A Simple Matrix Alteration Method

Thomas Wieder

http://orcid.org/0000-0001-9390-468X
https://www.researchgate.net/profile/Thomas_Wieder

Abstract

We use a well known formula given by C.G. Broyden for the solution of an equation system in a slightly different manner. We alter the given matrix $J$ of a given linear matrix equation $y_0 = Jx_0$ into a matrix $H$ of same size such that for fixed $x_0$ we receive a desired result $y_1 = Hx_0$ different from $y_0$ and that the difference $\|H - J\|_2$ is minimal in the Euclidian norm. Some examples for the application of this simple matrix alteration method are provided.

Mathematics Subject Classification: 15A24, 15A283

Keywords: matrix alteration, linear system, input-output model

1 Introduction

Consider a linear equation system in matrix form like

$$y = Jx$$  \hspace{1cm} (1)

where $x$ and $y$ are column vectors of length $n$ and $m$ and $J$ is a $m \times n$ matrix. The elements of $x$, $y$ and $J$ are designated as $x(k)$, $y(l)$ and $j(k,l)$ with $k = 1 \ldots n, l = 1 \ldots m$. Here we assume that $x(k)$, $y(l)$ and $j(k,l)$ are real numbers. As well known, eq. (1) offers two common ways for evaluation. If we are given with the input vector $x$ and coefficient matrix $J$, then we can calculate from them the output vector $y$, this is the easy forward direction. Or we are given with the output vector $y$ and $J$, then we can solve for $x$, this is the more difficult backward direction. In the following we will concentrate on a third direction, that is the alteration of $J$ into another matrix $H$. 
At first we want to give a motivation for the third direction. Let us think of a so-called complex system $S$ in broad terms. $S$ may take in some input $x$ and converts it via $J$ to some output $y$. We now assume that in the $y$-space some region exists which is desirable for certain reasons. That is, we or $S$ itself want to get $y$ to lie in that advantageous region. Now let us map this broad idea to eq. (1). We are given with a concrete input vector $x_0$ which results according to eq. (1) in an output vector $y_0$

$$y_0 = Jx_0.$$  

For fixed $x_0$ we want to receive $y_1$ instead of $y_0$ (since $y_1$ lies in the favorable region but $y_0$ in an unfavourable one). Then we have to alter $J$, that is its elements $j(k,l)$, accordingly. We thus arrive at a replacement of matrix $J$ by a new matrix $H$ and we get the new equation

$$y_1 = Hx_0$$

where we still utilize the same input $x_0$ as in eq. (2). Now, if we change from $J$ to $H$, then we can put some conditions on $H$. Imposing additional conditions on a solution of a matrix equation (when working in the backward direction) is a common approach and is called matrix regularization. We demand $H$ to be as close as possible to $J$. This request must be justified for the particular $S$ under consideration but appears to be reasonable generally and offers a method for the determination of $H$.

2 Main Results - Broyden’s formula

Centuries ago C.G. Broyden [1, 2] derived a class of methods to iteratively calculate the solution vector $x$ of a non-linear equation system.\footnote{The referees of a previous version of this paper rightly pointed me to Broyden’s formula.} In this section we continue our reasoning and arrive at one of Broyden’s formulas, that is eq. 9. The only difference is in the use of eq. (9), we utilize the formula in order to alter the matrix in (2) from $J$ into $H$.

From our reasoning we now are in the position to derive a formula which returns $H$ from given $x_0$, $y_0$, $y_1$ and $J$. We are looking for a solution for $H$ which fulfils eq. (3) and which minimizes the difference $\|H - J\|^2_2$ in the Euclidean norm (also called 2-norm). For that purpose, we use the Lagrange multiplier method as described in many textbooks and lecture notes on linear algebra in the context of least squares, see for example [3, 4]. Thus, we define the Lagrangian

$$L(H, \lambda) = \lambda^\top(y_1 - Hx_0) + \frac{1}{2}\|H - J\|^2_2$$\hspace{1cm}(4)
where \( \lambda \) is the column vector with the \( m \) Lagrangian multipliers \( \lambda(l) \) and \( \lambda^\top \) is its transposed, i.e. a row vector. For the following step we need from matrix calculus that \( \delta \lambda^\top H x_0 / \delta H = \lambda x_0^\top \). It is also helpful to see that \( x_0 x_0^\top \) is a matrix while \( x_0^\top x_0 \) is a scalar. Finally, we assume that \( x_0 \neq 0 \).

Then the first derivative of eq.(4) with respect to \( H \) is

\[
\frac{\delta L(H, \lambda)}{\delta H} = (H - J) - \lambda x_0^\top.
\]

(5)

Setting eq. (5) equal to zero we get

\[
0 = (H - J) - \lambda x_0^\top = (H - J)x_0 - \lambda x_0^\top x_0
\]

(6)

by right multiplication with \( x_0 \). That is, we have

\[
\lambda x_0^\top x_0 = (H - J)x_0.
\]

(7)

Solving for \( \lambda \) then results in

\[
\lambda = \frac{(H - J)x_0}{x_0^\top x_0} = \frac{y_1 - y_0}{x_0^\top x_0}
\]

(8)

were we used eq. (3) and eq. (2) in the last step. From eq. (6) we also get our final expression for \( H \) from which together with eq. (8) follows

\[
H = J + \lambda x_0^\top = J + \frac{(y_1 - y_0)x_0^\top}{x_0^\top x_0}.
\]

(9)

Finally, multiplying eq. (9) by \( x_0 \) results in our expression for \( y_1 \) as

\[
y_1 = Hx_0 = (J + \frac{(y_1 - y_0)x_0^\top}{x_0^\top x_0})x_0.
\]

(10)

We designate the application of eq. (9) as "matrix alteration" since we alter \( J \) to \( H \). In addition to condition \( \| J - H \|_2^2 = \min \) we are free to introduce further restrictions. For example, we could consider a matrix containing only integer elements and request that the absolute sum over its elements remains constant. We even could keep all elements unchanged but just reshuffle them within the matrix. In this manner we arrive at a bunch of variants for matrix alteration which could be of interest in their own.\(^2\)

\(^2\)The matrix alteration according to eq. (9) can also be applied to the linear equation

\[
y_0 = Jx_0 + u_0.
\]

(11)

with constant column vector \( u \) of length \( n \). We just subtract \( u_0 \) from both sides of eq. (11) and yield eq. (2) again.
3 Application Examples

Our Python program CALC_MAT_H evaluates eq. (9) and is given in the appendix.

3.1 Example: Input-Output Analysis

From quantitative economics we can take a prominent subject as our first example, the so-called Input-Output Analysis [5]. This analysis plays a role in production planning and in the analysis of the production of an economy. Let us look at an economy composed of $n$ sectors with their vector of total output $\mathbf{o}$, a column vector of $n$ rows. These sectors receive input from $m$ sources. In particular, the sectors belong to the sources themselves. Thus, each sector produces from its input some output which partially enters the production as input to the sectors or leaves the production process as produced goods which cover the demand $\mathbf{d}$. These input-output relations are summarized in the so-called input-output table $\mathbf{A}$ with $k = 1, \ldots, n$ and $l = 1, \ldots, m$. Its element $a(k, l)$, called input-output coefficients, reflect the input from sector $k$ to sector $l$. Let us restrict to the case that the only sources are the mentioned sectors itself and thus $m = n$ for simplicity.

In quantitative economy the following model (Leontief model) for the input-output analysis is well known:

\[
\mathbf{o} = \mathbf{A}\mathbf{o} + \mathbf{d}
\]

from which one derives that

\[
\mathbf{o} = (\mathbf{I} - \mathbf{A})^{-1}\mathbf{d}
\]

with $\mathbf{I}$ the identity matrix and $(\mathbf{I} - \mathbf{A})^{-1}$ as the so-called input-output inverse.

In our notation $(\mathbf{I} - \mathbf{A})^{-1} = \mathbf{J}$, $\mathbf{o} = \mathbf{y}$, $\mathbf{d} = \mathbf{x}$. The following example is taken from [5]

\[
\mathbf{y}_0 = \mathbf{J}\mathbf{x}_0 \equiv \begin{pmatrix} 100 \\ 400 \\ 500 \end{pmatrix} = \begin{bmatrix} 1,489 & 0,319 & 0,053 \\ 0,425 & 2,234 & 0,372 \\ 0,638 & 0,851 & 1,808 \end{bmatrix} \begin{pmatrix} 30 \\ 140 \\ 200 \end{pmatrix}. \tag{14}
\]

We now demand that for fixed $\mathbf{x}_0^\top = (30, 140, 200)$ we will arrive at $\mathbf{y}_1^\top = (200, 300, 600)$. Then $\mathbf{J}$ must be altered to $\mathbf{H}$ which we calculate by aid of eq. (9) and get

\[
\mathbf{y}_1 = \mathbf{H}\mathbf{x}_0 \equiv \begin{pmatrix} 200 \\ 300 \\ 600 \end{pmatrix} = \begin{bmatrix} 1,538 & 0,550 & 0,383 \\ 0,375 & 2,002 & 0,041 \\ 0,687 & 1,082 & 2,139 \end{bmatrix} \begin{pmatrix} 30 \\ 140 \\ 200 \end{pmatrix}. \tag{15}
\]
Our aim was to fulfill a higher demand $d = y_1$ which we achieved by boosting the economical indices expressed in matrix $H$ but it is a totally different question how to boost the economy itself in order to arrive at these altered indices.

### 3.2 Example: Tank system

The tank problem taken from [6] is a good demonstration for a simple inhomogeneous linear differential equation system with initial conditions. Let us imagine a water tank system consisting of two connected tanks with water flowing during time $t$. There is an inlet to the first tank, a connection to the second tank and finally an outlet from the second tank. The constant water supply to tank 1 via inlet is $u(1)$ volume units per time unit (vu/tu). Matrix element $j(1,1)$ corresponds to the contribution of the water height $x(1)$ to its change over time $dx(1)/dt$. If the outflow from tank 1 to tank 2 has a capacity $-\alpha$, then $j(1,1) = -\alpha$. Correspondingly, $j(1,2)$ relates to the contribution of the water height $x(2)$ to the change over time $dx(1)/dt$. Since no water flows from tank 2 to tank 1 we have $j(1,2) = 0$. Similarly we have $j(2,1) = \alpha$ taking into account the intake of tank 2 from tank 1. Also, if the outlet has capacity $-\beta$, then $j(2,2) = -\beta$. In the initial set-up we have water supply to the first tank only of 1 vu/tu and thus $u(1) = 1$ and $u(2) = 0$. Then the tank problem reads as

$$\frac{dx(1,t)}{dt} = -\alpha x(1,t) + 0 x(2,t) + u(1) \quad (16)$$

$$\frac{dx(2,t)}{dt} = \alpha x(1,t) - \beta x(2,t) + 0. \quad (17)$$

In order to carry out numerical experiments, let us put $-\alpha = -1$, $-\beta = -1$ along with $x_0(1) = 2$ and $x_0(2) = 2$. As the solution of the differential equation system eq. (16, 17) we gain (see figure (1))

$$x(1,t) = \exp(-t) + 1 \quad (18)$$

$$x(2,t) = t \exp(-t) + \exp(-t) + 1. \quad (19)$$

Furthermore let us calculate $dx(1,t)/dt = y_0(1)$ and $dx(2,t)/dt = y_0(2)$ from eqs. (16, 17). We arrive at $y_0^\top = (-1, 0)$ since

$$y_0 = Jx_0 + u_0 \equiv \begin{pmatrix} -1 \\ 0 \end{pmatrix} = \begin{bmatrix} -1 & 0 \\ 1 & -1 \end{bmatrix} \begin{pmatrix} 2 \\ 2 \end{pmatrix} + \begin{pmatrix} 0 \\ 0 \end{pmatrix}. \quad (20)$$

After all these preparations we now request the water tank system to adopt to the requirement $y_1^\top = (-2, 0)$ with fixed $x_0^\top = (2, 2)$ as before. From eq. (9) and eq. (10) we receive

$$y_1 = Hx_0 + u_0 \equiv \begin{pmatrix} -2 \\ 0 \end{pmatrix} = \begin{bmatrix} -1.25 & -0.25 \\ 1 & -1 \end{bmatrix} \begin{pmatrix} 2 \\ 2 \end{pmatrix} + \begin{pmatrix} 1 \\ 0 \end{pmatrix}. \quad (21)$$
We solve the resulting differential equation system
\[
\begin{align*}
\frac{dx(1,t)}{dt} &= -1.25 x(1,t) - 0.25 x(2,t) + 1 \\
\frac{dx(2,t)}{dt} &= 1 x(1,t) - 1 x(2,t) + 0
\end{align*}
\]
and get a new solution
\[
\begin{align*}
x(1,t) &= \frac{2}{3} + \exp \left( -\frac{9t}{8} \right) \left( -4\sqrt{15} \sin \left( \frac{\sqrt{15}t}{8} \right) + 4 \cos \left( \frac{\sqrt{15}t}{8} \right) \right) \\
x(2,t) &= \frac{2}{3} + \exp \left( -\frac{9t}{8} \right) \left( 4\sqrt{15} \sin \left( \frac{\sqrt{15}t}{8} \right) + 4 \cos \left( \frac{\sqrt{15}t}{8} \right) \right)
\end{align*}
\]
as depicted in figure (1). How do we interpret $H$? Tank 1 increases its delivery to tank 2 by 0.25 but now tank 2 feeds tank 1 by 0.25 vu/tu. Since $-(0.25 + 0.25) \times 2 = -1$ tank 1 can provide 1 additional vu/tu to tank 2 as required.

Figure 1: Tank system, $t =$ time, $x(t) =$ water height.

### 3.3 Example: Spring mass system

We now turn to a homogeneous linear differential equation of second order as our third example for the application of eq. (9). The one-dimensional spring
mass system performs harmonic oscillations described by the equation of motion \( my'' + ky' = 0 \) where \( y = y(t) \) is the elongation of the mass \( m \) during time \( t \). For the following, we set the spring constant \( k \) and the mass \( m \) both equal to one. As well known, this differential equation of second order can be converted into a system of two differential equations of first order [6]. Let be \( x(1, t) = y \) and \( x'(2, t) = y' \); then one obtains

\[
y = Jx \equiv \begin{pmatrix} \frac{dx(1, t)}{dt} \\ \frac{dx(2, t)}{dt} \end{pmatrix} = \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix} \begin{pmatrix} x(1, t) \\ x(2, t) \end{pmatrix}.
\]

(26)

We can now look for some initial conditions on \( x(1, t) \) and \( x(2, t) \). If we set \( x(1, t=0) = 0 \) and \( x(2, t=0) = 0 \), then from eq. (26) we receive \( dx(1, t)/dt = 0 \) and \( dx(2, t)/dt = 0 \) and from that the solution to eq. (26) is for all \( t \)

\[
x(1, t) = 0 \quad \text{and} \quad x(2, t) = 0.
\]

(27)

(28)

While this initial state is boring (the mass does not move), still this is an interesting situation for control theory. In control theory, the system is brought from an initial state to a desired final state by aid of a state transition matrix. In the present case, one wants the system state to be different from zero and therefore one imposes for some time \( t = T \) new conditions on eq. (26). For example one may set \( x(1, t = T) = 1/2 \) and \( x(2, t = T) = 1/2 \) and [6] outlines the further procedure to calculate from the state transition matrix the steering control = external input to the system which shifts the system into the final state. Our approach is different and consequently we arrive at a complete different solution. First of all, we can not apply the matrix alteration method eq. (9) for \( x_0 = 0 \) but for a periodic system we may translate from point \( t = 0 \) to point \( t = 2\pi \) and we still get the same solution for eq. (26). We put \( T = 2\pi \) and for \( x(1, t = T = 2\pi) = 1/2 \) and \( x(2, t = T = 2\pi) = 1/2 \) we calculate by aid of eq. (9) that

\[
y = Hx \equiv \begin{pmatrix} \frac{dx(1, t)}{dt} \\ \frac{dx(2, t)}{dt} \end{pmatrix} = \begin{bmatrix} -0.46021126 & 0.53978874 \\ -0.46021126 & 0.53978874 \end{bmatrix} \begin{pmatrix} x(1, t) \\ x(2, t) \end{pmatrix}.
\]

(29)

For this differential equation system eq. (29) we find the non-periodic solution

\[
x(1, t) = \frac{\exp(t/4\pi)}{2 * \exp(1/2)}
\]

(30)

\[
x(2, t) = \frac{\exp(t/4\pi)}{2 * \exp(1/2)}
\]

(31)

which is depicted in figure (2) along with the solution eq. (28) which is equal to zero for all \( t \). Our spring-mass-system has gained miraculously an external driving force which let the elongation \( y \) monotonously grow ad infinitum with \( y(2\pi) = 1/2 \) as we wished.
3.4 Example: Robot at fixed position

In our examples discussed so far, we calculated $H$ for fixed $x_0$ and desired $y_0$. We easily can swap the roles of fixed and variable values among $x$ and $y$ and still apply eq. (9). That is, we are given with $x$ which changes from value $x_0$ to some other value $x_1$ but we want to arrive at the same $y_0$ in both cases. Then our equations are $y_0 = Jx_0$ and $y_0 = Hx_1$. In order to apply eq. (9) we just take $x_1$ for $x_0$ and we just take $y_0$ both for $y_0$ and $y_1$. As an example, let us treat the task to keep a robot at a fixed position $y_0 = (1, 1, 1)^T$ in Euclidean 3d-space. At time instant $t_0$ the robot reacts to input according to reaction $J = [(1, 0, 0), (0, 1, 0), (0, 0, 1)]$ (i.e. here $J$ is equal to the unit matrix) which results for input $x_0$ in $y_0 = Jx_0$ as desired. At time instant $t_1$ the input changes to $x_1 = (6, -6, 9)^T$ and the robot changes its reaction to $H = [(0.8039, 0.1960, -0.2941), (0.2745, 0.7254, 0.4117), (-0.3137, 0.3137, 0.5294)]$ which results in $y_0 = Hx_0$ as desired.

4 Discussion

Let us contemplate on the relevance of our matrix alteration method. In the example Input-Output Analysis we are faced with the problem that the input $x$ is fixed to a value of $x_0$ but we need to deliver an altered output in order to fulfil the demand which has changed from $d_0$ to $d_1$. Our only possibility
is to change the economy, that is the matrix $A$ irrespective how difficult this approach may be.

Such a situation is common. Think of a trading company *Purchase&Sales Ltd* which buy and sell quantities $q_{\text{in}}$ and $q_{\text{out}}$ of the product $P$. Unfortunately *Purchase&Sales Ltd* are faced with notoriously high purchase prices $pp$ and low sales prices $sp$ for their product $P$. Thus, their input vector is composed of $q_{\text{in}}$, $q_{\text{out}}$, $pp$, $sp$ while their output is their revenue $r$. If the *Purchase&Sales Ltd* management is discontent with $r$ then the management initiates a restructuring of *Purchase&Sales Ltd*. More generally, we may be responsible for the state $y$ of a system $S$ and we have the means to restructure $S$. Whereby $S$ is represented by some matrix $J$ and we thus want to shift $y$ in a favorable region (of some state space which we do not define here). We may call $J$ Input-Output Matrix or Structure Matrix or Reaction Matrix depending on the concrete $S$ under consideration. In any case, if we have the chance to get from $y_0 = Jx_0$ to $y_1 = Hx_0$ with $y_1$ more favorable than $y_0$, then we will take this opportunity.

A most prominent example for $S$ is ourselves. In the sense of this discussion, education is the powerful means to make a person capable to process input $x_0$ and return a $y_1$ which is more than the average answer $y_0$.

On the other hand, obviously matrix alteration methods according to eq. (9) or similar ones are not in widespread use. This fact may be partially caused by the circumstance that a matrix alteration does not lead to unique results unless some additional constraint is applied. Furthermore, it may be difficult or even impossible to change the underlying system $S$ in a way compatible with the results of eq. (9). Still, these facts do not explain the lack of matrix alteration methods in the literature. Then again, the term ”matrix alteration” can be understood independently of eq. (9) and then many methods exist which we can subordinate under matrix alteration, think for example of matrix regularization or of matrix completion. In the later case we are given with an incomplete matrix, some elements are missing and we have to find reasonable values for them. Also, think of principal component analysis were we try to get rid of not so important dimensions of eq.(9) in order to receive a clearer picture of the data. Finally, let us come to an example of recently increasing importance, that is deep learning. Broadly speaking in deep learning we train a machine to arrive at a matrix which returns optimal output $y$ for given input $x$ according to the eq. (9). It may turn out that matrix alteration according to eq. (9) can serve as a tool for deep learning, for example in order to find some good initialization values for the matrix.

### A  
**Python program CALC_MAT_H (J,y1,x0,u0)**

```python
import sys
import numpy as np
def CALC_MAT_H (J,y1,x0,u0):
    # Matrix J maps vector x0 to vector y0 according to y0 = J * x0 + u0.
    # CALC_MAT_H calculates from given J, x0 and u0 the matrix H
```
which maps vector $x_0$ to vector $y_1$ according to $y_1 = H \cdot x_0 + u_0$.

$H$ is as close as possible to $J$.  
February 2019, thomas.wieder@t-online.de

```python
try:
    # *** Check input ***
    assert isinstance(J, np.ndarray)
    assert isinstance(y1, np.ndarray)
    assert isinstance(x0, np.ndarray)
    assert isinstance(u0, np.ndarray)
    H = np.array([])
    dim1_x0 = x0.shape[0]
    dim2_x0 = x0.shape[1]
    if dim2_x0 != 1:
        print("Error in CALC_MAT_H: x0 is not a column vector! x0 = ",x0)
        quit
    dim1_u0 = u0.shape[0]
    dim2_u0 = u0.shape[1]
    if dim2_u0 != 1:
        print("Error in CALC_MAT_H: u0 is not a column vector! u0 = ",u0)
        quit
    if dim1_u0 != dim1_x0:
        print("Error in CALC_MAT_H: Vectors u0 and x0 are of different length!")
    dim2_J = J.shape[1]
    if dim2_J != dim1_x0:
        print("Error in CALC_MAT_H: dim2_J != dim1_x0 !")
    print("Error in CALC_MAT_H: dim2_J = ",dim2_J," dim1_x0 = ",dim1_x0)
    return H
    quit

    # *** Calculate H ***
    x0T = np.transpose(x0)
    x0Tx0 = x0T.dot(x0)
    if x0Tx0 == 0.0:
        H = J
        print("CALC_MAT_H: H = J")
        return H
    quit

    y0 = np.matmul(J,x0)
    y0 = y0 + u0
    AUX = np.matmul((y1 - y0),x0T)
    AUX = AUX/x0Tx0
    H = J + AUX
    # *** Exit ***
except: # catch *all* exceptions
    e = sys.exc_info()[0]
    print(<p>Error: %s</p>" % e)
print("CALC_MAT_H: H")
return H
```

References


Received: July 23, 2019; Published: August 5, 2019