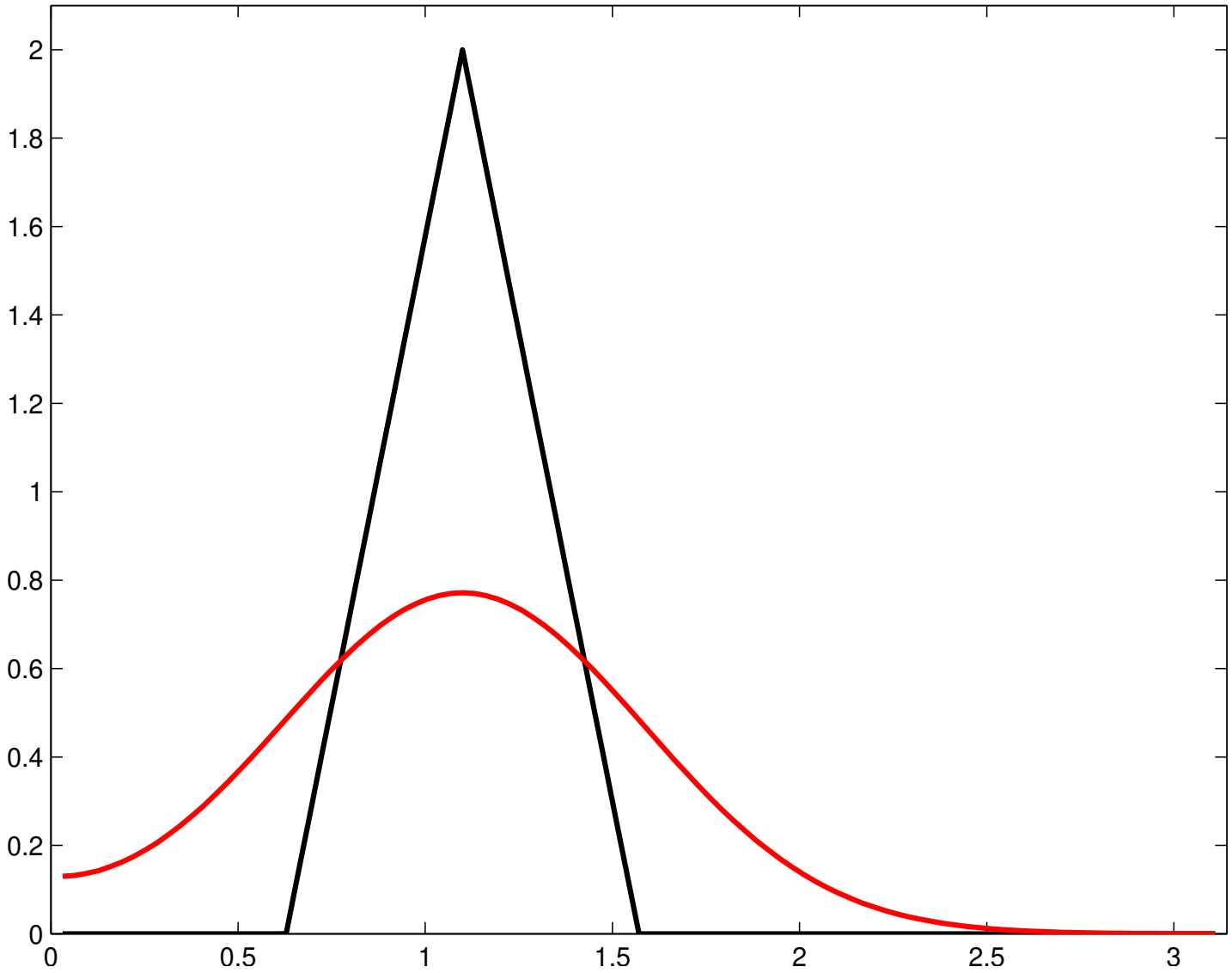
In Matlab, the matrices B and $A = e^{TB}$ can be formed by the following script, which also forms the SVD and plots the singular values for A:

```
T = 0.1; % say
N = 100;
h = pi/N;
B = diag(ones(N-2,1),-1) - 2*eye(N-1) + diag(ones(N-2,1),1);
B(1,1) = -1; % the left boundary condition
B(N-1,N-1) = -1; % the right boundary condition
B = B/h^2;
A = expm(T*B);
[U S V] = svd(A); % SVD
semilogy(diag(S), 'LineWidth', 2);
```


10^{0}
10^{-2}
10^{-4}
10^{-6}
10^{-8}
10^{-10}
10^{-12}
10^{-14}
10^{-16}
10^{-18}
10^{-20}
0
10
20
30
40
50
60
70
80
90
100

Let us next form a 'wedge function', which serves as the initial heat distribution, and compute the corresponding final distribution at $T = 0.1$:

```
x = linspace(h,pi-h,N-1); % the grid points
a = 40/3/pi; b1 = -8/3; b2 = 20/3; % coefficients
f = [a*x(1:35) + b1, -a*x(36:end) + b2]';
ind = f > 0;
f = f.*ind;
w = A*f; % final distribution
plot(x, f, 'k', 'LineWidth', 2);
hold on
plot(x, w, 'r', 'LineWidth', 2);
axis([0, pi, 0, 2.1])
hold off
```


2
1.8
1.6
1.4
1.2
1
0.8
0.6
0.4
0.2
0
0
0.5
1
1.5
2
2.5
3

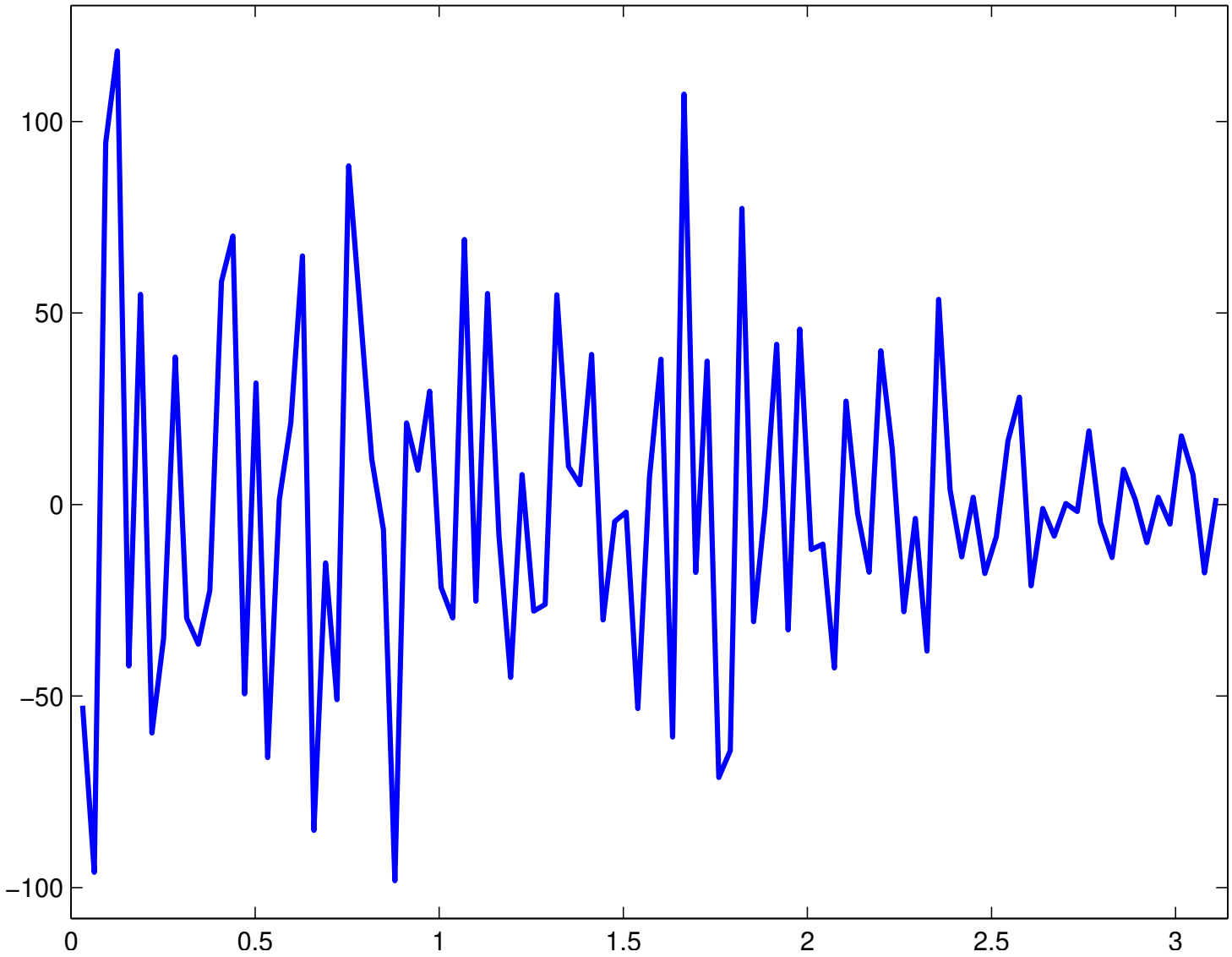
Let us be a bit silly and try to recover the initial heat distribution by inverting A:

```
f_stupid = A\w;
plot(x, f_stupid, 'LineWidth', 2);
```

which results in a catastrophe as demonstrated on the next slide. This is not surprising since writing

```
rank(A)
```

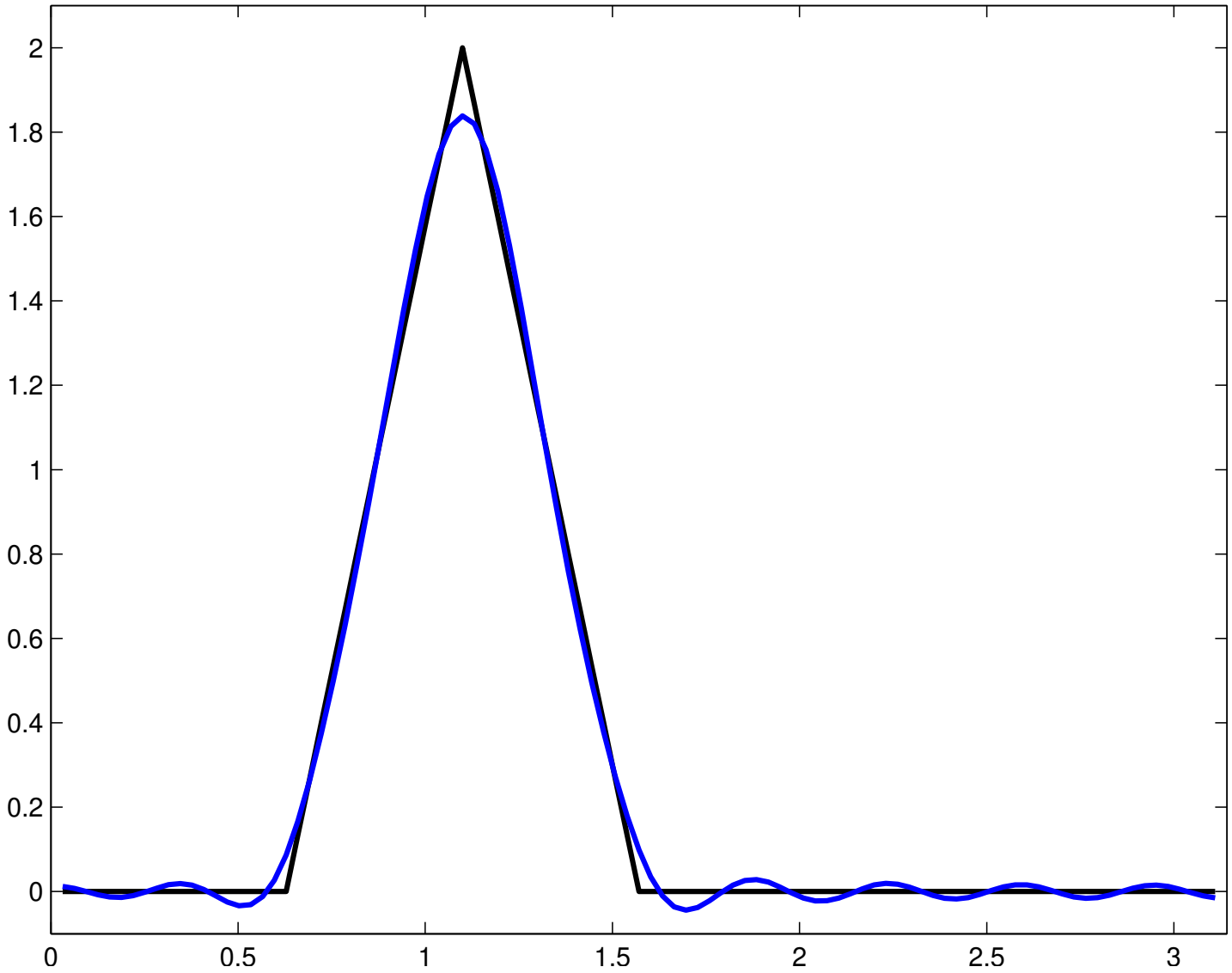
in Matlab, gives the value 18. In other words, from Matlab's numerical point of view, A has only 18 linearly independent columns — in particular, A is not (numerically) invertible.


100
50
0
−50
−100
0
0.5
1
1.5
2
2.5
3

Let us be more clever and compute the truncated SVD solution for $k = 18$:

```
k = 18; % the (numerical) rank of A
d = diag(S); % the singular values
idk = [1./d(1:k); zeros((N-1)-k,1)]; % invert only 18
iBk = V*diag(idk)*U'; % the corresponding 'inverse'
fk = iBk*w; % the 'solution'
plot(x, f, 'k','LineWidth', 2); hold on
plot(x, fk, 'LineWidth', 2); hold off
```

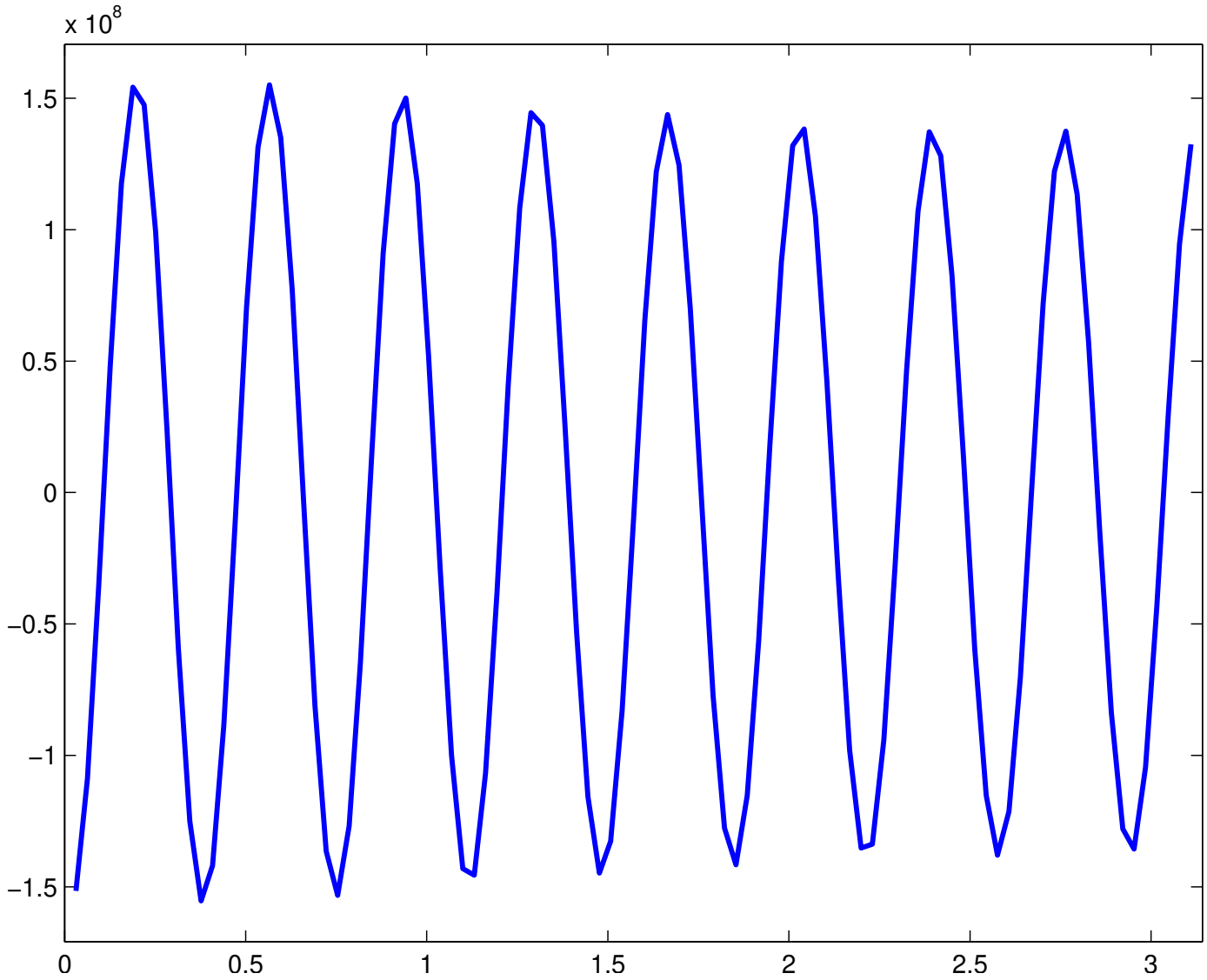
We have, actually, committed a severe *inverse crime*: If an inverse problem is solved using the same discretization with which the data was generated, the results are typically overly optimistic. This problem could be circumvented, e.g., by interpolating onto a sparser grid before the inversion. The 'inverse crime effect' can also be reduced by the addition of artificial noise.


2
1.8
1.6
1.4
1.2
1
0.8
0.6
0.4
0.2
0
0
0.5
1
1.5
2
2.5
3

In practice, the measurement is always inaccurate. Let us thus add just a tiny bit of noise in the measurement — so tiny that one could barely recognize it with naked eye. (In fact, this noise level corresponds approximately to the discrepancy between data sets simulated with the above introduced difference scheme and with an alternative method based on FFT and the SVD of the original solution operator E_T.)

```
wn = w + 0.001*randn(N-1,1); % noisy data
fkn_stupid = iBk*wn;
plot(x, fkn_stupid, 'LineWidth', 2);
```

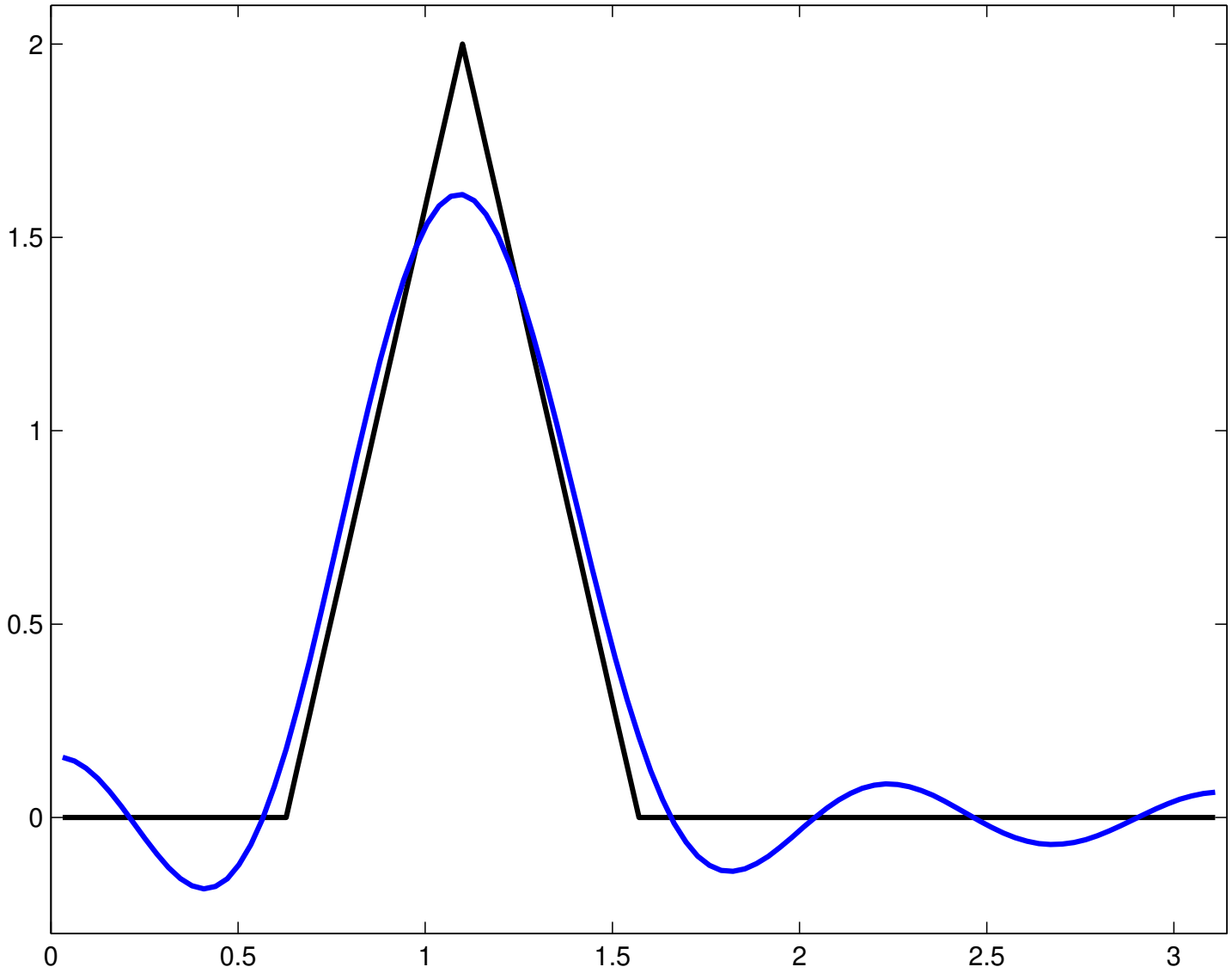
As demonstrated on the next slide, this approach does not work anymore. The reason is the following: The inverse of the 18th singular value is approximately $3.15 \cdot 10^{12}$, which means that the (ever so tiny) component of the noise vector in the direction v_{18} is heavily magnified.


x 10^8
1.5
1
0.5
0
−0.5
−1
−1.5
0
0.5
1
1.5
2
2.5
3

By trial and error, we decide to take the largest $k = 8$ singular values into account when computing the truncated SVD solution:

```
k = 8;
idk = [1./d(1:k); zeros((N-1)-k,1)];
iBk = V*diag(idk)*U';
fkn = iBk*wn;
plot(x, f, 'k','LineWidth', 2);
hold on
plot(x, fkn, 'LineWidth', 2);
hold off
```

This is pretty much the best one can do without additional information about the shape of the initial heat distribution. (For example, if we knew beforehand that f is piecewise linear, such information could be incorporated in the inversion algorithm, which would surely result in better reconstructions.)


2
1.5
1
0.5
0
0
0.5
1
1.5
2
2.5
3