TANA15/Lecture 9 - Contents

The Singular Value Decomposition

- Integral Equations.
- Application: Remote Sensing.

Sparse Matrices

- Compress Sparse Row storage.
- Stationary Iterative methods.

Integral Equations

Definition An *integral operator* $K : f \mapsto g$ can be written

$$g(x) = \int_a^b k(x,x')f(x')dx',$$

where $k(x,x')$ is the *kernel*.

Remark The operator maps $f(x) \in \mathcal{X}$ onto $g(x) \in \mathcal{Y}$ where $\mathcal{X}$ and $\mathcal{Y}$ are suitable function spaces. Usually $C^{(0)}([a,b])$, $C^{(1)}([a,b])$, $L^2([a,b])$, etc.

The spaces can be equipped with a *scalar product*

$$(f_1,f_2) = \int_a^b f_1(x)f_2(x)dx.$$

Definition An operator is *linear* if

$$K(\alpha_1 f_1 + \alpha_2 f_2) = \alpha_1 K f_1 + \alpha_2 K f_2,$$

for all f_1, f_2 in $\mathcal{X}$ and $\alpha_1,\alpha_2 \in \mathbb{R}$.

Lemma The integal operator K is linear.

Remark Operators on function spaces are studied in *Functional analysis*. How to turn this into Linear algebra?

Method By *Discretization* we mean replacing functions by vectors, i.e.

$$f(x) \Longrightarrow f = (f(x_1)\,, f(x_2)\,, \ldots\,, f(x_n))^T \in \mathbb{R}^n.$$

The operator is discretized using the *collocation method*

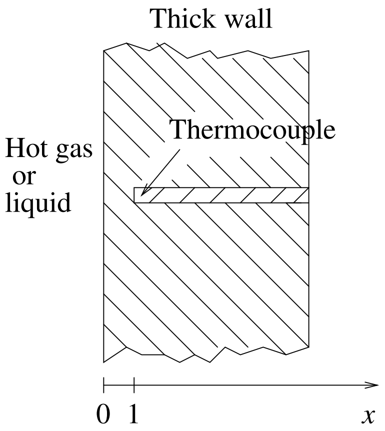
$$g(x_j) = \frac{b-a}{n} \sum_{i=1}^{n} k(x_j,x_i')f(x_i'), \quad j = 1,2,\ldots,n.$$

We get $Kf = g$, where $K \in \mathbb{R}^{n\times n}$. The scalar product (f_1,f_2) is also discretized

$$(f_1,f_2) = \frac{1}{n}\sum_{i=1}^{n} f_1(x_i)f_f(x_i).$$

Remark The function $f(x)$ can be recreated from the vector f using an interpolation scheme.

Remote Sensing: Temperature Measurements



Problem Find $f(t) = T(0,t)$ using measurements $g_m(t) \approx T(1,t)$.

Lemma The operator mapping $f(t) = T(0,t)$ onto the measurements $g(t) = T(1,t)$ is

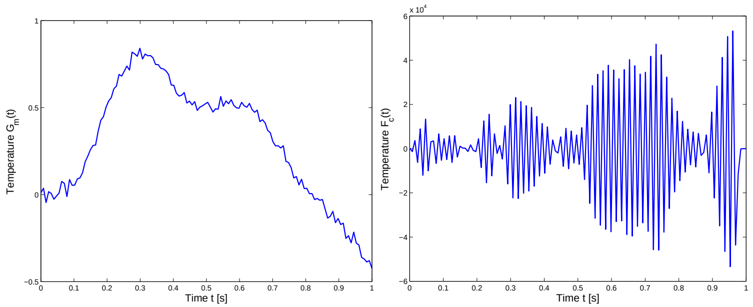
$$g(t) = (Kf)(t) = \int_0^t k(t-\tau) f(\tau) d\tau, \quad k(t) = \frac{\exp(-\frac{1}{4t})}{2t^{3/2}\sqrt{\pi}}.$$

Discretize Use a grid $0 = t_0 < t_1 < \ldots < t_{n-1} = 1$ to approximate the operator equation $(Kf)(t) = g(t)$ by a linear system

$$g = K_n f, \qquad K_n \in \mathbb{R}^{n \times n}.$$

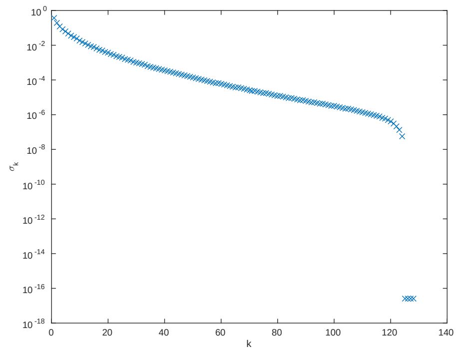
Idea Given a vector $g_m \in \mathbb{R}^n$ containing (noisy) measurements we solve the linear system $K_n f = g_m$.

Example Collect $n = 128$ noisy measurements in a vector g_m and attempt to compute $f_c = K_n^{-1} g_m$.

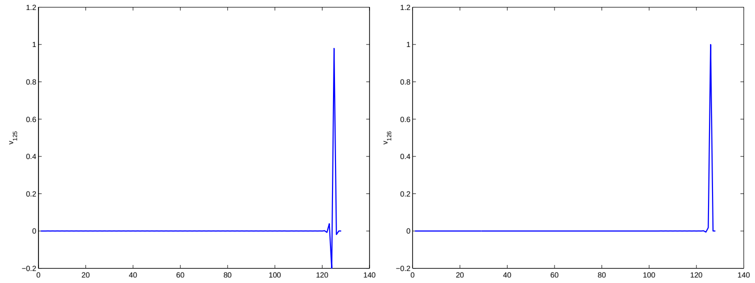


The data vector g_m (left) and the numerical solution f_c to the linear system of equations. The problem is very ill–contitioned!

Analysis Compute $K_n = U \Sigma V^T$. Plot the singular values $\{\sigma_k\}$.

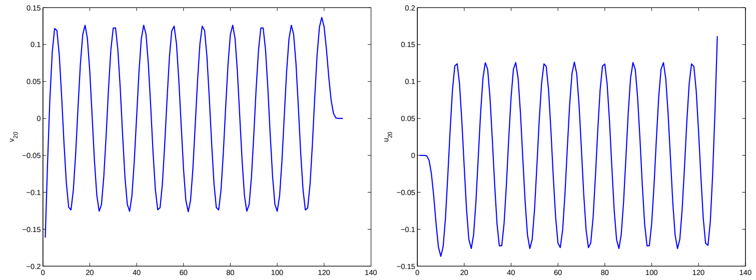


Results The singular values decrease from $\sigma_1 \approx 0.36$ continuously to $\sigma_{124} \approx 5.6 \cdot 10^{-8}$. The last 4 singular values are much smaller than the others. The problem is very *ill-conditioned!*



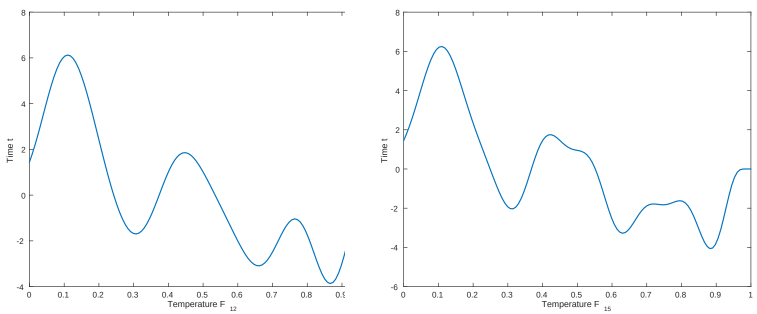
The basis functions v_{125} and v_{126}. Those components of $f(t)$ are multiplied by $\sigma_{125} = 3.1 \cdot 10^{-21}$ and $\sigma_{125} = 5.9 \cdot 10^{-24}$ respectively.

Conclusion There is a *time delay* in the problem. The signal $f(t)$ for t close to 1 doesn't have time to propagate through the medium and influence the measurement location $g(t)$. These components must be removed from the problem!



The basis functions v_{20} and u_{20}. The corresponding singular value is $\sigma_{20} = 3.6 \cdot 10^{-3}$. There is damping of high frequency components. The operator K is *smoothing*!

Conclusion Suppose the measurement errors are at most $\varepsilon = 10^{-2}$ then only the first $k = 12$ ($\sigma_{12} = 0.0129$), or at most $k = 15$ ($\sigma_{12} = 0.0077$), components $g_m^T u_k = \sigma_k f^T v_k$ are above the noise level.

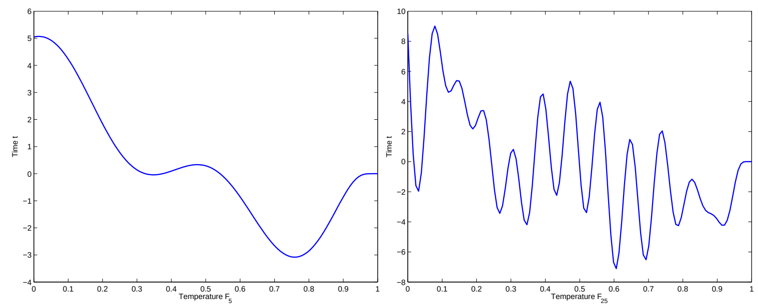


Numerical solution We include $k = 12$ and $k = 15$ components in

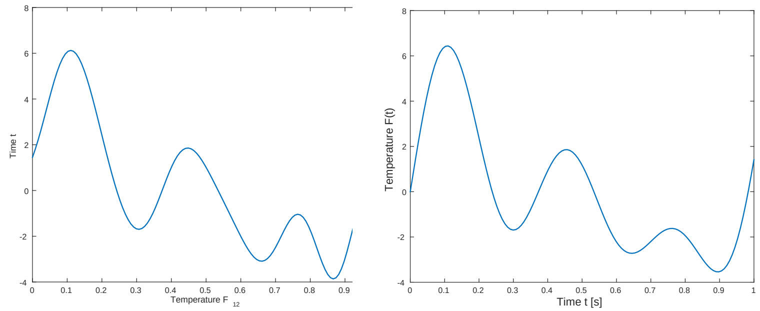
$$f^{(k)} = \sum_{j=1}^{k} \frac{u_j^T g_m}{\sigma_j} v_j.$$

Remark Both solutions are fairly similar. Only the components we belive to be accurate are included!

Numerical Solutions We include $k = 5$ or $k = 25$ components.



Remark Too few components and we miss features. Too many and the solution is mostly noise.



The best numerical uses $k=12$ singular components. Also the exact solution of the problem $f(t)$. Good accuracy except for the last 5 grid points.

Remark The SVD can reveal alot of information regarding a linear system of equations!

Application: Surface temperature on a steel roll

Example A steel roll of radius R_3 has been fitted with thermocouples at $r=R_2$. The interior of the roll is at $r=R_1$. Initially the roll is at a constant temperature. Find the transient surface temperature $T(x,R_3,t)=f(x,t)$ using measurements $T(x,R_2,t)=g(x,t)$.

Model The temperature in the steel roll $T(x,r,t)$ satisfies

$$\begin{cases} (kT_x)_x+\frac{1}{r}(rkT_r)_r=\rho c_p T_t, & \text{in } (a,b)\times(0,t_{end})\times(R_1,R_3),\\ T(x,R_3,t)=f(x,t), & \\ kT_r(x,r,t)=0, & \text{for } r=R_1,\\ T_x(x,r,t)=0, & \text{for } x=a, \text{ or } x=b,\\ T(x,r,0)=f(x,0), & \text{on } (L_1,L_2)\times(R_0,R_3)\times\{0\}, \end{cases}$$

where $f(x,0)$ is a constant function so $T(x,r)$ is also constant.

Lemma The mapping $K: f(x,t)\mapsto g(x,t)$ is linear.

Remark If we introduce a scalar product

$$(f_1,f_2)=\int_{x=a}^{b}\int_{t=0}^{t_{end}} f_1(x,t)f_2(x,t)dxdt,$$

then we can use the singular value decomposition.

Observation If we discretize K using $n=100$ grid points in x and $m=200$ in t then the matrix $K^{(nm)}$ is of dimension $nm=2\cdot 10^4$. Its not feasible to compute the SVD. Alternative?

Sparse Matrices

Observation In applications often matrices are *sparse*, i.e. most elements $a_{ij}=0$.

To store the full matrix A still requires n^2 slots of memory and a matrix–vector multiply,

$$y=Ax, \qquad y_i=\sum_{j=1}^{n} a_{ij}x_j,$$

still requires $2n^2$ floating point operations.

Idea Store only the non–zero elements $a_{ij}\neq 0$. Implement matrix–vector multiply so only need `nnz(A)` floating point operations.

Can store larger matrices and have a faster matrix–vector multiply!

Sparse Matrix Storage Schemes

Example Suppose the full matrix is

$$A = \begin{pmatrix} 1.1 & 0 & 0 & 3.7 & 0 & 0 & 0 & -1.2 & 0 & 0 \\ 0 & 1.2 & 0 & 0 & 0 & 4.3 & 0 & 0 & -1.9 & 0 \\ 0 & 0 & 2.1 & 0 & 0 & 0 & 0 & -1.8 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 3.1 & 0 & 0 & -1.5 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

The elements of the matrix A is stored using three vectors

`Elements`= $(1.1, 3.7, -1.2, 1.2, 4.3, -1.9, 2.1, -1.8, 3.1, -1.5)$
`ColumnIndex` = $(1\,,\,4\,,\,8\,,\,2\,,\,6\,,\,9\,,\,3\,,\,8\,,\,2\,,\,5)$
`RowEndIndex` = $(3\,,\,6\,,\,8\,,\,8\,,\,10)$

Matlab `S=sparse(A)`. This is called *Compress Sparse Row*.

Sparse Matrix Operations

Suppose A is a sparse matrix stored in the CSR format and that $\eta = nnz(A)$ is the number of non-zero elements of A.

Lemma Storing a matrix in CSR format requires $\mathcal{O}(\eta)$ slots of memory and a matrix–vector multiply $y = Ax$ uses $\mathcal{O}(\eta)$ operations.

Remark Matrix–Matrix multiply $C = AB$ should be avoided since the C usually isn't sparse.

Strongly favours iterative methods that only use matrix-vector multiply $y = Ax$, and often $y = A^T x$. Solve linear systems, compute eigenvalues, etc.

Origin of Sparse Matrices

Example Let $\Omega = [0, 1] \times [0, 1]$ and suppose we want to solve the boundary value problem,

$$\Delta u = 0, \quad \text{in } \Omega, \quad \text{and,} \quad u = g \text{ on } \partial\Omega.$$

We discreize Ω using a uniform mesh

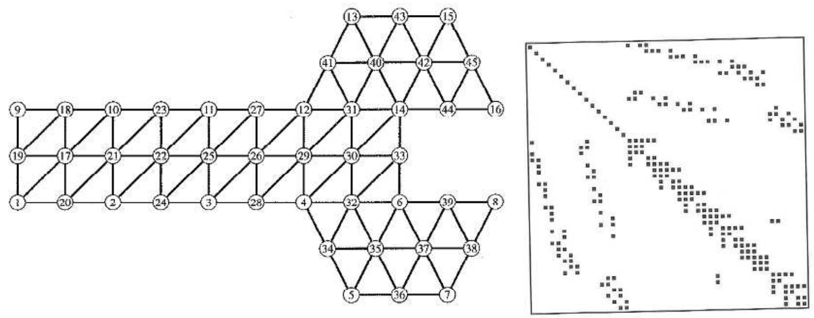
$$(x_i, y_j) = (i\Delta x, j\Delta y), \qquad 0 \leq i, j \leq N - 1.$$

The differential equation is approximated by,

$$u_{i,j-1} + u_{i-1,j} + u_{i+1,j} + u_{i,j+1} - 4u_{i,j} = 0, \qquad 1 \leq i, j \leq N - 2.$$

We obtain an $N^2 \times N^2$ matrix A with 5 non-zero elements on each row!

Typically want to use as large N as possible.



Example A finite element model and the resulting stiffness matrix. Here $a_{i,j}$ is non-zero only if nodes N_i and N_j are neighbours.

Example An iterative method for computing an eigenvalue is the power method.

Algorithm Take $q^{(0)}$ such that $\|q^{(0)}\|_2 = 1$. For $k = 1, 2, \ldots,$ do

$$\begin{aligned} w^{(k)} &= Aq^{(k-1)}, \\ \rho_{k-1} &= (q^{(k-1)})^T w^{(k)}, \\ q^{(k)} &= w^{(k)}/\|w^{(k)}\|_2. \end{aligned}$$

Then $(\rho_k, q^{(k)})$ converges to the eigenpair (λ_1, x_1).

Remark The power-iteration only uses matrix-vector multiply to compute eigenvalues and eigenvectors.

Classic Iterative Methods

Let A be a sparse matrix. We want to solve the linear system,

$$Ax = b.$$

Definition An *iterative method* creates a sequence $\{x_k\}$ given a starting approximation x_0. The method is *convergent* if $x_k \to x$, as $k \to \infty$.

Remark Efficient use of sparsity if the next iterate x_{k+1} is created from x_k using matrix–vector multiplications $y = Ax$ or $y = A^T x$.

Jacobis Method

Let x be the solution of the linear system. Then the ith component of the residual $b - Ax$ is

$$b_i - \sum_{j=1}^{n} a_{ij} x_j = 0, \quad \text{or,} \quad x_i = \frac{1}{a_{ii}} \left(b_i - \sum_{j=1, j \neq i}^{n} a_{ij} x_j \right).$$

Given a starting approximation $x^{(0)}$ we solve using fixed point iteration to obtain,

$$x_i^{(k+1)} = \frac{1}{a_{ii}} \left(b_i - \sum_{j=1, j \neq i}^{n} a_{ij} x_j^{(k)} \right), \quad i = 1, \ldots, n.$$

This is called *Jacobis method*.

Stationary Iterative Methods

Lemma Let $A = M - N$ be a splitting. A solution of $Ax = b$ is a *fixed point* to the iteration $x^{(k+1)} = M^{-1} N x^{(k)} + M^{-1} b$.

Example Let $M = D = \operatorname{diag}(A)$ and $N = A - M$. Then $x^{(k)} = D^{-1}(D - A)x^{(k)} + D^{-1} b$ is the Jacobi method.

Lemma The iteration $x^{(k+1)} = Gx^{(k)} + c$ is convergent if $\rho(G) < 1$.

Definition A matrix A is *diagonally dominant* if

$$|a_{ii}| \geq \sum_{j=1, j\neq i}^{n} |a_{ij}|, \qquad i = 1, 2, \ldots, n,$$

with strict inequality for at least one i.

Theorem If A is *diagonally dominant* then the *Jacobi iteration* is convergent.

Remark Matrices obtained by discratizing PDEs are usually diagonally dominant.

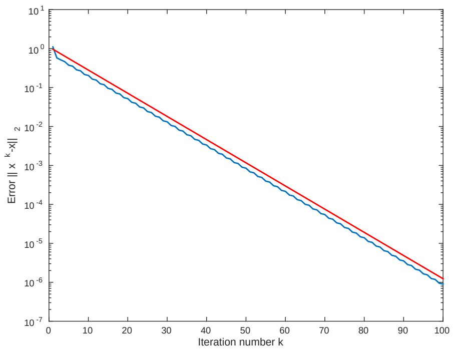
Example - Jacobi Iteration

Create a linear system of equations

```
>> A=[3 1 0 0 ; -1 2 1 0 ; 0 -2 3 1;0 0 -2 2];
>> x=ones(4,1);b=A*x;
>> A
  A =
     3     1     0     0
    -1     2     1     0
     0    -2     3     1
     0     0    -2     2
```

The matrix A is diagonally dominant.

The Jacobi iteraton is $x_{k+1} = Gx_k + c$, where $G = D^{-1}(D - A)$.



The convergence history $\|x^{(k)} - x\|_2$ (blue) for the Jacobi Iteration. Also theoretical convergence curve $\|x^{(0)} - x\|_2 \rho(G)^k$ (red).

Remark This is very slow convergence.

Lemma The *Landweber iteration*,

$$x^{(k+1)} = x^{(k)} + \omega A^T (b - Ax^{(k)}),$$

is *convergent* if $0 < \omega < 2/\sigma_1^2$.

Remark If the Landweber iteration converges then $A^T(b - Ax^*) = 0$ so we have the least squares solution.

The convergence is *linear*. We need faster methods.