Applied Numerical Linear Algebra. Lecture 6

Special Linear Systems

It is important to exploit any special structure of the matrix to increase speed of solution and decrease storage. We will consider only real matrices:

- s.p.d. matrices,
- symmetric indefinite matrices,
- band matrices,
- general sparse matrices,
- dense matrices depending on fewer than n^2 independent parameters.

Example: solution of ODE

Consider the ordinary differential equation (ODE) $y''(x) - p(x)y'(x) - q(x)y(x) = r(x)$ on the interval $[a, b]$ with boundary conditions $y(a) = \alpha$, $y(b) = \beta$. We also assume $q(x) \geq \underline{q} > 0$. This equation may be used to model the heat flow in a long, thin rod, for example. To solve the differential equation numerically, we *discretize* it by seeking its solution only at the evenly spaced mesh points $x_i = a + ih$, $i = 0, \ldots, N+1$, where $h = (b-a)/(N+1)$ is the mesh spacing. Define $p_i = p(x_i)$, $r_i = r(x_i)$, and $q_i = q(x_i)$.

We need to derive equations to solve for our desired approximations $y_i \approx y(x_i)$, where $y_0 = \alpha$ and $y_{N+1} = \beta$. To derive these equations, we approximate the derivative $y\prime(x_i)$ by the following *finite difference approximation*:

$$y\prime(x_i) \approx \frac{y_{i+1} - y_{i-1}}{2h}.$$

(Note that as h gets smaller, the right-hand side approximates $y\prime(x_i)$ more and more accurately.) We can similarly approximate the second derivative by

$$y\prime\prime(x_i) \approx \frac{y_{i+1} - 2y_i + y_{i-1}}{h^2}.$$

Inserting these approximations into the differential equation yields

$$\frac{y_{i+1} - 2y_i + y_{i-1}}{h^2} - p_i \frac{y_{i+1} - y_{i-1}}{2h} - q_i y_i = r_i, \quad 1 \leq i \leq N.$$

Multiplying by $-h^2/2$ we get:

$$-\frac{y_{i+1}}{2} + y_i - \frac{y_{i-1}}{2} + p_i \frac{y_{i+1} h}{4} - p_i \frac{y_{i-1} h}{4} + q_i y_i \frac{h^2}{2} = -\frac{-\, r_i h^2}{2}$$

Rewriting this as a linear system we get $Ay = b$, where

$$y = \begin{bmatrix} y_1 \\ \\ \vdots \\ \\ y_N \end{bmatrix}, \quad b = \frac{-h^2}{2} \begin{bmatrix} r_1 \\ \\ \vdots \\ \\ r_N \end{bmatrix} + \begin{bmatrix} (\frac{1}{2} + \frac{h}{4} p_1)\alpha \\ 0 \\ \vdots \\ 0 \\ (\frac{1}{2} - \frac{h}{4} p_N)\beta \end{bmatrix},$$

and recalling

$$-\frac{y_{i+1}}{2}+y_i-\frac{y_{i-1}}{2}+p_i\frac{y_{i+1}h}{4}-p_i\frac{y_{i-1}h}{4}+q_iy_i\frac{h^2}{2}=-\frac{-\,r_ih^2}{2}$$

we have

$$A=\left[\begin{array}{ccccc} a_1 & -c_1 & & & \\ -b_2 & \ddots & \ddots & & \\ & \ddots & \ddots & -c_{N-1} \\ & & -b_N & a_N \end{array}\right], \qquad \begin{array}{rcl} a_i & = & 1+\frac{h^2}{2}q_i, \\ b_i & = & \frac{1}{2}[1+\frac{h}{2}p_i], \\ c_i & = & \frac{1}{2}[1-\frac{h}{2}p_i]. \end{array}$$

Note that $a_i > 0$ and also $b_i > 0$ and $c_i > 0$ if h is small enough. This is a nonsymmetric *tridiagonal* system to solve for y. We will show how to change it to a symmetric positive definite tridiagonal system, so that we may use *band Cholesky* to solve it.

Choose $D = diag(1, \sqrt{\frac{c_1}{b_2}}, \sqrt{\frac{c_1 c_2}{b_2 b_3}}, \ldots, \sqrt{\frac{c_1 c_2 \cdots c_{N-1}}{b_2 b_3 \cdots b_N}})$. Then we may change $Ay = b$ to $\underbrace{DAD^{-1}}_{\tilde{})(Dy)=Db}$ or $\tilde{A}\tilde{y} = \tilde{b}$, where

$$\tilde{A} = \begin{bmatrix} a_1 & -\sqrt{c_1 b_2} & & & \\ -\sqrt{c_1 b_2} & a_2 & -\sqrt{c_2 b_3} & & \\ & -\sqrt{c_2 b_3} & \ddots & & \\ & \ddots & \ddots & -\sqrt{c_{N-1} b_N} \\ & & -\sqrt{c_{N-1} b_N} & a_N \end{bmatrix}.$$

It is easy to see that $\tilde{A}$ is symmetric, and it has the same eigenvalues as A because A and $\tilde{A} = DAD^{-1}$ are *similar*. We will use the next theorem to show it is also positive definite.

Gershgorin's Theorem

THEOREM (Gershgorin) *Let B be an arbitrary matrix. Then the eigenvalues λ of B are located in the union of the n disks*

$$|\lambda - b_{kk}| \leq \sum_{j \neq k} |b_{kj}|.$$

Proof. Given λ and $x \neq 0$ such that $Bx = \lambda x$, let $1 = ||x||_\infty = x_k$ by scaling x if necessary. Then $\sum_{j=1}^{N} b_{kj} x_j = \lambda x_k = \lambda$, so $\lambda - b_{kk} = \sum^{N}_{\substack{j=1 \\ j \neq k}} b_{kj} x_j$, implying

$$|\lambda - b_{kk}| \leq \sum_{j \neq k} |b_{kj} x_j| \leq |b_{kj}|. \quad \square$$

Examples of using Gershgorin's circle theorem

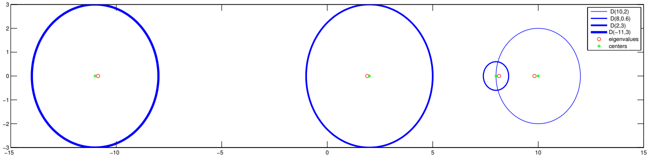
Example 1.
Use the Gershgorin circle theorem to estimate the eigenvalues of

$$A = \begin{bmatrix} 10 & -1 & 0 & 1 \\ 0.2 & 8 & 0.2 & 0.2 \\ 1 & 1 & 2 & 1 \\ -1 & -1 & -1 & -11 \end{bmatrix}.$$

Starting with row one, we take the element on the diagonal, a_{ii} as the center for the disc. We then take the remaining elements in the row and apply the formula:

$$\sum_{j \neq i} |a_{ij}| = R_i$$

to obtain the following four discs:
$D(10, 2), D(8, 0.6), D(2, 3),$ and $D(-11, 3)$
The eigenvalues are: 9.8218, 8.1478, 1.8995, -10.86



Example 1

In example 1 the eigenvalues are: 9.8218, 8.1478, 1.8995, -10.86

Examples of using Gershgorin's circle theorem

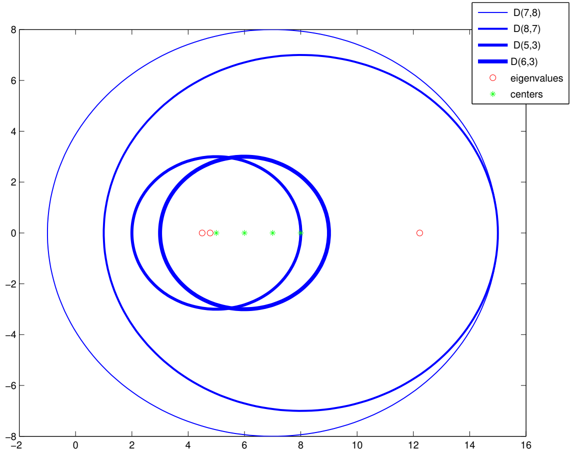
Example 2.
Use the Gershgorin circle theorem to estimate the eigenvalues of

$$A = \begin{bmatrix} 7 & 5 & 2 & 1 \\ 2 & 8 & 3 & 2 \\ 1 & 1 & 5 & 1 \\ 1 & 1 & 1 & 6 \end{bmatrix}.$$

Starting with row one, we take the element on the diagonal, a_{ii} as the center for the disc. We then take the remaining elements in the row and apply the formula:

$$\sum_{j \neq i} |a_{ij}| = R_i$$

to obtain the following four discs:
$D(7,8), D(8,7), D(5,3), D(6,3)$
The eigenvalues are: 12.2249 + 0.0000i; 4.4977 + 0.6132i; 4.4977 - 0.6132i; 4.7797 + 0.0000i;



Example 2

In example 2 the eigenvalues are: 12.2249 + 0.0000i; 4.4977 + 0.6132i; 4.4977 - 0.6132i; 4.7797 + 0.0000i.

Example: ODE (continuation)

- If h is so small that for all i, $|\frac{h}{2}p_i| < 1$, then

  $$|b_i|+|c_i| = \frac{1}{2}\left(1+\frac{h}{2}p_i\right)+\frac{1}{2}\left(1-\frac{h}{2}p_i\right) = 1 < 1+\frac{h^2}{2}\underline{q} \le 1+\frac{h^2}{2}q_i = a_i.$$

  Therefore all eigenvalues of A lie inside the disks centered at $1+h^2q_i/2 \ge 1+h^2\underline{q}/2$ with radius 1; in particular, they must all have positive real parts.
- Since $\tilde{A}$ is symmetric, its eigenvalues are real and hence positive, so $\tilde{A}$ is positive definite. Its smallest eigenvalue is bounded below by $\underline{q}h^2/2$.

Linear Least Squares Problems

- Suppose that we have a matrix A of the size $m \times n$ and the vector b of the size $m \times 1$. The linear least square problem is to find a vector x of the size $n \times 1$ which will minimize $||Ax - b||_2$.
- In the case when $m = n$ and the matrix A is nonsingular we can get solution to this problem as $x = A^{-1}b$.
- When $m > n$ (more equations than unknows) the problem is overdetermined
- When $m < n$ (more unknows than equations) the problem is underdetermined
- Applications: curve fitting, statistical modelling.

Matrix Factorizations that Solve the Linear Least Squares Problem

The linear least squares problem has several explicit solutions that we will discuss:

1. normal equations: the fastest but least accurate; it is adequate when the condition number is small.
2. QR decomposition,

   is the standard one and costs up to twice as much as the first method.
3. SVD,

   is of most use on an ill-conditioned problem, i.e., when A is not of full rank; it is several times more expensive again.
4. transformation to a linear system, lets us do iterative refinement to improve the solution when the problem is ill-conditioned. Can be adapted to deal efficiently with sparse matrices [Å. Björck. Numerical Methods for Least Squares Problems].

We assume initially for methods 1 and 2 that A has full column rank n.

Linear Least Squares Problems

Further we assume that we will deal with overdetermined problems when we have more equations than unknowns. This means that we will be interested in the solution of linear system of equations

$$Ax = b, \tag{1}$$

where A is of the size $m \times n$ with $m > n$, b is vector of the size m, and x is vector of the size n.
In a general case we are not able to get vector b of the size m as a linear combination of the n columns of the matrix A and n components of the vector x, or there is no solution to (1) in the usual case. In this chapter we will consider methods which can minimize the residual $r = b - Ax$ as a function on x in principle in any norm, but we will use 2-norm because of the convenience from theoretical (relationships of 2-norm with the inner product and orthogonality, smoothness and strict convexity properties) and computational points of view. Also, because of using 2-norm method is called least squares.

We can write the least squares problem as problem of the minimizing of the squared residuals

$$\|r\|_2^2 = \sum_{i=1}^{m} r_i^2 = \sum_{i=1}^{m} (Ax_i - b)^2. \tag{2}$$

In other words, our goal is to find minimum of this residual using least squares:

$$\min_x \|r\|_2^2 = \min_x \sum_{i=1}^{m} r_i^2 = \min_x \sum_{i=1}^{m} (Ax_i - b)^2. \tag{3}$$

Normal Equations

Our goal is to minimize the residual $r(x) = ||Ax - b||_2^2$. To find minimum of this functional and derive the *normal equations*, we look for the x where the gradient of $||Ax - b||_2^2 = (Ax - b)^T(Ax - b)$ vanishes, or where $r'(x) = 0$. So we want

$$\begin{aligned} 0 &= \lim_{e \to 0} \frac{(A(x+e) - b)^T(A(x+e) - b) - (Ax - b)^T(Ax - b)}{||e||_2} \\ &= \lim_{e \to 0} \frac{2e^T(A^TAx - A^Tb) + e^TA^TAe}{||e||_2} \end{aligned}$$

The second term $\frac{|e^TA^TAe|}{||e||_2} \le \frac{||A||_2^2||e||_2^2}{||e||_2} = ||A||_2^2||e||_2^2$ approaches 0 as e goes to 0, so the factor $A^TAx - A^Tb$ in the first term must also be zero, or $A^TAx = A^Tb$. This is a system of n linear equations in n unknowns, the normal equations.

Data fitting

In this example we present the typical application of least squares called data or curve fitting problem. This problem appear in statistical modelling and experimental engineering when data are generated by laboratory or other measurements.
Suppose that we have data points $(x_i, y_i), i = 1, ..., m$, and our goal is to find the vector of parameters c of the size n which will fit best to the data y_i of the model function $f(x_i, c)$, where $f : R^{n+1} \to R$, in the least squares sense:

$$\min_c \sum_{i=1}^{m} (y_i - f(x_i, c))^2. \tag{4}$$

If the function $f(x, c)$ is linear then we can solve the problem (4) using least squares method.

The function $f(x,c)$ is linear if we can write it as a linear combination of the functions $\phi_j(x), j=1,...,n$ as:

$$f(x,c) = c_1\phi_1(x) + c_2\phi_2(x) + ... + c_n\phi_n(x). \tag{5}$$

Functions $\phi_j(x), j=1,...,n$ are called basis functions.
Let now the matrix A will have entries
$a_{ij} = \phi_j(x_i), i=1,...,m; j=1,...,n$, and vector b will be such that $b_i = y_i, i=1,...,m$. Then a linear data fitting problem takes the form of (1) with $x=c$:

$$Ac \approx b \tag{6}$$

Elements of the matrix A are created by basis functions $\phi_j(x), j=1,...,n$. We will consider now different examples of choosing basis functions $\phi_j(x), j=1,...,n$.

Problem of the fitting to a polynomial

In the problem of the fitting to a polynomial

$$f(x, c) = \sum_{i=1}^{d} c_i x^{i-1} \tag{7}$$

of degree $d-1$ to data points $(x_i, y_i), i = 1, ..., m$, basis functions $\phi_j(x), j = 1, ..., n$ can be chosen as $\phi_j(x) = x^{j-1}, j = 1, ..., n$. The matrix A constructed by these basis functions in a polynomial fitting problem is a Vandermonde matrix:

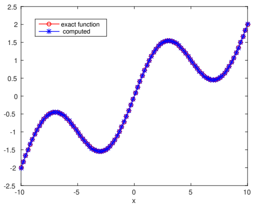
$$A = \begin{bmatrix} 1 & x_1 & x_1^2 & \dots & x_1^{d-1} \\ 1 & x_2 & x_2^2 & \dots & x_2^{d-1} \\ 1 & x_3 & x_3^2 & \dots & x_3^{d-1} \\ \vdots & \vdots & \ddots & \ddots & \vdots \\ 1 & x_m & x_m^2 & \dots & x_m^{d-1} \end{bmatrix}. \tag{8}$$

Here, $x_i, i = 1, ..., m$ are discrete points on the interval for $x = [x_{left}, x_{right}]$.

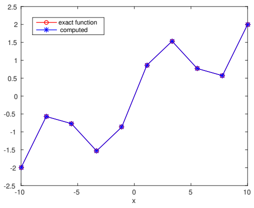
Suppose, that we choose $d = 4$ in (4). Then we can write the polynomial as $f(x, c) = \sum_{i=1}^{4} c_i x^{i-1} = c_1 + c_2 x + c_3 x^2 + c_4 x^3$ and our data fitting problem (6) for this polynomial takes the form

$$\begin{bmatrix} 1 & x_1 & x_1^2 & x_1^3 \\ 1 & x_2 & x_2^2 & x_2^3 \\ 1 & x_3 & x_3^2 & x_3^3 \\ \vdots & \vdots & \ddots & \vdots \\ 1 & x_m & x_m^2 & x_m^3 \end{bmatrix} \cdot \begin{bmatrix} c_1 \\ c_2 \\ c_3 \\ c_4 \end{bmatrix} = \begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ \dots \\ b_m \end{bmatrix}. \tag{9}$$

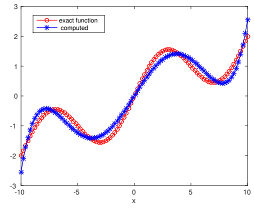
The right hand side of the above system represents measurements or function which we want to fit. Our goal is to find such coefficients $c = \{c_1, c_2, c_3, c_4\}$ which will minimize the residual $r_i = f(x_i, c) - b_i, i = 1..., m$. Since we want minimize squared 2-norm of the residual, or $\|r\|_2^2 = \sum_{i=1}^m r_i^2$, then we will solve the linear least squares problem.
Let us consider an example when the right hand side $b_i, i = 1, ...m$ is taken as a smooth function $b = sin(\pi x/5) + x/5$. Figure on the next slide shows polynomial fitting to the function $b = sin(\pi x/5) + x/5$ for different d in (7) on the interval $x \in [-10, 10]$. Using this figure we observe that with increasing of the degree of the polynomial $d - 1$ we have better fit to the exact function $b = sin(\pi x/5) + x/5$. However, for the degree of the polynomial more than 18 we get erratic fit to the function. This happens because matrix A becomes more and more ill-conditioned with increasing of the degree of the polynomial d. And this is, in turn, because of the linear dependence of the columns in the Vandermonde's matrix A.



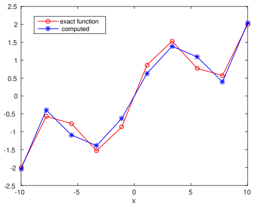
a) d=10



b) d=10



c) d=5



d) d=5

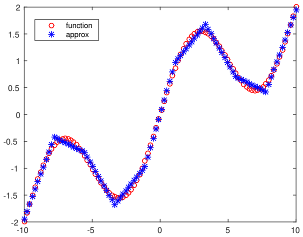
Figure: *Polynomial fitting for different d in (7) to the function* $b = \sin(\pi x/5) + x/5$ *on the interval* $x \in [-10, 10]$ *using the method of normal equations. On the left figures: fit to the 100 points* x_i, $i = 1, \ldots, 100$; *on the right figures: fit to the 10 points* x_i, $i = 1, \ldots, 10$. *Lines with blue stars represent computed function and with red circles - exact one.*

Approximation using linear splines

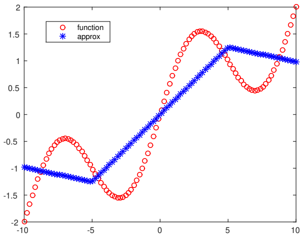
When we want to solve the problem (4) of the approximation to the data vector $y_i, i = 1, ..., m$ with linear splines we use following basis functions $\phi_j(x), j = 1, ..., n$, in (5) which are called also hat functions:

$$\phi_j(x) = \begin{cases} \frac{x - T_{j-1}}{T_j - T_{j-1}}, & T_{j-1} \leq x \leq T_j, \\ \frac{T_{j+1-x}}{T_{j+1} - T_j}, & T_j \leq x \leq T_{j+1}. \end{cases} \tag{10}$$

Here, the column j in the matrix A is constructed by the given values of $\phi_j(x)$ at points $T_j, j = 1, .., n$, which are called conjunction points and are chosen by the user. Using (10) we can conclude that the first basis function is $\phi_1(x) = \frac{T_2 - x}{T_2 - T_1}$ and the last one is $\phi_n(x) = \frac{x - T_{n-1}}{T_n - T_{n-1}}$.
Figure on the next slide shows approximation of a function $b = sin(\pi x/5) + x/5$ on the interval $x \in [-10, 10]$ using linear splines with different number n of conjunction points $T_j, j = 1, ..., n$.



a) n=10



b) n=5

Figure: *Polynomial fitting to the function $b = sin(\pi x/5) + x/5$ on the interval $x \in [-10, 10]$ using linear splines with different number n of conjunction points $T_j, j = 1, \dots, n$ in (10). Blue stars represent computed function and red circles - exact one.*

Approximation using bellsplines

In the case when we want to solve the problem (4) using bellsplines, the number of bellsplines which can be constructed are $n+2$, and the function $f(x,c)$ in (4) is written as

$$f(x,c) = c_1\phi_1(x) + c_2\phi_2(x) + ... + c_{n+2}\phi_{n+2}(x). \tag{11}$$

We define

$$\phi_j^0(x) = \begin{cases} 1, & T_j \le x \le T_{j+1}, \\ 0, & \text{otherwise.} \end{cases} \tag{12}$$

Then all other basis functions, or bellsplines, $\phi_j^k(x), j = 1, ..., n+2; k = 1, 2, 3$ are defined as follows:

$$\phi_j^k(x) = (x - T_k)\frac{\phi_j^{k-1}(x)}{T_{j+k} - T_j} + (T_{j+k+1} - x)\frac{\phi_{j+1}^{k-1}(x)}{T_{j+k+1} - T_{j+1}}. \tag{13}$$

Here, the column j in the matrix A is constructed by the given values of $\phi_j(x)$ at conjunction points $T_j, j = 1, .., n$ which are chosen by the user. If in (13) we obtain ratio $0/0$, then we assign $\phi_j^k(x) = 0$. We define additional three points T_{-2}, T_{-1}, T_0 at the left side of the input interval as $T_{-2} = T_{-1} = T_0 = T_1$, and correspondingly three points T_{n+1}, T_{n+2}, T_{n+3} on the right side of the interval as $T_n = T_{n+1} = T_{n+2} = T_{n+3}$. All together we have $n+6$ conjunction points $T_j, j = 1, ..., n+6$. Number of bellsplines which can be constructed are $n+2$.

If conjunction points T_j are distributed uniformly, then we can introduce the mesh size $h = T_{k+1} - T_k$ and bellsplines can be written explicitly as

$$\phi_j(x) = \begin{cases} \frac{1}{6}t^3 & \text{if } T_{j-2} \leq x \leq T_{j-1},\ t = \frac{1}{h}(x - T_{j-2}), \\ \frac{1}{6} + \frac{1}{2}(t + t^2 - t^3) & \text{if } T_{j-1} \leq x \leq T_j,\ t = \frac{1}{h}(x - T_{j-1}), \\ \frac{1}{6} + \frac{1}{2}(t + t^2 - t^3) & \text{if } T_j \leq x \leq T_{j+1},\ t = \frac{1}{h}(T_{j+1} - x), \\ \frac{1}{6}t^3 & \text{if } T_{j+1} \leq x \leq T_{j+2}, t = \frac{1}{h}(T_{j+2} - x). \end{cases} \tag{14}$$

In the case of uniformly distributed bellsplines we place additional points at the left side of the input interval as
$T_0 = T_1 - h, T_{-1} = T_1 - 2h, T_{-2} T_1 - 3h$, and correspondingly on the right side of the interval as
$T_{n+1} = T_n + h, T_{n+2} = T_n + 2h, T_{n+3} = T_n + 3h$. Then the function $f(x, c)$ in (4) will be the following linear combination of $n + 2$ functions $\phi_j(x)$ for indices $j = 0, 1, ..., n + 1$:

$$f(x, c) = c_1\phi_0(x) + c_2\phi_1(x) + ... + c_{n+2}\phi_{n+1}(x). \tag{15}$$

Figure on the next slide shows approximation of a function $b = sin(\pi x/5) + x/5$ on the interval $x \in [-10, 10]$ using bellsplines.

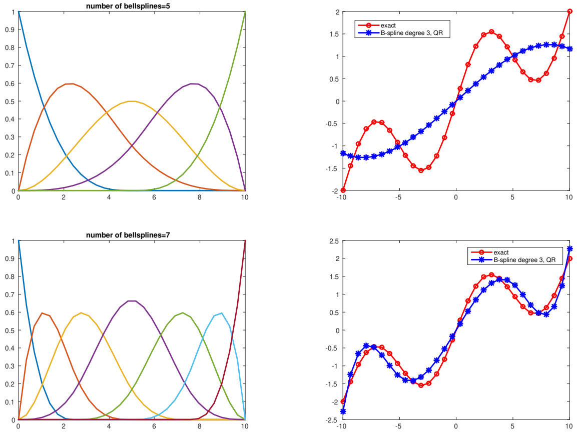


Figure: *Polynomial fitting to the function $b = sin(\pi x/5) + x/5$ on the interval $x \in [-10, 10]$ with different number of bellsplines. Blue stars represent computed function and red circles - exact one.*

Nonlinear least squares problems

Suppose that for our data points $(x_i, y_i), i = 1, ..., m$ we want to find the vector of parameters $c = (c_1, ..., c_n)$ which will fit best to the data $y_i, i = 1, ..., m$ of the model function $f(x_i, c), i = 1, ..., m$. We consider the case when the model function $f : R^{n+1} \to R$ is nonlinear now. Our goal is to find minimum of the residual $r = y - f(x, c)$ in the least squares sense:

$$\min_c \sum_{i=1}^{m} (y_i - f(x_i, c))^2. \tag{16}$$

To solve problem (16) we can still use the linear least squares method if we can transform the nonlinear function $f(x, c)$ to the linear one. This can be done if the function $f(x, c)$ can be represented in the form $f(x, c) = A \exp^{cx}, A = const$. Then taking logarithm of $f(x, c)$ we get: $\ln f = \ln A + cx$, which is already linear function. Then linear least squares problem after this transformation can be written as

$$\min_c \sum_{i=1}^{m} (\ln y_i - \ln f(x_i, c))^2. \tag{17}$$

Another possibility how to deal with nonlinearity is consider the least squares problem as an optimization problem. Let us define the residual $r : R^n \to R^m$ as

$$r_i(c) = y_i - f(x_i, c), \quad i = 1, ..., m. \tag{18}$$

Our goal is now minimize the function

$$F(c) = \frac{1}{2} r(c)^T r(c) = \frac{1}{2} \|r(c)\|_2^2. \tag{19}$$

To find minimum of (19) we should have

$$\nabla F(c) = \frac{\partial F(c)}{\partial c_i} = 0, \quad i = 1, ..., m. \tag{20}$$

Direct computations show that the gradient vector $\nabla F(c)$ is

$$\nabla F(c) = \frac{dF}{dc} = J^T(c) r(c), \tag{21}$$

where J^T is the transposed Jacobian matrix of the residual $r(c)$.

For a sufficiently smooth function $F(c)$ we can write its Taylor expansion as

$$F(c) = F(c_0) + \nabla F(c_0)(c - c_0) + O(h^2), \tag{22}$$

with $|h| = \|c - c_0\|$. Since our goal is to find minimum of $F(c)$, then at a minimum point c^* we should have $\nabla F(c^*) = 0$. Taking derivative with respect to c from (22) we obtain

$$H(F(c_0))(c - c_0) + \nabla F(c_0) = 0, \tag{23}$$

where H denotes the Hessian matrix of the function $F(c_0)$.

Using (21) in (23) we obtain

$$H(F(c_0))(c - c_0) + J^T(c_0)r(c_0) = 0, \tag{24}$$

and from this expression we observe that we have obtained a system of linear equations

$$H(F(c_0))(c - c_0) = -J^T(c_0)r(c_0) \tag{25}$$

which can be solved again using linear least squares method. The Hessian matrix $H(F(c_0))$ can be obtained from (21) as

$$H(F(c_0)) = J^T(c_0)J(c_0) + \sum_{i=1}^{m} r_i(c_0)H(r_i), \tag{26}$$

where $H(r_i)$ denotes the Hessian matrix of the residual function $r_i(c)$. These m matrices $H(r_i)$ are inconvenient to compute, but since they are multiplied to the small residuals $r_i(c_0)$, the second term in (26) is often very small at the solution c_0 and this term can be dropped out.

Then the system is transformed to the following linear system

$$J^T(c_0)J(c_0)(c - c_0) = -J^T(c_0)r(c_0), \tag{27}$$

which actually is a system of normal equations for the $m \times n$ linear least squares problem

$$J(c_0)(c - c_0) = -r(c_0). \tag{28}$$

The system (27) determines the Gauss-Newton method for the solution of the least squares problem as an iterative process

$$c_{k+1} = c_k - [J^T(c_k)J(c_k)]^{-1}J^T(c_k)r(c_k), \tag{29}$$

where k is the number of iteration.

An alternative to the Gauss-Newton method is Levenberg-Marquardt method. This method is based on the finding of minimum of the regularized function

$$F(c) = \frac{1}{2} r(c)^T r(c) + \frac{1}{2}\gamma(c - c_0)^T(c - c_0) = \frac{1}{2}\|r(c)\|_2^2 + \frac{1}{2}\gamma\|c - c_0\|_2^2, \tag{30}$$

where c_0 is a good initial guess for c and γ is a small regularization parameter. Then we repeat all steps which we have performed for the obtaining the Gauss-Newton method, see (21)-(26).

Finally, In the Levenberg-Marquardt method the linear system which should be solved at every iteration k is

$$(J^T(c_k)J(c_k) + \gamma_k I)(c_{k+1} - c_k) = -J^T(c_k)r(c_k), \tag{31}$$

and the corresponding linear least squares problem is

$$\begin{bmatrix} J(c_k) \\ \sqrt{\gamma_k} I \end{bmatrix} \cdot (c_{k+1} - c_k) \approx \begin{bmatrix} -r(c_k) \\ 0 \end{bmatrix}. \tag{32}$$

Example 1

Let us consider the nonlinear model equation

$$Ae^{\mathrm{E/T-T_0}} = \mathrm{y}. \tag{33}$$

Our goal is to determine parameters A, E and T_0 in this equation by knowing y and T. We rewrite (33) as a nonlinear least squares problem in the form

$$\min_{A,E,T_0} \sum_{i=1}^{m} (y_i - Ae^{\mathrm{E/T_i-T_0}})^2. \tag{34}$$

We will show how to obtain from the nonlinear problem (34) the linear one. We take logarithm of (33) to get

$$\ln A + \frac{E}{T - T_0} = \ln y. \tag{35}$$

Now multiply both sides of (35) by $T - T_0$ to obtain:

$$\ln A(T - T_0) + E = \ln y(T - T_0). \tag{36}$$

and rewrite the above equation as

$$T \ln A - T_0 \ln A + E + T_0 \ln y = T \ln y. \tag{37}$$

Let now define the vector of parameters $c = (c_1, c_2, c_3)$ with $c_1 = T_0, c_2 = \ln A, c_3 = E - T_0 \ln A$. Now the problem (37) can be written as

$$c_1 \ln y + c_2 T + c_3 = T \ln y, \tag{38}$$

which is already a linear problem. Now we can rewrite (38) denoting by $f(c, y, T) = c_1 \ln y + c_2 T + c_3$ as a linear least squares problem in the form

$$\min_c \sum_{i=1}^{m} (T_i \ln y_i - f(c, y_i, T_i))^2. \tag{39}$$

The system of linear equations which is needed to be solved is

$$\begin{bmatrix} \ln y_1 & T_1 & 1 \\ \ln y_2 & T_2 & 1 \\ \vdots & \vdots & \vdots \\ \ln y_m & T_m & 1 \end{bmatrix} \cdot \begin{bmatrix} c_1 \\ c_2 \\ c_3 \end{bmatrix} = \begin{bmatrix} T_1 \ln y_1 \\ T_2 \ln y_2 \\ \vdots \\ T_m \ln y_m \end{bmatrix} \tag{40}$$

Example 2

Suppose that the nonlinear model function is given as

$$f(x, c) = A\mathrm{e}^{\mathrm{c_1 x}} + \mathrm{Be}^{\mathrm{c_2 x}}, \quad \mathrm{A, B = const.} > 0, \tag{41}$$

and our goal is to fit this function using Gauss-Newton method. In other words, we will use iterative formula (28) for iterative update of $c = (c_1, c_2)$. The residual function will be

$$r(c) = y - f(x, c), \tag{42}$$

where $y = y_i, i = 1, ..., m$ are data points.

First, we compute Jacobian matrix $J(c)$, where two columns in this matrix will be given by

$$\begin{aligned} J(c)_{i,1} &= \frac{\partial r_i}{\partial c_1} = -x_i A \mathrm{e}^{\mathrm{c_1 x_i}}, \quad \mathrm{i = 1, ..., m}, \\ J(c)_{i,2} &= \frac{\partial r_i}{\partial c_2} = -x_i B \mathrm{e}^{\mathrm{c_2 x_i}}, \quad \mathrm{i = 1, ..., m}. \end{aligned} \tag{43}$$

If we will take initial guess for the parameters $c_0 = (c_1, c_2)_0 = (1, 0)$, then we have to solve the following problem at iteration $k = 1$:

$$J(c_0)(c_1 - c_0) = -r(c_0), \tag{44}$$

and the next update for parameters $c_1 = (c_1, c_2)_1$ in the Gauss-Newton method can be computed as

$$c_1 = c_0 - [J^T(c_0)J(c_0)]^{-1} J^T(c_0) r(c_0). \tag{45}$$

Here, $r(c_0)$ and $J(c_0)$ can be computed explicitly as follows:

$$r(c_0) = y_i - f(x_i, c_0) = y_i - (A\mathrm{e}^{1\cdot \mathrm{x_i}} + \mathrm{B}\mathrm{e}^{0\cdot \mathrm{x_i}}) = \mathrm{y_i} - \mathrm{A}\mathrm{e}^{\mathrm{x_i}} - \mathrm{B}, \mathrm{i} = 1, ..., \mathrm{m}, \tag{46}$$

and noting that $c_0 = (c_1, c_2)_0 = (1, 0)$ two columns in the Jacobian matrix $J(c_0)$ will be

$$\begin{aligned} J(c_0)_{i,1} &= -x_i A\mathrm{e}^{1\cdot \mathrm{x_i}} = -\mathrm{x_i}\mathrm{A}\mathrm{e}^{\mathrm{x_i}}, \quad \mathrm{i} = 1, ..., \mathrm{m}, \\ J(c_0)_{i,2} &= -x_i B\mathrm{e}^{0\cdot \mathrm{x_i}} = -\mathrm{x_i}\mathrm{B}, \quad \mathrm{i} = 1, ..., \mathrm{m}. \end{aligned} \tag{47}$$

Substituting (46), (47) into (44) yields following linear system of equations

$$\begin{bmatrix} -x_1 A\mathrm{e}^{\mathrm{x_1}} & -x_1 B \\ -x_1 A\mathrm{e}^{\mathrm{x_2}} & -x_2 B \\ \vdots & \vdots \\ -x_m A\mathrm{e}^{\mathrm{x_m}} & -x_m B \end{bmatrix} \cdot \begin{bmatrix} (c_1)_1 - (c_1)_0 \\ (c_2)_1 - (c_2)_0 \end{bmatrix} = - \begin{bmatrix} y_1 - A\mathrm{e}^{\mathrm{x_1}} - \mathrm{B} \\ y_2 - A\mathrm{e}^{\mathrm{x_2}} - \mathrm{B} \\ \vdots \\ y_m - A\mathrm{e}^{\mathrm{x_m}} - \mathrm{B} \end{bmatrix} \qquad (48)$$

which is solved for $c_1 - c_0$ using method of normal equations as

$$\begin{bmatrix} -x_1 A\mathrm{e}^{\mathrm{x}_1} & -x_1 B \\ -x_1 A\mathrm{e}^{\mathrm{x}_2} & -x_2 B \\ \vdots & \vdots \\ -x_m A\mathrm{e}^{\mathrm{x}_\mathrm{m}} & -x_m B \end{bmatrix}^T \cdot \begin{bmatrix} -x_1 A\mathrm{e}^{\mathrm{x}_1} & -x_1 B \\ -x_1 A\mathrm{e}^{\mathrm{x}_2} & -x_2 B \\ \vdots & \vdots \\ -x_m A\mathrm{e}^{\mathrm{x}_\mathrm{m}} & -x_m B \end{bmatrix} \cdot \begin{bmatrix} (c_1)_1 - (c_1)_0 \\ (c_2)_1 - (c_2)_0 \end{bmatrix}$$
$$= - \begin{bmatrix} -x_1 A\mathrm{e}^{\mathrm{x}_1} & -x_1 B \\ -x_1 A\mathrm{e}^{\mathrm{x}_2} & -x_2 B \\ \vdots & \vdots \\ -x_m A\mathrm{e}^{\mathrm{x}_\mathrm{m}} & -x_m B \end{bmatrix}^T \cdot \begin{bmatrix} y_1 - A\mathrm{e}^{\mathrm{x}_1} - \mathrm{B} \\ y_2 - A\mathrm{e}^{\mathrm{x}_2} - \mathrm{B} \\ \vdots \\ y_m - A\mathrm{e}^{\mathrm{x}_\mathrm{m}} - \mathrm{B} \end{bmatrix} \quad (49)$$

This system can be solved for $c_1 - c_0$, and next values c_1 are obtained by using (45).