REVISTA COLOMBIANA DE COMPUTACIÓN
Volumen 9, número 1
Págs. 58 - 71

Optimization's trajectory of an automatic manipulator using neural networks

Fedossova A.V. * Santoyo J.S. †

Fecha de Recibido: 06/05/2008; Fecha de Aprobación: 09/08/2008

Abstract

Neural networks are used to optimize the trajectory of an automatic manipulator. First, it is had an automatic manipulator of three degrees of freedom. The proposed problem consists of finding total optimal time of displacements which must adjust the trajectory using neural networks trained for that purpose. It is an optimization problem subject to an infinite set of restrictions. For to solve is used three types of neural networks of MATLAB 7.0.1

Keywords: *Neural Networks, Optimization, Semi-infinite programming*

1 Problem Definition

We considered a robot arm (manipulator) of j degrees of freedom θ1, θ2 and θj (j links). Each link has a length and mass associated. Since the robot position varies with time we can define a robot trajectory as a parametric curve:

$$\theta(T) = (\theta_1(T), \theta_2(T), ... \theta_l(T))^T, \quad T \in [\ 0, T_f\ ],$$

where l is the number of degrees of freedom and Tf is the total travel time. Let " . " indicate the derivative with respect to the time t and " ' " the one with respect to T.

We formulated the optimization problem of optimal trajectory follows [8]. Suppose that we know n points in the trajectory of the manipulator. Are

$$[\theta_1(T_1), \theta_1(T_2), ..., \theta_1(T_n)], [\theta_2(T_1), \theta_2(T_2), ..., \theta_2(T_n)], ..., [\theta_l(T_1), \theta_l(T_2), ..., \theta_l(T_n)]$$

* National University of Colombia - Campus Bogotá, Avenue Street 30 N° 45-03 – Bogotá, Colombia, afedosova@unal.edu.co
† Autonomous University of Bucaramanga, Street 48 No. 39 – 234 – Bucaramanga, Colombia, jsdiaz@unab.edu.co

vector points (nodes) which passes the trajectory of automatic manipulator. The proposed problem consists of finding total optimal time of displacements which must adjust to the trajectory using cubic splines constrained by the speed, acceleration and jerk. We get $t_1 < t_2 < ... < t_n$ a sequence at time where t_i this is the time where the robot is in the position [$\theta 1$ (T_i), $\theta 2$ (T_i), ..., θj (T_i)]. Natural constraints applied to the parametric curve are:

$$\sum_{i=1}^{t} \left(\frac{d\theta_i}{dT}\right)^2 > 0, \quad T \in (0, T_f)$$

$$\frac{d\theta}{dT}(0) = \frac{d\theta}{dT}(T_f) = 0$$

$$\frac{d^2\theta}{dT^2}(0) = \frac{d^2\theta}{dT^2}(T_f) \neq 0$$

Are $d_1 = t_2 - t_1$, $d_2 = t_3 - t_2$, ..., $d_{n-1} = t_n - t_{n-1}$ the displacement times. Is Q_{ij} cubic spline for the link *i* the robotic arm that approximates to $\theta_i(t)$ in [t_j , t_{j+1}].

Problem can then be formulated as SIP:

$$\min \sum_{j=1}^{n-1} d_j$$

$$\text{subject to } |Q'_{ij}(t)| \leq C_{i,1}$$

$$|Q''_{ij}(t)| \leq C_{i,2}$$

$$|Q'''_{ij}(t)| \leq C_{i,3}$$

$$i = 1, ..., l$$

$$d_j > 0 \quad j = 1, ..., n-1;$$

where $C_{j,1}$, $C_{j,2}$ and $C_{j,3}$ are the bounds for the velocity, acceleration and jerk, respectively, on joint *i*.

We introduce

$$t = r(T),\ T \in [0,1],$$

and is

$$g(T) = r'(T) > 0,\ T \in [\ 0,1\ ].$$

So the problem is reformulated as

$$r(0) = 0$$

$$r(1) \quad \text{is minimum}$$

$$r'(T) > 0 \quad T \in [\ 0,1\ ]$$

$$|\ \dot{\theta}_i^*\ | \leq C_{i,1}$$

$$|\ \ddot{\theta}_i^*\ | \leq C_{i,2}$$

$$|\ \dddot{\theta}_i^*\ | \leq C_{i,3}\ i = 1,...,l$$

and

$$\dot{\theta}^*\ (0) = \dot{\theta}^*\ (1) = 0$$

where $C_{i,1}$, $C_{i,2}$ and $C_{i,3}$ are the bounds for the velocity, acceleration and jerk, respectively, on joint i, and

$$\theta^*(t) = \theta(r^{-1}(T))$$

is the parametric curve.

We get

$$T = r^{-1}(t),\ \ t \in [\ 0, t_f\ ].$$

then,

$$\frac{dT}{dt} = \frac{1}{\frac{dr}{dT}} = \frac{1}{r'(T)} = \frac{1}{g(T)}.$$

By the chain rule we get,

$$\dot{\theta}_j = \frac{d}{dt}\{\theta_j(r^{-1}(t))\} = \frac{d}{dt}\{\theta_j(T)\}$$

$$\dot{\theta}_j = \frac{\theta'_j(T)}{g(T)}$$

$$\ddot{\theta}_j = \frac{g(T).\theta''_j(T).\frac{1}{g(T)} - \theta'_j(T).g'(T).\frac{1}{g(T)}}{g^2(T)}$$

$$\ddot{\theta}_j = \frac{\theta''_j(T) - \theta'_j(T).\frac{g'(T)}{g(T)}}{g^2(T)}$$

$$\dddot{\theta}_j = \frac{\theta'''_j(T) - 3\theta''_j(T).\frac{g'(T)}{g(T)} + \theta'_j(T).\frac{3(g'(T))^2 - g(T).g''(T)}{g^2(T)}}{g^3(T)} .$$

Rewriting the constraints of previous model we have:

$$(-1).|\ \dot{\theta}_j\ (T)| \ \geq\ (-1)\cdot C_{j,1},$$

$$\frac{(-1)}{C_{j,1}}.|\ \dot{\theta}_j\ (T)| \geq -1$$

if

$$|\ \dot{\theta}_j\ (T)| = \frac{\theta'_j(T)}{g(T)},$$

then

$$1 - \frac{(-1)^p}{C_{j,1}}.\frac{\theta'_j(T)}{g(T)} \geq 0.$$

Same way,

$$(-1).|\ \ddot{\theta}_j\ (T)| \ \geq\ (-1)\cdot C_{j,2}$$

we obtain

$$\frac{(-1)}{C_{j,2}}.|\ \ddot{\theta}_j\ (T)| \geq -1,$$

and

$$1 - \frac{(-1)^p}{C_{j,2}} . \frac{\theta''_j(T) - \theta'_j(T) . \frac{g'(T)}{g(T)}}{g^2(T)} \geq 0.$$

after

$$(-1) . | \dddot{\theta}_j (T)| \ \geq \ (-1) \cdot C_{j,3}$$

and

$$\frac{(-1)}{C_{j,3}} . | \ddot{\theta}_j (T)| \geq -1,$$

where we obtain

$$1 - \frac{(-1)^p}{C_{j,3}} . \frac{\theta'''_j(T) - 3.\theta''_j(T) . \frac{g'(T)}{g(T)} + \theta'_j(T) . \frac{3.(g'(T))^2 - g(T).g''(T)}{g^2(T)}}{g^3(T)} \geq 0,$$

where

$$p \in \{ \ 0,1 \ \}, T \in [\ 0,1 \ ]$$

and

$$j = 1, 2, 3.$$

Problem can then be rewrite as SIP:

$$\begin{aligned} &\min_{x \in R^n} \int_0^1 g(T) \, dT, \qquad (1) \\ &\text{subject to} \ \ g(T) > 0 \\ &| \ \dot{\theta}_{j(T)} \ | \ \leq \ C_{j,1} \\ &| \ \ddot{\theta}_j \ (T)| \ \leq \ C_{j,2} \\ &| \ \dddot{\theta}_j \ (T)| \ \leq \ C_{j,3} \\ &j = 1, 2, 3 \\ &\forall T \in [\ 0,1 \ ], \end{aligned}$$

where $C_{j,1}, C_{j,2}$ and $C_{j,3}$ are the bounds for the velocity, acceleration and jerk, respectively, on joint *i*. The end conditions velocity equal to zero, that means

$$\dot{\theta}_j = 0,$$

and eventually also acceleration equal to zero, .

$$\ddot{\theta}_j = 0,$$

are satisfied for

$$T \in [\ 0, \bar{T}\ ].$$

We consider with more detail the problem (1).

Mathematical model use B-Spline function in target function, then, it lets bring a curve given by some nodes which should generate the trajectory. A B-Spline is a linear combination of blending functions. A B-Spline can be represented by:

$$B_{k,\varepsilon}(t) = \sum_{i=1}^{n} x_i B_{i,k,\varepsilon}(t).$$

Approximating function *g(T)* in:

$$F(x) = \int_0^1 g(T)\, dT,$$

the target function of problem is:

$$F(x) = \sum_{i=1}^{n} a_i x_i \ ,$$
$$\text{con } a_i > 0,$$

subject to constraints

$$H_0 = g - \epsilon \geq 0$$

$$H_{j,1} = 1 - \frac{(-1)^p}{C_{j,1}} \cdot \frac{\theta_j'(T)}{g(T)} \geq 0$$

$$H_{j,2} = 1 - \frac{(-1)^p}{C_{j,2}} . \frac{\theta''_j(T) - \theta'_j(T).\frac{g'(T)}{g(T)}}{g^2(T)} \geq 0$$

$$H_{j,3} = 1 - \frac{(-1)^p}{C_{j,3}} . \frac{\theta'''_j(T) - 3.\theta''_j(T).\frac{g'(T)}{g(T)} + \theta'_j(T).\frac{3.(g'(T))^2 - g(T).g''(T)}{g^2(T)}}{g^3(T)} \geq 0,$$

$$\text{where } p \in \{\ 0,1\ \}, T \in [\ 0,1\ ] \text{ and } j = 1,2,3.$$

The proposed problem consists of finding total optimal time of displacements which must adjust to the trajectory using cubic splines constrained by the speed, acceleration and jerk.

2 Types of Neural Networks used in MATLAB

2.1 Neural Network Toolbox - newgrnn

Design generalized regression neural network Syntax:

net = newgrnn(P,T,spread)

Generalized regression neural networks (grnns) are a kind of radial basis network that is often used for function approximation. grnns can be designed very quickly.

newgrnn(P,T,spread) takes three inputs,

P = R x Q matrix of Q input vectors
T = S x Q matrix of Q target class vectors

spread. Spread of radial basis functions (default = 1.0)

and returns a new generalized regression neural network.

The larger the spread, the smoother the function approximation. To fit data very closely, use a spread smaller than the typical distance between input vectors. To fit the data more smoothly, use a larger spread.

2.2 Neural Network Toolbox - newlind

Design linear layer. Syntax:

net = newlind(P,T,Pi)

Newlind (P, T, Pi) takes these input arguments,

P = R x Q matrix of Q input vectors
T = S x Q matrix of Q target class vectors
Pi = 1 x ID cell array of initial input delay states

where each element Pi{i,k} is a Ri x Q matrix, and the default = [],and returns a linear layer designed to output T (with minimum sum square error) given input P.

2.3 Models

Design an optimization algorithm to solve a particular problem, it involves the achievement of efficiency in terms of computational runtime, analysis and selection of the most appropriate techniques to achieve a level of generalization that allows apply and extend it to solve other problems closely related.

2.3.1 First Model

First model has been developed using results from thesis of PhD. A. Ismael F. Vaz. The present model is trained, and adapted a simulated a neural network newgrnn of MATLAB. This network uses 50 inputs and 50 outputs.

2.3.2 Second Model

Second model has been developed using own results of stochastic outer approximations algorithm. The present model is trained, and adapted a simulated a neural network newlind of MATLAB. This network uses 50 inputs and 50 outputs.

2.3.3 Third Model

Third model uses results from thesis of PhD. Elke Haaren – Retagne. This model is trained, and adapted a simulated a neural network newgrnn of MATLAB. This network uses 20 inputs and 20 outputs.

The software based on neural network operates as follows:

Step 0: Input:
From a random $X1 \in X^0$
Fo1 = Fo2 = Fo3 = Y1 = Y2 = Y3 = 0;
Step 1: Random X1.
Step 2: Apply to netali1(X1); it's trains, adapts and simulates with random X1 where is obtains Fo1 and Y1.
Step 3: Apply to netali2(X1); it's trains, adapts and simulates with random X1 where is obtains Fo2 and Y2.
Step 4: Apply to netali3(X1); it's trains, adapts and simulates with random X1 and finally is obtains Fo3 and Y3.
Step 5: Saves all results of the parameters X1, Fo1, Fo2, Fo3, Y1, Y2 y Y3 in a system log.
Step 6: Show final results.

2.3.4 Parameters Description

Parameters used in the three models described above are:

"**X1"** represents the initial value of the variable used in each iteration.
"**Fo1 (objective function of the first model)"** that's the value of objective function at the optimal point.
"**Fo2 (objective function of the second model)"** that's the value of objective function at the optimal point.
"**Fo3 (objective function of the third model)"** that's the value of objective function at the optimal point.
"**Y1"** that's the optimal value of X variable in the model 1.
"**Y2"** that's the optimal value of X variable in the model 2.
"**Y3"** that's the optimal value of X variable in the model 3.

The figure below shows an approximation to networks that were used in the prototype, considering that number of inputs in the prototype are 50, number of hidden layers are 12 and number of outputs 9. The plot is complex graph.

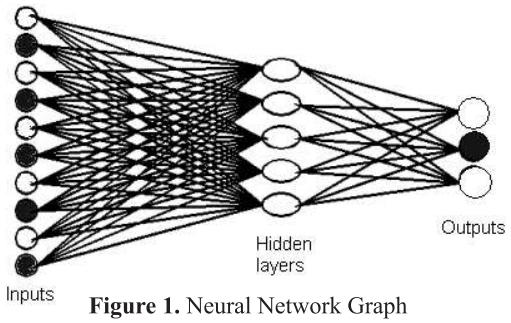

Figure 1. Neural Network Graph

3 Numerical Experiments

Tables 1 to 9 are some results of numerical experiments. These tables are composed of a variable *Input Vector* that represents the initial value of the vector used in each run, *Vector Output* is the optimal end value of final vector, *Function Value FO* is the value of the objective function at the optimal point.

Below in table 1 can be seen the results of a first run using model 1. In table 2 can be seen the results of a first run using model 2. In table 3 can be seen the results of a first run using model 3. In tables 1, 2 and 3 can compare the results using the same initial point. The most optimal result is the model 2 (second neural network).

Table 1. First Model – Vaz's trajectory No.1 [4]

Variables	Values
Input Vector	1.420310 1.420310 1.420310 1.420310 1.420310 1.420310 1.420310 1.420310 1.420310 1.420310
Output Vector	1.188585 0.974499 1.029212 0.988390 1.135009 1.188996 1.151483 1.020536 1.019925
Function Value FO	1.082927

Table 2. Second model – Trajectory with own values No.2 [13]

Variables	Values
Input Vector	1.420310 1.420310 1.420310 1.420310 1.420310 1.420310 1.420310 1.420310 1.420310 1.420310
Output Vector	0.099889 0.592549 1.020579 1.448669 1.311346 1.608055 0.842467 0.667546 0.280346
Function Value FO	1.081744

Table 3. Third model – Elke's trajectory No. 3 [5]

Variables	Values
Input Vector	1.420310 1.420310 1.420310 1.420310 1.420310 1.420310 1.420310 1.420310 1.420310 1.420310
Output Vector	1.093462 1.093511 1.093276 1.093472 1.093350 1.093851 1.093511 1.093340 1.093773
Function Value FO	1.093499

Below in table 4 can be seen the results of a second run using model 1. In table 5 can be seen the results of a second run using model 2. In table 6 can be seen the results of a second run using model 3. In tables 4, 5 and 6 can compare the results using the same initial point. The most optimal result is the model 2 (second neural network).

Table 4. First model – Vaz's trajectory No.4 [4]

Variables	Values
Input Vector	1.460300 1.386800 1.809300 1.826900 1.789600 2.070400 1.742300 1.558700 1.219900
Output Vector	1.193742 0.972270 1.064249 1.022088 1.179767 1.275979 1.190694 1.032919 1.006983
Function Value FO	1.120303

Table 5. Second model – Trajectory with own values No.5 [13]

Variables	Values
Input Vector	1.460300 1.386800 1.809300 1.826900 1.789600 2.070400 1.742300 1.558700 1.219900
Output Vector	0.116333 0.606328 0.860627 1.281481 1.159495 1.340740 0.710066 0.610640 0.362755
Function Value FO	0.947999

Table 6. Third model – Elke's trajectory No 6 [5]

Variables	Values
Input Vector	1.460300 1.386800 1.809300 1.826900 1.789600 2.070400 1.742300 1.558700 1.219900
Output Vector	1.093462 1.093511 1.093314 1.093493 1.093378 1.093819 1.093521 1.093345 1.093771
Function Value FO	1.093509

Below in table 7 can be seen the results of a third run using model 1. In table 8 can be seen the results of a third run using model 2. In table 9 can be seen the results of a third run using model 3. In tables 7, 8 and 9 can compare the results using the same initial point. Finally, the most optimal result is the model 2 (second neural network).

Table 7. First model – Vaz's trajectory No.7 [4]

Variables	Values
Input Vector	1.736200 1.614400 1.601800 1.569900 1.586400 1.614500 1.633300 1.507000 1.679300
Output Vector	1.229691 0.987925 1.045206 1.000451 1.155006 1.215173 1.177244 1.028245 1.037305
Function Value FO	1.102050

Table 8. Second model – Trajectory with own values No 8 [13]

Variables	Values
Input Vector	1.736200 1.614400 1.601800 1.569900 1.586400 1.614500 1.633300 1.507000 1.679300
Output Vector	0.229783 0.512740 0.945951 1.387158 1.243050 1.528205 0.754887 0.631899 0.173850
Function Value FO	1.017878

Table 9. Third model – Elke's trajectory No 9 [5]

Variables	Values
Input Vector	1.736200 1.614400 1.601800 1.569900 1.586400 1.614500 1.633300 1.507000 1.679300
Output Vector	1.093475 1.093515 1.093285 1.093475 1.093356 1.093847 1.093515 1.093342 1.093770
Function Value FO	1.093503

These results were compared with values obtained with other authors to solve the same problem and had noticed that we have good approximations (see *Function Value FO* in tables). This indicates that actually has succeeded to optimize the trajectory of an automatic manipulator using Neural Networks.

4 Conclusions

One of the major achievements has been to use concepts of semi-infinite programming and special Neural Networks for solving problems related to robotics.

Results obtained with Neural Networks by means of the computing tool prototype base don MATLAB, are appropriate for primary target of the research. Despite it is recommended for later studies, to use exactly initial and end points of trajectory in order to consider more appropriately, level of obtained improvement, since these comparisons were made starting off of itself model of trajectory and having like primary target minimize manipulator's time of route.

The use of MATLAB like programming tool, optimization, calculation and interface, facilitates the development to solve mathematical problems of high complexity, since development platforms such as C++, Java, among others, require of programming of mathematical tools that MATLAB already has implemented in his toolboxes, which allows saving effort of programming and lines of code.

Recent research opens the doors for development this thematic in the future, based mainly in solution generalization to problem created, in search an optimal trajectory for different types from robots, in addition to its physical implementation in a real robot, where it is necessary to consider a sizing of all the mechanical parameters and control, applied in a specific solution within an objective function.

References

[1] Fedossova A., Kafarov V., Mahecha Bohórquez D.P., Outer Approximations Algorithm for the Air Pollution Control Problem. 2004..

[2] Fedossova A., Stochastic outer approximations algorithms for convex semi-infinite programming problems, Ph.D. Thesis, Moscow State University, Moscow, 2000, (in Russian).

[3] Volkov Y.V., Zavriev S.K., A general Stochastic Outer Approximations Method, SIAM J. Control Optim., 35 (4), 1997, pp. 1387-1421.

[4] A. Ismael F. Vaz, Applications, methods and tools y for SIP, Ph.D. Thesis. Minho University, 2003, (in Portuguese).

[5] Elke Haaren - Retagne. A semi-infinite programming algorithm for trajectory planning, Ph.D. Thesis. Trier University, 1992.

[6] Belén Melián, José A. Moreno Pérez, J. Marcos Moreno Vega. Metaheuristics: A global view, Spain - 2003.

[7] Barbolla, Rosa. Optimization: questions, exercises and applications, Prentice Hall - 2001 (in Spanish).

[8] Ismael Vaz, Robot trajectory planning with semi-infinite programming, SIAM Journal on Optimization, vol. 9, no. 2, 2001.

[9] Carrillo Esneider y Mora Gleimer. Outer Approximations Algorithm for minimizing costs of Air Pollution Control Problem. Bucaramanga, 2003, p.138-139. Autonomous University of Bucaramanga. School of Systems Engineering.

[10] Fedossova A., Kafarov V., Mahecha Bohórquez D.P., Outer Approximations Algorithm for the Air Pollution Control Problem. Colombian Journal of Computation. Vol. 4 No. 2, 2003.

[11] Chi, Mei-Hsiu, Linear semi-infinite and nondierentiable programming methods for robot trajectory planning problems. Ph.D. Thesis, Iowa University, 1991.

[12] Medvedev, V.C.; Potemkin, V.G., Neural Network, MATLAB 6.0. Dialog- MIFI, Moscow, 2002 (in Russian).

[13] Fedossova A., Santoyo J. Stochastic Algorithm for Search an Optimal Trajectory of an Automatic Manipulator. Colombian Journal of Computation. Vol. 8 No. 2, Bucaramanga, December 2007