User Manual
MUscle **SI**mulation **CO**de

Dr. Oliver Kayser-Herold

06/17/2008

Contents

1 Introduction

The program is based on a modular object oriented code, which implements a flexible framework for programming simulation codes which deal with biological systems, in particular muscle cells.

Currently, most of the code is focused on Monte Carlo simulations of muscle cells. For a detailed introduction of the molecular structure of muscle cells refer to [3]. Basically the muscle cells consist of thick (myosin) and thin (actin) fibers which are arranged in a comb like structure. Between these fibers, crossbridges are dynamically created by molecules which are attached to the myosin fibers and bind to the actin.

The Monte Carlo simulation uses a 1D finite element code (cf. [1]) to model the fibers. To resemble the chemical reactions which happen in the crossbridges, the Monte Carlo simulation is used. Thus the numerical procedure can be seen as a staggering scheme. After computing the mechanics, the crossbridges may change their state which then influence the mechanics again.

Besides the Monte Carlo simulation, three other numerical procedures for simulating the mechanical response of muscle fibers are implemented.

2 Installation/Build

Currently the software can be installed in a Windows environment, using the Cygwin environment, under Linux and on Apple's OS-X.

2.1 Installation Windows XP/Cygwin

The code for simulating the muscle fibers itself should be rather unproblematic with respect to portability. Unfortunately, the deal.II library uses many advanced language features which are not supported by the usual development environments that are available under Windows. Therefore, the Cygwin library has to be used to provide Unix like API calls and the GNU compiler collection.

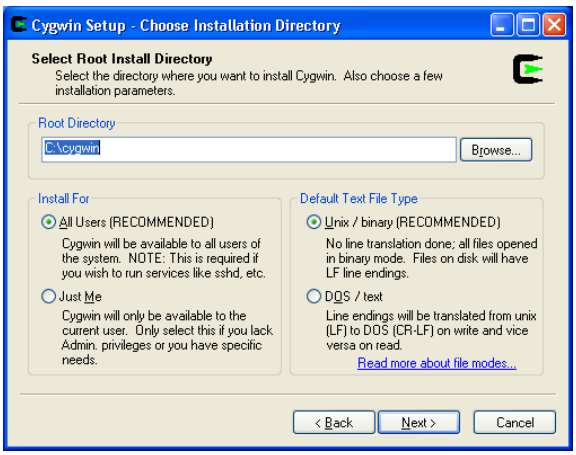

Figure 1: Initial dialog of cygwin setup program

2.1.1 Installing Cygwin

If the cygwin environment isn't already installed on the target machine, the first step consists of setting up the cygwin environment. For this purpose the file `setup.exe` from the website `www.cygwin.com` has to be downloaded and started. It automatically connects to the internet (if such a connection is available) and download other packages that are required for the installation.

Fig. 1 shows the dialog, that appears after starting the setup program. It enables the user to change certain parameters, like the default installation directory, etc. The default parameters are usually perfectly fine and shouldn't be modified.

The next dialog (Fig. 2) allows to specify a directory, where the packages that will be downloaded are to be stored. Usually, the default suggestion will work.

Since the setup program tries to automatically download the packages from the internet, a working internet connection has to be specified (the corresponding dialog is shown in Fig. 3). If anything other than the default setting (direct connection) needs to be used, consult the documentation of the cygwin setup program.

The next step is to specify a download site from which the packages should be downloaded (cf. Fig. 4). Any choice should be fine here, but might lead to differing installation speeds.

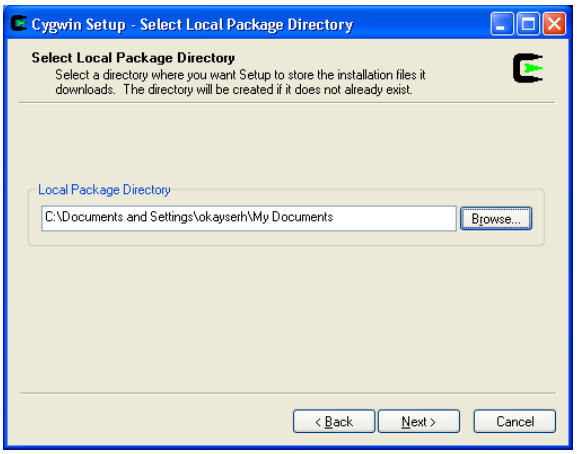

Figure 2: Initial windows of cygwin setup program

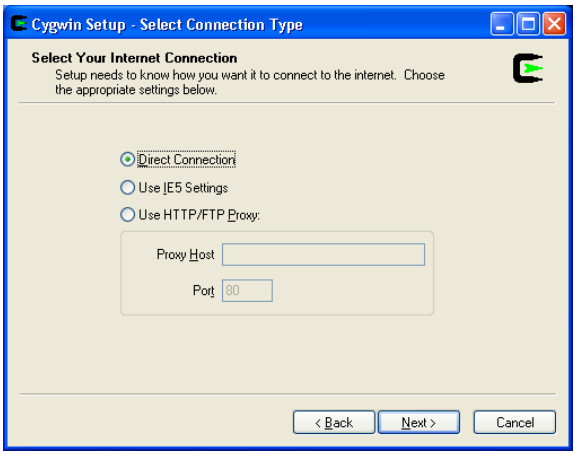

Figure 3: Initial windows of cygwin setup program

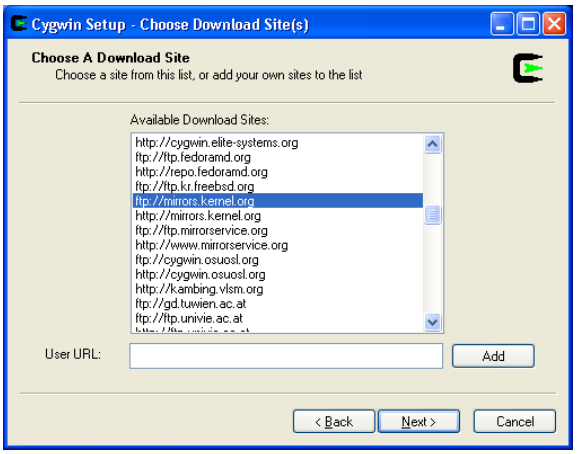

Figure 4: Initial windows of cygwin setup program

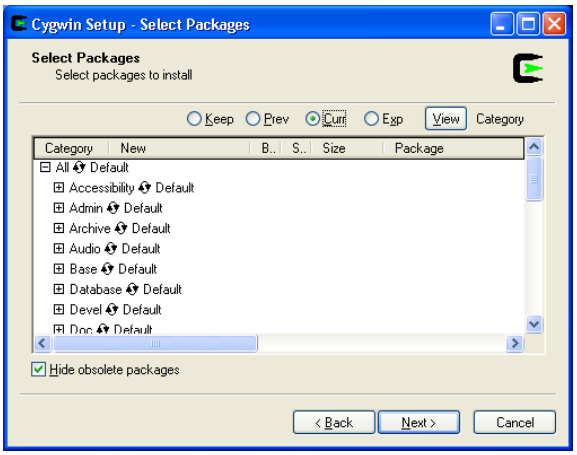

Figure 5: Initial windows of cygwin setup program

Fig. 5 shows the dialog that finally allows the user to select the packages, that should be installed. If a new installation is started, several packages will automatically be selected. These are crucial for the operation of the cygwin environment and should not be modified. For the compilation of the deal.II framework and the stochastic muscle code additional software developement packages have to be selected. It might be easier to select these, when the view is first switched to a list of all available packages. The view mode can be changed by clicking on the `view` button.

The following packages are needed for a successful compilation and later use of the program:

- emacs
- gcc-core
- gcc-g++
- gcc-g77
- gnuplot
- perl
- openssh
- cvs
- make
- lapack

If additional packages of the cygwin system should be installed or updated, it usually suffices to start the setup program again and select or deselect the required software (cf. also the cygwin website for further information).

2.1.2 Compiling the deal.II FEM framwork

Before the deal.II framework can be installed, the source code has to be downloaded from the homepage of the deal.II project: `www.dealii.org`. On the website a link to the download page can be found. The recommended

version is 6.1.0 without documentation and examples. It can be saved at any place in the filesystem.

Once, the cygwin environment has been successfully installed, the cygwin environment will give the following output when started for the first time:

```
Copying skeleton files.
These files are for the user to personalise
their cygwin experience.

These will never be overwritten.

`./.bashrc' -> `/home/okayserh//.bashrc'
`./.bash_profile' -> `/home/okayserh//.bash_profile'
`./.inputrc' -> `/home/okayserh//.inputrc'
```

Subsequent starts won't give this output, but instead show a user prompt, like the following:

```
okayserh@HP20439203671 ~
$ pwd
/home/okayserh
```

Usually, the command line starts in the home directory, which is `/home/okayserh` in the example shown above.

Now it is time to unpack the compressed source file in the home directory:

```
okayserh@HP20439203671 ~
$ tar --gzip --extract --file /cygdrive/c/Docum
ents\ and\ Settings/okayserh/Desktop/deal.nodoc-6.1.0.tar.gz
```

The part following the command `--file` has to be changed to the place, where the file was stored during the download. Cygwin mounts the windows drives in a subdirectory `/cygdrive/` plus the letter describing the hard disk partition. In the example, the file was saved on the users desktop.

After unpacking the archive, it is always good the verify that the everything is in place:

```
okayserh@HP20439203671 ~
$ ls
deal.II

okayserh@HP20439203671 ~
$ cd deal.II/
```

The directory `deal.II` should contain the following files and directories:

```
okayserh@HP20439203671 ~/deal.II
$ ls
Makefile          Version.in  common        contrib  lac
README            aclocal.m4  configure     deal.II  lib
README.configure  base        configure.in  doc      reconfigure
```

If everything is in place, the next step is the configuration of the library, which should usually run fully automatically:

```
okayserh@HP20439203671 ~/deal.II
$ ./configure --with-umfpack

[....]
```

Appending `--with-umfpack` is very important, since the direct solver *UMFPACK* is used as primary solver for linear systems of equations in most parts of the program. In case of error messages, make sure that the cygwin packages given in section 2.1.1 were selected. If this doesn't help, a look to the deal.II mailing list, which can be found on the same website as the source code, might give some hints.

After a successful configuration, the output should end with the following paragraph:

```
Please add the line
   export PATH=$PATH:/home/okayserh/deal.II/lib
to your .bash_profile file so that windows will be
able to find the deal.II shared libraries when
executing your programs.
```

Finally, the actual compilation has to be started by:

```
okayserh@HP20439203671 ~/deal.II
$ make -j 4 1d
[....]
```

Warnings that appear in the output, can usually be ignored. On a halfway modern computer, the compilation will take between 30 and 60 minutes to finish.

A final step is to add the line given in the configuration output to the local configuration. An arbitrary editor may be used to edit the file `/home/okayserh/.bashrc` (the *okayserh* has to be replaced by the name found in the actual home directory. If a windows editor is to be used, the cygwin directories are usually located under `c:\cygwin\home\okayserh` for the example given above.

At the end of the file, the following line (or whatever is returned by the configuration script) has to be appended:

```
export PATH=$PATH:/home/okayserh/deal.II/lib
```

This should complete the installation of the deal.II framework.

2.1.3 Compiling the stochastic code

2.2 Installation Linux

2.3 Macintosh OS-X

3 Preliminaries

As the primary purpose of the program is the comparison of results for different experimental protocols, usually the first step is the definition of a protocol that defines the time course of the mechanical loads (input file `protocol.dat`). After that the general simulation parameters and the output have to be specified (input file `simulation.prm`). Finally specific files which provide the parameters for the selected models and muscle lattice have to be created (the filenames depend on the selected model and lattice geometry).

Except for the file `protocol.dat`, all parameter files follow the conventions which come from the parmeter handler concept of the *deal.II* library, which provides the basis for the program. The *deal.II* parameter files are subdivided into sections, which can contain one or more parameters. Usually the sections divide the available parameters into logically related sets of parameters. The following paragraph shows an example of the input file `simulation.prm`:

```
# Listing of Parameters
# ---------------------
subsection global
  set K_xb        = 1.0
  set dt          = 1e-5
  set model       = Huxley
  set numerics    = MonteCarlo
  set random_seed = 0
end
subsection output
  set distribution = 99
  set energy_distribution = 99
  set force_distribution = 99
  set filament     = 99
  set macro        = 1e-10
  set transition   = 99
end
```

Each of the subsections starts with the keyword `subsection` followed by the name of the subsection. The available section names will be discussed in detail in the other parts of this manual. After that one or more lines defining the parameters in that section follow. For each parameter the line starts with the keyword `set` and the subsequent name of the parameter. Finally the value which should be assigned to that particular parameter comes behind an equal sign which separates the parameter value from its name. Generally floating point numbers, integer numbers and textual values are admissible as values for a parameter. Which one of these three basic types is admissible depends on the parameter and will be discussed in the section about the parameter file.

The program looks for the parameter files in the directory from which is was started. After each time step, the program writes the current time step

Column	Description	Remarks
1	Time	Floating point
2	Boundary type	Integer, cf. Tab. 2
3	Function type	Integer, cf. Tab. 3
4	Function parameter a	Floating point
5	Function parameter b	Floating point, optional
6	Function parameter c	Floating point, optional
7	Function parameter d	Floating point, optional

Table 1: Columns in the protocol file

Type	Description
0	Prescribed displacments
1	Prescribed force

Table 2: Available boundary conditions

number to the console. Error messages will be written to the console as well. In case of a severe error, the program will usually abort the computation. An error, which can frequently be observed, is:

```
Internal Error!!!! Detach, Sum 1.662999e+00
```

This error indicates, that time step size was to large for a rate. If this error occurs only from time to time, the results should usually not be significantly affected. Otherwise the time step size should be reduced.

3.1 The Protocol File

As already mentioned the protocol file controls the type and time course of mechanical loads which will be prescribed during the simulation run. It contains one or more lines, which are composed of at least 4 entries.

The first parameter defines the time at which the simulation should switch from one mode to another mode. Hence the first line should start with $t = 0$. After that the type of boundary condition is defined. Currently four boundary conditions are implemented:

While this field defines the type of boundary, the value for this boundary condition can currently be defined by three different types of functions: The

Type	Description
0	constant function, $f = a$
1	linear function, $f = f = a * t + b$
2	sine function, $f = a * \sin(b * 2\pi * t + d * \pi) + c$
3	twitch, $f = c * (\exp(-t/a) - \exp(-t/b))/(tp + d)$

Table 3: Available functions

twitch is a special function, which was introduced in the context of the activation models. It simulates a short stimulation of the muscle, where the parameters control the time course and amplitude of the stimulation. This function is of limited use for mechanical stimulus, although it certainly can be used.

The parameter t in the above mentioned functions is not the absolute time. It is the time from the beginning of that protocol step!. Thus when a new step in the protocol is reached, t will start again from zero.

One example, which defines isometric conditions followed by isotonic conditions with a prescribes force per filament of $430pN$ looks like:

```
0.0 0 0 0
0.5 1 0 430
1.0 1 0 430
```

At $t = 0.5$, the boundary mode is switched from prescibed displacements to prescribed forces. The last line defines the end of the simulation.

Another example for a sinosoidal prescribed displacement at one end is:

```
0.0 0 0 0
0.1 0 2 50 10 0.0
0.5 0 0 0
```

After a short initial phase, the sinosoidal displacement is prescribed at one end of the filament. Again the last line defines the end of the simulation run.

An abritrary number of lines is allowed in this protocol file. But currently no comments may be added!.

If the used numerical scheme is not the Monte Carlo code, the force that is specified in the protocol file is a normalised force, where a value of 1 corresponds to the maximum force under isometric conditions.

3.2 Protocol File Examples

The following paragraphs display some of the protocols that can be implemented by using the protocol file.

Gradual Step

Instead of having an instant shortening, the shortening is spread over a short time interval during which the displacement follows a linear function.

```
0.0 0 0 0 0
0.2 0 1 -40000 0
0.2001 0 0 -4 0
0.3 0 0 0 0 0
```

The crucial part is in the second and third line of the protocol file, where the function type is switched to a linear function with negative slope of -40000. After $1/10000s$, the function is switched back to a constant function, which realises the final value of the displacement. The resulting displacement curve is shown in Fig. 6.

Sine/Cosine Function

To have a continuous first time derivative of the displacement at the point where the protocol switches from constant displacement to sinosoidal displacement, we change the sine function to a cosine by adding an offset of $1/2\pi$. At time $t = 0.6s$, the frequency is reduced to half of what it was before.

```
0.0 0 0 0 0
0.2 0 2 4 10 -4 0.5
0.6 0 2 4 5 -4 0.5
1.0 0 0 0 0 0
```

Results of this protocol are shown in Fig. 7.

Multiple Steps

Having multiple steps is quite simple. It just requires changing the constant displacement at certain timesteps as shown in the following protocol file:

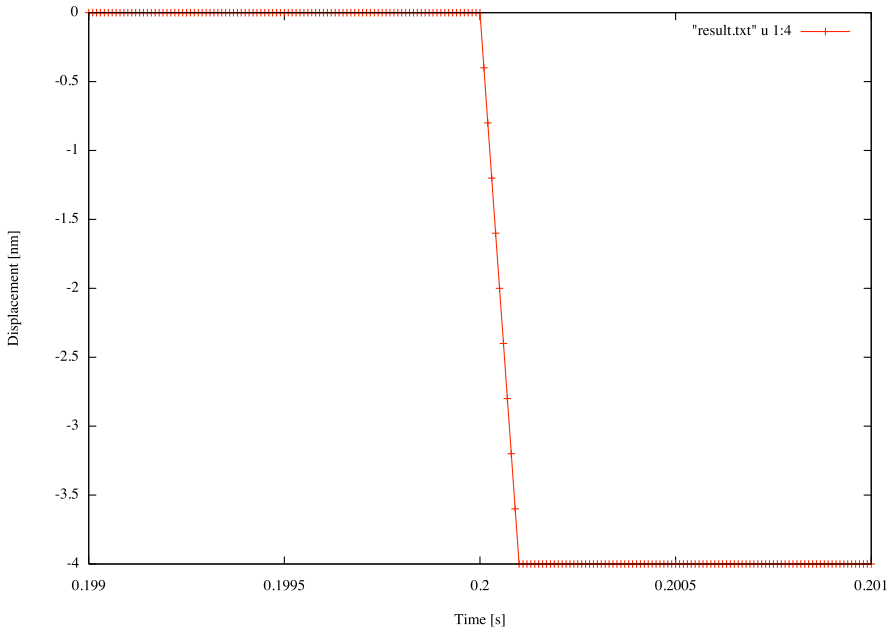

Figure 6: Gradual step

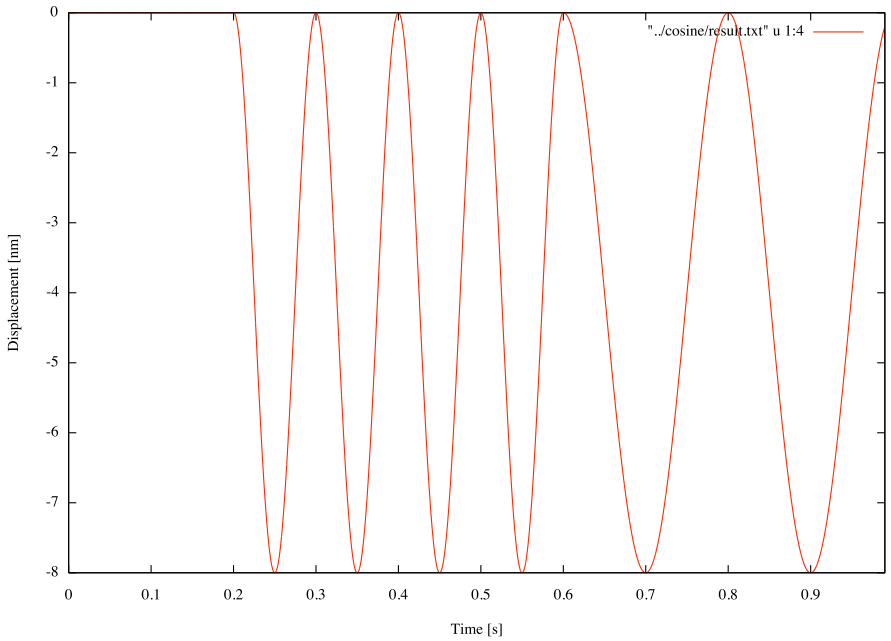

Figure 7: Changing frequency

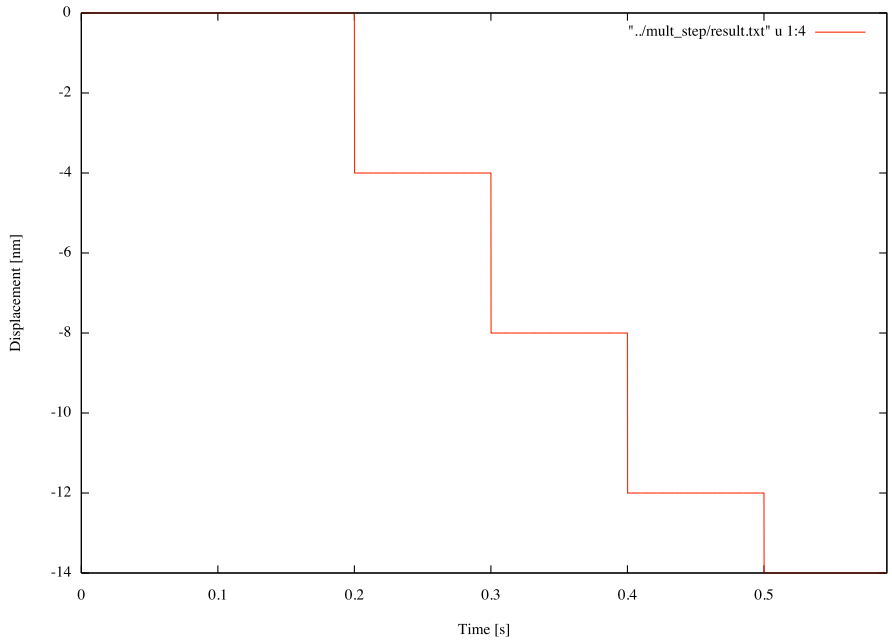

Figure 8: Multiple Step

```
0.0 0 0   0 0
0.2 0 0  -4 0
0.3 0 0  -8 0
0.4 0 0 -12 0
0.5 0 0 -14 0
0.6 0 0   0 0
```

It is certainly possible to combine the multiple steps with the technique for gradual steps as shown in the previous paragraph. Results of the protocol are displayed in Fig. 8.

Multiple Sine Functions Another option is having a sine function with differing amplitude.

```
0.0 0 0 0 0
0.2 0 2 4 10 -4 0.5
0.5 0 2 8 10 -8 0.5
1.0 0 0 0 0 0
```

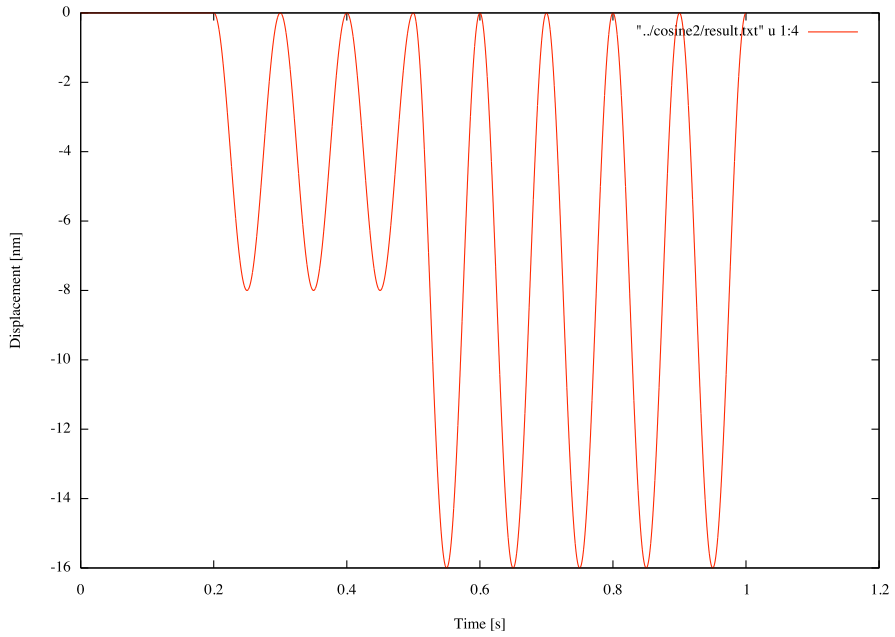

Figure 9: Sine function with diffing amplitude

Again this is just the combination of two different sine functions. Fig. 9 shows the resulting displacement.

Switch from Isometric to Isotonic Finally, another example for switching from prescribed boundary displacements (isometric conditions) to prescribed forces (isotonic conditions).

```
0.0 0 0 0 0
0.5 1 0 500 0
1.0 0 0 0 0 0
```

The results are shown in Fig. 10. It can be clearly seen that initially the displacements are fixed, while after 0.5 seconds, the displacements are a result of the prescribed force.

3.3 Global Simulation Parameters

The next file, which has to be present in all simulation runs is `simulation.prm`. Besides the model type and the lattice geometry type, some numerical pa-

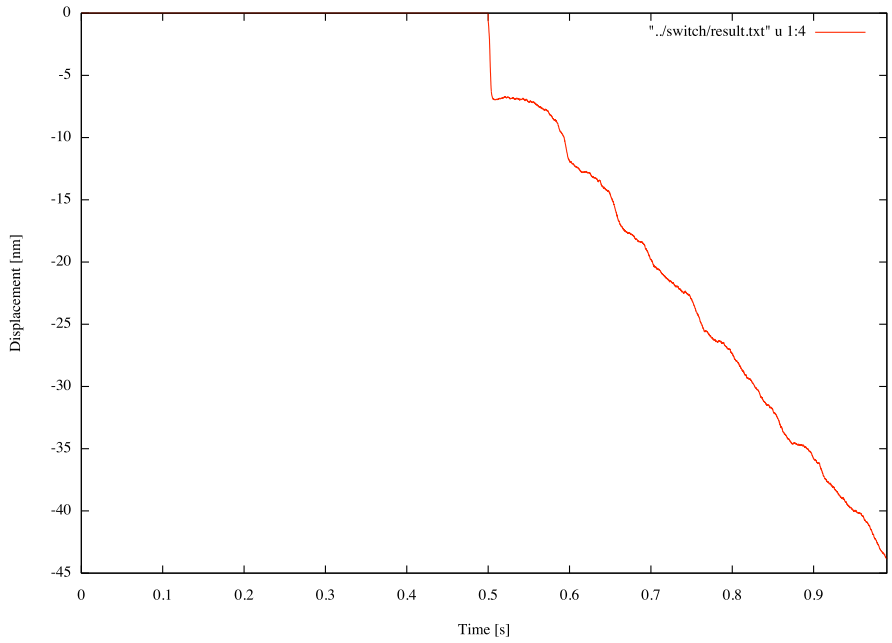

Figure 10: Switch from prescribed displacements (Isometric) to prescribed forces (Isotonic)

rameters and the structure of the output files are defined in this parameter file. An example which contains all currently implemented parameters is shown below:

```
# Listing of Parameters
# ---------------------
subsection global
  set K_xb        = 1.0
  set dt          = 1e-5
  set model       = Huxley
  set numerics    = MonteCarlo
  set random_seed = 0
end

subsection output
  set distribution = 99
  set energy_distribution = 99
  set force_distribution = 99
  set filament     = 99
  set macro        = 1e-10
  set transition   = 99
end
```

The parameters which belong to the subsection `global` are explained in Tab. 4. Those in subsection `output` can be found in Tab. 5.

3.4 Numerics

Monte Carlo Simulation Code

The Monte Carlo (MC) simulation is a computationally expensive, but quite accurate way of simulating the biomechanical responses of muscle fibers. Many of the components of the MC simulation can be easily exchanged by modifying the corresponding parameters.

`numeric_mc.prm`

```
# Listing of Parameters
```

Parameter	Type	Description
K_xb	Floating Point	The crossbridge stiffness. It defines the stiffness of the linear element which is used to represent the elastic crossbridge, which connects the thin with the thick filament
dt	Floating Point	Time step size
model	One of `Huxley`, `Duke`, `Daniel`, `Pate`, `David9`	This parameters defines the model for the state transitions. More detailed descriptions can be found in the sections about specific models.
numerics	One of `MonteCarlo`, `Char`, `Integral`, `FEM`	So far four numerical procedures for simulating muscle fibers are implemented. One of these can be selected by this parameter.
random_seed	Integer	Although the algorithm can be seen as a Monte Carlo simulation the results are supposed to be completely deterministic. This is due to the pseudorandom generator which generates always the same sequence of random numbers when it is started with the same initial seed. In some rare cases it might be useful to start the random generator with a different seed, which can be specified in this field. Generally it seems to be wise to start all runs with the default random seed (0) to get comparable results.

Table 4: General simulation parameters, which define the used actomyosin cycle, the lattice model and other components.

Parameter	Type	Description
distribution	Floating Point	Time interval for writing the crossbridge distribution
force_distribution	Floating Point	Time interval for writing the crossbridge force distribution. The force results from the linear relationship between crossbridge strain and force.
energy_distribution	Floating Point	Time interval for writing the crossbridge energy distribution. Energy is the one half of the squared strain multiplied by the crossbridge stiffness.
filament	Floating Point	Time interval for writing the individual forces of the sarcomers
macro	Floating Point	Time interval for writing the summation parameters
transition	Floating Point	Time interval for writing the transition matrix

Table 5: Parameters in the simulation control, which control the frequency for writing certain information to the output files.

```
# --------------------
subsection monte_carlo
  set activation = None
  set filaments  = 100
  set geometry   = Vertebrae
  set full_sarcomer = 0
end
subsection energy_dist
  set start_x = 0.0
  set end_x = 200.0
  set dx = 1.0
end
subsection force_dist
  set start_x = -10.0
  set end_x = 50.0
  set dx = 0.5
end
```

Besides the primary parameter section, there are two additional parameter sections *energy_dist* and *force_dist*, which offer the parameters shown in table 6. These parameters can be used to define the bins that are used to create outputs of the energy and force distribution of the attached myosin heads. The discrete structure of the MC code makes it impossible to directly output distributions. Therefore, the discrete heads have to be sorted into bins according to a certain scheme. Depending on the other parameters of the simulation and the desired result, different sizes of bins may be more appropriate in one or the other situation.

Method of Characteristics Simulation Code

Another well established numerical scheme for solving the hyperbolic equations that were developed to describe the biochemical and mechanical responses of muscle fibers is the method of characteristic. This numerical scheme does not show the problems that appear in schemes like the finite difference or the finite element method when large convective terms are present. But this advantage comes with disadvantages, too. The convective terms move the discretisation points in the domain. Therefore, the points will wander out of the domain of interest after some time.

By discarding solution points at the end of the domain and shifting the

Parameter	Type	Description
activation	One of `None`, `Fixed`, `Geeves6S`, `Lumped2S`, `FChain`	Selects, whether an activation model should be used and if, what type of activation model.
filaments	Integer (> 0)	Defines the number of filaments in parallel.
geometry	One of `Simple`, `Vertebrae`, `Flight`, `David`	Defines what geometry model should be used for the muscle. The details can be found in the section about the specific geometries.
full_sarcomer	Integer ≥ 0	This parameter has two functions. A value greater zero switches the code for full sarcomers on, while the default value of zero leads to the standard simulation of a half sarcomere. When the full sarcomere mode is used, this parameter defines the number of full sarcomeres in series. I.e. a number of one simulates one full sarcomer, while a number of two simulates two full sarcomeres in series. It should be noted that the computational effort approximately increases with $filaments * full\_sarcomer$.

Parameter	Type	Description
start_x	Floating Point	Start point for distribution bins.
end_x	Floating Point	End point for distribution bins.
dx	Floating Point	Size of distribution bin.

Table 6: Parameters for defining the bins that are used to evaluate head distributions.

Parameter	Type	Description
dx	Floating Point (> 0)	Spacing of discretisation points.
x_start	Floating Point	Initial start point of the computational domain.
x_end	Floating Point	Inital end point of the computational domain.

points back into the domain of interest, the implemented algorithm avoids this problem. Nonetheless, in some cases this procedure may still lead to problems in cases, where the detachment happen slowly over a huge domain. A careful analysis of the results is always recommended, when a new model is tested with this algorithm.

Another problem is the introduction of numerical oscillations, when discontinuous rate functions are used. Most recently proposed schemes of the actomyosin cycle use rather smooth rate functions. Thus, this problem is best seen with Huxleys two state model. Since the results of interest are usually some integrals over the state distribution, the practical implications of these oscillations are usually rather benign.

`numeric_char.prm`

```
# Listing of Parameters
# ---------------------
subsection characteristics
  set dx      = 0.1
  set x_end   = 50.0
  set x_start = -50.0
end
```

The initial start and end points are moved in each time step, when convective terms are present. Thus, the coordinates of these points change over the simulation. As already mentioned in the previous section, the algorithm shifts the points back into the original domain.

Integral Method

Careful analysis of the results obtained with the method of characteristics and those from the MC simulation revealed a subtle difference in the resulting

Parameter	Type	Description
dx	Floating Point (> 0)	Spacing of discretisation points.
x_start	Floating Point	Initial start point of the computational domain.
x_end	Floating Point	Inital end point of the computational domain.
r	Floating Point	The new approach needs an additional parameter that describes the percentage of detached heads under isometric conditions. A value of 0.2 corresponds to 20% percent detached heads. This parameter can be used to adjust the initial slope.

head populations. This difference is not a numerical artifact, but turned out to be the result of a slight difference in the model assumptions. The integral method tries to imitate the behaviour observed in the MC code with a continuous approach to replicate the MC results with less computational effort.

A detailed report about the underlying ideas is in preparation. The basis of this algorithm is still similar to the method of characteristics.

`numeric_char_int.prm`

```
# Listing of Parameters
# ---------------------
subsection characteristics_integral
  set dx      = 0.1
  set x_end   = 20.0
  set x_start = -20.0
  set r = 0.1876
end
```

Finite Element Method

The finite element method (FEM) is an other possible numerical scheme to solve the hyperbolic equations that describe the biochemical processes in

Parameter	Type	Description
global_refine	Integer	Number of global refinement steps. Initially the finite element mesh is initialised with a single element that goes from xstart to xend. In each refinement step, every element is split into two equal halfs. Thus, one refinement step leads to two elements, two lead to four and so on.
p_order	Integer	Polynomial order of the ansatzfunctions used in the element. One denotes linear shape functions, two quadratic, etc.
xend	Floating Point	Inital end point of the computational domain.
xstart	Floating Point	End of the FE domain. Has to be larger than xstart!

muscle fibers. It is implemented in the muscle simulation code, but should be considered largely experimental. Furthermore, a proper stabilization approach for the convective terms is currently missing in the implementation.

`numeric_fem.prm`

```
# Listing of Parameters
# ---------------------
subsection fem
  set global_refine = 6
  set p_order       = 3
  set xend          = 44
  set xstart        = -44
end
```

3.5 Output Files

During a program several output files will be created. Some of them serve mostly the purpose of verfication, while others contain the valueable results

of the simulation runs. All files contain the data in ASCII format. Although these files consume more space then binary formats, they have several advantages:

- Easier to process with scripting languages like *Python*
- No problems with big/little endianess
- Can be edited (if necessary)
- Enables preliminary views of data while simulation is still running

In the following paragraphs, the structure of the different output files will be shortly explained.

3.5.1 `rates.dat`

When the program is started, the rate functions, which are the basis for the Monte Carlo state transitions are computed for a certain number of discrete strains. in the range from $x = -15$ to $x = 15$ with a spacing of $\Delta x = 0.1$. Each line of the file contains all rate functions for a single strain x, which is always in the first column. The number of columns in a line depends on the model and is $n * (n-1) + 1$ with n being the number of states. Hence for a 3 state model a typical line looks like:

$x \quad k_{12} \quad k_{13} \quad k_{21} \quad k_{23} \quad k_{31} \quad k_{32}$

3.5.2 `states.dat`

Another interesting information about the used model is the stationary distribution of the crossbridge states. When the rate functions are used in a continuous model, this model will reach a stationary state at $t = \infty$. If there's no convection, the problem reduces to finding the solution of the linear system

$$\hat{\mathbf{K}}(x)\boldsymbol{T} = \boldsymbol{R}(x). \tag{1}$$

for each strain x. For a three state model the matrices are

$$\hat{\mathbf{K}} = \begin{pmatrix} (-k_{12} - k_{21} - k_{23}) & (k_{32} - k_{12}) \\ (k_{23} - k_{13}) & (-k_{13} - k_{31} - k_{32}) \end{pmatrix} \tag{2}$$

with right hand side vector:

$$\boldsymbol{R} = \begin{pmatrix} k_{12} \\ k_{13} \end{pmatrix} \tag{3}$$

. The stationary distribution of the third state can easily be found by using

$$\sum_{i=1..3} T_i = 1. \tag{4}$$

A line in the file has then the following content:

$x \quad T_1 \quad \ldots \quad T_n$

3.5.3 `version.dat`

To keep track of the version and to have some information about the version of the program which was used to create specific results, the version number of the program is written to this output file. Unfortunately, there is no CVS-tag to automatically insert the date of the last *commit* command into the source code. Therefore this version number may currently not always be accurate. It looks like:

```
$Id: manual.tex,v 1.10 2008/08/14 19:31:50 okayserh Exp $
```

3.5.4 `result.txt`

In the result file the macroscopic mechanical properties of the complete system are summarised. Thus it is probably the most important piece of information which is generated by a simulation run. The length of the lines varies again with the used model as it also contains some statistics about the crossbridges. Tab. 7 explains the different columns in the output file.

Note: If either the multiple sarcomere mode or an activation model is used for a simulation run, additional columns will appear in the result file.

Note: The standard deviation is computed bya fast algorithm, which might be plagued by round off errors! For a precise standard deviation, create the full distribution output to write the raw data and use a postprocessing script to accurately determine the standard deviation.

Parameter	Description
Time	Simulation time
Force	The force per filament which is generated by the model. Due to the Monte Carlo simulation, its smoothness depends strongly on the number of simulated sarcomers
Displacement, first myosin node	The displacements on this node display either the prescribed displacements or the displacements without the effects of the extensible filaments
Displacement, last actin node	Basically the same as the previous value. This time the extensibility of the filaments can be seen.
Std. Dev. Force	Standard deviation of the force per filament.
Std. Dev. Disp., first myosin node	Same as above for the displacement of the first myosin node
Std. Dev. Disp., last actin node	Same as above for the displacement of the last actin node
Number of attached crossbridges per filament	This number must be seen in relation to the total number of available crossbridges.
Number of crossbridges in state 1	If state 1 is a detached state, this value will always be 0
...	...
Number of crossbridges in state n	If state n is a detached state, this value will always be 0
Displacement, M-Line	This value is only of interest if the full sarcomer is simulated. It displays the displacement of the M-Line, which connects the left half sarcomer and the right half sarcomer.

Table 7: Columns in the result file

3.5.5 `distrib.dat`

For some purposes it is quite useful to have the time course of the state distribution of the crossbridges. That is the number of crossbridges in a certain state with a certain strain. This information is quite bulky and should therefore not be written in every single timestep even when this is supported by the program.

Two different representations of the crossbridge distribution are mixed together in this file. Due to the discrete nature of the model only the strain and state of the single crossbridges are known. If a "continuous" distribution is needed for comparing the results with results of the continuous model, an additional processing step is required. In this step, the relevant domain of the crossbridge strains is discretised into small segments. Then the attached crossbridges are sorted into one of the small segments and counted for that segment. When all crossbridges have been processed, this gives a discrete approximation of the continuous crossbridge distribution field.

In this file the data is organised in datasets, which are written whenever the specified output time interval (parameter `distribution` in parameter file `simulation.prm`) is completed. The dataset starts with a line where the simulation time of the following dataset is found. Then several lines with the discrete approximation of the distribution field follow. These lines consist of the strain x in the first column and the crossbridge distribution for the states 1 to n in the second to $n+1$th column. A typical section in that part looks like:

```
[...]
-4.5 0 0 0 1 1 4 70 150 0
-4 0 0 0 0 4 6 127 194 4
-3.5 0 0 0 0 14 25 108 278 10
-3 0 0 0 0 48 70 133 277 10
-2.5 0 0 0 3 117 207 99 223 13
-2 0 0 0 6 195 378 66 109 22
[...]
```

for a 9 state model.

After that the accumulated raw data can be found in the dataset. Here the single myosin heads on the thick filament are evaluated. For each head a statistic over the available sarcomers is created.

```
0 0.01 0.02 0.02 0.21 0.35 0.38 0.01 0 0
1 0.12 0.06 0.35 0.01 0.04 0.05 0.13 0.21 0.03
2 0.02 0.04 0.14 0.5 0.02 0.02 0.04 0.13 0.09
3 0.17 0.01 0.21 0.32 0.01 0.01 0.08 0.15 0.04
4 0.15 0.01 0.53 0.13 0.03 0.05 0.03 0.06 0.01
5 0.06 0.04 0.37 0.09 0.15 0.23 0.02 0.02 0.02
6 0.07 0.02 0.53 0.22 0.02 0.09 0.02 0.01 0.02
```

This is again data which was generated by a 9 state model. The first number in the line identifies the myosin head. As every single sarcomer is constructed in the same way, this number uniquely identifies a myosin head. The following numbers denote the fraction of these heads in a certain state. Certainly the first number corresponds to the first state and so on. Thus the number of columns is again $n + 1$.

3.5.6 `displacement.dat`

The displacement data file reveals the displacement field of the finite element models, which are used to simulate the mechanics of the filaments. Similar to the distribution output file, it contains several data sets. Each data set starts with the simulation time in a single line. After that the displacement fields of the different filaments follow in separate blocks. Hence the number of these blocks depends on the used lattice geometry, which determines how many filaments are present.

Each displacement field is a set of lines, where the first number in each line gives the original coordinate of the finite element node and the second number displays the displacement of that node. At each time, where an output is requested the block of data starts with a single floating point number, which is the simulation time. In the next line, the displacement field of the first fiber follows. Each of the followings displacement fields for the other fibers is separated by an empty line.

3.5.7 `filaments.dat`

For some simulations it is desireable to no only have the condensed information of the full ensemble of sarcomers but also to have the mechanical data

for each single sarcomer. As this data can again be quite large, it should usually only be written in selected time intervals.

The format of this output file is quite simple. Currently each line has 3 columns which contain time, force, and displacement of a single sarcomer. The number of sarcomers which were used for the simulation run determines the number of lines when the next data set (for the next simulation time) follows.

A typical file looks like:

```
[...]
1.000000e-04 -0.000000e+00 0.000000e+00
1.000000e-04 2.527154e+01 0.000000e+00
1.000000e-04 -0.000000e+00 0.000000e+00
1.000000e-04 -0.000000e+00 0.000000e+00
1.010000e-02 1.816975e+02 0.000000e+00
1.010000e-02 1.817469e+02 0.000000e+00
1.010000e-02 1.927373e+02 0.000000e+00
1.010000e-02 1.881731e+02 0.000000e+00
[...]
```

4 Muscle Lattices

On the molecular level the muscle fibers exhibit a complicated microstructure, which has a significant influence on the mechanical properties of muscle. This muscle structure strongly influences the possibilities of the myosin head to bind to the actin fiber.

To evaluate the effect of different microstructures onto the mechanical properties, models of different complexity were integrated into the program. The following sections will give a brief overview about geometry models and the parameters which can be used to adjust these models.

4.1 Linear Lattice

The linear lattice does not resemble an actual muscle structure. It mostly serves as a simple verification tool because it comes closest to the continuous

models, which do not include any assumptions about the microstructure of the muscle.

Basically the muscle consists of a single actin and a single myosin fiber. The actin fiber has a length which is specified by the parameter `length_actin`. There is no explicit parameter for the length of the myosin fiber. It is determined by the spacing between the heads `dx_myosin` and the number of myosin heads `myosin_heads`. Thus the length is $dx_{myosin} \cdot heads_{myosin}$. The lattice structure does not influence the binding probabilities. Both fibers do not start at the same position. The offset between the two fibers is defined by `myosin_offset`. The mechanical properties which are used in the finite element model of the fibers are defined by `a_actin`, `e_actin`, `a_myosin` and `e_myosin` for the actin fiber and the myosin fiber respectively. The parameter starting with `a_` defines the crosssectional area of the fiber in nm^2 while the parameter starting with `e_` defines the elastic modulus in $TODO$.

The last parameter `xamax` defines the search interval for finding matching actin binding sites, which are in the range of the myosin head. If the myosin head is attached to the myosin fiber at position x, it can potentially bind to all binding sites in the interval $[x - x_{amax} \dots x + x_{amax}]$. Clearly the strain dependent rate functions have to allow this binding.

`Simple, geom_simple.prm`

```
# Listing of Parameters
# ---------------------
subsection actin
  set a_actin      = 5.04
  set dx_actin     = 2.77
  set e_actin      = 1.27e3
  set length_actin = 1050.0
end

subsection myosin
  set a_mysoin      = 5.0
  set dx_myosin     = 14.3
  set e_mysoin      = 3.25e3
  set myosin_heads  = 49
  set myosin_offset = 125.0
end
```

```
subsection numerics
  set xamax = 20.0
end
```

4.2 Vertebrae Lattice

Most of the parameters which are offered in the Vertebrae lattice model have the same meaning as in the simple lattice. Basically there are two differences. Instead of one actin fiber in the simple model, there are two actin fibers which need separate parameters for the elastic modulus and the crosssectional area. The second new parameter is related to the axial structure of the fibers. It defines the number of actin binding sites after which the actin the binding sites completed a full rotation. The default value of 13 should be correct in most cases. Other values do not lead to problems with the program but should only be used when it is clear what is done. The same holds for the elastic modulus of the two actin fibers. Although the program supports different elastic moduli for the two actin fibers, which are used in the simulation, but in most scenarios it does not seem sensible to use this freedom.

`Vertebrae`, `geom_vertebrae.prm`

```
# Listing of Parameters
# ---------------------
subsection actin
  set a_actin1     = 50.4
  set a_actin2     = 50.4
  set actin_period = 13
  set dx_actin     = 2.77
  set e_actin1     = 1.27e3
  set e_actin2     = 1.27e3
  set length_actin = 1050.0
end

subsection axial
  set d_actin_myosin = 30.0
  set radius_actin   = 3.5
```

```
  set radius_myosin  = 7.0
end

subsection myosin
  set a_mysoin      = 50.0
  set dx_myosin     = 14.3
  set e_mysoin      = 3.52e3
  set myosin_heads  = 49
  set myosin_offset = 125.0
end

subsection numerics
  set xamax = 15.0
end
```

In the section `axial` the axial configuration of the actin and myosin fibers can be defined. The parameter `d_actin_myosin` defines the distance between the center of the myosin fiber (x_m, y_m) and the center of the actin fiber (x_a, y_a) (cf. Fig. 12). As the name suggests, `radius_actin` and `radius_myosin` defines the radii of the actin and myosin fiber respectively. These radii correspond to the radii of the circles draw around (x_a, y_a) and (x_m, y_m) in Fig. 12 respectively. Hence they do not have to correspond to the real radii of in the muscle fibers but should rather be considered tools to define a proper dependence of the binding rates on the axial angles.

In the following a few comments about the algorithm to determine the angles will be given. On the actin, the binding sites are equally spaced along the fiber. But looking into the direction of the fiber, the binding sites rotate around this fiber (cf. Fig. 11). After 13 binding sites, the binding sites have again the same angular positions. In literature a value of 5.5nm can be found for the spacing along the axis.

The binding sites on the myosin fiber have a different structure. Angularly they have a structure which is similar to a 3 blade fan. I.e. they are separated by an angle of 120 deg. In the following these three binding sites, which have the same longitudinal position along the myosin fiber will be called *head*. Each triple of heads will be called a *crown*. Looking along the myosin fiber, the crowns are placed in a distance of $14.3nm$ apart. Starting with crown 1 at position $0nm$, crown 2 follows at position $14.3nm$ and is rotated

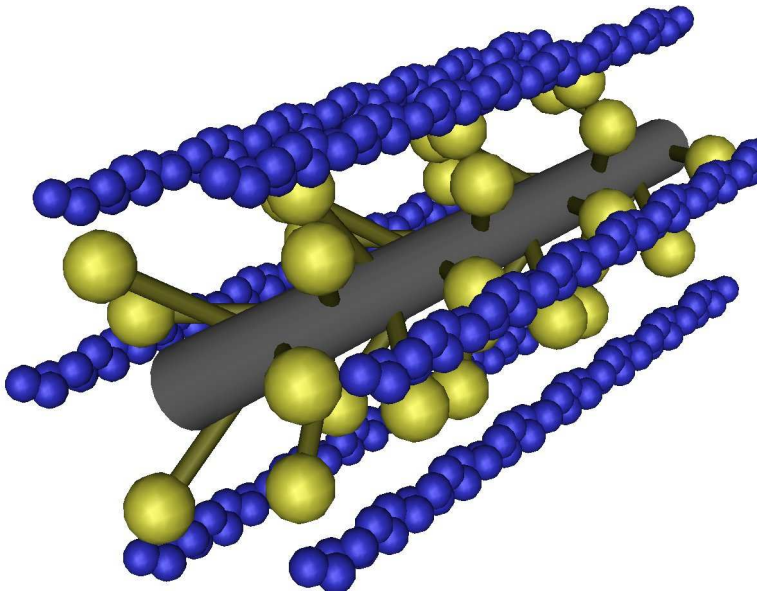

Figure 11: Sketch of the 3D structure of the muscle lattice

40° clockwise, while crown 3 follows at position $28.6nm$ and is rotated 40° counterclockwise. At $43.9nm$ again a crown 1 follows, which has the same angular orientation as the one on position $0nm$ (cf. Fig. 13).

This geometrical setup of the muscle structure leads to the situation that the spatial periodicity along the fibers differs between the myosin and actin fiber. Hence the angles under which the myosin heads and the actin binding sites will come together always varies. If the filaments are assumed to be elastic, the geometric setup also varies throughout the timecourse of the simulation. Therefore the proposed model recomputes the geometric configuration of the binding sites in each time step.

It is assumed that the probability that a head on the myosin fiber can bind to a neighboring actin site is strongly influenced by the angle between the myosin head and the actin site which is called θ. Additionally also the angle between the actin site and the center of the myosin fiber is computed and will be denoted by β. Fig. 12 shows the basic geometric setup.

Looking at a crosssection which normal is aligned with the myosin fiber, the structure shown in Fig. 13 is found. The three subfigures show the structure at the positions of the three different crowns. This hexahedral structure

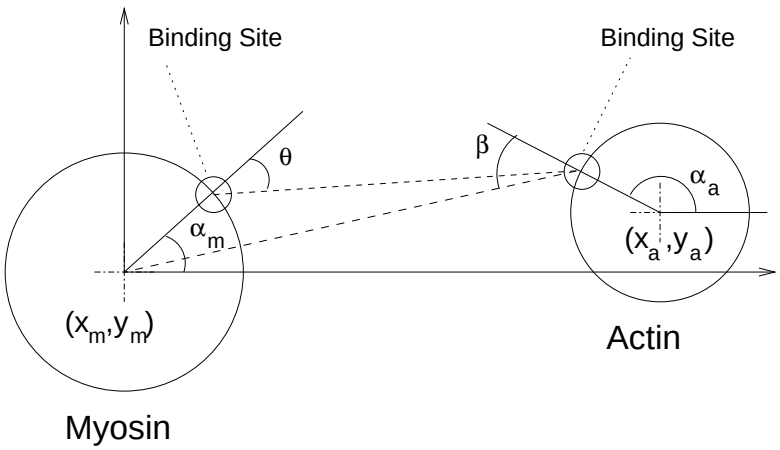

Figure 12: Sketch of the geometric setup used to compute the binding angles.

regularly continues in all directions. Obviously the structure is rotationally symmetric with a rotation angle of 120°. Although the state of each head is tracked separately, the six actin fibers surrounding the myosin fiber are put together into two explicit actin fibers denoted by A_1 and A_2. This is reasonable as each actin fiber can be bound by one of the heads of one of the three surrounding myosin fibers. Therefore the bindings of all three heads on one crown, which in a real setting would be bound to different actin fibers, are bound to the A_1 and A_2 fibers. As those do not receive bindings from other myosin fibers, the overall number of bindings on one actin fiber should be equivalent to the real case, where one actin fiber receives bindings from three myosin fibers.

The currently implemented approach analyses the angles on a case by case basis. I.e. for each combination of head and crown an explicit formula has been derived to determine the angles β and θ. Although this part of the code seems to work fine, it is potentially prone to errors. To verify the above mentioned code an alternative procedure was developed, which computes the angles β and θ for a general case. Then the different cases are casted into this general framework.

This general procedure looks at the geometric relations between a single myosin and a single actin fiber. The setup of this general case is shown in Fig. 12. The required parameters are the center coordinates of both fibers (x_a, y_a) and (x_m, y_m)), the radius of the fibers r_a, r_m and the angle of the

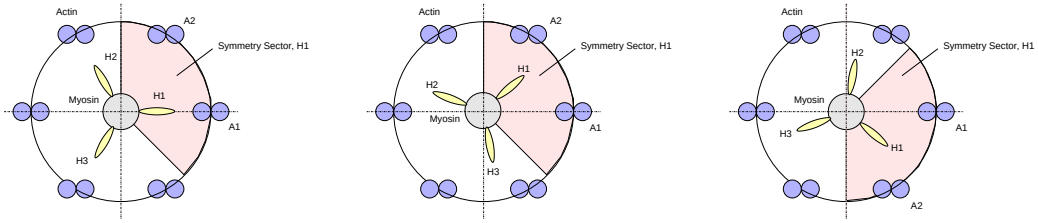

Figure 13: Sketch of the Three Crowns on the myosin Fiber

binding sites with respect to the X-axis (α_a, α_m).

In a first step the algorithm computes the cartesian coordinates of the actin- and myosin binding site denoted by (x_{ba}, y_{ba}) and (x_{bm}, y_{bm}). Then the coordinate system is shifted to place the origin into the center of the myosin binding site. For the angle between the X-Axis and actin binding site we have:

$$\tilde{\theta} = \arctan\left(\frac{y_{ba} - y_{bm}}{x_{ba} - x_{bm}}\right). \tag{5}$$

Subtracting the angle α_m, gives then the binding angle θ:

$$\theta = \tilde{\theta} - \alpha_m. \tag{6}$$

The angle β is determined with a very similar procedure. First the origin of the coordinate system is shifted to the center of the actin binding site. After that the angle between the center of the myosin fiber and the X-Axis is computed:

$$\tilde{\beta} = \arctan\left(\frac{y_a - y_{bm}}{x_a - x_{bm}}\right) \tag{7}$$

and corrected by the actin angle:

$$\beta = \alpha_a - \tilde{\beta}. \tag{8}$$

4.3 David Smith's Lattice

This lattice model was originally used in D.Smiths original 9 state model to include the lattice structure. To be able to compare the results which were obtained with the original code with the results from the Monte Carlo simulation the algorithm was reimplemented. It has many similarities with the

code for the vertebrae lattice. The main difference is that it does not modify the binding probabilities according to the axial configuration but instead strictly differentiates the binding sites in those, where the myosin head can bind and those where no binding is possible under any circumstances.

David, geom_david.prm

```
# Listing of Parameters
# ---------------------
subsection actin
  set a_actin1     = 50.4
  set a_actin2     = 50.4
  set actin_period = 13
  set dx_actin     = 2.77
  set e_actin1     = 1.27e3
  set e_actin2     = 1.27e3
  set length_actin = 1050.0
end

subsection myosin
  set a_mysoin      = 50.0
  set dx_myosin     = 14.3
  set e_mysoin      = 3.52e3
  set myosin_heads  = 49
  set myosin_offset = 125.0
end

subsection numerics
  set model = C4
  set xamax = 15.0
end
```

4.4 Flight Muscle

In insect flight muscle, the 3D lattice structure differs from that found in striated muscle of vertebrates. The myosin *crowns* have 4 instead of 3 myosin heads each. Furthermore the angle between two crowns in not 40^o but 45^o.

Therefore the axial structure of the myosin crowns repeats at every second crown.

Furthermore, the different structural setup requires three actin fibers instead of two in the model of the vertebrae lattice structure.

Note: Although this lattice model is implemented it has not yet been thoroughly verified.

`Flight`, `geom_flight.prm`

```
# Listing of Parameters
# ---------------------
subsection actin
  set a_actin1     = 50.4
  set a_actin2     = 50.4
  set a_actin3     = 50.4
  set actin_period = 13
  set dx_actin     = 2.77
  set e_actin1     = 1.27e3
  set e_actin2     = 1.27e3
  set e_actin3     = 1.27e3
  set length_actin = 1050.0
end

subsection axial
  set d_actin_myosin = 30.0
  set radius_actin   = 3.5
  set radius_myosin  = 7.0
end

subsection myosin
  set a_mysoin      = 50.0
  set dx_myosin     = 14.5
  set e_mysoin      = 3.52e3
  set myosin_heads  = 49
  set myosin_offset = 125.0
end

subsection numerics
```

```
  set xamax = 15.0
end
```

5 Activation Models

So far the biochemical mechanisms of muscle activation haven't been understood completely. Due to the accurate model of the molecular structure of muscle fibers, the Monte Carlo simulation code provides an excellent testbed for examining different hypotheses of muscle activation.

One common property of the activation models is the protocol file, which prescribes a time course of the parameter that regulates the activation. While this is usually the calcium concentration, other possibilities may appear in some of the simpler models.

The filename of the protocol file for the activation is `ca_protocol.dat` and the general structure is equivalent to that of the protocol file for mechanics (cf. subsection 3.1). In contrast to the protocol file for the mechanical loading, the calcium protocol file does not have different boundary conditions. Hence, the functions always prescribe the time course of the calcium concentration or an alternative activation parameter.

5.1 Fixed Number of Blocked Sites

This model of activation blocks or opens a prescribed percentage of the actin binding sites. It does not use any parameter file. The values in the activation protocol file should take values between zero and one. They prescribe the ratio of blocked actin sites to open actin sites. A random process is used to assign the state open or closed to the actin sites. Even when myosin heads are still bound to an actin site, the actin site can be switched to closed state.

5.2 6 State Rigid Segment Model

The 6 state model with rigid segments is based on a biochemical model of the actin sites that was proposed by D. Smith and M. Geeves in [4].

```
# Listing of Parameters
# ---------------------
subsection six_state
  set e0   = 0.0
  set ep   = 0.001
  set k1t  = 1.24e6
  set k2t  = 0.16e6
  set k_21 = 100.0
  set k_25 = 10000.0
  set k_32 = 100.0
  set k_36 = 100000.0
  set k_65 = 10.0
  set lam  = 10.0
end
subsection structure
  set segment_length = 7
  set strict_closing = 1
end
```

5.3 2 State Rigid Segment Model

5.4 Flexible Chain Model

The most advanced activation model is the flexible chain model. It incorporates the flexibility of the tropomyosin chain, which covers the actin binding sites. One of the drawbacks of this model is the tight coupling between the biochemical reactions of the myosin head and the actin binding site. This coupling requires specifically designed models of the actomyosin cycle.

The parameters are defined in the file `mg_old.prm`:

```
# Listing of Parameters
# ---------------------
subsection chain
  set actin_distance = 5.5e-9
  set phi_minus      = -0.3
  set phi_plus       = 0.2
  set sd_fac         = 0.6
```

```
  set segment_length = 7
  set xi             = 4.5e7
end
subsection rates
  set kb_eff = 25.0
  set km     = 2.4e6
  set kps    = 4820
  set kt_eff = 50000.0
end
```

6 Huxley's 1957 Model

This was the first model, which went away from the concept of simple mechanical elements and instead attributed the macrocopic behaviour of the muscle to the microscopic distribution of crossbridges.

One of the two states represents a detached myosin head, while the other state is used for attached myosin heads. The rate functions of the model are:

$$k_{12}(x) = \begin{cases} x(f_1/h) & \text{for} \quad 0 \le x < h \\ 0 & \text{else} \end{cases} \tag{9}$$

$$k_{21}(x) = \begin{cases} x(g_1/h) & \text{for} \quad 0 \le x \\ g_2 & \text{else} \end{cases} \tag{10}$$

It is selected by using the parameter `Huxley` in the file `simulation.prm`. The model parameters can then be specified in `model_huxley.prm` which typically looks like:

```
# Listing of Parameters
# ---------------------
subsection huxley
  set f1 = 48.5
  set g1 = 11.2
  set g2 = 208.0
  set h  = 15.6
end
```

Parameter	Type	Description
actin_distance	Floating Point (> 0)	Distance between two actin sites. For accuracy this number should coincide with the number that is used in the lattice geometry parameter file.
phi_minus	Floating Point	Tropomyosin chain angle (in radians) when the tropomyosin chain is pulled down by troponin.
phi_plus	Floating Point	Tropomyosin chain angle (in radians) when the tropomyosin chain is pulled up by a myosin head in R-state.
sd_fac	Floating Point	Scaling factor for the standard deviation that is computed with the numerical model. For details refer to the technical report about the flexible chain model.
segment_length	Integer	The number of actin sites between two troponin sites (including the troponin sites). I.e. a number of 7 means that there are 6 regular actins between two troponin sites.
xi	Floating Point	Parameter ξ in the original paper about the flexible chain model. It defines the elasticity of the flexible chain.
kb_eff	Floating Point	Rate for binding of the troponin site. This is a scaling factor that yields the effective rate together with the tropomyosin chain position and standard deviation.
kt_eff	Floating Point	Rate for unbinding the troponin site.
km	Floating Point	Binding of Myosin. This parameter has a slighly different meaning in the coupled activation/mechanics model. It is a scaling factor that works on top of the regular myosin binding rate, which is multiplied by another factor that represents the current state of the tropomyosin chain.
kps	Floating Point	Scaling factor for the powerstroke state. Since the actual strain dependent transition rate towards the powerstroke comes from the used model of the actomyosin cycle, this factor will only be included into the overall reaction rate. It can be used to tune the powerstroke transition to match the results without

7 Duke 3 State Model

In Huxleys model the force originates in the unsymmetric rate functions, which ensure that the Mysosin heads preferably attach on the actin fiber on positions where they immediately generate force through the crossbridge elasticity. A more realistic model introduces an additional attached state where the myosin head attaches in a neutral position and the force is generated by a swinging arm motion of the myosin head.

One of these 3 state model was proposed by Duke et.al. in [2]. It uses the following rate functions:

$$k_{12}(x) = c_3 \exp\left(\frac{-Kx^2}{2k_BT}\right) \tag{11}$$

$$k_{23}(x) = \begin{cases} k_{1\to 2} & \text{for} \quad k_{32}(x) \le k_{32}^{cap} \\ k_{32}^{cap} \exp((G_2(x) - G_3(x))/k_BT) & \text{else} \end{cases} \tag{12}$$

$$k_{31}(x) = k_{ADP}^0 \exp(-K\delta(x - x_0)/k_BT) \tag{13}$$

$$k_{13}(x) = 0 \tag{14}$$

The remaining reverse reaction rates k_{21} and k_{32} are determined by Gibbs relation:

$$\frac{k_{ij}}{k_{ji}} = \exp((G_i(x) - G_j(x))/k_bT). \tag{15}$$

and the following energy levels:

$$G_1 = 0 \tag{16}$$

$$G_2 = c_1 + Kx^2/2 \tag{17}$$

$$G_3 = c_2 + K(x + d)^2/2. \tag{18}$$

The crossbridge stiffness K is equal to `K_xb` in the global parameter file `simulation.prm`. Finally the relation between the parameter file and the mathematical expressions above is given in Tab. 8.

It is selected by using the parameter `Duke` in the file `simulation.prm`. The model parameters can then be specified in `model_duke.prm` which typically looks like:

```
# Listing of Parameters
```

k_BT	k_BT
d	D
c_1	GBind · k_BT
c_2	c_1 + GStroke · k_BT
δ	delta
γ_1	gamma1
$k_{1\to 2}$	k_12
k_{bind}	k_bind
k_{ADP}	k_adp
k_{32}^{cap}	k_21_cap

Table 8: Relation between mathematical expressions and entries in parameter file

```
# ---------------------
subsection duke
  set D        = 10.5
  set GBind    = 6.4
  set GStroke  = 15.0
  set delta    = 1.0
  set kBT      = 4.14
  set k_12     = 1000.0
  set k_21_cap = 100.0
  set k_adp    = 71.0
  set k_bind   = 170.0
end
```

8 David Smith's 9 State Model

For a detailed description of the parameters in the 9 state model, refer to [5] and [6]. In the following parts only the sections, which are specific to the Monte Carlo code will be discussed.

The first and most important selection is the type of model that should be used. In the present version of the program, the C3 and C4 version of D.Smiths model are implemented. They can be selected by setting the parameter `model` in the first subsection to either `C3` for the C3 version or to

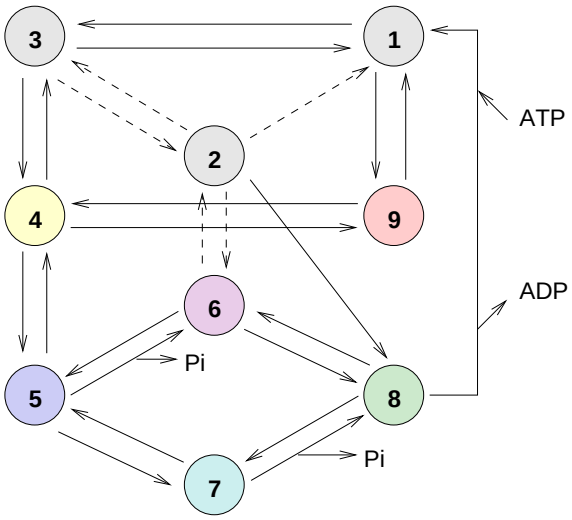

Figure 14: Scheme of the biochemical states of the myosin head in the 9 state model.

C4.

Due to program internals, the use of the exact rate functions is very slow. Therefore it is recommended to use a linear or stepfunction approximation of the exact rate functions which is significantly faster. The type of approximation can be selected by setting the parameter `interpolation` to either `Exact`, `Linear`, or `Step` where the meaning of these values is obvious.

The algorithm for computing the approximate rate functions precomputes the exact rate functions on a set of support points before the actual computation starts. When the Monte Carlo algorithm requests some rate function, the algorithm looks for the closest precomputed support points and either returns the value on those support points or determines a linear interpolation between the two neighbouring support points.

Clearly the accuracy of the approximation depends on the resolution of these support points. The parameter `dx` determines the step size between two support points, `xs` defines the first support point and `xe` the last one. The default value should be sufficiently accurate.

It is selected by using the parameter `David9` in the file `simulation.prm`. The model parameters can then be specified in `model_david.prm` which typically looks like:

```
# Listing of Parameters
# ---------------------
subsection david
  set model = C4
end

subsection misc
  set dco    = 7.0
  set eta1   = 10.0
  set eta2   = 10.0
  set eta3   = 10.0
  set g1     = 314
  set g2     = 631
  set g3     = 1e9
  set gmax   = 1e5
  set kmax   = 1e6
  set lambda = 1.0
end

subsection numerics
  set dx            = 0.02
  set interpolation = Linear
  set xe            = 20.0
  set xs            = -20.0
end

subsection powerstroke
  set h  = 4.0
  set hd = 4.0
  set ht = 1.0
end

subsection rates
  set k_12 = 0.0
  set k_13 = 100.0
  set k_18 = 5000.0
  set k_19 = 100.0
```

```
  set k_21 = 1.0
  set k_23 = 1e-3
  set k_26 = 5.0
  set k_28 = 100.0
  set k_31 = 10.0
  set k_32 = 0.1
  set k_34 = 100.0
  set k_43 = 10.0
  set k_45 = 10000.0
  set k_49 = 8.0
  set k_53 = 0.0
  set k_54 = 200.0
  set k_56 = 50.0
  set k_57 = 2.0e5
  set k_62 = 0.2
  set k_65 = 5.0
  set k_68 = 2.0e5
  set k_73 = 0.0
  set k_75 = 100.0
  set k_78 = 80.0
  set k_81 = 5000.0
  set k_82 = 0.002
  set k_86 = 100.0
  set k_87 = 8.0
  set k_91 = 10.0
  set k_94 = 80.0
end

subsection state4
  set cc = 0.8
  set dm = 4.61
  set dp = 4.61
end

subsection state5
  set cc = 0.25
  set dm = 7
  set dp = 35.3
```

```
end

subsection state6
  set cc = 0.25
  set dm = 7
  set dp = 20.6
end

subsection state7
  set cc = 0.8
  set dm = 5
  set dp = 26.2
end

subsection state8
  set cc = 0.8
  set dm = 5
  set dp = 21.6
end

subsection state9
  set cc = 0.8
  set dm = 4.61
  set dp = 4.61
end
```

References

[1] Bathe, Klaus-Jürgen: *Finite Element Procedures*. Prentice-Hall, Englewood Cliffs, N.J., 1996.

[2] Duke, T.A.J.: *Molecular model of muscle contraction*. Proc. Natl. Acad. Sci. USA, 96:2770–2775, 1999.

[3] McMahon, Thomas A.: *Muscles, reflexes, and locomotion*. Princeton University Press, Princeton, N.J., 1984.

[4] Smith, D. A. and M. A. Geeves: *Cooperative Regulation of Myosin-Actin Interaction by a Continuous Flexible Chain II: Actin-Tropomyosin-Troponin and Regulation by Calcium*. Biophysical Journal, 84:3168–3180, May 2003.

[5] Smith, D.A., M.A. Geeves, J.Sleep and S.M. Mijailovich: *Towards a unified theory of muscle contraction. I: Foundations*. Biophys. J., 2007. submitted.

[6] Smith, D.A. and S.M. Mijailovich: *Towards a unified theory of muscle contraction. II: Predictions with the mean-field approximation*. Biophys. J., 2007. submitted.