WEB ENABLED ROBOT DESIGN AND DYNAMIC CONTROL SIMULATION SOFTWARE SOLUTIONS FROM TASK POINTS DESCRIPTION

Tarek M. Sobh*, Bei Wang*† and Sarosh H. Patel‡

University of Bridgeport
Bridgeport, CT 06601 U.S.A.

Abstract: In this work, we propose a web-based solution for robot design and dynamic control simulation based on given task point descriptions. The software combines and utilizes the computational power of both the Mathematica and Matlab packages. Given the location and velocity of each task point, our approach formulates the complete design of a 3 DOF robot model by computing its optimal dynamic parameters such as link length, mass and inertia. Further more, our package suggests the optimal control parameters (Kp, Kv) for the dynamic control simulation.

Keywords: Robot design, modeling, manipulability, web-based, trajectory generation, optimization, control loop simulation, robot control, dynamic control simulation.

1. Introduction

A task point's description includes a set of desired positions of an end-effector in the physical space and velocities at these points at a particular instant of time. The problem being considered in this work is to obtain the optimal robot design and dynamic control strategy in such a way that the task can be carried out with maximum manipulability and minimum error in reaching the desired positions and velocities.

Current robotic researches based on task descriptions have focused on the trajectory generation [1, 4] and kinematic synthesis [2] without taking dynamic parameters into consideration. Some research emphasizes the design and numerical optimization of manipulability and cost function [2]. Our robot design and simulation software not only uses the kinematic parameters of task points in designing the manipulability function, but also utilizes the dynamic description of each task point (velocity, etc). Therefore, it offers the user a complete dynamic robot model instead of a simple kinematic description.

Instead of a windows application which requires local installation of Mathematica and Matlab, our web utility enables users to access computational services provided by the server. Therefore, local installations of the complicated mathematical packages are no longer needed.

2. Theory

2.1 Manipulability

A robot configuration can be said to be best suited for a particular task, if the manipulability or the dexterity of the manipulator at the set of task points is high. In the past few years numerous approaches have been proposed for calculating the manipulability. The manipulability of a manipulator at a particular point can be defined as the *"the ability of the manipulator to accelerate in all directions from that point"*.

Yoshikawa [3] suggested one such measure based on the volume of the manipulability ellipsoid as derived from manipulator kinematic properties i.e., the Jacobian. Yoshikawa's manipulability measure is based on kinematic data and it calculates the manipulability based on how far is the point from a singularity and thus, be able to exert forces and move uniformly in all directions.

$$\eta_{yoshi} = \sqrt{\det\left|J(q)J^T(q)\right|} \qquad \text{--- (1)}$$

Where $J(q)$ is the velocity Jacobian at that point and q is the joint variable vector.

2.2 The Cost Function

Deciding on the best possible geometric model for a manipulator is a very difficult problem in kinematics as well as mechanics. The equations describing the kinematic behavior of the links are nonlinear and have many variables in the order of thousands [2]. In some cases there might not be any closed solution, whereas in other cases there might be more than one

* Department of Computer Science and Engineering
† Contact author: beiwang@bridgeport.edu
‡ Contact author: saroshp@bridgeport.edu, Department of Electrical Engineering

closed solution [2]. One of the best ways (though not the fastest) to solve such a problem is using the theory of optimization. Using optimization, all the possible options are evaluated using a *Cost Function* [2] or *Objective Function* [9]. Depending on whether the Cost Function has to be maximized or minimized the best possible solution is chosen. Multiple factors influencing the modeling technique can be incorporated into the Cost Function.

The criteria used to form the cost function are:

1. Manipulability
2. Accuracy
3. Distance from the point

The cost function [2] is given by

$$F(K, q1, q2, \ldots qm) = \xi * L + \sum_{i}^{m} \frac{1}{w_i^2 + b} + \varepsilon * D_i^2 \qquad \text{--- (2)}$$

Where K is the DH parameter of the robot

$q1, q2 \ldots qm$ are the joint vectors of the task points

ξ is the dumping factor

w_i is the manipulability

ε is the weight factor

D_i is the distance between the point and the origin of the end-frame of the robot

b is the extra term to eliminate singularity

$L = \sum_{\substack{j \\ d_j \notin q}}^{N} (a_j + d_j)$ where a_j and d_j are the length and offset Denavit-Hartenberg parameters respectively

The optimization parameters are all the Denavit-Hartenberg parameters other than the joint variables.

2.3 Optimizing the Cost Function

The cost function is a minimizing function. Minimizing the function provides the optimal values for the DH table. The function is optimized using the *steepest descent algorithm* [2], which finds the minima by searching in the direction opposite to the gradient. If the range of the function does not contain negative values, the function always converges, except in a few rare cases when the gradient disappears [2].

2.4 Calculation of Dynamic Parameters

The Robot Design module calculates the manipulator DH table. But the dynamic parameters like Mass, Center of Gravity and Inertia are required for the closed loop control simulation of the robot. Our software simulation package calculates these on the following assumptions:

1. The manipulator links are solid and cylindrical in shape.
2. All links have uniform density (uniform mass distribution).
3. All the links are made of the same material.
4. There are a finite number of actuators and sensors with known specifications that can be used in the design.

The user is requested to enter the link radii. The link radii are necessary for calculating the dynamic parameters of the manipulator. The parameters are calculated as follows:

1. Mass - The user is requested to specify the link radius. Knowing the volume and density the mass of each link can be easily calculated.

$$m_i = \pi r_i^2 a_i d \qquad \text{--- (3)}$$

where m_i is the mass of the ith link,

r_i is the radius of the ith link,

a_i is the length of the ith link,

d is the density of the link material

2. Center of Gravity – The center of gravity is calculated geometrically with respect to the link coordinate frame.
3. Inertia – The Inertias of the links has to be calculated about the respective Center of Gravities. Since the links are considered to be cylindrical, the Inertia about the axis of a cylinder is given by:

$$I_1 = \frac{1}{2} m_i r_i^2 \qquad \text{--- (4)}$$

Using the perpendicular axis theorem the Inertia along the other two axes is given by:

$$I_2 = I_3 = \frac{1}{4} m_i r_i^2 \qquad \text{--- (5)}$$

where m_i is a Mass of the ith link

r_i is the Radius of the ith link.

2.5 Trajectory Generation

The main purpose for trajectory generation is to produce a timed path, which can be tracked by a manipulator. All the task points are linked to generate a trajectory from the initial to the final point on a common time scale. Our implementation uses a seven-degree polynomial to generate the trajectory. The control loop is implemented over to support this trajectory.

2.6 The PD Control Loop

Most of the control algorithms use proportional derivative (PD) control. In software a local PD control loop [8] is applied to each link independently. It is advantageous to use a PD control loop for the following reasons [6]:

1. It is simple to implement.
2. Since it involves few calculations, it is ideal for real time control provided that the choice of Kp and Kv is appropriate to give optimum control.
3. The behavior of the system can be controlled by changing the feedback gains.

The torque to be applied to the manipulator in order to attain the desired position is calculated using Forward Dynamics [7]. The feedback loop in the case of a computer simulation of the control loop encodes Inverse Dynamics, where as in the case of real time control, the sensors provide the feedback. Figure 1 below displays a block diagram commonly found in robot prototyping research [5].

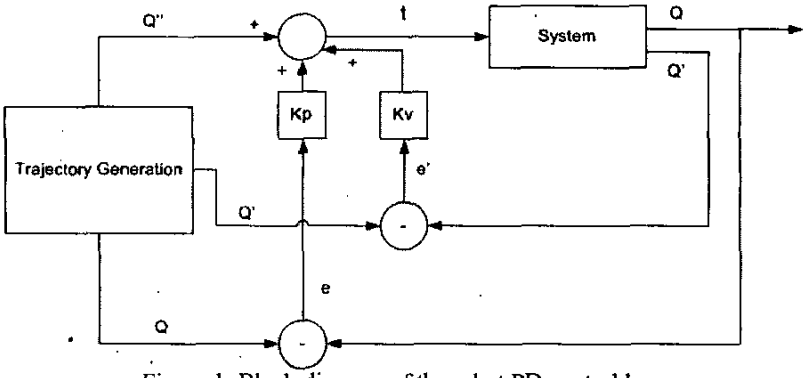

Figure 1: Block diagram of the robot PD control loop

Where Kp is the proportional gain, Kv is the derivative gain, e is the error in position and e' is the error in velocity.

At every cycle in the Control Loop the manipulator state is compared with the desired position. This gives the *Position Error* at that instant. Similarly, the *Velocity Error* is also calculated, by comparing the instantaneous velocity of each of the links with the desired velocity at that instant. The torque to be applied at that instant is given by:

Torque = (Kp * *Position Error*) + (Kv * *Velocity Error*)

2.7 Optimization of Kp and Kv

The proposed software optimizes the values of the loop gains (Kp and Kv) by trying all possible values within a given range. The user can also specify this range. The criteria for deciding the optimum values for Kp and Kv is that the sum of the square of errors about the desired trajectory should less than a specified threshold. The user can also run the simulation for particular value of Kp and Kv.

3. The Software Package

Our software package includes four major modules: a web interface, a kinematics design module, a dynamic design module and a dynamic control simulation module. It is intended that a user would specify the coordinates of each task point, the velocity of the end-effector at each point and the specific time to reach each point. Through kinematic and dynamic computations, our implementation desires a complete optimal robot model with dynamic parameters. It further performs the dynamic control simulation of the obtained robot model and suggests the best control parameters, such as Kp and Kv.

3.1 The Web Interface

The web interface is developed using Java Server Page Technology and Servelet. It is the central control module that communicates with the Mathematica and Matlab applications running on the server. The basic aim is to interact with the user and invoke the respective modules. It uses a modified version of JLink to communicate with Mathematica and a modified version of JMatlink to communicate with Matlab. It controls the Mathematica kernel from JSP pages and Matlab from JMatservlet (part of JMatLink).

The web interface communicates with the kinematics design module and allows the user to enter task point variables, such as coordinates, velocities and time scale. It also interacts with the dynamic design module and a dynamic control simulation module based on user specified radii for each link and range of dynamic simulation variables (Kp, Kv). It provides the web users with a complete design of the kinematic and dynamic modules, and also the optimized PD control variables. The web implementation converts the detailed control simulation results, such as the desired path vs. obtained path curves, into jpeg images so that they can be easily viewed on the web page.

3.2 Kinematic Design Module

The kinematic design module generates the best kinematics robot configuration with the maximum manipulability at user-specified task points. With modifications based on the kinematics synthesis package [2] build on top of *Robotica* package (v.3.60, Copyright 1993 Board of Trustees, University of Illinois), this module applies numerical optimization in constructing a kinematics robot model. The user (a robot designer) enters a set of task points into this module and obtains a robot configuration in the form of a DH table, describing the optimal kinematics properties of the three-link robot.

The main Mathematica (V. 4.1, Wolfram Research Inc. 2002) procedure that triggers the optimization event is defined as:

DesignRobot [task_points, configuration, precision, size_dump, file_name]

Where task_points is a matrix with xyz coordinates of task points; configuration is a string of 'R's and 'P's describing prismatic or rotational joints. For example, "RRR" stands for an articulated manipulator. Our module tries all possible joints combinations for a 3-link robot, achieving the best configuration. Precision and size_dump handle the precision and dimensions of the robot. At last, file_name specifies the location in which the robot configuration file is stored [2].

3.3 Dynamic Design Module

While the Kinematic model is only concerned with the kinematic properties of the robot, the dynamic module involves user input as task point's velocities and operation time. Running in the MATLAB (V. 6.1, MathWorks Inc. 2001) environment, the input to this module is the file, which is generated by Mathematica. This module reads the robot configuration file generated by the kinematic design module.

Using the DH parameters and the formulae discussed in the previous section, this module generates the Dynamic parameter matrix 'dyn'. The dynamic matrix is a (n x 20) matrix where n is the number of degrees of freedom of the manipulator. This matrix defines all the dynamic parameters of the manipulator. The structure of the dyn matrix is as follows [12]:

1	alpha	Link twist angle
2	a	Link Length
3	theta	Link rotation angle
4	d	Link offset distance
5	sigma	Joint Type, 0 for revolute, none-zero for prismatic
6	mass	Mass of the link
7	rx	Link CoG with respect to the link coordinate frame
8	ry	
9	rz	
10	lxx	Elements of link inertia tensor about the link CoG
11	lyy	
12	lzz	
13	lxy	
14	lyz	
15	lxz	
16	Jm	Amateur inertia
17	G	Reduction gear ratio. Joint speed / link speed
18	B	Viscous friction, motor referred
19	Tc+	Coulomb friction (positive rotation), motor referred
20	Tc-	Coulomb friction (negative rotation), motor referred

Table 1: Structure of the dyn matrix

In order to generate the dynamic parameters the user is asked to input the radii of the links. Based on the radii, the mass of the links, and successively the center of gravity (CoG) and Inertia are calculated. Finally, a Robot object is created in MATLAB using the dynamic parameters calculated; this robot has all the dynamic properties as desired.

3.4 Dynamic Control Simulation Module

Programmed in the MATLAB environment, this module deals with the control simulation of the dynamic robot model. The user is asked to enter the co-ordinates of points with respect to a time frame and the velocities at those points. The points specified by the user are linked using a seventh order polynomial in order to generate a trajectory. The PD control loop simulation runs using this generated trajectory.

The user has a choice of either specifying the values of Kp and Kv or else specifying a range of values for Kp and Kv and the step increment. In the second case the software finds the optimum values for both Kp and Kv. The same is the case with the update frequency, where the user can either specify a value or range with the increment, in which case the software decides the optimum value.

4. Results

This section includes results from a complete sample program run. The user specifies a number of task points (figure 2). Then the user specifies coordinates and velocity of each task points with respect to a time scale (figure 3). Figure 4 and figure 5 are part of the dynamic model generation and dynamic control simulation, showing the best Kinematic and Dynamic model (DH table and Dynamic table), as well as the optimal values of Kp and Kv for each link. Figure 6 to figure 11 are the results of the dynamic simulation, displaying the desired path and obtained path for each link.

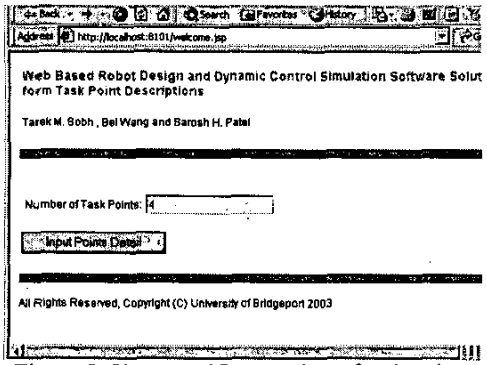

Figure 2: User specifies number of task points

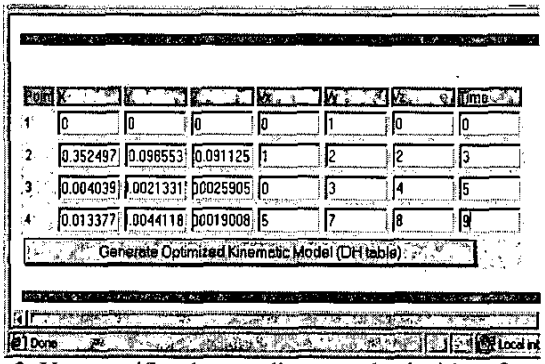

Figure 3: User specifies the coordinates and velocities of each task points with respect to a time scale

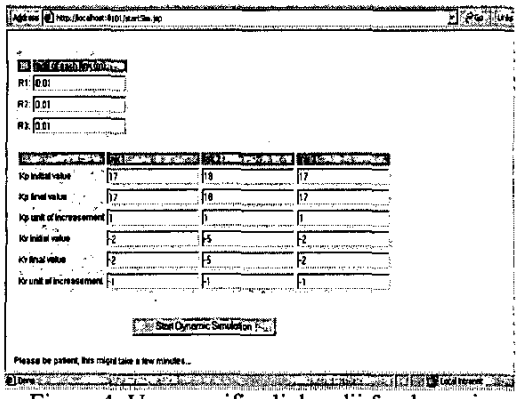

Figure 4: User specifies link radii for dynamic model generation, and Kp, Kv initialization for dynamic PD control simulation

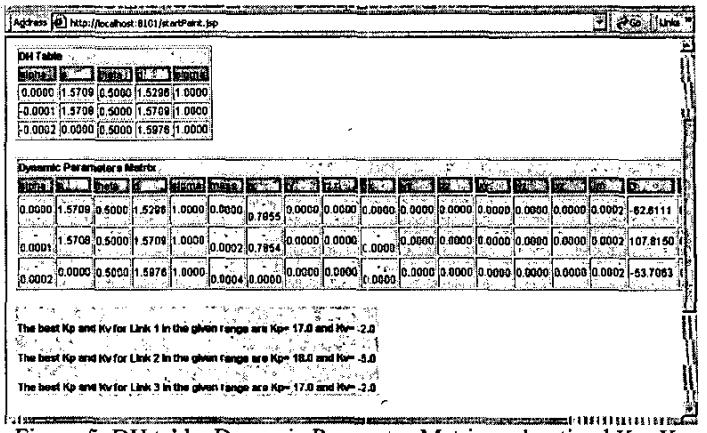

Figure 5: DH table, Dynamic Parameter Matrix and optimal Kp, Kv values for each link

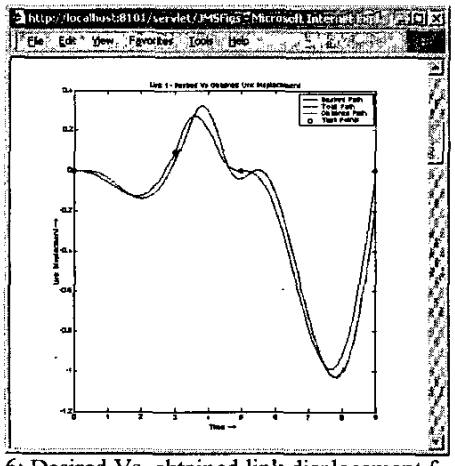

Figure 6: Desired Vs. obtained link displacement for link 1

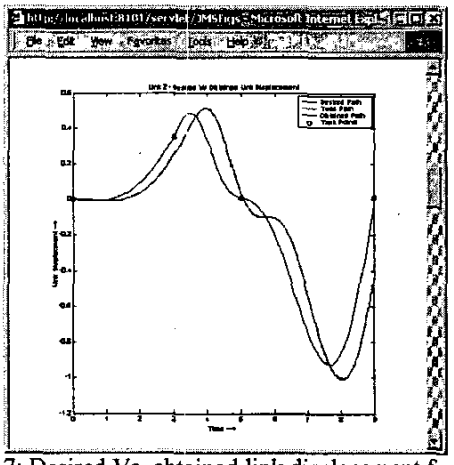

Figure 7: Desired Vs. obtained link displacement for link 2

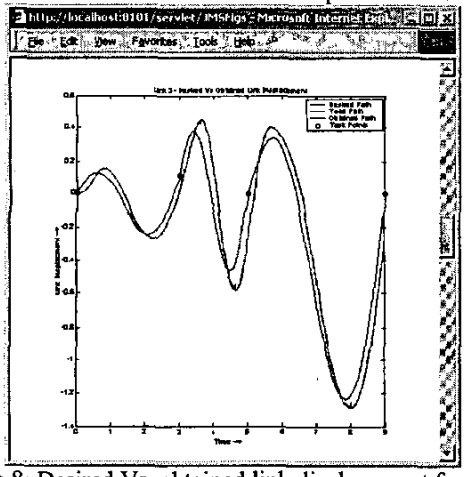

Figure 8: Desired Vs. obtained link displacement for link 3

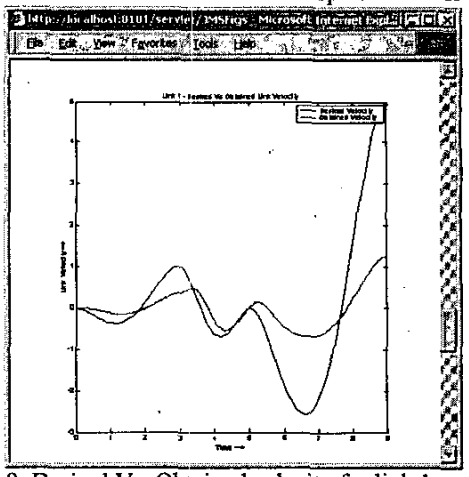

Figure 9: Desired Vs. Obtained velocity for link 1

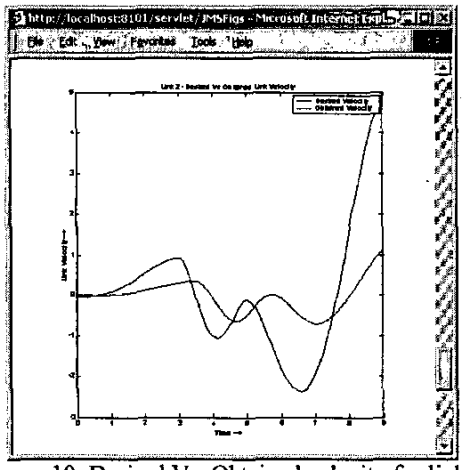

Figure 10: Desired Vs. Obtained velocity for link 2

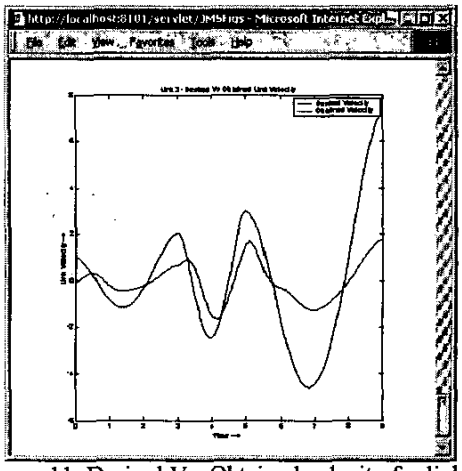

Figure 11: Desired Vs. Obtained velocity for link 3

5. Conclusion

In this paper we have presented a web-enabled software utility for the complete design and optimum dynamic control of a manipulator from a task points' description. The software generates the basic configuration of a manipulator based on user specified task points, in order to attain the greatest manipulability in the workspace. This software also provides the optimum values of Kp, Kv for optimum dynamic control.

6. Future Development

The potential future enhancements to this software include:

1. Building better objective functions, by including more design criteria and assigning weighing coefficients to each of them according to their importance.
2. Customizable objective functions, by which the user can design a manipulator by defining his/her own design criteria.
3. Implementation of advanced and better trajectory generation algorithms.
4. Faster algorithms for calculation of inverse kinematics.
5. Designing a numerical solution package for inverse kinematics for a few common robot models.
6. Implementation of Proportional Integral Derivative (PID) control in addition to Proportional Derivative (PD) control, in order to further minimize the error.

References

1. Lloyd J., Hayward V. "A Discrete Algorithm for Fixed-path Trajectory Generation at Kinematic Singularities", *IEEE Int. Conf. on Robotics and Automation*, Minneapolis (1996)
2. Sobh T. and Toundykov D. "Kinematic Synthesis of Robotic Manipulators from Task Descriptions". To appear in *IEEE magazine on Robotics and Automation*, summer (2003)
3. Yoshikawa T. "Manipulability of Robot Mechanisms". *International Journal of Robotics Research*, vol.4, pp.3--9 (1985)
4. Pires E., Machado J. and Oliveira P. "An Evolutionary Approach to Robot Structure and Trajectory Optimization", *10ih International Conference on Advanced Robotics*, pg. 333-338, Budapest, Hungary, August (2001)
5. Sobh, T., Dekhil, M., Henderson T., and Sabbavarapu A. "Prototyping a Three Link Robot Manipulator", *International Journal of Robotics and Automation*, Vol. 14, No. 2 (1999)
6. Dekhil, M., Sobh T., Henderson T., Sabbavarpu A. and Mecklenburg R. "Robot manipulator prototyping *(Complete design review)*", University of Utah (1994)
7. Spong M. and Vidyasagar. "Robot Dynamics and Control", Wiley, New York (1989)
8. Grigorian M., Sobh T. "Design-Simulation-Optimization Package for a Generic 6-DOF Manipulator with a Spherical Wrist", to appear in *Journal of system Analysis, Modeling and Simulation*, April (2003).
9. Pamanes J., Montes P., Cuan E., Rodriguez F. "Optimal Placement and Synthesis of 3R Manipulator", *International Symposium of Robotics and Automation*, Monterrey, Mexico (2000)