# Chapter 14
# Humanoid Robotics

Even before modern robotics began to develop, philosophers, engineers, and artists were interested in machines similar to humans. The first known example of a humanoid mechanism, which design has been preserved and can still be rebuilt today, is a mechanical knight created by Leonardo da Vinci and presented to the Milanese ruler Ludovico Sforza around 1495. The mechanism had a kinematic structure similar to present humanoid robots and it could move by a system of wires and pulleys. More recently writers like Karel Čapek and Isaac Asimov thought of robots that have a form similar to humans. There are several reasons why humanoid robots are thought to be interesting:

- Human environments are built for humans, therefore a general-purpose robot designed for human environments, e.g., homes, factories, hospitals, schools, etc., should have a form similar to humans to successfully operate in such environments.
- It is more natural for humans to interact and communicate with robots that look and behave in like humans.
- A humanoid robot can serve as an experimental tool to test the theories about human behavior created by computational neuroscientists, interested in how the human brain operates.

It can be said that modern humanoid robotics started with a series of humanoid robots created at the University of Waseda in Tokyo, Japan. The first of these robots was WABOT-1 created in 1973.

Despite recent progress in related areas such a soft robotics and artificial intelligence, humanoid robots that can operate in human-populated environments, where they collaborate and communicate with people in a natural way, are still only a distant dream. Currently, humanoid robots are at the stage where they can execute a variety of tasks. Tasks that are for example used in humanoid robot competitions, e.g. DARPA Robotics Challenge, include:

1. **Drive**: drive a utility vehicle down a lane blocked with barriers.
2. **Egress**: get out of the vehicle and locomote to a specified area.
3. **Door**: open a door and travel through a doorway.
4. **Valve**: turn a valve actuated by a hand-wheel.

M. Mihelj et al., *Robotics*, https://doi.org/10.1007/978-3-319-72911-4_14

5. **Wall**: use a tool (drill or saw) to cut through a concrete panel.
6. **Surprise** task, which was not known until the day of competition: remove a magnetic plug from one socket, insert it in a different socket.
7. **Rubble**: cross a debris field or negotiate irregular terrain.
8. **Stairs**: climb the stairs.

Modern humanoid robots can already execute such tasks autonomously, providing the approximate state of the environment is known in advance. However, it is still difficult for modern humanoid robots to perform such tasks without some prior information about the environmental conditions that can be exploited by a programmer to prepare the humanoid robot for the execution of multiple tasks. Integration and continuous sequencing of multiple robot actions remains a problem and some degree of teleoperation is still needed when performing longer task sequences.

While most of the standard robotics methodologies regarding robot kinematics, dynamics, control, trajectory planning, and sensing are relevant also when developing humanoid robots, humanoid robotics needs to deal with several specific issues. The foremost is the problem of biped locomotion and balance. Unlike other robots, humanoid robots must walk and keep balance during their operation. In the aforementioned robotics challenge, locomotion turned out to be one of the biggest issues. The basic indicator that describes the balance of a humanoid robot is the concept of zero-moment point, usually abbreviated as ZMP. The concept of ZMP was introduced by Miomir Vukobratović in 1968. It is still the most widely used approach for generating dynamically stable walking movements in which the supporting foot or feet keep contact with the ground surface at all times. This is important to prevent the robot from falling. The basic concepts related to ZMP are described in Sect. 14.1.

Another specific issue that arises when programming humanoid robots is the very high number of degrees of freedom they require compared to standard industrial robots. While typical industrial robots only have 6 and seldom 7 degree of freedom, humanoid robots often have more than 30 degree of freedom. For example, one of the best known humanoid robots Honda Asimo has 34 degree of freedom: 3 in the head, 7 in each arm (3 in the shoulder, 1 in the elbow, and 3 in the wrist), 1 in the waist, 6 in each leg, and 2 in each hand. Such a large number of degrees of freedom makes classical robot programming with teach pendants and textual programming languages impractical. Instead we can exploit the similarity between humanoid robots and humans. Because of this similarity, humanoid robots can perform tasks in a similar way as humans do. This fact gives rise to an idea that instead of programming a humanoid robot, a human teacher can show to the robot how to execute the desired task. The robot can then attempt to replicate the human execution. This way of robot programming is called programming by demonstration or imitation learning. Its successful application requires that a robot transfers the demonstrated motion to its own kinematic and dynamic structure. Furthermore, since natural environments are rarely static but often change, the robot cannot simply replicate the observed movements. Instead, the observed movements should be adapted to the current environmental conditions. These topics are discussed in Sect. 14.2.

## 14.1 Biped Locomotion

Biped locomotion is an important topic in humanoid robotics. Here we focus on walking, which is distinguished from other forms of biped locomotion such as running by the constraint that at least one foot must always be in contact with the ground. As explained in the introduction, most of the modern humanoid robots exploit the zero-moment point principle to generate stable walking patterns.

### *14.1.1 Zero-Moment Point*

Throughout this section, we assume that the floor is flat and orthogonal to gravity. We start by analyzing the distribution of a vertical component of ground reaction forces (i.e., the component orthogonal to the ground, as shown in Fig. 14.1). The *zero-moment point* is defined as the point where the resultant of these forces intersects with the ground. We first focus on the motion in the sagittal plane (i.e., the plane that divides the body into the left and right part). As shown in Fig. 14.1, a component of the ground reaction force orthogonal to the ground must be positive at all contact points, otherwise the foot would lose contact with the ground as it is not rigidly attached to it. The zero-moment point $p_x$ according to the above definition can be calculated as follows

$$p_x = \frac{\displaystyle\int_{x_b}^{x_f} x f_z(x)\mathrm{d}x}{f_n}, \tag{14.1}$$

$$f_n = \int_{x_b}^{x_f} f_z(x)\mathrm{d}x, \tag{14.2}$$

where $f_z(x)$ is the vertical component of the ground reaction force at contact point $x$ and $f_n$ the net vertical ground reaction force. The reason why $p_x$ is called zero-moment point becomes clear if the moment at $p_x$ is calculated:

$$\begin{aligned}\tau(p_x) &= -\frac{\displaystyle\int_{x_b}^{x_f} (x - p_x) f_z(x)\mathrm{d}x}{f_n} = -\left(\frac{\displaystyle\int_{x_b}^{x_f} x f_z(x)\mathrm{d}x}{f_n} - p_x \frac{\displaystyle\int_{x_b}^{x_f} f_z(x)\mathrm{d}x}{f_n}\right) \\ &= -(p_x - p_x) = 0.\end{aligned} \tag{14.3}$$

Here we integrated the moment $\tau = -(x - p_x) f_z$ across the whole sole area, i.e. $x_b \leq x \leq x_f$. Thus the net moment at the zero-moment point $p_x$ is equal to zero. The zero-moment point is usually abbreviated as ZMP. It is the point on the ground surface where the net angular momentum is equal to zero. If it exists, ZMP is constrained to lie within the support polygon.

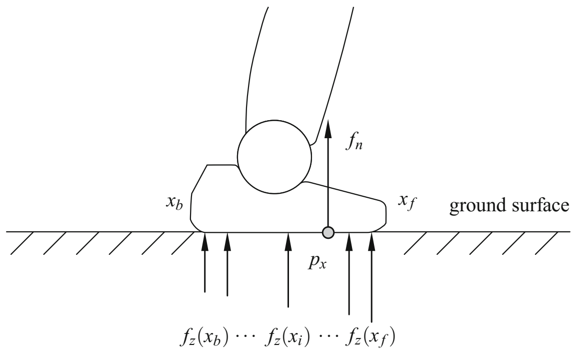

**Fig. 14.1** Ground reaction forces $f_z(x_i)$ at different contact points $x_i$. The zero-moment point $p_x$ and the net ground reaction force orthogonal to the support surface $f_n$ are calculated according to Eqs. (14.1) and (14.2), respectively

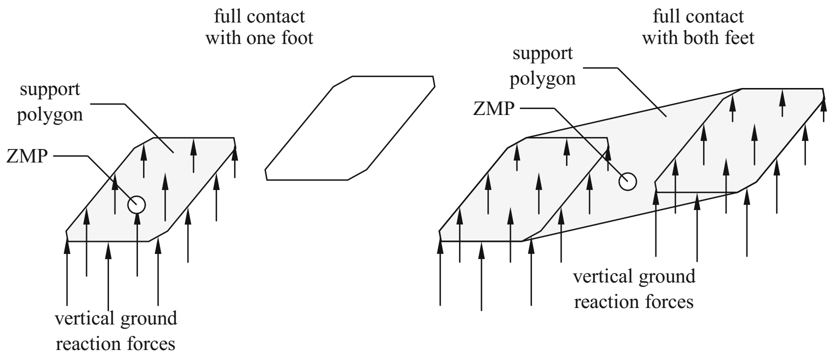
full contact with one foot
support polygon
ZMP
vertical ground reaction forces
full contact with both feet
support polygon
ZMP
vertical ground reaction forces

**Fig. 14.2** Support polygon (the area enclosed by a gray line) is defined as the convex hull of all points in contact with the ground. Left: support polygon corresponds to the sole area when only one foot is in full contact with the ground. Right: support polygon corresponds to the convex hull of the corners of both feet when both feet are in full contact with the ground

For general humanoid robot walking in 3-D, lateral motion should also be considered. As shown in Fig. 14.2, we must distinguish between two cases: either only one foot is in full contact with the ground or both feet are in full contact with the ground. The ground is assumed to be flat at height $p_z$. The derivation of ZMP is based on the relationship between the moment about point $\boldsymbol{p} = (p_x, p_y, p_z)$ of the vertical ground reaction force $[0, 0, f_z(\boldsymbol{\xi})]^{\mathrm{T}}$ at all points $\boldsymbol{\xi} = (\xi_x, \xi_y, p_z)$ on the contact surface. The moment is given by

$$\boldsymbol{\tau}(\boldsymbol{p}) = (\boldsymbol{\xi} - \boldsymbol{p}) \times \begin{bmatrix} 0 \\ 0 \\ f_z(\boldsymbol{\xi}) \end{bmatrix} = \begin{bmatrix} (\xi_y - p_y) f_z(\boldsymbol{\xi}) \\ -(\xi_x - p_x) f_z(\boldsymbol{\xi}) \\ 0 \end{bmatrix}. \tag{14.4}$$

To obtain the moment about point $\boldsymbol{p} = (p_x, p_y, p_z)$ due to the orthogonal ground reaction forces $[0, 0, f_z(\boldsymbol{\xi})]^{\mathrm{T}}$ arising at all points of contact $\boldsymbol{\xi}$ between the sole and the ground, we need to integrate across all points of contact

$$\boldsymbol{\tau}_n(\boldsymbol{p}) = \int_S \begin{bmatrix} \xi_x - p_x \\ \xi_y - p_y \\ 0 \end{bmatrix} \times \begin{bmatrix} 0 \\ 0 \\ f_z(\boldsymbol{\xi}) \end{bmatrix} \mathrm{d}S = \begin{bmatrix} \int_S (\xi_y - p_y) f_z(\boldsymbol{\xi}) \mathrm{d}S \\ -\int_S (\xi_x - p_x) f_z(\boldsymbol{\xi}) \mathrm{d}S \\ 0 \end{bmatrix}, \quad (14.5)$$

where $S$ denotes the area of contact. Similarly as in 2D case, the point on the ground where the moment of the normal of the ground reaction force becomes zero (i.e., the zero-moment point $\boldsymbol{\tau}_n(\boldsymbol{p}) = 0$), is given by

$$\boldsymbol{p} = \begin{bmatrix} p_x \\ p_y \\ p_z \end{bmatrix} = \begin{bmatrix} \dfrac{\int_S \xi_x f_z(\boldsymbol{\xi}) \mathrm{d}S}{f_n}, & \dfrac{\int_S \xi_y f_z(\boldsymbol{\xi}) \mathrm{d}S}{f_n}, & p_z \end{bmatrix}^{\mathrm{T}}, \quad (14.6)$$

where

$$f_n = \int_S f_z(\boldsymbol{\xi}) \mathrm{d}S \quad (14.7)$$

is the sum of ground reaction forces orthogonal to the ground at all contacts between the sole and the ground.

On a real humanoid robot, ZMP (if it exists), is guaranteed to lie within the support polygon because if the contact between the sole and the ground surface exists, the component of the ground reaction force orthogonal to the ground must be positive. Otherwise the contact between the sole and the ground surface would be lost as the robot is not fixed to the ground and therefore cannot generate negative vertical ground reaction forces. The humanoid robot can control its posture with its feet only if the ZMP exists inside the support polygon. Otherwise the robot loses contact with the ground and cannot control the posture with its feet any more.

### *14.1.2 Generation of Walking Patterns*

In biped walking, the robot's feet alternate between two phases:

- *stance* phase in which the foot's location should not change,
- *swing* phase in which the foot moves.

Figure 14.3 shows these two distinct phases in the gait cycle: when both feet are in contact with the ground, the robot is in *double* support phase. The feet do not move in this phase. Once one of the feet starts moving, the robot transitions from double to *single* support phase, in which one of the two feet moves. The single support phase is followed by another double support phase once the foot in the swing phase establishes a contact with the ground.

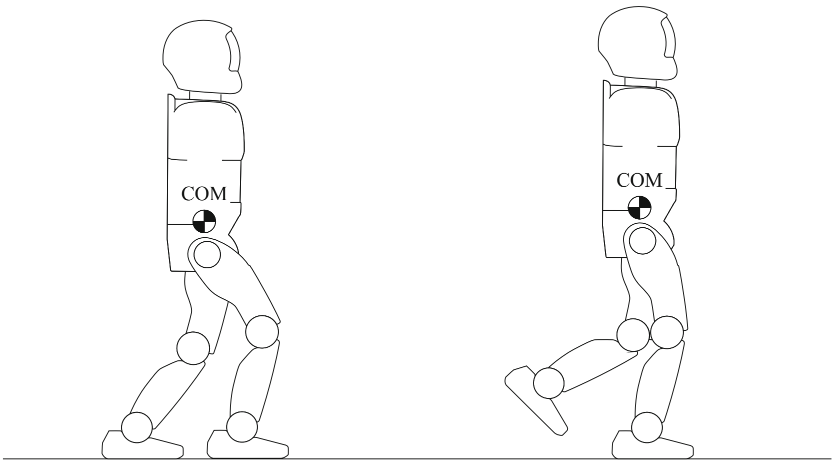

**Fig. 14.3** Single and double support phase. In the double support phase, both feet are in contact with the ground and the robot's weight is supported by both legs. In the single support phase, one foot is in motion, whereas the other foot supporting the robot is in contact with the ground

In ZMP-based walking, one or both feet of the robot are always in contact with the ground. Thus ZMP exists and the robot can keep balance by making sure that the support polygon contains the ZMP. However, the robot cannot directly control the ZMP as defined in Eqs. (14.1) and (14.6). We therefore introduce the concept of center of mass (COM). ZMP can be controlled by exploiting its relationship with COM.

*Center of mass* (COM) is defined as the average position of all body parts of a humanoid robot, weighted with the mass of body parts. For a robot with $D$ rigid links, COM can be calculated as:

$$\boldsymbol{c} = \frac{\sum_{i=1}^{D} m_i \boldsymbol{c}_i}{M}, \ M = \sum_{i=1}^{D} m_i, \tag{14.8}$$

where $m_i$ is the mass of $i$-th link and $\boldsymbol{c}_i$ its position, which can be calculated by direct kinematics provided the center of mass of each link is known in the link's local coordinates. With some approximations, the relationship between ZMP and COM can be specified as follows

$$p_x = c_x - \frac{(c_z - p_z)\ddot{c}_x}{\ddot{c}_z + g}, \tag{14.9}$$

$$p_y = c_y - \frac{(c_z - p_z)\ddot{c}_y}{\ddot{c}_z + g}, \tag{14.10}$$

where $p_z$ denotes the height of the ground floor, $g$ is the gravity constant, and $\boldsymbol{c} = (c_x, c_y, c_z)$ and $\boldsymbol{p} = (p_x, p_y, p_z)$ are the coordinates of COM and ZMP, respectively. Note that if the robot is at rest, i.e. $\ddot{c}_x = \ddot{c}_y = 0$, then ZMP and the projection of COM coincide as $p_x = c_x$ and $p_y = c_y$. Note also that if the ground is flat and orthogonal to gravity, as we assumed in Sect. 14.1.1, $p_z$ is a constant.

In general we distinguish between *static* and *dynamic walking*. Static walking is defined as any stable walking motion where the projection of COM always stays inside the support polygon. This means that if the robot completely stops moving at any moment during walking, it does not fall down because for the robot at rest, the projection of COM onto the ground surface is equal to the ZMP (see Eqs. (14.9) and (14.10)). In static walking the motion must generally be slow so that the projection of COM is close to the ZMP. This kind of walking typically requires large feet and strong ankle joints to generate sufficient forces at the ankles. As the robot's motion becomes faster, ZMP and the projection of COM become more different and stability cannot be ensured by controlling the projection of COM only.

More effective walking behaviors are generated by dynamic walking patterns, where the projection of COM is not equal to ZMP and can fall outside of the support polygon during some period of motion. A ZMP-based dynamic walking pattern is shown in Fig. 14.4. Such patterns are planned so that the ZMP remains within the boundary of the support polygon in all phases of walking. This can be accomplished as follows:

- Specify the Cartesian motion of the robot's feet. Here the robot's step length and timing of foot motion is prescribed.
- Specify the reference ZMP trajectory so that ZMP remains within the support polygon at all times.
- Determine the humanoid robot's upper body motion in order to realize the reference ZMP motion. This can be accomplished using Eqs. (14.9) and (14.10).
- The humanoid robot's leg motion is finally calculated from the body and feet motion using inverse kinematics.

The motion of COM is not fully specified by Eqs. (14.9) and (14.10) as there are only two equations and three unknown parameters. To fully specify the motion of COM and consequently the motion of the humanoid robot's upper body, an additional constraint must be imposed. There are several possible approaches. The simplest among them is to set the height of COM to a constant value (i.e., $c_z = \text{const}$, $\ddot{c}_z = 0$). With this assumption, the motion of COM is fully specified by Eqs. (14.9) and (14.10). A more adaptable and active motion can be achieved if $c_z$ is allowed to vary.

Note that the above approach determines the motion of COM without considering the legs. However, since most of the mass is usually concentrated in the upper body of a humanoid robot and since it is not necessary to follow the prescribed ZMP trajectory exactly, the above approach is sufficient to generate dynamically stable walking patterns.

If an accurate model of the robot is available, biped walking can be realized by simply following a predetermined walking pattern. Due to noise and model inaccuracies, in practice such an approach usually does not result in a stable walking behavior

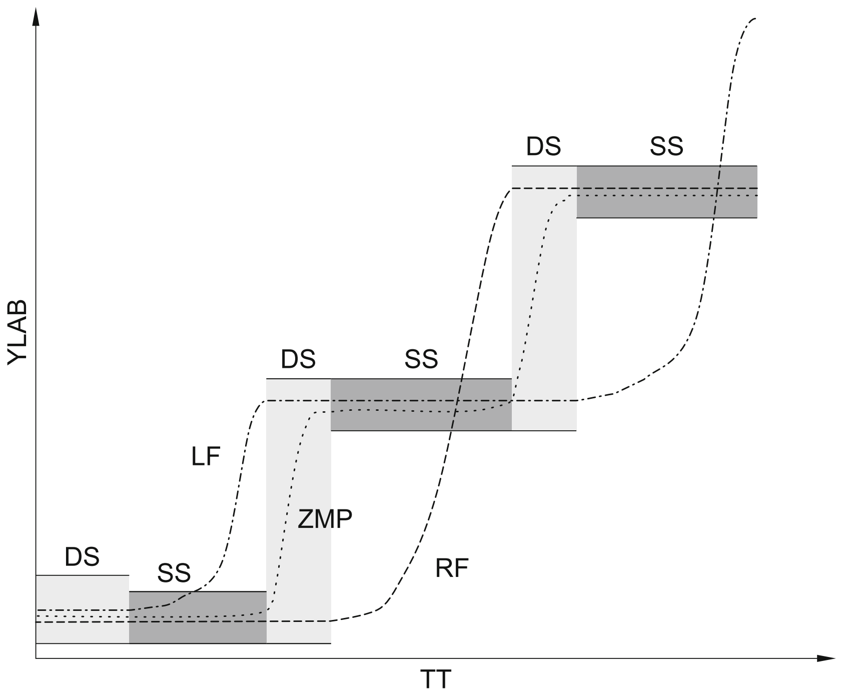

**Fig. 14.4** An example ZMP-based walking pattern in sagittal plane. The robot starts with both feet placed roughly parallel on the ground and then generates three steps, starting with the left foot. The shaded areas show the extent of support polygon during single support phase (dark shaded area) and double support phase (light shaded area). The ZMP trajectory (dotted) is planned in such a way that it remains within the support polygon during the whole duration of walking. The trajectories of both feet (left: dashed dotted, right: dashed) are also shown

without supplementing the precomputed walking pattern with a stabilizer that modifies the pattern according to the sensory input provided by gyros, accelerometers, force sensors, cameras, etc.

It should be pointed out that ZMP is not the only principle that can be used to generate stable walking patterns. It is possible to generate a walking pattern where a robot is unstable for some period of motion. Such walking patterns must be planned so that the robot can recover from instabilities before falling to the ground.

## 14.2 Imitation Learning

To fully exploit their potential, humanoid robots should be able to perform a variety of tasks in unstructured environments (e.g., people's homes, hospitals, shops, offices, and even outdoor environments). The aforementioned robotics challenge was geared towards humanoid robots at disaster sites. Unlike many industrial environments, where robots are widely used today, such environments cannot be prepared in advance to ease the operation of a humanoid robot. The programming of humanoid robots is further complicated by the large number of degrees of freedom involved in humanoid robot motion. Hence classic robot programming techniques based on teach pendants, carefully prepared off-line simulation systems, and programming languages are not sufficient for humanoids. Instead, it is necessary to equip humanoid robots with learning and adaptation capabilities. This way they can be programmed more easily and even autonomously acquire additional knowledge.

Learning of humanoid robot behaviors is a difficult problem because the space of all humanoid robot motions that needs to be explored is very large and increases exponentially with the number of degrees of freedom. A solution to this problem is to focus learning on those parts of the robot motion space that are actually relevant for the desired task. This can be achieved by *imitation learning*, also referred to as *programming by demonstration*. With this approach, a human teacher demonstrates to a robot how to perform the desired task. For it to work, a robot must be able to extract the important information from human demonstration and replicate the essential parts of task execution. While in most cases it is not necessary to exactly replicate the demonstrated movements to successfully execute the desired task, it is advantageous if the robot can mimic the demonstrated movement as much as possible. Since the body of a humanoid robot is similar to a human body, imitation learning is often a good approach to focus learning on the relevant parts of humanoid robot motion space.

### *14.2.1 Observation of Human Motion and Its Transfer to Humanoid Robot Motion*

There are many possible measurement systems and technologies that can be used to observe and measure human movements. They include

- optical motion capture systems,
- ensembles of inertial measurement units (IMU),
- computer vision methods for the estimation of human motion,
- passive exoskeletons,
- hand guiding.

In the following we explain the major advantages and drawbacks of these systems.

#### 14.2.1.1 Optical Tracking Devices for Human Motion Capture

Optical trackers are based on a set of markers attached to a human body. Markers can either be passive or active. Passive markers are made of retroreflective materials, which reflect light in the direction from where it came. In systems with passive markers, cameras are equipped with a band of infrared light emitting diodes (LEDs). The emitted light bounces off the marker back in the direction of the camera, making the marker much brighter than any other point in the image. This property makes retroreflective markers easy to detect in camera images. Using triangulation, a 3D marker location can be calculated if the marker is detected in at least two simultaneously acquired camera images. The predicted motion of visible markers is used to match the visible markers extracted at two successive measurement times.

Unlike passive markers that reflect light, active markers are equipped with LEDs and thus emit their own light. Consequently they must be powered. Optical trackers with active markers usually illuminate only one marker at a time for a very short time. Thus the system always knows which marker is currently visible, thereby providing the identity of the marker. For this reason, optical tracking systems with active markers can cope with temporary occlusions more effectively than systems with passive markers because an occluded active marker can be identified once it becomes visible again. This is not the case with passive markers. On the other hand, since active markers require power, they need to be connected to a power source with cables. This makes them more cumbersome to use than passive markers that require no cables.

To measure human motion, both passive and active markers must be attached to the human body segments at appropriate locations. Usually at least three markers are attached to each body segment, otherwise the location of rigid body segments cannot be estimated. Various special motion capture suits were designed in the past to ease the attachment of markers to the relevant body segments.

Optical tracking systems with active or passive markers provide 3-D locations of markers attached to the human body that are currently in view. The 3-D position and orientation of a body segment can be estimated if at least three markers attached to the segment are visible. In order to reproduce the observed motion with a robot, this information needs to be related to the robot motion. To a certain degree of accuracy, human motion can be modelled as an articulated motion of rigid body parts. If a humanoid robot kinematics is close enough to the human body kinematics, we can embed it into a human body as shown in Fig. 14.5. Such an embedding can later be used to estimate the joint angles from the orientations of successive body segments. Let us assume that the orientation of two successive body segments is given by orientation matrices $\mathbf{R}_1$ and $\mathbf{R}_2$ and that the joint linking the two segments consists of three successive joint axes $\mathbf{j}_1$, $\mathbf{j}_2$ and $\mathbf{j}_3$ with rotation angles denoted by $\varphi$, $\theta$, and $\psi$, respectively. We further assume that two consecutive joint axes are orthogonal and that all three axes intersect in a common point. In such an arrangement, the three joint angles correspond to the Euler angles introduced in Chap. 4. There are altogether 12 different joint axis combinations that cover every possible arrangement of axes in joints with three degrees of freedom. In Fig. 14.5, torso, neck, shoulder, wrist, and ankle joints can be described by an appropriate combination of Euler angles. The relationship between these values is given by

$$\mathbf{R}_1 = \mathbf{R}(\mathbf{j}_1, \varphi)\mathbf{R}(\mathbf{j}_2, \theta)\mathbf{R}(\mathbf{j}_3, \psi)\mathbf{R}_2 = \mathbf{R}(\varphi, \theta, \psi)\mathbf{R}_2. \tag{14.11}$$

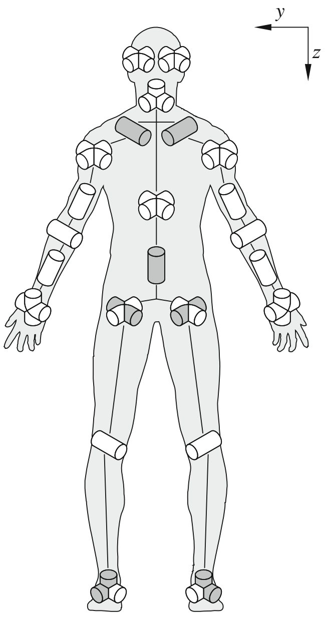

**Fig. 14.5** Kinematic structure of a humanoid robot. In the upright position with extended arms and legs, all joint axes are parallel to one of the three main axes of the body (forward/backward: $x$ axis, left/right: $y$ axis, up/down: $z$ axis)

The joint angles $\phi$, $\theta$, and $\psi$ can then be calculated by solving equation

$$\mathbf{R}(\varphi, \theta, \psi) = \mathbf{R}_1 \mathbf{R}_2^{\mathrm{T}}. \tag{14.12}$$

This equation depends on the choice of joint axes $\mathbf{j}_1$, $\mathbf{j}_2$, and $\mathbf{j}_3$. The observed motion can be replicated by a robot once all relevant joint angles from the embedded model have been estimated.

Optical tracking systems can also accurately estimate the absolute position and orientation of the human body in a world coordinate system. As the root of a humanoid robot's kinematics is typically assumed to be at the local coordinate frame attached to the torso, the estimated position and orientation of the torso corresponds to the absolute position and orientation of the human body in world coordinates.

#### 14.2.1.2 Inertial Measurement Units (IMUs)

Inertial measurement units (IMUs) contain different sensors including accelerometers to measure 3D linear acceleration and gyroscopes for measuring the rate of change of 3D orientation (i.e., angular velocity). IMUs also often include magnetometers to provide redundant measurements to improve accuracy and reduce the drift. From these data, the position and orientation of an IMU can be estimated as explained in Sect. 7.2.6.

In the context of transferring human motion to humanoid robot motion, IMU data can be used to estimate the position and orientation of each body segment which has an IMU attached. Just like with marker-based trackers, the joint angles can be estimated from orientations of successive body segments using Eq. (14.12).

Unlike optical tracking systems, IMUs do not suffer from occlusions as no external cameras are needed to measure the IMU motion. On the other hand, IMUs are not as accurate as optical tracking systems as they involve integration of linear acceleration and angular velocity. The integration can also cause drift, especially when estimating the absolute body position and orientation in space. Drift can be reduced by developing appropriate filters that exploit redundancy existing in the measurements obtained from accelerometers, gyros, and magnetometers.

#### 14.2.1.3 Passive Exoskeletons and Hand Guiding

A crucial issue that all of the above systems must deal with is that they measure human motion without considering the differences between the human and robot kinematics and dynamics. Such measurements must often be adapted to the robot constraints, otherwise the robot cannot execute the demonstrated movements. Alternatively, a nonlinear optimization problem can be formulated to adapt the demonstrated motion to the capabilities of a target robot.

The problem of transferring human motion to robot motion can be avoided by applying different measurement systems. One possibility is to design a special passive device, which is worn like exoskeleton with the degrees of freedom that correspond to the robot degrees of freedom. The passive exoskeleton must be designed in such a way that it does not restrict motion for most movements. It has no motors, but it should be equipped with goniometers to measure the joint angles. The joint angles measured by the exoskeleton can be used to directly control the robot if the kinematics of the target robot corresponds to the kinematic of the exoskeleton. One drawback of passive exoskeletons is that like clothes, they must be built to the specific size of a human demonstrator.

As explained in Sect. 12.3.2, some robots can be physically guided through the desired movements (see also Fig. 14.6). During hand guiding the movement is recorded by the robot's own joint angle sensors and is thus by default kinematically feasible. This approach is effective if the robot is compliant and can compensate for gravity, so that a human demonstrator can easily move it in the desired direction.

The main drawback of hand guiding is that the demonstration of the desired motion is less natural for a human demonstrator than for example when marker-based tracking systems are used. Thus with such systems it is sometimes not as easy to demonstrate complex movements. For example, hand guiding is not effective to demonstrate complex dancing movements. On the other hand, dancing can be easily demonstrated by a human directly and measured with an optical tracker, IMUs, or a passive exoskeleton.

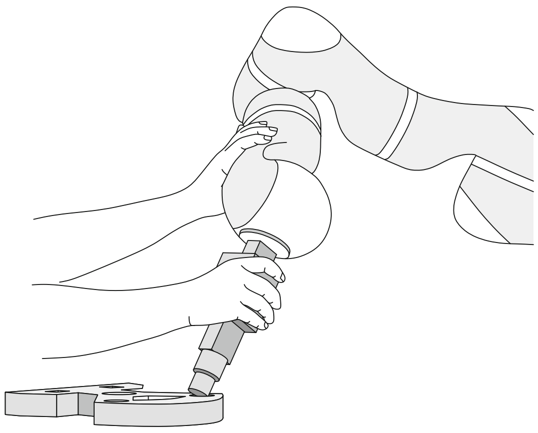

**Fig. 14.6** Demonstration of peg-in-hole task by kinesthetic teaching. Human demonstrator guides the anthropomorphic arm through the task execution with its own hands

## 14.2.2 Dynamic Movement Primitives

In Sect. 14.2.1 we discussed how to measure human demonstrations and how to transform the measured movements into the robot joint angle trajectories. In some cases it is also necessary to adapt the measured motion to the kinematic and dynamic capabilities of the target robot. Typically, we end up with a measurement sequence

$$\{\mathbf{y}_d(t_j), t_j\}_{j=1}^{T}, \tag{14.13}$$

where $\mathbf{y}_d(t_j) \in \mathbb{R}^D$ are the measured joint angles at time $t_j$, $D$ is the number of degrees of freedom, and $T$ is the number of measurements. This sequence defines the reference trajectory. However, for effective control we need to generate motor commands with the servo rate of the target robot. The robot's servo rate is often higher than the capture rate of the measurement system. Thus from the measurement data (14.13) we need to generate a continuous reference trajectory in order to generate motor commands to control the robot at the appropriate rate.

In this section we introduce *Dynamic Movement Primitives* (DMPs), which provide a comprehensive framework for the effective imitation learning and control of robot movements. DMPs are based on a set of nonlinear differential equations with well-defined attractor dynamics. For a single robot degree of freedom, here denoted by $y$ and taken to be one of the $D$ recorded joint angles, the following system of linear differential equations with constant coefficients is analyzed to derive a DMP

$$\tau \dot{z} = \alpha_z(\beta_z(g - y) - z), \tag{14.14}$$

$$\tau \dot{y} = z. \tag{14.15}$$

Note that auxiliary variable $z$ is just a scaled velocity of the control variable $y$. Constants $\alpha_z$ and $\beta_z$ have an interpretation in terms of spring stiffness and damping. For the appropriately selected constants $\alpha_z, \beta_z, \tau > 0$, these equations form a globally stable linear dynamic system with $g$ as a unique point attractor. We often refer to $g$ as the *goal* of the movement. This means that for any start configuration $y(0) = y0$, variable $y$ reaches the goal configuration $g$ after a certain amount of time, just as a stretched spring, upon release, will return to its rest position. $\tau$ is referred to as the *time constant*. It affects the speed of convergence to the attractor point $g$.

### *14.2.3 Convergence Properties of Linear Dynamic Systems*

Let us analyze why the above system is useful. We start by writing down a general solution of the non-homogenous linear differential equation system (14.14) and (14.15). It is well known that the general solution of such a system can be written as a sum of the particular and homogeneous solution

$$\begin{bmatrix} z(t) \\ y(t) \end{bmatrix} = \begin{bmatrix} z_p(t) \\ y_p(t)] \end{bmatrix} + \begin{bmatrix} z_h(t) \\ y_h(t) \end{bmatrix}. \tag{14.16}$$

Here $[z_p(t), y_p(t)]^{\mathrm{T}}$ denotes any function that solves the linear system (14.14)–(14.15), while $[z_h(t), y_h(t)]^{\mathrm{T}}$ is the general solution of the homogeneous part of Eqs. (14.14)–(14.15), i.e.,

$$\begin{bmatrix} \dot{z} \\ \dot{y} \end{bmatrix} = \frac{1}{\tau}\begin{bmatrix} -\alpha_z(\beta_z y + z) \\ z \end{bmatrix} = \mathbf{A}\begin{bmatrix} z \\ y \end{bmatrix}, \quad \mathbf{A} = \frac{1}{\tau}\begin{bmatrix} -\alpha_z & -\alpha_z\beta_z \\ 1 & 0 \end{bmatrix}.$$

It is easy to check that constant function $[z_p(t), y_p(t)]^{\mathrm{T}} = [0, g]^{\mathrm{T}}$ solves the equation system (14.14) and (14.15). Additionally, it is well known that the general solution of homogeneous system (14.17) is given by $[z_h(t), y_h(t)]^{\mathrm{T}} = \exp(\mathbf{A}t)\,\mathbf{c}$, where $\mathbf{c} \in \mathbb{R}^2$ is an arbitrary constant. Thus, the general solution of Eqs. (14.14) and (14.15) can be written as

$$\begin{bmatrix} z(t) \\ y(t) \end{bmatrix} = \begin{bmatrix} 0 \\ g \end{bmatrix} + \exp(\mathbf{A}t)\,\mathbf{c}. \tag{14.17}$$

Constant $\mathbf{c}$ should be calculated from the initial conditions, $[z(0), y(0)]^{\mathrm{T}} = [z0, y0]^{\mathrm{T}}$. The eigenvalues of $\mathbf{A}$ are given by $\lambda_{1,2} = \left(-\alpha_z \pm \sqrt{\alpha_z^2 - 4\alpha_z\beta_z}\right)/(2\tau)$. Solution (14.17) converges to $[0, g]^{\mathrm{T}}$ if the real part of eigenvalues $\lambda_{1,2}$ is smaller than 0, which is true for any $\alpha_z, \beta_z, \tau > 0$. The system is critically damped, which means that $y$ converges to $g$ without oscillating and faster than for any other choice of $\mathbf{A}$, if $\mathbf{A}$ has two equal negative eigenvalues. This happens at $\alpha_z = 4\beta_z$ where $\lambda_{1,2} = -\alpha_z/(2\tau)$.

### *14.2.4 Dynamic Movement Primitives for Point-to-Point Movements*

Differential equation system (14.14)–(14.15) ensures that $y$ converges to $g$ from any starting point $y_0$. It can therefore be used to realize simple point-to-point movements. To increase a rather limited set of trajectories that can be generated by (14.14) and (14.15) and thus enable the generation of general point-to-point movements, we can add a nonlinear component to Eq. (14.14). This nonlinear function is often referred to as *forcing term*. A standard choice is to add a linear combination of radial basis functions $\Psi_i$

$$f(x) = \frac{\sum_{i=1}^{N} w_i \Psi_i(x)}{\sum_{i=1}^{N} \Psi_i(x)} x(g - y_0), \tag{14.18}$$

$$\Psi_i(x) = \exp\left(-h_i\,(x - c_i)^2\right), \tag{14.19}$$

where $c_i$ are the centers of radial basis functions distributed along the phase of the trajectory and $h_i > 0$. The term $g - y_0,\ y_0 = y(t_1)$, is used to scale the trajectory if the initial and / or final configuration change. As long as the beginning and the end of movement are kept constant, this scaling factor has no effect and can be omitted. *Phase* variable $x$ is used in forcing term (14.18) instead of time to make the dependency of the resulting control policy on time more implicit. Its dynamics is defined by

$$\tau \dot{x} = -\alpha_x x, \tag{14.20}$$

with the initial value $x(0) = 1$. A solution to (14.20) is given by

$$x(t) = \exp\left(-\alpha_x t/\tau\right). \tag{14.21}$$

The appealing property of using the phase variable $x$ instead of explicit time is that by appropriately modifying Eq. (14.20), the evolution of time can be stopped to account for perturbations during motion. There is no need to manage the internal clock of the system. We obtain the following system of nonlinear differential equations

$$\tau \dot{z} = \alpha_z(\beta_z(g - y) - z) + f(x), \tag{14.22}$$

$$\tau \dot{y} = z. \tag{14.23}$$

The phase variable $x$ and consequently $f(x)$ tend to 0 as time increases. Hence the influence of nonlinear term $f(x)$ decreases with time. Consequently, through the integration of system (14.22)–(14.23) the system variables $[z, y]^{\mathrm{T}}$ are guaranteed to converge to $[0, g]^{\mathrm{T}}$, just like the linear system (14.14)–(14.15). The control policy specified by variable $y$ and its first- and second-order derivatives defines what we call a *dynamic movement primitive* (DMP). For a system with many degrees of freedom, each degree of freedom is represented by its own differential equation system (14.22)–(14.23), whereas the phase $x$ is common across all the degrees of freedom. This can be done because phase Eq. (14.20) does not include variables $y$ and $z$.

It is usually sufficient to determine the parameters $c_i$ and $h_i$ of Eq. (14.19) by setting a predefined distribution pattern and increasing the number of base functions $N$ until the desired reconstruction accuracy can be achieved. For example, for a given $N$ we can define

$$c_i = \exp\left(-\alpha_x \frac{i-1}{N-1}\right),\ i = 1, \ldots, N, \tag{14.24}$$

$$h_i = \frac{2}{(c_{i+1} - c_i)^2},\ i = 1, \ldots, N-1,\ h_N = h_{N-1}. \tag{14.25}$$

Note that $c_1 = 1 = x(0)$ and $c_N = \exp(-\alpha_x) = x(t_T)$.

In the equations above, $\alpha_x,\ \alpha_z$, and $\beta_z$ are set to constant values. The values must be chosen in such a way that the convergence of the underlying dynamic system is ensured as explained in Sect. 14.2.3. This is the case if we set $\alpha_x = 2$, $\beta_z = 3$, $\alpha_z = 4\beta_z = 12$.

DMPs were designed to provide a representation that enables accurate encoding of the desired point-to-point movements and at the same time permits modulation of different properties of the encoded trajectory. In this context, the shape parameters $w_i$ are determined so that the robot can accurately follow the desired trajectory by integrating the equation system (14.20), (14.22), and (14.23). The other parameters are used for modulation and to account for disturbances.

For a movement with two degrees of freedom, Fig. 14.7 shows a graphical plot of attractor fields generated by the dynamic movement primitive. The attractor field changes with the evolution of phase $x$. As long as the robot follows the demonstrated trajectory, the attractor field directs the robot to move along the demonstrated trajectory. However, if the robot is perturbed and deviates from the demonstrated trajectory, the attractor fields generated along the phase $x$ directs the robot so that it reaches the desired final configuration (goal), albeit along a modified trajectory.

A trajectory can be reproduced from a fully specified DMP by integrating Eqs. (14.22), (14.23), and (14.20) using Euler integration method:

$$z_{k+1} = z_k + \frac{1}{\tau}\left(\alpha_z(\beta_z(g - y_k) - z_k) + f(x_k)\right)\Delta t, \tag{14.26}$$

$$y_{k+1} = y_k + \frac{1}{\tau} z_i \Delta t, \tag{14.27}$$

$$x_{k+1} = x_k - \frac{1}{\tau}\alpha_x x_k \Delta t, \tag{14.28}$$

where $\Delta t > 0$ is the integration constant usually set to the robot's servo rate. The initial parameters for integration must be set to the current state of the robot, which

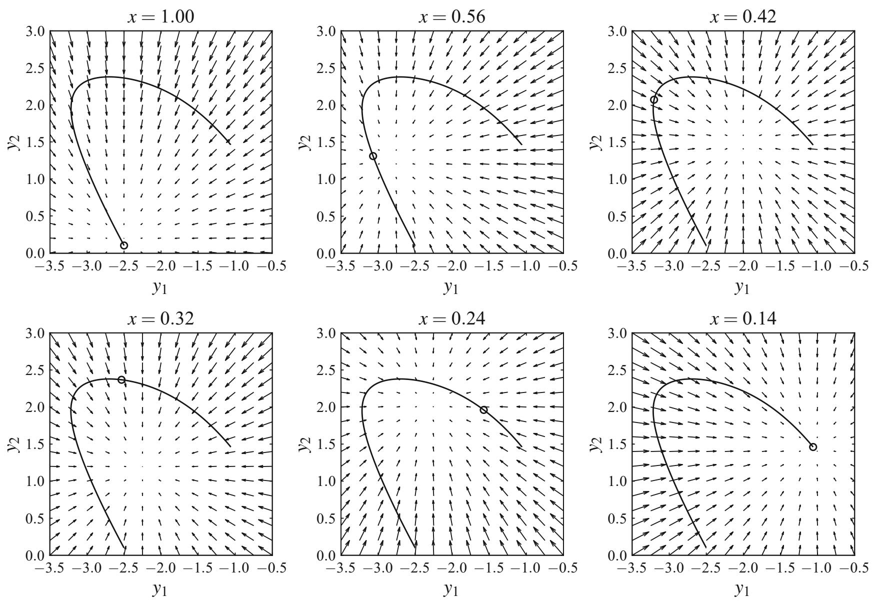

**Fig. 14.7** Plots of attractor fields generated by a DMP specifying the motion of a robot with two degrees of freedom $y_1$, $y_2$ as it is integrated along the phase $x$. The arrows in each plot show $\dot{z}_1$, $\dot{z}_2$ at different values of $y_1$, $y_2$ at the given phase $x$, assuming that only $y_1$ and $y_2$ have changed compared to the unperturbed trajectory. The circles show the desired configurations $y_1$, $y_2$ at the given phase $x$

at the beginning of motion is assumed to be at the given initial position and with zero velocity. This results in the following initialization formulas: $y_0 = y0,\ z_0 = 0,$ $x = 1$.

### *14.2.5 Estimation of DMP Parameters from a Single Demonstration*

To estimate the DMP representing the measurement sequence (14.13), we first compute the derivatives $\dot{\mathbf{y}}_j$ and $\ddot{\mathbf{y}}_j$ by numerical differentiation. For any of the $D$ degrees of freedom $y$, we obtain the following measurement sequence

$$\{y_\mathrm{d}(t_j),\ \dot{y}_\mathrm{d}(t_j),\ \ddot{y}_\mathrm{d}(t_j)\}_{j=1}^{T}, \tag{14.29}$$

where $y_d(t_j),\ \dot{y}_d(t_j),\ \ddot{y}_d(t_j) \in \mathbb{R}$ are the measured positions, velocities, and accelerations on the training trajectory and $T$ is the number of sampling points. Using the DMP movement representation, the trajectory of any smooth movement can be approximated by estimating parameters $w_i$ of Eq. (14.18). For this purpose we rewrite the system of two first-order linear Eqs. (14.22) and (14.23) as one second-order equation. This is done by replacing $z$ with $\tau \dot{y}$ in Eq. (14.22). We obtain

$$\tau^2 \ddot{y} + \alpha_z \tau \dot{y} - \alpha_z \beta_z (g - y) = f(x), \tag{14.30}$$

with $f$ defined as in Eq. (14.18). Note that time constant $\tau$ must be the same for all degrees of freedom. A possible choice is $\tau = t_T - t_1$, where $t_T - t_1$ is the duration of the training movement. On the other hand, the attractor point $g$ varies across the degrees of freedom. It can be extracted directly from the data: $g = y_\mathrm{d}(t_T)$. Writing

$$F_d(t_j) = \tau^2 \ddot{y}_\mathrm{d}(t_j) + \alpha_z \tau \dot{y}_\mathrm{d}(t_j) - \alpha_z \beta_z (g - y_\mathrm{d}(t_j)), \tag{14.31}$$

$$\mathbf{f} = \begin{bmatrix} F_d(t_1) \\ \dots \\ F_d(t_T) \end{bmatrix}, \; \mathbf{w} = \begin{bmatrix} w_1 \\ \dots \\ w_N \end{bmatrix},$$

we obtain the following system of linear equations

$$\mathbf{Xw} = \mathbf{f}, \tag{14.32}$$

which must be solved to estimate the weights of a DMP encoding the desired motion. The system matrix $\mathbf{X}$ is given by

$$\mathbf{X} = (g - y_0) \begin{bmatrix} \frac{\Psi_1(x_1)}{\sum_{i=1}^N \Psi_i(x_1)} x_1 & \dots & \frac{\Psi_N(x_1)}{\sum_{i=1}^N \Psi_i(x_1)} x_1 \\ \dots & \dots & \dots \\ \frac{\Psi_1(x_T)}{\sum_{i=1}^N \Psi_i(x_T)} x_T & \dots & \frac{\Psi_N(x_T)}{\sum_{i=1}^N \Psi_i(x_T)} x_T \end{bmatrix}. \tag{14.33}$$

The phase sampling points $x_j$ are obtained by inserting measurement times $t_j$ into Eq. (14.21). The parameters $\mathbf{w}$ can be calculated by solving the above system of linear equations in a least-squares sense. An example DMP estimation is shown in Fig. 14.8. The calculated DMP ensures that the robot reaches the attractor point $g$ at time $t_T$. Since DMPs have been designed to represent point-to-point movements, the demonstrated movement must come to a full stop at the end of the demonstration if the robot is to stay at the attractor point after $t_T$. If any other type of motion is approximated by a DMP, the robot will overshoot the attractor point and return back to it after the dynamics of the second-order linear system of differential equations starts dominating the motion. At least theoretically, the velocity does not need to be zero at the beginning of movement, but it is difficult to imagine a real programming by demonstration system in which such a trajectory would be acquired.

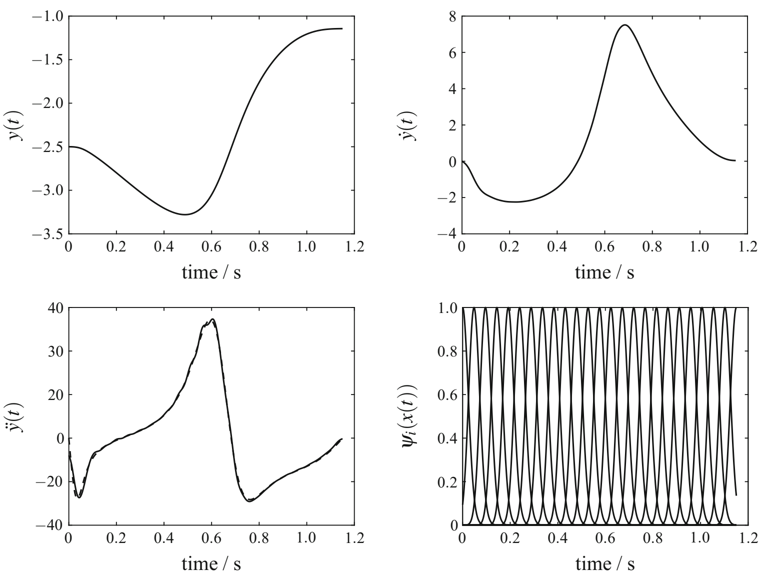

**Fig. 14.8** Time evolution of an example dynamic movement primitive: control variable $y$ and its derivatives, phase $x$, and radial basis functions $\psi_i$ are all shown with solid lines. Dashed lines show the demonstrated values of $y$, $\dot{y}$ and $\ddot{y}$

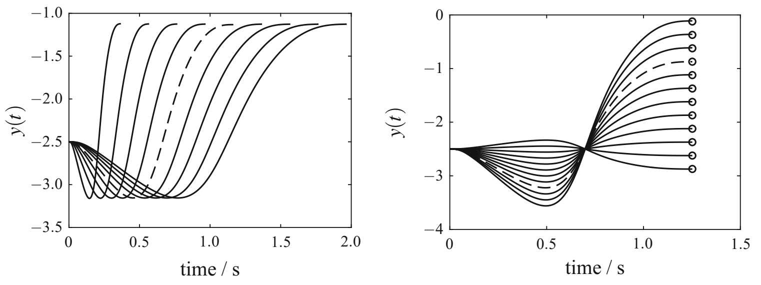

**Fig. 14.9** DMP modulations. The dashed trajectories show the original DMP without applying any modulation. **Left**: Time modulation. Solid trajectories show DMPs with changed $\tau$. **Right**: Goal modulation. Solid trajectories show DMPs with changed goal $g$. Circles show the goal position

## *14.2.6 Modulation of DMPs*

An important advantage of DMPs is that they enable easy modulation of the learnt movement. Figure 14.9 left shows that by changing parameter $\tau$, the movement can be sped up or slowed down. The same figure also shows that by changing the goal parameter $g$, the final configuration on the trajectory can be changed so that the robot moves to a new goal. The term $y_0 - g$ in the forcing term (14.18) ensures that the movement is appropriately scaled as the goal or initial configuration changes.

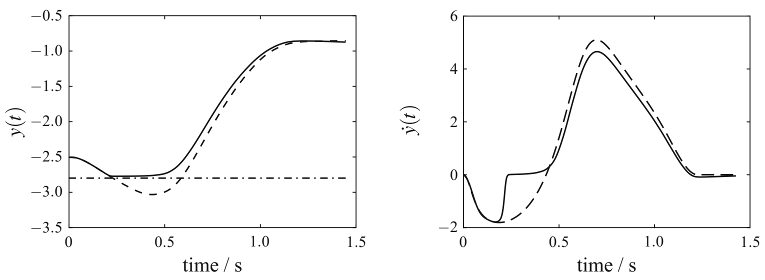

**Fig. 14.10** DMP modulation with joint limit avoidance at $y = -2.8$. The solid trajectories show the DMP trajectory and its velocity obtained by integrating (14.34) instead of (14.23), while the dashed trajectories show the original DMP and its velocity without applying any modulation

More complex modulations involve changing the underlying differential Eqs. (14.22), (14.23), and/or (14.20). For example, Eq. (14.23) can be changed to

$$\tau y = z - \frac{\rho}{(y_L - y)^3} \tag{14.34}$$

to implement the avoidance of a lower joint limit. This happens because once $y$ starts approaching $y_L$, the denominator in Eq. (14.34) becomes small and there is a significant difference between integrating Eq. (14.23) or (14.34). Figure 14.10 right shows that the second term in Eq. (14.34) acts as a repulsive force, preventing $y$ from approaching $y_L$ too closely. On the other hand, the denominator in Eq. (14.34) remains large as long as the joint angle $y$ is far away from the joint limit $y_L$. Thus in this case there is little difference between integrating Eq. (14.23) or (14.34) and the DMP generated trajectory follows the demonstrated movement. Note that it is not necessary to learn new parameters $w_i$, goal $g$, or time constant $\tau$ because of modulation. They can remain as they were initially learnt. Only Eq. (14.23) must be changed to (14.34) to ensure joint limit avoidance during on-line control.

The appealing property of applying the phase variable instead of time is that we can easily modulate the time evolution of phase, e.g., by speeding up or slowing down a movement as appropriate by means of coupling terms. Instead of integrating Eqs. (14.20) and (14.23) at time of execution, the modified Eqs. (14.20) and (14.36) could be integrated

$$\tau \dot{x} = -\frac{\alpha_x x}{1 + \alpha_{px}(y - \tilde{y})^2}, \tag{14.35}$$

$$\tau \dot{y} = z + \alpha_{py}(y - \tilde{y}), \tag{14.36}$$

where $y$ and $\tilde{y}$ respectively denote the desired and actual robot joint angle position, respectively. If the robot cannot follow the desired motion, $\alpha_{px}(y - \tilde{y})^2$ becomes large, which in turn makes the phase change $\dot{x}$ small. Thus the phase evolution is stopped until the robot catches up with the desired configuration $y$. This will

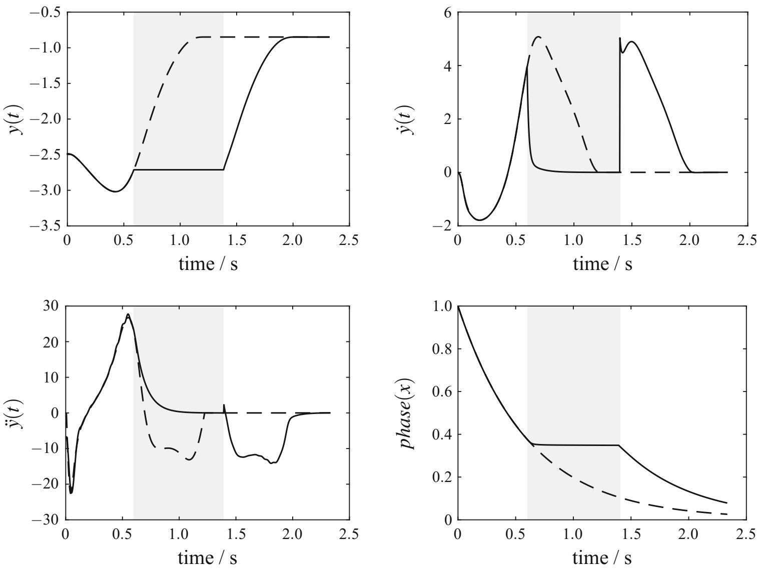

**Fig. 14.11** The effect of phase stopping caused by blocking the evolution of joint position $\tilde{y}$ in the time interval [0.6, 1.4] (grey area). The dashed trajectories show the original DMP, velocity, acceleration and phase evolution, while the solid trajectories show their counterparts from the perturbed motion with phase stopping enabled. Note that outside of the time interval [0.6, 1.4] where the joint motion is blocked, the robot accurately follows the desired motion

eventually happen due to the added term in Eq. (14.36). On the other hand, if the robot follows the desired movement precisely, then $\tilde{y} - y \approx 0$ and Eqs. (14.35) and (14.36) are no different from Eqs. (14.20) and (14.23), respectively. Thus in this case the DMP-generated movement is not altered. Figure 14.11 illustrates the effect of phase stopping when the robot's motion is temporarily blocked.

In summary, DMPs provide an effective representation for learning humanoid robot trajectories and to control humanoid robots. They are based on autonomous, nonlinear differential equations that are guaranteed to create smooth kinematic control policies. An important property of DMPs is that they can be learnt from a single demonstration of the desired task. They have several advantages compared to other motor representations including

- they possess free parameters that are easy to learn in order to reproduce any desired movement,
- they are not explicitly dependent on time and allow for time modulation,
- they are robust against perturbations,
- they are easy to modulate by adapting various parameters and equations.

Due to their flexibility and robustness, DMPs are considered a method of choice when learning robot trajectories from single demonstrations.