

Automatic collision avoidance in commercial aircraft three dimensional flights, using neural networks and non-linear programming

M.A.Christodoulou and C.Kontogeorgou
Systems Division
Department of Electronic and Computer Engineering
Technical University of Crete
Chania, Crete, 73100, Hellas (Greece)
manolis@ece.tuc.gr, xkontogeorgoy@isc.tuc.gr

Abstract—In recent years there has been a great effort to convert the existing Air Traffic Control system into a novel system known as Free Flight. Free Flight is based on the concept that increasing international airspace capacity will grant more freedom to individual pilots during the enroute flight phase, thereby giving them the opportunity to alter flight paths in real time. Under the current system pilots must request, then receive permission from air traffic controllers to alter flight paths. Understandably the new system allows pilots to gain the upper hand in air traffic. At the same time, however, this freedom increase pilot responsibility. Pilots face a new challenge in avoiding the traffic shares congested air space. In order to ensure safety, an accurate system, able to predict and prevent conflict among aircrafts is essential. There are certain flight maneuvers that exist in order to prevent flight disturbances or collision and these are graded in the following categories : vertical, lateral and airspeed. This work focuses on airspeed maneuvers and tries to introduce a new idea for the control of Free Flight, in three dimensions.

I. INTRODUCTION

A system of many aircraft is a real time system. An accurate real time system requires the minimum possible response time so that it functions well.

In dealing with this specific problem we focus on two basic areas in the creation of an accurate control system based on airspeed maneuvers: **optimum (minimum) velocity changes** [1] and **quick response** of the system. Initiating slight changes in velocity, requires certain amount of time. Furthermore, an immediate response from the system, requires a quick change computation. If those times are minimal then the total processing time is also minimized, which is a basic principle in real time systems.

By considering the importance of response time in relation to the prevention of air traffic conflict, we concluded that the most appropriate way to update velocities is the use of a neural network. We know that a trained neural network has very short response time.

Neural networks [2] can be characterized as computational entities with particular properties such as the ability to adapt or learn, to generalize, as well as to cluster or organize data, and whose operation is based on parallel processing. The intriguing question however, is to what extent does the neural approach prove to be better suited to certain applications rather than other models.

II. PURPOSE OF THIS PAPER

The goal of this paper is the creation of a neural network that can predict the optimal velocity change of two aircraft in order to avoid an imminent conflict.

First and foremost, collision cases were gathered for the creation of the specific neural network. This task was managed with Matlab which created random flights with true velocities. In this way, a great number of cases are gathered but many are irrelevant to the problem, so the method of sample deletion is used.

The next step is the creation of a nonlinear program in GAMS [3] tool to find the new velocities of each aircraft in order to avoid conflict. The function of this program is to find the minimum velocity changes. Furthermore, it is important to mention that for each conflict case there is a unique file in GAMS [3], giving solution to the problem as well as a specific file keeping the velocity changes. Because of the great amount of data two side programs in C++ are used in order to edit these files (GAMS, velocity) in a few seconds while manually it would have taken days for this work.

The final step is the collection of data to train the neural network [2]. Thus it was decided that the inputs of the neural network, known as training set, would consist of initial and final positions in the control sphere (see chapter III), as well as initial velocities. The desired output, known as the target set, of the neural network are the velocity changes [1].

III. PROBLEM APPROACH

First of all a true scenario is represented for the Free Flight problem. There is a sphere of radius 108 kilometers, termed here as the control zone [4].

The object of our study is a pair of aircraft flying in the sphere. Each airplane has random initial and final configuration points in the control sphere. We consider the motion of the two aircraft being linear with constant velocities in order to simplify the problem. When the two objects become closer to each other than 9 kilometers we assume that a conflict occurs [5].

First, we create a sphere of radius 108 kilometers and two aircraft randomly distributed in it. The first aircraft (as well as the second) has random initial points $P_{aircraft1}(x_i, y_i, z_i)$

and final points $P_{aircraft1}(x_f, y_f, z_f)$ on the sphere surface.

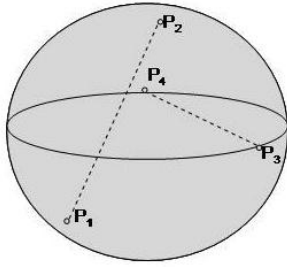


Fig. 1. P1 (P3) is the insertion point, P2 (P4) the escape point in the sphere of the first (second) aircraft.

Furthermore, to determine each aircraft in a unique way, we use the two spherical angles, θ and ϕ [10]. Notice that for the two dimensional space we need only one angle to define the direction of an object. Thus for the 3 dimensional space we need 2 angles. Where

$$-\frac{\pi}{2} \leq \theta \leq \frac{\pi}{2} \quad (1)$$

and

$$0 \leq \phi < 2\pi \quad (2)$$

To resume, we have the position of an object totally determined by the point: $P_{aircraft1}(x_i, y_i, z_i, \theta_i, \phi_i)$. We now define a relationship between the two objects so we consider the route as a function of time. Thus the configuration of an aircraft at time t is:

$$P_{aircraft1}(x_t, y_t, z_t, \theta_t, \phi_t).$$

To test whether the objects collide, we check the distance between them during a specific period of time [6].

IV. EQUATIONS FOR COLLISION SCENARIOS

The equations [10] expressing the motion from the initial to the end points are:

$$\begin{cases} x = x_1 + a_1\lambda \\ y = y_1 + a_2\lambda \\ z = z_1 + a_3\lambda \end{cases} \quad (3)$$

Where

$$a = (a_1, a_2, a_3) = (x_2 - x_1, y_2 - y_1, z_2 - z_1) \quad (4)$$

and λ is a variable.

We select random values for the two input angles ϕ_i and θ_i and for the output angles ϕ_f and θ_f in the sphere surface, and we compute the cartesian coordinates as follows:

$$x = R\sin(\phi)\cos(\theta) \quad (5)$$

$$y = R\sin(\phi)\sin(\theta) \quad (6)$$

$$z = R\cos(\phi) \quad (7)$$

Where the θ and ϕ angles are shown in the figure below:

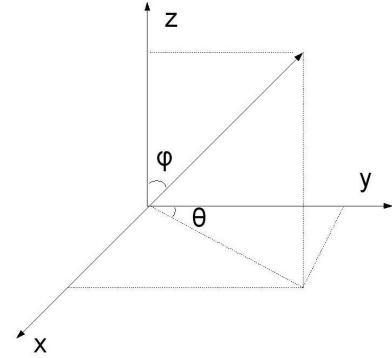


Fig. 2. The angles that determine the position of the aircraft in the 3D space.

By using the fact that the two aircraft follow a linear path with constant velocities we get the following equation.

$$s = u_1t \quad (8)$$

On the other hand the length s covered by an object in a flight is equal to :

$$s = \sqrt{(x - x_1)^2 + (y - y_1)^2 + (z - z_1)^2} \quad (9)$$

using the usual mathematical type of distance.

By equating (8) and (9) we get:

$$u_1t = \sqrt{(x - x_1)^2 + (y - y_1)^2 + (z - z_1)^2} \quad (10)$$

If we replace term $(x - x_1)$ with its equal from equation (3) $a_1\lambda$ we finally have:

$$u_1t = \lambda\sqrt{(a_1)^2 + (a_2)^2 + (a_3)^2} \quad (11)$$

and if we set the equation: $u_1t = \lambda|a|$ by analyzing the above equation we have:

$$\lambda = \frac{u_1t}{|a|} \quad (12)$$

Furthermore we can replace the term λ with its equal in equation (3) and finally we get the equations below:

$$x = x_1 + \frac{a_1u_1}{|a|}t \quad (13)$$

$$y = y_1 + \frac{a_2u_1}{|a|}t \quad (14)$$

$$z = z_1 + \frac{a_3u_1}{|a|}t \quad (15)$$

With these functions we know where an aircraft is positioned at time t . So we know the distance between the two aircraft at every time instant t . The distance between them is:

$$d = \sqrt{X_{diff}^2 + Y_{diff}^2 + Z_{diff}^2}$$

knowing that:

$$\begin{aligned} X_{diff} &= x_i(t) - x_j(t) \\ Y_{diff} &= y_i(t) - y_j(t) \\ Z_{diff} &= z_i(t) - z_j(t) \end{aligned}$$

V. OPTIMUM VELOCITY CHANGES WITH THE HELP OF GAMS

1) *Basic ideas*: The main decision variable that determines whether or not we have conflict is the distance between the two aircraft. If this distance is less than 9 km then we have conflict. Based on this fact, the distance is defined as a function of time. Thus the solution for the avoidance of conflict lies in keeping the distance between the two objects greater than 9 km during the entire route. First of all, we sectioned the time of the route into a number of parts. To begin with, we know that the aircraft fly with a speed equal to 15 km per minute and that the control sphere has a diameter of 216 km. It is easily understandable that the greatest distance that can be covered is 216 km and this can be done in 14.4 minutes. Whether the segment of time is a minute, half a minute or a quarter of a minute depends on how much accuracy we require. In the specific solution, the amount of the time is a quarter of a minute. Since we decided how to split the time, we now require, for all these time segments, the distance between the two objects to be greater than 9 km. According to this, the velocities of the two collision objects will keep changing until the desired ones are found [7], [8], [9].

2) *Algorithm for GAMS*: The velocity change occurs when two aircraft are flying in specified directions and must avoid a conflict by implementing a velocity magnitude change. Each aircraft can change its velocity by a quantity q that can take on positive or negative values. It is understandable that aircraft can not make exorbitant changes in their velocities, instead they can make limited ones. More specifically, there exist upper and lower bounds of velocities for each aircraft given by the following equation:

$$U_{i,min} \leq U_i \leq U_{i,max} \quad (16)$$

The velocity bounds differ from aeroplane to aeroplane. The new velocity is obtained from the old one augmented by a value of q (negative or positive) and must not exceed the minimum and maximum bounds. This can be represented by the following equation:

$$U_{i,min} \leq U_i + q_i \leq U_{i,max} \quad (17)$$

The usual bounds for a commercial aircraft must not exceed 1% of their nominal velocity. We choose stricter bounds in our case, 0.5%. Consequently q has lower and upper bounds as follows

$$-0.99 \leq q_i \leq 0.99 \quad (18)$$

In our algorithm we consider that two aircraft conflict if the distance between them is less than 9 km. The distance between them is given by the following equation:

$$d = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2 + (z_i - z_j)^2}$$

As mentioned before, if we manage to change the velocities in a way that augments the distance between them of more than 9 km, we will have conflict resolution. The basic issue is that there are an infinite number of values that can manage this resolution but actually only one set will be optimum according to the hierarchy defined in our algorithm. Therefore, we want to achieve $\min|q_i|$, $i = 1, 2$.

Furthermore in non-linear programming there are some constraints. Our algorithm, is based on a very simple idea: By taking the route of the aircraft as a function of time we can split the route into time slices. An example shown in the figure reveals the idea.

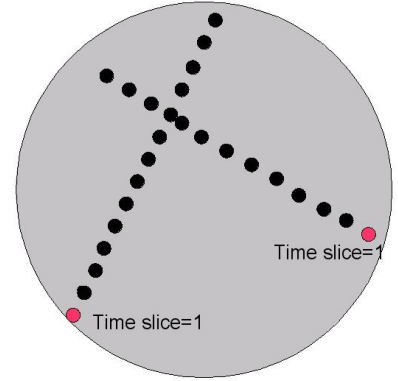


Fig. 3. Position of each aircraft at every time slice

In the above figure, we see the positions of the aircraft in time slice 1 in red. We examine, for each time slice, the distance between the two current positions d_{12} . Whenever $d_{12} \leq d_{col}$, a collision occurs. Therefore, if for all these positions, all the distances are made greater than 9 km, then the conflict has been successfully avoided.

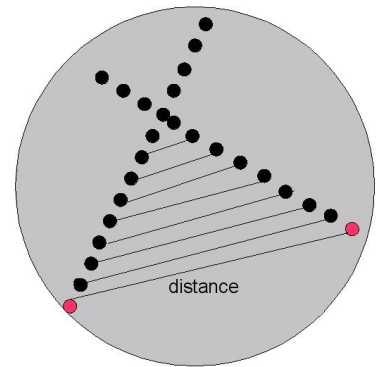


Fig. 4. Distances between aircraft position

Thus, the constraints in our non-linear program according to [1] are the following:

$$dis(1, i, j) > 9 \quad (20)$$

and

$$dis(2, i, j) > 9 \quad (21)$$

and

.

$$d(n, i, j) > 9 \quad (22)$$

The maximum time n in our problem is equal to 50. Consider that the max distance, each aircraft can travel is 216 km and with a velocity of 15km/minute, it needs 14,4 minutes in total. So by taking almost a quarter of minute as our time slice, we need at most 50 slices.

At the end of each GAMS program we get the optimum velocity changes. By applying these changes, actually we decrease the velocity for one of the two aircraft and increase the other's. As a result, the aircraft passes the crucial conflict point at another time slice and thus the conflict is avoided.

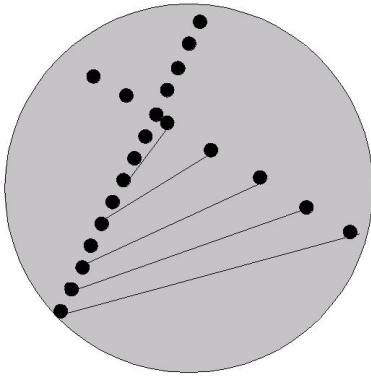


Fig. 5. With the velocity changes the aircraft pass through the crucial point at different time slices

3) *Implementation of the problem:* GAMS [3], [11], [13], [14] is an important tool in non linear programming. It was used to compute the minimum velocity change of two aircraft that are about to conflict, so that their conflict to be avoided. The algebraic representation of conflict avoidance problem is represented in the format below:

Indices:

i =number of aircrafts

t =timeslice

Given Data:

Coordinates:

$x_i = x$ initial coordinate for the i aircraft in Km

$y_i = y$ initial coordinate for the i aircraft in Km

$z_i = z$ initial coordinate for the i aircraft in Km

$x_{fin} = x$ final coordinate for the i aircraft in Km

$y_{fin} = y$ final coordinate for the i aircraft in Km

$z_{fin} = z$ final coordinate for the i aircraft in Km

Decision Variables:

f equals to $\sum(i, q_i)$

Constraints:

q_i where q is belong in the set $[-0.9,0.99]$

Objective Function:

Considering that:

- $A_1(j) = x_{final}(i) - x_i$
 - $A_2(j) = y_{final}(i) - y_i$
 - $A_3(j) = z_{final}(i) - z_i$
 - $u(i)$ are the initial velocities of the two aircrafts
 - $q(i)$ are the velocity changes
 - $number(t)=0,1,2,3,\dots,50$
 - $dis(i) = \sqrt{(A_1(i))^2 + (A_2(i))^2 + (A_3(i))^2}$
 - considering that
- $$B_x(i) = x(i) + \frac{A_1(i)}{distance(i)}(u(i) + q(i))0,5number(t)$$
- $$B_y(i) = y(i) + \frac{A_2(i)}{distance(i)}(u(i) + q(i))0,5number(t)$$
- $$B_z(i) = z(i) + \frac{A_3(i)}{distance(i)}(u(i) + q(i))0,5number(t)$$

Then the objective function will be:

$$dis_t(i, j) = \sqrt{(B_x(j) - B_x(i))^2 + (B_y(j) - B_y(i))^2 + (B_z(j) - B_z(i))^2}$$

The complexity of the algorithm is $O(n^2)$ and that is because we have n aircrafts then we are going to get complexity $\binom{n}{2}25 = \frac{n!}{(n-2)!*2!*25} = \frac{n(n-1)}{2}$.

VI. CONFLICT SCENARIOS

Below there are given some conflict scenarios:

Example:

Aircraft ₁	Start	Final
x	-52.114	-33.968
y	-94.561	71.402
z	-2.5359	-73.566
Aircraft ₂	Start	Final
x	-76.198	-28.02
y	-61.507	44.038
z	45.55	-94.549

The velocity changes are: for the first aircraft 0.023 and for the second -0.021.

A. Cases with 3 aircraft

The algorithm used for the conflict solution of two aircraft applies also in cases with more than two. It is noticed that for three and four airplanes the algorithm is fast enough.

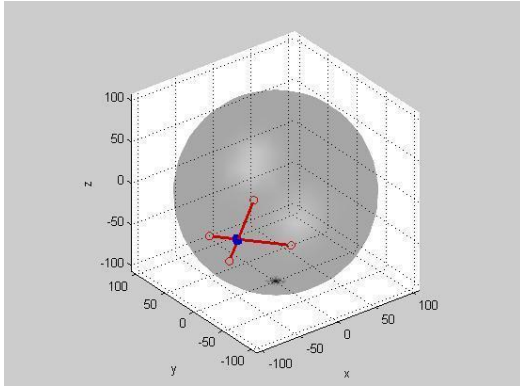


Fig. 6. Example

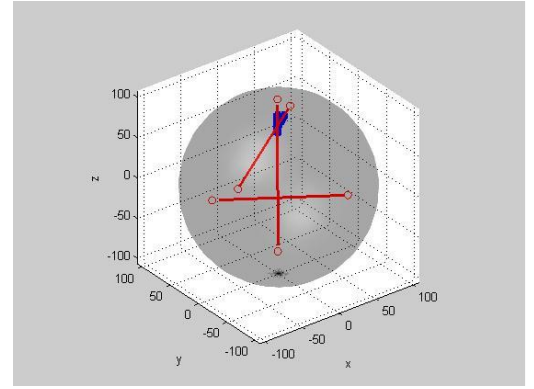


Fig. 8. Conflict Example with three aeroplanes

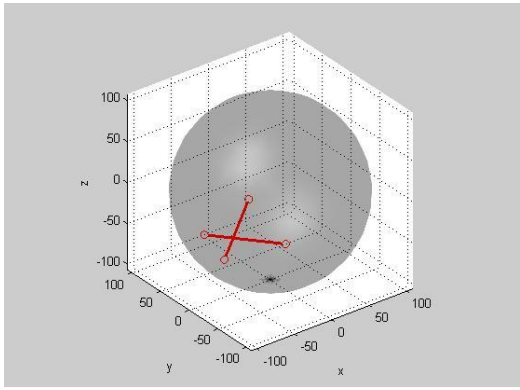


Fig. 7. Example after conflict resolution

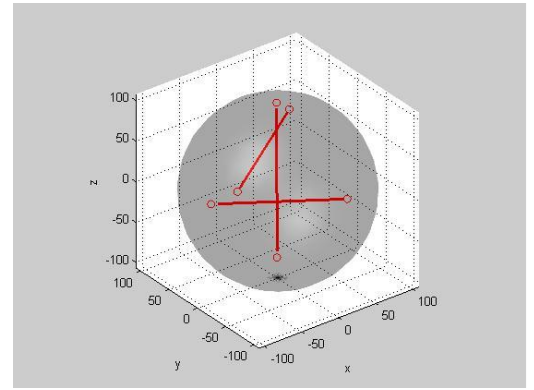


Fig. 9. Conflict Resolution with three aeroplanes

For more than four airplanes the computational complexity is increased but still we take results fast. We set out an example for three aircraft.

Aircraft ₁	Initial	Final
x	3.2643	16.535400
y	-16.0633	93.59
z	106.749	-51.2973

Aircraft ₁	Initial	Final
x	-0.39850	20.0728
y	1.4585	27.527401
z	107.99	-102.4857

Aircraft ₁	Initial	Final
x	-103.034	15.146100
y	-17.2012	-102.261
z	27.4251	31.2734

The optimum velocity changes for each aircraft are : - 0.9 for the first airplane, 0.761 for the second and 0 for the third. The same algorithm was tested also for four aeroplanes with very good response time.

VII. CONFLICT AVOIDANCE WITH THE NEURAL NETWORK

In our problem the neural network selected is of a back propagation type with two hidden layers [14], [15]. One input layer with twelve inputs and one output layer with two outputs. The twelve inputs are ordered by the way they are written in the train file.

Inputs:

- The initial positions of the first aircraft $x_{in1}, y_{in1}, z_{in1}$
- The initial positions of the second aircraft $x_{in2}, y_{in2}, z_{in2}$
- The final positions of the first aircraft $x_{fin1}, y_{fin1}, z_{fin1}$
- The final positions of the second aircraft $x_{fin2}, y_{fin2}, z_{fin2}$

The first hidden layer has 50 Neurons and the second 40 neurons. We decided to choose these numbers of neurons for each layer, as this resulted in minimal training and testing error.

Outputs:

- The velocity change for the first aircraft ΔV_1
- The velocity change for the second aircraft ΔV_2

We finally use 3500 data for training and 600 data for testing [16]. The reliability of the network is examined by the method of cross validation. Cross-validation is a method to estimate the generalization error based on "resampling".

In k-fold cross-validation, data is divided into k subsets of (approximately) equal size. The net is trained k times, each time leaving out one of the subsets from training. In the beginning, the data are separated in ten parts (k=10), then each time 8 different parts are chosen to be the training data and the two remaining parts, the testing data. The neural network is trained with 10 slightly different sets of data. Then the final training and testing error is estimated by taking the average over all the errors resulted from each time we train the network.

The number of epochs essential for the training of the neural network is 700. The transfer function used in every unit is the hyperbolic tangent sigmoid function.

A. Evaluation of the neural network

By following the above mentioned states, we train the feed forward back propagation neural network. In a first approach we get a testing error of 6% which is a considerably big. In this part of the thesis we have the following issues:

- First, it would be satisfying to have a testing error of 0,1% but in such big range of values a 6% could be considered as satisfactory.
- Second, the issue of the highest importance in this study, is the accuracy of the system, especially while this study is related to aircraft where there is no tolerance in errors. Also the solutions taken from GAMS are optimum for each case and as it was mentioned above if the second decimal of one of the solutions is changed we are going to have conflict again. But the basic thing that was not mentioned until now is : “what we mean by using the word change”: decrease or increase? The answer is simple, because if the solution is negative, by providing a more negative solution, nothing will happen, but by increasing the solution, it becomes a fact that a conflict will occur. Similarly, if we have a new velocity bigger than the old one and we increase it even more, nothing will happen.

Sometimes the optimum velocity is not acceptable because if we slightly change it we will fail in solving the problem. As we can see in the following scheme there are two optimum velocities, one for the first aircraft and one for the second. Assume the case where the first aircraft has to decrease its original velocity and the second to increase it. Because we have optimum velocity changes if we give at the negative solution a more negative value, we still avoid the conflict but if we give a solution less negative than the optimum we will present a conflict. As we want a safe belt in our system, we adjust the solution with a value of 0,12. We either add or subtract this value depending on the category of solution as it was mentioned above. This approach makes the system more accurate and the testing error of 6% acceptable.

Two matrices with error information for 10 different training sets used through cross validation are represented below.

Set/Error	Average Training Er
set1	$13 * 10^{-4}$
set2	10^{-4}
set3	$18 * 10^{-4}$
set4	$9 * 10^{-4}$
set5	$4,7 * 10^{-4}$
set6	10^{-3}
set7	$12,3 * 10^{-4}$
set8	$11,98 * 10^{-4}$
set9	$3,2 * 10^{-4}$
set10	$6,81 * 10^{-4}$

Set/Error	Average Testing Er
set1	6.16
set2	6.53
set3	5.25
set4	5.89
set5	5.93
set6	6.01
set7	5.78
set8	4.99
set9	6.24
set10	5.9

VIII. CONCLUSIONS

A new algorithm that operates in real time for 3-dimensional collision avoidance in free flight is presented. The algorithm is proven via well established simulations, to be well functioning in cases where many aircraft are involved.

REFERENCES

- [1] M.A.Christodoulou and S.G. Kodaxakis, “Automatic Commercial Aircraft Collision Avoidance in Free Flight: The Three Dimensional Problem”, *IEEE Transactions on Intelligent Transportation Systems*, vol.7, No.2, pp. 242-249, 2006
- [2] Ben Krose, Patrick van der Smagt, *An introduction to Neural Networks* Eighth Edition, chapter2 p16-18, 1996
- [3] Gams site: www.gams.com
- [4] I.Kodaxakis, *Free Flight in Commercial aircraft: A new algorithm for automatic conflict avoidance in 3-D space*, Masters Thesis, Department of Electronic and Computer Engineering, Technical University of Crete, 2005
- [5] www.faa.gov
- [6] Krozel, J.Peters, M.Seagull Techno, “Strategic conflict detection and resolution for free flight”, *Decision and Control, Proceedings of the 36th IEEE Conference*, 10-12-07
- [7] Merrill Skolnik, *Introduction to radar systems*, McGraw-Hill Company, INC, Tokyo
- [8] <http://ams.allenpress.com>
- [9] <http://mdl.csa.com/partners/viewrecord.php>
- [10] Earl W.Swokowski, *Calculus with analytic Geometry second edition*, chapter 14 p.680
- [11] C.A.Floudas, *Non-Linear and Mixed-Integer Optimization*, Eighth Edition, 1996
- [12] Richard E.Rosenthal, *Af Gams tutorial*
- [13] D.P.Bertsekas, *Non-Linear Programming*, Boston M: Athena Scientific, 2005
- [14] Ben Krose, Patrick van der Smagt, *An introduction to Neural Networks*, Eighth Edition, chapter2 p20, 1996
- [15] Ben Krose, Patrick van der Smagt, *An introduction to Neural Networks*, Eighth Edition, chapter3 p23-24, 1996
- [16] Gill P. R., Murray W. and Wright M. H., *The Levenberg-Marquardt Method*, chapter4.7.3 in *Practical Optimization*, London: Academic Press, pp. 136-137, 1981.