AC 2008-266: CONSTRAINED OPTIMIZATION TECHNIQUES IN DYNAMIC SYSTEMS

Horacio Vasquez, UTPA

Robert Freeman, UTPA

Gerhart Hanson, UTPA

Constrained Optimization Techniques in Dynamic Systems

Abstract

This paper presents applications of constrained optimization techniques to common dynamic systems studied in undergraduate Mechanical Engineering courses. Practical examples have been used to introduce undergraduate students to optimization methods to solve relatively simple ballistic and other common dynamics problems. In particular, two dynamics problems with unknown traveling time are solved in this paper: a) optimizing a projectile trajectory to strike a target and b) optimizing the discrete sequence of forces acting throughout the displacement of a mass. The material presented in this paper is intended to be studied by students in elective biomechanics or control courses in the Mechanical Engineering Department at the University of Texas-Pan American.

1. Background

Dynamic system problems often require optimization of time, energy, forces, or other parameters. The mathematical model of a dynamic system can be represented in the form of an optimization problem, in which a cost function needs to be minimized subjected to equality, inequality, lower bound and/or upper bound constraints. Such optimal control problem can be solved using any of a variety of techniques available to perform parameter optimization of a modified cost function that incorporates information of the constraints. For example, analysis of human movement can be performed by converting an optimal control problem into a parameter optimization problem and using any adequate nonlinear programming algorithm an optimal solution can be computed[1]. Goh and Teo[2] and Powell[8] presented numerical solution approaches to optimal control problems with general constraints using parameter optimization techniques after including information of the constraints into the cost function. When the traveling time is part of the cost function to be minimized, the dynamic optimization problem is also called a ballistic problem.

Nonlinear programming methods are used to solve parameter optimization problems[2]. In this paper, the conjugate gradient (CG) and the Levenberg-Marquardt (LM) algorithms are used to solve two sample problems. LM is an algorithm for numerical optimization with relatively fast convergence[5]. The advantage of the LM technique is its results always move in the direction towards the minimization of the cost function. However, the most significant disadvantage of this technique is the computation of the modified pseudo-inverse of the Jacobian, $[J^TJ + \mu_k I]^{-1}$, for every batch iteration. Where J is the Jacobian and μ_k is a coefficient that is changed to adapt the algorithm to the progress accomplished after every iteration. The CG algorithm has a slower converging rate compared to the LM algorithm, and it requires the computation of the gradient of the cost function and minimization along a line in order to move the unknown parameters towards the optimal solution[5,7]. That is, a minimization subproblem needs to be solved to find the step size to be taken in the direction determined by the CG method.

As part of this project, future work will be performed to optimize the discrete sequence of forces acting throughout the displacement of a mass, using a more realistic muscle as the actuator that creates the motion of the mass. This would require to use a three-element Hill-type muscle model, with its contractile element in series with an elastic element, and both of them in parallel to another elastic element[3,4,6].

2. Optimization of projectile trajectory to strike target

This optimization problem consists of determining the launching angle of a projectile and the minimum traveling time to reach a target position subjected to a given initial velocity. As shown in Figure 1, for a target position with final "x" and "y" coordinates equal to 5 and 10 m, respectively, and an initial velocity v_0 of 15 m/s, there are two possible solutions for the launching angle: θ_1=73.07° and θ_2=80.36°, with the corresponding traveling times of 1.145 and 1.990 seconds. As shown in Figure 1, the thick line is the trajectory with the minimum traveling time for the projectile to hit the target.

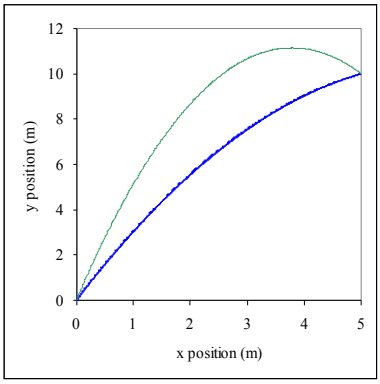

Figure 1. Projectile trajectories to strike target.

As studied in basic Mechanics and Physics courses, and neglecting friction, the equations for the motion of the projectile are:

$$x = v_0 \cos(\theta) t \tag{1}$$

$$y = v_0 \sin(\theta) t - \frac{1}{2} g t^2 \tag{2}$$

In the following section, this projectile problem is converted into an optimization problem and numerical optimization techniques are used to find its solution.

2.1 Converting the projectile problem into a constrained optimization problem

In order to determine the optimal projectile trajectory, the traveling time becomes the cost function to be minimized, with the projectile launching angle, θ, as the independent variable.

$$C(\theta) = t_f = \frac{x_f}{v_0 \cos(\theta)} \tag{3}$$

After substituting the time obtained from equation (1) into equation (2), an equality constraint is generated:

$$h(\theta)=y_f\cos^2(\theta)-x_f\sin(\theta)\cos(\theta)+\frac{g}{2}\left(\frac{x_f}{v_0}\right)^2=0 \tag{4}$$

Therefore, the constrained optimization problem consists of minimizing C(θ) in equation (3), subject to h(θ) = 0 in equation (4). The Lagrange multiplier theorem indicates that at a given local minimum θ^* of the cost function C(θ), there exits scalars λ_i (one per constraint), called Lagrange multipliers, such that $\nabla C(\theta^*) + \Sigma\lambda_i\nabla h_i(\theta^*) = 0$. The augmented Lagrange method[9,10] consists of the Lagrange multipliers method plus penalty functions used to create the augmented Lagrangian function A(θ); as presented in equation (5), A(θ) becomes the new cost function to be minimized:

$$A(\theta)=C(\theta)+\lambda h(\theta)+r_p h^2(\theta) \tag{5}$$

$$A(\theta)=\frac{x_f}{v_0\cos(\theta)}+\lambda h(\theta)+r_p h^2(\theta) \tag{6}$$

where λ is the Lagrange multiplier and r_p is a penalty factor associated with the equality constraint in equation (4). The Lagrange multiplier is updated after each iteration using equation (7):

$$\lambda_{k+1}=\lambda_k+2r_p h(\theta) \tag{7}$$

Equation (7) is based on optimization theory and though its validity is not proven here, the reader is encouraged to see references[9,10,11] for more details. In addition, the penalty factor, r_p, is chosen small for the first iteration and it is increased after each subsequent iteration[9]; but, it needs to be restricted to a maximum value, r_{p_max}. Equation (8) is recommended[9] to update r_p.

$$(r_p)_{k+1}=2(r_p)_k \tag{8}$$

2.2 The conjugate gradient optimization algorithm

The conjugate gradient (CG) method consists mainly of 4 steps[5,9,10], which are presented next as they are applied to the particular problem of the projectile in equations (3) and (4):

a) Determine the gradient of the cost function with respect to the parameters to be optimized; in the case of the projectile problem, the gradient is with respect to θ:

$$\nabla A_k=\frac{dA}{d\theta} \tag{9}$$

Start with $\lambda_0 = 0$; and $(r_p)_0 = 0.1$; and use $r_{p\_max} = 100$ and an initial guess for the values of the parameters to be optimized, $\theta_0 = \pi/4$ radians, for example. The gradient can be found with an approximation, using a small increment, for instance $\varepsilon = 1\text{x}10^{-8}$, in the value of the independent variables:

$$\nabla A_k=\frac{A(\theta+\epsilon)-A(\theta)}{\epsilon} \tag{10}$$

In the projectile example, the initial gradient of the cost function is:

$$\nabla A_0 = -5.6186 \tag{11}$$

b) Determine the search direction, make the search direction for the first iteration $p_0 = -\nabla A_0$, and find α_κ to minimize the cost function along such direction. In subsequent iterations, use p_k as indicated in step c). The new value of the parameters to be optimized are found with the following equation:

$$\theta_{k+1} = \theta_k + \alpha_k \, p_k \tag{12}$$

for the first iteration:

$$\theta_1 = \theta_0 + \alpha_0 \, p_0 \tag{13}$$

which for our projectile problem is:

$$\theta_1 = \frac{\pi}{4} + \alpha_0 (5.6186) \tag{14}$$

Determine α_k to minimize $A(\alpha)$:

$$A(\alpha) = A\left(\theta = \frac{\pi}{4} + 5.6186\,\alpha\right) \tag{15}$$

The strategy that was used to determine α_k is called golden section method[9], which consists on testing a sequence of values of α until the search is narrowed down to an interval in which the minimum of the cost function is expected to be[5,9]. Three points around the minimum value were used to approximate the cost function, $A(\alpha)$, with a quadratic function for which the minimum was found as an approximation of the minimum of $A(\alpha)$. Therefore, notice that to find the step size, α, another minimization problem has to be solved. For the projectile problem of equations (3), (4), and (14), during the first iteration, it was found that $\alpha_0 = 0.0474$ to minimize the cost function in equation (6). It can be observed from Figure 2 that the minimum of $A(\alpha)$ is in fact at $\alpha = 0.0474$.

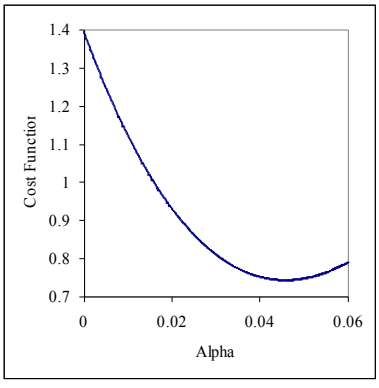

Figure 2. Cost function $A(\alpha)$ versus α.

c) Subsequent search directions: find a new search direction p_k, for k = 1,2,3,...

$$p_k = -\nabla A_k + \beta_k p_{k-1} \tag{16}$$

where, β is a value computed to make p_k and p_{k-1} conjugate directions. As presented in by Nocedal[10], the Polak-Ribiere value of β is:

$$\beta_k = \frac{\nabla A_k^T (\nabla A_k - \nabla A_{k-1})}{\nabla A_{k-1}^T \nabla A_{k-1}} \tag{17}$$

The conjugate gradient method was developed to solved linear system of equations, Qx = b, where Q, x, and b are matrices. This problem is stated as an optimization problem requiring the minimization of a new function $f(x) = \frac{1}{2}x^TQx - b^Tx$, whose solution can be obtained by a sequence of iterations to find x starting at an initial guess and advancing in the direction of vectors that are conjugate with respect to matrix Q[10].

d) Assess the convergence of the algorithm: if a satisfactory minimum of the cost function is not reached yet, repeat the procedure by going back to step b).

After 13 iterations, it was determined that the optimal value of the projectile traveling time is 1.145 seconds and the launching angle is 73.07°; which are the same results as the theoretical solution. The results of the 13 iterations are presented in table 1.

Notice that at the optimal point λ = 2.652, from table 1. Also, it was determined, after 13 iterations, that the gradient of the cost function and of the equality constraint at the optimal point are:

$$\nabla C(\theta) = 3.762 \tag{18}$$

$$\nabla h(\theta) = -1.418 \tag{19}$$

Therefore, these values of the gradients satisfy the Kuhn-Tucker condition at the optimal point:

$$\nabla C(\theta) = -\lambda \nabla h(\theta) \tag{20}$$

which is a requirement for constrained optimality[9]. For an explanation of the geometrical meaning of the Lagrange multipliers the reader should consult the references[9,10,11].

Table 1. Optimal control iteration results

Iteration	Angle (degrees)	λ	r_p	$A(\theta)$	$h(\theta)$
0	45.000	0.000	0.1	1.399	3.045
1	60.251	0.171	0.2	0.745	0.853
2	64.568	0.351	0.4	0.894	0.450
3	67.203	0.559	0.8	0.979	0.260
4	68.999	0.809	1.6	1.037	0.157
5	70.304	1.111	3.2	1.080	0.094
6	71.310	1.458	6.4	1.110	0.054
7	72.044	1.830	12.8	1.129	0.029
8	72.557	2.180	25.6	1.139	0.014
9	72.870	2.445	51.2	1.144	0.005
10	73.016	2.591	100.0	1.145	0.001
11	73.063	2.641	100.0	1.145	0.000
12	73.071	2.650	100.0	1.145	0.000
13	73.073	2.652	100.0	1.145	0.000

3. Optimization of force acting on mass-spring-damper (MSD) system

A optimal control problem that consists of optimizing the discrete sequence of forces acting on a mass-spring-damper system is solved in this section. First, the optimal control problem was converter into a parameter optimization problem, and second, non-linear programming methods were used to determine the optimal parameters while satisfying the constraints. Figure 3 shows the mass-spring-damper system and the external force acting on the mass.

The mass is assumed to be 1 kg and needs to be translated from position x = 0 to a desired position, x_{des}, which is assumed to be 0.1 meters in this example. The damper has a viscous friction of 10 N/(m/s) and the spring has a stiffness of 500 N/m. The challenge in this optimization problem is to use a number (21 in our case) of consecutive discrete force values to achieve this desired motion in the shortest possible time and also with minimum jerk. In addition, when the desired position is reached, the velocity and acceleration of the mass must be zero. It can be assumed that the force is created by a muscle and that the maximum muscle force value F_o is 200 Newtons and that there are not force-length-velocity effects; therefore, at any time, the muscle force can take any value between 0 and F_o. The mass-spring-damper system in Figure 3 has the following mathematical model, as a system of first-order differential equations:

$$\dot{x} = v$$

$$\dot{v} = \frac{(f - b\,v - k\,x)}{m} \tag{21}$$

where x is the position and v is the velocity of the mass; f is the external force that could be a force developed by the muscle shown in Figure 3.

The cost function, C, to be minimized is made up by the position, velocity, and acceleration errors, which must become zero at the final time. Therefore, the cost function is obtained based on the desired final state of the mass. When the mass reaches the desired position, its final velocity and acceleration must be zero in order for the mass to remain at that position. The minimization of the following equation will converge to the desired state of the mass in Fgire 3.

$$C=\left(x_{des}-x_f\right)^2+\left(v_{des}-v_f\right)^2+\left(f_f-k\,x_f-b\,v_f\right)^2 \tag{22}$$

where the subindex f means final value. Note that in this problem, the desired final velocity, v_{des}, is zero and the cost function is made up by the sum of the squared errors of final position, velocity, and acceleration.

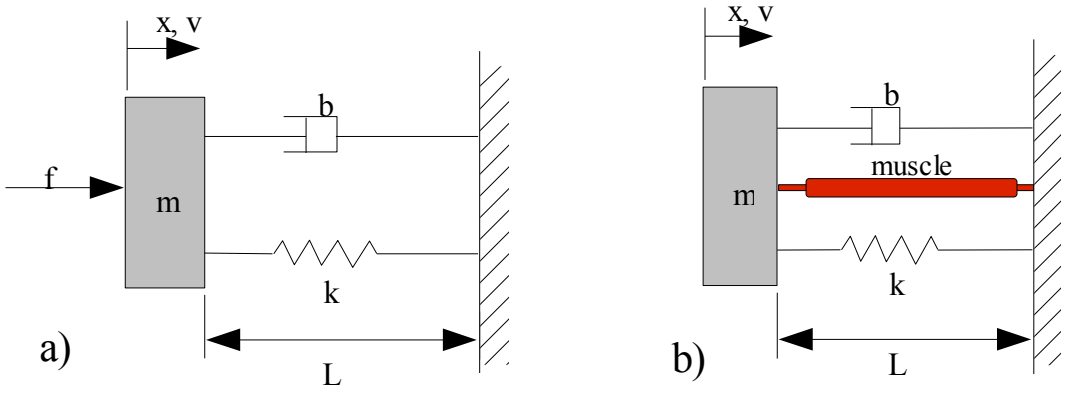

Figure 3. Mass-spring-damper system: a) an external force, f, and b) a muscle applying the external force.

Assume, for simplicity, that the force, f, can take a value between 0 and F_o at any time, and make the activation of the muscle, u(t), equal to 21 consecutive discrete parameters between 0 and 1 throughout the time of the motion:

$$u(t)=u_1, u_2, \ldots u_{21} \tag{23}$$

The muscle force is determine with the next equation,

$$F_M=u(t)F_o \tag{24}$$

Since the final time, t_f, is unknown and it has to be minimized to achieve the translation of the mass as quickly as possible. Therefore, the final time parameter, t_f, is added to the cost function and a new differential equation is also added to the system. Defining the variables $x_1 = x$, $x_2 = v$, and making x_3 equal to the normalized time $\tau = t/t_f$, a new system of first order differential equations is created, but the derivatives are with respect to τ.

$$\dot{x}_1 = t_f x_2$$
$$\dot{x}_2 = \frac{t_f}{m}\left(f - k\,x_1 - b\,x_2\right) \tag{25}$$
$$\dot{x}_3 = t_f$$

τ is the normalized time with respect to the final time, t_f, as explained by Pandy[1].

Note that the right hand side of each differential equation has been multiplied by the final time, t_f. Having normalized the time with respect to the final time, the differential equations must now be integrated with respect to τ from 0 to 1, and the t_f will be the final value of x_3. A convenient term is added to the cost function in equation (22) in order to minimize the traveling time to perform the motion as quickly as possible:

$$C = \left(x_{des} - x_{1f}\right)^2 + \left(v_{des} - x_{2f}\right)^2 + \left(f_f - k\,x_{1f} - b\,x_{2f}\right)^2 + \left(\int_0^1 t_f\, d\tau\right)^2 \tag{26}$$

The normalized final time, τ, becomes an additional term of the vector of unknown parameters to be optimized. When using 21 nodes for the discrete force,

$$u = \left[u_1, u_2, \ldots u_{21}, \tau\right] \tag{27}$$

In order to minimize jerk and generate an optimal solution for which the change in force between nodes is smooth, another term is included in the cost function[7], as shown in equation (28) .

$$C = \left(x_{des} - x_{fi}\right)^2 + \left(v_{des} - v_{fi}\right)^2 + \left(f_{fi} - k\,x_{fi} - b\,v_{fi}\right)^2 + \left(\int_0^1 t_f\, d\tau\right)^2 + \int_0^1 \left(\frac{\dot{f}}{F_o}\right)^2 d\tau \tag{28}$$

Consequently, another differential equation is added to the system:

$$\dot{x}_4 = t_f \left(\frac{u_{i+1} - u_i}{h}\right)^2 ; i = 1,2,\ldots N_{nodes-1} \tag{29}$$

where h is the normalized time step between nodes:

$$h = \frac{1}{N_{nodes} - 1} \tag{30}$$

The cost function is rewritten as a function of the state variables and final conditions:

$$C = \left(x_{des} - x_{1f}\right)^2 + \left(v_{des} - x_{2f}\right)^2 + \left(f_f - k\,x_{1f} - b\,x_{2f}\right)^2 + \left(x_{3f}\right)^2 + \left(x_{4f}\right)^2 \tag{31}$$

where x_{3f} and x_{4f} are final time and jerk, which are two new terms added to the cost function to be minimized.

In summary, the optimization problem consists of determining the vector of parameters u in equation (27) that minimizes the cost function C in equation (22), subjected to the system of first order differential equations (25) and (29).

3.1 Parameter Optimization Solution of MSD System

a) An initial guess of vector *u* is made and the discrete muscle force F_M is computed using equation (24).

b) The values of the state variables after time t_f are determined using forward integration of the differential equations, performed with the 4th order Runge-Kutta method.

c) Use an optimization algorithm to obtain the optimal vector u.

3.1.1 Levenberg-Marquardt Optimization Algorithm.

Newton's optimization method consists of using a second-order Taylor series to approximate the function to be minimized[12]:

$$f(z_{i+1}) = f(z_i) + f^1(z_i)(z_{i+1} - z_i) + \frac{1}{2}(z_{i+1} - z_i)^2 f^2(z_i) \tag{32}$$

where in a multidimensional problem *z* is a vector and f^1 becomes the gradient matrix and f^2 becomes the Hessian matrix. The minimization of the function is performed by iterations using the following equation:

$$z_{i+1} = z_i - \frac{f^1(z_i)}{f^2(z_i)} \tag{33}$$

The Levenberg-Marquardt optimization algorithm is a modification of the Newton's method and was designed to minimize functions that are sums of squares of other functions[5]. As explained by Hagan et al.[5], to find the optimal vector *u,* a column vector of errors, *E(x)*, is created as in the next equations:

$$E(x) = \left[E_1; E_2; E_3; E_4; E_5\right]$$

$$E(x) = \left[(x_{des} - x_{1f}); (v_{des} - x_{2f}); (f_f - k\,x_{1f} - b\,x_{2f}); x_{3f}; x_{4f}\right] \tag{34}$$

and the value of the cost function in equation (34) is computed as follows:

$$C = E^T(x)\,E(x) \tag{35}$$

Next, each parameter in the *u* vector is individually increased a small amount ε (for example ε = 1e-6), and each time forward integration of the differential equations is performed, and the new value of the vector E(x) is computed to numerically generate the Jacobian matrix, J:

$$J = \begin{bmatrix} \frac{\partial E_1}{\partial u_1} & \frac{\partial E_1}{\partial u_2} & \cdots & \frac{\partial E_1}{\partial u_{N_{nodes}}} & \frac{\partial E_1}{\partial \tau} \\ \frac{\partial E_2}{\partial u_1} & \frac{\partial E_2}{\partial u_2} & \cdots & \frac{\partial E_2}{\partial u_{N_{nodes}}} & \frac{\partial E_2}{\partial \tau} \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ \frac{\partial E_5}{\partial u_1} & \frac{\partial E_5}{\partial u_2} & \cdots & \frac{\partial E_5}{\partial u_{N_{nodes}}} & \frac{\partial E_5}{\partial \tau} \end{bmatrix} \tag{36}$$

Notice that the partial derivatives in (37) are approximated by using incremental amounts of the terms in vector u, similar to the way done in equation (10). Compute the new vector u_{new}, using a small scalar μ of about 0.01 as recommended by Hagan et al.[5]:

$$u_{new} = u_{old} - \left[J^T J + \mu I\right]^{-1} J^T E(x) \tag{37}$$

Use u_{new} to find the value of the new cost function. Accept u_{new}, but only if the new value of the cost function is less than its previous value. Otherwise, multiple μ by 10 (or other constant > 1) and try another u_{new} until the new value of the cost function is less than the previous one. Use the new vector u_{new} and repeat the entire process until a satisfactory minimum cost function is found.

In equation 37, the gradient is given by $2J^TE(x)$ and the Hessian is approximated by $2[J^TJ+\mu I]$, as explained by Hagan et al.[5]. It is important to mentions that the LM algorithm behaves like the steepest descent method when μ is large, and like the Newton's method when μ approaches zero.

After 100 iterations of the LM algorithm, it was obtained that the time required to move the mass a displacement of 0.1 meters is t_f= 0.114 seconds. The top plot of Figure 4 shows 21 values of the muscle activation parameter, u(t), which multiplied by the optimal muscle force, F_o, will result in the actual force acting on the mass, m. The resultant motion of the mass is indicated in the two lower plots of Figure 4. The last value of u(t) in the top plot of Figure 4 is the optimal final time, t_f.

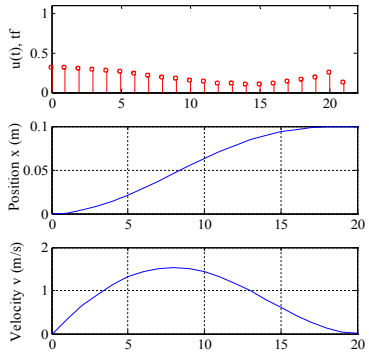

Figure 4. Force optimization for displacement of the mass m to x_{des}=0.1 meters.

There are some optimization techniques that start with an initial approximation of the inverse of the Hessian.

3.2 Adding a Velocity Inequality Constraint to the MSD System Problem

In general, optimization problems include inequality, equality, upper bound, and/or lower bound constraints of the inputs and/or state variables. For example, it might be required that the velocity of the mass in the previous MSD system be always less or equal to 1 m/s.

$$x_2 \leq 1 \tag{38}$$

which is the same as

$$x_2 - 1 \le 0 \tag{39}$$

One convenient way to take this constraint into account is by including it in the cost function, as indicated by Powell[8] and by Pandy et al.[1], and adding a new differential equation to the system:

$$\dot{x_5} = t_f \; min^2\left(-\left(x_2 - 1\right), 0\right) \tag{40}$$

Notice that when equation (40) is integrated from 0 to 1, the final value of x_5 must be zero in order to guarantee that the constraint was always satisfied. Therefore, adding $(x_{5f})^2$ to the cost function will put effort to satisfy this constraint by forcing it to be zero. The new cost function is:

$$C = \left(x_{des} - x_{1f}\right)^2 + \left(v_{des} - x_{2f}\right)^2 + \left(f_f - k\,x_{1f} - b\,x_{2f}\right)^2 + \left(x_{3f}\right)^2 + \left(x_{4f}\right)^2 + \left(x_{5f}\right)^2 \tag{41}$$

The new vector of errors is:

$$E(x) = \left[E_1; E_2; E_3; E_4; E_5; E_6\right]$$

$$E(x) = \left[\left(x_{des} - x_{1f}\right); \left(v_{des} - x_{2f}\right); \left(f_f - k\,x_{1f} - b\,x_{2f}\right); x_{3f}; x_{4f}; x_{5f}\right] \tag{42}$$

The results obtained after solving this constrained parameter optimization problem are presented in Figure 5. The final time was $t_f = 0.149$ seconds. Note that the desired final position, velocity and acceleration were achieved, and the maximum velocity of the mass was always less of equal to 1 m/s. Comparing Figure 4 and Figure 5 shows the different response obtained by including the constraint of the maximum acceptable velocity of the mass.

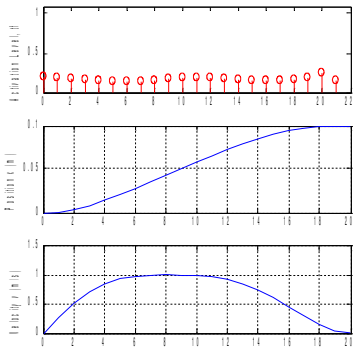

Figure 5. Force optimization using the LM method for displacement of the mass, m, with x_{des}= 0.1 meters and velocity constraint v ≤ 1 m/s.

3.3 Conjugate Gradient (CG) Algorithm using the Polak-Ribiere Approach

The CG optimization algorithm was presented in section 2.2, and the only variation required for the MSD problem, is that optimization of multiple variables is needed, instead of just one variable like in section 2.2. The gradient of the cost function with respect to the parameters to be optimized is:

$$\nabla C = \left[\frac{\partial J_c}{\partial u_1} \quad \frac{\partial J_c}{\partial u_2} \quad \cdots \quad \frac{\partial J_c}{\partial u_{N_{nodes}}} \quad \frac{\partial J_c}{\partial \tau} \right]^T \tag{43}$$

and the rest of the procedure remains the same as in section 2.2

For the same problem of the MSD system solved in section 3.2, but now using the CG optimization method, it was determined that after 1000 iterations, the final time, t_f, is 0.140 seconds, and the response of the system as well as the optimal discrete force acting on the mass is presented in Figure 6.

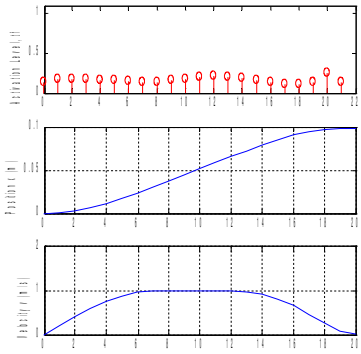

Figure 6. Force optimization using the CG method for displacement of the mass, m, with x_{des}= 0.1 meters and velocity constraint $v \leq 1$ m/s.

Even though the results are similar, it was determined that the LM method converges faster than the CG method. Notice that the system responses in Figure 5 and Figure 6 are similar. There is a small difference in the final time, but the MSD system reached the same objective position and also satisfied the maximum velocity constraint. The difference obtained with the two optimization methods in the response of the mass and the final time, t_f, is probably due to the number of iterations allowed for each algorithm.

4. Conclusion

This paper presented applications of constrained optimization techniques in common dynamic systems studied to introduce undergraduate Mechanical Engineering students to optimization methods to solve relatively simple ballistic and other common dynamic problems. In the first problem, optimization techniques were used to determine the minimum traveling time and projectile launching angle required to strike a target, given the initial velocity of the projectile. This problem was solved by implementing the augmented Lagrange method in Matlab and the results were validated by comparison with the actual solution of the problem. For the second problem, constrained optimization techniques were used to determine the optimal discrete sequence of forces that a muscle-like actuator needs to apply in order for a mass to move from one position to another in minimum time while satisfying equality and/or inequality constraints. A set of first order differential equations representing the model of a mass-spring-damper system

was numerically integrated to determine the final values of position, velocity, and acceleration after applying a first estimate of the unknown discrete sequence of forces during a first guess of the unknown traveling time. The implemented optimal control algorithm not only determined the minimum traveling time of the mass but also the discrete force sequence required to satisfy the constraints and to perform the desired motion of the mass.

5. References

[1] Pandy, M., Anderson, F., and Hull, D. 1992. Parameter optimization approach for the optimal control of large-scale musculoskeletal systems. Journal of Biomechanical Engineering, Transactions of the ASME, v 114, n 4, Nov., 1992, p 450-460

[2] Goh, C. and Teo, K. 1988. Control parametrization: a unified approach to optimal control problems with general constraints. Automatica, v 24, n 1, Jan, p 3-18

[3] Sim, E.; Levine, W.; Zajac, F. 1989. Some results on modeling and computing the controls that produce maximal height jumps. IEEE International Conference on Control and Applications. p 881 – 888.

[4] Garner, B. and Pandy, M. 2003. Estimation of Musculotendon Properties in the Human Upper Limb. Annals of Biomedical Engineering, Vol. 31, pp. 207–220

[5] Hagan, M., Demuth, H., and Beale, M. 1995. Neural Network Design. Boston: PWS Publishing.

[6] Pandy, M.. Zajac, F., Sim E., and Levine, W. 1990. An Optimal Control Model for Maximum-Height Human Jumping. J. Biomech. Vol. 23, pp. 1185-1198

[7] Pandy, M., Garner, B., and Anderson, F. 1995. Optimal Control of Non-ballistic Muscular Movements: A Constraint-Based Performance Criterion for Rising from a Chair. Journal of Biomechanical Engineering 117(1) pp. 15-26.

[8] Powell, M. 1978. A fast algorithm for nonlinearly constrained optimization calculations. Springer Verlag, Vol 630, pp. 144:157.

[9] Vanderplaats,G. 1984. Numerical optimization techniques for engineering design : with applications. McGraw-Hill.

[10] Nocedal, J. and Wright, S. 2000. Numerical Optimization. Springer.

[11] Arora, J.. 1989. Introduction to Optimum Design. McGraw-Hill.

[12] Chapra, S. and Canale, R. 2006. Numerical Methods for Engineers. McGraw-Hill.