Pro-active trajectory formation
using a biomimetic and a biomechanical model

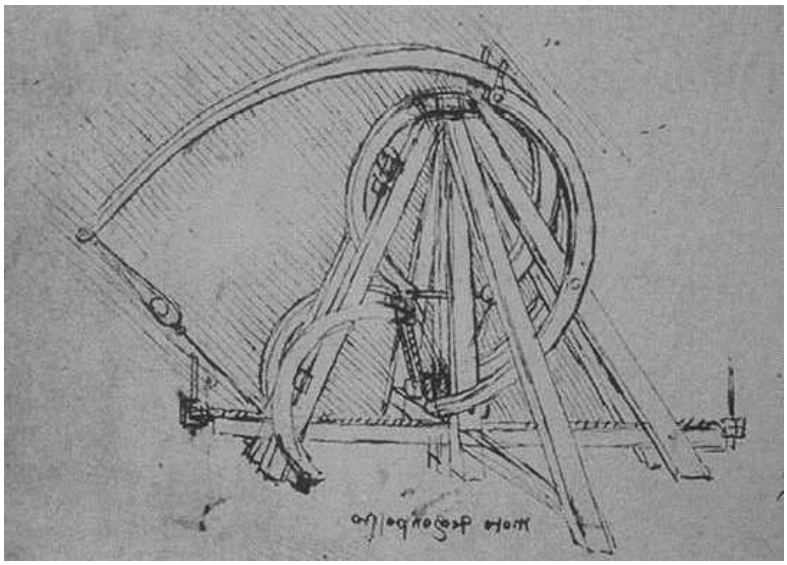


Figure 1: Leonardo's 'ballista'

Student:
A.J.Brussee
Oostersingel 9a
9713 EW Groningen
StudNr: s0965596
e-mail: alex@ai.rug.nl

Internal coordinator:
Prof. Dr. L.B.R.Schomaker
Rijksuniversiteit Groningen
Grote Kruisstraat 2/1
9712 TS Groningen

Artificial Intelligence
Rijksuniversiteit Groningen
November 15, 2004

Abstract

This MSc thesis describes biomechanically inspired models for ego-motion trajectory formation in robotics. The first model uses direct muscle control as a model for ballistic trajectory formation in ego-motion. The second model is based on optimality constraints on known limb movement found in nature [23, 41]. In other words, two models for jump and catch in 2D are introduced. The notion of an internal motor command model for trajectory formation is based upon findings in humans and animals [35, 26, 41].

Traditional AI relies on reactive models. In such models, trajectory formation is heavily dependent on its surrounding environment [42, 37]. Biomechanical systems often possess many reactive components, but these components offer no complete explanation for their trajectory formation. To explain these differences between theory and reality, the two models investigate possible explanations that may give us a better understanding of the rules underlying trajectory formation in biomechanical systems.

The two models differ in their perspective, but both try to relate findings in biomechanical systems to these rules. The direct muscle control model models the human muscle and emulates neural motor commands to form a trajectory. The second model develops a theory using constraints and rules that simulates empirically found trajectory formation.

The trajectory from the direct muscle control model is transformed to steer a differentially controlled robot. This path is then fitted to the trajectory generated with the empirical model. The resulting trajectory can thus be used in pro-active robotic behaviors.

The model was tested for relevant trajectories. It generated good results for trajectories with a low curvature. A more complicated muscle model would lead to even better results.

Contents

1 Introduction

This MSc thesis describes two models for motion trajectory formation for a differentially controlled robot. The idea is based on ethological evidence of such systems in nature and by functional constraints from mechanics. This chapter picks up an historical viewpoint toward the limitations of other models commonly used in AI and explains the benefits of the assumption of a more complex approach to modeling biomechanical systems. A chronological description of AI is used to gain insight in the various problems that are associated with autonomous systems. At the end of this chapter the requirements for the model explained in this MSc thesis will be introduced.

1.1 Approaches to trajectory formation

Artificial Intelligence became a distinct research field in the 50's. Hallmark was the Dartmouth summer research conference held in August 1955. McCarty et al. had a proposal for the Darthmouth summer research project on artificial intelligence, which stated the following in the first line:

> *"We propose that a 2 month, 10 man study of artificial intelligence be carried out during the summer of 1956 at Dartmouth College in Hanover, New Hampshire. The study is to proceed on the basis of the conjecture that every aspect of learning or any other feature of intelligence can in principle be so precisely described that a machine can be made to simulate it."*

Figure 2: shakey, controlled by the symbolic program 'STRIPS'.

1.1.1 Symbolism

At first, robot control focused on symbolic representations that could 'get the job done'. STRIPS [14] is an example of a symbolic language that uses logical rules to determine the robot's next action. This type of approach fits in a paradigm called symbolism. Symbolism states that every computation, including brain function, is based upon symbolic representation. Two powerfull statements stand ground for the symbolic paradigm.

The first statement is the Church-Turing thesis, which states that the Turing machine [4] is capable of computing every computable function. The Turing machine is in essence an abstract computing device. The state of the machine, the data it reads, and its instruction set determine its function.

The second statement is the next hypothesis about the human mind:

- Language perceived is symbolic.
- It reflects a state of mind.
- Deliberations can be followed using language.
- Human thinking is symbolic.

If humans used symbols, a computer may be able to simulate a human being. To test this, Turing developed the 'Turing test', in which a test subject communicates through a console with another human, or a computer. If the test subject can be fooled to be communicating to a human while there is a computer at the other end, the test is a success. Autonomous control was implemented by modeling behavior as symbolic processing. Advantages to this approach are the following [25]:

- The resulting model is a working computer model and the symbolic representation is a direct reflection of the human ideas about cognitive function.
- Basic processing steps are made explicit.
- The modeling process clarifies functionality that was previously related to functionality implicitly.

The symbolic paradigm is focused on problemsolving. Search algorithms like A*, bi-directional search, simulated annealing were invented [40]. Reasoning languages like STRIPS and analytical languages using λ-calculus, based on symbolic representations and using logic, were constructed. After all these inventions, the Turing test remained an open challenge and robots performed no 'real' behavior. Searle counterpointed the Turing test with his famous 'Chinese room experiment', stating that the symbols that could be used in the Turing test were not grounded in the real world, and thus had no meaning. This philosophical debate is still in discussion. In spite of its early success it did not take long before the disadvantages of the symbolic approach became apparent [25]:

- Symbolic processing can be completely different from a biological neural architecture. It follows that a symbolic processing step may be absent or faulted when compared with a biological system.
- The stress on symbolic processing interferes with the control of continuous variables.
- A symbolic architecture has 'machine-like' properties because of its modular nature. Dealing with this by adding stochastic variation to the output of the symbolic modules that make up the symbolic architecture is not enough.

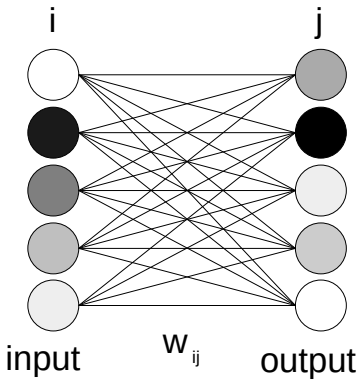
These disadvantages would eventually lead to new ideas about human cognition. This led to the revival of connectionism.

1.1.2 Connectionism

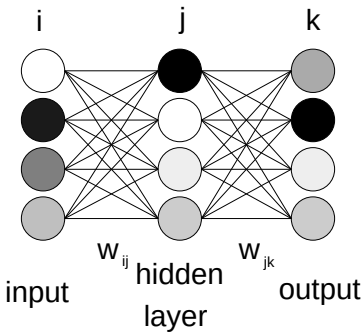
Neuroscience, psychology and ethology inspired artificial intelligence. From neuroscience came several neural models, which aimed to simulate the brain. One of the first models came from McCulloch & Pitts in 1943 [39], who assumed the neuron to be a TLU (Threshold Logical Unit). After that, Hebb introduced the Hebbian learning rule [12], which enabled a neural network to learn. Hebb defended the notion of an 'autonomous central process', that intervenes in the S-R (stimulus response) pair. This means neural information processing would mean more than reflexes alone.

Another important notion from the early days was the 'perceptron' [16], in which Rosenblatt elaborated on Hebb's model, introducing the LTU (Linear Threshold Unit) used in a general model called the 'perceptron'. An important biological neural model was that of Hodgkin and Huxley [3]. Their model showed how electric circuits can correctly represent a neuron's electrochemical properties.

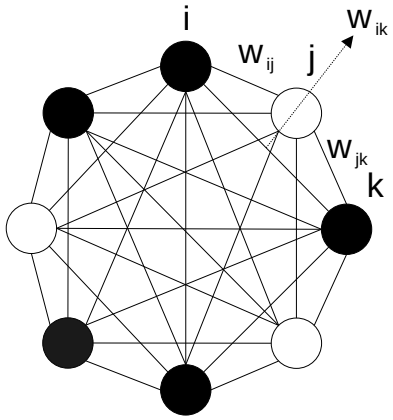
When the drawbacks of symbolism became apparent, connectionism revived. The 'machine-like' properties of symbolic models were replaced with simulations of biological neural architectures. Neural architectures represent input-output mechanisms with the neuron as basic element. Different techniques of representing a neural network, inspired by biological research as well as by earlier findings have led to a wide variety of neural networks. Their models all use connections between simulated neural patterns. The difference between the models concentrates on the connectivity of their basic units, the neuron. This means that there are different rules for the connections, and that the global structural properties a neural network vary. Three examples of connectionist neural networks can be seen in Figure 3.



(a) Typical double layer neural network architecuture. There are three components, the layer of input cells, a bundle of connections and a layer of output cells. Weights W_{ij} indicate the connection strengths between the different input and output cells. This connection determines the output strength of the output cells.



(b) Multi layered network using the same architecture, with one hidden layer, j. reprinted from [38],p.44 Next to the extra layer, Rumelhart, Hinton and Williams (1986) [10] introduced the error back propagation rule to train these networks.



(c) Hopfield network consisting of a single layer of fully interconnected neuron cells. The neuron cells can be active or non-active, mostly represented by the values $\{1, -1\}$. The network is created by supplying an input pattern vector. The network is used to restore distorted patterns into the proper learned patterns. That is why it is called an associative network.

Figure 3: Three types of connectionist neural networks

1.1.3 Behavior-based robotics

After 20 years of research, it became apparent that something more than symbolism and connectionism was needed to solve the problems associated with autonomous systems. This impasse caused researchers to re-evaluate their findings. Specialized areas of research made place for Behavioral-based robotics, a simplified approach that gave new insight into many problems that still existed in robotics [42, 37].

In the 80's, when behavior-based robotics came to life, most robot control programs had no direct coupling of sensing and acting. Deliberative control was considered crucial for autonomous intelligence. Behavior-based robotics showed that behavioral complexity can emerge in absence of deliberative control. Figure 4 shows contrasting points between deliberative and reactive design ([38], p. 20). Note that reactive control lacks the predictive capabilities which present in deliberative control. Behavior-based robotics took animals (ethology) as the basis for artificial intelligence.

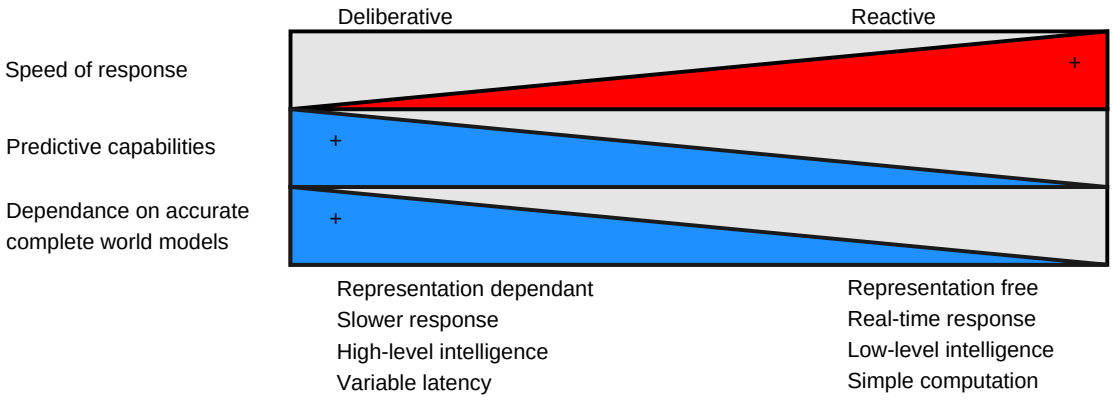


Figure 4: Differences between deliberative and reactive control. The darker parts indicated an increase of quantity. Note that reactive control lacks the predictive capacities which are present in deliberative control.

Ethology is the study of animal behavior in its natural environment. This alternation in thinking about artificial intelligence was a reaction to the failure of planning systems used in robotics. It appeared that reactive behavior is faster and more efficient. The neuroscientific field shifted back to the connectionist approach, and the planning mechanism moved aside for an ethological view. The animal behavior modelled by reactive control has three important classes [38]:

- Reflexive behavior. Involuntary responses triggered by certain environmental stimuli. Two well known examples are the moro-reflex in infants, and the knee-jerk reflex.
- Taxes. Behavioral responses that orientate an animal toward or away from a stimulus. Examples are insects circling around a light in the evening, the scent of a predator scaring away prey.
- Fixed-action patterns. Response patterns triggered by a stimulus differ from reflexive behavior by the fact that fixed-action patterns persist longer than the stimulus itself. An example of a fixed-action pattern is the pattern of aggressive (agonistic) display from the 'betta splendens' (male siamese fighting fish) in reaction to complex visual stimuli.

Grey Walter's 'tortoise'

Grey Walter was a respected neurophysiologist, who pioneered in research on EEG. He invented the first EEG brain topography machine. In the late 40's he carried out research on mobile autonomous robots, trying to model brain function. He studied the role of simple reflexive behavior and came up with several robots, called 'tortoises' (Figure 5), 'Elsie' and 'Elmer' [19]. The robots showed that simple reflexive behavior could be the basis for more complex emergent behavior. Grey Walter's ideas influenced the science of cybernetics, which was starting to emerge by that time.

Figure 5: Grey Walter's 'tortoise', using behavior-based navigation.

Braitenberg's Vehicles

Valentino Braitenberg revived Grey Walter's idea's three decades later. Braitenberg is a well known professor specialized in biological cybernetics. Braitenberg's fame is due to the purely hypothetical invention of his vehicles. Braitenberg describes 14 creatures [42]. The first and simplest creature is only able to drive forward. The other creatures are given increasingly complex behavior. With these thought experiments, Braitenberg tries to show the simplicity that can underly complex behavior. This is Braitenberg's law of uphill analysis and downhill invention.This means that one can easily make complex behavior. On the other hand, analysis of such a system without prior knowledge about its programming will become increasingly difficult with little added complexity. After Braitenberg's first, simple reactive examples, he continues to elaborate on his vehicles, speculating about more complex behavior like planning, learning, vision and more.

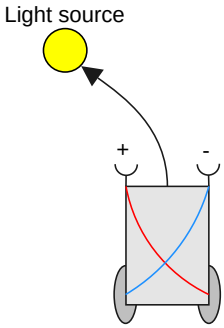


Figure 6: Braitenberg's phototropic vehicle '2b', that navigates toward light sources.

Brooks 'subsumption architecture'

A second example of Behavior-based robotics is Brooks' [37] work in the field of AI. His view on AI was controversial because it collided with ideas from classical AI. In his work, he criticized connectionism and symbolism, calling them GOFAI, Good Old Fashioned AI, a term that was introduced by Haugeland [22]. Instead of the traditional 'sense-plan-act' architecture, he invented the 'subsumption architecture'. A stimulus would elicit a hardwired response. The more responses hardwired in the robot, the more chances it has to be successful. This close-coupling of sensors and actuators creates less delay between sensing and acting, thus preventing overshoot (Figure 7), but creates a new problem. The problem is that a robot using this architecture has no intelligent (deliberative) behavior. Brooks' argument is that intelligent behavior is not created by deliberative control, but by decomposition of complex behavior into simple behavior. The 'subsumption architecture' uses this principle. In this architecture allows simple reactive behaviors to be active in parallel, competing for a controlling role.

Problems with purely reactive systems
Many robots in today's world operate in static, known environments. The goal of AI is to make an autonomous system that is capable of interacting with an unknown environment. In order to reach this goal, deliberative and reactive behavior have to be balanced. Deliberative behavior is planned behavior. Reactive behavior is direct, working like a reflex. Braitenberg's vehicles and Brooks' 'subsumption architecture' are examples of reactive behavior ('sense-act'). The problem with these approaches is, that there is no planning, and thus no 'intelligence'. A more practical but serious problem is that reactive control breaks when there is a time delay in the sense-act loop. In this case, a corrective action is always too late and this can cause 'hunting', as displayed in Figure 7. This is untenable in biological systems, since many signal-propagation delays exist in the neural system.

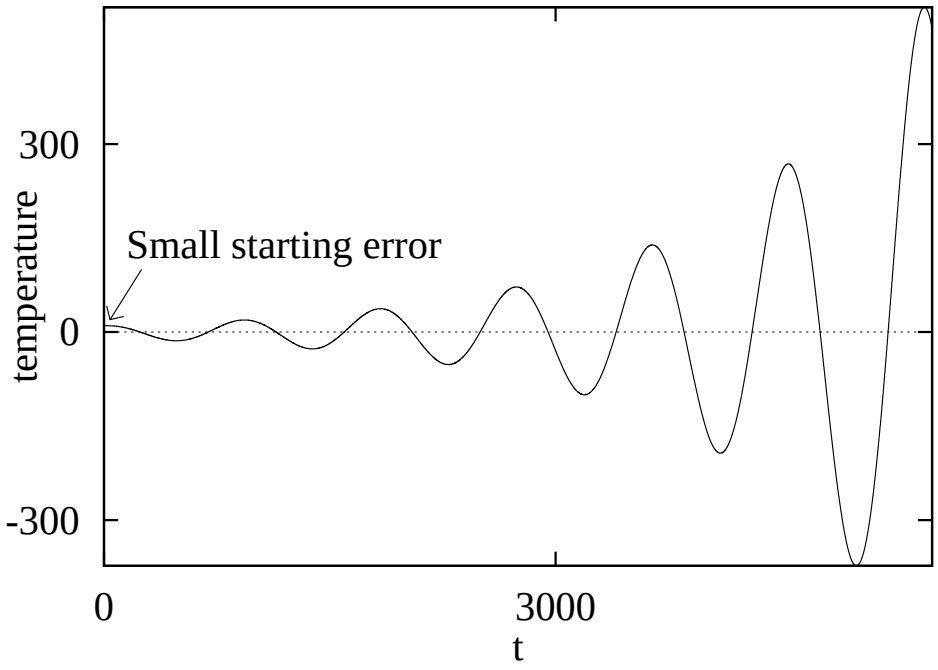


Figure 7: The hunting effect in a thermostat with a delay. The desired temperature is 0. The small starting error is corrected, but because the regulator reacts with a time delay (lag), the correction causes the temperature to overshoot below zero. In reaction, the regulator reacts by counteracting, but overshoots again because of the lag. The overshoot increases because every time the regulator tries to correct the problem, it adds more energy in the wrong direction than in the right direction. The energy buildup from the exponential growth of the overshoot maxima and minima will eventually cause a crash or satiate the systems output.

1.2 Pro-active behavior

The rationale behind behavior-based robotics is the assumption that biological evolution in nature naturally evolved mechanisms capable to overcome the caveats of the physical world. After ten years of research, behavior-based robotics has not succeeded in outperforming other paradigms researching autonomous systems and their control. Behavior-based robotics assumes that complex behavior is an emergent property that springs from the combination of simple behaviors. Behavior-based robotics uses reactive components (trophism) and rejects any form of interference between stimulus and response. In this way, the mechanism that controls the robot is 'representation free'. This circumvents the debate from symbolism regarding the grounding of symbols. Besides, it is biologically inspired.

Even though Braitenberg's creatures can follow complex and mathematically untracable paths, they do not show complex behavior. Even the combination of several trophisms do not cause more complex behavior to emerge. It seems plausible that complex behavior needs some kind of internalized motor command model for trajectory formation. Experiments [43] show that reactive behavior is often accompanied by more complex reactions.This observation implies the existence of an internalized motor command model in biological systems. When this is the case, a system can plan its behavior in advance and it will become pro-active. Note that the use of an internalized motor command model still leaves room for a biological perspective, but does allow interaction between stimulus and response.

Behavior-based robotics puts insect behavior central in its debate to show that simple behavior is the basis for more complex behavior. This can be counterpointed by the observation that an animal with considerable less neural complexity, the jumping spider, uses pro-active behavior to jump toward a potential prey.

The jumping spider, *Araneae, Salticidae*

To illustrate the existence of pro-active trajectory planning in nature, the jumping spider is a good example. Experiments with humans [35] and primates [13] have shown us that it is likely that internalized models for motor command exist and that reafference is compared with an efference copy (see §3.7 for more information). Insect behavior is less complex than that of a primate. This is why observing insect behavior is more likely to provide us with a working model for trajectory formation in animals.

Even though there are several differences between a spider's muscles and muscles from vertebrates, they share some basic principles that are the same. All animal muscles are characterised by:

- Visco-elastic properties of muscle tissue.
- Temporal delay in neural transmission.
- Temporal delay in neuromechanic transmission.

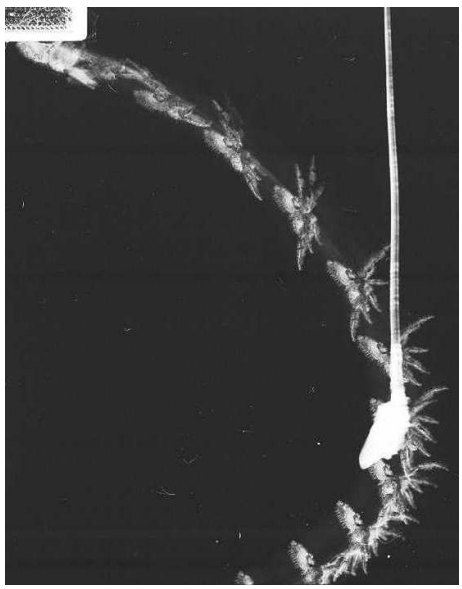


Figure 8: *Phidippus princeps* jumping to prey upside down

Figure 9: *Phidippus pulcherrimus* taking attack position

The jumping spider uses its trajectory planner to estimate jump velocity and direction to jump toward its target. The trajectory planner enables the spider to adapt its jump to varying start-target situations. The jumping distance from starting position to target location can be more than 10 cm [29]. Evidence that the spider uses a planning system is shown in Figure 9, where the spider positioned itself on the pole before jumping. The repositioning movement on the pole shows that the spider successfully uses a ballistic trajectory planner capable of using an abstract representation of the directional inclination needed to use gravity as a variable.

> *By moving prey toward these spiders from different directions, it is easy to demonstrate that direction relative to gravity is a key determinant of the maximum distance at which the spider will launch an attack.*
>
> David E.Hill.[5]

Hill's [5] experiment reveals that the spider uses different takeoff-speeds and directions in response to various target locations.The spider will even jump correctly from an upside down position (Figure 8). Hill made a small program which calculates the spiders offset speed and angle for a jump.

Another study by Forster et al [27] looked at the path a jumping spider makes toward its target. When a jumping spider spots a possible target, it will stalk its prey in a crouching movement. If the spider is close enough, it will attack, using its jumping capability. The stalking path can be modeled by the following formula:

$$\overrightarrow{Travel} = \epsilon \frac{1}{\delta+1} \cos(\phi)(\frac{1}{\delta+1}\overrightarrow{Prey}[t] + \frac{\delta}{\delta+1}\overrightarrow{Prey}[t+\delta])$$

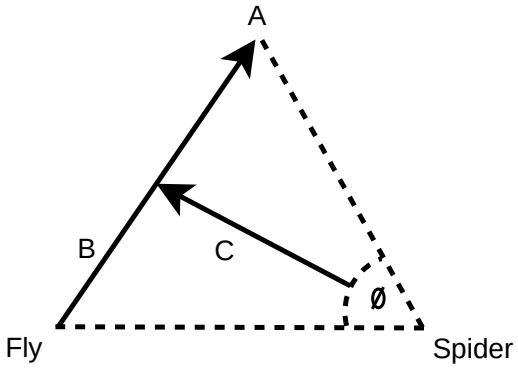


Figure 10: Prey position prediction by the jumping spider. The spider estimates the prey position and adjusts its travelling direction pro-actively to intercept. B is the travel vector of the fly. Point A indicates the position of the fly at $[t + \delta]$. C is the actual travel direction the spider takes to intercept the fly. The angle ϕ incdicates the angular position change from the fly in δ time.

Here, ϵ is the time interval between computations, ϕ is the angle between the target position and the target position in the future at $[t + \delta]$ in respect to the spider (see Figure 10). Furthermore, δ is the time delay used to estimate the target's real position. A simulation program was used to show that this formula correctly predicts the spider's path. In order to make the path fit, the delay time, δ, has to be altered.

The experiments show us the spider uses trajectory planning to jump and a re-afferent signal to stalk its prey. Both studies suggest that there is a relationship between stalking and getting the right position to jump. The jumping spider illustrates the use of an internalized trajectory planner in connection to sensory input. The different paths Forster found in his experiment suggest that the spider may be able to adjust the delay time in order to obtain another path.

1.3 Models for pro-active behavior

There are several models combining reactive behavior with a more complex set of underlying rules. We already saw Brooks' [37] subsumption architecture, that tried to explain complex behavior as an emergent property of simple reactive behavior. The potential field method has combined the behavior-based approach with experimental findings in the frog [13] to come up with a model that emulates complex behavior. The dynamic window approach [8] took a look at the kinematics of the robot and modeled this in a pro-active version of behavior-based obstacle avoidance.

Potential field method
The potential field method [32] is often used for robot control. This approach uses a two dimensional Cartesian grid to represent the robot's surroundings, this grid is divided into cells. Each cell has a steering vector. This vector points away from obstacles, and toward attractive elements (target location) (see fig 11). From this idea grew the 'Vector Field Histogram' [21] which uses a one dimensional polar form which is a directional representation of the two dimensional grid. Both types of robot navigation are reactive. Bizzi [13] showed how the leg of a frog can be controlled. He electrically stimulated the spinal cord at the place where the leg sends a motor command message to the leg. He found out that there was a 'convergent force-field' (CFF) which can be modeled by a vector field with an equilibrium point at which the force vectors converged. This point was found out to be the steady state of the limbwhere no power is applied to the muscles. The other points in the CFF represent a predefined muscle command structure. This shows that a frog has a predefined reaction for muscle movement. The alleged existence of this template of movement plays a significant role in the visco-elastic muscle model. More evidence supporting the model in this direction is described in paragraph §3.6.

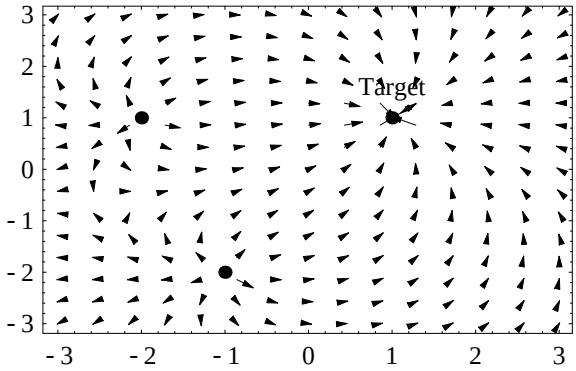


Figure 11: a vector field histogram with two repulsive objects and one attracting target. The arrows are a summation of the attractive and repulsive laws that operate in the area.

Even though the potential field method is a reactive model, it depends on an internalized motor command model that represents the grid for motor control. The CFF found in the frog supports these assumptions. The existence of the CFF also shows us how simple reactive behavior in biological systems may be steered by an internalized motor command model. We can now conclude that the assumption that behavior-based robotics made that there should be no interference between stimulus and response is not always true in biological systems. There has to be some kind of neural architecture that represents such a CFF. The only problem is that the symbols used in the model may incorrectly represent its biological functionality. This can be avoided by researching the constraints which determine the behavior that is to be modelled.

Dynamic Window Approach
The dynamic window approach [8] is an engineering approach to autonomous robot control that makes use of the kinematics of the robot to choose its next action. The robot's place is described by the triplet $<x, y, \theta>$. The robots dynamic configuration is described by its speed and rotational speed, $<v(t), \omega(t)>$. Together with the robots possible acceleration or deceleration, an effector space can be created. The robot can now compute possible future positions. It then chooses the preferred position taking into account target, clearance and velocity. This system's advantage is that it allows the robot to travel at a high speed and is very accurate. Obstacles are avoided in a pro-active manner, since the dynamic window approach adequately handles problems concerning lag etc. A disadvantage is that the approach is still reactive. It needs to compute its next action each time step.

The values of the effector space are determined by several trophism, like obstacle clearance and target location. Even though this idea originates from behavior-based robotics, it uses many artificial assumptions that are not backed up by ethological evidence.

1.4 Proposed model

In the design of a movement control mechanism there are two important questions:

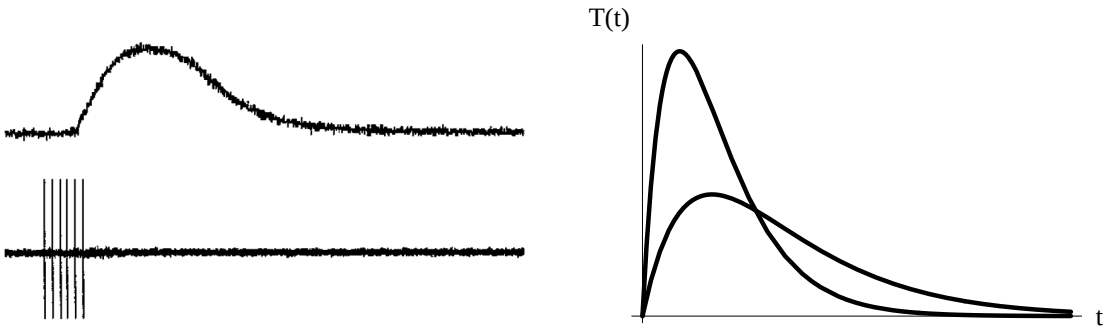
- What type of representation is used? There is for example the choice between a symbolic or a connectionist representation.
- What type of approach is used in the movement control model? Examples are behavior-based control, an engineering approach or Brooks' subsumption architecture.

Note that there is not always a clear distinction between representation and approach.

The reactive approach to movement control has some serious flaws:

- Movement control using a reactive approach is only a small subset of all possible control mechanisms.
- In a dynamic world where movement strategies are crucial (fight, flight, prey catching), reactive control alone would always be too slow due to unavoidacble delays:
  - Processing delays (computation time).
  - (neural) transmission delays.
  - Inertia, neuromechanic delay.
- Reactive control is behavior-based, which implies that it mimics characteristics from animal behavior. The electromotors used in reactive control differ from muscles. The basic operations of technical electromotors are analogue, fluent and smooth. Controlling a muscle on the other hand , requires non-linear mechanics for fast, pro-active behavior.

The fact that a muscle has other basic operations than those from technical electromotors, can be seen in Figures 12(a) and 12(b). Instead of a linear relation like that of a electromotor, the basic response of a muscle is a curved tension buildup and decay. This response will only be elicited when the input of the motorneuron connected to the muscle has exceeded a threshold that will activate the muscle. Figure 12(a) displays the response of a muscle from a lobster, Figure 12(b) displays a simulated response that in used in the model from this MSc thesis.



(a) Twitch response of an isotonic contraction of a dorsal dilate muscle in the lobster in response to a 0.25 sec 30 Hz stimulus

(b) Two 'twitch' responses from equation (20), using different time constants, represented by ω_1, ω_2 in the formula.

Figure 12: Real (Figure 12(a)) and simulated (Figure 12(b)) twitch responses

Requirements for movement control
There are several basic requirements important in movement control:

1. Pro-active control for dynamic contexts (like prey catching).
2. Efficient speed control, especially in case of linear trajectories.
3. Capacity of planned distance-coverage.
4. Obstacle avoidance capability using curvilinear trajectories.

The movement model described in this thesis is expected to be able to cope with the following behaviors:

- Generating control for a grabbing movement in proximal ego space. Proximal ego space is the space where objects can be reached without ego motion (Figure 13, see also §6.3).
- Jump and catch trajectory estimation.
- Long distance ego motion.

In Figure 13 it is displayed what type of spatial movements there are. They range from the private space to the constant velocity planning space. Grabbing is a type of movement that is in the private space, jump and catch is in the monphasic ego-motion space and long distance ego motion is in the constant velocity planning space.

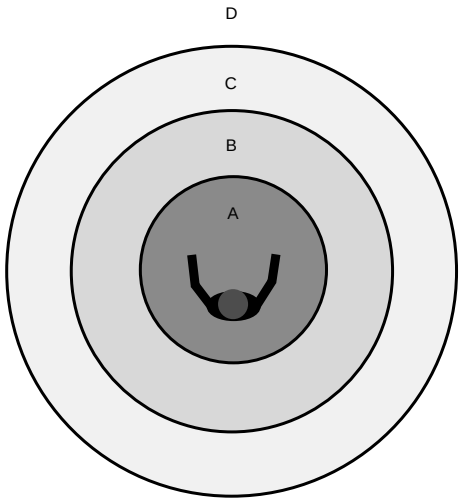


Figure 13: General model for types of movement related to targeted movements.
A. The private space, represents the space in which targets can be reached without ego-motion or torso movement.
B. The partial ego-motion space, describes the space that requires movement of the torso in order to reach for a target.
C. The monophasic ego-motion space, requires a jump or an ego-motion that has a monophasic speed profile.
D. The constant velocity planning space, where the distance to be travelled is becoming too large to jump, or too large to have a bell-shaped speed profile. It may now be now useful to obtain a constant travelling velocity.

When we combine the requirements and the desired behaviors, there are two important issues that determine the design of the motion control model described in this MSc thesis. The jump and catch behavior is based on animal movement. This is why a movement control model with simulated neural motor control type actuators is needed. When we want our model to be able to generate grabbing movements and long distance ego motion, biomimetic trajectory generation is needed. This can be achieved by a biomimetic model capable of representing the desired trajectories. When an obstacle must be avoided, it can be represented by a via point in a trajectory (see §14 for more information) Combination of the neural motor control and the required biomimetic properties makes up the resulting motion control model described in this thesis.

Goals
We now know that we need an actuator simulation based on neural muscle motor control and a biomimetic model describing a desired trajectory. To achieve this, the next goals are to be achieved:

1. Identify biomimetic modes of motor control beyond reactive servo control.
2. Design a model capable of generating a trajectory with simulated muscle properties, using simple muscle-like effectors.
3. Design a model capable to generate trajectories based on biomimetic properties.
4. Coupling the first model with the second model to be able to use the muscle-like effectors to generate a trajecory with biomimetic properties.

The fourth goal embodies the first three goals. We will show that, using a visco-elasctic muscle model for single-jointed movement, a range of required trajectories can be generated. For trajectories which are subject to constraints in terms of speed and shape, a general mechanism for optimization is needed in order to properly estimate movement control parameters. Fortunately, such a optimization mechanism has been proposed. This model encorporates observed contraints from experiments with human and primate limb movement [41]. A dominating constraint is the 'minimum jerk' criterion. This criterion is a measure for the energy required to move an object over a path. In essence, 'minimum jerk' is the change of acceleration. When the measure of 'jerk' is low for a certain trajectory, this means that an object travelling this path uses less energy. Sudden changes in speed will increase the 'jerk' measure. In this case it will cost more energy to traverse the path. Besides, the sudden changes in speed can cause other negative effects, like an increased chance of collision.

Hogan & Flash [41] showed that trajectories from observed movements closely resembled trajectories generated with this 'minimum jerk' constraint. This optimization mechanism is a biomimetic model since it mimics natural movement. In order to use this model for robot control, a general movement control mechanism is needed. This model will be based on a muscle model. In this way, the movement control mechanism is beyond reactive servo control. Given these two models, one bottom-up (muscle-based) and one top-down (biophysics-based, using the 'minimum jerk' criterion), we can now realize the estimation of muscle movement control parameters, using the biomimetic model as the guiding optimization principle (i.e., the 'trainer').

The generalised visco-elastic model and the biomimetic model
The described motion generation model model is the combination of two trajectory formation models. The first model is a biomechanical model that simulates the visco-elastic properties of the muscle. The second model is based on constraints found in known human and primate limb movement (see §2.2 for more information). The motor command model is pro-active because it plans the trajectory in advance. The model plans a trajectory which can be executed without further computation or sensory intervention. The processing time gained this way is left to other processes. When the controlled robot is in immediate danger, more 'primitive' reflexive behavior without much computational load can take control. This behavior is equipped to deal with sensory lag by making a prediction of its future position, a method that is often used in nature, using a so called 'efference copy' [43, 27, 29].

The visco-elastic model is based on biomechanics. The model simulates the movement produced by a muscle. In order to translate muscle activity to path generation for a differentially controlled robot, two muscle activity patterns represent the left- and rightwheel speed of a robot, resulting in x,y positions of a robot. The robot that can be steered this way is directionally controlled. The left and right wheel accelerate according to the 'muscle's' activity.

The generated path is then fitted to a fifth order bezier spline, which is optimized for minimum jerk (see §5.2 for more information). Bezier splines are widely accepted as a model for biomimetic movement [23]. The combination of the two connects empirical findings in biomimetics with empirical findings in muscle activation [5]. An implementation of the model is made in C++ and in Java. The process of trajectory formation consists out of three steps.

- Identification of the via point and the target location.
- Trajectory formation using minimum jerk.
- Approximating a trajectory that resembles the minimum jerk trajectory with the visco-elastic muscle model.

Requirement 3, the capacity of planned distance-coverage, requires the introduction of a via point. This requirement introduces curvilinear trajectories that necessitates the introduction of a special construct which allows the trajectory controller to deviate from a straight line. This construct is called a 'via point'. A reason to deviate from a straight path would be an obstacle that must be avoided. How this via point affects the trajectory can be seen in Figure 14. Note that the speed profiles are also displayed. We will see that the speed profile shape is a typical measure for the 'minimum jerk' criterion (see §3.9, §6.3 for more information).

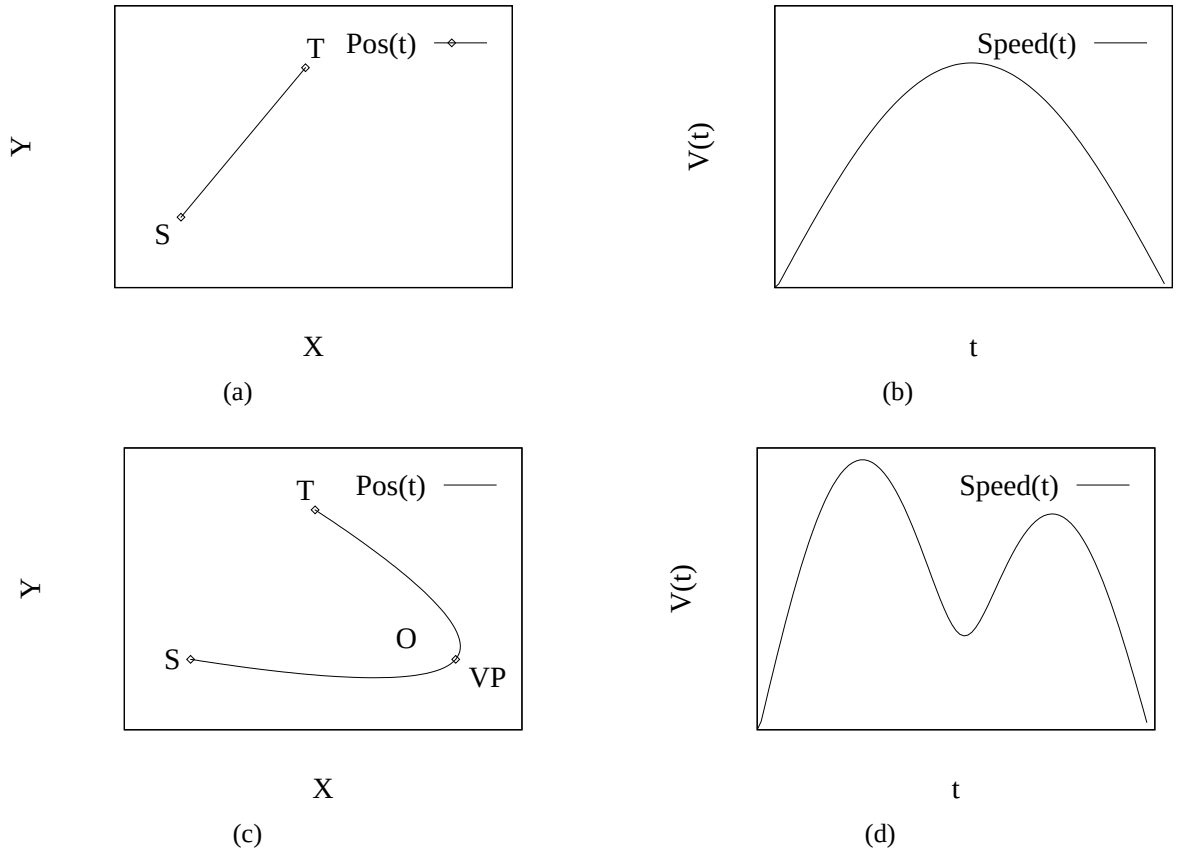


Figure 14: Example trajectories, where S is the starting point, T is the target point, VP is a via-point, O is an obstacle and Pos(t) is the robot's position at time t. Figure 14(a) is a straight trajectory. In Figure 14(c) a viapoint that goes around an obstacle (O) is introduced to this trajectory. In the figures on the right side one can see the effect on the speedprofiles on the introduction of the via-point. Figures 14(b) and 14(d) belong to the trajectories in Figures 14(a) and 14(c) respectively.

The visco-elastic muscle model is simple and does not accurately incorporate all features biomechanical systems possess. The functionality of the model has been limited to show that a trajectory formation that uses the minimum jerk principle is a good method to reduce the many to one problem that occurs with a multiple degrees of freedom system. The body is a multiple degrees of freedom system, where every degree of freedom is a joint that connects two parts of the body that can move independently. The biomimetic model reduces the possible types of movement by stating that they must obey the minimum energy constraint, expressed as minimum jerk. The result is that the visco-elastic muscle model represents the possible movements the system can make, and these movements are constrained by the minimum jerk criterion. When the visco-elastic muscle model is expanded to contain more degrees of freedom, its restriction to the minimum jerk criterion ensures that ambiguities in possible movements are excluded.

A remaining problem is to generate a trajectory with the visco-elastic muscle model that represents the trajectory generated with the minimum jerk criterion. To deal with the fact that the problem partly remains, the BFGS (see §5.1 for more information) algorithm is introduced as a representation of the computation normally executed by the neural motor control in animals. This algorithm is a gradient descent method of optimization. The BM is optimized for minimum jerk. We want the VM to be energy efficient as well. This is why the BFGS algorithm is used to find the right variables to fit the VM to the BM.

2 Coupling the biomechanic and the biomimetic models.

2.1 Ideas behind the trajectory formation models

The fundamental idea behind the trajectory formation model is to gain more insight in motion planning and motion in general. Biomimetics has received a great deal of attention from the film industry, medical science, biology, AI and other (scientific) paradigms. These studies have mainly concentrated on imitating human or animal movement. Even though the models are accurate, less attention is payed to the possible role of an internalized motor command model.

Human and primate limb movement are complex and not easy to simulate. The many degrees of freedom make it hard to come up with a model that predicts what movement will be generated for an arbitrary task. Several studies have successfully concluded that certain constraints operate on the effector space of motion generation (see §2.2 for more information). In motion generation, one of the most apparent types of constraints is the minimum jerk criterion. Jerk is the derivative of acceleration, which gives an insight in the variation of acceleration. If the overall change of acceleration is low, the system that is moving with this speed profile uses less energy and avoids 'jerky' movement which can result in damaging speed 'jolts'. These 'jolts' can damage the moving system's components, and it is less prepared for a sudden change in movement.

This MSc thesis looks at ballistic trajectory formation because of its 'aim and shoot' character. It assumes that a biological system avoids the lagging problem by the use of an open loop system (see §3.5 for more information) that generates a preplanned movement scheme that is constrained by the minimum jerk criterion. This is where the coupling between the visco-elastic muscle model and the biomimetic motion generation model comes into play. A biomimetic trajectory generator that simulates a general limb movement[23] is optimized for the minimum jerk constraint. Next, a visco-elastic muscle model is used to generate plausible 'twitch like' speed profiles. These profiles are transformed for a differentially controlled robot and fitted on the generated biomimetic trajectories. The combination of the two trajectory formation models couples the visco-elastic muscle model with the biomimetic model.

2.2 Human arm movement

The equilibrium point hypothesis (EPH) has been the dominant theory for movement and posture control over the past 30 years [34]. The EPH tries to explain movement and posture control by stating that the brain controls movements by adjusting equilibrium points. These equilibrium points are states in which the mechanical framework of the body can be. To move a limb toward a target position, the stretch and contraction of appropriate muscles is adjusted so that the limb will settle in a new equilibrium point, which brings it at the targeted position.

Hogan & Flash [41] wrote a mathematical model describing arm movement. They discovered that the trajectories smoothness could be described with the minimum jerk criterion. Their study was in support of the EPH [7], even though the smoothness description in terms of minimum jerk is unrelated to the type of neural control implied by the EPH. Later research showed that the EPH may have serious flaws. The first mayor drawback was the omittance of the dynamic behavior of muscles. This was incorporated in the EPH by Gribble et al[20], 1998. Even with this adjustment, there are some experiments [33] that could not be explained by the EPH.

This is why the biomimetic model concentrates on the findings from Hogan & Flash [41] and Hale [23] of mathematical approaches that describe movements. Whether or not the EPH is correct is also independent of the visco-elastic muscle model used to approximate these mathematical models.

The notion of the equilibrium points from the EPH concerns a complicated and accurate model of human arm movement, which would come into play when the visco-elastic muscle model would be more complex.

2.3 The visco-elastic muscle model for trajectory generation (VM)

The VM uses a biomechanical representation of a muscle to simulate a real one. There are many different types of motor command models that steer many different types of muscles in varying positions in biological systems. because it would be unfeasible to model all of muscle configurations, a simplified general muscle model is used. The same goes for the many different motor command neurons.

The model does not simulate any particular movement from an animal. However, the model closely adheres to findings from research in limb movement and the biomechanical nature of muscles. General characteristics from these findings are incorporated in the muscle model to find out if these characteristics naturally comply to the constraints (minimum jerk) of limb movement found in humans and primates. This would certainly provide us with evidence for the need of an internalized motor command model. It would also give us an insight in how constraints like energy preservation and 'safe movements' are 'built-in' (see §3.9 for more information) as physical properties of biological systems.

2.4 The biomimetic trajectory formation model (BM)

The first chapter showed us that symbolism and connectionism did not lead to the desired result. Behavior based robotics had some promising initial results, but the behavior it could simulate could not generate more complex behavior and had some serious flaws. The example from the jumping spider showed us that the complexity of behavior can not solely be dependent on simple reactive behavior. This is why the notion of an internalized motor command model was introduced. Because brain functions can not be reproduced or simulated with great precision, this model is obscured from the researcher. That is why the observed behavioral constraints that can be measured determine the BM.

Because we look at trajectory formation, it seems rational to investigate general properties of known limb movement. These properties are then used in the BM to generate a trajectory. The most important property of observed trajectories is the 'minimum jerk' criterion. The BM generates a trajectory based on this criterion. This trajectory is then used to fit the VM. In this way, a theorized model of empirical findings in limb movement is fitted to a mechanical model that represents a biological system.

The BM has the ability to generate any trajectory between any number of points. because of the simplifications that were made to the VM to gain an insight in the general characteristics of muscle mechanics, the BM deals with monophasic movement as the basis of more complex trajectory formation. A monophasic movement is characterised by a single speed buildup and decay (see Figure 14(b)), whereas a biphasic movement has two speed buildups (see Figure 14(d)).

3 The visco-elastic muscle model, the VM

3.1 Simplifications used in the model

The VM is the biomechanical part of the trajectory generating model that results from the combination of the VM and the BM. In §2.3 it was explained that the VM must be simplified to keep focus on the internalized motor command model. In this subsection it will be explained which simplifications are chosen and why.

General muscle characteristics
Generally speaking there are two kinds of muscles, those of insects those of other vertebrates. Since the minimum jerk hypothesis has been researched using the latter muscle type, the biomechanical properties of muscles from vertebrates will be used. These muscles can generally be classified into three types of muscle [9] (p.165-212):

- Type I, slow twitch, oxidative, slow fatiguation, makes up 50-55% of the muscle.
- Type IIa, fast twitch, oxidative, intermediate fatiguation, makes up 30-35% of the muscle.
- Type IIb, fast twitch, anaerobic, fast fatiguation, makes up 15% of the muscle.

These muscles are controlled by a motor unit. A motor unit is a motor neuron and all its muscle cells (fibers) that it innervates. The motor units control only one type of muscle. That is why there are three types of motor units, Type I, Type IIa and Type IIb. A motor unit follows an 'all-or-none' principle. This means that when a motor unit sends an impulse to its innervated muscle cells, either all or none of the muscle cells will contract. Motor units have different sizes. Large motor units cause more contraction and thus a greater force production in the muscle. The motor units comply to the so called 'size-principle'. Small motor units will respond to a small stimulation frequency, large ones will likewise respond to a greater stimulation. Once this stimulation increases, the motor units that are already active will produce increasingly greater muscle force outputs. In this way a muscle can be gradually controlled, and depending on the contraction speed needed, the distinct types of muscles can be activated.

The simplified model
Next to this, simplified explanation of muscle contraction there are many other factors that determine limb movement. Examples are the place of attachment of the muscle on the bone, the arrangement or pennation of the muscle fibers in a muscle, etc... When one wants to make an accurate model of a real body, this has to be taken into account.

We are interested to see if there is a link between physical constraints and the possible movements from a muscle. That is why the simplest form of muscle activation is used to look at the generated muscle activity. This form is a twitch response. A twitch is the weakest voluntary contraction that a muscle can make. Even though their underlying chemical process differs somewhat, the twitch response from the three types of muscles can be described using the same biomechanical muscle model.

Another reason that makes the twitch a good starting point to investigate an internalized motor command model is that its characteristics have the 'aim and shoot' character that we need. This ensures us that there is no stimulus-response neural interference with the preplanned muscle contraction, and it complies to the monophasic character used in the trajectories generated.

3.2 Biologically inspired motor command model

The VM is based on muscle movement in animals. The movement of the muscle is dependent of its state of contraction. This state is mainly influenced by three dimensions: force, velocity and length. The contraction is steered by motor neural activation. When a muscle is activated by a neural impulse, it contracts and relaxes in three stages (Figure 15):

- latent period - no change in length, this time is needed for chemical processes to 'ready' the muscle for contraction.
- contraction period - muscle becomes active and tension increases.
- relaxation period - muscle relaxes (tension decreases) and tends to return to its original length.

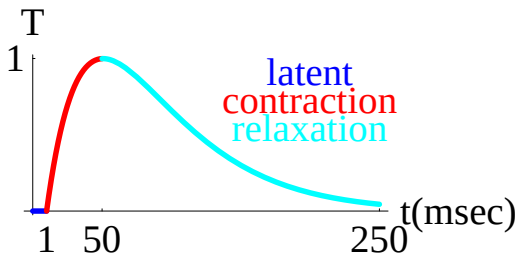


Figure 15: After the occurrence of a spike of the motor neuron in the spine, there are three stages of activation in the muscle: The latent period, during which the spike travels from the motor neuron through the motor axon toward the muscle fiber (1-2 msec). The contraction period, during which the filaments in the sarcomere slide past each other (10-100 msec).Relaxation period, during which the muscle tension decreases and the chemical balance before the twitch is restored (10-100 msec).

After a motor neuron has sent a pulse to a muscle, the muscle contraction will follow a path which can be modeled by a mass-spring system or another possible activity pattern that represents a muscle contraction model. This leads to a stretch curve as described in (see §3.4 and §3.4.2 for more information).

3.3 Biomechanical properties of the muscle

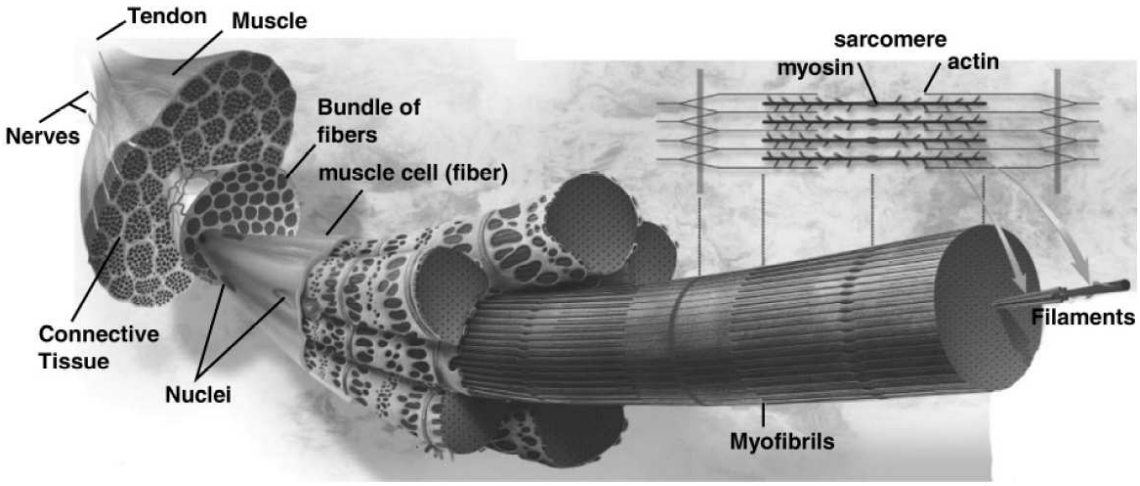


Figure 16: A muscle's interior (reprinted from [36]). The sarcomere (upper right corner) is the contractile element which causes the twitch.

A muscle is connected to the bone with a tendon. The tendon has elastic properties and connects the contractible parts of the muscle to the bone. The contractible parts consist of muscle's connective tissue. In this tissue one can find muscle fibers. These fibers are made up of tens of thousands of thread-like myofibrilla, which can contract, relax, and elongate. These myofibrilla are in turn made up of up to millions of bands laid end-to-end called sarcomeres. A sarcomere consist out of myofilaments. These myofilaments are made out of overlapping thick and thin filaments. These are made out of contractile proteins. These proteins are mostly actin and myosin.

When a muscle contracts, a neural stimulation is sent to the muscle fibers. This signals a calcium flow that causes the thick and thin filaments to slide across one another. The sarcomere shortens, and the muscle contracts. A muscle fiber cannot partly contract. This is why the twitch response is the basic building block to model general muscle activity.

Most muscles are positioned on the skeleton to form opposing groups. The muscles in such a group are called the agonist and the antagonist. When the agonist contracts, the antagonist stretches. The agonist is the muscle group that is intentionally contracted. An example of two muscles in such a group are the biceps and the triceps.

3.4 Hill's visco-elastic muscle model

3.4.1 Hill's quick-release experiment

Muscle mechanics have been studied since the early twenties last century. Several results from these studies are the basis for today's research. In 1938, A.V. Hill [5] pioneered this research area and postulated the visco-elastic muscle model (Figure 20). He performed an experiment to find out the viscoelastic properties of the muscle (Figure 17). Muscles have two kinds of force, active and passive [6]. Hill's experiment shows how a muscle can be modeled mathematically (Figure 20). In the experiment, a bar is used that pivots around a fixed point (see Figure 17). At the left end a weight is attached, at the right a muscle. The right end is held in its place by a catch mechanism that can be released at any moment.

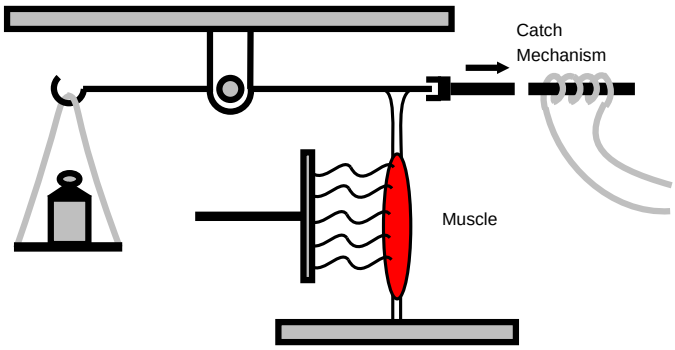


Figure 17: Experimental setup that A.V.Hill used to demonstrate the viscoelastic properties of a muscle. First, the muscle is electrically stimulated, generating a muscle tension T_0, then the catch is released. Now the mass pulls at the muscle with a force T. The empirical data, changes in muscle length and muscle force, can then be used to construct a viscoelastic model of the muscle.

Mechanics of passive muscle force

When released, the weight pulls at the muscle with a force T. The muscle is stimulated electrically, at maximal stimulation. When the muscle is stimulated without the catch released, a tension T_0 builds up in the muscle. T_0 is an isometric force that builds up in the muscle, called isometric because the muscle length can not change at this moment.

Now the catch is released. As a reaction, the muscle suddenly shortens, the force dropping from T to T_0. Call the shortening length X_2. This sudden drop of force and sudden change in length suggests something in the muscle acted as a spring. In the muscle model this is the series element, SE. Its stiffness is K_{SE}, made visible in Figure 20.

After this rapid change in force and length, a more gradual process follows. The muscle is still shortening, but much slower. The force does not change during this phase. At the first part of the reaction after the release, the muscle reacted immediately to the change in force caused by the release of the catch. The muscle now shortens gradually, and the force does not change. A mechanical analogy of this response is a shock absorber that acts on a spring. This is also known as a Voight body (see Figure 18(a)), in which a spring and a dash pot (shock absorber) are placed parallel. In the muscle model this is represented by the passive element, PE. Its length is represented by X_1, the dash pot's viscosity by B and the passive element's spring constant by K_{PE}.

We now have a model to represent the passive forces that govern the movement of a muscle. The passive part can be represented by a Zener model (Figure 18(c),18(d)). We can now investigate the passive properties of a muscle. Later on we will look at the active properties of a muscle, and complete the model by the addition of an active component. The resulting model is Hill's muscle model in Figure 20.

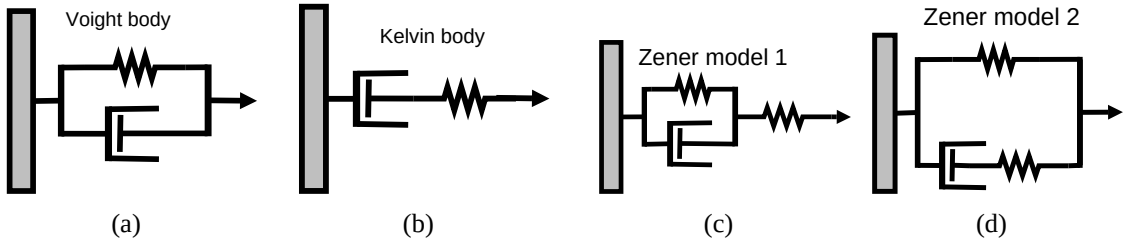


Figure 18: Several models (bodies) that used in Hill's viscoelastic muscle model (Figure 20). Models 18(a),18(b) are the Voight model and the Kelvin model. The last two, models 18(c) and 18(d), are two representations of the Zener model, which is either the Voight model in series with a spring, or the Kelvin model in parallel with a spring.

To explain the mechanics of the passive muscle force, Zener model 1 is used (Figure 18(c)). It contains a Voight body with a dash pot that has viscosity B and a spring X_1 with resting length x_{r1} This makes up the passive element, that contracts after rapid contraction of the spring that makes up the series element. This spring has a stiffness called K_{PE}, a length X_1, and a resting length x_{r2}. For a spring, stiffness relates to change in length: $K = \frac{\Delta F}{\Delta L}$. A dashpot has the next relation: $T = B\dot{x}$, force is viscosity times speed.The passive and series element are in series, so they have the same tension.

We now have the following formula's for their tension:

$$T = K_{SE}(X_2 - x_{r2}) \tag{1}$$
$$T = K_{PE}(X_1 - x_{r1}) + B\dot{X}_1 + A \tag{2}$$

The total length of the muscle ($X = X_1 + X_2$) gives us the next relations:

$$x_r = x_{r1} + x_{r2} \tag{3}$$
$$X - x_r = (X_1 - x_{r1}) + (X_2 - x_{r2}) \tag{4}$$

We can now substitute these to get the next relations:

$$X_2 = \frac{T}{K_{SE}} + x_{r2} \xrightarrow{\frac{d}{dt}} \dot{X}_2 = \frac{\dot{T}}{K_{SE}} \tag{5}$$
$$T = K_{PE}(X - x_r) - K_{PE}(X_2 - x_{r2}) + B\dot{X}_1 + A \tag{6}$$
$$T = K_{PE}(X - x_r) - K_{PE}\frac{T}{K_{SE}} + B(\dot{X} - \dot{X}_2) + A \tag{7}$$

This gives the next relation between muscle force and muscle length, which can be rearranged to represent the rate of change in force that is typical for a muscle:

$$T = K_{PE}(X - x_r) - K_{PE}\frac{T}{K_{SE}} + B(\dot{X} - \frac{T}{K_{SE}}) + A \tag{8}$$
$$\dot{T} = \frac{K_{SE}}{B}(K_{PE}\Delta x + B\dot{X} - (1 + \frac{K_{PE}}{K_{SE}})T + A) \tag{9}$$

Now that we know the equations that describe the muscle's response to the catch release experiment, we can extract the stiffness K_{SE} and K_{PE}. Remember that a sudden change in force (ΔF) and a sudden change in length (Δx) were observed in the first part of the experiment after releasing the catch. With $K = \frac{\Delta F}{\Delta L}$, we can directly infer the stiffness of the series elastic element, where K_{SE} is K, ΔF is the change in force that occurs immediately after the catch release, and ΔL is the displacement that accompanies the release, called Δx.

After the sudden change in force, the force in the muscle gradually declines to a steady state value. By comparison of the steady state values before and after the stretch, the stiffness of the parallel element, K_{PE}, can be found:

Before Stretch:

$$T_1(1 + \frac{K_{PE}}{K_{SE}}) = K_{PE}(X_1 - x_r) + A \tag{10}$$

After stretch:

$$T_2(1 + \frac{K_{PE}}{K_{SE}}) = K_{PE}(X_2 - x_r) + A \tag{11}$$

Change in force:

$$\Delta T(1 + \frac{K_{PE}}{K_{SE}}) = K_{PE}\Delta x \rightarrow \frac{K_{SE} + K_{PE}}{K_{PE}K_{SE}} = \frac{\Delta x}{\Delta T} \tag{12}$$

The last variable we need to find is the viscosity's constant, B. After the sudden stretch, the length does not change, so we know $\dot{x} = 0$ during the second part of the stretch. Taking $t = 0$ for the start of the sudden muscle stretch, we get:

$$T(0) = K_{SE}\Delta x + A \tag{13}$$

As a result, we get the next first order differential equation (Where $B\dot{x}$ is left out because $\dot{x} = 0$):

$$\dot{T}(t) = \frac{K_{SE}}{B}(K_{PE}\Delta x + A) - \frac{K_{SE}}{B}(1 + \frac{K_{PE}}{K_{SE}})T(t) \tag{14}$$

This equation is of the form $\dot{T}(t) = a_1 - a_2 T(t)$, which has a solution of the form $T(t) = \frac{a_1}{a_2} + (ce^{-a_2 t))}$. The term c can be derived from the initial conditions, that depend on the tension at $t = 0$, $T(0) = K_{SE}\Delta x + A$. Solving the equation gives the following formula for $T(t)$:

$$\boxed{T(t) = \frac{K_{SE}(A + K_{PE}\Delta x)}{K_{PE} + K_{SE}} + \frac{K_{PE}A + K_{SE}^2\Delta x}{K_{PE} + K_{SE}} e^{-\frac{K_{PE}+K_{SE}}{B}t}} \tag{15}$$

From an exponential function $e^{(-\frac{t}{\tau})}$ we know that the time constant τ is determined by the decay time at which the function has made approximately 63% of its decline. This time can be extracted from the quick release experiment, and B can be solved with the next equation: $\frac{B}{K_{PE}+K_{SE}} = \tau$.

Because we are trying to make a model that emulates general muscle characteristics, we will not use empirical data regarding a specific muscle type. Values for K_{PE}, K_{SE}, B can now be generalised. We leave out K_{PE}. The time constant is still determined by stiffness and viscosity, K and B. These steps can be imagined by taking out the passive spring element from Figure 20, connected to the model by dotted lines. The result gives an equation of the form $T(t) = A + ce^{-\frac{K}{B}t}$. The constant value c can be calculated from initial conditions, resulting in $c = T_0(0)$.We now have two equations that determine tension buildup and decay for the passive element in Hills visco-elastic muscle model (leaving out A for now):

Tension builds up:

$$T(t) = T_0(0)(1 - e^{-\frac{K}{B}t}) \tag{16}$$

Tension decays:

$$T(t) = T_0(0)e^{-\frac{K}{B}t} \tag{17}$$

The reaction of the passive element to two step functions is shown in Figure 19. At $t = 0$ the tension starts to build up. At this moment the tension builds up according to equation (16). At $t = 2$ the square wave input is stopped. Between $2 < t < 3$ the tension starts to drop according to equation (17). Now a second square starts building up tension in the muscle. According to the principle of superposition, the residue (lower curved line after $t = 3$) from the first response is added to the second muscle tension buildup (upper curved line after $t = 3$).

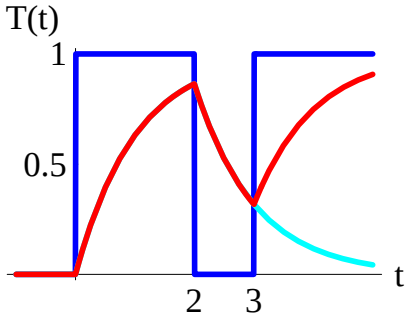


Figure 19: The muscle response to a square wave input when $0 < t < 2$. At $t = 2$ the square wave stops. From here the tension decreases. At $t = 3$ the second square wave starts. The tension starts to increase again. The tension is now the sum of the residua from the tension evoked with the first pulse and the tension generated by the second pulse. The residue after the second pulse is the lowest line in the graph after $t = 3$.

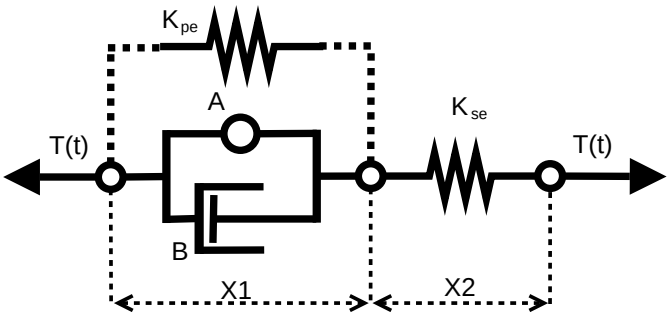


Figure 20: Hill's biomechanical muscle model. $T(t)$ stands for the tension or force. $A.S.$ is the active state or input to the muscle. B is the damping factor, represented by a capacitor. X_1, X_2 Are the contractile element and the length of the series elastic element. K is the series elastic element.

Mechanics of active and passive muscle force in a single isometric muscle twitch

Besides maximum tension that built up electrical stimulation of the muscle in the quick-release experiment, one can also apply a single pulse. This causes the muscle to build up isometric tension. The resulting active contraction in the active component A causes a twitch response.

Shadmehr and Wise [36] sampled the results from this twitch response and fitted an equation of the next form to the data: $T(t) = A(e^{-\frac{t}{\omega_1}} - e^{-\frac{t}{\omega_2}})$. Mechanically this can be represented by imagining the active element as a Kelvin body that receives a dirac pulse (this closely resembles a pulse coming from a motor neuron). Using a Laplace transformation, we can now derive the formula that was used to fit the data from Shadmehr and Wise's experiment.

We now know that the input from the active element resembles the same type of response as from the passive element. The tension buildup from the active component is now assumed to be of the same type as the tension response from the passive element. There are two time constants, ω_1 and ω_2. We perform a Laplace transformation on the input from the active component ($\omega = \frac{K}{B}, A = T_0(0)$):

$$h(t) = Ae^{-t\omega_1} \xrightarrow{\mathcal{L}} h(s) = \frac{A}{s + \omega_1} \tag{18}$$

We now have the Laplace transform of the input from the active component. For the passive component, this is the same formula, only the time constant (ω) is different. In the Laplace domain, we can multiply the Laplace formulas for both responses, and perform an inverse Laplace transform:

$$h(s) = \frac{A}{s+\omega_1}\frac{A}{s+\omega_2} \xrightarrow{\mathcal{L}^{\mathcal{T}}} h(t) = \frac{A}{\omega_2 - \omega_1}(e^{-t\omega_1} - e^{-t\omega_1}) \tag{19}$$

We now have the formula that can be used to describe tension in response to a pulse:

$$T(t) = \frac{A}{\omega_1 - \omega_2}(e^{-\frac{1}{\omega_1}t} - e^{-\frac{1}{\omega_2}t}) \tag{20}$$

The resulting impulse response is equivalent to an electric ciruit that is a second order RC filter. Taking different time constants, different response shapes are possible, as seen in Figure 22. The response found is the representation of the fundamental mechanical change in a muscle during a twitch, as proposed by A.V.Hill. This similarity is visible in Figure 21 and Figure 22. The left side shows the twitch response from a lobster to an isotonic contraction of a dorsal dilator muscle, in response to a 0.25 sec 30Hz stimulus. The right Figure shows two possible 'twitch' responses obtained using the model. The java application allows the user to experiment with this twitch response.

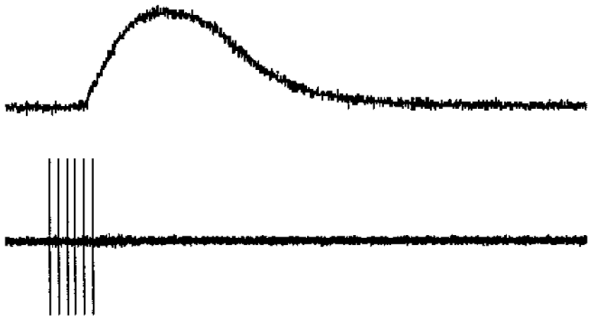


Figure 21: Twitch response of an isotonic contraction of a dorsal dilate muscle in the lobster in response to a 0.25 sec 30 Hz stimulus

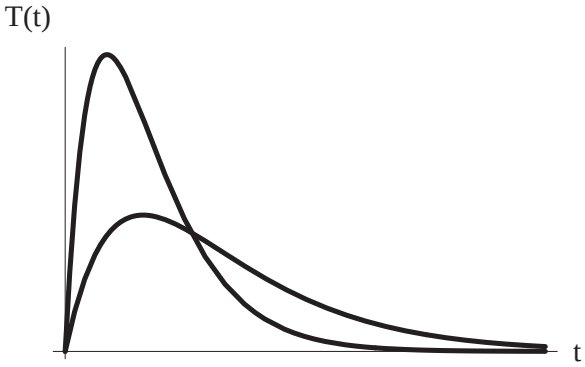


Figure 22: two 'twitch' responses from equation (20), using different time constants, represented by ω_1, ω_2 in the formula.

In a muscle there are three variables that influence the output power of a muscle: force, velocity and length. This means that a muscle that is in a state of contraction has other dynamic constants that a muscle that is in its resting position. This means a twitch response variates depending on the muscle's state. How these three dimensions are related to each other is explained below.

3.4.2 Hill's equation and the power velocity curve

So far we have described the force-time relationship of Hill's muscle model for isometric contraction. The force is the isometric tension of the muscle in response to a stimulus. An important aspect of muscle force and its function is not taken into account here, that is the power output of the muscle. The muscle can output both force and shortening velocity. This eventually gives the force-velocity curve, as seen in Figure 23(a) .

The power output in a muscle is ofcourse a very important notion in biological systems. The bicycle is an example of an 'interface' which maximizes the maximum power output by keeping the load equal while stepping on the pedals. The power output of a muscle can be calculated by multiplying force (tension) and speed. In Figure 23(a), the intersection between the straight and the curved line is the maximum power output (this is the maximum value in Figure 23(c)). The power output for different force-velocity relationships can be seen in Figure 23(c).

When a mechanical load is applied to a muscle, it will cost power to counteract it. This will result in a reduced shortening velocity, and eventually, when the load is to heavy, the muscle will stretch. For this 'power-lengthening velocity' an inverse 'power-shortening velocity' relation exists. This is not depicted because the VM only deals with pro-active planning that involves no stretch due to a mechanical load. A more complete muscle model could be made by inclusion of these changing dynamics.

Next to the shortening velocity of a muscle, the muscle dynamics of a muscle is also dependent on the actual length of a muscle. This relation is illustrated in the right part of Figure 23(b). The actual values of the forces and lengths varies in different species, and even in different types of muscles within the body. This is why the values are normalized to give an overall impression of muscle dynamics.

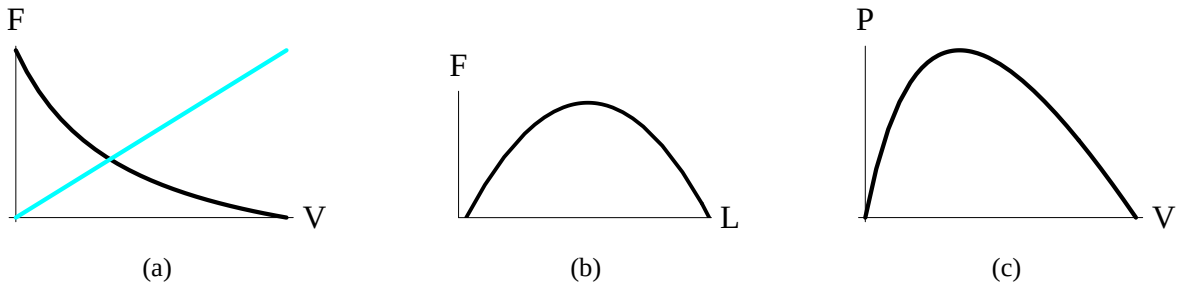


Figure 23: graphs that show the relation between force (F), velocity (V) and length (L), according to Hill's equation (21). 23(a) represents the relation between force and shortening velocity in a muscle, 23(b) represents the relation between force and muscle length and 23(c) is the product of force and shortening velocity (represented by power (P)), related to shortening velocity. The intersection of the two lines in 23(a) gives the maximum mechanical force of the muscle.

To describe the mechanics involved in the speed and force interaction, the sarcomere stands model. The total sarcomere interaction can de be described by Hill's equation:

$$(T + a)(V + b) = (T_0 + a)b \quad (21)$$

We can rewrite this equation to get the (T)ension and the (V)elocity of the muscle. Multiplying these gives the power output that is the straight line in Figure 23(a):

$$T = \frac{(T_0 + a)T}{V + b} - a, \quad V = \frac{(T_0+a)T}{T+a} - b, \quad P = VT = \frac{bT(-T + T_0)}{a + T} \quad (22)$$

Where P describes the force generated by a muscle, V is the speed at which a muscle contracts, a and b are constant. The constant a describes the force expended to make the muscle contract, and b describes the smallest contraction rate of the muscle.

We can get the normalized force-velocity by normalizing V (dividing by V_{max}, the maximum muscle velocity) and then dividing by T_0 and rearranging the formula:

$$V = \frac{(T_0 + a)T}{T + a} - b \quad \frac{V}{V_{max}} = \frac{(T_0+a)k}{T+a} - k = V' \tag{23}$$

$$V' = \frac{(1 + a/T_0)k}{T/T_0 + a/T_0} - k \quad = \frac{(1+k)k}{T'+k} - kV' = \frac{1-T'}{1+T'/k} \tag{24}$$

The normalized force-length relationship is described by the following formula [17]:

$$\frac{T_l}{T_{l,0}} = k_1(\frac{L}{L_0})^2 + k_2(\frac{L}{L_0}) + k_3 \tag{25}$$

Combining the normalized force-velocity and the normalized force-length curves gives an over-all view of the interaction of these three variables as seen in Figure 24. Where $T_{l,0}$ is the maximum isometric force generated by the contractile element at muscle length L. The values k_1, k_2, k_3 can be extracted from physiology literature. In general, the next values are possible for k_1, k_2, k_3: $-4.5 < k_1 < -13.5, 9 < k_2 < 28.2, -3.5 < k_3 < -14.0$. T_l is the isometric force generated by the contractile element at muscle length L. To generate the force-velocity-length curve, a combination of the two formulas is used:

$$T_v = T/T_0 \tag{26}$$

$$T_l = T_{l,0}(k_1(\frac{L}{L_0})^2 + k_2(\frac{L}{L_0}) + k_3) \tag{27}$$

$$F_v l(T_v, T_l) = ((T_l + a)b)/(T_v + b) - a \tag{28}$$

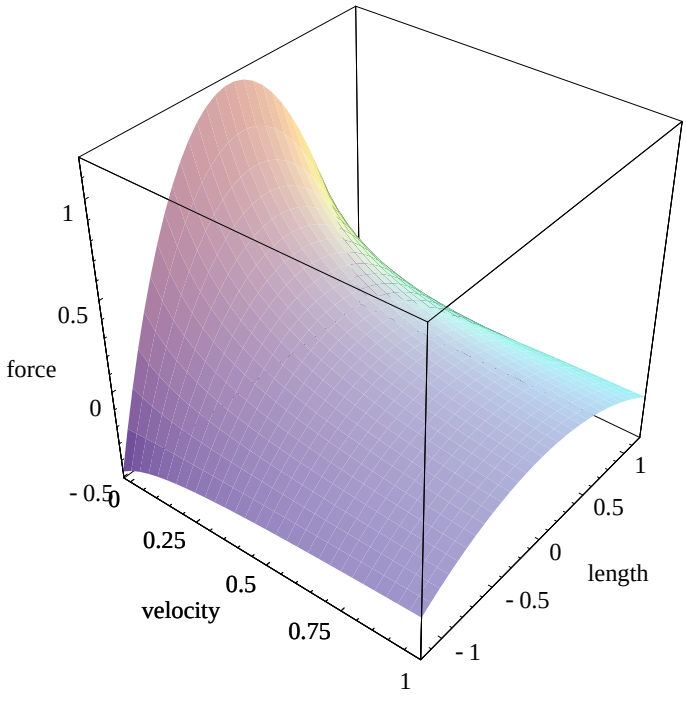


Figure 24: Normalized Force-Velocity-Length relationship of a muscle, showing how much force can be generated in relation to muscle velocity and muscle length.

Using the muscle model for robot control
Now that we have investigated muscle mechanics and set up formulas to simulate a twitch response, we can use the model to simulate speed profiles generated by the tension in a twitch. To use the muscle model as an actuator in autonomous systems, the muscle contraction from the twitch response stands model for motor control. The muscle contraction can stand for energy fed to the motor, speed of the wheels, or translation into Cartesian coordinates (x,y). The muscle contraction is instantiated through a pulse as effector. The evolution of the signal is from there on the response of the modeling system chosen (mass-spring, gauss, sinus). Differences in left and right wheel pulses, time-delays, steering-influences (e.g. an extra signal) and other aspects influence the robot's path.

After generation of the twitch response, a transformation for differential control is made. The path generated is then fitted on the spline (see §4.3 for more information). This approximation is made by the Broyden-Fletcher-Goldfarb-Shanno (BFGS) quasi-newton optimizer. This numerical method makes an unconstrained search which minimizes multivariate functions. The returned result are the variables needed to control the model's variables. The model using the twitch will return $< \omega_1, \omega_2, A_1, \omega_3, \omega_4, A_2 >$, from formula (20), where ω_n are the time-constants, and A_n are the amplifications (remember that this was the starting tension $T_0(0)$). This model can be representing a limb movement model when the model is used to represent muscles in a biomechanical limb system. In our case with the differentially controlled robot, six variables are returned since there are two muscles in the model, one to steer each robot wheel (see §3.8.2 for more information).

3.5 Error propagation in control loops

Biological systems respond to sensory stimuli in order to reach an objective. When there is a difference between the objective and the current state observed by sensory input, action is taken. Since this activity influences the sensory input, this type of control is called 'closed loop feedback control' (see Figure 25). When sensory input is perfect, the desired objective will be obtained. Because of propagation delay (lag), faulty sensor readings and other causes, the sensory input may not be accurate. This causes the feedback loop to send back a signal that is contaminated with an error proportion. The system will undergo an oscillation due to this error, also called 'hunting'. This can be counteracted by damping the signal between the sensor and the effector. This will result in a system that is less susceptible to oscillation. The downside of this procedure is that it slows down reaction time. A balance between feedback and damping has to be found. An example of such a balance is the stalking behavior of the jumping spider (see Figure 8), where the delay time δ is the error factor that regulates the balance between feedback and damping.

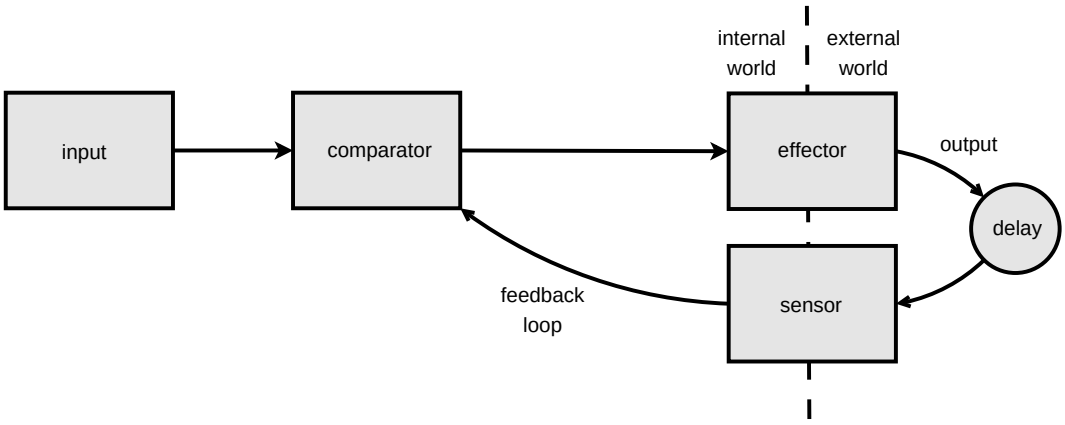


Figure 25: First order feedback loop system (reprinted from [25],p.3). The feedback loop introduces a time delayed feedback and a signal error proportion that may cause the system to become unstable.

The VM uses an open loop type of control. This bypasses the problem of balancing feedback and damping. Instead, the generation of motor commands is based on knowledge about the controlled object. The differences between the feedback loop and the open loop control are visualized in Figure 26. In the open loop system a feedback is obtained by observation of results from the action after the action has been executed. This feedback influences the inverse model to make it react more accurately in the future. This type of control is needed when the delay in the feedback signal is too large and there is no valid direct feedback. An example of a delay that is too large is the spider-jump. The spider cannot afford to stalk its prey when the prey becomes aware of its presence, so it has to react immediately. Another example is ball-catching. The grip force needs to be estimated before the ball is caught, because a feedback response to the gripping force would not be fast enough to approximate the grip force needed to catch the ball [44]. The pro-active open loop does not mean no feedback loops are present. When needed, it can override existing feedback loops to execute a planned action.

3.6 The motor command model

Hill's work played an important role in bio mechanics. Hill's equation (eq 21) is still used to model muscles from animals and humans. However, human arm movement is more than muscle mechanics. In order to use one's arm to reach a target pose, motor commands are needed. Shadmehr [35] and Kawato [28] both showed the likeliness of the existence of internalized models for motor command control. Shadmehr performed experiments with humans who had to perform a grasping task. Their grasping arm was held in an artificial force field when they performed the task. This disturbed the kinetics of the subject's arm movement. Different tests showed that the adaption of the subject's (supposed) internal motor command model had to be an adaptive model with a generalied character. The experiment also showed that the model had to be there, because there was no other way to account for rapid adaptive changes is the subject's arm movement. Kawato also showed this. He stated that a simple feedback loop would be to slow to control human arm movement. (see Figure 26, upper part) After this he proposed the feed forward control. For this model to be correct, there has to be an inverse dynamics model to predict arm movement in order to 'outsmart' the delay in a feedback loop (see Figure 26, lower part). This feed forward model was not enough to account for the movement of the human arm. Kawato proposed that an internal model was also used. He combined the dynamic model and the representation of an internalized model.

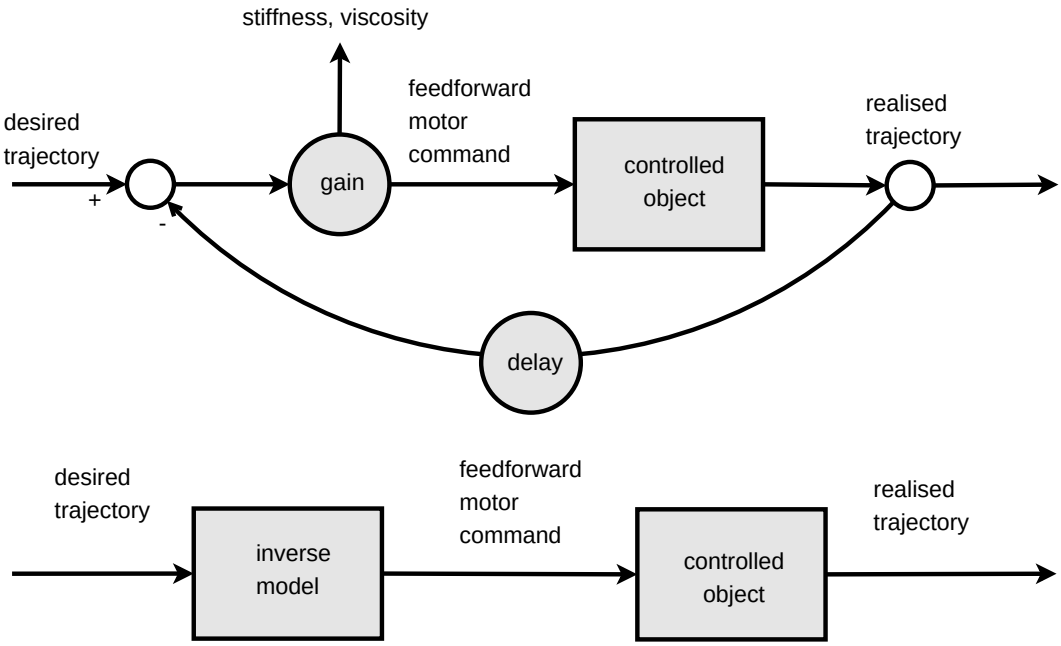


Figure 26: Upper part - closed loop feedback model Lower part - open loop control using inverse model. These models illustrate the advantage of the use of an inverse model to predict the realised trajectory. When using the second model, there is no danger of a feedback loop that may cause instabilities.

3.7 Importance and biological foundation of notion of the efference-copy

In Figure 7 is shown how a 'sense-act' feedback loop can lead to faulty behavior. Biological systems use the 'efference-copy' [11] to anticipate these kind of errors. The 'efference-copy' is a copy of the 'efferent' motor signals. Efferent signals are signals that from the CNS to the muscles or other effectors in the body. Afferent signals are dispatched in the oppsite direction, from sensory organs to the CNS. This efference copy can be compared with the results obtained through afferent sensory information (delayed via a buffer, $\Delta T = T_{lag}$). This is called reafference, a comparison that yields the error between the desired action and the performed action. An error between the two can then be corrected (see Figure 27). Shadmehr [35], as well as Kawato [28], showed that there is a flexible internal model to accurately control movement dynamics. The efference-copy is used in a feed forward loop to anticipate errors, thus circumventing the lag error as seen in Figure 7. The generalied visco-elastic muscle model tries to combine these two by a making such a model, and using the prediction of this model for motor control (the 'efference-copy') to avoid lagging effects. The result is a preplanned feed forward trajectory. The control loop including the efference-copy is shown in Figure 27.

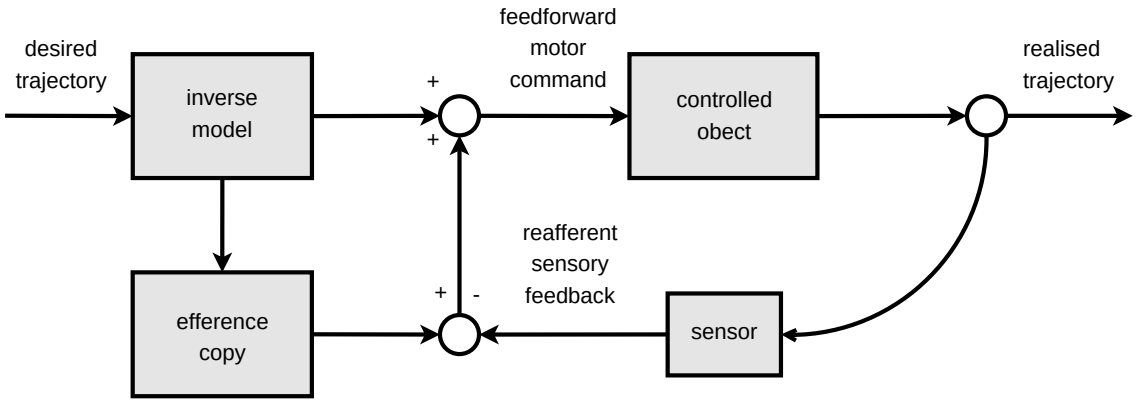


Figure 27: Control loop using the efference-copy. because an open loop is not able to respond with a sensory feedback loop, the efference copy predicts sensory information so that the lag between acting and sensing can be compared to a predicted sensory input. In this way the system counteracts the sensory lag.

3.8 Using the motor command model

3.8.1 Example of an internalized motor command model that uses biomechanics

The use of bio mechanics in robotics has shown some good results. Fagg[2] has used an internalized model based upon it to make a model to simulate human arm movement. In Fagg's experiment, a visco-elastic muscle model is used to control a robotic arm. In their experiment they use what is called a 'muscle synergy space'. This space is a collection of muscle activation patterns (there are several muscles in their experiment), which can be controlled in a coordinated fashion. Their proposed 'muscle synergy space' is based upon primate research by Miller [26] which showed that this kind of 'muscle synergy space' control is present in the primate's magnocellular red nucleus. Coordinated muscle movement can thus be learned in the cerebellum of these primates. In essence, the generalised muscle model is using the same basic elements to construct a mapping from the robot's motor commands to its environment. Fagg also used an efference-copy to control his robot.

3.8.2 Conversion from speedprofile to steering speeds

Since we have no robotic arm to test the biomechanical model, the muscle twitch contraction speeds will be represented by a translation to robot control. The robot is differential drive controlled, and there are several ways to represent the twitch speeds in connection to robotic control. To see what the result of these different translations are, there is a java applet (see §A.1 in Appendix A for more information) that simulates a robot controlled by the biomechanical model.

Cartesian

The Cartesian method of control is the simplest way to calculate the path of the robot. The discretisezed speeds from the twitch response are read into a buffer. Each buffer item represents the next time step in the discrete process that underlies this path generation. This buffer contains a new x position and new y position for the robot. In the applet, the old position (x, y) is used to calculate the directional angle of the robot, called `phi` in the program.

Speed

The speed method of control is a little more complicated. The next position of the robot is now dependent on the robots angle and its angular speed. Figure 28(a) shows the circular path that results from two different wheel speeds. In each time-step, calculations are made with a constant left-wheel and right-wheel speed. This means that a discretization with increasingly larger time steps becomes increasingly inaccurate, since no speed variations are taken into account. Because of the small time steps, this has no significant impact on the resulting path. The new $< x, y, \theta >$ array can be calculated by looking at the circular path the robot travels and the angle that the robot has made during each time period. Looking at Figure 28(a) it follows that for the center of the robot, the vector giving the direction of forward motion is the next:

$$\frac{dx}{dt} = \frac{Vr + Vl}{2}\cos(\theta(t)) \tag{29}$$

$$\frac{dy}{dt} = \frac{Vr + Vl}{2}\sin(\theta(t)) \tag{30}$$

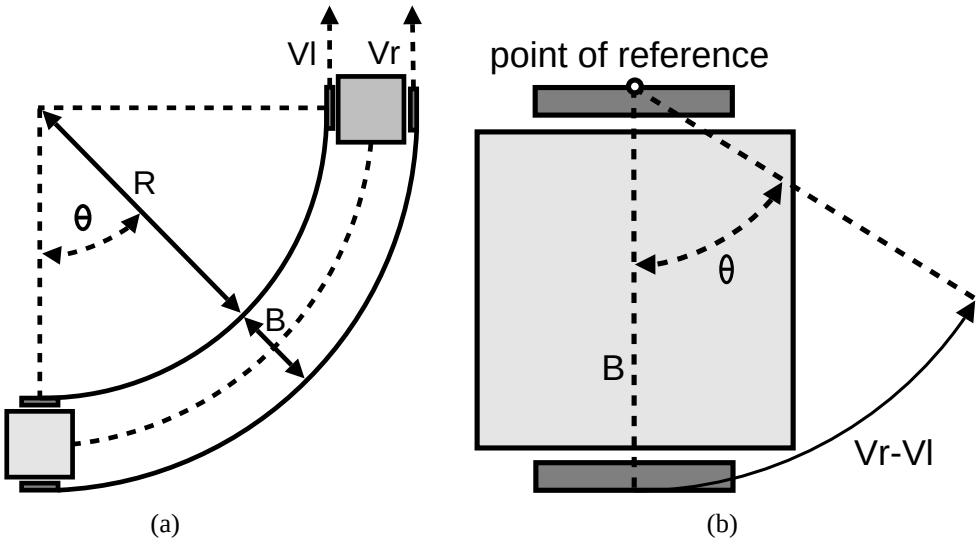


Figure 28: Differential control of a robot with two different reference points. When the robot travels with two constant right- and left wheel speeds that are not the same, its path will be circular, which is visible at the left side. When one substracts the right and left wheel speed, the robot travels a circle around the wheel with the lowest speed. These two points of reference give an insight how to solve the problem of differential steering.

$\frac{\Delta\theta(t)}{\Delta(t)}$ is the rotational speed. This speed is also time dependent. Taking the center of the left wheel as a point of reference (see Figure 28(b)), a formula for $\theta(t)$ can be made. Since the center of the left and right wheel are perpendicular, their rotation angle, θ, is the same as the θ in Figure 28(a).

For the center of the robot, $\theta(t)$ can be described as follows:

$$\frac{d\theta}{dt} = \frac{Vr - Vl}{b} \tag{31}$$

After integration we get:

$$\theta(t) = \frac{Vr + Vl}{b}t + \theta_0 \tag{32}$$

We can now fill in $\theta(t)$:

$$\frac{dx}{dt} = \frac{Vr + Vl}{2}\cos(\frac{Vr + Vl}{b}t + \theta_0) \tag{33}$$

$$\frac{dy}{dt} = \frac{Vr + Vl}{2}\sin(\frac{Vr + Vl}{b}t + \theta_0) \tag{34}$$

Integration gives the result:

$$x(t) = x_0 + \frac{b(Vr + Vl)}{2(Vr - Vl)}(\sin(\frac{(Vr - Vl)t}{b} + \theta_0) - \sin\theta_0) \tag{35}$$

$$y(t) = y_0 + \frac{b(Vr + Vl)}{2(Vr - Vl)}(\cos(\frac{(Vr - Vl)t}{b} + \theta_0) - \cos\theta_0) \tag{36}$$

This calculation is made by the 'speed' function, that returns the next x, y, θ for the next time interval. This function is also a private function. The programming has been done in a similar way as the formula choice.

Energy
The 'energy method' is a transformation of speed by $E = \frac{1}{2}mv^2$. This is exactly the same as with speed, but now the square root of the energy is taken. This gives the speed necessary to calculate the next robot position.

3.9 Energy constraints in biological systems

The next point in the assumption of more complex modeling than simple reactive behavior is derived from numerous studies of primate- and human arm movement [30, 41]. These studies show us that most movements are energy-conserving, and adapted to the type of animal that is researched. This energy conservation means that the trajectory paths of the movements researched have minimum jerk speed profiles. This can be modeled by the optimization of trajectory formation with fifth order splines (see §5.2 for more information). This minimum jerk assures the animal from a risk-free movement with a speed that is safe to its biomechanical properties. This means it is not likely that an animal will damage itself while performing a limb movement. This constraint is crucial to survival. An example of this system to fail is the rabbit. When lifted in the air, its 'kick' movement that normally enables the rabbit to scatter away to a safer environment, can break its own spine. Ofcourse, a rabbit in its natural environment is rarely 'lifted from the ground', so this risky ability weighted itself between spine injury and fleeing speed. It is obvious that a robot will also profit from this structure-preserving trajectory formation, since its delicate parts are also easily damaged and will also suffer less from wear and tear.

3.10 Importance of the monophasic jump in ego-motion

The trajectory formation model that steers the robot is based on a monophasic speed profile that is generated by the VM. Insertion of a via point essentially combines two monophasic movements. The emphasis that has been put upon the monophasic character of the ballistic movements the robot can make has several reasons:

Repeated occurrence of monophasic movement
The first reason to take the sigmoid buildup and decay as characteristic feature of the speed profile is its repeated occurrence in nature. The second law of motion from Newton states that $Force = Mass{\cdot}Acceleration$. This generates a linear speedprofile. In most cases, the force gradually increases during the motion, especially in biological systems. This essentially causes a sigmoidal buildup and decay (bell-shaped) of speed of most objects that are set in motion. An example is the muscle's gradual buildup of tension, and thus a sigmoidal speed buildup when the muscle shortens.

Connection of constrained movement with the internalized motor command model
The second argument is the role of the internalized motor command model. The trajectory it generates is limited not only by the visco-elastic muscle model, but also by a number of constraints. As we will see in the §4.2, a smooth (bell-shaped) speed profile is one of them. This is automatically closely related to the monophasic character of a muscle twitch. Besides this, the internalized motor command model also generates a safe departure and arrival at the start and end position by a gradual increase and decrease of speeds at the start and end of a trajectory. This reduces the chance of collision, and gives the robot time to correct for errors caused by incorrect position estimates of the target position. These errors could for example be caused by inaccurate depth estimation when using automated vision.

Dynamics involved with the jump region
The last argument has to to with the space surrounding the robot. For a human beeing, this space can roughly be built up as seen in Figure 29. In the Figure, there are four circles, representing a region that requires a different type of movement:

- The private space, representing the space in which targets can be reached without ego-motion or torso movement.
- Partial ego-motion, the space where movent of the torso is needed in order to reach or manipulate objects. Full ego-motion is not needed.
- The region of space where monophasic ego-motion is needed. In order to reach a target, a jump or ego-motion is needed. When the person decides to walk to the target, he or she will not obtain a constant speed, thus the speed profile is generally bell-shaped. In both cases this is monophasic movement.
- Constant velocity planning space. The distance to be travelled is becoming too large to jump, or too large to have a bell-shaped speed profile. It may now be useful to obtain a constant travelling velocity.

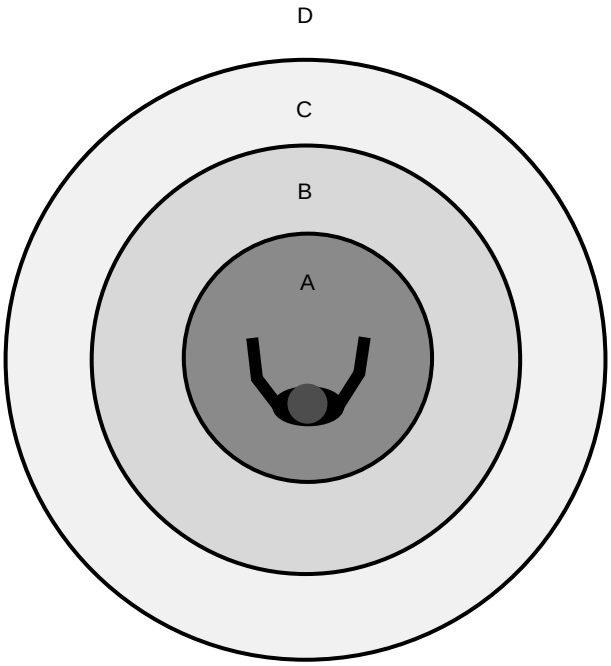


Figure 29: General model for types of movement related to targeted movements.
A. The private space, represents the space in which targets can be reached without ego-motion or torso movement.
B. The partial ego-motion space, describes the space that requires movement of the torso in order to reach for a target.
C. The monophasic ego-motion space, requires a jump or an ego-motion that has a monophasic speed profile.
D. The constant velocity planning space, where the distance to be travelled is becoming too large to jump, or too large to have a bell-shaped speed profile. It is now useful to obtain a constant travelling velocity.

Figure 29 only gives a general indication of the types of movement that fall under a given distance between the target and the robot's position. In the case of the jumping spider for example, the private space is relatively small, and the jumping space much larger. A more extreme example of a large monophasic ego-motion space is the jumping region of a flee, that spans up to 200 times its body length.

The important notion of the circles in the motion space is the monophasic character of the first three types of movement. In §4.2 we will see that the private space of humans and primates have speed profiles that consist of monophasic or combinations of monophasic movement. It is fairly safe to assume that torso movement also complies to the 'smoothness' of this type of movement. When we look at the 'monophasic ego-motion' space, this consists either of a ballistic jump or a bell-shaped speed profile that allows the avatar to quickly accelerate and decelerate. That is why the robot controlling speed profile is generated using the biomechanical model that generates monophasic and biphasic speed profiles.

4 The biomimetic model, the BM

Now that we have a way to model an isolated twitch movement we will look at trajectory formation. Trajectory formation describes the movement of a specific part of the body from an initial to a target position.

4.1 Extracorporal trajectory formation

In order to generate such a trajectory, one needs a model to predict this trajectory. An important question in this matter is whether to look at extracorporal movement or to look at the motions that can be produced with a biomechanical model.

Researchers in favor of that latter model assumed that joint position and movement are used by the central nervous system (CNS) for trajectory formation [41]. This greatly reduced the complexity of the control problem associated with limb movement with many degrees of freedom. The model states that the CNS keeps the angular velocities between the joints in a constant ratio.

Even though the CNS has to coordinate the joint position in some way, other research [41] showed that it is improbable that there is a connection between trajectory formation and a representation of joints and their angular velocities. This research came up with a connection between trajectory formation and commonalities in extracorporal trajectories. Lashely [24] and Bernstein [31] introduced the idea that it was possible that the CNS forms extracorporal trajectories and uses this trajectory to come up with the motor commands that achieve this start to target limb movement. Studies [1, 18] supported this idea by experiments in planar, unconstrained human and primate movements.

Their experiments showed that hand movements between two targets generated roughly straight, single-peaked (monophasic), bell-shaped speed profiles. These results are independent of the part of the workspace in which the movement was performed. These common invariant features of the movement were only present in the extracorporal coordinates of the hand. This is a strong indicator that trajectory formation takes place in terms of hand movement and not in terms of joint rotations.

4.2 Mathematical model for trajectory formation

Hogan & Flash presented a mathematic model to predict the arm movements they observed in their experiment. For their experiment they obscured the arm position of the subject by a plexiglass plate. They restrained the shoulder and the subject sometimes wore a brace to constrain the wrist. During the experiment the subject was instructed to move their hand from an initial to an end position. Four types of experiments recorded different types of movement. These movements were measured and analyzed. They found out that only a few of the possible trajectories occurred in the experiment. This led them to conclude that there was an underlying adaptive process generating movements that are optimised kinematic or dynamic variables the trajectory. The optimization of these variables is a rational assumption when modeling a biological process because of the analogy of optimization in nature resulting from natural selection.

The critical step in this analysis is the choice of an optimizing function. Since arm movement trajectories tend to get more smooth when trained, Hogan & Flash stated that this smoothness must be part of the optimizing function that is biologically built in in humans. For Hogan & Flash [41] this was the starting point for an optimization based mathematical description of arm movements in primates.

Eventually they found that a trajectory that is optimized to a minimum jerk optimization criterion successfully predicted qualitative and quantitative features of single-joint forearm movements. It is the minimum jerk optimality constraint that will be used to generate trajectories in this MSc thesis.

Based on their observation that training smoothens hand movement trajectories Hogan & Flash [41] stated the next optimizing rule:

> *Generate the smoothest motion to bring the hand from the initial position to the final position in a given time. The hand must move to the final position through a via point at an unspecified time.*

The first line in this statement concerned point to point movements. The second sentence referred to experiments that involved trajectories curved around an obstacle or following a curved line that was printed on the plexiglass. In their research they found that constraints like maximum torque, maximum torque change or maximum acceleration are not important. The speeds, acceleration and torque changes recorded in the experiment did not reach the limits of neuromuscular performance. Without these constraints it was determined that a fifth order polynomial could describe the observed trajectories with enough accuracy.

4.3 Polynomials and Splines

To describe the trajectory we need a way to model fifth order polynomials. A Spline is a piecewise polynomial. The pieces that make up the spline are connected at the knots through which the pieces are interpolated (see fig31). The general form of a polynomial: $\sum_0^{i=n} c_i t^i$. For $n = 3$ this would give $x(t) = c_0 + c_1 t + c_2 t^2$. c is the coefficient vector which determines the shape of the particular piece of the curve between two knots. The polynomial is used over an interval from $t = 0$ to $t = 1$ to draw its part of the curve.

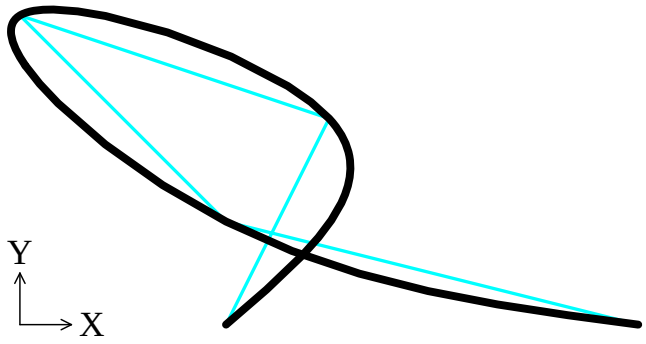


Figure 30: cubic spline through knots {0, 0}, {1, 2}, {-2, 3}, {0, 1}, {4, 0}

4.4 Types of splines

There are many ways to interpolate data using polynomials. Lagrangian interpolation is one of the most straight forward interpolations, using a polynomial of degree $N-1$ to pass through $N-1$ points. Its main disadvantage is that there is no easy way to manipulate speed, acceleration and jerk. Besides, the Lagrangian interpolation is infamous for its instability at higher order polynomials.

Other types of interpolation have knot to knot controlled polynomials of degree N. Some well-known splines of this category are:

- Bezier splines, uniform interpolation with piecewise polynomials of order N.
- Hermite splines, a family member of bezier, with $C0$ and $C1$ continuity at the knots.
- Catmull-Rom splines, a special case of spline with built in $C0$ and $C1$ continuity.
- B-splines, have $C0, C1, C2$ continuity, but are not guaranteed to pass through a knot.
- Natural cubic splines, with their first and second derivatives set to zero at the start and end point.

To minimize an interpolated path through several knots, characteristics of several types of these splines were used to create a bezier spline.

4.5 N-order splines

A bezier curve is specified by a number of points (control points). The number of control points determine the order of the spline. A line can be described by a bezier with a first order polynomial, $x(t) = a_1 t + a_0$ The resulting line is graphically depicted in Figure 31. P_0 and P_1 are the endpoints, Q can be any point on the line. A parametric representation of $Q_{(t)}$ is given by $Q_{(t)} = (1-t)P_0 + (t)P_1$, where t runs from 0 to 1. We are trying to find a_1 and a_0 from $x_{(t)}$. Rewriting $Q_{(t)}$ gives $Q_{(t)} = (P_1 - P_0)t + P_0$, which shows us that $a_1 = (P_1 - P_0)$ and $a_0 = P_0$. The representation of $Q_{(t)}$ can be simplified to $\sum_0^{i=n} B_i P_i$, where $B_0 = 1 - t, B_1 = t$.

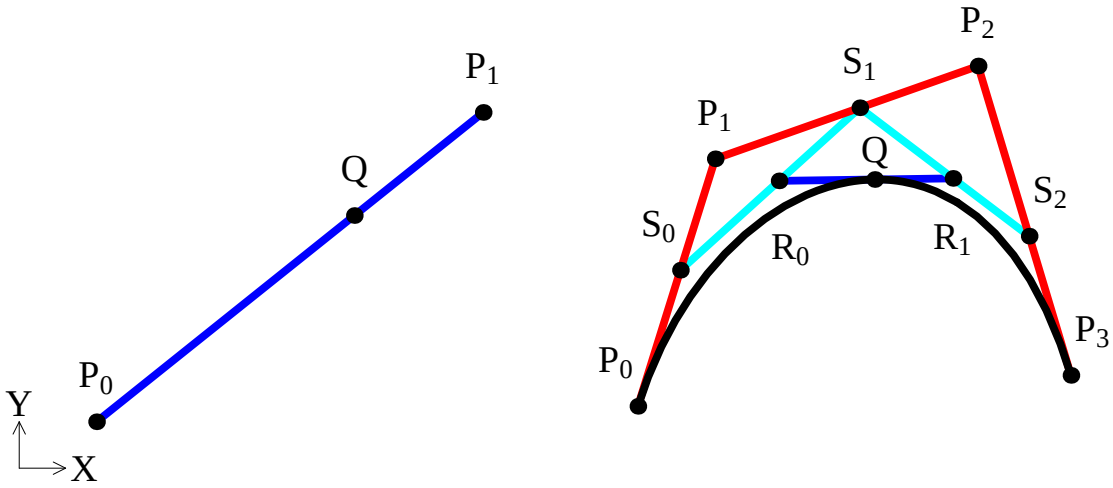


Figure 31: Left: linear spline, right: cubic spline. The path of point Q is the spline. This path is constructed by the control points. For the linear spline (left picture) these control points are P_0, P_1. In the case of the linear spline there is only one control point, Q, that travels from P_0 to P_1.The path of Q on the cubic is related to the traversal of the control points over the lines on which they are placed. Picture on the right hand: for the cubic spline the control points are $P_0, P_1, P_2, P_3, \quad S_0, S_1, S_2, \quad R_0, R_1$. In the case of the cubic spline there are several control points traveling at the same time, eventually creating the path of Q. Two examples in the case of the cubic spline are S_0 traveling from P_0 to P_1 and Q traveling from P_0 to P_3. All points that travel, travel in one second from their start point to their end point.

A bezier curve with a higher order like the third order cubic spline in Figure 31 (right) is defined by four control points, P_0, P_1, P_2, P_3, where P_0 and P_3 are the start and end points. Just like in the linear case we can express $Q_{(T)}$ in terms of P_0, P_1, P_2, P_3:

$$Q_{(t)} = (1-t)R_0 + (t)R_1 \tag{37}$$
$$R_0 = (1-t)S_0 + (t)S_1 \tag{38}$$
$$R_1 = (1-t)S_1 + (t)S_2 \tag{39}$$
$$S_0 = (1-t)P_0 + (t)P_1 \tag{40}$$
$$S_1 = (1-t)P_1 + (t)P_2 \tag{41}$$
$$S_2 = (1-t)P_2 + (t)P_3 \tag{42}$$

We can now substitute S_0 with (40), R_0 with $(1-t)((1-t)P_0 + (t)P_1) + (t)((1-t)P_1 + (t)P_2)$. After filling in all substitutions, we get $Q_{(T)}$ expressed in P_0, P_1, P_2, P_3:

$$Q_{(T)} = P_0(1-t)^3 + P_1 3t(1-t)^2 + P_2 3t^2(1-t) + P_3(t)^3 \tag{43}$$
$$Q_{(T)} = P_0 - 3P0t + 3P_1t + 3P_0t^2 - 6P_1t^2 + 3P_2t^2 - P_0t^3 + 3P_1t^3 - 3P_2t^3 + P_3t^3 \tag{44}$$

We can now express $Q_{(t)}$ in terms of P_0, P_1, P_2, P_3 and t^0, t^1, t^2, t^3, which will give us the desired coefficients for the polynomial that describes the curve. This gives the following formula:

$$Q_{(t)} = 1P_0 - 3(P_0 - P_1)t + 3(P_0 - 2P_1 + P_2)t^2 + 1(P_0 - 3P_1 + 3P_2 - P_3)t^3 \tag{45}$$

The resulting formula has two regularities, which are caused by the repeated substitution of the control points. In a sense we created a pascal table, multiplied by a pascal row. The triangles below are different forms of 'pascal's triangle'. The entries in Pascal's triangle are called binomial coefficients. The entries can be found with the following formula: $[n,k] = \frac{n!}{k!(n-k)!}$, where n is the row, and k the column.

$$\begin{array}{ccc}
1 & 1 & = (1-t)^0 \\
1 \quad 1 & 1-t & = (1-t)^1 \\
1 \quad 2 \quad 1 & 1-2t+t^2 & = (1-t)^2 \\
1 \quad 3 \quad 3 \quad 1 & 1-3t+3t^2-t^3 & = (1-t)^3 \\
1 \quad 4 \quad 6 \quad 4 \quad 1 & 1-4t+6t^2-4t^3+t^4 & = (1-t)^4 \\
1 \quad 5 \quad 10 \quad 10 \quad 5 \quad 1 & 1-5t+10t^2-10t^3+5t^4-t^5 & = (1-t)^5
\end{array}$$

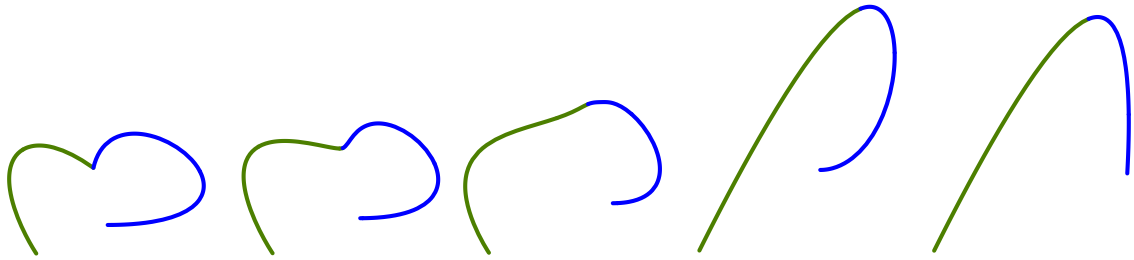
Because of the recurrent replacement of the control points to obtain the formula for $Q_{(t)}$, the pascal triangle also occurs between the brackets of $Q_{(t)}$. This leads to the next matrix multiplications for the coefficients in the e polynomial describing the curve associated with the control points:

$$\begin{pmatrix} P_0 & 0 & 0 & 0 & 0 & 0 \\ P_0 & -P_1 & 0 & 0 & 0 & 0 \\ P_0 & -2P_1 & P_2 & 0 & 0 & 0 \\ P_0 & -3P_1 & 3P_2 & -P_3 & 0 & 0 \\ P_0 & -4P_1 & 6P_2 & -4P_3 & P_4 & 0 \\ P_0 & -5P_1 & 10P_2 & -10P_3 & 5p_4 & -P_5 \end{pmatrix} \begin{pmatrix} 1 \\ 5 \\ 10 \\ 10 \\ 5 \\ 1 \end{pmatrix} = \begin{pmatrix} P_0 & 0 & 0 & 0 & 0 & 0 \\ 5P_0 & -5P_1 & 0 & 0 & 0 & 0 \\ 10P_0 & -20P_1 & 10P_2 & 0 & 0 & 0 \\ 10P_0 & -30P_1 & 30P_2 & -10P_3 & 0 & 0 \\ 5P_0 & -20P_1 & 30P_2 & -20P_3 & 5P_4 & 0 \\ P_0 & -5P_1 & 10P_2 & -10P_3 & 5P_4 & -P_5 \end{pmatrix}$$

In which the last matrix is the solution, and every row represents a coefficient. For example, the third coefficient, which belongs to t^2 in $Q(t)$ is: $10P_0 - 20P_1 + 10P_2$. This way we can now represent a spline piece of any order, given its control points. In this case we used a fifth order spline because this order allows optimization of minimum jerk, since the polynomial is from an order which allows us to take the derivative associated with the minimum jerk criterion (the variation of acceleration).

4.6 Connecting N-order spline-pieces through N knots with CN continuity

We are now able to move from point A to B along a spline piece. The next step is to be able to move from B to C. Even though only one via point is of our interest in this thesis, the following method describes a method to find a spline through N knots while maintaining Cn continuity.



(a) $C0$ (b) $C1$ (c) $C2$ (d) $C3$ (e) Minimum Jerk

Figure 32: Cn continuity. Cn-continuity indicates that a spline with a connecting knot is continuous in its N-th derivative in the knot. The knots are indicated by the color change.

4.6.1 Continuity

$C0$ continuity means that the splines connect at knot intersection. There is $C1$ continuity when the first derivative, speed is continuous. $C2$ adds continuity to the second derivative, which means acceleration is continuous. These types of continuity are illustrated in (Figure 32). Different types of splines have different ways to ensure Cn continuity. Ofcourse these differences only relate to continuity in derivatives of the polynomials.

4.6.2 Ensuring continuity

To ensure a polynomial is continuous to Cn, the control points must be constrained. For example, to ensure C0 continuity, the first piece of the spline (the right parts in Figure 32), ending in the first knot must connect with the left (second) part of the spline. We know that t runs from 0 to 1 in the red and blue part. At the start of the blue part, $t = 0$. We can now fill in the x_1 coordinate (first knot's coordinate), and substitute t with 0, yielding the next relation: $x(t = 0) = x_1 = c_0 + c_1 0 + c_2 0^2 + c_3 0^3 + c_4 0^4 + c_5 0^5$, thus $c_0 = x_1$ is solved. Note that because the dimensions $(x, y, ...)$ are linearly independent, we can do these calculations separately. For example we take two polynomials:

$$x_0 = c_0 + c_1 t + c_2 t^2 + c_3 t^3 + c_4 t^4 + c_5 t^5 \tag{46}$$

$$x_1 = d_0 + d_1 t + d_2 t^2 + d_3 t^3 + d_4 t^4 + d_5 t^5 \tag{47}$$

C_0 continuity means that two adjacent splines are joined in the knot. Higher order continuity means that the n^{th} derivative from the connecting splines is continuous.

C_1 continuity means that the first derivative is continuous, C_2 means that the second derivative is continuous, etc ... To make the splines C_n continuous, their control points are related to each other. The next constraints ensure C_n continuity:

$$
\begin{aligned}
C0 \longrightarrow \quad & d_0 = c_5 && (48)\\
C1 \longrightarrow \quad & (d_1 - d_0) = (c_5 - c_4) && (49)\\
C2 \longrightarrow \quad & (d_2 - d_1) - (d_1 - d_0) = (c_5 - c_4) - (c_4 - c_3) && (50)\\
C3 \longrightarrow \quad & ((d_3 - d_2) - (d_2 - d_1)) - ((d_2 - d_1) - (d_1 - d_0)) = && (51)\\
& ((c_5 - c_4) - (c_4 - c_3)) - ((c_4 - c_3) - (c_3 - c_2))
\end{aligned}
$$

The pattern in these equations is the same as with the pascal trees from $Q_{(t)}$. This also results in a pascal-triangle based matrix for the calculation of the derivatives:

$$
\begin{aligned}
d_0 = \quad & & c_5 && (52)\\
d_1 = \quad & d_0 & +c_5 - c_4 && (53)\\
d_2 = \quad & 2d_1 - d_0 & +c_5 - 2c_4 + c_3 && (54)\\
d_3 = \quad & 3d_2 - 3d_1 + d_0 & +c_5 - 3c_4 + 3c_3 - c_2 && (55)
\end{aligned}
$$

The regularity in this matrix allows an easy calculation for n-th derivatives in the second knot. For the next knot, just treat the second part of the bezier as the first part to get the control points for the third part in case there are two knots. This can be extended to any number of knots. We can now make a spline of any order with Cn continuity with as many knots as needed.

4.6.3 Final result

To complete the special case for the fifth order spline with one via-point, in the matrix below are coefficients for x_0, x_1. Note that in P_n, n goes from 0 to 7. The last two, P_6 and P_7, are the target coordinates. This result is achieved by making the multiplications of §4.5 and combining them with the continuity constraints from (52, 53, 54, 55):

$$
\begin{pmatrix} c_0 \\ c_1 \\ c_2 \\ c_3 \\ c_4 \\ c_5 \end{pmatrix} =
\begin{pmatrix}
P_0 & 0 & 0 & 0 & 0 & 0 \\
-5P_0 & 5P_1 & 0 & 0 & 0 & 0 \\
10P_0 & -20P_1 & 10P_2 & 0 & 0 & 0 \\
-10P_0 & 30P_1 & -30P_2 & 10P_3 & 0 & 0 \\
5P_0 & -20P_1 & 30P_2 & -20P_3 & 5P_4 & 0 \\
-P_0 & 5P_1 & -10P_2 & 10P_3 & -5P_4 & P_5
\end{pmatrix}
$$

$$
\begin{pmatrix} d_0 \\ d_1 \\ d_2 \\ d_3 \\ d_4 \\ d_5 \end{pmatrix} =
\begin{pmatrix}
0 & 0 & P_5 & 0 & 0 & 0 \\
0 & 0 & 5P_5 & -5P_4 & 0 & 0 \\
0 & 0 & 10P_5 & -20P_4 & 10P_3 & 0 \\
0 & 0 & 10P_5 & -20P_4 & 30P_3 & -10P_2 \\
0 & 5P_6 & -75P_5 & 140P_4 & -90P_3 & 20P_2 \\
P_7 & -5P_6 & 49P_5 & -85P_4 & 50P_3 & -10P_2
\end{pmatrix}
$$

5 Trajectory generation using BFGS

This chapter will describe the methods that were used fit the biomechanical model that generates the robot's trajectory to the minimum jerk spline model that generates an extracorporal trajectory. The optimizer used is the BFGS algorithm. This algorithm is used to find a spline with minimum jerk and to couple the VM to the generated spline from the BM. The BFGS algorithm represents part of the trajectory formation mechanism from a biological system that has successfully learned to make a monophasic movement as described in the previous chapters. There is no evidence that this representation (the BFGS optimization) also occurs in biological systems. This is why the BFGS algorithm is only described superficially.

5.1 Finding a solution using BFGS

To minimize the cost function the Broyden-Fletcher-Goldfarb-Shanno quasi-newton optimizer from the GSL-library was used. Traditional Newton optimization [15] is based on the idea that the the extremum (a minimum in our case) is characterized by its derivatives being zero. To find this extremum one needs two essential parts, the derivative and a gradient. The derivative ensures that the algorithm descend in into the right direction, the gradient ensures that its step size is aligned with the shape of the function surface. The gradient is large when it encounters a surface part that optimizes slowly, and small when the surface optimizes fast.

At first the algorithm looks at a local quadratic approximation to the nonlinear function to find its extremum. This is done by a Taylor series approximation. If the function is quadratic, a solution will be found in one single step. Otherwise, the solution found by the Taylor series is used to compute a new point. Now the Hessian matrix is used to determine the direction of the next search step, and the gradient determines its size. The Hessian matrix is the matrix of partial second derivatives. When the Hessian cannot be derived analytically because the problem is intracable (this is the case with the translation to differential robot control) a numerical estimation can be used instead. The gradient can also be numerically estimated. When the second derivatives are known, these can also be used directly (this is the case in obtaining a minimum jerk spline).

The numerical calculation of the Hessian is computationally expensive. One of the key insights to reduce computational load came from Broyden (1969). The natural solution to the Hessian update is the use of a system of linear equations expressing the ratio of change in gradient to the change in parameters. This is called the quasi-newton condition. Broyden suggested a solution in the form of a secant update. In later research he and his colleagues improved the search method. The BFGS optimizer is now widely accepted as the best performing optimizer for unconstrained gradient search. The algorithm also works for the minimization of the jerk in the splines, and produces accurate results.

5.2 Optimizing for minimum jerk

Now that we have a formula for the construction of an fifth order spline through a via point with $C3$ continuity, it is possible to extract the cost function F. This cost function must minimize the third derivative, $\frac{d^3x}{dt^3}$. When one wants to use a minimization-algorithm, the cost function must always guarantee a minimum. Since the integral of the third derivative is a measure for the total jerk, and this integral can take on negative numbers in certain parts of the formula, the integral must be squared. This resolves the problem of a guaranteed minimum.

We can now define the cost function:

$$F = \int_1^0 ((\frac{d^3x}{dt^3})^2 + (\frac{d^3y}{dt^3})^2)dt \qquad (56)$$

This is the function Hogan & Flash [30] also used in their research. The function is linearly independent, so the x and y dimensions can be optimized individually. To define this function, the final result (see §4.6.3 for more information) is used. This formula then differentiated three times ($\Delta^3 t$) and this result must be squared, and then integrated from $0 \rightarrow 1$ (t in the spline parts runs from $0 \rightarrow 1$). When we take $P = (P_0 \quad P_1 \quad P_2 \quad P_2 \quad P_3 \quad P_4 \quad P_5 \quad P_6 \quad P_7)$, we need to optimize the next matrix:

$$240PP^T \begin{pmatrix} 3 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -15 & 20 & 0 & 0 & 0 & 0 & 0 & 0 \\ 10 & -30 & 30 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -180 & 510 & 0 & 0 & 0 & 0 \\ 0 & 5 & 320 & -1920 & 1840 & 0 & 0 & 0 \\ -1 & 0 & -200 & 1200 & -2320 & 736 & 0 & 0 \\ 0 & 0 & 30 & -180 & 355 & -230 & 20 & 0 \\ 0 & 0 & -10 & 60 & -120 & 79 & -15 & 3 \end{pmatrix}$$

5.3 Optimizing the VM for the generated spline from the BM

The final matrix consist of single multiplications or quadratic multiplications ($240PP^T$). This optimization could also be calculated, but the BFGS algorithm needs no more steps that such a calculation would require. Besides, the BFGS algorithm's solution is always optimal since the gradient is known and can be solved.

The next step is fitting the VM to the spline generated. The VM generates two twitch responses that are used as speeds for the right and left wheel. These speed profiles are then translated using the math described in equations (35, 36). After this conversion the formula that describes the robot's path is linearly dependent, and it is now impossible analytically deduce the derivative. This is why the objective function has to be implicitly related to the generated result. The objective function used is a least squares fit. This is the generalised form of the least squares fit as cost function, where $\Delta x_n^2, \Delta y_n^2$ are the squared differences between the coordinates, x_n, y_n, of the path from the VM and from the BM:

$$F = \sum_{i=0}^{n} \sqrt{\Delta x_n^2 + \Delta y_n^2} \qquad (57)$$

Unfortunately there is no longer a guarantee that the BFGS algorithm finds the lowest minimum. As with most complex numerical problems, several issues concerning the initial conditions and the specific spline configuration make the BFGS algorithm unpredictable. This is discussed in §7.1.

6 Results

We now know how to generate a biomimetic trajectory that has a minimum jerk energy profile with the use of fifth order splines. From chapter 3 we know how we can model a trajectory which can be steered with muscle like effectors. To optimize the VM trajectory to fit on the BM trajectory, we make use of the BFGS algorithm. To test this procedure of optimization, a number of experiments are performed and discussed in this chapter.

6.1 Implementation

Both models are programmed in C++, and tested under Windows XP. With some slight modifications the code can be ported to GNU C++. For the BFGS algorithm the 'GSL' package, which is the 'GNU scientific library', 'from http://sources.redhat.com/gsl/' was used. The BFGS algorithm uses the gsl_multimin_fdfminimizer_type that comes from gsl_multimin.h. This type of minimizer can be used for several optimization algorithms. In this case the minimizer was initialized to gsl_multimin_fdfminimizer_vector_bfgs.

The program consist of three classes, class MinJerkSpline, class PathModel and class Spline. The main program reads three pairs of $< x, y >$ coordinates from the command line that make up the starting coordinate, the via point coordinate and the target coordinate. The coordinates are passed to a class instance of MinJerkSpline, which calculates the control points for the splines that connect the three points. This is an optimization from the BFGS algorithm for formula (56). After this, the control points are passed to a class instance of Spline, where the target coordinates are calculated. These coordinates are then passed to an instance of class PathModel, that fits the VM to the target, using the least squares method. After this the generated data is stored. The plots and tables that are relevant to the tests performed can be found in the appendix.

6.2 Monophasic trajectories

6.2.1 Goal

The models were tested on point to point movement and point to point with a via-point. The first experiment tests the trajectory formation model on point to point movement with only an agonist muscle, using formula (20) for the twitch response. Four distances were chosen, $20, 30, 50$ and 100. because the distance was dimensionless, the results can be related to any real-world distance, and any possible muscle.

The purpose of the experiment is to show a relation between different speed profiles for the VM and the BM when travelling over different distances in the same time period. This time period is normalized to 1. The time could also be adjusted to model a real-world situation. The fact that the same time period is taken reflects findings from Hogan & Flash [30], who researched the fact that targeted monophasic movement has a fixed time period that is independent of the travelled distance with monophasic movement.

6.2.2 Result

The speed profiles of the VM and the BM are in Figure 36(a) in Appendix A. Numerical data concerning the experiment can be found in tables 1, 2 and 3. The paths generated are depicted in Figure 36(b). The experiment shows data for travel distances $20, 30, 50, 100$, where the endpoints are indicated with a dot on the x-axis. The dots have been added because the splines are straight lines, which makes them invisible in Figure 36(b).

6.2.3 Conclusion

The results show us that the travelled path fits very well in cartesian coordinates. The plots in Figure 36(b) show an accurate fit, even though it may seem otherwise because the y-axis represents a very small distance in relation to the x-axis. From this stretch in the y-axis we can see that the algorithm cannot find an exact minimum. The algorithm is therefore stopped after 1000 iterations. Even then the results are still converging toward a better fit.

Besides we can conclude that the fit of the VM in terms of its speed profile do not fit the minimum jerk model. This is not very surprising, since the VM is the trajectory that can be generated using one muscle. An improvement would be to add a second muscle that can function as an antagonist. This would enable the speeds from the VM to drop faster and approach 0 where the spline's speed profile does (at $t = 1$ in Figure 36(a)).

In table 2 in Appendix A it becomes clear that the coordinates of the control points are proportional to the targeted distance. because the spline generated by the BM can de decoupled in x and y coordinates, this also goes for other types of trajectories. This suggest that the VM should be able to change the spring time constants in combination with its generated force (the pulse magnitude in our case) in order to be able to generate minimum jerk optimized trajectories.

6.3 From monophasic to biphasic movement

6.3.1 Goal

The second test shows the difference between monophasic and biphasic movement. To test these differences, the test started out with a monophasic 'start-via point-endpoint' trajectory with length 40. After this, the via-point is shifted up in the y-direction with an increasing height. The goal of this experiment was to see the behavior of the BM and the VM fit with an increasingly more distant via-point. Because the VM can only generate monophasic movement, it can only give us an insight of how the biphasic properties affect a monophasic movement.

6.3.2 Result

The resulting speed profiles generated by both the BM and the fitted VM can be found in Figure **??**. The x, y plots are in Figure **??**. Numerical data concerning the experiment can be found in tables 4, 5 and 6. The plots are made with an end point at $x = 40$, and via point y-offsets $0, 1, 5, 10, 15, 20, 30, 40, 80$.

6.3.3 Conclusion

Since the VM can only produce monophasic movement, the increasing difference between the targeted BM and the fitted VM is not surprising. In §7.2 a proposal will be made to increase the VM's predictive power to simulate minimum jerk optimal movement with a biphasic nature. This experiment clearly shows us that the VM is not capable of generating biphasic movement. Hogan & Flash [41] showed us that movements of this type can be separated in two distinct movements. In the experiment this means that when the y-offset from the via point reaches a certain point, the movement should be composed of two muscle actions. This implies that a via point with too much vertical offset from the line from starting point to end point should be considered as a path with two trajectories. The criterion for the decision to consider the via point as an endpoint could come from the optimality criterion. When the cost function becomes too large, the path can be cut in two pieces. This criterion has to be established experimentally.

6.4 Trajectory formation with arbitrary via points

6.4.1 Goal

The first two test reveal that the VM is capable of generating monophasic point to point trajectories that closely resemble the trajectory generated with the BM. When the VM is used for the generation of biphasic movement, it is capable of generating a trajectory that has about the same cartesian coordinates as the BM. More important is the generated speed profile, which does not have a good fit to the speed profile generated with the BM. This is due to the limitations of the VM, that is only capable of generating reasonable approximations with monophasic movement. This test looks at the VM's performance on 24 different trajectories generated by the BM. These trajectories shed light on the whole spectrum of possible trajectories when moving on a 'start-via point-endpoint' trajectory.

6.4.2 Result

The idea behind the chosen trajectories is that they only start to differentiate after the via point. When one takes this via point as a center of departure, one could draw concentric circles on which one could place endpoints that determine the shape of the total trajectory. By varying the angle and radius one can extract a number of points that are of interest. The lower half of this imaginary circle can be left out due to symmetry, and the large angles are left out because they tend toward monophasic movement. Figures 38 and 39 give an idea of the VM's behavior with different endpoints.

6.4.3 Conclusion

When we look at Figures 38 and 39 we can clearly see that the VM fails to generate a correct path at sharp angles. At the large angles, the VM starts to generate trajectories that start to look like the targetted trajectories. Figure 39(e) illustrates the effect of taking the least squares as cost function. The curling ends from the paths generated with the BM are a result from this least squares fit, because this cost function allows a misfit in order to keep the most points at the best place. This suggests that the cost function should be altered to put weight on the angular differences, speed differences and endpoint coordinates. The weight of these features can be determined by constraints for the trajectory to be generated. When a robot moves fast and needs to plan many trajectories, angular differences could be an issue. When a robot needs to move very precise, the coordinates of the via point and endpoint become important constraints.

7 Conclusion

7.1 General discussion

The experiments show us that the BFGS algorithm always generates a solution for the BM. Fitting the VM to the BM with BFGS has proven to be more difficult. One could even question the very nature of this experiment by stating that the generated spline is already a good trajectory generator that has no need to be fitted with a visco-elastic muscle model. Even though I would not get into a discussion about this with an engineer, there are severely reasons why this is a useful test.

- This experiment is designed to look at the way animals generate movement. Smooth movements with low speeds at the starting point, via point and endpoint are energy preserving (minimum jerk), allow the animal to adjust its trajectory at the right moment and are safer in case of a collision. Besides, the energy surplus from the mechanical smoothing of the trajectory is stored in the spring like elements in the muscle. Ofcourse there is no energy storage in a robot, but the robot could be altered to allow such an energy storage by the use of mechanics that work in the same way a muscle does. In essence, even though the spline provides the mathematical advantages, the VM should make up the mechanical properties that would have to fit the BM.
- Another reason to perform this test is to see how an internalized model can reduce computational load. This experiment made use of the BFGS algorithm, which can solve a wide variety of mathematical formulae numerically. The experiment clearly shows commonalities that could be incorporated in the internalized model for trajectory formation (for example the effect of end point distance and its relation to the activation levels in the VM). In this respect, the EPH tries to accommodate these features by stating that the properties of a VM are known a priori and can be actuated by a neural representation of a less complex mechanical representation. Even though the EPH is still under debate, is becomes clear that investigation of a VM in respect to the BM could uncover a simplified representation for a multiple degree of freedom problem often found in animals.
- Last but not least it has to be said that the model used in this experiment is a great oversimplification of the model intended by the VM. It has already been discussed that modeling the muscle is no easy task. Some critical decisions concerning simplification have to be made. Even if we envision a more complicated model it remains to be seen if the results will be satisfying. In spite of this, the model does give us an insight in how animal muscles behave and how they might be controlled by the brain. This opens up the door to the design of robotic limbs with biological properties. Designing a robotic limb may even incorporate advantages that may otherwise remain concealed.

When we look at these reasons to investigate how we could steer a biomechanical model, it becomes clear what the requirements of a autonomous system should be. When we want an autonomous system capable of interactions similar to human and animal interactions with their environment, the design of such a system should incorporate biomechanical requirements where needed.

A big part of this MSc thesis deals with the problem of choice when trying steer such a system. There are many different ways to represent a motor command model that has its biological equivalent in neural motor command. Without this discussion there would be no good insight in the possible solutions of such a problem. In this MSc thesis, the BFGS algorithm represents the solving mechanism that connect empirical data concerning observed trajectories to the biomechanical muscle model.

The next step would be the construction of a robot that incorporates the best working theory of biomechanics to test motor command models other than a representation by the BFGS algorithm. A good starting point would be the EPH since it is one of the most accurate models.

7.2 Future Research

Even though this experiment investigates biomimetic robot movement with a biological inspiration, no real-life test has been performed. Only in the real word the advantages and disadvantages can be revealed. This MSc thesis' focal point is a differentially steered robot. Ofcourse it would be better to test the model on a robotic arm, and possibly on a humanoid robot. Values determining the cost function could be optimized using a neural network that could learn to use minimum jerk optimized movement by trial and error.

Before these kind of tests can take place, a more elaborate muscle model must be developed. This model should not only make use of a more sophisticated muscle model, but also take into account the mechanical reality of a robot. The simple addition of an antagonist and a mass could prove to be a good supplement for the muscle model used in this thesis. See Figure 33 for the representation of a mechanical model for an agonist and an active or passive antagonist. The impulse response of these systems is visible in Figure 34 and 35. In the second figure is the impulse response where the antagonist has passive properties, which means that it stretches and produces no active force.

The first part (from $t = 0$ to $t = 35$) of Figure 35 clearly shows a response that corresponds to the expected speed profile as seen in the splines from the BM. In spite of this correspondences, the VM would need to incorporate more muscle characteristics, like the nonlinear aspect of a muscle, the translation from the joints and more complicated dynamic interactions like the effect of muscle mass.

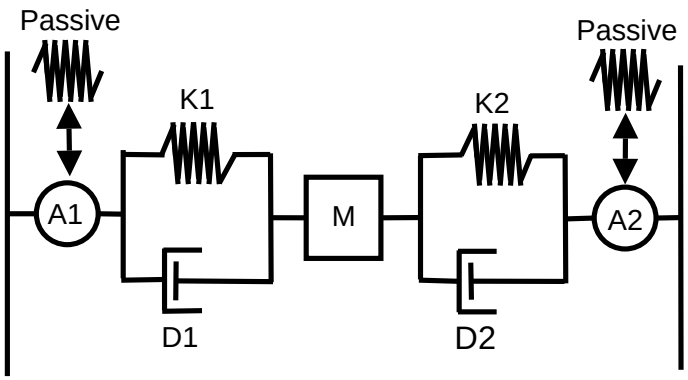


Figure 33: A more complicated muscle model with active elements $A1, A2$. These elements respond like the spring damper elements next to the center mass. When a muscle is passive, these elements may be represented by a spring element. The parts left and right of the mass represent the agonist and the antagonist.

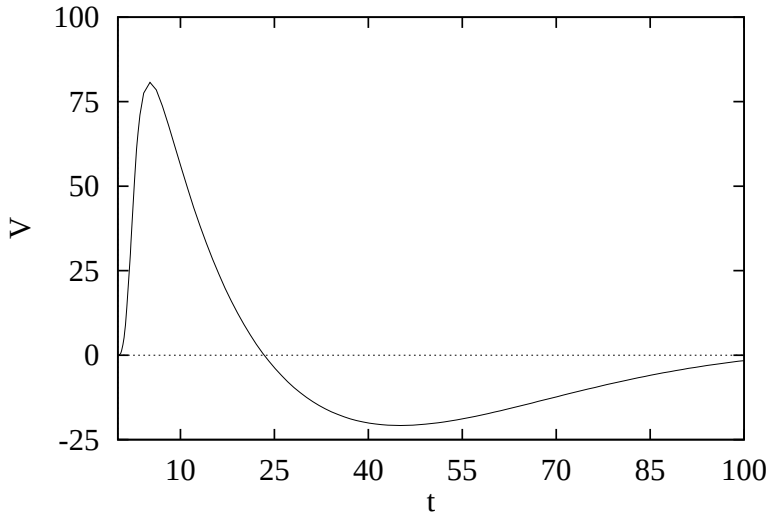


Figure 34: Impulse response from electric circuit equivalent to the mechanic model from Figure 33. Both the agonist and the antagonist are stimulated with a pulse.

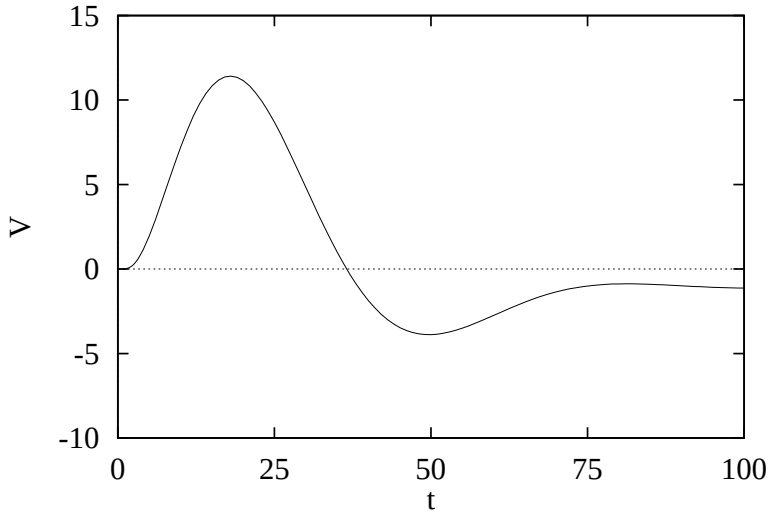
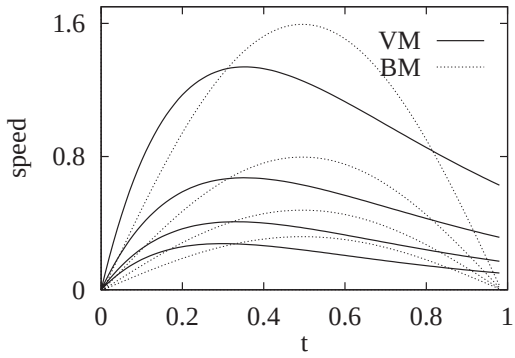


Figure 35: Impulse response from electric circuit equivalent to the mechanic model from Figure 33. In this response the agonist is stimulated, and the antagonist reacts with its passive properties (replace antagonist's active element with a spring in the mechanical model).

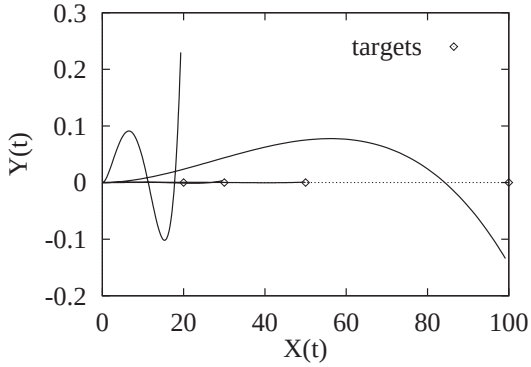
A Appendix

In Figure 36(a) are the speedprofiles for the tested monophasic trajectories. The trajectories are straight line distances with the next lengths: 20, 30, 50, and 100. The dotted lines are the speeds from the BM, the solid lines are from the VM. The solid lines from the VM clearly show an overshoot in the last part, which means that the end speeds are too high (they are supposed to end in zero speed). This is caused by the fact that the surface area of both models needs to be equal in order to cover the right distance measured in x, y coordinates. The overshoot can be attributed to the fact that the muscle model is incapable of generating the curvature in the speedprofile from the BM. This could be accomplished by the introduction of an antagonist muscle to counteract the slow decay from the typical 'twitch response' from the VM.

In Figure 36(b) are the trajectories from the speed profiles in Figure 36(a). Again, the dotted lines are the trajectories from the BM. They are straight trajectories, that is why the dots mark the endpoints. The curved solid lines are the trajectories generated by the VM. Note that the deviation from the BM trajectories are less than a percent, the y-axis is stretched to show the differences in detail. They would be fitted perfectly if the left and right wheel speeds are equal. Since the BFGS algorithm uses graded descent, and the hessian is an approximation, no exact fit can be made. The BFGS algorithm will slowly converge to equal speeds, but never reach an exact solution.



(a) Monophasic speed profiles for travel speed along the trajectory for the VM and the BM with different travel distances in arbitrary units. Increasing height means increasing point to point distances (20, 30, 50 and 100).

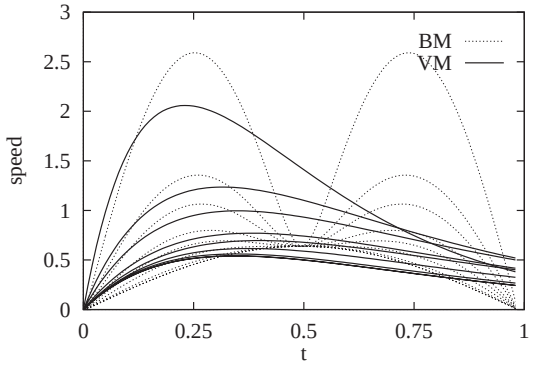


(b) Travelled paths for point to point monophasic movement in arbitrary units. The BM travels along a straight line in the x-direction for target distances (20,0), (30,0), (50,0) and (100,0). The curved lines are the trajectories generated by the VM. Note that the deviation from the BM trajectories are less than a percent, the y-axis is stretched to show the differences in detail.

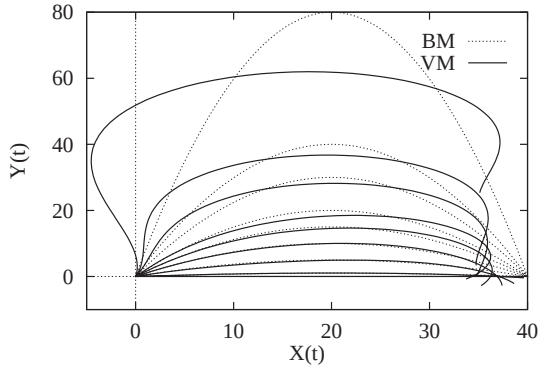
Figure 36: Test results from monophasic movement. Values in arbitrary units.

The next figures show the speed profiles (Figure 37(a)) and the trajectories (Figure 37(b)) for a test where a trajectory is generated that starts out as a straight line. After the straight trajectory a via point with an increasing y-offset is introduced. At first the offset is 0, thereafter $1, 5, 10, 15, 20, 30, 40, 80$. When the y-offset gets larger, the speed profile from the BM becomes biphasic. When the via point becomes more distant from the initial straight line the peaks at the left and right side in the speedprofiles become more distinct (this indicates the biphasic property).

The fitted trajectory from the VM starts out with a good fit, but as the y-offset increases the fit becomes increasingly inaccurate. In Figure 37(a) it becomes clear that the VM can only generate trajectories with a monophasic profile, which can be seen by the fact that the VM fits have a fit that is adjusted to one peak in the speedprofile from the BM (the first, left peak in the dotted lines). This problem can be solved by the introduction of more muscles in the VM.



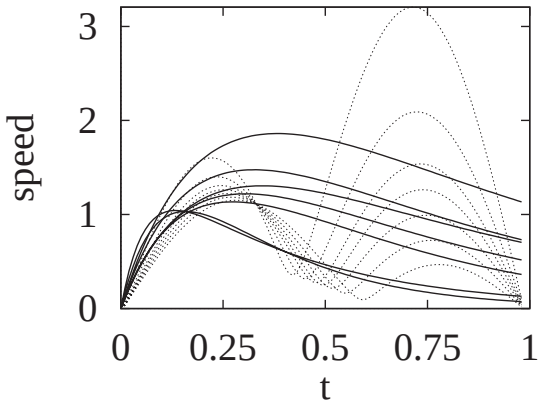
(a) Speed profiles for biphasic movements with a via point. The dotted lines are the trajectories from the BM, the solid lines are from the VM. The starting and end points are all equal. The via point is in between the start and end point and increases in y-offset with each test. The start and end points have coordinates $(0, 0)$ and $(0, 40)$, the viappoints $(20, \{0, 1, 5, 10, 15, 20, 30, 40, 80\})$



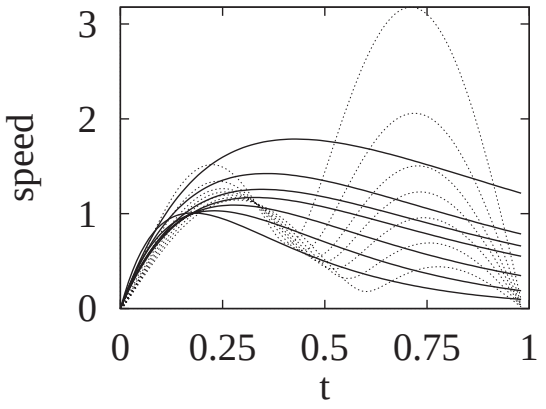
(b) Trajectories for biphasic movement with a via point, values in arbitrary units. The dotted lines are the trajectories from the BM, the solid lines are from the VM. The fit of the VM gets worse with more distant via points. This is caused by the fact that the VM can only produce a monophasic speed profile that has no exact fit to the trajectory generated by the BM.

Figure 37: Test results from monophasic movement. Values in arbitrary units.

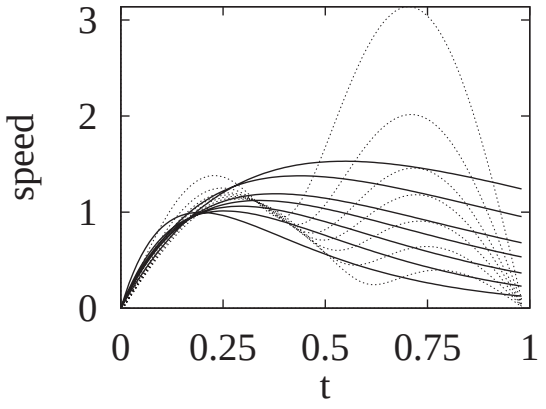
The next two pages show the results from the test with arbitrary trajectories with a via point (see §6.4 for more information). In Figures 38(a), 38(b), 38(c), 38(d), 38(e) are the speed profiles for trajectories with an increasing angle between the start-via point and via point-target lines. In Figures 39(a), 39(b), 39(c), 39(d), 39(e) are the trajectories that belong to the speedprofiles. Both tests essentially show that a sharp angle, whereby the biphasic characteristics are getting more distinct, causes a less accurate fit of the VM on the BM. This property is not only visible in the speed profiles, but also in the trajectories. An interesting result is the curling effect of the VM in Figure 39(a). The curl is caused by the limitations from the VM. Again, these curling effect would decline with the introduction of a more complicated muscle model.



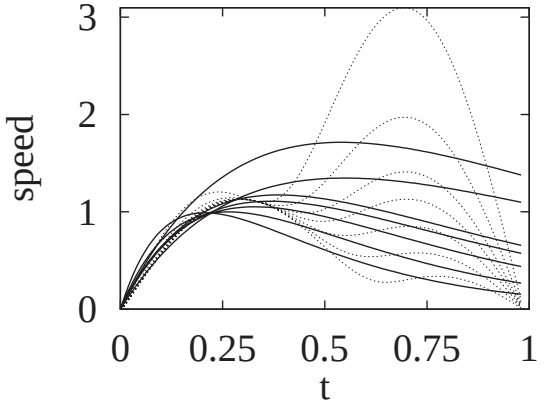
(a) angle: 22^o



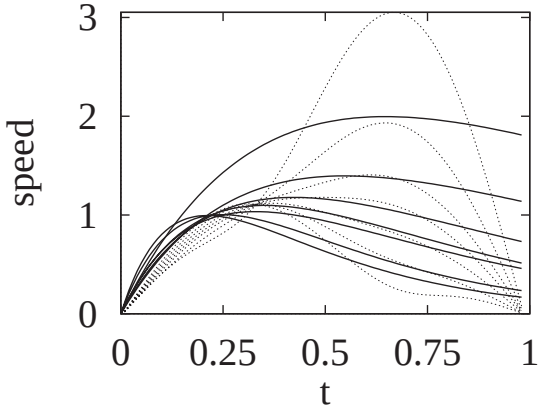
(b) angle: 45^o



(c) angle: 67^o

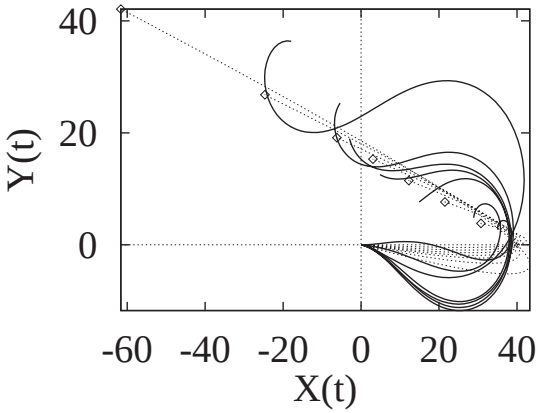


(d) angle: 90^o

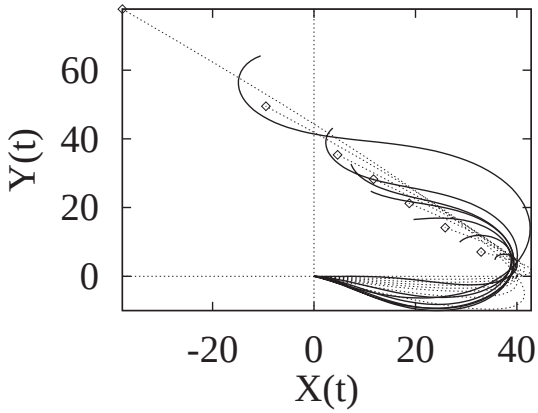


(e) angle: 135^o

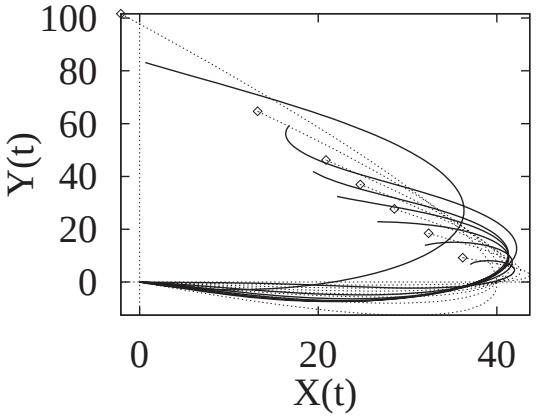
Figure 38: Speed profiles for trajectories for next angles in start-via-end point: $22, 45, 67, 90, 135$. All plots are measured in arbitrary units.



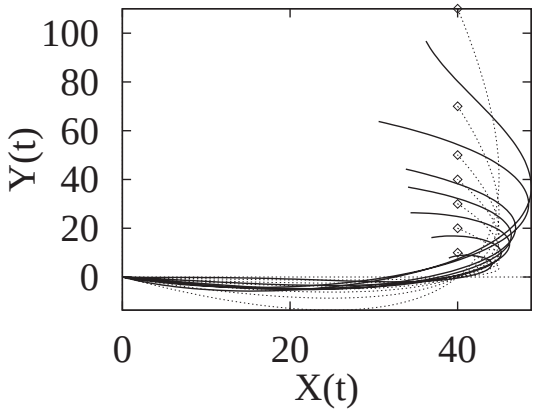
(a) angle: 22^o



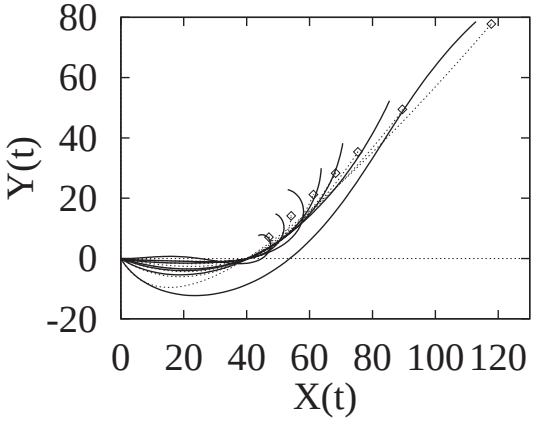
(b) angle: 45^o



(c) angle: 67^o



(d) angle: 90^o



(e) angle: 135^o

Figure 39: Plotted paths in cartesian coordinates for trajectories for next angles in start-via-end point: $22, 45, 67, 90, 135$. All plots are measured in arbitrary units.

The next pages show numerical data from the first two tests. The first three tables (Tables 1, 2, 3) show the results for the first test with the monophasic trajectories, the following three (Tables 4, 5, 6) show the results for the second test with the biphasic trajectories. The first tables (Tables 1, 4) show the coordinates from the BM and the VM for start, via and end points (there are no via points in the first test). The last column shows the value from the cost function. Tables 2 and 5 contain the control points from the BM splines. All splines contain a knot (the via point). Two splines connect the knot to the starting point and the end point. The first spline that connects the starting point to the knot has control points labelled c_n. The second part has control points labelled d_n. Since the first test has a straight trajectory, there is no need for a spline in the y-axis direction (see Table 2). The second test with the biphasic movement has a curved line, that is why are splines for the y-axis are added. Tables 3 and 6 show the results for the VM model. The values, $< \tau_1, \tau_2, A >$ are from equation (20). These values make up the muscle type control. The time constants determine visco-elastic properties needed for the type of trajectories. These values are now used for control of a differentially steered robot, but they could determine visco-elastic properties for a robot that uses muscle like control.

How the cost function relates to change in distance in the first test can be seen in Figure 40(b) for the BM and in Figure 40(a) for the VM. Figures 41(b) and 41(a) show the same for the second test. Note that even though the results for the BM are optimal, the graph shows an increase in the cost function. The cost function returns a value that is related to the 'jerkiness' of the trajectory, which increases with an increase in distance. The values from the least squares fit with the VM are much larger, but show a quadratic increase similair to the increase from the BM model. Note that the VM graph from the monophasic trajectories has a greater resemblance to the monophasic BM graph than the VM graph from the biphasic test to the corresponding biphasic graph for the BM. These irregularities show that the VM model becomes more unpredictable when a via point is added to the trajectory, which is to be expected in case of a biphasic speedprofile.

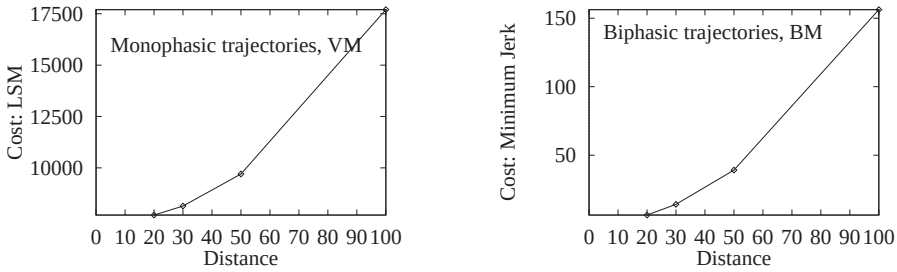


Figure 40: Relation of cost function to change in distance.

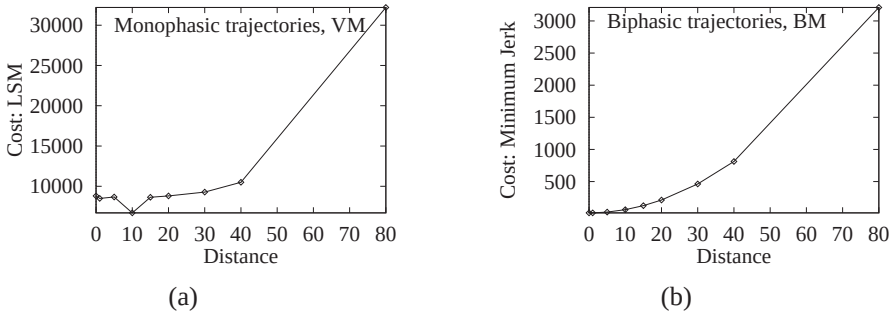


Figure 41: Relation of cost function to change in y-offset.

Table 1: Results monophasic trajectories. In the first columns are the coordinates, in the last the error as described in §6.

	start		end		cost (F)	
	x	y	x	y	initial	final
BM	0.00	0.00	20.00	0.00	$7.36 \cdot 10^6$	6.25
VM	0.00	0.00	19.31	0.22	$2.05 \cdot 10^4$	$7.70 \cdot 10^3$
BM	0.00	0.00	30.00	0.00	$8.61 \cdot 10^6$	14.06
VM	0.00	0.00	29.50	0.00	$3.82 \cdot 10^4$	$8.14 \cdot 10^3$
BM	0.00	0.00	50.00	0.00	$3.79 \cdot 10^6$	39.06
VM	0.00	0.00	49.78	0.00	$9.58 \cdot 10^4$	$9.70 \cdot 10^3$
BM	0.00	0.00	100.00	0.00	$7.37 \cdot 10^5$	156.25
VM	0.00	0.00	99.88	0.00	$3.70 \cdot 10^5$	$1.77 \cdot 10^4$

Table 2: Values found for splines, control points c_n for the first part of the spline, d_n for the second part. Only movement in x direction.

Δx	c_1	c_2	c_3	c_4	c_5	c_6	d_1	d_2	d_3	d_4	d_5	d_6
20	0.00	0.00	1.25	3.75	6.88	10.00	10.00	13.13	16.25	18.75	20.00	20.00
30	0.00	0.00	1.88	5.63	10.31	15.00	15.00	19.69	24.38	28.13	30.00	30.00
50	0.00	0.00	3.125	9.38	17.19	25.00	25.00	32.81	40.63	46.88	50.00	50.00
100	0.00	0.00	6.25	18.75	34.375	50.00	50.00	65.63	81.25	93.75	100.00	100.00

Table 3: Values found for the VM. These values fill in the two time contstants τ_1, τ_2 and the activation A from equation (20).

Δx		τ_1	τ_2	A
20	right	19.40	48.23	24.54
	left	25.29	35.68	22.89
30	right	32.33	34.10	36.88
	left	32.97	33.44	36.87
50	right	35.14	35.31	64.40
	left	35.19	35.25	64.40
100	right	35.87	35.87	130.37
	left	35.90	35.89	130.48

Table 4: Results 'monophasic to biphasic' trajectories. In the first columns are the coordinates, in the last the error as described in §6.

	start		via		end		cost (F)	
	x	y	x	y	x	y	initial	final
BM	0.00	0.00	20.00	0.00	40.00	0.00	$9.92 \cdot 10^5$	12.5
VM	0.00	0.00	20.27	0.00	39.65	0.00	$6.33 \cdot 10^4$	$8.81 \cdot 10^3$
BM	0.00	0.00	20.00	1.00	40.00	0.00	$4.07 \cdot 10^5$	13.00
VM	0.00	0.00	21.28	1.05	39.63	−0.38	$6.38 \cdot 10^4$	$8.50 \cdot 10^3$
BM	0.00	0.00	20.00	5.00	40.00	0.00	$3.83 \cdot 10^5$	25
VM	0.00	0.00	21.37	4.95	38.64	−2.11	$6.63 \cdot 10^4$	$8.76 \cdot 10^3$
BM	0.00	0.00	20.00	10.00	40.00	0.00	$7.80 \cdot 10^4$	62.5
VM	0.00	0.00	21.00	10.01	37.40	−2.73	$7.04 \cdot 10^4$	$6.70 \cdot 10^3$
BM	0.00	0.00	20.00	15.00	40.00	0.00	$2.84 \cdot 10^6$	125
VM	0.00	0.00	20.93	14.63	35.27	−2.75	$7.56 \cdot 10^4$	$8.64 \cdot 10^3$
BM	0.00	0.00	20.00	20.00	40.00	0.00	$3.92 \cdot 10^4$	213
VM	0.00	0.00	21.40	18.47	33.73	−0.74	$8.22 \cdot 10^4$	$8.81 \cdot 10^3$
BM	0.00	0.00	20.00	30.00	40.00	0.00	$5.33 \cdot 10^6$	463
VM	0.00	0.00	21.53	28.14	34.82	0.12	$1.24 \cdot 10^5$	$9.28 \cdot 10^3$
BM	0.00	0.00	20.00	40.00	40.00	0.00	$2.49 \cdot 10^6$	813
VM	0.00	0.00	22.41	36.47	34.75	3.38	$1.25 \cdot 10^5$	$1.05 \cdot 10^4$
BM	0.00	0.00	20.00	80.00	40.00	0.00	$4.79 \cdot 10^6$	$3.21 \cdot 10^3$
VM	0.00	0.00	25.02	60.5	35.12	25.36	$2.98 \cdot 10^5$	$3.22 \cdot 10^4$

Table 5: Values found for splines, control points c_n for the first part of the spline, d_n for the second part. Movement in x and y direction.

Δy		c_1	c_2	c_3	c_4	c_5	c_6	d_1	d_2	d_3	d_4	d_5	d_6
0	x	0.00	0.00	2.50	7.50	13.75	20.00	20.00	26.25	32.50	37.50	40.00	40.00
	y	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
1	x	0.00	0.00	2.50	7.50	13.75	20.00	20.00	26.25	32.50	37.50	40.00	40.00
	y	0.00	0.00	0.25	0.75	1.00	1.00	1.00	1.00	0.75	0.25	0.00	0.00
5	x	0.00	0.00	2.50	7.50	13.75	20.00	20.00	26.25	32.50	37.50	40.00	40.00
	y	0.00	0.00	1.25	3.75	5.00	5.00	5.00	5.00	3.75	1.25	0.00	0.00
10	x	0.00	0.00	2.50	7.50	13.75	20.00	20.00	26.25	32.50	37.50	40.00	40.00
	y	0.00	0.00	2.50	7.50	10.00	10.00	10.00	10.00	7.50	2.50	0.00	0.00
15	x	0.00	0.00	2.50	7.50	13.75	20.00	20.00	26.25	32.50	37.50	40.00	40.00
	y	0.00	0.00	3.75	11.25	15.00	15.00	15.00	15.00	11.25	3.750	0.000	0.000
20	x	0.00	0.00	2.50	7.50	13.75	20.00	20.00	26.25	32.50	37.50	40.00	40.00
	y	0.00	0.00	5.00	15.00	20.00	20.00	20.00	20.00	15.00	5.00	0.00	0.00
30	x	0.00	0.00	2.50	7.50	13.75	20.00	20.00	26.25	32.50	37.50	40.00	40.00
	y	0.00	0.00	7.50	22.50	30.00	30.00	30.00	30.00	22.50	7.50	0.00	0.00
40	x	0.00	0.00	2.50	7.50	13.75	20.00	20.00	26.25	32.50	37.50	40.00	40.00
	y	0.00	0.00	10.00	30.00	40.00	40.00	40.00	40.00	30.00	10.00	0.00	0.00
80	x	0.00	0.00	2.50	7.50	13.75	20.00	20.00	26.25	32.50	37.50	40.00	40.00
	y	0.00	0.00	20.00	60.00	80.00	80.00	80.00	80.00	60.00	20.00	0.00	0.00

Table 6: Values found for the VM. These values fill in the two time constants τ_1, τ_2 and the activation A from equation (20).

Δx		τ_1	τ_2	A
0	right	34.30	34.62	50.62
	left	34.42	34.50	50.62
1	right	34.45	34.46	50.55
	left	31.14	38.40	51.18
5	right	34.67	34.67	52.38
	left	23.85	53.84	58.16
10	right	37.12	37.12	60.99
	left	31.33	46.89	66.27
15	right	39.28	39.29	72.65
	left	40.27	41.45	79.09
20	right	44.67	31.61	77.64
	left	38.98	39.18	83.77
30	right	19.93	64.69	105.99
	left	36.01	36.01	100.12
40	right	57.09	18.12	117.60
	left	32.53	32.53	111.90
80	right	33.48	15.72	133.21
	left	23.69	23.69	134.12

A.1 Java applet

To get a first impression of the muscle model and the different possible translations to robot control, an applet gives insight in the possible trajectories. The VM has been implemented in java. The applet is divided in a left and a right side. The part on the right side has two graphic displays of the input to the robot. The upper and lower graphics canvas are the power inputs for the right and the left wheel, for the differentially controlled robot. By changing the values on the slide bars the formula's input and onset values can be manipulated. By checking on or off the radio buttons, several changes can be made to the general features of the path generator.

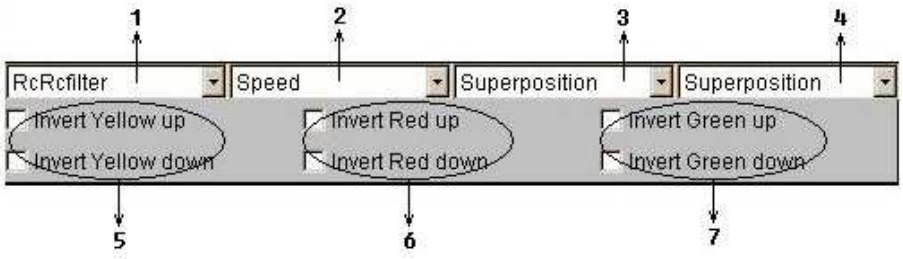


Figure 42: Upper left corner of Java Applet

Applet layout.
In the left upper corner there are several choice boxes and some radio buttons, as can be seen in Figure 42. The four choice boxes (1 to 4 in Figure 42) have the following selections:

1. Choice of formula used to simulate muscle contraction.

   (a) RcRcFilter: Use the second order equation with two time constants as found in (20).
   (b) Sinus: Use a sinoid formula to simulate the response to a steering impulse.
   (c) Gauss: Use Gauss formula to simulate response to muscle activation. This normal distribution is a muscle reaction which can be found in real muscles as response to neural input. The bell shaped curve occurs in many biological systems and closely resembles the speed profile of a monophasic minimum jerk trajectory.

2. Choice of formula used to map muscle tension from the right graphic canvasses to wheel speed.

   (a) Speed: Model's response is represented as wheel speed. The response can be seen in the right graphic canvas. The yellow line is the response. The upper yellow response is the left-wheel speed. The lower yellow response is the right-wheel speed.
   (b) Cartesian: Equal use of the response as with choice 'speed', but now the yellow response represents x and y coordinates instead of left-wheel and right-wheel speed.
   (c) Energy: Equal use of the response as with choice 'speed', but now the yellow response represents energy instead of speed. This conversion complies to $E = \frac{1}{2} \cdot m \cdot V^2$.

3. Choice indicating if superposition (addition) or multiplication is used as a basis for the agonist (yellow) antagonist (green) interaction, for the lower graphics canvas.

4. Same function as the previous choice button, but now referring to the upper graphics canvas.

The radio buttons, in Figure 42 labeled as number 5, 6 and 7, invert the time scales for the formulas in the right graphic canvasses. The buttons in 5 can invert the yellow (agonist) upper and lower formulas. Buttons under 6 do this for red (influence on both upper and lower agonist), under 7 for green (antagonist).

Furthermore there are two buttons in the bottom left side. The go button starts the simulation, in which the robot travels the calculated path visible in the graphic canvas above. Since the robot has a variable speed it is interesting to see its speed change as is travels the path. The reset button has an obvious function, resetting all values in the applet to their original state.

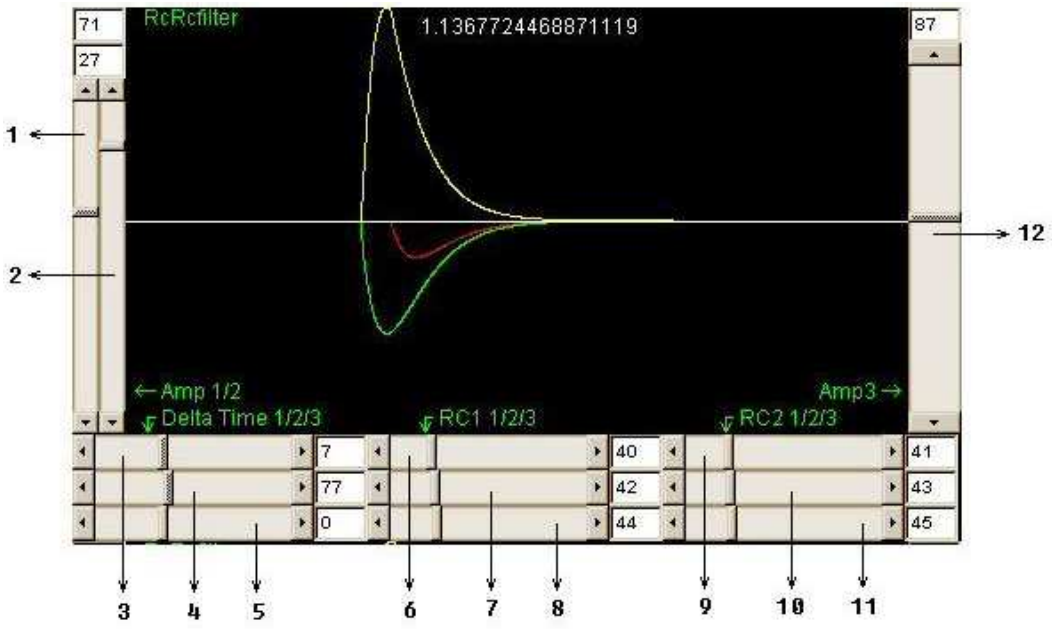


Figure 43: Graphics canvas from the java applet displaying muscle responses needed to steer the robot. Slidebars $1, 2$ and 12 control the amplification of the formulas used in the model. Slidebars $3, 4$ and 5 allow the user to interact with the onset time of the formulas. Slidebars $6, 7, 8$ and $9, 10, 11$ control the value of (mostly timeconstants) variables in the formulas.

The Right upper graphic canvas (see Figure 43) displays the formula to steer the left-wheel. The same goes for the lower right canvas, except for the fact that this canvas represents the right-wheel speed. Surrounding the canvas there are several slide-bars. Slide bars 1,2 and 12 are the amplification part. In (20) the amplification is $\frac{1}{\omega_1 - \omega_2}$. Slide bar 1 amplifies the yellow (agonist) line, 2 the red line (influence), and 3 the green line (antagonist). Slide bars 3,4 and 5 can be used to time-shift the yellow, red and green line. Slide bars 6,7 and 8 can be used to alter the first time constant. This was represented as ω_1 in (20). The last three, 9,10 and 11, can alter the second time-constant, ω_2.

Formulas

The formulas used can be found in the class-structure Formulas.java. This class permits the user to send in the 'state' of the 'muscle'. It then returns the next time step, using the appropriate formula. There are three formulas in the the class formulas. They all use the same format, so new formulas can easily be added to the class. In effect, this is simulating the linear process in a discrete fashion.

These are the formulas currently embedded in the program:

- formula (20): $T(t) = \frac{1}{\omega_1 - \omega_2} e^{-\frac{1}{\omega_1}t} - \frac{1}{\omega_1 - \omega_2} e^{-\frac{1}{\omega_2}t}$
- sinus: $T(t) = \sin(t)$
- Gauss distribution: $T(t) = \frac{1}{\sigma\sqrt{2\pi\sigma^2}} e^{\frac{-(t-\mu)^2}{2\sigma^2}}$

New formulas can be added by making a new private function in the class 'formulas'.

Path generation
Another feature included in the formulas-class is path calculation. Three types can be chosen from the choice box in the applet: 'speed', 'Cartesian' and 'energy'.

User interface
The user interface is based on the graphics packages from java. There are three canvasses in the program. Two indicate the muscle response, one depicts the robot's path. The path is stored in a temporal buffer containing the pixels that lay on it. The robot's directional orientation is also stored in this buffer. The class containing these buffers is called Path canvas. The input could be sensory input, and the visualization could be actual motor control.

List of Figures

References

[1] Bizzi E. Abend W.E. and Morasso P. Human arm trajectory formation. *Brain*, 105:331–348, 1982.

[2] Fagg A.H., Sitkoff N., and Barto A.G. A model of cerebellar learning for control of arm movements using muscle synergies. *Proceedings of the IEEE International Symposium on Computational Intelligence in Robotics and Automation*, 1:6–12, 1997.

[3] Hodgkin A.L. and Huxley A.F. A quantitative description of membrane current and its application to conduction and excitation in nerve. *J. Physiol. (Lond.)*, 117:500–544, 1952.

[4] Turing A.M. On computable numbers, with an application to the entscheidungsproblem. *Proceedings of the London Mathematical Society*, 2(42):230–265, 1936.

[5] Hill A.V. The heat of shortening and the dynamic constants of muscle. *Proc. Roy. Soc.*, 126 B:136–195, 1938.

[6] Hill A.V. *First and last experiments in muscle mechanics*. Cambridge university press: Cambridge, 1970.

[7] Chapple W Hogan N Bizzi E, Accornero N. Posture control and trajectory formation during arm movement. *J Neuroscience*, 4(11):2738–44, 1984.

[8] Fox D., Burgard W., and S. Thrun. The dynamic window approach to collision avoidance. *IEEE Robotics and Automation*, 4:23–33, 1997.

[9] Winter D.A. *Biomechanics and Motor Control of Human Movement*. Wiley-Interscience, 1990.

[10] Rumelhart D.E., Hinton G.E., and Williams R.J. Learning representations by back-propagating errors. *IEEE Journal Of Robotics And Automation*, 1:533–536, 1986.

[11] Merfeld D.M. Must all action halt during sensorimotor mismatch ? *Behavioral and Brain Sciences*, 24(1):189–190, 2001.

[12] Hebb D.O. *The organization of behavior*. New York: Wiley, 1949.

[13] Bizzi E., F.A. Mussa-Ivaldi, and S. Giszter. Computations underlying the execution of movement: a biological perspective. *Department of Brain and Cognitive Sciences, Massachusetts Institute of Technology*, 253(5017):287–91, 1991.

[14] Fikes R. E. and Nilsson N. Strips: A new approach to the application of theorem proving to problem solving. *Artificial Intelligence*, 5(2):189–208, 1971.

[15] Kreysig E. *Advanced engineering mathematics*. Wiley, new York, 1999.

[16] Rosenblatt F. The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review*, 65:386–408, 1958.

[17] Klute G. *Artificial Muscles: Actuators for Biorobotic Systems*. Ph.d. thesis, University of Washington, Department of Bioengineering, 1999.

[18] Massey J.T. Georgopoulus A.P., Kalaska J.F. Spatial trajectories and reaction times of aimed movements: Effects of practise, uncertainty and change in target location. *J. Neurophysiol*, 46:725–743, 1981.

[19] Walter W. Grey. An imitation of life. *Scientific American*, 1:42–45, 1950.

[20] Ostry D.J. Sanguineti V. Gribble P.L. and LaBoissiere. Are complex control signals required for human arm movement? *J. Neurophysiol*, 79:1409–1424, 1998.

[21] Borenstein J. and Koren Y. Real time obstacle avoidance for fast mobile robots in cluttered environments. *IEEE Journal of Robotics and Automation*, 7:535–539, 1990.

[22] Haugeland J. *Artificial Intelligence: The Very Idea*. Cambridge, MA: MIT Press, 1985.

[23] Hale J.G. *Biomimetic motion synthesis for synthetic humanoids*. Ph.d. thesis, University of Glasgow, 1999.

[24] Lashley K.S. The problem of serial order in behavior. *Cerebral mechanisms in behavior*, 1:112–136, 1951.

[25] Schomaker L. Simulation and recognition of handwriting movements. Technical report, Nijmegen institute for cognition and information, March 1991.

[26] Miller L.E. and J.C. Houk. Motor coordinates in primate red nucleus: Preferential relation to muscle activation versus kinematic variables. *Journal of Physiology (London)*, 488:533–548, 1995.

[27] Forster L.M., McMasters Rd., and Forster M.R. How do jumping spiders catch up on their prey ? : A model for pursuit behaviour. (araneae; salticidae). *Philosophy Department, University of Wisconsin, Madison, USA.*, 1999.

[28] Kawato M. Internal models for motor control and trajectory planning. *Curr Opin Neurobiol.*, 9(6):718–727, 1999.

[29] Robinson M.H. and Valerio C.E. Attacks on large or heavily defended prey by tropical salticid spiders. *Psyche, Cambridge*, 84 (1):1–10, 1977.

[30] Hogan N. The mechanics of multi joint posture and movement control. *Biological Cybernetics*, 52:315–331, 1985.

[31] Bernstein N.A. *The coordination and regulation of movements*. Oxford, New York: Pergamon Press, 1967.

[32] Khatib O. Real-time obstacle avoidance for manipulators and mobile robots. *IEEE International Conference on Robotics and Automation*, 5:90–98, 1985.

[33] DiZio P. Issues in human movement control raised by studies in unusual force environments. Scientific retreat summary, Center for Complex Systems Scientific Retreat, Marine Biological Laboratory Woods Hole, Massachusetts, 1996.

[34] Shadmehr R. The equilibrium point hypothesis for control of posture, movement and manipulation. *In: Handbook of Brain Theory and Neural Networks, M. A. Arbib (ed), MIT Press*, 1:370–372, 1998.

[35] Shadmehr R. and Fernando A. Mussa-Ivaldi. Computational elements of the adaptive controller of the human arm. In Jack D. Cowan, Gerald Tesauro, and Joshua Alspector, editors, *Advances in Neural Information Processing Systems*, volume 6, pages 1077–1084. Morgan Kaufmann Publishers, Inc., 1994.

[36] Shadmehr R. and Wise S. A mathematical muscle model. Core course on Neuroscience, JHU School of Medicine, Supplementary materials for muscle models, 2003.

[37] Brooks R.A. A robust layered control system for a mobile robot. *IEEE Journal Of Robotics And Automation*, RA-2:14–23, 1986.

[38] Arkin R.C. *Behavior-Based Robotics*. MIT Press, 1998.

[39] McCulloch W. S. and Pitts W. H. A logical calculus of the ideas immanent in nervous activity. *Bulletin of Mathematical Biophysics*, 5:115–133, 1943.

[40] Russel S. and Norvig P. *Artificial intelligence, a modern approach*. Prentice Hall, International editions, 1995.

[41] Flash T. and Hogan N. The coordination of arm movements: an experimentally confirmed mathematical model. *J Neuroscience*, 5(7):1688–703, 1985.

[42] Braitenberg V. *Vehicles, experiments in synthetic psychology*. MIT Press, 1986.

[43] Holst E. von and Mittelstaedt H. Das reafferenzprinzip: Wechselwirkungen zwischen zentralnervensystem und peripherie. *Naturwissenschaften*, 37:464–476, 1950.

[44] Flanagan J.R. Wolpert D.M. Motor prediction. *Current Biology*, 11-18:R729 32, 2001.