Investigating the Use of Iterative Learning Control and Repetitive Control to Implement Periodic Gaits

R.W. Longman[1] and K.D. Mombaur[2]

[1] Dept. of Mechanical Engineering, Columbia University, New York, 10027, USA
rwl4@columbia.edu

[2] IWR, Universität Heidelberg, Im Neuenheimer Feld 368, 69120 Heidelberg, Germany
katja.mombaur@iwr.uni-heidelberg.de

Summary. In the next few years considerable effort will be expended to make humanoid robots that can do true dynamic walking, or even running. One may numerically compute a desired gait, e.g. one that has been optimized to be asymptotically stable without feedback. One would normally give the gait as commands to the controllers for the robot joints. However, control system outputs generally differ from the command given, and the faster the command changes with time, the more deviation there is. Iterative learning control (ILC) and repetitive control (RC) aim to fix this problem in situations where a command is repeating or periodic. Since gaits are periodic motions, it is natural to ask whether ILC/RC can be of use in implementing gaits in hardware. These control concepts are no substitutes for feedback control but work in combination with them by adjusting the commands to the feedback controllers from a higher level perspective. It is shown that the gait problem does not precisely fit either the ILC or the RC problem statements. Gait problems are necessarily divided into phases defined by foot strike times, and furthermore the durations of the phases are not the same from cycle to cycle during the learning process. Several methods are suggested to address these issues, and four repetitive control laws are studied numerically. The laws that include both position and velocity error in the updates are seen to be the most effective. It appears that with appropriate refinement, such generalized RC laws could be very helpful in getting hardware to execute desired gaits.

1 Introduction

In the last few years, many humanoid and biped walking robots have been built executing periodic or quasi-periodic gaits [1, 2]. So far such robots are rather slow moving compared to their biological counterparts, and the traditional control approach keeps them as close as possible to a quasi-static equilibrium during the motion, e.g. [5, 25]. As research progresses into making robots that do true dynamic walking or running, in addition to dealing with the dynamic behavior of the nonlinear multibody robot system, it will become necessary to address the imperfect dynamics of any feedback control system that is used to attempt to execute the chosen periodic gait.

In recent years, the fields of iterative learning control (ILC) and repetitive control (RC) have appeared [3, 19], the goal of which is to improve the performance of feedback control systems by adjusting the commands given to them. In principle, these techniques could be combined with any chosen feedback control concept. ILC and RC have now developed to the point that commercial use of the methods has started to appear. Recently, robots have been delivered to Daimler-Chrysler in Detroit using ILC to improve performance. And RC is now being used at the factory to learn a command for each track of computer disk drives. The improved performance allows the tracks to be closer together, and hence the disks can store more data. Similar methods are being used to speed up chip manufacturing, allowing the manufacturing hardware to operate faster while maintaining the needed precision, and hence increase productivity.

ILC suddenly began to develop quickly in 1984 motivated by robots performing repetitive operations in a manufacturing environment. Each time the task is performed, the robot restarts from the same initial conditions. When a feedback control system is given a tracking command, the response is not the same as the command, even under ideal circumstances with perfect measurements and no plant noise disturbances. Generally, the faster the requested motion in the tracking command, the larger the discrepancy between what is asked for and what is produced. The purpose of ILC is to use the error observed in the previous run (or repetition) to adjust the command in the current run, aiming to converge to that command that actually produces the desired trajectory. ILC asks the control system to execute commands that are not what you want the controllers to do, so that they actually do what you want them to do.

RC is a closely related type of control. Instead of making repeated runs of a desired finite time trajectory, each time starting from the same initial condition, RC aims to perfectly execute a periodic command, or to perfectly execute a constant command in the presence of a periodic disturbance (or to execute a periodic command with a periodic disturbance, each having the same period). The RC law learns from the measured error in the previous period (or cycle) instead of the previous run, adjusting the command in the current period, aiming to get to zero tracking error. Transients can propagate from one period to the next in RC, but cannot go from one run to the next in ILC, and this results in the two problems having different conditions for stability, i.e. for convergence to zero tracking error.

As gait research progresses from relatively slow robot walking motions to full dynamic walking, and then to running, the issues of imperfect execution of high speed commands by feedback control systems will become a serious issue. The desired periodic gaits are commands to the feedback controllers for each robot joint. If the discrepancy between the commanded trajectory and the trajectory executed by the feedback controllers is large enough, the robot will fall. Since ILC and RC are new fields of control theory that address how to make feedback control systems actually perform desired repeating or periodic motions, it is natural to ask whether ILC or RC can be used to implement high speed gaits. It is the purpose of this paper to make an initial evaluation of what ILC/RC can do for this problem, and to put forward some concepts that might address the issues that are raised.

2 Feedback Control System Errors that ILC/RC can Fix

Consider the performance of typical feedback control systems executing a time varying command. Suppose one wishes to control the output $y(t)$ of a first order system (the plant) $dy/dt + ay = w + v$ where $w(t)$ is a variable we can manipulate to change y, e.g. we can apply a torque to a robot link to make the output angle change. Typically, whatever variable we can manipulate, nature can also influence with various disturbances, e.g. in the case of a robot link, gravity can supply a torque history as the link follows the desired path, and v denotes such disturbances. Now consider applying a proportional controller to make the output follow a command. Then $w(t) = Ke(t) = K(y_C(t) - y(t))$ and the manipulated variable w is proportional to the measured error $e(t)$, the command $y_C(t)$ minus the measured output $y(t)$. The performance is then predicted by the closed loop differential equation

$$\frac{dy(t)}{dt} + (K+a)y(t) = Ky_C(t) + v(t) \tag{1}$$

whose solution is

$$y(t) = e^{-(K+a)t}y(0) + \int_0^t e^{-(K+a)(t-\tau)}Ky_C(\tau)d\tau + \int_0^t e^{-(K+a)(t-\tau)}v(\tau)d\tau \tag{2}$$

The middle term on the right is the part of the solution that is responding the command we give the system. But it is not equal to the command. Instead it is a convolution integral of the command, creating a form of weighted average of all the commands we have given the system in the past. The weighting factor decays going backward in time, so that more recent commands are more important in the weighted average. Therefore, for any command that is changing with time, the feedback control system does this convolution integral of what we asked it to do, not what we asked it to do. And the faster the command changes with time, the more effect the averaging has on the result. The first term on the right represents transients and is the response to initial conditions. The last term on the right gives the effect of disturbances on the performance of the control system. If the system has the same disturbance every time we give the command, then this disturbance is also a source of repeating error.

It is the purpose of ILC and RC to converge to a command $y_C(t)$ that is no longer equal to the desired output $y_D(t)$, but one which has the property that the system output given by the right hand side of (2) is in fact equal to $y_D(t)$. In the case of ILC the command has the property that over a finite time interval starting at time zero, the right side of (2) converges to the desired trajectory as the repetitions progress. In the case of RC, the command has the property that as time increases, the right hand side converges to the desired trajectory. Both fix deterministic errors in following a command, and also cancel any disturbances that repeat. ILC also learns to handle the first term on the right, since it is present in every run, while RC learns to get zero error as time goes to infinity, and for an asymptotically stable system the first term disappears with time.

The iterations solve an inverse problem of finding the input that produces the desired output. In the simple example above, one can directly solve this inverse problem, e.g. using the desired output $y_D(t)$ for the output in (1) the command needed is:

$$y_C(t) = \frac{1}{K}\left(-\frac{dy_D(t)}{dt} - (K+a)y_D(t) + v(t)\right) \tag{3}$$

There are usually difficulties with this in practice. First, when done in discrete time, the inverse problem is usually ill-conditioned [11]. Second, if the external disturbance $v(t)$ is an important aspect of the problem, one needs to know this function which may be difficult. And third, the solution is only as good as the model is. ILC and RC find an inverse solution iterating with the real world behavior, instead of the model, without needing to know $v(t)$, and without totally relying on a model.

3 ILC/RC Background

The most effective ILC/RC design methods are based on linear systems theory, and the discussion presented here is limited to this approach. Results have been generated for doing nonlinear ILC on equations having the form of multibody dynamic systems. These results are likely to be not as practical as the linear design methods. First, they rely on all of the dynamics in the physical system having the multibody dynamics form, which is often not the case when actuators and sensors and effective feedback controllers are included. Second, they create very complex control laws that are more complicated to implement. And third, linear methods as in [4] can converge to tracking errors that approach the minimum possible error, the repeatability level of the system, and do so relatively quickly without requiring the complicated modeling of the nonlinear system. Figure 1 shows the robot used in [4], and

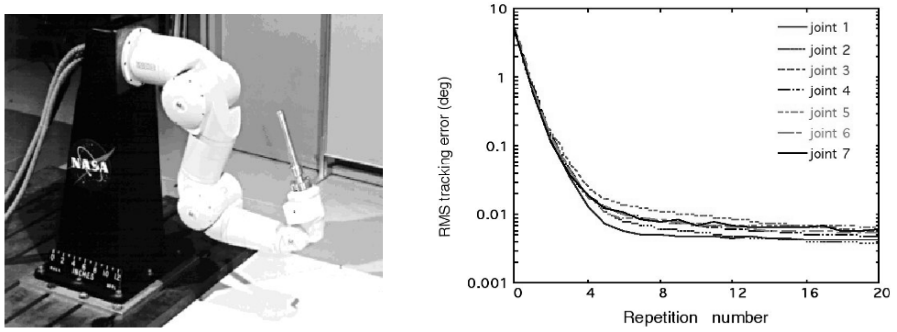


Fig. 1. Robotics Research Corporation robot, and RMS error vs. repetitions using an ILC law

also the tracking error for each ILC repetition for each robot link following a high speed trajectory versus repetitions. The error is decreased by a factor of roughly 1000 in about 12 runs. Note that the final error level is actually below the repeatability level of the robot when measured on a day to day basis, so the ILC is fixing errors of the size of how different the robot behaves from one day to the next. No amount of modeling could predict such errors. The fact that ILC does not rely heavily on a model allows it to fix such errors. Nevertheless, in gait problems one might want to revisit the question of the usefulness of using fully nonlinear methods, or at least using some form of feedback linearization.

There is a related issue for robot applications. One may consider creating an ILC or RC law that has multiple inputs and outputs, one for each of the joints variables. On the other hand, it is much simpler to use a decentralized ILC or RC approach, that applies a separate independent law to each of the separate feedback control systems for each robot joint as if there were no coupling between joints in the nonlinear dynamics. Again, the results in [4] are obtained using decentralized ILC, suggesting that the simple decentralized approach can be very effective in robot applications.

Both ILC and RC must necessarily be implemented by digital control methods, because the control updates are based on data from a previous repetition or a previous period, and therefore must be measured and stored in a computer or microprocessor. One will normally use a zero order hold on the input signal that the ILC or RC adjusts. Consider ILC. The objective is to perform a finite time trajectory, and get zero error at the sample times, i.e. we want the output $y(kT)$ to converge to the desired output $y_D(kT)$ for $k = 1, 2, 3, \ldots, N$. Here T is the sample time interval of the digital control system, and the desired trajectory is N steps long. The error is $e(kT) = y_D(kT) - y(kT)$. The simplest form of ILC is based on pure integral control concepts being applied in repetitions to each of the time steps of the problem. Stated in words for a robot link, if the robot link were 2 degrees too low at a certain time step in the last run or repetition, then add 2 degrees, or 2 degrees times a learning gain ψ, to the command this repetition. Mathematically, this is written as

$$u_j(kT) = u_{j-1}(kT) + \psi e_{j-1}((k+1)T) \tag{4}$$

where j is the current repetition number, $j-1$ is the previous repetition. Based on the ILC description above, u represents the command to the feedback control system, but we note that some of the ILC literature accomplishes the learning by modifying the error signal going into the controller or the manipulated variable coming out of the controller instead, and then the u in (4) either represents the error signal from the comparator or the w output of the controller as discussed above (1) [20]. The +1 in the argument of the error is introduced to account for the one time step delay going through the feedback control system (or the plant equations), i.e. there is usually a one time step delay from the time step in which one changes the command (or the manipulated variable) to the first time step in the output where a resulting change is observed. The computations of the command (or manipulated variable) history to use in the next run can be made between runs, computed in a batch mode.

The RC equivalent of this learning law is used when one wants to execute a periodic command, and this time the period is N time steps long. The mathematical expression of the corresponding RC law becomes

$$u(kT) = u((k-N)T) + \psi e((k-N+1)T) \tag{5}$$

Instead of looking back to a previous repetition, one looks back one period. Note that unlike ILC which makes a batch update of the command history for the next repetition, RC normally runs with updates made every time step, in real time.

ILC law (4) is almost always stable for sufficiently small gains ψ, but the learning transients are very likely to be impractical [8]. However, there is an important exception to this that occurs when the sample time of the ILC updates is sufficiently long that the system comes close to a steady state response before the next update arrives. RC law (5) is usually unstable. In both cases the error may decrease very substantially in the first few iterations or periods, and then the error starts to grow [9, 10]. It can be that one is satisfied with this level of improvement and simply stops the process when the error is a minimum. To improve performance and obtain stability robustness one normally generalizes the above laws to include a dynamic compensator in place of the gain ψ, and introduces a zero-phase low-pass filter cutoff of the learning [4, 24, 22, 21, 8]. Equations (4) and (5) can take the form

$$\underline{u}_j = F\left[\underline{u}_{j-1} + L\underline{e}_{j-1}\right] \tag{6}$$

$$U(z) = F(z)z^{-N}\left[U(z) + L(z)E(z)\right] \tag{7}$$

In (6) the underbar indicates a column matrix of the history of the associated variable in a repetition, and F and L are matrices representing the low pass filter and the compensator, respectively. Equation (7) for RC converts to the z-transform domain with $F(z)$ and $L(z)$ being the transfer functions of the cutoff filter and the compensator.

4 Dynamic Models for Walking Motions

The purpose of this section is to present the mathematical models of walking to which the concepts of ILC/RC will be applied. We start by giving the general form of dynamic walking models, and then present the specific stiff-legged biped walker used for the numerical tests later in this paper.

Mathematical models of gaits involve distinct model phases with possibly different degrees of freedom, each described, in the general form, by a different set of differential equations. These can be ordinary differential equations (ODEs)

$$\dot{q}(t) = v(t) \tag{8}$$

$$\dot{v}(t) = a(t) = M^{-1}(q(t), p) \cdot f(q(t), v(t), w(t), p) \tag{9}$$

In these equations, the vector q contains the position variables of the system, and v the corresponding velocities; together they form the vector of state variables $x^T = (q^T, v^T)$. The vector y used in the context of ILC and RC is typically equal to q. The scalar t is the physical time, a the vector of accelerations, $w(t)$ are the input torques or forces of the robot, and p is the vector of model parameters (like geometric or inertial data). M denotes the mass matrix, and f the vector of forces.

Alternatively, depending on the choice of coordinates, one may obtain a system of differential-algebraic equations (DAE) of index 3 for some or all phases

$$M(q(t), p) \cdot a = f(q(t), v(t), u(t), p) - G^T(q(t), p)\lambda \tag{10}$$

$$g_{pos}(q(t), p) = 0 \tag{11}$$

with the Lagrange multipliers λ, the constraint equations g_{pos}, and their partial derivatives $G = \frac{\partial g_{pos}}{\partial q}$. We formulate the DAEs in the equivalent index 1 form with invariants

$$\dot{q}(t) = v(t) \tag{12}$$

$$\dot{v}(t) = a(t) \tag{13}$$

$$\begin{pmatrix} M(q(t), p) & G^T(q(t), p) \\ G(q(t), p) & 0 \end{pmatrix} \begin{pmatrix} a \\ \lambda \end{pmatrix} = \begin{pmatrix} f(q(t), v(t), w(t), p) \\ \gamma(q(t), v(t), p) \end{pmatrix} \tag{14}$$

$$g_{pos} = g(q(t), p) = 0 \tag{15}$$

$$g_{vel} = G(q(t), p) \cdot \dot{q}(t) = 0 \ . \tag{16}$$

with the abbreviation

$$\gamma(q(t), v(t), p) = -v^T \, \frac{d\, G(q(t), p)}{d\, q} \, v \; . \tag{17}$$

Phase boundaries are implicitly defined by the roots of switching functions

$$s_i(t, q(t), v(t), p) = 0 \; . \tag{18}$$

At these switching points, there may be discontinuities in the right hand side of the linear system, i.e. $\Delta f(q, v, w, p), \Delta\gamma(q, v, p)$ (which translates into discontinuities in the accelerations Δa), or even in the velocities, $\Delta v(t, q, v, w, p)$, i.e. in the state variables themselves. Walking problems also involve a number of complex linear and nonlinear, coupled and decoupled equality and inequality constraints; e.g. the periodicity constraints on the state variables (or a subset thereof) $\tilde{x}(T_{cycle}) = \tilde{x}(0)$. The cycle time T_{cycle} is generally a priori unknown. In this paper, we investigate the simple example of a planar stiff-legged biped walker with two degrees of freedom. The state variables of this robot are the stance leg angle ϕ_1 and the swing leg angle ϕ_2, and the corresponding velocities $x^T = (\phi_1, \phi_2, \dot{\phi}_1, \dot{\phi}_2)$. The robot has two torque actuators - one corresponding to each degree of freedom - the first one $w_1(t)$ at the hip, and the second one $w_2(t)$ at the ankle to replace the action of a foot with an actuated ankle joint. For repetitive control problems it is convenient to introduce a second set of state variables corresponding to the torques with $\bar{x}^T = (\theta_1, \theta_2, \dot{\theta}_1, \dot{\theta}_2)$, where

$$\theta_1 = \phi_1 - \phi_2 \tag{19}$$

$$\theta_2 = \phi_1 \tag{20}$$

This model can be considered as an extension of the classical passive-dynamic stiff-legged bipeds of McGeer [12]. The robot is shown in Fig. 2. It is characterized by three free parameters $p = (m, l, c)^T$ with

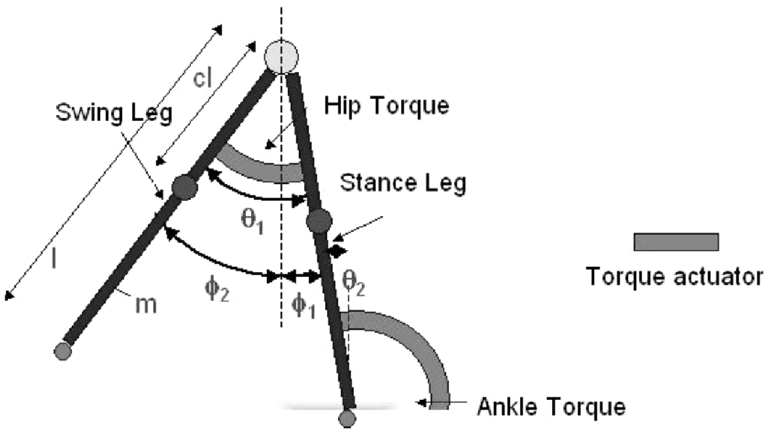


Fig. 2. Sketch of stiff-legged biped investigated in this paper

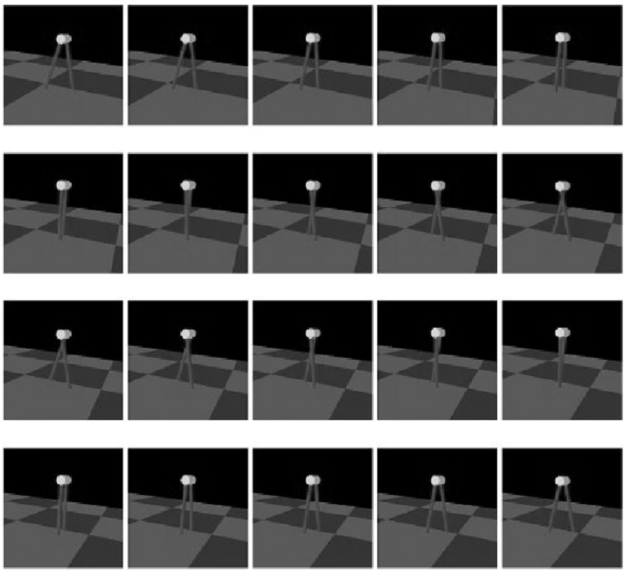


Fig. 3. Two steps of periodic reference solution of stiff-legged biped

- the mass of each leg, denoted by m
- the leg length l
- the relative location of the leg's center of mass measured from the hip, c.

Using these three parameters, the moment of inertia Θ of a leg is defined as

$$\Theta = \frac{1}{6} m l^2 (1 + 2c^2 - 2c) \ . \tag{21}$$

One cycle of the gait model includes one step of the robot followed by a leg switch, and not a full physical gait cycle consisting of two steps (as presented in Fig. 3). Applying periodicity constraints to this model assures the generation of equal right and left steps, which would not necessarily be the case otherwise. One cycle of this model consists of one continuous phase describing the forward swing and a discrete phase including the sudden change of velocities at foot contact and the leg switch.

The dynamic equations of this robot model are

$$M \cdot \begin{pmatrix} \ddot{\phi}_1 \\ \ddot{\phi}_2 \end{pmatrix} = F \tag{22}$$

with mass matrix

$$M = \begin{pmatrix} m_{11} & m_{12} \\ m_{21} & m_{22} \end{pmatrix} \tag{23}$$

$$\text{with } m_{11} = 2ml^2 + \Theta - 2ml^2c + ml^2c^2 \tag{24}$$

$$m_{12} = ml^2c \sin\phi_2 \sin\phi_1 - ml^2c \cos\phi_2 \cos\phi_1 \tag{25}$$

$$m_{21} = ml^2c \sin\phi_2 \sin\phi_1 - ml^2c \cos\phi_2 \cos\phi_1 \tag{26}$$

$$m_{22} = ml^2c^2 + \Theta \tag{27}$$

and force term

$$F = \begin{pmatrix} -m\dot{\phi}_2^2 l^2 c \sin\phi_2 \cos\phi_1 + m\dot{\phi}_2^2 l^2 c \cos\phi_2 \sin\phi_1 \\ +2mgl \sin\phi_1 - mgl \sin\phi_1 c + w_1 + w_2 \\ \\ -m\dot{\phi}_1^2 l^2 \sin\phi_1 c \cos\phi_2 + m\dot{\phi}_1^2 l^2 \cos\phi_1 \\ c \sin\phi_2 - mglc \sin\phi_2 - w_1 \end{pmatrix} \tag{28}$$

The end of the step is determined by the equation

$$s(t, x, p) = \phi_1 + \phi_2 = 0 \tag{29}$$

The torques at hip and ankle are produced by feedback control systems using proportional control and rate feedback following a commanded trajectory $\theta_{i,C}$:

$$w_i(t) = K_1(\theta_{i,C} - \theta_i) - K_2\dot{\theta}_i \tag{30}$$

More details about this robot model as well as a description of possible extensions using springs and dampers in parallel with the torque actuators are given in [18].

5 Open Loop Stable Gaits

Previous research by the authors established that it is possible to have running gaits that are open-loop stable, meaning that they will return to the stable gait after small enough disturbances to position and velocity. This is accomplished without any feedback adjustment of the torque histories being applied to each joint. In the motions of ballet dancers and athletes, one suspects that there is often some inherent stability of the motions used, and we see that in running, hopping and somersaults this is also possible for robots [14, 17, 16]. In is generally preferable to create systems that are open loop stable and use feedback to augment the stability, than to rely totally on feedback to stabilize the motion. Pilots prefer to fly airplanes that do not immediately go unstable if there is a failure in the attitude control system.

With this in mind, we consider implementing an open loop stable gait to test the principles of ILC and RC. Numerically solving for such gaits for actuated robots is not trivial and requires an appropriate selection of model parameters p and driving torques $w(t)$. We have developed special numerical optimization techniques [17, 18] that can produce self-stabilizing solutions for walking robots. They have been applied to stabilize a series of different monopod and biped robots [15], one new example is given in another paper in this proceedings volume [13]. As a stability measure, we use the spectral radius of the Jacobian of the Poincaré map which is associated with the periodic solution. If the spectral radius is smaller than one, the solution is asymptotically stable, and if it is larger than one, the solution is unstable. For the stiff-legged biped described above, we have determined an open loop stable solution that is characterized by a spectral radius of 0.7, well below one, but which is also very efficient and requires only small torque inputs. The parameters p of this solution are $m = 1$ kg, $l = 0.1\,\mathrm{m}$, and $c = 0.25$; the cycle time is 0.4586 s. The initial values are $x_0^T = (0.25, -0.25, -1.943, -2.688)^T$. The corresponding torque inputs as well as the trajectories of the angles and rates are shown in Fig. 4. More information about the solution, as well as the objective functions used to create it, are given in [18].

6 Learning to Execute Open Loop vs. Closed Loop Stable Gaits

6.1 Problem Statement

One can pose a couple of different kinds of gait problems that might benefit from use of ILC or RC:

- **Problem 1.** As discussed above, one expects that there are benefits to using gaits that are open loop stable, so that there is already some inherent stability to the motion. An open loop stable solution obtained numerically gives a torque input history for each joint, the resulting output history or gait, and its derivative. The next step is to design a feedback control system for each link, since ILC and RC normally adjust the command to feedback control systems. The objective of the ILC/RC is to succeed in making the control system in hardware execute the chosen gait, i.e. the chosen time histories for each joint angle.
- **Problem 2.** If one were not concerned with open loop stable solutions, one could include the feedback control system equations with the robot dynamic equations, and design the gait using as inputs the position command histories given to the feedback control system instead of the torques applied to the robot dynamics. One can include the controller gains as parameters to be optimized (as well as the original model parameters p above) while finding the gait based on a chosen optimality criterion. Of course, the solution is most likely not open loop stable. Because the feedback controller equations have been used in the design of the gait, the hardware would

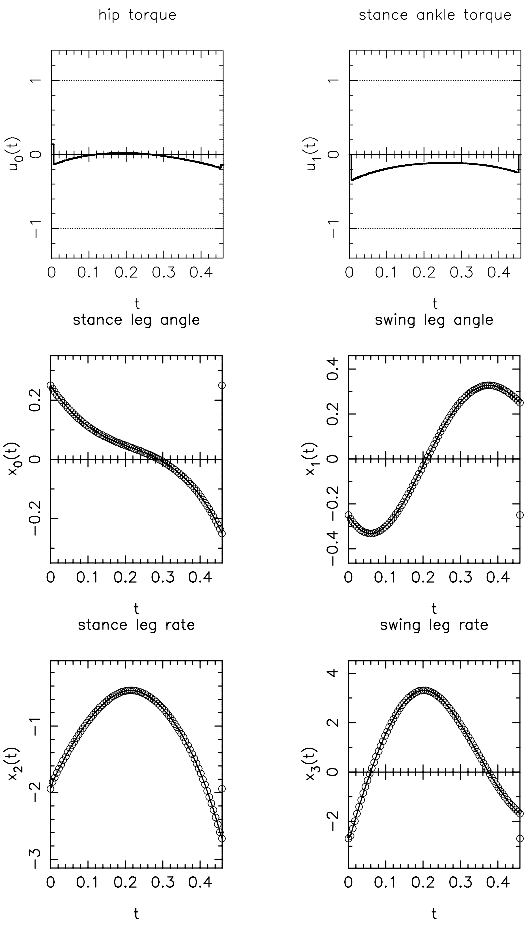


Fig. 4. Torques and states for open-loop stable solution

actually perform the gait desired if the model used for both the robot dynamics and the control system dynamics perfectly represent the behavior of the hardware. The ILC/RC might be used to perfect the execution of the gait by fixing errors caused by inaccurate or incomplete modeling of the robot dynamics, actuators, and control system components.

6.2 Implementation Issues

For both problems the ILC/RC result serve as the basis for implementing the desired motion on level ground. With sufficient stability perhaps this is all that one needs for reasonably level ground. For more uneven terrain, one may next try to design an outer loop that adjusts the commands to the feedback control systems to handle such things as uneven ground. The outer loop is analogous to the upper level trajectory generator in industrial robots and might use additional sensor information such as vision feedback to modify the gait mode to adapt to the terrain.

This paper will consider issues in addressing Problem 1. ILC/RC are methods of solving inverse problems: given a desired output of some dynamic system or component, find the input that would produce that output. It is normally done with the physical robot hardware, but of course one can also use a mathematical model. This gives rise to three possible levels of implementing ILC/RC:

(i) Do the iterations on the hardware.
(ii) Do the iterations on a computer using a model. If the model is good enough, this can work when the solution is applied to the hardware.
(iii) Do (ii) to get an initial command to give in hardware, and then continue the ILC/RC iterations in hardware to correct for any deficiencies in the model.

The numerical studies reported below, directly illustrate (i) where one presumes the computer model is functioning as the real world model. They also automatically illustrate the process one goes through in performing (ii). And then by introducing some changed parameter, i.e. inaccurately modeled parameter, and continuing the learning process one can illustrate (iii).

There are two short cuts for accomplishing (ii), one of which simply eliminates the need for (ii) altogether. These are: use of torque feedforward in the control system design, and do an inverse problem on the controller alone, instead of the complete system.

6.3 Torque Feedforward

Perhaps the most logical implementation of the open loop stable gait solution is to use torque feedforward as in Fig. 5. The solution is a torque history $w(t)$, a desired output history or gait, $y_D(t)$, and its derivative $\dot{y}_D(t)$. Since we consider a decentralized implementation, there is a separate controller for each joint angle with its own desired output history $y_D(t)$. If the actuator actually applies this torque to the robot links, and we give $y_D(t)$ as the command $y_C(t)$, and if the robot model was perfect, then the error signal $e(t)$ would be zero in the block diagram. Then the feedback only starts to function if there is some deviation from the desired trajectory. Several comments apply:

(1) The actuator may not apply the torque we want. If it is a DC motor with voltage input being adjusted by the controller (and the feedforward signal) it will not apply the intended torque, but if one can use current as the input it will accomplish the goal if one knows the motor constants correctly. In order to try to do this, one often uses a current feedback loop around the motor. In addition, the back electro motive force (emf) introduces a rate feedback inherent in the motor. Hence, the actuator has dynamics of its own and will not exactly apply the intended torque.

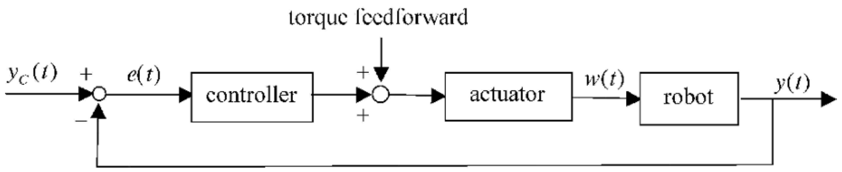


Fig. 5. Feedback control system using torque feedforward

(2) In positioning systems it is often desirable to use rate feedback, meaning the feedback loop takes a measurement of the output, makes a separate measurement of the output rate, which is multiplied by a constant and then the two are added together to produce the signal subtracted from the command to get the "error", $e(t)$. If this is being done, then one must compute what this signal would be when the output is the desired output, and use it as the command given the control system.
(3) The approach totally avoids solving any inverse problem. But it does not fix any errors related to use of an imperfect model or an imperfect actuator, although the feedback loop may make partial adjustments. Then one can apply ILC/RC in hardware to fix remaining errors.

6.4 Inverting Controller Equations

Consider the feedback control block diagram of Fig. 6 including the rate feedback that is typically used in robotics. The usual ILC/RC application solves the inverse problem, given the desired output $y(t) = y_D(t)$, find input $y_C(t)$ to produce it. In the process of having solved for the desired periodic gait, we know more than just the desired output, we also know its derivative, and the torque $w(t)$. Therefore, we can instead solve the inverse problem for the blocks introduced for control: given output $w(t)$, desired position history $y_D(t)$ and its velocity (which together determine the feedback signal) find $y_C(t)$. In the examples below, we use an idealized proportional control system with rate feedback, and in this simple case doing the suggested inverse problem is simple and immediate. Suppose that the actuator can be represented by a simple gain, and this gain can be combined with the proportional control gain in the controller block, and the product called K_1. The feedback signal is $y_D(t) + K_2\dot{y}_D(t)$, where K_2 is the rate feedback gain. Then

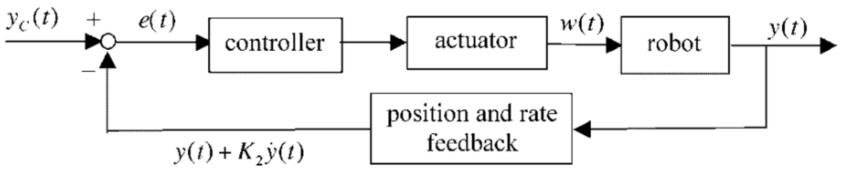


Fig. 6. Feedback control system using rate feedback

$$w(t) = K_1 \left(y_C(t) - [y_D(t) + K_2\dot{y}_D]\right) \tag{31}$$

Substituting the computed open loop stable time histories, one can solve for the needed command $y_C(t)$ to produce the gait. In simple situations one can do this. In the real world the control system design is likely to be more complicated, requiring inversion of dynamic equations, which is the domain of ILC/RC. Classical control system designers are likely to introduce a low pass filter, possibly as a noise filter, possibly as a way to control the bandwidth of the system to avoid exciting vibrations. They are likely to introduce compensators to modulate the frequency response behavior, which introduces extra dynamics with both poles and zeros. And, as discussed above, the actuator can have some dynamics. Just introducing the back emf of a motor puts in a rate feedback loop feeding the motor, which is missing in the block diagram. If one has a full computer model of all of this, one can aim to solve this inverse problem, and ILC/RC might again be an appropriate method to use.

7 Some Non-Standard ILC/RC Problems

The gait problem does not immediately fit the ILC or RC descriptions. The following two non-standard ILC/RC problems address some of the issues related to gait problems.

Timing Belt Drive Problem. Figure 7 shows a double reduction timing belt drive system that might be used in a copy machine when one needs to have a very uniform velocity of the output shaft. Using a well designed velocity feedback control system, the frequency content of the velocity error is given in Fig. 8 (left) [6, 7]. All of the peaks are related to inaccuracies in the gearing, and include errors that have the periods of one rotation for each shaft, and each belt, including fundamental and harmonics. In addition, the large peaks at 80 Hz and 240 Hz are at the frequencies for tooth meshing of each timing belt. Because gearing is involved, all of these frequencies have a common period which can be used by a repetitive control system to eliminate the peaks. The best experimental result is shown in Fig. 8 (right) where all of the peaks have been eliminated. However, this problem does not completely

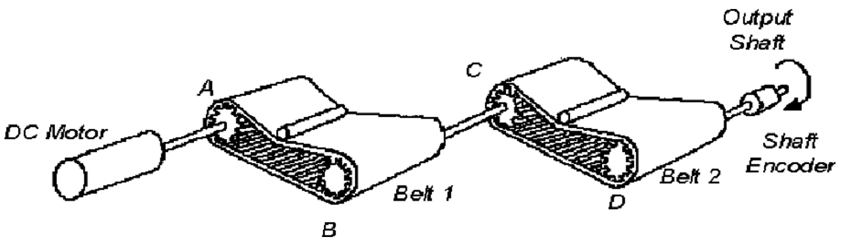


Fig. 7. Double reduction timing belt drive system

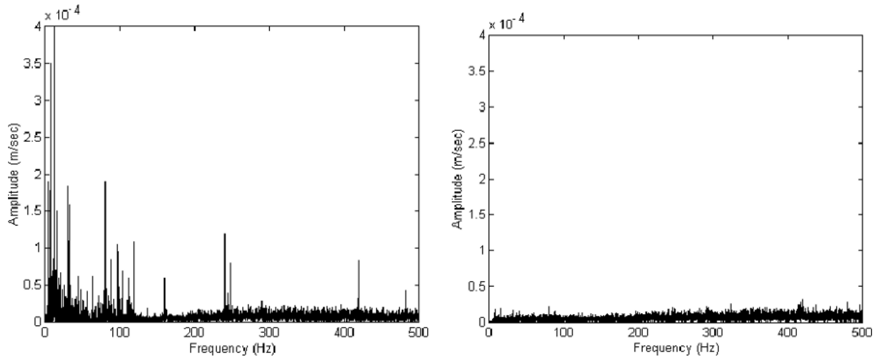
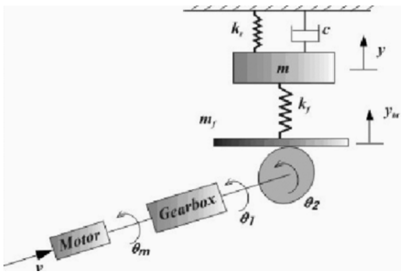


Fig. 8. Frequency spectrum of velocity error of timing belt system using feedback only (*left*) and at repetition 50 using batch phase cancellation RC (*right*)

fit the standard repetitive control assumptions, because as the velocity varies, so does the period. The error to be corrected is actually periodic with the output shaft rotation angle, not with time. To address this issue, these experiments used an index pulse on the output shaft to know when the next period started. The data from the previous period was taken at evenly spaced time steps, and sometimes had more steps than the correct number, and sometimes had less. Some ad hock rule was used to decide what to do when there was missing data for the update. If the period had varied more, one might have made some adjustments to match the time scales for each period. If one has measurements of the angle at each time step, one could do interpolation in the data to explicitly make updates related to each angle instead of each time. There are in fact many repetitive control applications that have this same modeling difficulty.

Cam Problem. Figure 9 (top) shows a cam follower system driven by a DC motor, nominally at a constant speed. The cam is normally designed with its lift curve, the following dwell, the return curve, and the subsequent dwell, all assumed to be made with a prescribed constant cam rotation rate. Of course, as the cam rotates the lever arm to the contact point lengthens and shortens making the resistance to rotation vary with cam angle. The change in resistance to motion is a disturbance to any speed control system. This means that the resulting lift displacement history, dwell durations, and return history as a function of time are not the intended histories or durations. In addition, there are imperfections in the machining of the cam. Suppose we would like to fix these errors by using the velocity control system for the shaft rotation rate to speed up and slow down the cam in such a way as to have the lift and return curves be the intended functions of time, and have the dwell parts be the desired time durations.



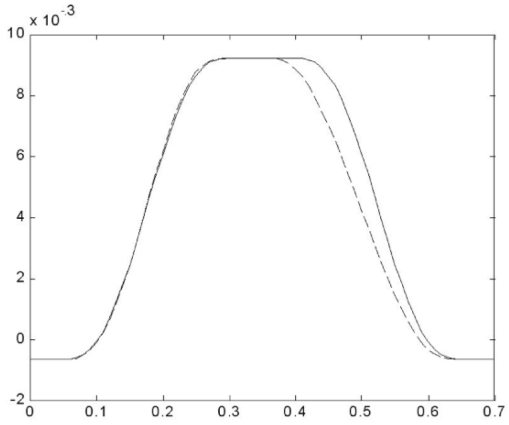


Fig. 9. Cam follower system, and comparison of desired cam displacement (*solid line*) and displacement after 50 repetitions

This problem has several unusual aspects. First, there are four phases, the rise, the dwell, the return, and the following dwell. Second, during the dwell phases presumably the cam has a constant radius so that there is no change in lift, but even if this is not true one is not able to fix the problem by changing the speed. So the only objective to be accomplished in the dwell phases is to make sure they have the right time durations, so that the next phase starts at the right time. Reference [23] reports both simulations and experiments in learning to get the desired output curves. What was done was to learn the lift phase first. Once it converged to a satisfactory error level, the iterations to learn the next phase started. The dwell phases computed the average velocity of the dwell from the end minus start angles, divided by the end minus start times. The error in this average velocity was multiplied by a gain such as ψ and added to the command given in the previous learning cycle to produce the command during the dwell this cycle. Figure 9 (bottom) shows an example of learning iterations after the lift part of the curve has been learned, and iterations are being made to get the top dwell right. Errors in the return are being ignored until the learning of the top dwell is complete. It is fortunate in this problem that the initial conditions for the phase being learning are not heavily dependent on whether the previous phase has been learned already, because the dwell phases allow time for decay of transients. Hence, this problem was treated as a sequence of ILC problems for each phase learning one at a time. The simple learning law of equation (4) was used, but with a chosen time step lead in the error term, creating a linear phase lead compensation [8]. No attempt was made to use a zero-phase low-pass filter to stabilize the process. This was done partly for simplicity, and partly because of nonlinearities in the system. A relatively long sample time for the ILC updates was used to slow the growth of the instability, and the learning process was stopped before the instability started to become evident in the behavior.

8 Approaches to Handling the Non-Standard Nature of Gait Problems

Applying ILC or RC to the gait problem has some of the same issues as the above problems, but introduces additional difficulties as well. The equations are highly nonlinear, and include jump discontinuities. It could be a challenging problem to deal with the nonlinear nature in some direct way, and one would prefer to try to approach the problem as in the nonlinear cam problem, using a simple learning law that might improve the error substantially before an instability sets in, and then stop the learning when the error is a minimum. Note however, that such methods are more successful in ILC than RC, and the gait problem seems to have more relationship to RC problems. We comment that learning high frequency components can be slow and can create stability problems, and discontinuities even in the derivative of variables being controlled introduce some high frequency components. The problem has distinct phases with ground impacts denoting the start of a new phase. As in the timing belt problem, the period or duration of a phase varies with each cycle, the index pulse or impact indicating when the next cycle or phase starts. As in the cam problem the duration of each phase is important. The phases in the cam problem used different learning laws and started with reasonably repeating initial conditions when learned in sequence, so treatment as a sequence of ILC problems worked. But like RC the initial conditions for each period or phase in the gait problem do not repeat until convergence, indicating use of an RC formulation. The RC control laws from phase to phase need not be particularly different, but the fact that the phases have different duration may introduce jumps in the learning process across phase boundaries, the same points where there can be jump discontinuities in velocities. It is not clear how these jumps will affect the learning process. In the next section several different learning laws will be studied. One immediate issue to consider is the question of how to look back at the current phase in the previous cycle to pick which data point is most relevant to the current update. The standard RC looks back at the corresponding time step (modified by the usual one step time delay). But since the duration is different from one run to the next, perhaps it would be better if one looked at data corresponding to the same percentage of time in the cycle.

9 Numerical Investigation of Possible Learning Schemes

In this section we present four different approaches to the repetitive control of walking motion and discuss their effects on the example of the stiff-legged biped described in Sect. 4 with the simple feedback controller (30). Each law describes an algorithm to compute the inputs $y_{C,j}(kT)$ to the feedback controller at sample times k ($k = 1, ...N_j$) of cycle j, depending on the inputs of the previous cycle $y_{C,j-1}$, errors of the previous cycle etc. All these laws have in common that they rely on a synchronization of the phase with a significant event occurring once per cycle – in this case the touchdown of the swing foot, which is also very easy to detect in practice. This event is starting a new phase with relative time in the phase equal to zero. Therefore – even though the problem is closer to RC - we prefer to use a notation which is more of ILC type with the sampling time index k reset to 0 in every cycle. Note that this is purely by mathematical convenience and does not influence results.

All RC laws presented depend on one or more gains that can be tuned and typically have a large impact on the performance. We have investigated some sets of constants for each law (without doing a real optimization); and we present for each law the best constants we have found so far. No proof of convergence is given for these learning laws. Given the nonlinear nature of the problem with multiple phases and jump discontinuities it might be very difficult to establish such proofs. However, as noted in [8], good learning transients can be more important than stability (i.e convergence of the learning scheme). Furthermore [9] demonstrates that unstable learning laws can be very useful in applications.

In order to allow a comparison of the different laws, we display the following result plots:

- for each law, the error histories of angles θ_1 and θ_2 over time (shifted by the duration of the first cycle), see Figs. 10, 12, 14, and 16.
- for each law, a comparison of the outputs for angle θ_1, the corresponding reference trajectories, and the commanded trajectory at the beginning ($t = 0\,\text{s}...2\,\text{s}$) and at the end ($t = 18\,\text{s}...20\,\text{s}$) of the investigated learning process, see Figs. 11, 13, 15, and 17
- the RMS errors for each cycle of all four laws, in overview Fig. 18
- the development of cycle times α_j over a number of cycles for all four laws, in Fig. 19.

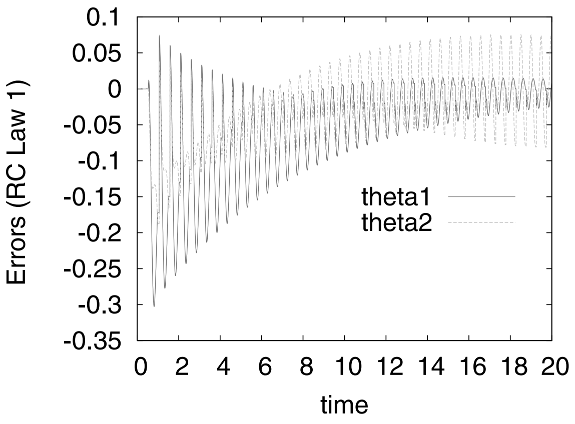


Fig. 10. Error histories of angles θ_1 and θ_2 using RC law 1

9.1 RC Law 1

$$y_{C,j}(kT) = y_{C,j-1}(kT) + \psi_1 e_{j-1}((k+1)T) \tag{32}$$

$$\text{with} \quad e_{j-1}(kT) = y_D(kT) - y_{j-1}(kT) \tag{33}$$

This first law is the most simple and straightforward one: The error term e_{j-1} that is used to correct the input of the system only compares the actual trajectory of the previous cycle and the reference trajectory at identical time points. The knowledge about the duration of the previous cycle is not used. The constant ψ_1 is chosen as 0.1.

As shown in Fig. 10, the law works quite well – despite its simplicity – to reduce the position variable errors (especially the RMS error of the relative hip angle θ_1 is reduced, while the error of the absolute stance leg angle θ_2 goes up a little again). However, this law does a poor job correcting the wrong initial cycle times (Fig. 19). Figure 11 gives some more details about the learning process, since it shows a comparison of the actual output angle θ_1, the desired angle $\theta_{1,D}$, and the commanded angle $\theta_{1,C}$ at 2 different intervals of the learning process. The upper picture shows the first few cycles: as in all other cases we start by commanding the desired trajectory, with the result that the output trajectory is far off. The lower picture shows cycles between 18 and 20 s (after roughly 36–40 cycles modified by law 1), with the desired and actual trajectory being quite close, and the commanded trajectory being quite different. These pictures also show the adjustment of phase times (the reference trajectory does a step change to zero after termination of the step, while the actual output step is still not finished) with a large error (18%) in the beginning, and a smaller, but still a significant difference (8%) at the end.

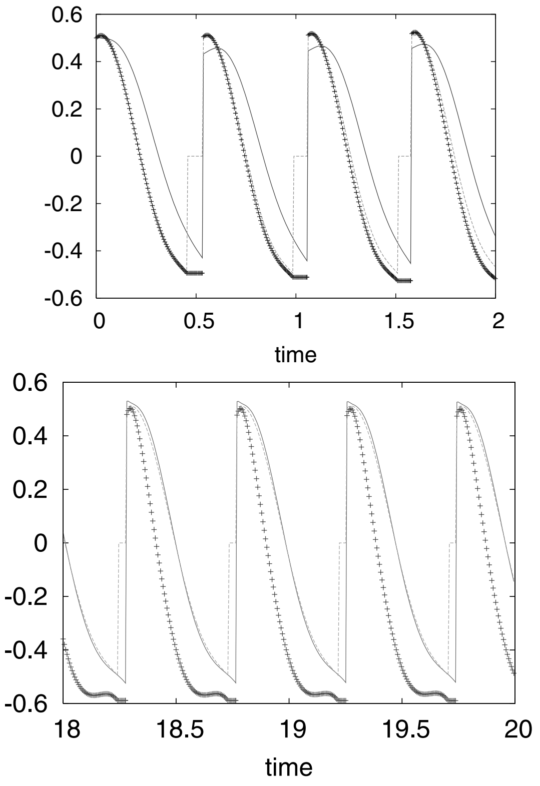


Fig. 11. Comparison of output trajectory (*solid line*), reference trajectory (*dashed lines*) and commanded trajectory (*crosses*) for angle θ_1 using RC law 1, at beginning (*top figure*) and end (*bottom figure*) of learning process

9.2 RC Law 2

$$y_{C,j}(kT) = c_1(\alpha_{j-1}) \cdot (y_{C,j-1}(\alpha_{j-1}kT) + \psi_1 e_{j-1}((k+1)T) \quad (34)$$

$$\text{with} \quad e_{j-1}(kT) = y_D(kT) - y_{j-1}(\alpha_{j-1}kT) \quad (35)$$

$$\alpha_{j-1} = \frac{T_{cycle,j-1}}{T_{cycle,ref}} \ . \quad (36)$$

In this law, the constant ψ_1 is again 0.1. This second law uses the factor α_{j-1} to introduce information about the previous cycle time and by computing the error between corresponding (and not identical) points of the current and reference cycle. For the evaluation of the right hand side of eqn. (35), linear interpolation is used between sample points. But it is important to note that while this error computation may be the more logical one, it does not punish

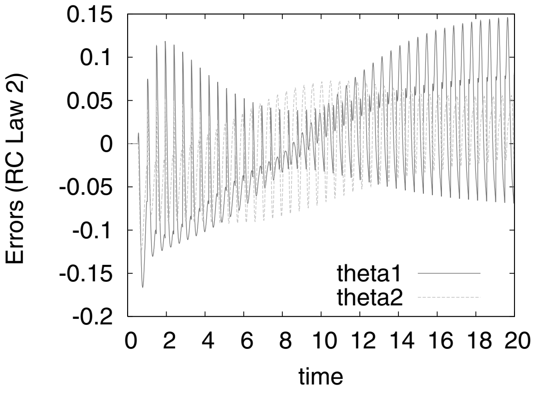


Fig. 12. Error histories of angles θ_1 and θ_2 using RC law 2

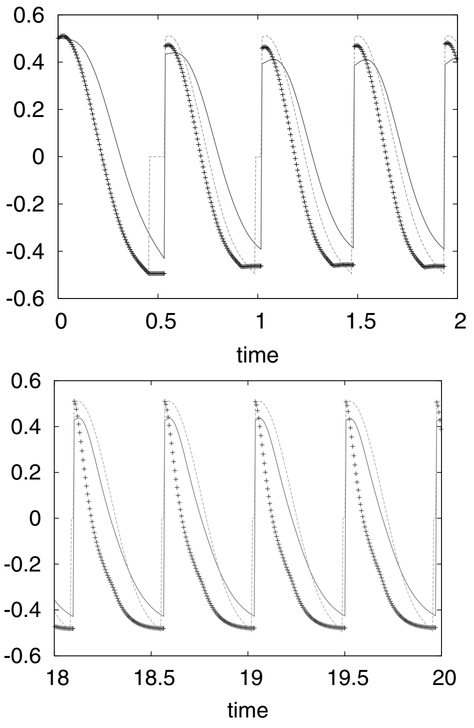


Fig. 13. Comparison of output trajectory (*solid line*), reference trajectory (*dashed lines*) and commanded trajectory (*crosses*) for angle θ_1 using RC law 2, at beginning (*top*) and end (*bottom*) of learning process

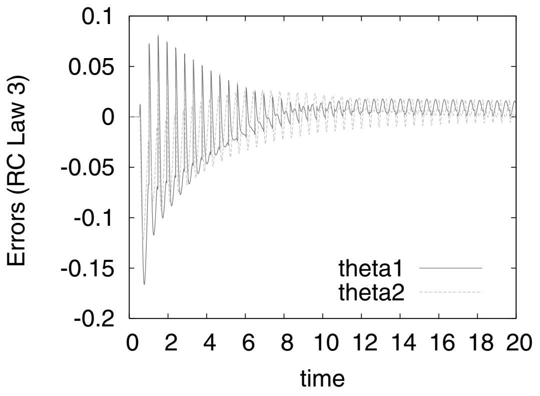


Fig. 14. Error histories of angles θ_1 and θ_2 using RC law 3

any errors of the cycle time; in fact a linearly scaled slower or faster cycle would actually lead to an error of zero. So it is important to introduce some correcting factor for wrong α_{j-1} which we do in the form of c_1, which is a function of α_{j-1} that has to satisfy $c_1(1) = 1$. Again, there are obviously many possible choices, and in this case we have set it to

$$c_1(\alpha_{j-1}) = \frac{1}{\sqrt{\alpha_{j-1}}} \, . \tag{37}$$

As shown in Fig. 19, this law does a better job than the first one in correcting the cycle duration. The reduction of position errors is roughly the same as for the first law with the inverse effect on the two degrees of freedom: this time errors of the stance leg angle are corrected better than errors of the relative hip angle (see Fig. 18). However, as in the case of law 1, there is no continuous reduction in one of the position errors, and the development beyond the investigated time frame is unclear. But we expect that it would be possible to improve the performance of this law with a tuned factor $c_1(\alpha)$.

There is however another possibility to improve the adjustment of cycle times (instead of using factor c_1) which is the inclusion of error terms on velocity level. The performance of this approach is investigated in the following two RC laws, numbers 3 and 4.

9.3 RC Law 3

$$\begin{aligned} y_{C,j}(kT) &= y_{C,j-1}(\alpha_{j-1}kT) + \psi_1 e_{pos,j-1}((k+1)T) \\ &\quad + \psi_2 e_{vel,j-1}((k+1)T) & (38) \\ \text{with} \quad e_{pos,j-1}(kT) &= e_{j-1}(kT) = y_D(kT) - y_{j-1}(\alpha_{j-1}kT) & (39) \\ e_{vel,j-1}(kT) &= \dot{y}_D(kT) - \dot{y}_{j-1}(\alpha_{j-1}kT)) & (40) \\ \text{and} \quad & \alpha_{j-1} \text{ as above.} & (41) \end{aligned}$$

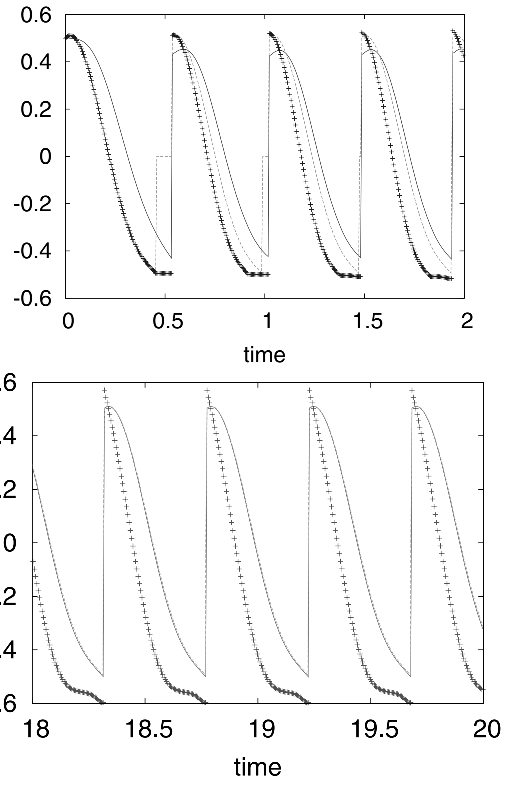


Fig. 15. Comparison of output trajectory (*solid line*), reference trajectory (*dashed lines*) and commanded trajectory (*crosses*) for angle θ_1 using RC law 3, at beginning (*top*) and end (*bottom*) of learning process

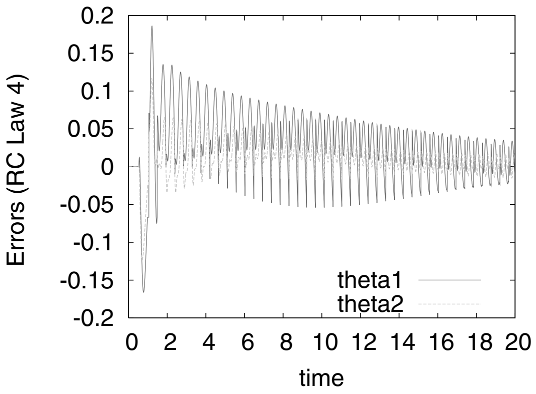


Fig. 16. Error histories of angles θ_1 and θ_2 using RC law 4

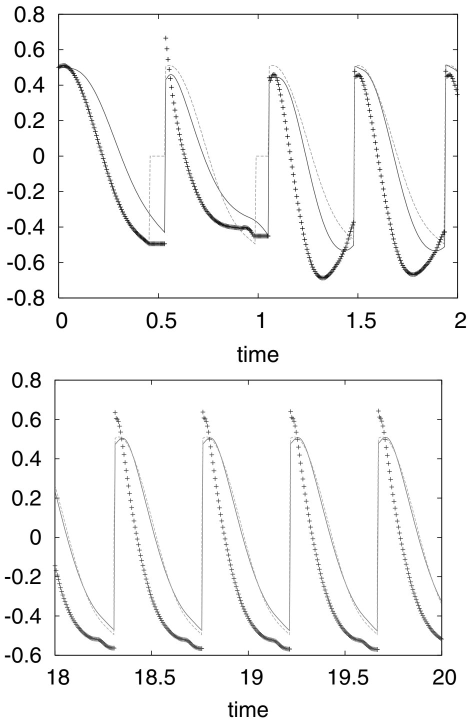


Fig. 17. Comparison of output trajectory (*solid line*), reference trajectory (*dashed lines*) and commanded trajectory (*crosses*) for angle θ_1 using RC law 4, at beginning (*top*) and end (*bottom*) of learning process

This law stems from the above RC law 2 skipping the leading factor c_1, but adding another correcting term which is proportional to the velocity errors. The constants are chosen as $\psi_1 = 0.1$, and $\psi_2 = 0.01$. As the Figs. 14, 15, 18 and 19 show, this law works extremely well both in correcting state errors and cycle duration. The cycle duration α_j is correct to 3 digits after only 3 cycles. The difference between the desired and actual output angle θ_1 is barely visible after 18 s (in the lower part of Fig. 15).

9.4 RC Law 4

$$\begin{aligned} y_{C,j}(kT) &= y_{C,j-1}(\alpha_{j-1}kT) + \psi_1 e_{pos,j-1}((k+1)T) \\ &\quad + \psi_2(1-\alpha_{j-1})^2 e_{vel,j-1}((k+1)T) \qquad (42) \\ \text{with} \quad & \alpha_{j-1}, e_{pos,j-1}, e_{pos,j-1} \text{ as above.} \qquad (43) \end{aligned}$$

RC law 4 is a modified version of law 3 with an additional factor in front of the velocity error term. The motivation behind this was to avoid asking too

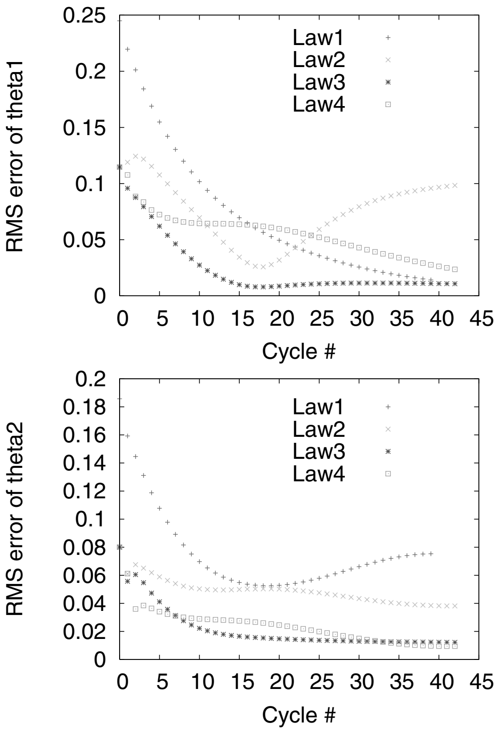


Fig. 18. RMS errors of θ_1 and θ_2 for RC laws 1–4

much from the RC controller, namely to correct errors in $2n$ state variables (the positions and velocities) while only modifying n input variables (the commanded position histories). The constants chosen in this case are $\psi_1 = 0.1$ and $\psi_2 = 8.0$. As Figs. 16, 17, 18 and 19 show, the performance of this law is also very good, comparable to that of law 3. In the particular case investigated here, law 4 seems to do slightly better on the absolute stance angle, and law 3 does better on the realtive hip angle corrections. While the start of the learning process according to law 4 (Fig. 17) clearly shows a different behavior than in the case of law 3, there is a clear resemblance of the commanded trajectories after 20s of the learning processes following laws 3 and 4.

9.5 Discussion of Simulation Results

Four different methods of RC for gaits have been studied here. One can either learn from the error in the previous cycle for the corresponding time step (as implemeted in law 1), or for the corresponding percent of the time used for that phase in that cycle (laws 2–4). The second approach is expected to significantly improve the size of deviations tolerated by the algorithm before the process goes unstable. However, since (without any other correcting terms)

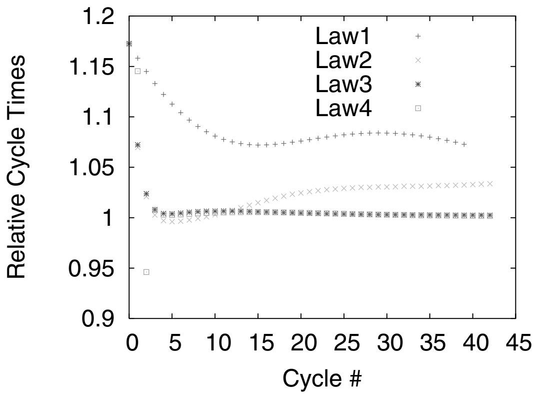


Fig. 19. Development of cycle times for RC laws 1–4

it does not penalize the error in the duration of the phase, one would like to introduce some extra aspect to the learning process to do so, e.g. introduce a factor depending on the relative cycle change (as in law 2).

The two most promising of the four RC approaches for gait problems that we investigated seemed to be the two laws that were based on both position and velocity errors in the previous cycle (laws 3 and 4). For the computation of errors and new commands the duration of the previous cycle (relative to the desired reference cycle time) was explicitely taken into account in both cases. In digital control one would not normally include both terms (position and velocity errors) because the number of input variables, i.e. the commands given for each time step, are not enough to independently control both the position and the velocity of the output at each time step. Hence in law 4 we included (in contrast to law 3) a cancellation of the velocity error term in the case of correct cycle time adjustment. After these first results for a specific walker and a specific feedback control law, it is hard to judge which of the two approaches might perform better in general. We think that both laws deserve further investigation on more test examples, also including more extensive studies of the most suitable choices of gains in the laws.

10 Conclusions

The concepts of ILC and RC aim at improving the performance of feedback control systems by adjusting the commands given to them for the execution of a repetitive or cyclic maneuver. The purpose of this paper is twofold: first, to discuss the general issues of transferring the ideas of ILC/RC to gait problems; and second, to present particular implementations in the form of four RC laws applied to a simple robot model with simple feedback control laws.

It has been shown that the problem of fixing errors in hardware execution of periodic gaits does not perfectly fit the problem formulations for either ILC or RC, but is closest to that of RC. The gait problem differs in that it must be separated into phases that start at foot strike, and that the durations of the phases can vary each cycle until reaching convergence to the desired gait.

Four different methods of addressing these extra issues have been presented. The results are summarized in Sect. 9.5 above. The two most promising RC approaches investigated included both an update on the command based on the error of the previous cycle, and a second update term based on the velocity error, where we also studied including a cancellation of the velocity term in the case of correct cycle time adjustment. Both laws deliver excellent results of adjusting cycle time and eliminating tracking errors. We intend to perform further investigations along these lines, involving other robot models and combining the concept of RC with other underlying feedback control systems used in contemporary walking robots.

We note that ILC and RC are notorious for exhibiting substantial decay in the error followed eventually by growth of the error, and much of the literature seeks ways to address this problem. No attempt has been made here to determine whether the RC laws result in asymptotical convergence to the desired solution when applied to the nonlinear robot dynamic equations. However, even if the laws are unstable, they may be very useful in practice. One simply uses the RC law to decrease the error, and turns it off when the error starts to grow – an approach that is used in the computer disk drive industry to good effect.

Our results suggest that with appropriate modifications it will be possible to use repetitive control concepts to significantly improve the execution of chosen periodic gaits by real walking robots.

Acknowledgments

The first author would like to express his thanks to the Alexander von Humboldt Foundation for a research award that partially supported the research presented here. The second author gratefully acknowledges financial support by the State of Baden-Württemberg through the Margarete von Wrangell program.

References

[1] *Proceedings of the IEEE-RAS International Conference on Humanoid Robots (Humanoids, 2005).* IEEE, Tsukuba, Japan, 2005.

[2] K. Berns. Walking machines catalogue. *http://www.fzi.de/ipt/WMC/walking-machines-katalog/walking-machines-katalog.html*, 2000.

[3] Z. Bien and J.-X. Xu, editors. *Iterative Learning Control: Analysis, Design, Integration, and Applications.* Kluwer Academic Publishers, Boston, 1998.

[4] H. Elci, R. W. Longman, M. Phan, J.-N. Juang, and R. Ugoletti. Discrete frequency based learning control for precision motion control. In *Proceedings of the 1994 IEEE International Conference on Systems, Man, and Cybernetics*, pp. 2767–2773, San Antonio, TX, Oct. 1994.

[5] Honda. Asimo. *http://asimo.honda.com*, 2006.

[6] Y. P. Hsin, R. W. Longman, E. J. Solcz, and J. de Jong. Experimental comparisons of four repetitive control algorithms. In *Proceedings of the 31st Annual Conference on Information Sciences and Systems*, pp. 854–860, Johns Hopkins University, Department of Electrical and Computer Engineering, Baltimore, Maryland, 1997.

[7] Y. P. Hsin, R. W. Longman, E. J. Solcz, and J. de Jong. Experiments bridging learning and repetitive control. *Advances in the Astronautical Sciences*, 95:671–690, 1997.

[8] R. W. Longman. Iterative learning control and repetitive control for engineering practice. *International Journal of Control*, 73(10):930–954, 2000.

[9] R. W. Longman and Y.-C. Huang. Use of unstable repetitive control for improved tracking accuracy. *Adaptive Structures and Composite Materials: Analysis and Applications, ASME*, AD-45, MD-54:315–324, Nov. 1994.

[10] R. W. Longman and Y.-C. Huang. The phenomenon of apparent convergence followed by divergence in learning and repetitive control. *Intelligent Automation and Soft Computing, Special Issue on Learning and Repetitive Control*, 8(2):107–128, 2002.

[11] R. W. Longman, Y.-T. Peng, T. Kwon, H. Lus, R. Betti, and J.-N. Juang. Adaptive inverse iterative learning control. *Advances in the Astronautical Sciences*, 114:115–134, 2003.

[12] T. McGeer. Passive dynamic walking. *International Journal of Robotics Research*, 9:62–82, 1990.

[13] K. D. Mombaur. Performing open-loop stable flip-flops – an example for stability optimization and robustness analysis of fast periodic motions. In *Fast Motions in Robotics and Biomechanics – Optimization and Feedback Control*, Lecture Notes in Control and Information Science. Springer, 2006.

[14] K. D. Mombaur, H. G. Bock, J. P. Schlöder, and R. W. Longman. Human-like actuated walking that is asymptotically stable without feedback. In *Proceedings of IEEE International Conference on Robotics and Automation*, pp. 4128–4133, Seoul, Korea, May 2001.

[15] K. D. Mombaur, H. G. Bock, J. P. Schlöder, and R. W. Longman. Open-loop stability – a new paradigm for periodic optimal control and analysis of walking mechanisms. In *Proceedings of IEEE CIS-RAM 04, Singapore*, 2004.

[16] K. D. Mombaur, H. G. Bock, J. P. Schlöder, and R. W. Longman. Self-stabilizing somersaults. *IEEE Transactions on Robotics*, 21(6), Dec. 2005.

[17] K. D. Mombaur, R. W. Longman, H. G. Bock, and J. P. Schlöder. Open-loop stable running. *Robotica*, 23(01):21–33, January 2005.

[18] K. D. Mombaur, R. W. Longman, J. P. Schlöder, and H. G. Bock. Optimizing spring-damper design in human-like walking that is asymptotically stable without feedback. 2006, submitted.

[19] K. Moore and J.-X. Xu. Special issue on iterative learning control. *International Journal of Control*, 73(10), July 2000.

[20] S. J. Oh, R. W. Longman, and Y.-P. Hsin. The possible block diagram configurations for repetitive control to cancel periodic plant or measurement dis-

turbances. In *Proceedings of the 2004 AIAA/AAS Astrodynamics Specialist Conference*, Providence, RI, Aug. 2004.

[21] B. Panomruttanarug and R. W. Longman. Frequency based optimal design of FIR zero-phase filters and compensators for robust repetitive control. *Advances in the Astronautical Sciences*, 123, to appear.

[22] B. Panomruttanarug and R. W. Longman. Repetitive controller design using optimization in the frequency domain. In *Proceedings of the 2004 AIAA/AAS Astrodynamics Specialist Conference*, Providence, RI, Aug. 2004.

[23] N. Phetkong, M. S. Chew, and R. W. Longman. Morphing mechanisms part 1: Using iterative learning control to morph cam follower motion. *American Journal of Applied Sciences*, 2(5):897–903, 2005.

[24] A. M. Plotnik and R. W. Longman. Subtleties in the use of zero-phase low-pass filtering and cliff filtering in learning control. *Advances in the Astronautical Sciences*, 103:673–692, 1999.

[25] Sony. Qrio – Sony Dream Robot. *http://www.sony.net/SonyInfo/QRIO/*, 2006.