Improved Joint Control using a Genetic Algorithm for a Humanoid Robot

Jonathan Roberts[1], **Damien Kee**[2] **& Gordon Wyeth**[2]
[1]CSIRO Manufacturing Science & Technology
PO Box 883, Kenmore, Qld 4069, Australia
[2]School of Information Technology & Electrical Engineering
University of Queensland, St. Lucia, Qld 4072, Australia

Abstract

This paper describes experiments conducted in order to simultaneously tune 15 joints of a humanoid robot. Two Genetic Algorithm (GA) based tuning methods were developed and compared against a hand-tuned solution. The system was tuned in order to minimise tracking error while at the same time achieve smooth joint motion. Joint smoothness is crucial for the accurate calculation of on-line ZMP estimation, a prerequisite for a closed-loop dynamically stable humanoid walking gait. Results in both simulation and on a real robot are presented, demonstrating the superior smoothness performance of the GA based methods.

1 Introduction

The University of Queensland GuRoo humanoid project was started in 2001 with the aim of developing a medium sized humanoid robot to be used for research. The current robot can perform a number of basic tasks, such as stand on one leg, crouch, bow, and can also walk. Two walking gaits have been developed so far with GuRoo now able to walk unassisted.

1.1 Current gait (open-loop, off-line ZMP)

GuRoo currently walks using a so-called Zero Point Moment (ZMP) stable gait. The ZMP is the point on the ground plane where the total forces and moments acting on the robot are zero. A biped robot is dynamically stable if the ZMP lies within the support polygon [Vukobratovic and Juricic, 1969].

The method for generating the ZMP stable gait revolves around the use of parameterised loci of foot movement, where the parameters are determined using a Genetic Algorithm [Wyeth *et al.*, 2003]. This process was performed off-line using a dynamic simulation of GuRoo. The gait generated is run on the real GuRoo robot in an open-loop fashion, with the gait generating the desired joint velocities. The ZMP is therefore not currently calculated on GuRoo.

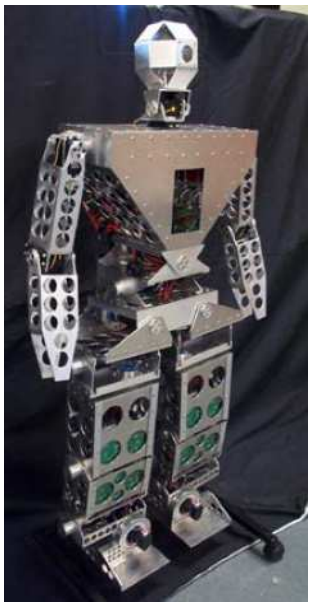

Figure 1: GuRoo.

1.2 Future gaits (closed-loop, on-line ZMP)

It is of course more desirable to have a closed-loop gait, where the ZMP is calculated on-line and therefore the robot is capable of reacting to any external force, or instabilities (the open-loop gait is not perfect). The calculation of the ZMP can be simplified by assuming a point-mass model where the calculation is performed at the centre-of-mass of each link. This requires a knowledge of the forward kinematics to calculate the position of each link, the joint positions (currently available using encoders), the pose of the robot (roll and pitch angles) and the accelerations at each link.

The pose (roll and pitch angles) and the acceleration of a single link may be determined using an Inertial Measurement Unit (IMU) which is strapped down to the link in question. Forward kinematics can then be used to derive the pose and accelerations of the remaining links.

GuRoo has recently been fitted with a low-cost IMU (the EiMU from CSIRO). This device uses a 3-axis rate gyro and

3-axis accelerometer to calculate roll and pitch angle information, along with the acceleration information needed for on-line ZMP calculation. The unit is mounted in the head of GuRoo.

1.3 "The shakes"

A significant problem with on-line ZMP calculation is measurement of link accelerations. Until now, the PI joint controllers have been hand-tuned to minimise tracking error. However, this has been at the expense of joint error smoothness. The hand tuning of these controllers has resulted in a system that is good enough for GuRoo to walk successfully, but has resulted in a robot with a high-frequency shake. This shake is too small to see by looking at GuRoo, but can be easily heard as a rattling noise. The result of this shake is extremely high oscillatory accelerations at each link, making the resultant ZMP values unusable. One solution is to heavily filter the acceleration values prior to the ZMP calculation. However, a more elegant solution is to eliminate the shakes.

High-frequency shaking is also bad for GuRoo mechanical health. A smoother robot will last longer.

1.4 Paper overview

The remainder of this paper deals with the problem of achieving smooth joint motion in order to guarantee smooth link accelerations, and hence a usable (low noise) on-line ZMP estimate. The following section briefly summarizes the GuRoo development platform, including the hardware, control architecture and the simulator. Section 3 then describes the problem of tuning the joint controllers in order to achieve a smooth motion and outlines a potential solution to this problem using a Genetic Algorithm. Section 4 describes the implementation of the GA using the simulation environment, and shows the results. Sections 5 and 6 show some experiments on the simulated and real robots respectively using the gains generated by the GA. Finally, Section 7 lists some conclusions.

2 The GuRoo Development Platform

The GuRoo is an anthropomorphic robot with 23 degrees of freedom. GuRoo is 1.2m tall, weighs 38kg with batteries and is constructed out of aluminium. This section briefly summaries the GuRoo development platform. A detailed description of the GuRoo robot and simulator may be found in [Wyeth *et al.*, 2001; Kee *et al.*, 2003].

2.1 Joints and actuators

Figure 2 shows a CAD model of GuRoo indicating the location of the 23 joints. The unnaturally wide legs are the result of the length of the actuators used for the ankles, knees and hips. The multiple degrees of freedom found in the human hips and shoulders (ball joints) are implemented in GuRoo using small sequential links. Table 1 outlines the type and number of all joints. Brushed DC motors are used to power the joints of the lower limbs and spine (capable of maximum continuous output torque of 10Nm with a maximum speed of 5.2rad/s at 2A current consumption). All these joints use the same motor and gear-head combination (these motors comprise 33% of the weight of the robot). The joints of the upper body (arms, shoulders, neck and head) are actuated by large RC servo motors (capable of 1.4Nm output torque at speeds of 5.2rad/s).

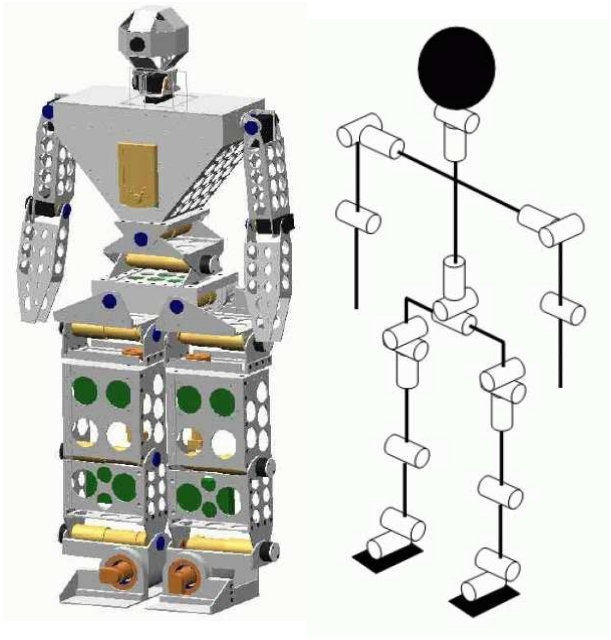

Figure 2: GuRoo: CAD model (left), location of the 23 joints (right).

Joint	Type	Axis	No.
Head/Neck	RC Servo	Pitch + Yaw	2
Shoulder	RC Servo	Pitch + Roll	2x2
Elbow	RC Servo	Pitch	2x1
Spine	DC Brushed	Pitch + Roll + Yaw	3
Hip	DC Brushed	Pitch + Roll + Yaw	2x3
Knee	DC Brushed	Pitch	2x1
Ankle	DC Brushed	Pitch + Roll	2x2

Table 1: Summary of the 23 joints ("2x" indicates left and right side).

In the unpowered state and when lifted from the ground, the legs will naturally swing together (due to the centre of mass of the leg being located outside the line of the hip joint). To overcome this problem, two torsion springs are located in parallel with the hip roll actuators (1Nm/degree). These springs are set such that when unpowered, the legs of the robot hang straight down.

2.2 Control architecture

The control architecture consists of six distributed joint controller boards and either a Compaq iPaq PDA or standard PC as the central controller (Figure 3). The six joint controller boards are based around the TMS320F243 DSP from Texas Instruments. These boards communicate via the CAN Bus.

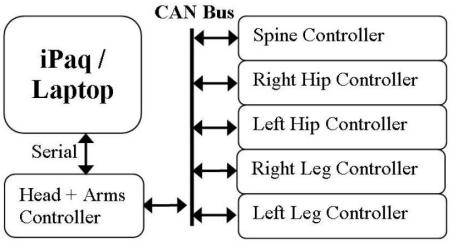

Figure 3: The distributed control architecture.

Control of the 15 DC brushed motors is split evenly between 5 joint controller boards (known as motor controller boards), each being responsible for 3 motors. All power electronics has been custom designed and implemented on the joint controller boards. Each board reads the current position of its joints and implements local PI controllers.

The sixth joint controller board controls the position of the upper body RC servos, and is known as the servo controller board. The central controller calculates the desired velocities for each joint and sends them to the servo controller board via an RS232 serial link. These commands are then forwarded onto the motor controller boards via the CAN Bus. Finally, the EiMU module also sends its data out on the CAN Bus.

2.3 The simulator

The GuRoo simulator is an important part of the overall GuRoo development platform. Its development has been the key to the overall success so far of the GuRoo project.

The simulator is based on the DynaMechs dynamic simulation package [McMillan, 1995] which uses the OpenGL 3D graphics libraries. The basic DynaMechs package has been adapted to include the specific characteristics of GuRoo. More specifically, the distributed control architecture is fully simulated including simulations of the joint controller boards and CAN Bus communications (including delays). The modified Denavit-Hartenburg parameters and CAD surface area provide the graphical representation of the GuRoo simulation (Figure 4).

The choice of DynaMechs and hence OpenGL means that the simulation runs on multiple platforms such as MS-Windows, Linux, and Solaris.

3 Controller Tuning

The remainder of this paper will deal with the lower 15 joints only, ie. the high-power DC brushed motor joints (spine, hips, knees and ankles).

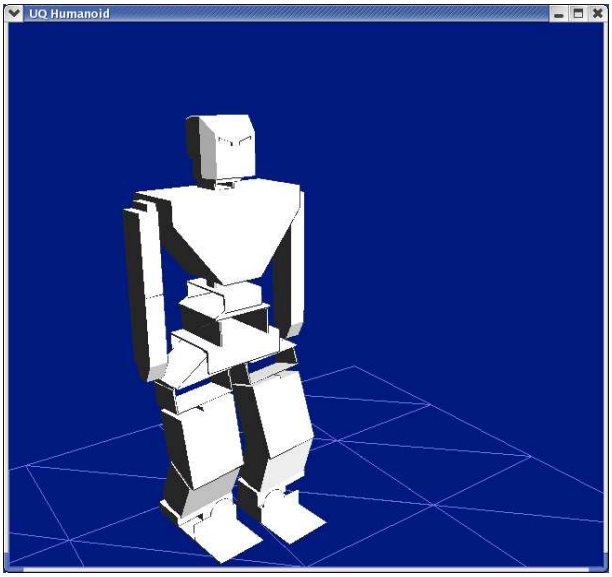

Figure 4: Screen shot of the graphical part of the GuRoo dynamic simulation.

3.1 The problem

The aim of the GuRoo project is to create a humanoid robot that can perform human-like motions and hence tasks, such as walking, crouching, reaching, etc. In order for these motions to be realised, it is essential that the lowest-level controllers, the joint controllers, perform well.

Each joint controller is implemented as a simple PI controller around joint velocity. Each joint must therefore be tuned, i.e. optimal values of the P and I gains must be found for each of the 15 joints.

It should be realised that all joints must remain active during all phases of a motion. Even the act of standing up straight and remaining still requires all joints to be active. GuRoo simply slumps to the floor if it loses power! This implies that all 15 joint controllers must be tuned simultaneously in order to achieve the best outcome. An oscillation in one joint (because its controller is poorly tuned) will effect the performance of every other controller due to subsequent vibration and motion (generated at that joint) being transmitted through the robot to all the other joints.

3.2 Gain scheduling

All leg joints (ankles, knee and hips) currently use a single set of gains for both the supported phase (where the leg is in contact with the ground) and the unsupported phase (where the leg is in the air and not in contact with the ground). The loads experienced at the joints during these two phases are quite different, and so gain scheduling seems an obvious and natural way of dealing with these differences. That is, we will use one set of gains for the supported phase and another set for the unsupported phase.

3.3 Reducing the problem

We are interested in tuning 15 joints simultaneously, each with two gains (a P and an I gain), making 30 gains in total. However, there is symmetry in GuRoo, and the legs can be considered to be identical. Hence, the problem maybe reduced to finding 18 gains (6 for the spine and 12 for a leg). If we are to use gain scheduling on the legs then we must find 30 gains (6 for the spine and 24 for a leg).

In order to simplify the problem, we can restrict the possible values of the gains to a range that we know is roughly correct. It is unrealistic to tune all 15 controllers simultaneously without restricting the values of the gains. The robot must actually walk most of the time for us to perform the tuning experiments.

The PI controllers are implemented on the motor controller boards. These boards have limited processing power and hence the controllers have been implemented using a bit shifting strategy for speed. Hence, the P and I gains are not floats, but integers with unit values of 1, 2, 4 and 8 which corresponds to bit shifting values of 0, 1, 2 or 3. This allows gains to be represented as a 2-bit number. When the output of the PI control law is computed, the proportional error and integral error are shifted the number of bits corresponding to the P and I gains respectively. We know from experiments that gains set in this range will allow GuRoo to walk (at least very roughly).

3.4 Genetic algorithms for controller tuning

There are numerous traditional methods available to the control engineer to tune PI controllers. A vast amount of material has been published in the area. A summary of the most popular techniques is given in [O'Dwyer, 2003]. It should be noted however, that most of these deal with the tuning of a single controller in isolation.

Since the early 1990's, Genetic Algorithms (GAs) have been successfully used to tune PI and PID controllers [Porter and Jones, 1992; Wang and Kwok, 1994; Herrero *et al.*, 2002]. As with more traditional approaches to tuning, most of these GA methods have only been applied to single controllers. However, [Bomfin *et al.*, 2000] use GAs to simultaneously tune multiple power system damping controllers.

The remainder of this paper demonstrates the use of Genetic Algorithms to simultaneously tune the 15 high power joints on GuRoo. Both gain scheduled and non-gain scheduled approaches are investigated with both being compared with the current hand-tuned solution.

4 Tuning

The current focus of GuRoo research is to develop a highly stable walking gait. Therefore the GA was run on the most stable walking gait so far developed for GuRoo [Wyeth *et al.*, 2003]. This gait was generated based on a method using the parameterised loci of foot movement, where the parameters of motion were themselves determined suing a GA.

The GuRoo simulator provides the perfect environment on which to run a GA. Running the GA on the real robot would be difficult due to the constant need to move the robot back to the starting position every run and the problem that a poor run, where the gains are sub-optimal, may result in a highly unstable robot, which may fall over.

4.1 Fitness function — tracking and smoothness

As with all applications of GAs the key to success is finding the best fitness function that describes the performance of the system. In the case of GuRoo's joint controllers we are interested in optimising two measures, tracking performance (minimising error) and tracking smoothness. The biggest single problem with the hand-tuned controllers was the noisy tracking performance which would give the robot a high-frequency shake.

The overall fitness function, f, used was therefore based on two separate fitness functions based on tracking error (f_t) and smoothness (f_s):

$$f = \frac{5.0}{f_t} + \frac{3.0}{f_s}, \tag{1}$$

$$f_t = \sum_{j=1}^{N} \sum_{t=1}^{T} |p^j_{err}(t)| \frac{\Delta t}{T} \tag{2}$$

$$f_s = \sum_{j=1}^{N} \sum_{t=1}^{T} |\ddot{p}^j_{err}(t)| \frac{\Delta t}{T} \tag{3}$$

where $p^j_{err}(t)$ is the joint tracking error of joint j at time t.

4.2 The GA and its parameters

The type of GA used for tuning was a so-called 'simple GA' described in [Goldberg, 1989]. The GAlib C++ library was used to run the GAs [Wall, 2000]. A simple GA uses non-overlapping populations and optional elitism, and creates an entirely new population of individuals each generation. The GA parameters used were as follows:

- Population size: 100
- Number of populations: 100
- Crossover: 60%
- Mutation: 10%
- Genome: a 36-bit long bit string of concatenated 18 2-bit gains for the non-gain scheduled method and a 60-bit long string of concatenated 30 2-bit gains for the gain scheduled method.

The GA was run over a complete walking cycle which consists of eight elements:

- right foot strikes ground
- right foot on ground
- left foot toe off
- left knee straighten

- left foot strikes ground
- left foot on ground
- right foot toe off
- right knee straighten

However, the simulation begins with the robot in the standing position. This pose is not part of the walking gait and so the robot must first move from the standing position into a walking pose. This takes place using a extra two elements. The first cycle of the walk is therefore 'non-standard' and hence the fitness function is only computed over the second walk cycle. Each walk cycle takes 6 seconds to run on a 2GHz PC (real-time) and so a complete run of a single GA individual takes 12 seconds. The total time to run the GA as described above was 1000 minutes (16.6 hours).

Note that an exhaustive search for the optimal gains would take a little over 68,000 years to complete running on the same PC.

4.3 GA results

Figure 5 shows how the two GA (gain scheduled and non-gain scheduled) runs converged. This figure clearly shows that the non-gain scheduled GA performed slightly better than the gain scheduled GA, and that the non-gain scheduled GA converged sooner, at 18 generations, compared with 59 generations for the gain-scheduled GA. It was expected that convergence would be quicker for the non-gain scheduled GA since the genome is shorter (36-bits compared to 60-bits). However, it is interesting that the non-gain scheduled GA actually performs better than the gain-scheduled GA. This is probably again due to the smaller genome.

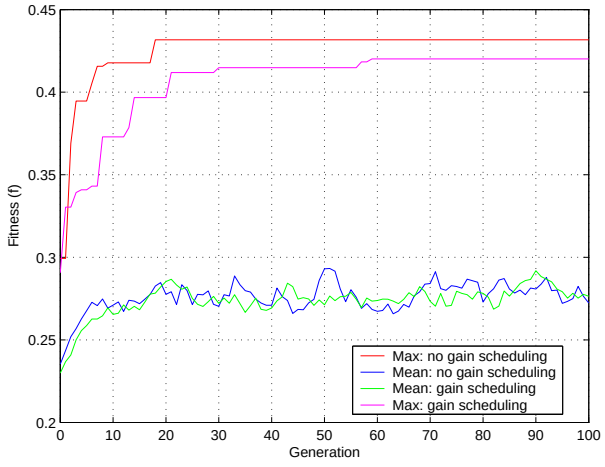

Figure 5: Convergence of the GA.

An interesting feature of both GAs is the fact that the mean of the generations did not converge to the max (Figure 5). It improved initially, but then showed no real signs of improvement. This is probably due to the fact that the GAs are already starting from a very roughly tuned point (due to the constraints applied to the gain values), and not a truly random point.

5 Simulation Experiments

The gains generated by both GA tuning methods (gain scheduled and non-gain scheduled) and the original hand-tuned gains were run on the simulated GuRoo over a number of walking cycles.

The results for all three cases are compared in Table 2. This table gives the break-down of the components of tracking and smoothness performance. From this table it can be seen that all three tuning methods track with approximately the same accuracy, with the non-gain scheduled GA method performing best and the other two equally well. However, this is not the case for the smoothness measure which shows that both GA tuning methods are far superior to the hand tuning method. Note that the non-gain scheduled GA performed best with respect to smoothness. On both tracking and smoothness the non-gain scheduled GA performed best.

Tuning Method	Tracking ($\frac{5.0}{f_t}$)	Smoothness ($\frac{3.0}{f_s}$)
Hand-tuned	0.116	0.152
GA no gain-scheduling	0.122	0.310
GA gain scheduling	0.115	0.305

Table 2: Simulation results (showing fitness measures) for three tuning methods tried (high number is better).

Figure 6 shows a comparison of the joint error tracking performance for two joints during a section of the simulated walk. The top figure shows the right knee joint and the bottom, the right ankle-pitch joint. The section of the walk covered by this figure is the start, which encompasses the motion that takes the robot from the standing position into the walk. The hand-tuned run exhibits a large error starting at the 1 second mark for the knee, and the 2 second mark for the ankle. These large errors are not present in both GA runs showing a superior tracking performance in certain situations.

Figure 7 shows a comparison of joint error smoothness for the same two joints (the right knee and right ankle-pitch). It is clear that both GA methods have produced far smoother results when compared to the hand-tuned method.

Finally, Figure 8 shows a comparison of the ZMP during four complete walking cycles for each of the three tuning methods. It can be seen that the hand-tuned method has produced a very noisy ZMP estimate with both GA methods producing much cleaner estimates. It should also be noted that during the hand-tuned run, the robot begins to walk off in the positive y-direction (this is clear at the 1m mark x-position). This is probably due to the high-vibration shake allowing the robot to twist slightly on the floor.

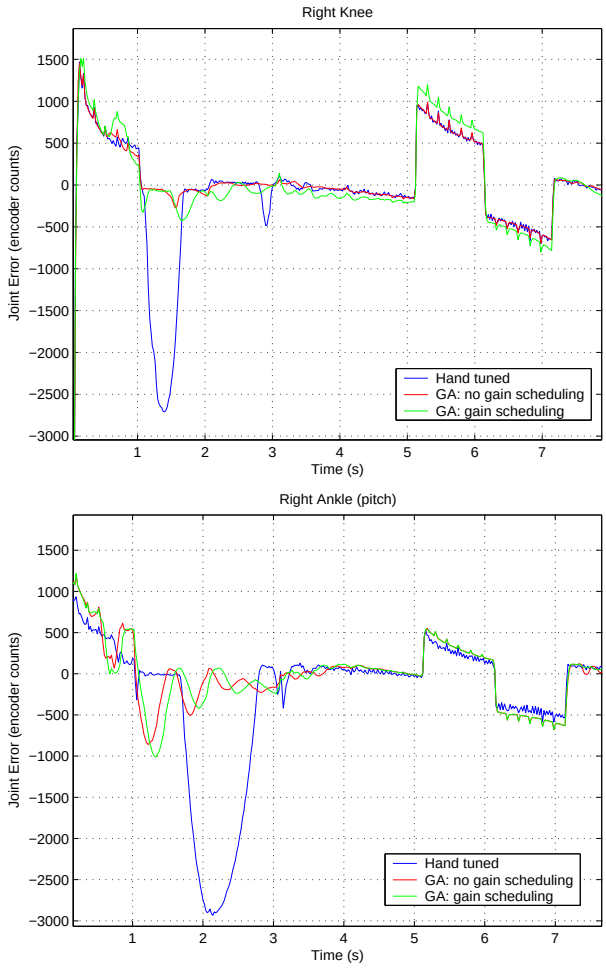

Figure 6: Comparison of joint error during simulated walking for the right knee (top) and right ankle (bottom).

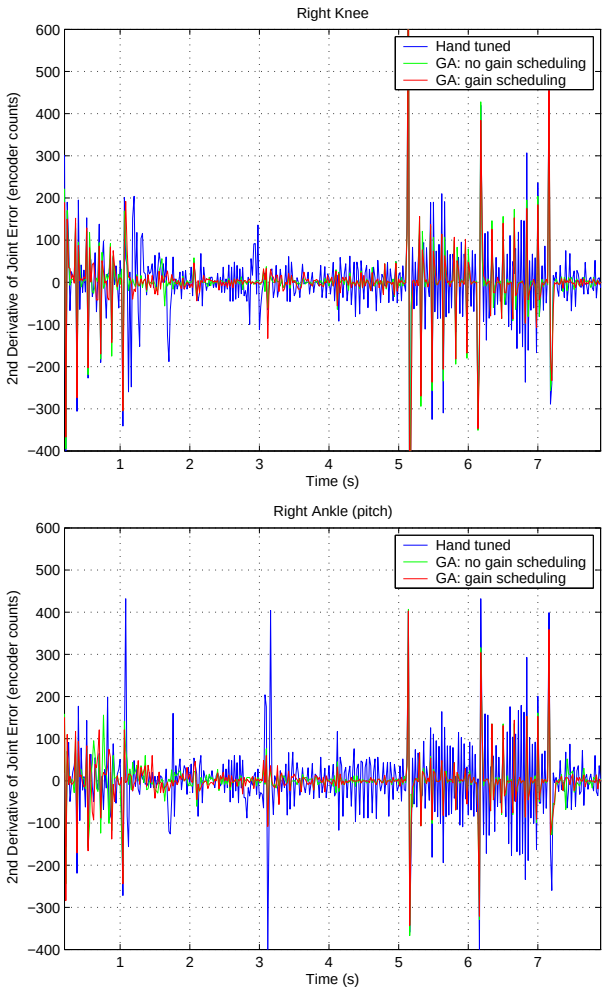

Figure 7: Comparison of joint error smoothness during simulated walking for the right knee (top) and right ankle (bottom).

6 Real Robot Experiments

The gains generated by the GA tuning method (non-gain scheduled) and the original hand-tuned gains were run on the real GuRoo robot over a number of walking cycles. We have yet to implement GA with gain scheduling on the real robot.

A comparison of the results is given in Table 3. This table gives the break-down of the components of tracking and smoothness performance. From this table it can be seen that the GA tuning method with no gain scheduling performs best with respect to tracking and smoothness.

It is interesting to note that the results differ significantly from the simulation case in that during the real experiments, the tracking performance was much worse, but the smoothness performance was slightly better. This can probably be explained by the fact that the real robot is much more compliant than the simulation model and hence 'wobbles' more when walking. This compliance has a natural smoothing tendency (hence smoother operation) but makes joint control more challenging (hence larger tracking errors).

Figure 9 shows a comparison of the joint error tracking performance for two joints during a section of the real walk. The top figure shows the right knee joint and the bottom, the right ankle-pitch joint. This is the same section of the walk used for the simulation results above. The figure for the right ankle shows a remarkable difference between the tuning methods. The comparative gains for this joint were $P = 1$, $I = 4$ for the hand-tuned method and $P = 2$, $I = 1$ for the GA non-gain scheduled method. This difference in gains produces a very different result.

Figure 10 shows a comparison of joint error smoothness for the same two joints (the right knee and right ankle-pitch). As with in simulation, the GA method produced smoother results when compared to the hand-tuned method (although the improvement is not as obvious as in the simulation).

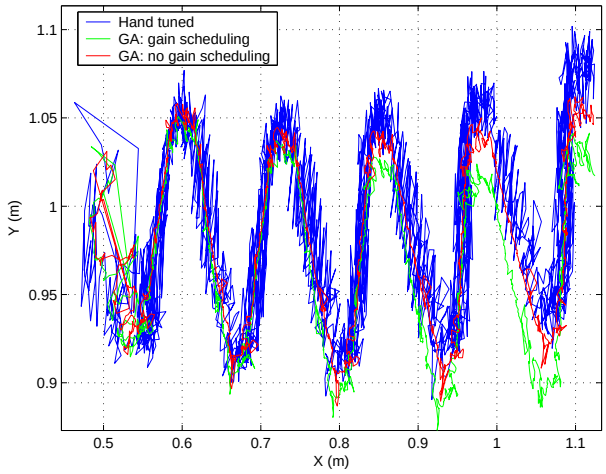

Figure 8: Comparison of ZMP for the three differently tuned simulations for four complete walking cycles.

Tuning Method	Tracking ($\frac{5.0}{f_t}$)	Smoothness ($\frac{3.0}{f_s}$)
Hand-tuned	0.009	0.539
GA no gain-scheduling	0.011	0.699

Table 3: Real robot results (showing fitness measures) for two tuning methods tried (high number is better).

7 Conclusions

This paper has described experiments conducted in order to simultaneously tune 15 joints of a walking humanoid robot. Two Genetic Algorithm (GA) based tuning methods were developed and compared against a hand-tuned solution. The first GA method used a single set of controller gains for all phases of the walking gait. The second method used gain scheduling and contained two sets of gains for each of the leg joint controller (one set for the supported phase and one set for the unsupported phase).

The system was tuned in order to minimise tracking error while at the same time achieve smooth joint motion. A fitness function was developed that incorporated both tracking error and smoothness performance. The GAs were run off-line using a humanoid simulation environment.

Simulation experiments were then conducted to compare the performance of the two GA tuning methods and the original hand-tuned method. The results of these experiments showed that both GA methods performed as well as the hand-tuned method (with one performing slightly better) while at the same achieving a significantly better smoothness performance.

Experiments on the real robot again showed that the GA method was superior, but the results differed significantly from the simulation case in that during the real experiments, the tracking performance of all methods was much worse, but

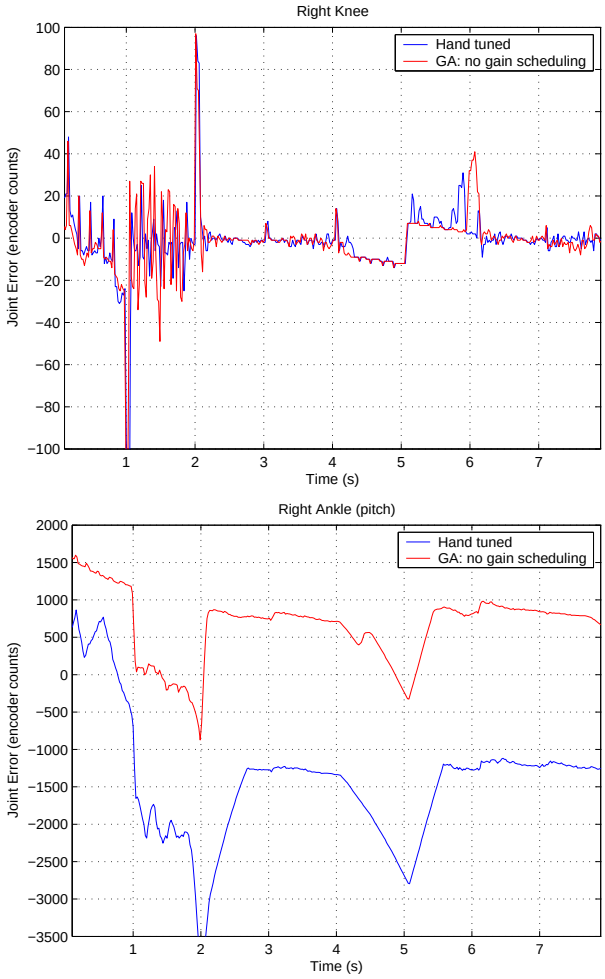

Figure 9: Comparison of joint error during real walking for the right knee (top) and right ankle (bottom).

the smoothness performance was slightly better.

Acknowledgments

The work described in this paper was carried out during the first author's secondment to the UQ robotics group. He would like to thank both CSIRO Manufacturing Science & Infrastructure and the University of Queensland for the opportunity to work within the UQ robotics group. The authors would also like to thank the many undergraduate students who have participated in the GuRoo project over the past three years.

References

[Bomfin *et al.*, 2000] A. L. B. D. Bomfin, G. N. Taranto, and D. M. Falcao. Simultaneous tuning of power system damping controllers using genetic algorithms. *IEEE Transactions on Power Systems*, pages 163–169, 2000.

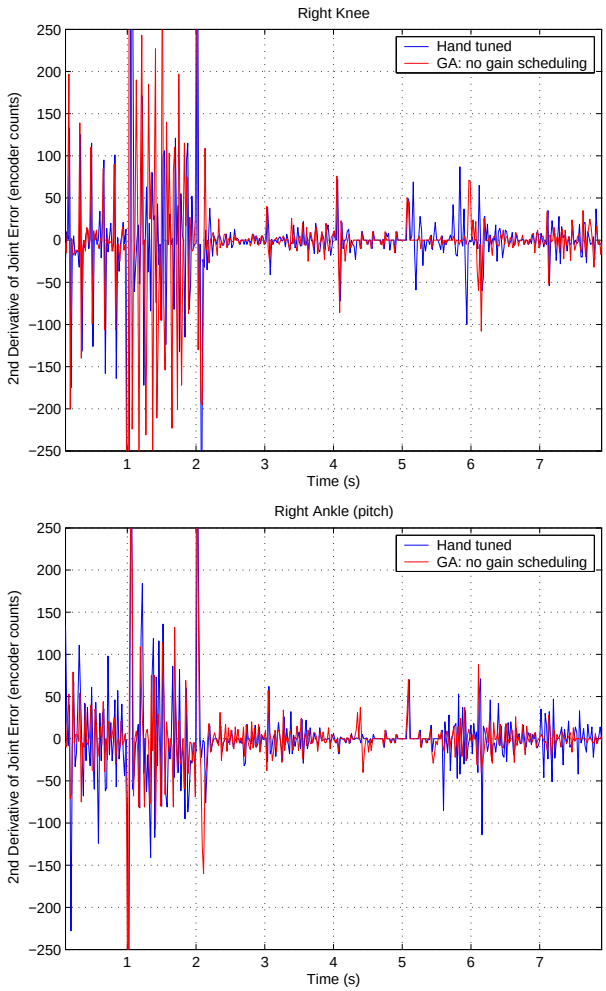

Figure 10: Comparison of joint error smoothness during real walking for the right knee (top) and right ankle (bottom).

[Goldberg, 1989] David Edward Goldberg. *Genetic Algorithms in Search and Optimization*. Addison-Wesley Pub. Co., 1989.

[Herrero *et al.*, 2002] J. M. Herrero, X. Blasco, M. Martinez, and J.V. Salcedo. Optimal PID tuning with genetic algorithms for non linear process models. In *IFAC 15th Triennial World Congress*, Barcelona, Spain, 2002.

[Kee *et al.*, 2003] D. Kee, G. Wyeth, A. Hood, and A. Drury. Guroo: Autonomous humanoid platform for walking gait research. In *Proceedings of the Conference on Autonomous Minirobots for Research and Edutainment (AMiRE2003)*, Brisbane, Australia, February 2003.

[McMillan, 1995] S. McMillan. *Computational Dynamics for Robotic Systems on Land and Underwater*. PhD thesis, Ohio State University, USA, 1995.

[O'Dwyer, 2003] Aidan O'Dwyer. *Handbook of PI and PID Controller Tuning Rules*. Imperial College Press, 2003.

[Porter and Jones, 1992] B. Porter and A. H. Jones. Genetic tuning of PID controllers. *Electronics Letters*, 28(9):843–844, 23 April 1992.

[Vukobratovic and Juricic, 1969] M. Vukobratovic and D. Juricic. Contribution to the synthesis of biped gait. *IEEE Transactions on Biomedical Engineering*, 16(1), 1969.

[Wall, 2000] Matthew Wall. GAlib C++ library. http://lancet.mit.edu/ga/, 2000.

[Wang and Kwok, 1994] P. Wang and D. P. Kwok. Optimal design of PID process controllers based on genetic algorithms. *Control Engineering Practice*, 2(4):641–648, 1994.

[Wyeth *et al.*, 2001] G. Wyeth, D. Kee, Mark Wagstaff, Nathaniel Brewer, Jared Stirzaker, Timothy Cartwright, and Bartek Bebel. Design of an autonomous humanoid robot. In *Proceedings of the Australian Conference on Robotics and Automation (ACRA 2001)*, Sydney, Australia, November 2001.

[Wyeth *et al.*, 2003] Gordon Wyeth, Damien Kee, and Tak Fai Yuk. Evolving a locus based gait for a humanoid robot. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2003)*, Las Vegas, USA, October 2003.