Automated Controller Gain Tuning of a Multiple Joint Robot Based on Modified Extremum Seeking Control

Michael Chan*, Kyoungchul Kong**, Masayoshi Tomizuka***

*Mechanical Engineering Department, University of California, Berkeley, CA 94720, USA, (e-mail: michael.l.g.chan@berkeley.edu)
** Mechanical Engineering Department, University of California, Berkeley, CA 94720, USA, (e-mail: kckong@me.berkeley.edu)
*** Mechanical Engineering Department, University of California, Berkeley, CA 94720, USA, (e-mail: tomizuka@me.berkeley.edu)

Abstract: Nonlinear programming (NLP) is a field in mathematics that focuses on iteratively decreasing an arbitrary cost function. In this paper, a NLP approach based on extremum seeking control (ESC) is modified for the purpose of autonomously tuning the controller gains of robots. The effectiveness and flexibility of the approach is demonstrated through experimental results on a six degrees-of-freedom (DOF) industrial robot. Both single joint, and multiple joint tuning experiments were performed.

1. INTRODUCTION

Industrial robots have been used for several decades. More recently, these robots have expanded to encompass a large portion of the electronics industry. Unlike traditional robotic applications, such as the automotive industry, consumer electronics generally operate on a shorter product cycle; hence the robots used in this industry are subjected to more frequent process changes. Moreover, these robots are also expected to perform their tasks with accuracy and precision, which is challenged by variable environments such as the operation temperature, lubricant condition, and so on. Therefore, the performance of these manipulators should be continuously monitored and the controller gains need to be re-tuned after each process change. In an environment with frequent process changes, manual retuning of these robots becomes an expensive and time-consuming process. Development of an online automatic gain tuning algorithm can mitigate such problems and may play a crucial role for robot manufacturers to remain profitable in the modern competitive environment.

There are many existing techniques to tune controller gains (Astrom et. al., 1993). Many classical techniques provide controller parameters that minimize a desired objective function based on a mathematical model, e.g., linear quadratic control (Franklin et. al., 2006). These methods, however, depend on the accuracy of the model parameters and may require an additional tuning process in order to obtain the desired performance in practice. An alternative approach is to experimentally obtain controller parameters with minimal knowledge on the system model.

Iterative feedback tuning (IFT) is an example of such a method (Hjalmarsson et.al., 1998). IFT has been shown to be an effective technique in various applications. The approach, however, is not without limitations. Restricted cost function design and dependence of system linearity are primary drawbacks of IFT. Since robot manipulators exhibit nonlinear dynamic characteristics due to large friction introduced by gear-heads and time-varying inertias that depend on the robot posture, IFT may not be the ideal technique for tuning the controller parameters of robots.

In this paper, a controller tuning approach based on nonlinear programming (NLP) is proposed and applied to an industrial robot manipulator. NLP is a field in mathematics that focuses on developing algorithms to minimize arbitrary cost functions. Note that the solutions to these optimization processes can be subjected to both equality and inequality constraints. Due to such flexibilities of NLP, it has already been applied to various applications in control systems; for example, computing the controller gains to minimize the H∞ norm or identifying unknown plant parameters through transfer function fitting. While the IFT technique also follows the principle of NLP, it sets a restriction in cost function design and assumes system linearity, namely in the process of estimating the output gradient, whereas the proposed method utilizes NLP for general cost functions without any assumptions in system characteristics.

The proposed method minimizes cost functions in an iterative manner. More specifically, the NLP algorithm used in the proposed gain tuning method adjusts the free variables (i.e., controller gains) so that the cost function decreases for each iteration. There are many different methods for solving NLP problems, such as sub-gradient methods, Lagrangian multiplier methods, and branch and bound techniques (Bertsekas, 1999, Boyd and Vandenberghe, 2004). The majority of these methods, however, rely on a family of descent algorithms known as gradient methods. There are many variations of gradient methods: e.g., the steepest descent method and the Newton method.

Most gradient method variants require knowledge of the cost function gradient. In the case of controller tuning, it is...
generally not feasible to have a closed-form expression for such a quantity since it would require exact knowledge of the plant. Instead, a real-time perturbation-based method for obtaining the gradient is used. A real-time optimization scheme based on a perturbation method is often called extremum seeking control (ESC) (Ariyur and Krstic, 2003). The basic concept of the automatic gain tuning method proposed in this paper is similar to ESC, but it utilizes peak filters to obtain the gradient from the perturbed cost value, while conventional ESC uses wash-out filters. The use of peak filters reduces the number of parameters to be determined for the optimization process.

The effectiveness of the proposed approach is validated by experimental results in this paper. The experiment is performed on a six degrees-of-freedom industrial robotic manipulator. Both single joint and multi joint tuning experiments were performed to demonstrate the flexibility of the proposed method.

The remainder of the paper is as follows: Section 2 talks about the experimental setup. Section 3 details theory behind the NLP approach. Topics in this section include information regarding gradient estimation and cost function design. Section 4 presents experimental results for both the single joint and multiple joint tuning process. And finally, Section 5 summarizes the findings of this paper.

2. EXPERIMENTAL SETUP

The experiments in this paper are performed on a FANUC M-16i/B industrial robot shown in Fig 1. The M-16i/B robot has six joints driven by indirect drive mechanisms. Each of the joint actuators has a built-in motor encoder. Further technical details about this robot can be found in (FANUC Robotics, 2010).

2.1 Robot Controller

The M-16i/B robot is equipped with a commercial controller that is capable of position and velocity feedback control. The M-16i/B utilizes a decentralized controller scheme. The controller structure for each joint is given in Fig 2. Notice that the controller scheme contains both an inner velocity feedback loop and an outer position feedback loop.

Fig 1 FANUC M-16i/B experimental robot. The six joints are labeled from J1 through J6

Fig 2 Decentralized controller structure for each joint

2.2 Real-Time Implementation Layout

In real-time systems, it is critical that hardware performance is not hindered by software computational limitations. Thus it is important to ensure that the software runs as quickly and consistently as possible. In this paper, all components of the algorithm are implemented on a Windows platform running MATLAB. This platform, however, does not guarantee a real-time environment due to unexpected interferences from virus scanning software and other event logging processes.

To overcome these problems, two computers are used for the real-time control system setup. A host computer operating the Windows platform is first used to design the NLP algorithm. Once this is done, the desired control algorithm is uploaded to another computer, called the target computer, which is running MATLAB’s xPC Target software. The xPC Target is MATLAB’s real-time implementation software and is designed to efficiently perform real-time processing. The sampling time for the xPC Target is 0.5ms. Additionally, the robot motor positions, motor velocities, and motor currents are available to the target computer through a high-speed serial bus (HSSB) interface. The connection between the host and target computer is severed whenever the robot is in motion.

Since the proposed NLP algorithm is iterative, the connection between the host and target computers is reestablished after each iteration. This is done so that all the post processing can be done on the host computer. Ideally, the post processing should also be handled by the target computer, but this task is transferred to the host computer here for debugging purposes.

3. REAL-TIME OPTIMIZATION FOR AUTOMATIC CONTROLLER GAIN TUNING

3.1 Nonlinear Programming

On a fundamental level, the goal of NLP is to minimize an arbitrary cost function. For the purpose of controller gain tuning, it is important to properly select the cost function such that minimizing such a quantity properly enhances the performance of the robot manipulator. For example, if tracking performance is only of interests, the cost function can be designed to be a weighted sum of the tracking error norms. Additionally, since NLP operates iteratively, it is also important to properly select an iteration period for evaluating the cost function. Since most industrial robots perform the same task repetitively, it is intuitive to select each NLP iteration to coincide with a single completion of the robot’s desired task.
For an arbitrary cost function $J(x_k)$ at iteration $k$, it is desirable to select $x_{k+1}$ such that

$$J(x_{k+1}) < J(x_k)$$

where $x_k$ is the optimization variables at the $k$th iteration, i.e., $x_k$ is the controller parameters to be tuned by the proposed gain tuning method. A general class of algorithms used to enforce (1) is known as gradient methods. Using gradient methods, the parameter $x$ is updated in the following form

$$x_{k+1} = x_k - a_k D_k \nabla J(x_k)$$

(2)

where $a_k$ and $\nabla J(x_k)$ denote the stepsize and cost function gradient at iteration $k$, respectively. $D_k$ is a positive definite matrix that can be used to scale the descent direction to improve the convergence of the algorithm. As previously mentioned, knowledge of $\nabla J(x_k)$ is the crux of using gradient methods. In most practical cases, it is impossible to obtain a closed expression for $\nabla J(x_k)$.

### 3.2 Gradient Estimation

In this paper, a perturbation approach proposed by Kong and Tomizuka (Kong et al., 2008) is used to approximate the gradient of the cost function. If the controller gains are perturbed, then the resulting cost function can be expressed by a Taylor series expansion as

$$J(x_k + \epsilon_k) = J(x_k) + \epsilon_k^T \nabla J(x_k) + H.O.T.$$

(3)

where

$$x_k = [K_p \ K_v \ K_i]^T \in \mathbb{R}^3$$

(4)

$$\epsilon_k = [a_1 \sin(\omega_1 k) \ldots a_3 \sin(\omega_3 k)]^T \in \mathbb{R}^3$$

(5)

where $a_i$ and $\omega_i$ are the perturbation amplitude and frequency respectively. $J$ is the number of joints being tuned. One requirement for the perturbation frequencies is that they are independent of each other, e.g. $\omega_k \neq \omega_i \forall i,j,k$ and $\omega_i \neq \omega_j \forall i \neq j$. By choosing independent frequencies, the perturbation frequencies in (3) are isolated to the second term, because all the higher order terms past the gradient contain cross frequency terms. These cross frequency terms, e.g., $\sin(\omega_i) \sin(\omega_j) \sin(\omega_k)$, have frequency contents $|a_i \pm a_j|$ and $|a_i \pm a_j \pm a_k|$, respectively. This allows the $i^{th}$ element of the second term in (3) to be extracted by passing the perturbed cost function through a bandpass filter with a narrow pass band centered about $\omega_i$. This process is depicted in the left half of Fig. 3.

The bandpass filters used in this paper all have transfer functions in the form

$$p_i(z) = \frac{b_{oi}(z-1)}{z^2 + a_{1i}z + a_{0i}}$$

(6)

where

$$a_{0i} = e^{b_{oi}}$$

$$a_{1i} = -2e^{-0.5b_{oi}}\cos\left(\frac{1 - b^2}{2} \omega_i\right)$$

$$b_{oi} = \frac{e^{j2b_{oi}}} {e^{j2b_{oi}} - 1}$$

Note that for a given $\omega_i$, $b$ is the only design variable for the bandpass filters. The magnitude of $b$ is inversely proportional to the width of the filter pass band. In this paper, the filter coefficient is selected such that $b = 0.1$. The frequency response of such a filter is given in Fig. 4. A more detailed explanation of the filter design process can be found in (Kong et al., 2008).

Once each component of the gradient is isolated by their respective bandpass filters, the final step is to extract out only the gradient component of each term. By modulating each gradient component by its respective perturbation value, the gradient term can be expressed as

$$a_i \sin(\omega_i k) \nabla J_i(x_k) \cdot a_i \sin(\omega_i k) = a_i^2 \sin(\omega_i k)^2 \nabla J_i(x_k)$$

(7)

$$a_i^2 \nabla J_i(x_k) \frac{1}{2} (1 - \cos(2\omega_i k))$$

From (7), the gradient term can be extracted by filtering each modulated term by a low pass filter whose amplitude is $2/a_i^2$ with a cutoff frequency lower than the two times the lowest perturbation frequency. Once each of these terms is modulated and filtered, they can be concatenated to form the gradient vector. This process is shown in the right half of Fig. 3. As a final note, an inherent assumption made in this section is that the gain tuning environment is time-invariant or slowly time-varying, i.e., the dominant factor that influences the value of the cost function should be the controller gains. Consequently, this assumption also mandates that the nonlinearities of the system, such as Coulomb friction in the gear train, and other disturbances remain relatively consistent.
between iterations. Otherwise, the time varying nature of the system would invalidate (3). However, this condition is easily satisfied during gain-tuning processes of robots when they repeat the same tasks. Furthermore, if each tuning iteration is selected to be a single completion of the robot’s task, then the nonlinearities experienced by the robot for each iteration should remain relatively constant as well.

### 3.3 Cost Function Selection

Once the gradient is obtained, (2) can be used to update the controller gains to iteratively decrease the cost function. The cost function should be selected based on the desired performance objective. This is similar to the process of selecting appropriate weighting matrices in linear quadratic (LQ) controller design. Like LQ, the objective functions for NLP can also balance quantities such as control effort against tracking performance.

The proposed gain tuning method does not rely on a model and thus can be applied to an extensive range of problems. However, the control performance optimized by the proposed method is highly dependent on the cost function design. Furthermore, the choice for the stepsize, \( \alpha_k \), is also heavily dependent on the choice of cost function. If the stepsize is selected appropriately such that \( f(x_{k+1}) - f(x_k) < 0 \) \( \forall k \), stability of the system can be guaranteed. (note: assuming that the cost function includes the physical states of the robot, the cost function \( f(x_k) \) can be treated as a Lyapunov function candidate.). In practice, however, it is difficult to achieve such objectives; the cost function often blows up. Namely, it is important to select a cost function such that stability robustness of the tuned controller gains is considered during the optimization process. An approximate robot model can be used for this purpose. Although this process re-introduces a mathematical model into the optimization process, the accuracy of the mathematical model is not necessarily required. The model can be used to approximate the closed loop poles of the system at iteration \( k + 1 \) at which then the stepsize is selected to enforce the real parts of every pole to remain strictly negative. While it is desirable for the model to be locally accurate with respect to the initial controller gains and posture, the role of the model is to provide an upper estimate on the stepsize. Hence the stepsize used in practice may be significantly smaller than the stepsize predicted by the model if the accuracy of the model is poor. This allows for a tradeoff between the model accuracy and the algorithm convergence rate. The constraints imposed by the model can be treated as inequality constraints in the NLP problem. They can then be lifted into the cost function with barrier functions. Barrier functions are commonly used in a class of NLP algorithm known as interior point methods. These barriers have the property of becoming extremely large when the constraints become close to being violated. Following this formulation, the cost function used in this paper is in the form

\[
J(K_k) = \sum_{i=1}^{p} e_i^TW_{i,k}e_i + \sum_{j=1}^{M} \frac{\lambda_{j,k}}{\Re \{c_{j,k}\} + \delta_{j,k}}
\]  

where the index \( i \) represents the different measurement components used to evaluate the cost function, e.g., \( e_{1,k} \) can be a vector containing the total position error for each joint at the \( k^{th} \) iteration and \( e_{2,k} \) can represent velocity error, and so on. \( W_{i,k} \in \mathbb{R}^{N \times N} \) is a diagonal positive semidefinite weighting matrix where \( N \) is the total number of joints in the robot. The index \( j \) counts the total number of closed loop poles, \( \lambda_{j,k} \). The parameters \( \lambda_{j,k} \) and \( \delta_{j,k} \) are parameters that can be used to fine tune the sensitivity of the barrier functions. Note that when \( \delta_{j,k} = 0 \) then the second term in (8) will become large as \( \Re \{c_{j,k}\} \) approach zero. \( \delta_{j,k} \) essentially limits how close the closed loop poles can approach the imaginary axis. It is important to note that the choice for \( \delta_{j,k} \) must keep the initial controller gains feasible, e.g. if the initial controller gains are such that \( \max_j \Re \{c_{j,k}\} = -0.5 \) then \( \max_j (\delta_{j,1} < 0.5) \).

Intuitively, the choices for \( e_{i,k} \) are dependent on the available measurements. Since the M-16/B robot has built-in motor encoders that provide individual motor positions and velocities, the one norm of the motor position and velocity errors at each iteration are used for \( e_{1,k} \) and \( e_{2,k} \) respectively for the experimental study in this paper.

### 4. EXPERIMENTAL RESULTS

In this section, experimental results for single joint experiments and multiple joint experiments are presented. A single joint tuning experiment denotes that only a single joint is being tuned with the remaining five joints being fixed where as a multiple joint experiment tune multiple joints simultaneously. In these experiments, the parameter \( D_k \) from (2) is taken to be identity and the stepsize \( \alpha_k \) is selected such that the calculated closed loop poles remain strictly on the left side of the complex plane. In this paper, the linearized two mass model described in (Han et al., 2009) was used to approximate the closed loop poles for each joint of the robot. The transfer function of this model is given as

\[
G_m = \frac{\dot{\theta}_m(s)}{u(s)} = \frac{j L_c s^2 + d_j s + d_m s + k_j}{j m L_s^3 + j d_s^2 + j k_s s + d_m s + d_j s + k_j}
\]  

where

\[
J_d = J_m(d_j + d_i) + J_i\left(\frac{d_i}{N^2} + d_m\right)
\]

\[
J_k = J_m k_j + J_i k_j + (d_j + d_i)d_m + \frac{d_i d_j}{N^2}
\]

where \( J_m \) and \( J_i \) are the motor side and load side interia. \( d_m, d_j \) and \( d_i \) are the motor side, joint side, and load side viscous damping coefficients. \( k_j, u \) and \( \theta_m \) are the joint stiffness coefficient, motor torque input, and motor side position respectively. The values for these parameter are obtained by a MCG robot model.

The stepizes used for the experiments in this paper are selected based on a successive stepsize reduction algorithm. More specifically, a fixed initial stepsize, \( \alpha_{k,0} \), is used. Prior to updating the controller gains, the updated closed loop poles are calculated. If any of the poles were to violate the
barrier constraints, then the stepsize would be reduced by \( \frac{1}{2} \) before recalculating the updated poles. This process is repeated until the cost function is fully minimized and all the closed loop poles remain feasible under the barrier constraints. In the case where a feasible stepsize cannot be obtained after 250 successive reductions, the algorithm will then restart with a smaller initial stepsize.

4.1 Single Joint Tuning Experiment

In this section, two examples are given to demonstrate the effectiveness of the proposed method. On the FANUC robot, the larger joints (i.e., joints labeled as J1, J2, and J3 in Fig. 1) are less sensitive to variations of the controller gains than the smaller joints (J4, J5, and J6 in Fig. 1). The descriptions “larger” and “smaller” refer to both the physical size of the joints as well as load capacity. The joints with a larger motor and higher design load tend to have a larger stable range of controller gains when compared to the smaller joints. Furthermore, the two mass model used for stepsize selection seems to characterize the larger joints better than the smaller joints. As a result, the larger joints on the FANUC robot were considerably easier to tune than the smaller joints. To demonstrate the broad applicability of this approach, the examples in this section will tune J1 and J6 respectively.

J1 is the largest joint of the M-16iB robot. In this controller tuning experiment for a single joint, the cost function is selected to be

\[
J(x_k) = e_k^TW_1e_k + \sum_{i=1}^{n} \frac{\lambda}{\text{Re}(\epsilon_{i,k}) + \delta}
\]

where \( e_k \) and \( \dot{e}_k \) are the sum of the absolute values of position and velocity errors for J1 at the \( k^{th} \) iteration. The errors are calculated by taking the difference between the reference signals and the measurements from the J1 motor encoder. \( W_1, W_2, \lambda, \) and \( \delta \) were selected to be 1, 0.3, 1.0e-3, and 0.15 respectively. The stepsize was selected such that \( \alpha_{k,0} = 0.005 \). In the experiment, the robot was instructed to sweep a 22.5° arc at constant velocity over 0.5 seconds and then return to the original position. Fig. 5 plots the cost function versus iteration for the controller tuning experiment of the J1 joint. From this figure, it can be seen that the barrier functions did not play a significant role in the controller tuning process. Also note that about 35 iterations were needed before the cost function began to decrease significantly. This has to do with both the fact that this joint is not very sensitive and also that because the gradient estimation approach uses extensive filtering in the iteration domain, a few iterations are required before accurate filtering results can be obtained.

The joint labeled as J6 is the most distal joint of the M-16iB robot, which exhibits the smallest inertia among the six joints. The structure of the cost function is the same as (10) but the parameters \( W_1, W_2, \lambda, \) and \( \delta \) were selected to be 1, 0.3, 1.0e-2, and 0.5 respectively. Notice that \( \delta \) had to be increased significantly, which in turn increased the sensitivity of the barrier functions. This was to ensure stability during the gain tuning procedure. The stepsize for this experiment was selected such that \( \alpha_{k,0} = 0.0005 \). As far as the trajectory is concerned, the joint was instructed to sweep a 45° arc at constant velocity over 1.0 second and then return back to the initial position. Fig. 6 plots the cost function versus iteration for the controller tuning experiment. Unlike the case with tuning J1, the barrier functions played a significant role in the tuning procedure. It is important to point out that while the cost function began decreasing after 7 iterations, the cost function remained relatively constant after the barrier functions became significant. This is primarily due to the fact that the stepsize is related to the overall strength of the barriers. If the barriers get big, then stepsize becomes progressively smaller to avoid overstepping the barrier constraints. As a result, the activation of the barrier functions does not have any detrimental effects on the part of the cost function that represent the overall performance of the robot.

4.3 Multiple Joint Tuning Experiment

In this section, J1, J2, and J3 of the FANUC robot were tuned simultaneously. The objective function is still the same as (10) but the parameters \( W_1, W_2, \lambda, \) and \( \delta \) were selected to be the 3 × 3 identity matrix, 0.3W1, 1, and 0 respectively. The trajectory commanded J1, J2, and J3 to simultaneously sweep a 22.5° arc at constant velocity over one second before reversing the trajectory to return to the initial position. The stepsize in this experiment was chosen such that \( \alpha_{k,0} = [0.0002 0.0002 0.001]^T \) for J1, J2, and J3, respectively. Fig. 7 plots the time history of the position tracking performance for the three joints. The red line indicates the reference where as the blue and green lines indicated the actual robot tracking.
performance from the initial and final gains respectively. Note that in all cases, the final gains track the reference significantly better than the initial gains. Fig. 8 shows the effectiveness of the NLP approach. Note that the barrier functions did not play a significant role in this gain tuning process; hence the stepsizes were relatively constant. As a result, unlike Fig. 6, the cost function in this experiment continued to decrease through the entire tuning process.

In both the single joint and multiple joint tuning experiments, the tuned gains decreased the covariances of the position and velocity tracking error by between 50-70%.

5. CONCLUSIONS AND FUTURE WORKS
An automatic gain tuning approach was successfully implemented on an industrial robot in this paper. The approach draws upon a measurement-based gradient estimation technique proposed earlier (Kong et. al., 2008). Through the use barrier functions, the proposed scheme was shown to be robust enough to work in application. The feasibility of this approach was demonstrated by single and multiple joint tuning experiments. Due to the nature of nonlinear programming, it is important to note that the proposed algorithm only achieves a local minimum hence the performance can vary depending on the initial conditions.

Although barrier functions were able to stabilize the tuning process, these functions rely on an approximate model. If the model is poor, then the barriers can lead to very conservative results. Hence, it is desirable to incorporate data-based constraints rather than using the current model-based ones. This idea is a topic for future investigation.

6. AWKNOWLEDGMENTS
This work was supported in part by FANUC Ltd., Japan.

REFERENCES


![Fig. 7 Position time history plot for multiple joint tuning](http://www.fanucrobotics.com/file-repository/Datasheets/Robots/M-16iBT.pdf)

![Fig. 8 Plot of cost function versus iteration for multiple joint tuning experiment](http://www.fanucrobotics.com/file-repository/Datasheets/Robots/M-16iBT.pdf)