JOURNAL OF APPLIED BIOMECHANICS, 2004, **20**, 421-449

Forward Dynamics of Two-Dimensional Skeletal Models. A Newton-Euler Approach

L.J. Richard Casius, Maarten F. Bobbert, and Arthur J. van Soest
Vrije Universiteit, Amsterdam

Mathematical modeling and computer simulation play an increasingly important role in the search for answers to questions that cannot be addressed experimentally. One of the biggest challenges in forward simulation of the movements of the musculoskeletal system is finding an optimal control strategy. It is not uncommon for this type of optimization problem that the segment dynamics need to be calculated millions of times. In addition, these calculations typically consume a large part of the CPU time during forward movement simulations. As numerous human movements are two-dimensional (2-D) to a reasonable approximation, it is extremely convenient to have a dedicated, computational efficient method for 2-D movements. In this paper we shall present such a method. The main goal is to show that a systematic approach can be adopted which allows for both automatic formulation and solution of the equations of kinematics and dynamics, and to provide some fundamental insight in the mechanical theory behind forward dynamics problems in general. To illustrate matters, we provide for download an example implementation of the main segment dynamics algorithm, as well as a complete implementation of a model of human sprint cycling.

Key Words: human, segments, degrees of freedom, modeling, equations of motion, motion simulation, kinematic constraints

The ability of human beings to control their movements to meet specific goals is truly remarkable. In controlling movements, two systems are involved: the nervous system and the musculoskeletal system. The nervous system sends electrical signals to the muscles referred to as muscle stimulation, whereupon muscles produce forces that actuate the skeleton. Generally the resulting movement produces feedback, allowing the nervous system to adapt the muscle stimulation. Figure 1 depicts the course of action schematically.

For some movements it is straightforward to define an objective performance criterion, such as the maximum height reached in vertical jumping or the highest

The authors are with the Faculty of Human Movement Sciences, Vrije Universiteit, Van der Boechorststraat 9, 1081 BT Amsterdam, The Netherlands.

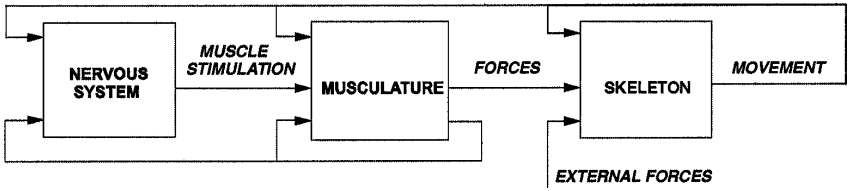

Figure 1 — Interaction between the nervous system and the musculoskeletal system.

power transferred to the crank in sprint cycling. Strictly speaking, the intrinsic properties of the musculoskeletal system determine the theoretically maximum achievement. Actual achievement, however, depends largely on the control of the musculature (Bobbert & van Soest, 1994). Regardless of the strength of the muscles, without optimal coordination the system does not realize its potential performance. To study the relation between properties of the musculoskeletal system and the maximum achievable performance, as well as coordination of the musculature, we cannot rely solely on experiments with humans. Many variables of interest simply cannot be observed or manipulated experimentally. Consider for example the following questions:

- What happens to the maximum jump height if the strength of a particular muscle increases by 10%, and how should muscle stimulation be adapted to produce maximum height?
- How should muscle stimulation be adapted to obtain maximum power output when the height of a bicycle saddle is altered?
- What would happen to a cyclist's maximum power output if the origin and insertion of a particular muscle were situated differently?

Obviously such questions are difficult if not impossible to address experimentally, either for ethical or practical reasons. Therefore, mathematical modeling and computer simulation play an important role in the search for answers to these questions. In forward simulation of the movements of the musculoskeletal system, muscle stimulation as a function of time is the input, and the resulting movement is the ultimate output. During a simulation, the researcher has full control of the input and the parameters of the model, and full access to all variables. Given a set of parameters, an optimal movement might be found by systematically trying different muscle stimulation patterns. This normally requires sophisticated optimization techniques such as simulated annealing (Goffe et al., 1994) or genetic algorithms (Soest & Casius, 2003). The modeling and simulation approach has successfully been applied in different studies over the last decades. For example, several authors have published on vertical or long jumping (Bobbert & van Soest, 1994; Hatze, 1981a; Pandy et al., 1990) and sprint cycling (Neptune, 1999; Soest & Casius, 2000). Due to the progress made in the field and the ever-increasing computer power, its applicability is recognized by a steadily growing audience.

In this paper we shall focus on the skeletal system. (For those interested in modeling the musculature, see Barret et al., 2002; Hatze, 1981b; Soest & Bobbert, 1993; Zajac & Gordon, 1989; Zajac & Winters, 1990). In the next section, the equations of motion for a chain model of the skeleton are derived using the Newton-Euler approach. To solve the equations on a computer, the section on Generic

Algorithm turns them into an equivalent system $A \cdot x = b$, and provides a generic algorithm for automatically doing so. Although this paper focuses on forward dynamics, the algorithm is perfectly suitable for inverse dynamics or any mix of forward and inverse dynamics. To perform a simulation, the section on Simulating a Movement briefly introduces the differential equations that govern the motion of the skeleton needed for numerical integration. This is followed with a section on Kinematic Constraints which shows how kinematic acceleration constraints can be added to the equations of motion to impose restrictions on the movement. The final section is a comprehensive example concerning sprint cycling.

To illustrate the basic principles, example code is provided on an accompanying web site (http://www.ifkb.nl/downloads/casius), written in Matlab (The MathWorks, Inc.; http://www.mathworks.com). Owing to the compact matrix manipulations, a rather complete example could be provided. The code has been written with readability rather than efficiency as the main purpose, so for a basic understanding one does not need prior knowledge of Matlab.

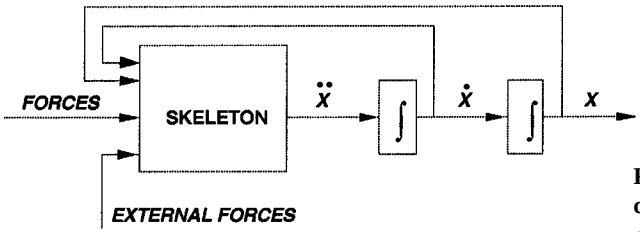

Figure 2 — Flow of control in forward dynamics.

Newtonian Equations of Motion for a 2-D Chain Model

Isolating the skeletal system from Figure 1 leaves us with the system depicted in Figure 2. The skeleton is actuated by forces generated by muscles, and by external forces such as gravity. Given the current position and velocity of the skeleton, the acceleration with respect to an inertial frame of reference, i.e., a frame of reference fixed to earth, can be calculated. This is generally referred to as *forward* dynamics, or also *direct* dynamics. As we shall not discuss muscle dynamics in this paper, the mechanical effect of muscle forces is represented by net joint moments. In systems that do incorporate muscle dynamics, the calculation of net joint moments from muscle forces is just an additional step taken prior to calculating the skeletal dynamics. Thus, replacing muscle forces by net joint moments is not detrimental to the discussion of skeletal dynamics.

Compared to *inverse* dynamics, that is, calculating net joint moments from (experimental) kinematical data, forward dynamics is much more complicated because, as a rule, it involves solving a system of coupled differential equations. The main goal of this paper is to show that a systematic approach can be adopted which allows for both automatic formulation and solution of the equations of kinematics and dynamics.

Actually, such an approach underlies several commercially available software packages such as ADAMS (Automated Dynamical Analysis of Mechanical Systems, Mechanical Dynamics Inc., Ann Arbor, MI), DADS (Dynamic Analysis

and Design System, Computer Aided Design Software Inc., Oakdale, IA), and SD/FAST (PTC, Needham, MA). Notwithstanding the potential use in mechanical engineering, commercial packages often lack the efficiency that can be obtained by writing problem-specific software and do not always offer the flexibility desired by biomechanical researchers (Bogert, 1990). Therefore several packages have been developed in the academic world to meet specific needs, e.g., SPACAR (Werff, 1977) and MUSK (Casius, 1995). Even if commercial packages are employed for biomechanical problems, the researcher should have good insight into the theory underlying such computer programs.

Understanding equations of motion and solving differential equations via numerical integration are of vital importance to the development of human models and interpretation of the simulation results. In this paper we shall present a method for solving forward dynamics problems that is particularly efficient for 2-D chain models of the skeleton, which has successfully been applied in a variety of studies such as vertical jumping (Soest et al., 1994), sprint cycling (Soest & Casius, 2000), rowing (Soest & Smith, 2001), and postural control (Soest et al., 2003). Although some aspects of other methods will be briefly mentioned, for a more elaborate overview we refer to Haug (1989) and Schielen (1990).

Chain Model of the Skeleton

For simplicity, a skeletal model will be restricted to a 2-D chain of rigid segments. The rigidity assumption is customary in simulation studies, as it is believed that the movement of parts within a body segment is negligible compared to intersegmental motion. In passing, there is evidence that representing human segments as rigid bodies is not appropriate during impact (Dickenson et al., 1985; Gruber et al., 1987; 1998; Liu & Nigg, 2000). The assumption of two-dimensionality is clearly more restrictive. However, the examples in the previous section show that simple, well-defined movements in 2-D space are perfectly suitable for studying many aspects about the properties and coordination of the musculoskeletal system, simply because numerous human movements are 2-D to a reasonable approximation (e.g., jumping, lifting, rowing, postural control, cycling). Thus it is convenient to have a dedicated, computational efficient method for 2-D movements. Especially when simulations involve optimization of, say, an optimal control strategy, computational efficiency is extremely important (Soest & Casius, 2000, 2003).

Although the 2-D method presented in this paper can theoretically be generalized to 3-D analyses, the complexity should not be underestimated, and for that matter Kane's method may be a better choice (Yamaguchi, 2001). Nonetheless, the mechanical fundamentals of 2-D methods presented here provide insight into the theory underlying 3-D methods as well. The final assumption concerns the way the segments are attached to make up a chain. All segments are connected in hinge joints. This implies that translational motion within a joint is neglected.

Figure 3 shows a side view of a person performing a vertical squat jump. It also outlines a chain model of the skeleton. The segments are numbered from 1 to n, the joints from 1 to n+1, where n equals the number of segments. The first segment represents the feet, the second the lower legs, the third the upper legs, and the fourth the head, arms, and trunk (HAT). The circles represent the joints. The end joints—*virtual* joints—are used to connect the ends of the chain to the environment. In this example the first joint connects the feet to the floor. In this context

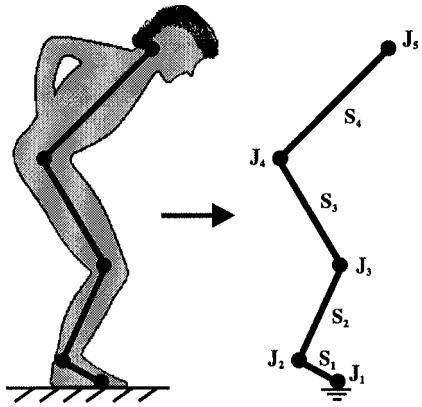

Figure 3 — Four-segment chain model of a human performing a vertical jump squat (arms akimbo).

we say the joint axis is *fixed*. The next three joints represent the ankle, knee, and hip joint. Since the end of the chain is airborne, the final joint is not shown in this example, but we still refer to it as a joint, which in this case is free. Although the first joint axis is fixed in this example, this is not a necessity. Both ends of the chain can be free, making the whole system airborne. Likewise, both ends of the chain may be attached to the environment. Intermediate joint axes can also be fixed, although from a computational point of view it may be more efficient to split the chain into two parts that happen to be attached to the same point. The implementation of the skeletal dynamics described below allows joint axes to change from fixed to free during a simulation. As a rule, the opposite is not the case, as this would require the implementation of impact dynamics (Gerritsen et al., 1995).

As a more complex example of a skeletal model, Figure 4 shows a cyclist. The hip joint axis is fixed, representing that the cyclist is attached to the saddle. At the hip joint, the chain branches off: one branch constitutes the second leg, the other the HAT segment. Although branches are not formally discussed, in the section on Branches we shall present a simple trick to implement them within the framework of the chain model. Note that a segment does not necessarily represent a human part. In Figure 4a the first and last segments represent a crank. One end of

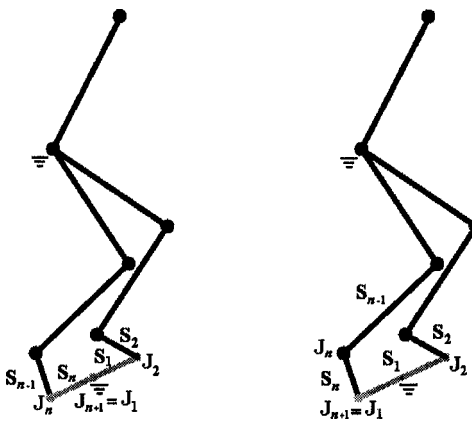

Figure 4 — Two skeletal models of a cyclist. Left: Two cranks are modeled that are fixed at the same point. Right: Only one crank is modeled, which is fixed at the rotational axis.

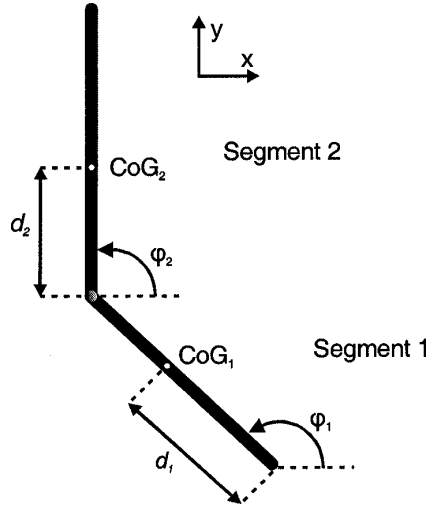

Figure 5 — CoG_1 and CoG_2 are the centers of gravity. In this paper, *proximal* refers to particles more ahead in the chain, while *distal* refers to particles more behind. Thus, d_1 is the distance from the proximal end of Segment 1 to CoG_1, while $\ell_1 - d_1$ is the distance from CoG_1 to its distal end.

each crank is fixed at exactly the same position; i.e., the first and last joint in the chain coincide. Later we shall see that provisions must be made to ensure that both cranks rotate at equal speeds around this point. Figure 4b shows a more elegant solution, in which only one crank is modeled. Both feet are connected to this crank. The center of gravity, which coincides with the rotational axis, is fixed to the environment.

Skeletal Parameters, Position, and Degrees of Freedom

Assuming that the center of gravity of a segment is located on the line between its adjacent joints, the intrinsic properties of an individual segment can be described by four parameters: mass (m), length (ℓ), distance from *proximal* end to the center of gravity (d) (see Figure 5), and moment of inertia relative to center of gravity (j). The orientation of a segment is given by its segment angle (φ), which is defined as the angle between the positive x-axis and the line from proximal to distal joint. Figure 5 shows that counterclockwise angles are taken to be positive. Given all segment angles, the position of any point in the chain relative to any other can be calculated. In Figure 5 for example, if (x_1, y_1) is the position of CoG_1 and (x_2, y_2) the position CoG_2, then:

$$x_2 = x_1 + (\ell_1 - d_1) \cdot \cos(\varphi_1) + d_2 \cdot \cos(\varphi_2)$$
$$y_2 = y_1 + (\ell_1 - d_1) \cdot \sin(\varphi_1) + d_2 \cdot \sin(\varphi_2)$$

Consequently, if the position of an arbitrary point with respect to the environment is known in addition to the segment angles, the position and orientation of the body is fully established.

The minimum number of coordinates needed to specify the position and orientation of all segments is called the number of degrees of freedom (*DF*) of the mechanical linkage. For an n-segment chain model, *DF* is at most $n+2$: n angular coordinates, and two Cartesian coordinates (an x- and y-coordinate) are sufficient to establish the orientation and position of the whole body. In many cases *DF* is

even lower due to (kinematic) constraints imposed on the system. In Figure 3 for example, the first segment is connected to the floor, so the *x*- and *y*-position of the first joint are no longer free. Consequently, two *DF* are lost, and the four segment angles alone make up a minimal set of coordinates that specify the position and orientation of the whole body. However, instead of the first segment angle, we could also take the *x*-coordinate of the second joint, or the *y*-coordinate, or one of the coordinates of the center of gravity, etc.

Any minimal set that specifies the position and orientation of the whole body is called a set of *generalized coordinates* (i.e., any mixture of angular and Cartesian coordinates is allowed) (Zajac & Winters, 1990; Zatsiorsky, 1998). The notion of *DF* is important in the field of motion simulations, as it equals the minimum number of equations of motion needed to model the motor task in question. Several methods for deriving a system of equations of motion are based on the concept of generalized coordinates, such as the commonly used Lagrangian methods and Kane's method (Kane & Levinson, 1985; Yamaguchi, 2001). In the next section we shall present a different approach, generally referred to as the Newton-Euler method.

Equations of Motion

The key step of the Newton-Euler method is to break down the system into individual segments. For each segment, a free body diagram is constructed showing all forces and moments acting on it. From the previous section we know that the *DF* of an *n*-segment chain in 2-D space equals *n*+2. Thus, for an isolated segment, *DF* = 1+2 = 3. Consequently we need three equations of motion for each segment. Regarding the linear accelerations of a segment, equations are derived using Newton's second law; for the angular acceleration, an Euler equation is used. If $\ddot{x}_c$ and $\ddot{y}_c$ denote the linear accelerations of the center of gravity, and $\ddot{\varphi}$ the segment angular acceleration, we get:

$$\sum F_x = m \cdot \ddot{x}_c$$
$$\sum F_y = m \cdot \ddot{y}_c$$
$$\sum M = j \cdot \ddot{\varphi}$$

The first two equations hold for any free body diagram, whereas the third only applies to rigid (i.e., nondeformable) bodies. All moments and forces that may act on a segment can conveniently be represented by the three forces and the three moments shown in Figure 6. First, reaction forces from adjacent segments (or the environment) act on both ends of the segment. Second, a net joint moment acts about both ends of the segment. In musculoskeletal models, a net joint moment represents the sum of moments of all forces produced by the muscles and passive structures that span the joint, calculated relative to the joint rotational axis. Finally, all external forces (e.g., gravity) can be represented by a net external force that applies at the center of gravity, together with a net external moment that represents the actual point of application of this force.

When two segments are assembled, they exert a force on each other. Suppose that Segments 1 and 2 are connected in a hinge joint (Figure 7). Let $F_{1,2}$ be the force exerted by Segment 1 on Segment 2, and $F_{2,1}$ the force exerted by Segment 2 on Segment 1. We do not need to represent both forces. After all, from Newton's third law of action and reaction, it follows that: $F_{2,1} = -F_{1,2}$. Therefore we only

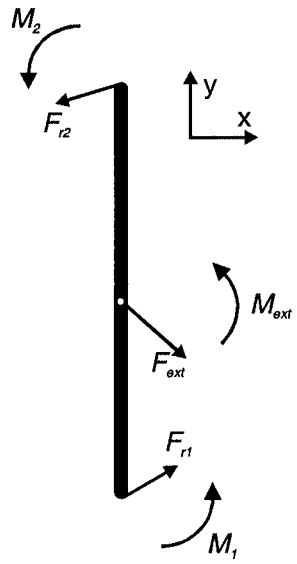

Figure 6 — Forces and moments acting on a segment. F_{r1} and F_{r2} denote joint reaction forces, and M_1 and M_2 the net moments about the joints. F_{ext} represents the external forces that apply at the center of gravity, and M_{ext} the external moment that acts from the environment.

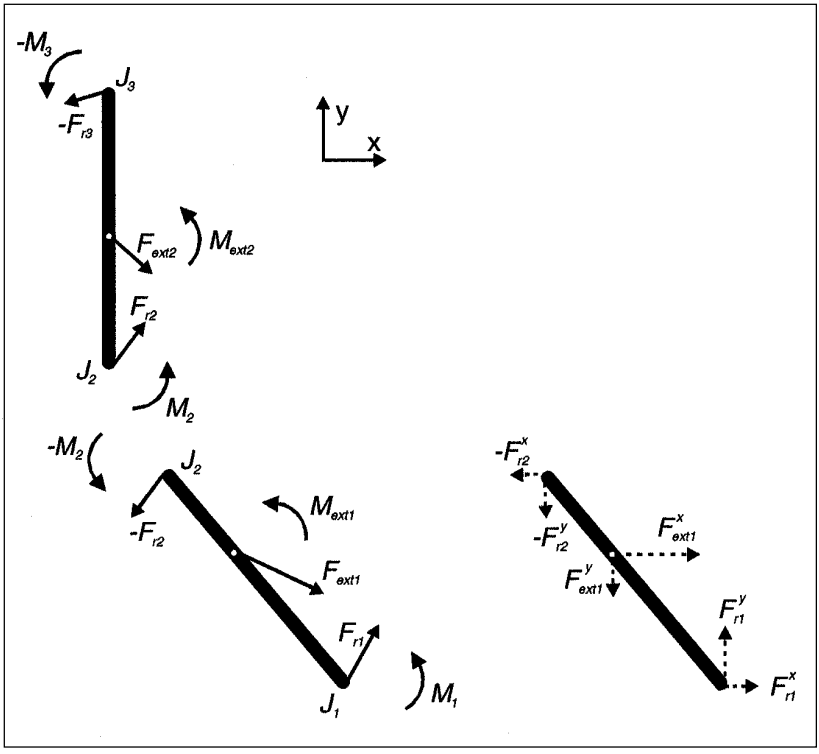

Figure 7 — Left: Two segments connected in a hinge joint, where action = – reaction. Note that the joints connect in J_2 but have been drawn separately for readability. Right: Horizontal and vertical components of the forces acting on the first segment.

define a single joint reaction force at each of the n+1 joints in our n-segment chain. Throughout the text, F_{ri} will be used to denote the joint reaction force at joint i, exerted by segment i–1 on segment i. For the most proximal joint, the reaction force is exerted by the environment, and for the most distal joint the force is exerted on the environment. For the net joint moments a similar argument holds. In Figure 7, the net joint moment that acts about the distal end of Segment 1 must equal the net joint moment that acts about the proximal end of Segment 2, albeit they work oppositely. Consequently we need to deal with only n+1 net joint moments. We shall use M_i to denote the net moment that acts about the proximal end of segment i (see Figure 7).

To write down the equations of motion in scalar form, all forces that act on a segment are decomposed in a horizontal and vertical component, as was done for the first segment in Figure 7. Concerning the signs, forces directed along the positive x-axis and y-axis are taken to be positive; moments are taken to be positive if they cause a counterclockwise acceleration. To specify the position of the system with respect to the environment, we shall use the x- and y-coordinate of the first joint in the chain. In this context the first joint will be referred to as the *base* of the chain, and its position will be denoted by (x, y).

Given Figure 7, we will formulate the equations of motion for a two-segment body. Since we have three equations of motion for each segment, we cannot have more than three unknowns. With regard to forward dynamics, in addition to the horizontal and vertical reaction force at the proximal joint, the angular acceleration of the segment acts as the third unknown. One problem concerns the linear accelerations in the first two equations. At first they also seem to be unknown. However, they can be expressed in terms of the segment angles, segment angular velocities, segment angular accelerations, and the linear accelerations of the base. Using the chain rule and product rule of differentiation, the horizontal accelerations can be expressed as:

$$
\begin{aligned}
x_1 &= x + d_1 \cdot \cos\varphi_1 \\
\dot{x}_1 &= \dot{x} + -d_1 \cdot \sin\varphi_1 \cdot \dot{\varphi}_1 \\
\ddot{x}_1 &= \ddot{x} + -d_1 \cdot \cos\varphi_1 \cdot \dot{\varphi}_1^2 + -d_1 \cdot \sin\varphi_1 \cdot \ddot{\varphi}_1 \\
x_2 &= x + \ell_1 \cdot \cos\varphi_1 + d_2 \cdot \cos\varphi_2 \\
\dot{x}_2 &= \dot{x} + -\ell_1 \cdot \sin\varphi_1 \cdot \dot{\varphi}_1 + -d_2 \cdot \sin\varphi_2 \cdot \dot{\varphi}_2 \\
\ddot{x}_2 &= \ddot{x} + -\ell_1 \cdot \cos\varphi_1 \cdot \dot{\varphi}_1^2 + -\ell_1 \cdot \sin\varphi_1 \cdot \ddot{\varphi}_1 + -d_2 \cdot \cos\varphi_2 \cdot \dot{\varphi}_2^2 + \\
&\qquad -d_2 \cdot \sin\varphi_2 \cdot \ddot{\varphi}_2
\end{aligned}
$$

Similarly, the vertical accelerations can be expressed as:

$$
\begin{aligned}
\ddot{y}_1 &= \ddot{y} + -d_1 \cdot \sin\varphi_1 \cdot \dot{\varphi}_1^2 + d_1 \cdot \cos\varphi_1 \cdot \ddot{\varphi}_1 \\
\ddot{y}_2 &= \ddot{y} + -\ell_1 \cdot \sin\varphi_1 \cdot \dot{\varphi}_1^2 + \ell_1 \cdot \cos\varphi_1 \cdot \ddot{\varphi}_1 + -d_2 \cdot \sin\varphi_2 \cdot \dot{\varphi}_2^2 + \\
&\qquad d_2 \cdot \cos\varphi_2 \cdot \ddot{\varphi}_2
\end{aligned}
$$

For the moment we shall assume that the base is fixed, so both $\ddot{x}$ and $\ddot{y}$ are zero. As noted, the segment angular accelerations $\ddot{\varphi}_1$ and $\ddot{\varphi}_2$ are unknown during forward dynamics. This implies that segment angles φ_1 and φ_2 and segment angular velocities $\dot{\varphi}_1$ and $\dot{\varphi}_2$ have to be known or we would still end up with more than three unknowns per segment. As will become apparent, they constitute the vector of

state variables. Initial values for the state variables must be specified before (numerical) integration can take place, and thereafter the integration routine gradually updates these variables until some desired state or time has been reached. In terms of the integration routine, the vector of state variables is the *integration constant*. Now that we know how the linear accelerations can be expressed in the unknowns $\ddot{\varphi}_1$ and $\ddot{\varphi}_2$, we can write the equations of motion. For the first segment they read:

$$\begin{aligned}
F_{r1}^x &+ -F_{r2}^x + F_{ext1}^x = m_1 \cdot \ddot{x}_1 \\
F_{r1}^y &+ -F_{r2}^y + F_{ext1}^y = m_1 \cdot \ddot{y}_1 \\
d_1 \cdot \sin\varphi_1 \cdot F_{r1}^x &+ -p_1 \cdot \sin\varphi_1 \cdot -F_{r2}^x + -d_1 \cdot \cos\varphi_1 \cdot F_{r1}^y \\
&+ p_1 \cdot \cos\varphi_1 \cdot -F_{r2}^y + M_{ext1} + M_1 + -M_2 = j_1 \cdot \ddot{\varphi}_1
\end{aligned}$$

where $p_i = \ell_i - d_i$, and $\ddot{x}_1$ and $\ddot{y}_1$ are as derived above. Similarly, for the second segment we get:

$$\begin{aligned}
F_{r2}^x &+ -F_{r3}^x + F_{ext2}^x = m_2 \cdot \ddot{x}_2 \\
F_{r2}^y &+ -F_{r3}^y + F_{ext2}^y = m_2 \cdot \ddot{y}_2 \\
d_2 \cdot \sin\varphi_2 \cdot F_{r2}^x &+ -p_2 \cdot \sin\varphi_2 \cdot -F_{r3}^x + -d_2 \cdot \cos\varphi_2 \cdot F_{r2}^y = \\
&+ p_2 \cdot \cos\varphi_2 \cdot -F_{r3}^y + M_{ext2} + M_2 + -M_3 = j_2 \cdot \ddot{\varphi}_2
\end{aligned}$$

If we substitute the expressions for the horizontal accelerations in the corresponding two equations, and move the unknown terms to the left and the known terms to the right, we arrive at:

$$\begin{aligned}
F_{r1}^x - F_{r2}^x + m_1 \cdot d_1 \cdot \sin\varphi_1 \cdot \ddot{\varphi}_1 &= -F_{ext1}^x + m_1 \cdot \ddot{x} - m_1 \cdot d_1 \cdot \cos\varphi_1 \cdot \dot{\varphi}_1^2 \\
F_{r2}^x - F_{r3}^x + m_2 \cdot \ell_1 \cdot \sin\varphi_1 \cdot \ddot{\varphi}_1 + m_2 \cdot d_2 \cdot \sin\varphi_2 \cdot \ddot{\varphi}_2 &= -F_{ext2}^x + m_2 \cdot \ddot{x} - m_2 \cdot \ell_1 \cdot \cos\varphi_1 \cdot \dot{\varphi}_1^2 - m_2 \cdot d_2 \cdot \cos\varphi_2 \cdot \dot{\varphi}_2^2
\end{aligned}$$

Likewise, the equations concerning the vertical accelerations are found:

$$\begin{aligned}
F_{r1}^y - F_{r2}^y - m_1 \cdot d_1 \cdot \cos\varphi_1 \cdot \ddot{\varphi}_1 &= -F_{ext1}^y + m_1 \cdot \ddot{y} - m_1 \cdot d_1 \cdot \sin\varphi_1 \cdot \dot{\varphi}_1^2 \\
F_{r2}^y - F_{r3}^y - m_2 \cdot \ell_1 \cdot \cos\varphi_1 \cdot \ddot{\varphi}_1 - m_2 \cdot d_2 \cdot \cos\varphi_2 \cdot \ddot{\varphi}_2 &= -F_{ext2}^y + m_2 \cdot \ddot{y} - m_2 \cdot \ell_1 \cdot \sin\varphi_1 \cdot \dot{\varphi}_1^2 - m_2 \cdot d_2 \cdot \sin\varphi_2 \cdot \dot{\varphi}_2^2
\end{aligned}$$

Finally, after reshuffling the known and unknown terms, the rotational equations look like:

$$\begin{aligned}
d_1 \cdot \sin\varphi_1 \cdot F_{r1}^x + p_1 \sin\varphi_1 \cdot F_{r2}^x - d_1 \cos\varphi_1 \cdot F_{r1}^y - p_1 \cos\varphi_1 \cdot F_{r2}^y - j_1 \cdot \ddot{\varphi}_1 &= -M_{ext1} - M_1 + M_2 \\
d_2 \cdot \sin\varphi_2 \cdot F_{r2}^x + p_2 \sin\varphi_2 \cdot F_{r3}^x - d_2 \cos\varphi_2 \cdot F_{r2}^y - p_2 \cos\varphi_2 \cdot F_{r3}^y - j_2 \cdot \ddot{\varphi}_2 &= -M_{ext2} - M_2 + M_3
\end{aligned}$$

Clearly a pattern shows in these equations. We do not need to puzzle over a potential third segment in the chain; the equations of motion for an extra segment can be written readily:

$$
\begin{aligned}
&F_{r3}^{x} - F_{r4}^{x} + m_3 \cdot \ell_1 \cdot \sin\varphi_1 \cdot \ddot{\varphi}_1 + m_3 \cdot \ell_2 \cdot \sin\varphi_2 \cdot \ddot{\varphi}_2 + m_3 \cdot d_3 \cdot \sin\varphi_3 \cdot \ddot{\varphi}_3 \\
&\quad = -F_{ext3}^{x} + m_3 \cdot \ddot{x} - m_3 \cdot \ell_1 \cdot \cos\varphi_1 \cdot \dot{\varphi}_1^2 - m_3 \cdot \ell_2 \cdot \cos\varphi_2 \cdot \dot{\varphi}_2^2 - m_3 \cdot d_3 \cdot \cos\varphi_3 \cdot \dot{\varphi}_3^2 \\
&F_{r3}^{y} - F_{r4}^{y} - m_3 \cdot \ell_1 \cdot \cos\varphi_1 \cdot \ddot{\varphi}_1 - m_3 \cdot \ell_2 \cdot \cos\varphi_2 \cdot \ddot{\varphi}_2 - m_3 \cdot d_3 \cdot \cos\varphi_3 \cdot \ddot{\varphi}_3 \\
&\quad = -F_{ext3}^{y} + m_3 \cdot \ddot{y} - m_3 \cdot \ell_1 \cdot \sin\varphi_1 \cdot \dot{\varphi}_1^2 - m_3 \cdot \ell_2 \cdot \sin\varphi_2 \cdot \dot{\varphi}_2^2 - m_3 \cdot d_3 \cdot \sin\varphi_3 \cdot \dot{\varphi}_3^2 \\
&d_3 \cdot \sin\varphi_3 \cdot F_{r3}^{x} + p_3 \sin\varphi_3 \cdot F_{r4}^{x} - d_3 \cos\varphi_3 \cdot F_{r3}^{y} - p_3 \cos\varphi_3 \cdot F_{r4}^{y} - j_3 \cdot \ddot{\varphi}_3 \\
&\quad = -M_{ext3} - M_3 + M_4
\end{aligned}
$$

In the above equations for the two-segment body, some terms on the left side contain the reaction force F_{r3}, which we defined earlier as the reaction force of the 2nd segment on the 3rd segment. These terms have only been included to show the regularity of the equations. Since the 3rd segment does not exist, the distal end of the 2nd segment is free, so F_{r3} (and also M_3) is zero. Thus, the only unknowns at the left side of the equations are indeed $F_{r1}{}^{x}$, $F_{r2}{}^{x}$, $F_{r1}{}^{y}$, $F_{r2}{}^{y}$, $\ddot{\varphi}_1$, and $\ddot{\varphi}_2$.

In the previous section we argued that the two segment angular accelerations alone make up a generalized set of coordinates for a two-segment body of which the first joint axis is fixed. The minimum number of equations needed to model this system should therefore be two. Apparently we have four equations more than needed. As the linear acceleration of any joint can be expressed in terms of the segment angular accelerations (as we did earlier for the center of gravity), it follows from $F = m \cdot a$ that reaction forces can also be expressed in terms of segment angular accelerations. Subsequently, the result can be substituted in the rotational equations to reduce our number of equations to two. The equation itself will of course be much more complicated, and the reaction forces will have to be calculated from the segment angular accelerations afterward. In this paper we shall continue to work with the $3 \cdot n$ equations that result from the Newton-Euler approach, and "let the machine do the dirty work" (Kernighan & Plauger, 1978). In the next section we will present a generic algorithm to automatically derive the equations of motion and solve for the unknowns, given a particular set of known variables.

Generic Algorithm for Solving the Equations of Motion

Matrix Notation of the Equations of Motion

First we shall rewrite in matrix form the equations of motions for the two-segment body of the previous section, so they can easily be solved on a computer by means of *Gaussian elimination* (Strang, 1980). Given that $F_{r3}{}^{x} = F_{r3}{}^{y} = M_3 = 0$, the six equations for horizontal and vertical accelerations and rotations turn into:

$$
A \cdot x =
\begin{bmatrix}
1 & -1 & 0 & 0 & m_1 \cdot d_1 \cdot \sin\varphi_1 & 0 \\
0 & 1 & 0 & 0 & m_2 \cdot \ell_1 \cdot \sin\varphi_1 & m_2 \cdot d_2 \cdot \sin\varphi_2 \\
0 & 0 & 1 & -1 & -m_1 \cdot d_1 \cdot \cos\varphi_1 & 0 \\
0 & 0 & 0 & 1 & -m_2 \cdot \ell_1 \cdot \cos\varphi_1 & -m_2 \cdot d_2 \cdot \cos\varphi_2 \\
d_1 \cdot \sin\varphi_1 & p_1 \cdot \sin\varphi_1 & -d_1 \cdot \cos\varphi_1 & -p_1 \cdot \cos\varphi_1 & -j_1 & 0 \\
0 & d_2 \cdot \sin\varphi_2 & 0 & -d_2 \cdot \cos\varphi_2 & 0 & -j_2
\end{bmatrix}
\cdot
\begin{bmatrix}
F_{r1}^{x} \\ F_{r2}^{x} \\ F_{r1}^{y} \\ F_{r2}^{y} \\ \ddot{\varphi}_1 \\ \ddot{\varphi}_2
\end{bmatrix}
=
$$

$$= \begin{bmatrix} -F^x_{ext1} + m_1 \cdot \ddot{x} - m_1 \cdot d_1 \cdot \cos\varphi_1 \cdot \dot{\varphi}_1^2 \\ -F^x_{ext2} + m_2 \cdot \ddot{x} - m_2 \cdot \ell_1 \cdot \cos\varphi_1 \cdot \dot{\varphi}_1^2 - m_2 \cdot d_2 \cdot \cos\varphi_2 \cdot \dot{\varphi}_2^2 \\ -F^y_{ext1} + m_1 \cdot \ddot{y} - m_1 \cdot d_1 \cdot \sin\varphi_1 \cdot \dot{\varphi}_1^2 \\ -F^y_{ext2} + m_2 \cdot \ddot{y} - m_2 \cdot \ell_1 \cdot \sin\varphi_1 \cdot \dot{\varphi}_1^2 - m_2 \cdot d_2 \cdot \sin\varphi_2 \cdot \dot{\varphi}_2^2 \\ -M_{ext1} - M_1 + M_2 \\ -M_{ext2} - M_2 \end{bmatrix} = b$$

In the *A* matrix we can identify several blocks that can easily be expanded if an extra segment is added to the chain. We shall take this up in the next section.

Generic Algorithm for Skeletal Dynamics

To allow for a translational displacement of the whole body with respect to the environment, we have introduced the most proximal point of the chain, i.e., the first (virtual) joint, as the base. So far we have assumed the base was fixed. Although the number of *DF* for an *n*-segment chain in two dimensions with fixed base is only *n*, $3 \cdot n$ equations of motions were needed using the Newton-Euler approach. Theoretically the base does not have to be fixed, so the number of *DF* may increase to $n+2$. To allow for a free base, we include the position and velocity of the base in the vector of state variables (i.e., they have to be known), and the acceleration of the base may act as unknown, analogous to the rotational displacements.

Surprisingly, the number of equations remains $3 \cdot n$ if the base is free. In fact the equations of motion do not change at all! It seems we now have a problem. After all, we introduced two additional unknowns, $\ddot{x}$ and $\ddot{y}$, so it seems we end up with a system of $3 \cdot n$ equations in $3 \cdot n+2$ unknowns, which cannot be solved. However, if the base is free (i.e., the system is airborne), the ground reaction forces must be zero. In other words, the variables $F_{r1}{}^x$ and $F_{r1}{}^y$ are no longer unknown! In summary, if the base is fixed, $\ddot{x}$ and $\ddot{y}$ are known (viz. zero), and $F_{r1}{}^x$ and $F_{r1}{}^y$ are unknown. If the base is free, $F_{r1}{}^x$ and $F_{r1}{}^y$ are known (viz. zero), and $\ddot{x}$ and $\ddot{y}$ are unknown. By appropriately exchanging known and unknown variables, the number of unknowns remains $3 \cdot n$.

The idea to exchange known and unknown variables can be extended further. Given $3 \cdot n$ equations of motion for a *n*-segment body, we can have at most $3 \cdot n$ unknowns. A close look at our equations reveals that we have a pool of no less than $7 \cdot n+5$ variables, namely:

$n+1$ horizontal joint reaction forces, $n+1$ vertical joint reaction forces,
$n+1$ joint moments, n horizontal external forces, n vertical external forces,
n external moments, n segment angular accelerations,
1 horizontal base acceleration, and 1 vertical base acceleration.

Although for forward dynamics problems the unknowns are more or less defined, in principle any of these variables may be unknown. As we can have at most $3 \cdot n$ unknowns, the only restriction we must impose is that at least $4 \cdot n + 5$ of the variables be known. Given the values for $4 \cdot n + 5$ variables, the equations of motion can be written in the matrix form $A \cdot x = b$, as we did in the previous section. All terms containing an unknown are moved to the left side of the equations; they will form matrix *A*. All other terms are moved to the right side of the equations;

they will form the inhomogeneous column vector *b*. This way we can construct a generic algorithm capable of calculating forward dynamics, inverse dynamics, and any mixture of both.

Let us name this algorithm *segdyn*. It takes as input parameters two arrays of length $7 \cdot n + 5$. The first array will be named *K*, the second *V*. The elements in *K* can take on the Boolean truth-values TRUE or FALSE. The elements in *V* represent the variables, so they take on floating point values. The idea is that the elements in *K* tell us which variables are known and which are not, and that *V* contains the appropriate values for the known variables. More precisely, if *K*[*i*] equals TRUE, *V*[*i*] should contain the appropriate value for the *i*th variable. The order of variables should be predefined. Assume we agree to the following order:

$V[1] \cdots V[n+1]$: $F^x_{r1}, F^x_{r2}, \cdots, F^x_{r\,n+1}$

$V[n+2] \cdots V[2n+2]$: $F^y_{r1}, F^y_{r2}, \cdots, F^y_{r\,n+1}$

$V[2n+3] \cdots V[3n+3]$: $M_1, M_2, \cdots, M_{n+1}$

$V[3n+4] \cdots V[4n+3]$: $F^x_{ext1}, F^x_{ext2}, \cdots, F^x_{ext\,n}$

$V[4n+4] \cdots V[5n+3]$: $F^y_{ext1}, F^y_{ext2}, \cdots, F^y_{ext\,n}$

$V[5n+4] \cdots V[6n+3]$: $M_{ext1}, M_{ext2}, \cdots, M_{ext\,n}$

$V[6n+4] \cdots V[7n+3]$: $\ddot{\varphi}_1, \ddot{\varphi}_2 \cdots, \ddot{\varphi}_n$

$V[7n+4]$: $\ddot{x}$

$V[7n+5]$: $\ddot{y}$

Initially we consider all variables unknown. Thus the equations of motion are written such that all terms which contain a variable from *V* appear at the left, and only terms that do not contain any variable from *V* appear at the right. The initial vector of unknowns, which we shall refer to as *x*', just equals *V*. The initial matrix containing the multipliers of the unknown variables, say *A*', will be a $3 \cdot n \times 7 \cdot n+5$ matrix. After all, since the column vector of unknowns *x*' has dimension $7 \cdot n+5$, the rows of *A*' must contain $7 \cdot n+5$ elements. The matrix *A*' is big and might look quite complex, but it is nothing more than the matrix *A* given for the two-segment body in the section on Equations of Motion, extended with the multipliers for the forces and moments that were previously part of the right side. For a general *n*-segment body, several blocks can be identified in *A*':

$A' =$

	F_r $2 \cdot (n+1)$	M $n+1$	F_{ext} $2 \cdot n$	M_{ext} n	$\ddot{\varphi}$ n	$\ddot{x}, \ddot{y}$ 2
$2 \cdot n$	*Block 1*	*Block 2*	*Block 3*	*Block 4*	*Block 5*	*Block 6*
n	*Block 7*	*Block 8*	*Block 9*	*Block 10*	*Block 11*	*Block 12*

The beautiful thing about *A'* is that all blocks but 5 and 7 are very simple and do not change over time. With the system $A \cdot x = b$ as a guide, they can readily be identified. For a 3-segment body they read:

$$A' = \left[\begin{array}{ccccccc|ccccc|ccc|cc} 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -m_1 & 0 \\ 0 & 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & -m_2 & 0 \\ 0 & 0 & 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & -m_3 & 0 \\ 0 & 0 & 0 & 0 & 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & -m_1 \\ 0 & 0 & 0 & 0 & 0 & 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & -m_2 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & -m_3 \\ \hline & & & & & & & & 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ & & & & & & & & 0 & 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ & & & & & & & & 0 & 0 & 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{array}\right]$$

Block 5

Block 7

During simulations, algorithm *segdyn* need only fill in these blocks once. Subsequent invocations of *segdyn* only require updating Blocks 5 and 7. These blocks need updating because they depend on the current orientation of the skeleton. Closely following $A \cdot x = b$ again, we arrive at the following blocks for a *n*-segment body:

Block 5 =

$$\begin{bmatrix} m_1 \cdot d_1 \cdot \sin\varphi_1 & 0 & \cdots & 0 & 0 \\ m_2 \cdot \ell_1 \cdot \sin\varphi_1 & m_2 \cdot d_2 \cdot \sin\varphi_2 & 0 & \cdots & 0 \\ & & \cdots & & \\ m_n \cdot \ell_1 \cdot \sin\varphi_1 & m_n \cdot \ell_2 \cdot \sin\varphi_2 & \cdots & m_n \cdot \ell_{n-1} \cdot \sin\varphi_{n-1} & m_n \cdot d_n \cdot \sin\varphi_n \\ -m_1 \cdot d_1 \cdot \cos\varphi_1 & 0 & \cdots & 0 & 0 \\ -m_2 \cdot \ell_1 \cdot \cos\varphi_1 & -m_2 \cdot d_2 \cdot \cos\varphi_2 & 0 & \cdots & 0 \\ & & \cdots & & \\ -m_n \cdot \ell_1 \cdot \cos\varphi_1 & -m_n \cdot l_2 \cdot \cos\varphi_2 & \cdots & -m_n \cdot \ell_{n-1} \cdot \cos\varphi_{n-1} & -m_n \cdot d_n \cdot \cos\varphi_n \end{bmatrix}$$

Block 7 =

$$\begin{bmatrix} d_1 \cdot \sin\varphi_1 & p_1 \cdot \sin\varphi_1 & 0 & 0 & \cdots & 0 & -d_1 \cdot \cos\varphi_1 & -p_1 \cdot \cos\varphi_1 & 0 & 0 & \cdots & 0 \\ 0 & d_2 \cdot \sin\varphi_2 & p_2 \cdot \sin\varphi_2 & 0 & \cdots & 0 & 0 & -d_2 \cdot \cos\varphi_2 & -p_2 \cdot \cos\varphi_2 & 0 & \cdots & 0 \\ & & & & & & \cdots & & & & & \\ 0 & 0 & 0 & \cdots & d_n \cdot \sin\varphi_n & p_n \cdot \sin\varphi_n & 0 & 0 & 0 & \cdots & -d_n \cdot \cos\varphi_n & -p_n \cdot \cos\varphi_n \end{bmatrix}$$

The initial column vector of inhomogeneous terms, say *b'*, is similar to the column vector that was given for the two-segment body in the section Equations of Motion, except that all terms containing a force or a moment are left out. For a general *n*-segment body, we get the following $3 \cdot n$ column vector *b'*:

$$
\begin{bmatrix}
\left.\begin{matrix} -m_1 \cdot d_1 \cdot \cos\varphi_1 \cdot \dot{\varphi}_1^2 \\ -m_2 \cdot (\ell_1 \cdot \cos\varphi_1 \cdot \dot{\varphi}_1^2 + d_2 \cdot \cos\varphi_2 \cdot \dot{\varphi}_2^2) \\ \cdots \\ -m_n \cdot (\ell_1 \cdot \cos\varphi_1 \cdot \dot{\varphi}_1^2 + \ell_2 \cdot \cos\varphi_2 \cdot \dot{\varphi}_2^2 + \cdots + \ell_{n-1} \cdot \cos\varphi_{n-1} \cdot \dot{\varphi}_{n-1}^2 + d_n \cdot \cos\varphi_n \cdot \dot{\varphi}_n^2) \end{matrix}\right\} n \\
\left.\begin{matrix} -m_1 \cdot d_1 \cdot \sin\varphi_1 \cdot \dot{\varphi}_1^2 \\ -m_2 \cdot (\ell_1 \cdot \sin\varphi_1 \cdot \dot{\varphi}_1^2 + d_2 \cdot \sin\varphi_2 \cdot \dot{\varphi}_2^2) \\ \cdots \\ -m_n \cdot (\ell_1 \cdot \sin\varphi_1 \cdot \dot{\varphi}_1^2 + \ell_2 \cdot \sin\varphi_2 \cdot \dot{\varphi}_2^2 + \cdots + \ell_{n-1} \cdot \sin\varphi_{n-1} \cdot \dot{\varphi}_{n-1}^2 + d_n \cdot \sin\varphi_n \cdot \dot{\varphi}_n^2) \end{matrix}\right\} n \\
\left.\begin{matrix} 0 \\ 0 \\ 0 \\ 0 \end{matrix}\right\} n
\end{bmatrix}
$$

Given the system $A' \cdot x' = b'$, constructing the actual system $A \cdot x = b$ is straightforward. We loop sequentially through the array of truth-values *K*, and if the *i*th element happens to be TRUE, the *i*th column of *A'* is removed; furthermore, this column is multiplied with the *i*th value of the vector *V* (i.e., with the known value for the *i*th variable), and the resulting column is subtracted from *b'*. In other words, we gradually move the known terms of $A' \cdot x'$ to the right side *b'*. The result will be a $3 \cdot n$ column vector *b'* holding solely known terms, and a $3 \cdot n \times 3 \cdot n$ matrix *A'*, which precisely match the vector *b* and the matrix *A* we were looking for. The resulting square system has a unique solution that can be obtained through Gaussian elimination, for example with the LINPACK routines *GEFA* and *GESL* (Dongarra et al., 1979), and the *i*th value in the resulting vector *x* will then contain the appropriate value for the *i*th element in *V* whose truth-value in *K* was set to FALSE. An example implementation of this algorithm, *segdyn*, can be found at http://www.ifkb.nl/downloads/casius

Simulating a Movement Via Numerical Integration

From the right side *b*, and Blocks 5 and 7 given in the section on Generic Algorithm, it follows that the segment angles and segment angular velocities must be known in order to solve the equations of motion. At the start of a simulation, at $t = t_0$, these values should be made available by the user. Thereafter the segment angles change because of the angular velocities, which in turn change due to the angular accelerations imposed by the moments acting on the segments. The same holds for the base position; the new position depends on the base velocity, which in turn depends on the base acceleration imposed by the forces acting on the segments. In formal terms we are dealing with a system of coupled second-order ordinary differential equations (ODEs). Assuming that all external forces and moments are either constant or a function of the current state, for any time *t*, we may write:

$$\ddot{\varphi}_i(t) = f_i(\varphi_1(t) \ldots \varphi_n(t), \dot{\varphi}_1(t) \ldots \dot{\varphi}_n(t), x(t), y(t), \dot{x}(t), \dot{y}(t), M_1(t) \ldots M_n(t))$$
$$\ddot{x}(t) = g(\varphi_1(t) \ldots \varphi_n(t), \dot{\varphi}_1(t) \ldots \dot{\varphi}_n(t), x(t), y(t), \dot{x}(t), \dot{y}(t), M_1(t) \ldots M_n(t))$$
$$\ddot{y}(t) = h(\varphi_1(t) \ldots \varphi_n(t), \dot{\varphi}_1(t) \ldots \dot{\varphi}_n(t), x(t), y(t), \dot{x}(t), \dot{y}(t), M_1(t) \ldots M_n(t))$$

where $\varphi_i(t_0)$, $\dot{\varphi}_i(t_0)$, $x(t_0)$, $y(t_0)$, $\dot{x}(t_0)$, and $\dot{y}(t_0)$ are known. The variables φ_i, $\dot{\varphi}_i$, x, y, $\dot{x}$ and $\dot{y}$ will be referred to as state variables (technically they are the state variables of the equivalent set of coupled first-order ODEs we shall run into shortly) and their values at $t = t_0$ make up the initial state.

To compute solutions of *first*-order ODEs, many efficient numerical integration routines are available. Therefore our set of coupled second-order ODEs needs to be rewritten as an equivalent set of first-order ODEs. To that purpose we introduce the auxiliary variables ω_i, v_x, and v_y, and turn the second-order ODEs in the following set of coupled first-order ODEs:

$$\dot{\varphi}_i(t) = \omega_i(t)$$
$$\dot{x}(t) = v_x(t)$$
$$\dot{y}(t) = v_y(t)$$
$$\dot{\omega}_i(t) = f_i(\varphi_1(t) \ldots \varphi_n(t), \omega_1(t) \ldots \omega_n(t), x(t), y(t), v_x(t), v_y(t), M_1(t) \ldots M_n(t))$$
$$\dot{v}_x(t) = g(\varphi_1(t) \ldots \varphi_n(t), \omega_1(t) \ldots \omega_n(t), x(t), y(t), v_x(t), v_y(t), M_1(t) \ldots M_n(t))$$
$$\dot{v}_y(t) = h(\varphi_1(t) \ldots \varphi_n(t), \omega_1(t) \ldots \omega_n(t), x(t), y(t), v_x(t), v_y(t), M_1(t) \ldots M_n(t))$$

Given these first-order ODEs, we can apply a numerical integration routine to find a value for each state variable at any given moment.

Kinematic Constraints

As noted in the section on Skeletal Parameters, an n-segment body in two dimensions has at most $n+2$ degrees of freedom. In this case both ends of the chain may move freely. When the base is fixed, DF are lost. The motion is limited due to the kinematic constraints imposed on the system. As we saw earlier, the base can be fixed by swapping the base accelerations to known (viz. zero) and the joint reactions forces that act on the base to unknown. To prescribe a segment angular acceleration, a similar approach can be taken. The angular acceleration is marked as known, and a previously known variable, for example the segment's external moment, is switched to unknown. Kinematic acceleration constraints like these are special in that they involve exactly one of the $7 \cdot n + 5$ variables listed in the section on Generic Algorithm. Thus no additional equations are needed. However, in many motor tasks the $3 \cdot n$ equations of motion are simply not sufficient to express all necessary information. These situations demand constraint equations to be added to the system $A \cdot x = b$.

Constrained multibody systems have been studied extensively, and many methods have been developed for dealing with them. If we limit ourselves to constraints that can be expressed as a function of the state variables (often called holonomic constraints), we generally have to deal with a set of governing equations consisting of n differential equations of motion, together with a set of m algebraic constraints. Such systems of mixed equations are called differential-algebraic equations, or DAEs for short. The problem with these systems is that they typically are difficult to solve numerically (Brenan et al., 1989). Formerly the constraint equations were differentiated to reduce the DAEs to ODEs so as to apply standard numerical integration routines. The major drawback of this method is that due to the numerical

error, the state variables do not satisfy the kinematic constraints, and drift away. To defy these problems, Baumgarte (1972) introduced a stabilization method based on feedback control, and Park and Chiou (1988) took a similar approach based on a penalty form of the constrained equations. Closely related to the penalty method is the Lagrange multiplier technique (Shabana, 1989). A different method is the coordinate partitioning method (Wehage & Haug, 1982), which uses the m constraint equations to reduce the system of n constrained equations to a system of n-m independent equations. For an overview of many methods and their pros and cons, we refer to Schielen (1990).

In the next section we shall take a different approach and show that all constraints that can be expressed in terms of the $7 \cdot n + 5$ variables outlined in the section on Generic Algorithm can naturally be incorporated into our system $A' \cdot x' = b'$, so that the *segdyn* algorithm will automatically generate the proper system $A \cdot x = b$. This method is easy to understand, and also to implement, and it superbly reveals the power of the system $A' \cdot x' = b'$. The potential flaw is that the constraints on positional level are enforced through accelerations constraints, and thus are subject to violations due to numerical error. Even very small errors in the desired acceleration of a specific point in a multibody chain can quickly lead to problems, since they lead to cumulative errors in velocity and position. A supposedly fixed point, i.e., zero velocity, will quickly drift from its desired position if the error in the acceleration is too large at any instant during a simulation. After all, an error in the acceleration will lead to a nonzero velocity, and the velocity will remain nonzero, causing the position to change during each integration step. Therefore it is theoretically unsound to impose positional constraints by plainly applying acceleration constraints. We will discuss this later.

Constraint Equations

As a first example, consider a two-segment body with fixed base. Suppose the column vector x of unknowns reads: $[F_{r1}{}^x, F_{r2}{}^x, F_{r1}{}^y, F_{r2}{}^y, \ddot{\varphi}_1, \ddot{\varphi}_2]^T$. To state that the angular acceleration of the second segment should be $2 \cdot \pi$ rad/s^2, we could mark it as known and let M_{ext2} take its place in x, as described above. Alternatively, we could add the following constraint to the equations of motion: $\ddot{\varphi}_2 = 2 \cdot \pi$. This requires adding the row [0 0 0 0 0 1] to the matrix A, and $2 \cdot \pi$ to the right side b. We now end up with a system of $3 \cdot n+1$ equations in $3 \cdot n$ unknowns, which has infinitely many solutions. The best we can do is calculate a least-squares solution, which minimizes the sum of squared violations of the equations (e.g., Strang, 1980, section 3.2–3.5). This is normally not what we want during forward dynamics, so an extra unknown should be added, for example M_{ext2} mentioned above. The terms involving this particular unknown have to be added to the equations of motion. Things can be simplified by making use of the general implementation outlined in the section on Generic Algorithm. Given the matrix A' and the initial right side b', we can make use of the vector x' which consists of all possible $7 \cdot n+5$ variables. We add a row to A' and b', mark the variable M_{ext2} in the vector K as unknown, and the proper system $A \cdot x = b$ will then automatically be generated by the software!

Obviously, adding equations is not very useful if we can just exchange known and unknown variables, but it does provide a method for imposing kinematic constraints if this is not possible. For example, suppose we would like to fix the end of the chain. In setting up the initial state vector, we can make sure the initial velocity of the end-point is zero, for example by setting $\dot{\varphi}_1 = \dot{\varphi}_2 = 0$. Yet if we apply forces

and moments to the system, it will surely start to move, so we need a way to enforce zero-acceleration during the simulation. Unfortunately, neither the horizontal nor the vertical acceleration are part of the $7 \cdot n+5$ variables. However, from the section on Equations of Motion we know how to express these linear accelerations in terms of the segment angular accelerations:

$$\ddot{x}_{end} = \ddot{x} + -\ell_1 \cdot \cos\varphi_1 \cdot \dot{\varphi}_1^2 + -\ell_1 \cdot \sin\varphi_1 \cdot \ddot{\varphi}_1 + -\ell_2 \cdot \cos\varphi_2 \cdot \dot{\varphi}_2^2 + -\ell_2 \cdot \sin\varphi_2 \cdot \ddot{\varphi}_2$$
$$\ddot{y}_{end} = \ddot{y} + -\ell_1 \cdot \sin\varphi_1 \cdot \dot{\varphi}_1^2 + \ell_1 \cdot \cos\varphi_1 \cdot \ddot{\varphi}_1 + -\ell_2 \cdot \sin\varphi_2 \cdot \dot{\varphi}_2^2 + \ell_2 \cdot \cos\varphi_2 \cdot \ddot{\varphi}_2$$

where ($\ddot{x}$, $\ddot{y}$) is the base acceleration and ($\ddot{x}_{end}$, $\ddot{y}_{end}$) is the acceleration of the end-point in the chain. Thus, to impose zero-acceleration of the end-point (i.e., $\ddot{x}_{end} = \ddot{y}_{end} = 0$), the following equations must be added:

$$\ddot{x} - \ell_1 \cdot \sin\varphi_1 \cdot \ddot{\varphi}_1 - \ell_2 \cdot \sin\varphi_2 \cdot \ddot{\varphi}_2 = \ell_1 \cdot \cos\varphi_1 \cdot \dot{\varphi}_1^2 + \ell_2 \cdot \cos\varphi_2 \cdot \dot{\varphi}_2^2$$
$$\ddot{y} + \ell_1 \cdot \cos\varphi_1 \cdot \ddot{\varphi}_1 + \ell_2 \cdot \cos\varphi_2 \cdot \ddot{\varphi}_2 = \ell_1 \cdot \sin\varphi_1 \cdot \dot{\varphi}_1^2 + \ell_2 \cdot \sin\varphi_2 \cdot \dot{\varphi}_2^2$$

Given the order of the variables in the vector V that we agreed upon in the section on Generic Algorithm,

$V[1] \ldots V[n+1]$: $F_{r1}^x, F_{r2}^x, \ldots, F_{rn}^x+1$

...

$V[5n+4] \ldots V[6n+3]$: $M_{ext1}, M_{ext2}, \ldots, M_{extn}$

$V[6n+4] \ldots V[7n+3]$: $\ddot{\varphi}_1, \ddot{\varphi}_2 \ldots, \ddot{\varphi}_n$

$V[7n+4]$: $\ddot{x}$

$V[7n+5]$: $\ddot{y}$

it follows that the rows which have to be added to A' begin with $6 \cdot n + 3 = 15$ zeros, followed by the terms that multiply the variables at the left side of the constraint equations:

$$A' = \begin{bmatrix} & & & & & & & & & & & & & & & & A' & & & \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -\ell_1 \cdot \sin\varphi_1 & -\ell_2 \cdot \sin\varphi_2 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \ell_1 \cdot \cos\varphi_1 & \ell_2 \cdot \cos\varphi_1 & 0 & 1 \end{bmatrix}$$

The known terms from the right side of the constraint equations are added to b':

$$b' = \begin{bmatrix} & b' & \\ \ell_1 \cdot \cos\varphi_1 \cdot \dot{\varphi}_1^2 & + & \ell_2 \cdot \cos\varphi_2 \cdot \dot{\varphi}_2^2 \\ \ell_1 \cdot \sin\varphi_1 \cdot \dot{\varphi}_1^2 & + & \ell_2 \cdot \sin\varphi_2 \cdot \dot{\varphi}_2^2 \end{bmatrix}$$

Finally we add two unknowns to the system. The obvious variables of choice are the reaction forces at the end of the chain, which were set to zero when the joint was free: $K[n+1] = 0$ and $K[2 \cdot n+2] = 0$. In the *segdyn* algorithm, constraints should be added after A' and b' have been set up, and before the elements of K are used to construct the proper system $A \cdot x = b$ (see section on Model Consisting of One Leg). Note that in contrast to the previous example, the constraint equations depend on the current value of φ_i and $\dot{\varphi}_i$ Thus they must be updated after every integration step.

Theoretically the acceleration of any joint (or center of gravity) can be pre-

scribed this way. However, for the intermediate joints a problem arises. First, the reaction forces at an intermediate joint are already unknown, so they cannot be used to extend the number of unknowns. Second, if the acceleration of an intermediate joint is prescribed, the action = –reaction principle no longer holds. Actually the first problem results from the second. Since the action = –reaction principle no longer holds, separate reaction forces are needed for the connected segments. The forces acting on the proximal segment are needed to make sure that the prescribed accelerations are met for the distal point of this segment. Likewise, the proximal end of the distal segment can only obey the prescribed accelerations through the forces acting on this particular segment. The forces at both segments are completely unrelated.

A simple but inelegant workaround would be to mark as unknown the external forces acting on the center of gravity of one of the segments. This will not affect the movement but it does have some drawbacks. First, since the action = –reaction principle is still violated, the joint reaction forces no longer represent what they stand for. They now represent the forces that act on one of the segments to ensure that the constraints are not violated. Of course this is rather a matter of bad terminology than a serious problem. Second, the forces acting on the other segment to ensure that the constraints are not violated are not explicitly available. Although the external forces contain the appropriate values to satisfy the equations, the "real" external forces, like gravity, are ignored. That is, the external forces represent the net result of the "real" external forces and the forces needed to satisfy the kinematic constraints. Finally, in complex applications, constraints may be added from different parts in the software. Some constraints may even be triggered by external events. By haphazardly changing variables to unknown, conflicts can occur.

An appropriate solution to remove the latter two drawbacks would be to extend the *segdyn* algorithm in such a way that the most typical kinematic constraints, such as prescribing rotational and translation accelerations, are handled automatically. Internally, six additional variables per segment could be defined: four constraint forces (two acting on the proximal joint and two acting on the center of gravity) and two constraint moments (one acting from the proximal joint and one acting from the environment). The number of variables would become $13 \cdot n + 5$, but write access to the $6 \cdot n$ constraint variables could be reserved exclusively to the algorithm. Initially they would be marked known (viz. zero). Only when a kinematic constraint is added, the appropriate variable is changed to unknown and treated like the other forces and moments (i.e., terms that multiply the variable end up in the matrix A, and the variable itself ends up in the vector x). The above procedure is followed in MUSK (Casius, 1995).

Accuracy of Acceleration Constraints

As said before, it is theoretically unsound to impose positional constraints by plainly applying acceleration constraints; at least some kind of iterative refinement may be necessary. However, many years of experience have led us to believe that the state of the art of hardware and (numerical) software does allow for plain use of acceleration constraints in many practical situations, although care should be taken. When using numerical methods, numerical errors are unavoidable but they can be minimized. To reduce round-off error, one should always use at least double precision floating point calculations. Depending on the sophistication of the compiler/

interpreter, special care should be taken when writing down expressions, which may also be helpful for gaining speed of execution (Burden & Faires, 1989; Stoer & Bulirsch, 1980; Wilkinson, 1963).

To avoid large round-off errors during Gaussian elimination, one should use a robust algorithm, preferably one provided by (or originated from) a sound mathematical library. As it is, naive application of Gaussian elimination results in a very poor algorithm when run on a computer (Strang, 1980, pp. 42-43). Most important, one should use an integration routine suitable for the kind of differential equations that need to be solved, preferably one that can guard the accuracy (alternatively, one should use a conservatively small integration step). Finally, even if all precautions are taken, one should always double check to be sure that the constraints are indeed not violated during the simulation. Several sophisticated methods correct the value of the state variables after each integration step to satisfy the constraint equations. A drawback of this method may be that it is not always evident if a simulation ran correctly. The method outlined in the previous section does not control the violations of the equations in any way. Therefore it is always trivial to check if serious numerical error entered into a simulation by inspecting the desired positions and velocities. In other words, constraint violations are always explicitly visible. In the section on Sprint Cycling we shall build a complete model of a human cyclist and elaborate on the accuracy of kinematic acceleration constraints in practice.

Branches

Acceleration constraints also provide a way to implement branches within the framework of the chain model. To illustrate this, recall the model from Figure 3. A simple trick to attach a second leg to this model is to divide the trunk segment by two equally heavy trunk segments, with equally large moments of inertia. The proximal end of the second trunk segment is then attached to the distal end of the first one; that is, the chain is just extended with the newly created segment. The initial angle of the second trunk segment is set to the angle of the first one minus π, so both segments overlap exactly. The initial angular velocities of the segments should be equal to each other. Finally, to ensure that the overlap is preserved, a kinematic constraint is needed to impose that the accelerations of both segments are equal at any given time during the simulation. To that purpose we need to add the constraint equation $\ddot{\varphi}_{t1} - \ddot{\varphi}_{t2} = 0$, where $\ddot{\varphi}_{t1}$ is the angular velocity of the first trunk segment and $\ddot{\varphi}_{t2}$ is that of the second. Unquestionably, this is very plain and surely there is a more elegant method to implement branches; nonetheless, in the section on Model Consisting of Two Legs we shall use it to model the two-legged cyclist from Figure 4.

Sprint Cycling: A Comprehensive Example

Model Consisting of One Leg

For the study about human sprint cycling described in Soest and Casius (2000), the software package MUSK (Casius, 1995) was used. The skeletal model reflected the experimental setup from Beelen et al. (1994), in which the participants had to pedal at a fixed crank angular velocity so as to maximize the time average of the power transferred to the crank. The velocity of the crank was controlled by exter-

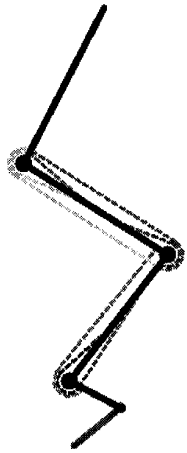

Figure 8 — Musculoskeletal model of a one-legged cyclist.

nal equipment, regardless of the forces the cyclist produced, so the legs could be assumed to be mechanically decoupled. Therefore the model was restricted to a single leg, and power output was doubled assuming left-right symmetry. A schematic representation of the model is given in Figure 8. The skeletal part consisted of five segments: the crank, foot, lower leg, upper leg, and head-arms-trunk segment. The latter segment was only necessary to have the hip spanned by muscles; its initial angle was set to 1.1 rad and kept constant during the entire simulation.The MUSK input file, *cycle.dat*, can be found at http:www.ifkb.nl/downloads/casius. Given this input, the equations of motion and the kinematic constraints are automatically derived in MUSK. The aim of this section is to rebuild the model in Matlab.

Several kinematic constraints must be defined. To fix the proximal end of the crank, zero-accelerations of the base must be imposed. To reflect that the cyclist remains seated on the saddle (in the experimental setup the cyclists were strapped to the saddle, preventing them from being lifted due to extremely high reaction forces), challenging kinematic constraints are needed. In addition to the linear acceleration constraints, the angular acceleration of the crank is set to zero. In the initial state vector, the angular velocity will be set to $-4 \cdot \pi$ rad/s, so the aim of this acceleration constraint is to preserve this clockwise rotation of 120 rpm during the entire simulation (as noted, the cyclists had to pedal at a fixed crank angular velocity). Finally, the angular acceleration of the last segment is set to zero, reflecting the fact that cyclists do not substantially move the upper body when holding the handle.

The initial state vector for the model consists of five segment angles, five segment angular velocities, a horizontal and vertical base position, and a horizontal and vertical base velocity. In the original model, the height of the saddle with respect to the base was based on the geometry of a bicycle. Since the position of the hip joint coincides with the saddle, there is only one *DF* left for the segment angles of the foot, lower leg, and upper leg, given a particular orientation of the crank. That is, for any particular crank angle, two of the three segment angles mentioned above directly follow from the third. The same holds for the angular velocities. Given an angular velocity for, say, the foot, the angular velocities of the upper and lower leg are laid down. The last point is very important. If the initial segment angular velocities of the foot and lower extremities are not carefully cho-

sen, the hip will drift away with constant velocity during the simulation. Zero-acceleration of the hip is enforced by kinematic constraints, but zero-velocity at the start of the simulation has to be enforced by carefully specifying the initial angular velocities of the segments.

To build the Matlab program, we shall make use of the general skeletal dynamics program, *segdyn*, to generate the 15 equations of motion. For this purpose the arrays *V* and *K* have to be set up properly. The various kinematic constraints (except to fix the base) will all be implemented by adding equations to the system $A' \cdot x' = b'$, even if they can be realized by adapting *V* and *K*. To get some action out of the cyclist, we need to apply net joint moments around the ankle, knee, and hip joint. In the original program these moments are delivered by the muscles. Since we do not implement muscle dynamics in this paper, the net joint moments that were delivered during a full revolution of the crank (i.e., 0.5 s) have been written to a file called *mom.mat*, with a fixed step size of 0.1 ms. After loading this file at the start of the program, the variable mom will contain 5001 rows of three numbers; the numbers in row *i* are the net joint moments around the ankle, knee and hip joint at $t = (i–1) / 10000$ s. The complete listing of the cycle program is given at http://www.ifkb.nl/downloads.casius and its main parts will be explained below.

The cycle program uses a slightly modified version of the *segdyn* program: one extra line is added that calls the routine in which all kinematic constraints are added to the system $A' \cdot x' = b'$ (in the code, *A'* is called *Atmp*, and *b'* just *b*; furthermore, ls is short for ℓ*sin, lc for ℓ*cos, phidsqr for $\dot{\varphi}^2$, and ni for n*i):

```
[K, Atmp, b] = add_cycle_constraints (K, Atmp, b, ls, lc, phidsqr)
```

In total, we need to add four constraints to the equations of motion. Two of them concern a segment angular acceleration: the angular acceleration of both the first and last segment should be zero. These constraints are straightforward, because the segment angular accelerations are part of the $7 \cdot n+5$ variables. From the section on Generic Algorithm we know that

$$V[6n + 4] \ldots V[7n + 3]: \quad \ddot{\varphi}_1, \ddot{\varphi}_2 \ldots, \ddot{\varphi}_n$$

To enforce zero-acceleration of the first segment (the crank), we create a vector of $7 \cdot n+5$ zeros, and replace the element in position $6 \cdot n+4$ with a 1. This vector constitutes the left side of the constraint equation, so it will be added to *Atmp*. The right side of the constraint equation, which represents the desired acceleration (i.e., zero), has to be added to the vector *b*. The constraint for the last segment is treated analogously:

```
ccrankphidd          = zeros (1,n7+5);
ctrunkphidd          = zeros (1,n7+5);
ccrankphidd(n6+4)    = 1;
ctrunkphidd(n7+3)    = 1;
Atmp                 = [Atmp; ccrankphidd; ctrunkphidd];
b                    = [b; 0; 0];
```

Note that the rows added to *Atmp* and *b* solely contain numbers. Therefore if we would store *Atmp* and *b* at the start of the simulation, these equations would not need updating (just as most of the blocks in *Atmp* would not need updating once they have been set up). In addition to these equations, we also have to add two

unknowns to the system. The obvious variables of choice are the external moments, which were set to zero in the main program. In the array *V*, the external moments reside at elements $n \cdot 5+4 \ldots n \cdot 6+3$, so we update *K* as follows:

```
K(n5+4) = 0;
K(n6+3) = 0;
```

Finally, we need to add two constraints to impose zero linear accelerations of the hip joint. These constraints are considerably more complicated. The linear accelerations of the hip can be expressed in terms of segment angular accelerations as follows (see sections on Equations of Motion, and Constraint Equations):

$$\ddot{x}_{hip} = \ddot{x} - \ell_1 \cdot \cos\varphi_1 \cdot \dot{\varphi}_1^2 - \ell_1 \cdot \sin\varphi_1 \cdot \ddot{\varphi}_1 - \ell_2 \cdot \cos\varphi_2 \cdot \dot{\varphi}_2^2 - \ell_2 \cdot \sin\varphi_2 \cdot \ddot{\varphi}_2 \cdots - \ell_4 \cdot \cos\varphi_4 \cdot \dot{\varphi}_4^2 - \ell_4 \cdot \sin\varphi_4 \cdot \ddot{\varphi}_4$$

$$\ddot{y}_{hip} = \ddot{y} - \ell_1 \cdot \sin\varphi_1 \cdot \dot{\varphi}_1^2 + \ell_1 \cdot \cos\varphi_1 \cdot \ddot{\varphi}_1 - \ell_2 \cdot \sin\varphi_2 \cdot \dot{\varphi}_2^2 + \ell_2 \cdot \cos\varphi_2 \cdot \ddot{\varphi}_2 \cdots - \ell_4 \cdot \sin\varphi_4 \cdot \dot{\varphi}_4^2 + \ell_4 \cdot \cos\varphi_4 \cdot \ddot{\varphi}$$

Given that $\ddot{x}_{hip} = \ddot{y}_{hip} = 0$, we arrive at the following constraint equations after moving the unknown terms to the left side:

$$\ddot{x} + -\ell_1 \cdot \sin\varphi_1 \cdot \ddot{\varphi}_1 + \cdots + -\ell_4 \cdot \sin\varphi_4 \cdot \ddot{\varphi}_4 = \ell_1 \cdot \cos\varphi_1 \cdot \dot{\varphi}_1^2 + \cdots + \ell_4 \cdot \cos\varphi_4 \cdot \dot{\varphi}_4^2$$

$$\ddot{y} + \ell_1 \cdot \cos\varphi_1 \cdot \ddot{\varphi}_1 + \cdots + \ell_4 \cdot \cos\varphi_4 \cdot \ddot{\varphi}_4 = \ell_1 \cdot \sin\varphi_1 \cdot \dot{\varphi}_1^2 + \cdots + \ell_4 \cdot \sin\varphi_4 \cdot \dot{\varphi}_4^2$$

Thus we have to loop over four segments to obtain the required equations:

```
cax          = zeroes (1,n7+5);
cay          = zeroes (1,n7+5);
rhsax        = 0;
rhsay        = 0;
cax (n7+4)   = 1;   % hor. base accel.
cay (n7+5)   = 1;   % vert. base accel.
for i = 1:4
    cax (n6+3+i)     = –ls(i);
    rhsax            = rhsax + lc(i)*phidsqr(i);
    cay (n6+3+i)     = lc(i);
    rhsay            = rhsay + ls(i)*phidsqr(i);
end
Atmp         = [Atmp; cax; cay];
b            = [b; rhsax; rhsay];
```

Clearly these constraints need updating after each integration step, since the elements depend on the current orientation and angular velocity of the first four segments. For simplicity, we will use the horizontal and vertical external force of the last segment as the new unknowns, but do recall from the section on Kinematic Constraints that this is not elegant:

```
K(n4+3) = 0;
K(n5+3) = 0;
```

Because the rotational acceleration equations neither directly nor indirectly depend on any state variable, they are not hard to satisfy. The kinematic constraints on the hip, however, are a different matter. In order to simulate more than one full crank cycle, we have run some tests with the original MUSK program, which has

the additional advantage that the simulations can be run much faster (considerably faster than real time on a moderately fast computer). Two integration methods were used (Stoer & Burlisch, 1980) and STEP (Shampine & Gordon, 1975). With Heun's method, four step sizes were used (*dt* = 1e-3, 2e-4, 5e-4, and 1e-4), and with STEP, eight fault tolerances (*eps* = 1e-3, 1e-4, ... , 1e-10); the lower the tolerance, the more integration steps are used. For each test, 10 crank revolutions were simulated. The position and velocity of the hip after 10 revolutions, as well as the power delivered to the crank during the last revolution, are reported in Table 1. For low integration accuracies, the cumulative error clearly shows in the hip position. The calculated power is also not very accurate when integration is sloppy.

With Heun's method, the deviation of the hip position is only in terms of micrometers when a step size of 0.2 ms or lower is used. With STEP, the deviation does not even show in the 8th decimal place when using a fault tolerance of 1e-6 or lower. In conclusion, even acceleration constraints that depend on various segment angles and segment angular velocities can adequately be used to impose constraints on position and velocity. However, and we cannot emphasize this too often, theoretically this is not guaranteed, so during each simulation one should check to see if the constraints are indeed not violated.

As a final remark, you may wonder how to calculate the power that is delivered to the crank. One possible answer lies in the (unknown) external moment that acts on the crank, which is precisely the variable that maintains zero-acceleration of the crank, i.e., nullifies the work that is transferred to the crank by the cyclist. Call this variable M_c, name the angular velocity of the crank $\dot{\phi}_c$, and define *P(t)* to be the product of $-M_c$ and $\dot{\phi}_c$ at time *t*. Then, $P(t)$ is the instantaneous power delivered to the crank, and the work *W* that is transferred to the crank during the full revolution is given by:

$$W = \int_0^{0.5} P(t) \cdot dt \quad \text{J},$$

and thus also:

$$W = \lim_{\Delta t \to 0} \sum_{i=0}^{n-1} P(t_i) \cdot \Delta t \quad \text{J}$$

where $t_0 = 0$ s, $t_n = 0.5$ s, and $\Delta t = 0.5/n$ s. For sufficiently small integration steps, this work can readily be calculated once we realize that routine *segdyn_cycle* returns the current value of M_c, in *Vnew(29)* (in the code, *Vnew* is a copy of *V* with all unknowns replaced by the calculated values):

```
Mold = Vnew(29);  W = 0; cnt = 1;
while ...
   ...
   if (cnt <= 5001)
      ...
      [stated, Vnew, ok] = segdyn_cycle(state, segparms, K, V);
      ...
      W = W – 0.5 * (Vnew(29) + Mold) * delta_time * state(6);
      Mold = Vnew (29);
   end
end
```

Table 1 Accuracy of Acceleration Constraints on the Hip During 10 Full Crank Revolutions of 0.5 s

Method	Accuracy (eps / dt)	# Integration steps	Power (W)	Hip position (m) x	Hip position (m) y	Hip velocity (m/s) Vx	Hip velocity (m/s) Vy
STEP							
	1e-3	8157	1084.5	–0.14245228	0.80896026	0.00008095	–0.00011669
	1e-4	16228	1076.8	–0.14269920	0.80929061	0.00000015	–0.00000053
	1e-5	20586	1076.4	–0.14270030	0.80929061	0.00000001	–0.00000016
	1e-6	22076	1076.6	–0.14270024	0.80929328	0.00000001	–0.00000000
	1e-7	24673	1076.4	–0.14270024	0.80929327	0.00000000	0.00000000
	1e-8	27971	1076.5	–0.14270024	0.80929327	–0.00000000	0.00000000
	1e-9	33802	1076.5	–0.14270024	0.80929327	–0.00000000	0.00000000
	1e-10	43254	1076.5	–0.14270024	0.80929327	–0.00000000	0.00000000
HEUN							
	1e-3	5000	1097.7	–0.14281498	0.80934918	0.00000003	0.00002149
	5e-4	10000	1081.5	–0.14272708	0.80929687	0.00000060	0.00000105
	2e-4	25000	1076.5	–0.14270475	0.80929350	0.00000002	–0.00000001
	1e-4	50000	1076.5	–0.14270138	0.80929333	–0.00000001	–0.00000000

Note: Two integration methods were used: STEP (Shampine & Gordon, 1975) and Heun's method. Depending on the integration method, the accuracy was varied either through the absolute fault tolerance *eps* (STEP) or the fixed step size *dt* (Heun's method) (Col. 2). Total number of integration steps and the power deliverd to the crank are listed in 3rd and 4th columns. Position and velocity of the hip at the end of the simulation are listed in the final columns. At the start of the simulation the position of the hip was (-0.14270024, 0.80929327) m.

Note that we store the crank moment after each integration step, to be able to use the avarage moment during a time interval. At the end of the program, the time average of the power transferred to the crank is calculated by dividing the total work by the total time (= 0.5 s), and to account for two legs, the result is multiplied by two:

```
s = sprintf ('Simulation finished at %.2f; Power: %f', time, 2*(W/time)); disp (s);
```

Using the net moments from the input file and a step size of 0.1 ms, this results in a power of 1076.3 W. Using a step size of 1.0 ms also results in a power output of 1076.3 W. This is considerably closer to the truth than the value listed in Table 1, because the net joint moments in the input file are not affected by the sloppy integration, whereas the solution of the differential equations corresponding to the muscle dynamics used in Table 1 is.

Model Consisting of Two Legs

To attach a second leg to the cyclist of the previous section, we shall use the trick outlined in the section on Branches. First we divide the trunk segment in two equally heavy trunk segments with equally large moments of inertia. The proximal end of the second trunk segment is attached to the distal end of the first one. The initial angle of the second trunk segment is set to the angle of the first one minus π, so both segments exactly overlap. The initial angular velocity of the second trunk is set to zero, equal to that of the first one.

Second, to ensure that the overlap is preserved, a kinematic constraint is added to impose that the accelerations of both segments are equal at any given time during the simulation. For this model, the trunk angle is supposed to be fixed, so we only need an additional constraint to enforce zero-acceleration of the second trunk, analogously to the constraint equation for the first trunk in the previous section:

```
ctrunk2phidd       = zeros (1,n7+5);
ctrunkphidd(n7+4) = 1;
Atmp               = [Atmp; ctrunk2phidd];
b                  = [b; 0];
K(n6+4)            = 0;  % external moment of second trunk becomes unknown
```

Note that in case the orientation of the trunk segment is not fixed, we need to add a constraint that enforces the angular accelerations of both trunk segments to be equal during the entire simulation.

Third, the segments that make up the second leg are attached to the chain. Note that the "second trunk segment," as well as the segments that make up the second leg, are sort of upside-down in the chain. After all, the proximal end is still defined as the end closest to the base. Therefore a vertical orientation of, say, the second lower leg corresponds with an angle of $-{}^1/_2 \cdot \pi$ rather than ${}^1/_2 \cdot \pi$. Furthermore, the distances from the proximal end to the center of gravity differ for corresponding segments from both sides of the body. For example, if ℓ and d are the parameters for the first lower leg, then, assuming symmetrical legs, the distance from the proximal end to the center of gravity for the second lower leg would become $\ell - d$, an unpleasant but inevitable consequence of the use of formal definitions in setting up the chain.

Finally, some adaptations must be made concerning the crank. In the approach taken in Figure 4a, two cranks are modeled. In addition to the linear acceleration constraints to fix the end-point of the chain (similar to those of the hip), this requires a constraint that imposes equal acceleration of both cranks. As the legs are no longer assumed to be mechanically decoupled, this has to be enforced by adding the equation $\ddot{\varphi}_{\text{crank1}} - \ddot{\varphi}_{\text{crank2}} = 0$. The external moment around one of the cranks may be used as an additional unknown, whereas the external moment around the other crank can be used to apply some kind of resistance, such as speed-dependent air friction.

The approach taken in Figure 4b only requires a single crank. In this model, care should be taken that the initial position and velocity of the end-points of the chain (the base, and the toes of the second leg) are equal. Furthermore, two complex constraint equations are needed to maintain equal position and velocity of both end-points. The joint reaction forces at the end of the chain are the obvious candidates for extending the number of unknowns. Clearly the approach consisting of two cranks is computationally less efficient than the model consisting of a single crank. The former approach requires 30 equations of motion and three kinematic constraints, whereas the latter approach only requires 27 equations of motion and two kinematic constraints.

As a side note, the power transferred to the crank cannot be calculated as before, since the external moment around the crank no longer nullifies the work delivered by the cyclist. Fortunately, the reaction forces that act upon the crank provide us with the same information. At time *t*, the instantaneous power *P(t)* delivered to the crank is given by:

$$P(t) = \dot{x}_{c1}(t) \cdot F_{c1}{}^{x}(t) + \dot{y}_{c1}(t) \cdot F_{c1}{}^{y}(t) + \dot{x}_{c2}(t) \cdot F_{c2}{}^{x}(t) + \dot{y}_{c2}(t) \cdot F_{c2}{}^{y}(t)$$

where $\dot{x}_{c1}$ and $\dot{y}_{c1}$ are the velocities of one end-point of the crank, $F_{c1}{}^{x}$ and $F_{c1}{}^{y}$ are the reaction forces that act on this end-point, $\dot{x}_{c2}$ and $\dot{y}_{c2}$ are the velocities of the other end-point of the crank, and $F_{c2}{}^{x}$ and $F_{c2}{}^{y}$ are the reaction forces that act upon this point. Thus, the work W_i transferred to the crank during the *i*th interval $[t_{i-1}, t_i]$ is given by: $W_i = 0.5 \cdot (P(t_{i-1}) + P(t_i)) \cdot (t_i - t_{i-1})$.

To conclude this section, note that it is even possible to model freewheeling for this two-legged cyclist. The trick is to define a virtual crank that can be coupled/decoupled to the real crank by means of appropriate kinematic constraints. Though this goes far beyond the scope of this paper, we mention it to indicate how sophisticated a simple 2-D chain model can become by applying kinematic constraints.

Summary

Many aspects regarding the properties and coordination of the musculoskeletal system can be adequately studied by performing simulations of relatively simple 2-D chain models of the skeleton. Generally a chain model consisting of *n* bodies has *n*+2 degrees of freedom. The minimum number of equations of motion is therefore also *n*+2. In the Newton-Euler approach a free body diagram is constructed for each segment. Although we end up with exactly three equations of motion per segment, the equations are relatively simple and a lot of information becomes available after the equations are solved. Due to the regularity of the equations, they can automatically be derived once the intrinsic parameters of the segments are known. By transforming the system of equations to the matrix form $A \cdot x = b$,

sound routines can be applied to solve the system. To perform a simulation, one must apply a numerical integration to solve the differential equations that describe how position and velocity of the segments depend on the acceleration. Finally, the general structure of the system $A \cdot x = b$ allows for a natural way to incorporate kinematic acceleration constraints. If care is taken, such constraints can adequately be used to impose constraints on position and velocity.

References

Barret, R., van Soest, A.J., & Neal, R. (2002). A computer graphics model of muscle activation and contraction dynamics. *Sports Biomechanics*, **105**, 121.

Baumgarte, J. (1972). Stabilization of constraints and integrals of motion in dynamic systems. *Computer Methods and Applications to Mechanical Engineering*, **1**, 1-16.

Beelen, A., Sargeant, A.J., & Wijkhuizen, F. (1994). Measurement of directional force and power during human submaximal and maximal isokinetic exercise. *European Journal of Applied Physiology*, **68**, 177-181.

Bobbert, M.F., & van Soest, A.J. (1994). Effects of muscle strengthening on vertical jump height: A simulation study. *Medicine and Science in Sports and Exercise*, **26**, 1012-1020.

Bogert, A.J. v.d. (1990). Musculoskeletal modelling: The DADS experience. *ISB Newsletter*, **39**, 4-6.

Brenan, K., Campbell, S., & Petzold, L. (1989). *Numerical solutions of initial-value problems in differential-algebraic equations*. New York: Elsevier.

Burden, R.L., & Faires, J.D. (1989). *Numerical analysis* (4th ed.). Boston: PWS-Kent.

Casius, L.J.R. (1995). *MUSK: A software system that supports computer simulations of large-scale realistic models of the neuro-musculo-skeletal system* (Final report of Cray Reasearch Grants Program CRG 94.15). Vrije Universiteit, Faculty of Human Movement Sciences.

Dickenson, J.A., Cook, S.D., & Leinhard, T.M. (1985). The measurements of shock waves following heel strike when running. *Journal of Biomechanics*, **18**, 415-422.

Dongarra, J.J., Bunch, J.R., Moler, C.B., & Stewart, G.W. (1979). *LINPACK user's guide*. Philadelphia: Society for Industrial and Applied Mathematics.

Gerristen, K.G.M., van den Bogert, A.J., & Nigg, B.M. (1995). Direct dynamics simulation of the impact phase in heel-toe running. *Journal of Biomechanics*, **28**, 661-668.

Goffe, W.L., Ferrier, G.D., & Rogers, J. (1994). Global optimization of statistical functions with simulated annealing. *Journal of Econometrics*, **60**(1/2), 65-100.

Gruber, K., Denoth, J., Stuessi, E., & Ruder, H. (1987). The wobbling mass model. *Biomechanics X-B*, Vol. 6b, 1095-1105.

Gruber, K., Ruder, H., Denoth, J., & Schneider, K. (1998). A comparative study of impact dynamics: Wobbling mass model versus rigid body models. *Journal of Biomechanics*, **31**, 439-444.

Hatze, H. (1981a). A comprehensive model for human motion simulation and its application to the take-off phase of the long jump. *Journal of Biomechanics*, **14**, 135-142.

Hatze, H. (1981b). *Myocybernetic control models of skeletal muscle: Characteristics and applications*. Pretoria: University of South Africa.

Haug, J.H. (1989). Computer-aided kinematics and dynamics of mechanical systems. Vol. 1: Basic methods. Needham Heights, MA: Allyn & Bacon.

Kane, T.R., & Levinson, D.A. (1985). *Dynamics: Theory and application*. New York: McGraw-Hill.

Kernighan, B.W., & Plauger, P.J. (1978). *The elements of programming style*. New York: McGraw-Hill.

Liu, W., & Nigg, B.M. (2000). A mechanical model to determine the influence of masses and mass distribution on the impact force during running. *Journal of Biomechanics*, **33**, 219–224.

Neptune, R.R. (1999). Optimization algorithm performance in determining optimal controls in human movement analyses. *Journal of Biomechanical Engineering*, **121**, 249-252.

Pandy, M.G., Zajac, F.E., Eunsup, S., & Levine, W.S. (1990). An optimal control model for maximum-height human jumping. *Journal of Biomechanics*, **23**, 1185-1198.

Park, K.C., & Chiou, J.C. (1988). Stabilization of computational procedures for constrained dynamical systems. *Journal of Guidance, Control, and Dynamics*, **11**, 365-370.

Schielen, W. (Ed.) (1990). *Multibody systems handbook*. Berlin: Springer Verlag.

Shabana, A. (1989). *Dynamics of multibody systems*. New York: Wiley.

Shampine, L.F., & Gordon, M.K. (1975). *Computer solution of ordinary differential equations. The initial value problem*. San Francisco: W.H. Freeman.

Soest, A.J.v., & Bobbert, M.F. (1993). The contribution of muscle properties in the control of explosive movements. *Biological Cybernetics*, **69**, 195-204.

Soest, A.J.v., Bobbert, M.F., & Ingen Schenau, G.J.v. (1994). A control strategy for the execution of explosive movements from varying starting positions. *Journal of Neurophysiology*, **71**, 1390-1402.

Soest, A.J.v., & Casius, L.J.R. (2000). Which factors determine the optimal pedaling rate in sprint cycling? *Medicine and Science in Sports and Exercise*, **32**, 1927-1934.

Soest, A.J.v., & Casius, L.J.R. (2003). The merits of a parallel genetic algorithm in solving hard optimization problems. *Journal of Biomechanical Engineering*, **125**, 141-146.

Soest, A.J.v., Haenen, W.P., & Rozendal, L.A. (2003). Stability of bipedal stance: The contribution of cocontraction and spindle feedback. *Biological Cybernetics*, **88**, 293-301.

Soest, A.J.v., & Smith, R.M. (2001). *Does the "no-slide-shooting" constraint limit rowing performance?* Paper presented at the Proceedings of the VIIIth International Symposium on Computer Simulation in Biomechanics, Milano, Italy.

Stoer, J., & Bulirsch, R. (1980). *Introduction to numerical analysis*. New York: Springer-Verlag.

Strang, G. (1980). *Linear algebra and its applications* (2nd ed.). New York: Academic Press.

Wehage, R.A., & Haug, E.J. (1982). Generalized coordinate partitioning for dimension reduction in analysis of constrained dynamic systems. *Journal of Mechanical Design*, **104**, 247-255.

Werff, K.v.d. (1977). *Kinematic and dynamic analysis of mechanisms, a finite element approach*. PhD Thesis, Delft University of Technology, Delft.

Wilkinson, J.M. (1963). *Rounding errors in algebraic processes*. Englewood Cliffs, NJ: Prentice Hall.

Yamaguchi, G.T. (2001). *Dynamic modeling of musculoskeletal motion. A vectorized approach for biomechanical analysis in three dimensions*. Norwell, MA: Kluwer Academic.

Zajac, F.E., & Gordon, M.E. (1989). Determinig muscle's force and action in multi-articular movement. *Exercise and Sport Science Reviews*, **17**, 187-230.

Zajac, F.E., & Winters, J.M. (1990). Modeling musculoskeletal movment systems: Joint and body-segment dynamics, musculotendinous actuation and neuromuscular control. In J.M. Winter & S.L.-Y. Woo (Eds.), *Multiple muscle systems* (pp. 121-148). New York: Springer Verlag.

Zatsiorsky, V.M. (1998). *Kinematics of human motion*. Champaign, IL: Human Kinetics.