Rapid Rehabilitation and Return to Function for Amputee Soldiers (R3FAS)
Subcontract to Orchard Kinetics LLC
Funded by State of Ohio Grant TECH-009

Software Package 1 and Progress Report
Ton van den Bogert (bogert@orchardkinetics.com)
October 27, 2011

Introduction

This document presents Software Package 1, and a report of work that was done at Orchard Kinetics with this software package. The document does not attempt to anticipate all questions that the reader may have. It is assumed that the reader will contact me if any questions arise.

It certain parts of the software not work on your system, this may be due to version differences in Matlab, Windows, screen resolution, etc. Please do not hesitate to contact me if there are any problems!

Contents of Software Package 1

package1.pdf	This file
gait2d.mexw32	Musculoskeletal model, a Matlab MEX function for 32-bit Windows
gait2d.m	API documentation for musculoskeletal model
gait2d_reference.pdf	Full reference and documentation for musculoskeletal model
gait2d_test.m	Matlab application to test the musculoskeletal model
gait2d_par.xls	Excel file with musculoskeletal model parameters
dyn.m	Matlab function for combined musculoskeletal-prosthesis model
optim.m	Matlab function that solves the optimal control problem (used by script.m)
script.m	Matlab script to set up and solve a series of optimal control problems
initmodel.m	Matlab function to initialize the model before using dyn.m
simulate.m	Matlab code to run forward dynamic simulations
stick.m	Matlab code to generate stick figure plots
anim.m	Matlab code to generate stick figure animations
report.m	Matlab code to generate a graphical report from an optimal control result
ipopt.mexw32	General purpose NLP solver IPOPT. You may need to move this file to "My Documents/MATLAB" if it complains about missing modules.
Winter/	Folder with normal gait data and associated optimal control results

What You Need

1. A 32-bit Windows operating system. Development was done on Windows 7, but XP and Vista should work. Please report problems.
2. Matlab. Development was done on Matlab 2011b, but versions no older than 2007a should be able to run the software. Please report any problems.
3. Optional: SNOPT solver for Matlab (from www.tomopt.com). Development was done with a version from early 2007. Later versions should work, but please report problems. SNOPT is optional because the software also works with IPOPT, which is included in the package.

Gait Data

The folder named "Winter" contains three data files which contain normal patterns of joint angles and ground reaction forces, at three speeds. Marko Ackermann originally copied the data manually from [1]. The files are:

- Winter_slow.mat
- Winter_normal.mat
- Winter_fast.mat

Each file contains a Matlab structure, named "gait", which contains the data:

```
>> load Winter/Winter_normal.mat
>> whos
  Name      Size             Bytes  Class     Attributes

  gait      1x1               4922  struct

>> gait

gait =

    triallist: {'Data from DA Winter: Biomechanics and Motor Control of Human Gait'}
          dur: [1.1396 0]
        speed: [1.3250 0]
         data: [51x5 double]
           sd: [51x5 double]

>>
```

dur is the duration of the gait cycle and its standard deviation, speed is the walking speed and its standard deviation. The matrix named "data" contains a full gait cycle of data (0% to 100%, in steps of 2%) for five variables: hip angle, knee angle, ankle angle, horizontal GRF, vertical GRF. The matrix sd is the between-trial standard deviation, measured at each time of the gait cycle.

You can use any other gait data (or running, sit-stand etc) with the software as long as it is formatted and stored in this way. The only requirement is that the movement is cyclic and that the file contains exactly one movement cycle.

When you start using your own subject data, I would recommend to store all subject-related files (gait data, and scaled musculoskeletal model XLS file, see below) in a separate folder for each subject.

Musculoskeletal Model

The musculoskeletal model has 50 state variables and 16 control variables (one for each muscle). It is fully documented in gait2d_reference.pdf.

During initialization, it reads many parameters from an Excel (XLS) file. The "standard model" is the file gait2d_par.xls. If you edit the file, please give it a new name so that the original remains intact. This XLS file is where you can scale the model to a subject's body mass and height. When you change

either of these, you will see other model parameters change automatically. You can also hard-code any parameter my replacing the formula by a number. This is what you would use, for instance, to model a reduced mass in shank and foot on the prosthetic side.

The model equations are coded inside the Matlab MEX function gait2d.mexw32, and the inputs/outputs are documented in gait2d.m. The source code for this function is proprietary, but the model equations are all included in gait2d_reference.pdf. The MEX function has a few additional undocumented features that were especially implemented for the R3FAS project. These features are also accessible via the Excel file, but having them in the API is more convenient because it makes them programmable. The following features are all used during initialization in initmodel.m:

gait2d('Set','Right BKA', anklestiffness);

> Make a below-knee amputee (BKA) model. Removes all ankle muscles on right side (Gastroc, Soleus, Tibialis Anterior), and add a torsion spring at the ankle with the specified stiffness value(in Nm / rad).

gait2d('Set','Right AKA', anklestiffness);

> Transforms a model into an above knee amputee (AKA) model. This is the same as BKA, but also removes the knee muscles on right side (Vasti, Rectus Femoris, Hamstrings). Usually a prosthetic knee model is added also, but that is implemented in dyn.m, not inside the gait2d model.

mass = gait2d('Get','Total Mass');

> Ask for the total mass of the model (in kg).

Fmax = gait2d('Get','Fmax');

> Ask for the maximal force values of all the muscles. The output is a 16x1 matrix with Fmax values.

You may notice (especially in animations) that the ground contact is rather soft in the model. I used a contact stiffness of 20 kN/m (row 4 in gait2d_par.xls). In previous work I had used 100 kN/m which cause much less penetration of heel and toe. With the stiffer contact, I found that the K2 power burst in the knee [12] was nearly absent, because the contact force moves to the forefoot too soon. This would not be desirable in this project. The softer contact seems to be a better representation of a deformable foot, and also results in more realistic joint moments and powers, especially at the knee.

Combined Musculoskeletal-Prosthetic Model

This is implemented in dyn.m. To support all possible prosthetic limb models, including those with internal state variables, we added four state variables and two control variables to the musculoskeletal model. The combined model therefore has 54 state variables u (and 54 state equations) and 18 control variables u. Not all prosthetic models use all variables, but the software will simply make the unused variables zero. This way, all model variations have the same number of state and control variables,

which makes it possible to use a solution from one model as initial guess for solving another type of model.

The combined model is implemented as an implicit differential equation:

f(x, dx/dt, u) = 0

The Matlab code in dyn.m computes the vector f and its three Jacobian matrices df/dx, df/dxdot, df/du. (xdot is short for dx/dt)

The dyn.m code looks at the value of the model.type variable, and then decides which prosthetic model to use. The following types are implemented:

'able'

This is the able-bodied model. The additional state variables are meaningless, and dyn.m includes state equations that make those variables zero when the model is solved:

f(51) = x(51)
f(52) = x(52)
f(53) = x(53)
f(54) = x(54)

The two additional control variables are meaningless also. The optimal control cost function will drive those to whatever is optimal, but of course the result is meaningless.

'bka'

This is the BKA model. The additional state variables and state equations are the same as for 'able'. The prosthetic foot/ankle is implemented inside gait2d.m as a torsion spring, and this is activated during model initialization in initmodel.m.

'aka'

This is an above-knee amputee model with the CCF knee. The knee is modeled using four differential-algebraic equations as described in the ASME JBE manuscript [2]:

$$\begin{aligned} \dot{s} - v_1 &= 0 \\ u_1(t)^2 C_1^2 \left(k\, s - \frac{M}{R\,A} - B_1 v_1 \right) - v_1 |v_1| &= 0 \\ R\,A\,\dot{\phi} - v_1 - v_2 &= 0 \\ u_2(t)^2 C_2^2 \left(\frac{M}{R\,A} + B_2 v_2 \right) + v_2 |v_2| &= 0 \end{aligned}$$

The additional state variables are:

x(51)	Volume of fluid stored in the accumulator (s)
x(52)	Flow rate in valve 1 (v1)
x(53)	Flow rate in valve 2 (v2)
x(54)	Knee moment generated by actuator (M)

The additional control variables are the controls for the two valves.
The model parameters are documented below when we go through script.m.

The C-Leg model is implemented as a special case of the CCF knee. We get the C-Leg model by setting C_1=0. This prevents fluid flow through the high pressure valve (v1=0) and the accumulator no longer matters, so we can set s=0 also. Valve 2 acts as a controlled damper.

The reverse engineered Mauch model still needs to be added to dyn.m. There will probably be one state variable which remembers whether the knee is in stance or swing mode.

Note that the combined model consists of differential and algebraic equations. We must therefore use implicit solution methods that are suitable for such systems. The Matlab solver for implicit differential equations (ode15i) does not work on this model because the Jacobian matrix df/dxdot does not have full rank. This package includes two solvers that work: one for open loop optimal control (optim.m) and one for simulation with a given controller (simulate.m).

optim.m

The optim function solves an optimal control problem to obtain a predictive simulation of movement. It finds state and control trajectories x(t) and u(t), for 0<t<T, for the combined musculoskeletal-prosthetic model such that:

- the system equations f(x, dx/dt, u) are satisfied at all times
- the state trajectory is perfectly periodic with period T and forward translation of v*T where v is the prescribed walking speed
- the objective function F is minimized

The objective function F has the following form:

$$F = \frac{W_{track}}{11}\left[\frac{1}{N}\sum_{i=1}^{10}\sum_{j=1}^{N}\left(\frac{s_{ij}-m_{ij}}{SD_{ij}}\right)^2+\left(\frac{s_{dur}-m_{dur}}{SD_{dur}}\right)\right]+\frac{W_{effort}}{16\,N}\sum_{i=1}^{16}\sum_{j=1}^{N}u_{ij}^{E}+\frac{W_{valve}}{2\,N}\sum_{i=17}^{18}\sum_{j=1}^{N}\dot{u}_{ij}^{2}$$

where

N is the number of time points in the gait cycle

s_{ij} is the simulated value of variable i at time j

m_{ij} is the measured value of variable i at time j (an average of human trials)

SD_{ij} is the between-trial SD of measured variable i at time j. Normalization of the tracking error by SD will ensure that variables that are very reproducible in the subject, will also be tracked more closely.

The first term is the tracking term, it encourages the model to follow measured able-bodied gait variables, i.e. to walk normally. 11 gait variables are considered in this tracking evaluation: three joint angles (left and right), horizontal and vertical ground reaction force (left and right), and the gait cycle duration. The second term is the effort term, it encourages the model to not use unnecessary muscle activation. We recommend to use an exponent E=3, which makes this term equivalent to a measure of fatigue [5]. The third term is the valve term, it penalizes the controls for position changes inside the valves. Each term can be weighted. Recommendations will be given below.

Optim.m solves the optimal control problem with methods described in [3] and [4]. The code in

optim.m is not intended to be customized by the user, only via script.m

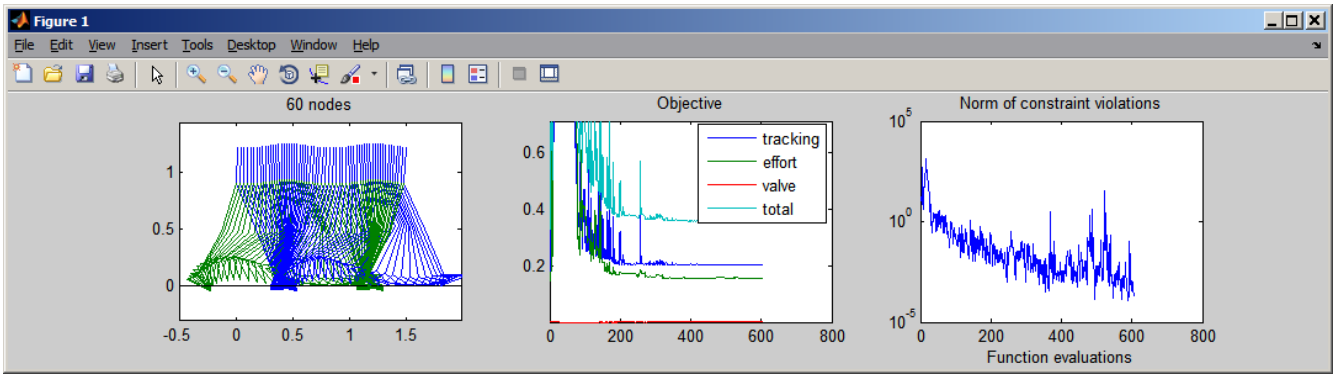
Optim.m will write its results on a file, and will write 'solved' on the screen when it finished successfully. When the optimization failed, it will print an informative message.

While optim.m is running, you will periodically get a progress report on the screen. Each progress report consists of one line of text and a figure update. The text looks like this:

```
841 -- Normc: 0.000103  Obj:  0.35561 =  0.20165 (track) +  0.15396 (effort) +  0.00000 (valves)
```

The first number (841) indicates how many state/control trajectories were evaluated so far. This number is actually higher than the number of iterations done by the solver. The second number (Normc) is the norm of constraint violations. It measures how well the system equations f(x,dx/dt,u)=0 were satisfied over the entire movement cycle. This number goes to zero during convergence. The remainder of the line reports the objective function (and its three components) for the most recently evaluated state/control trajectory.

Also a Figure window will be updated at regular time intervals:



On the left is a stick figure representation (N frames) of the most recently evaluated state/control trajectory. The middle figure shows how the objective function (and its three components) has varied during the entire course of the solution process. It should level off to a minimum during convergence. The rightmost plot shows how Normc has varied during the solution process. With problem.ConstraintTol = .001 (the recommended value), the solution process will usually not terminate until Normc goes below 0.00001 which indicates that the model equations and periodicity constraints were satisfied very well.

script.m

This is a Matlab script that can serve as a template and example for setting up an optimal control problem, or a series of problems, and solving them with optim.m

As delivered, script.m will run a series of four optimizations based on Winter's normal speed gait data:

- able bodied model
- BKA model
- C-Leg model

- CCF model

This series of optimizations requires about 35 minutes on my HP Pavillion Laptop running Matlab 2011b on Windows 7.

The user can easily customize this script, or make new versions, to run other optimizations.

We will now go through the script and explain the settings so you can customize it for your own needs.

```
movement = 'Winter/Winter_normal';
```

This tells the script that the tracking term in the optimization will use the measured gait data from the file Winter_normal.mat in the Winter folder. This is the gait data for normal walking speed. Substitute your own movement data file here as needed. You can also use 'Winter/Winter_fast' and 'Winter/Winter_slow' to use the fast and slow walking data files which are included in the package.

```
problem.Solver = 'IPOPT';
```

This tells optim.m to use the IPOPT solver. The IPOPT solver is free and included with this package. You can also specify 'SNOPT' which needs fewer iterations, but needs a better initial guess, as described in [4]. All the work presented in Progress Report (below) was done with IPOPT. SNOPT occasionally got stuck in ill-conditioned matrix situations, so IPOPT was actually more robust in that sense.

```
problem.checkderivatives = 0;
```

Keep this at zero. A value of 1 will tell optim.m to check the accuracy of the model's derivatives. This is time consuming and should only done during software testing.

```
problem.debug = 0;
```

Also for software testing, keep this at zero during the actual work.

```
problem.MaxIterations = 5000;
```

This tells the solver (IPOPT or SNOPT) how many iterations to do before giving up. If it does not converge in 5000, it will probably never converge and I recommend that you try again with a different initial guess or different model.

```
problem.ConstraintTol = .001;
problem.Tol = .0001;
```

These are the convergence criteria for the solver (IPOPT or SNOPT). These values were found to be accurate enough, but feel free to experiment. Lower values will require more iterations until convergence. If tolerance is too strict, the solver may keep iterating without ever stopping. Larger (looser) tolerance values will cause the solver to finish earlier, but possibly with a less accurate solution.

```
problem.Printinterval = 3.0;
```

This tells optim.m to give a progress report every 3 seconds.

```
problem.N = 60;           % number of collocation nodes in the movement cycle
```

The optimal control problem is solved with direct collocation on N time points in the movement cycle. See [3] for details. Higher values of N will make the optimization run more slowly, but the answer will be more detailed and more accurate. By performing series of optimizations with increasing N, we have determined that 60 is sufficiently detailed and accurate for a gait cycle. Other more complex movements may require more nodes.

```
ablemodel.parameterfile   = 'gait2d_par.xls';
ablemodel.type            = 'able';
ablemodel.datafile        = movement;
ablemodel.Wtrack    = 1;  % weight of tracking term in optimization objective
ablemodel.Weffort   = 20; % weight of muscle effort term in optimization objective
ablemodel.effort.fatigue  = 0;
ablemodel.effort.Fmaxweighted = 0;
ablemodel.effort.exponent = 3;
ablemodel.Wvalve    = 0.001;     % weight of valve cost in optimization objective
ablemodel.discretization = 'euler';
ablemodel.reducedW = 0;
```

This defines a specific XLS file for model parameters. If you have one that is scaled to a specific subject, you might use, for example: 'I:\data\subjects\subject001\gait2d_par_subject001.xls'. You get the idea. The movement data file can be specified in the same way.

Wtrack, Weffort, and Wvalve are the weights of the three terms in the optimization objective. These specific values were found to give realistic looking solutions, but experimentation is encouraged.

'euler' tells the direct collocation method is told to use the Euler discretization (see [3]). The other option is 'midpoint'. Euler is more stable, converges better, but less accurate and has a damping artifact (see http://en.wikipedia.org/wiki/Geometric_integrator). I find the Euler discretization with N=60 sufficiently accurate. This is based on comparing the results of N=60 and N=120. They are practically the same, which means that N=60 was good enough. The effort.exponent field is the E (exponent) value in the effort cost in the optimization objective. The other settings should not be changed and are for developer use only.

```
bkamodel                  = ablemodel;
bkamodel.type             = 'bka';
bkamodel.anklestiffness   = 450;       % stiffness (Nm/rad) of prosthetic ankle
```

'bkamodel' is the same as 'ablemodel', except that we tell the model that ankle muscles must be removed, and replaced by a torsion spring with stiffness 450 Nm/rad. This value is based on typical moment-angle relationships in able bodied subjects during gait (Hansen et al., J Biomech 37: 1467–1474, 2004)

```
clegmodel           = bkamodel;
clegmodel.type      = 'aka';
clegmodel.RA        = 7;   % knee actuator volume (cm^3), use zero to disable actuator
clegmodel.C1max     = 0;   % valve constant for valve 1, cm^3 s^-1 MPa^-0.5
                           % (use zero to keep valve closed)
clegmodel.C2max = 50;      % valve constant for valve 2, cm^3 s^-1 MPa^-0.5
clegmodel.B1 = 0.01;       % valve drag coefficient, MPa s cm^-3
clegmodel.B2 = 0.01;       % same for valve 2
clegmodel.L  = 0;          % actuator leakage parameter, cm^3 s^-1 (Nm)^-1
clegmodel.k  = 3.0;        % accumulator stiffness, MPa cm^-3
```

RA is an actuator constant, in cubic centimeters. It is the ratio between actuator flow rate (cm^3/s) and angular velocity of the knee (rad/s). Because of the principle of virtual work, and a fortunate cancelation of a factor 10^6 in the unit conversions, it is also the ratio between joint moment (in Nm) and fluid pressure in the actuator (in MPa). For a rotary actuator, it is the vane radius multiplied by total vane area (hence the symbol RA). For a linear actuator, it is the piston area multiplied by the actuator's moment arm.

C1 and C2 are the valve constants. We do not allow any flow in valve 1 in the C-Leg model, so C1 is zero here. B1 and B2 are the viscous drag coefficients for the valves and associated tubing. *L* is not

used yet by the software. *k* is the accumulator stiffness, the value of 3 is roughly based on [2]. This parameter is not used in C-Leg but will be used in the CCF leg (below). This value will affect performance of the CCF leg so please experiment with it. Future versions of optim.m will be able to optimize this parameter.

```
ccfmodel            = clegmodel;
ccfmodel.C1max      = 50;               % now we allow valve 1 to open also
```

This is all it takes to create the model with CCF knee, everything else was already there in clegmodel.

```
% solve the gait for the able-bodied model
problem.model              = ablemodel;
problem.initialguess       = 'mid';
problem.resultfile         = [movement '_result_able.mat'];
optim(problem);
disp('Hit ENTER to continue with next optimization, or CTRL-C to quit');pause;
```

This section of the script is where the actual work is done. We define which model to use, which initial guess to use, where to write the results, then it runs optim.m to solve the problem.

The remainder of script.m performs three additional optimizations, each time using the solution of one problem as an initial guess for the next problem.

The cleg and ccf versions of the model have more nonlinearity (specifically: the valve equations are quadratic in the controls and flow rates) and are somewhat less likely to converge than the able and bka models. For the specific settings in script.m, with the package as delivered, all four scripted optimizations will converge. This may, however, not always be the case when you start experimenting with changing the model parameters, other movements, etc. In case of non-convergence, here are some things you can do:

1. Retry the same optimization with a different initial guess. It often works well to use as initial guess a solution for a different gait speed, or even a different movement, if it was obtained with the same model.
2. Alter the model to make it less nonlinear, redo the optimization. If that converges, use it as an initial guess for the next optimization in which you have restored the nonlinearity of the original model. The alteration that seems to work best is to increase the ContactY0 parameter (row 6 of the XLS file), to make the contact model less nonlinear. A value of 0.001 seems to converge well, but the contact force field becomes a bit fuzzy. Even more with 0.01. The foot will feel some force even when not in contact with the ground, and this causes energy loss during the swing phase. You can map this force field by doing "`gait2d_test('grf')`", and zoom in as needed. The XLS file as delivered has ContactY0=0.001. A value of 0.0001 caused convergence failure for the initial guesses defined in script.m. For better accuracy, you can use the 0.001 solutions as an initial guess for a solution of the same problem, with ContactY0=0.0001. This will converge. You will probably see little difference between the two solutions, and this would confirm that this parameter was not critical after all.

When optim.m is finished, it will write the result on the file specified in problem.resultfile. This file will contain a single struct "result", which contains the following fields:

```
result =
        x: [54x60 double]
```

```
      u: [18x60 double]
    dur: 1.1398
  speed: 1.3250
  model: [1x1 struct]
  normc: 0.00001
      f: 0.7795
```

If you understand these variables, you can customize the postprocessing applications (anim, stick, report), or write your own. So here is a brief explanation.

x and u are the state and control vectors, solved for all 60 time nodes. dur is the predicted movement duration, it will usually be very close to the dur value in the original subject gait data file. speed is the prescribed speed (=forward translation per gait cycle, divided by dur). model is the same struct that was given to optim.m in problem.model. From model you can retrieve information about which objective function was used, which gait data was used, which parameter file was used, etc. normc is the norm of constraint violations in the solution. f is the final objective function value.

anim.m

This is a simple program to generate stick figure animations from an optimization or simulation result. To run it, type

```
>> anim('name_of_resultfile')
```

or simply

```
>> anim
```

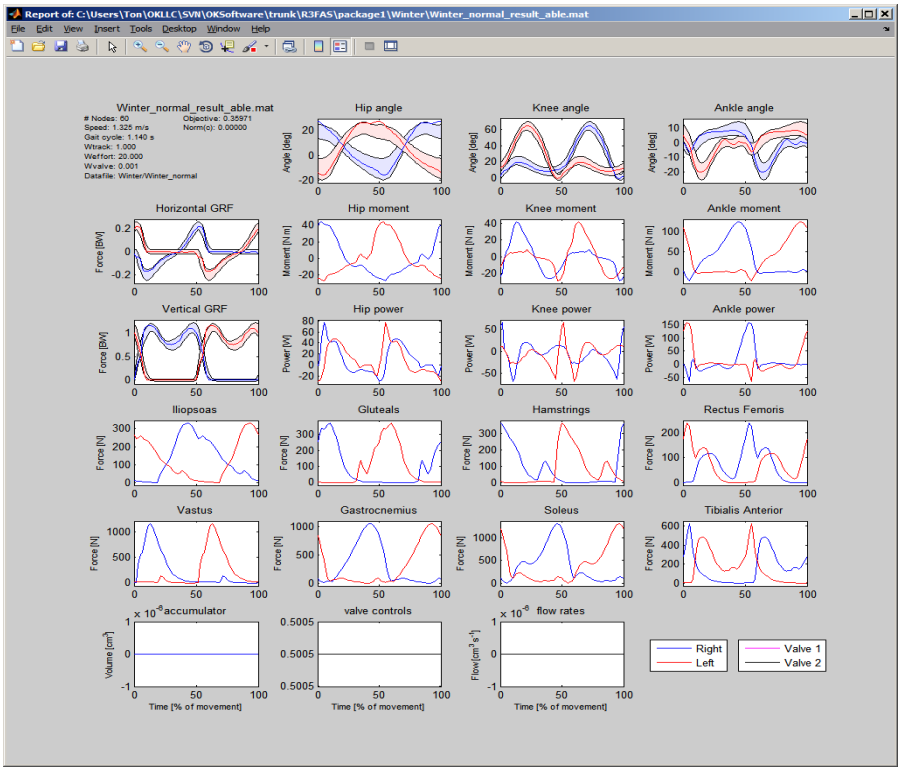
In the second case, a window will pop up that allows you to select the file you want to use. The animation will be stored in an AVI file with the same name as the resultfile, but with .avi extension. The file can be played with Windows Mediaplayer. The AVI file is rather large, so if you need to store many files, or e-mail them, I recommend compressing them with Windows Moviemaker.

stick.m

This program does the same as anim.m but produces a static stick figure drawing, showing the entire gait cycle. It will show only every 5th frame of the N frames in the result file. The image will be drawn in a Matlab Figure window. From there, you can use "Edit → Copy Figure", and then paste it into MSWord etc. You can also do "File → Save As..." to export the figure as an EPS file which can be imported in Adobe Illustrator, etc.

report.m

This program will display an extensive graphical report from an optimization or simulation result. All graphs will appear in one Figure window, as shown below.



When reading these graphs, the following may be helpful:

- The time axis goes from right heelstrike to the next right heelstrike. When the gait is symmetrical (which is usually the case with the able-bodied model), the vertical GRF plot will show that left heelstrike occurs exactly out of phase, i.e. at 50%.
- Knee moment is the sum of moments from prosthesis and from the musculoskeletal model (muscles and passive joint structures) . When we have a prosthesis in the model, knee muscles will no longer be there, but passive joint structures are still there to protect against hyperextension.
- Definitions of angles and moments and forces can be found in the gait2d reference manual (gait2de_reference.pdf).

With the report window on the screen, you can save it in several ways:

- In report window, use “Edit->CopyFigure to put the information on the clipboard. Then go to MSWord and use Edit->Paste (or CTRL-V) to insert the graphs. Best quality is obtained with Metafile option (look in Edit->CopyOptions). Unfortunately, if you do the same in OpenOffice, parts of the report do not get pasted :-(. Bitmap CopyOption always works reliably, but does not look as good.
- In report window, use File->SaveAs... to export to any other graphics file format. EPS (Encapsulated PostScript) gives best quality, but unfortunately the graphs do not quite fit on the page when I do this.
- In report window, use File->PrintPreview... to send the report to a printer or PDF file (if PDF

printer is installed). This is what I would recommend if you want to archive any of these reports.

- Use the "print" command from the Matlab console to send the report to a graphics file or printer. This can also generate a PDF file even if no PDF printer is installed on Windows. See Matlab Help for further information.

simulate.m

This program performs a single forward dynamic simulation using a given controller. The present version can only use open loop control, and loads the controls u(t) and initial state x(0) from a result file generated by optim.m.

To run a simulation, simply type "simulate(duration)" , where duration is the amount of time you want to simulate. A window will pop up to select the file that contains u(t) and x(0). The simulation will run, and the result will be stored on a file that has the same format as the files produced by optim.m. So you can use anim.m and stick.m to visualize these results of a . Report.m will also work, but the time axis may no longer represent one gait cycle, it will be the duration of the simulation.

The simulation will run faster than real-time on a typical PC.

If you simulate for a duration longer than one gait cycle, integration errors will start to accumulate and the model will eventually fall down even though the optimal control solution (one gait cycle) did not. Try it! This is nothing to worry about. This is expected because we are simulating an unstable model with open loop control.

Simulate.m contains two implicit solvers for the differential-algebraic equation f(x, dx/dt, u) = 0. The solvers are Backward (or Implicit) Euler, and Implicit Midpoint. These are decribed in http://en.wikipedia.org/wiki/Geometric_integrator. Simulate.m uses the solver specified by result.model.discretization. If the result file was produced by optim.m, the same discretization will be used.

Future versions may allow you to "plug in" a closed loop control for the valves in the prosthetic knee (u17 and u18), if that is useful to test the controllers being developed at CSU.

Future versions may also include perturbations and evaluations of stability, but without closed loop control for the muscles, the model will be very unstable and this may not be useful.

Closed loop control for the muscles is probably beyond the scope of this project, but if you are interested in this, look at work by Taga [6] and Geyer [7]. Geyer's musculoskeletal model is quite similar to ours, so his controller might work. The gait simulated by Geyer's controller is stable but may not be realistic. It is hand-crafted, without consideration of optimality, tracking or effort. This is a different approach to predictive simulation without use of optimal control. You simply put a prosthetic limb on the model, and run the simulation again. If the controller still works, you have a predictive simulation immediately. In my personal opinion, this is less scientific, because dependent on peculiarities of the controller you happen to use. Another controller, which works equally well, could give entirely different predictions.

Progress Report

Now some science.

If you run script.m as included in the package, you will generate four optimal control results in the Winter folder:

Winter_normal_result_able.mat
Winter_normal_result_bka.mat
Winter_normal_result_cleg.mat
Winter_normal_result_ccf.mat

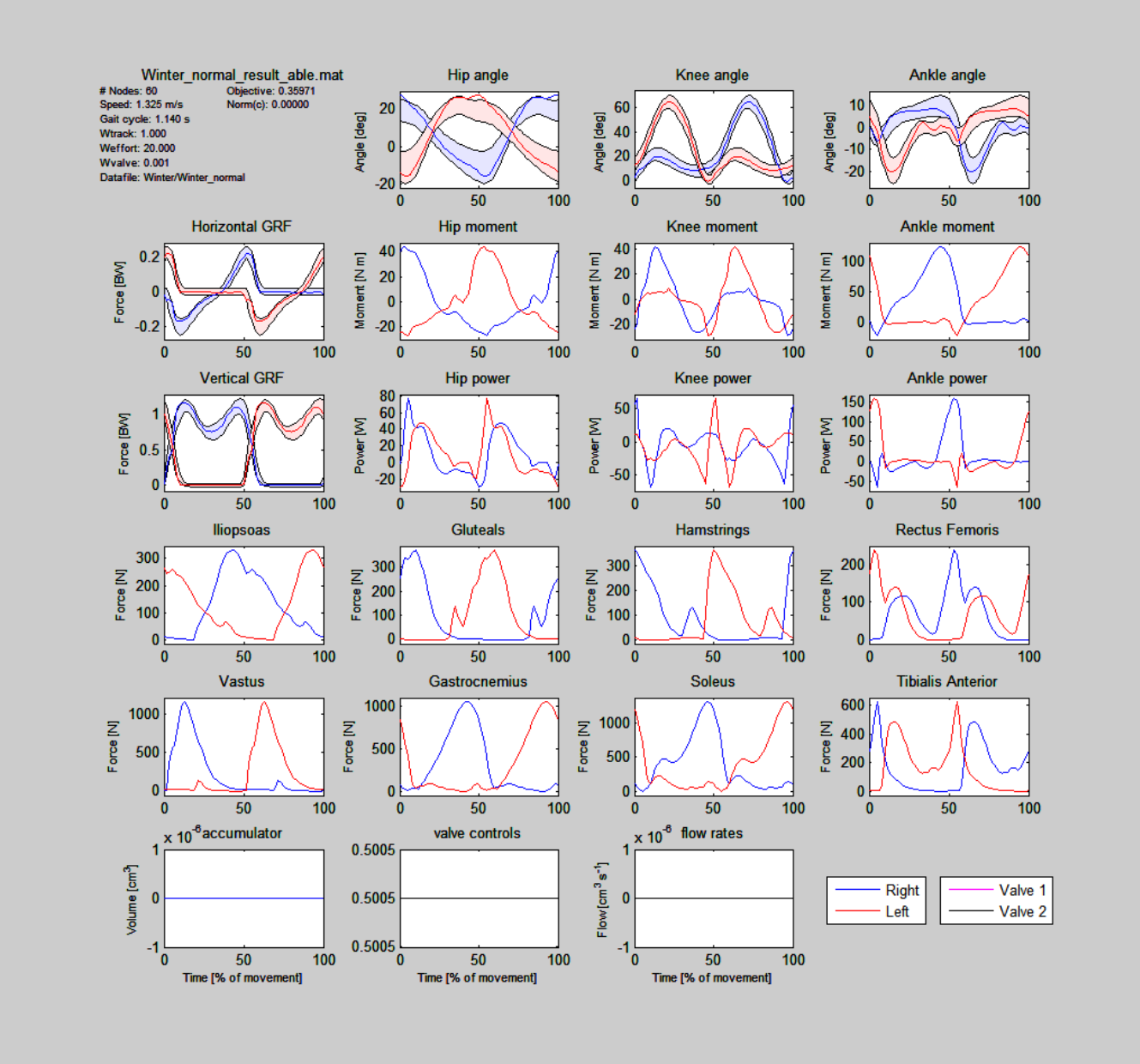
'Winter_normal' refers to the file Winter_normal.mat that was used to provide the tracking data for the optimal control problem.

The 'able' model optimization had the following console output:

```
1565 -- Normc: 0.000000  Obj:  0.35971 =  0.20333 (track) +  0.15639 (effort) +  0.00000 (valves)
```

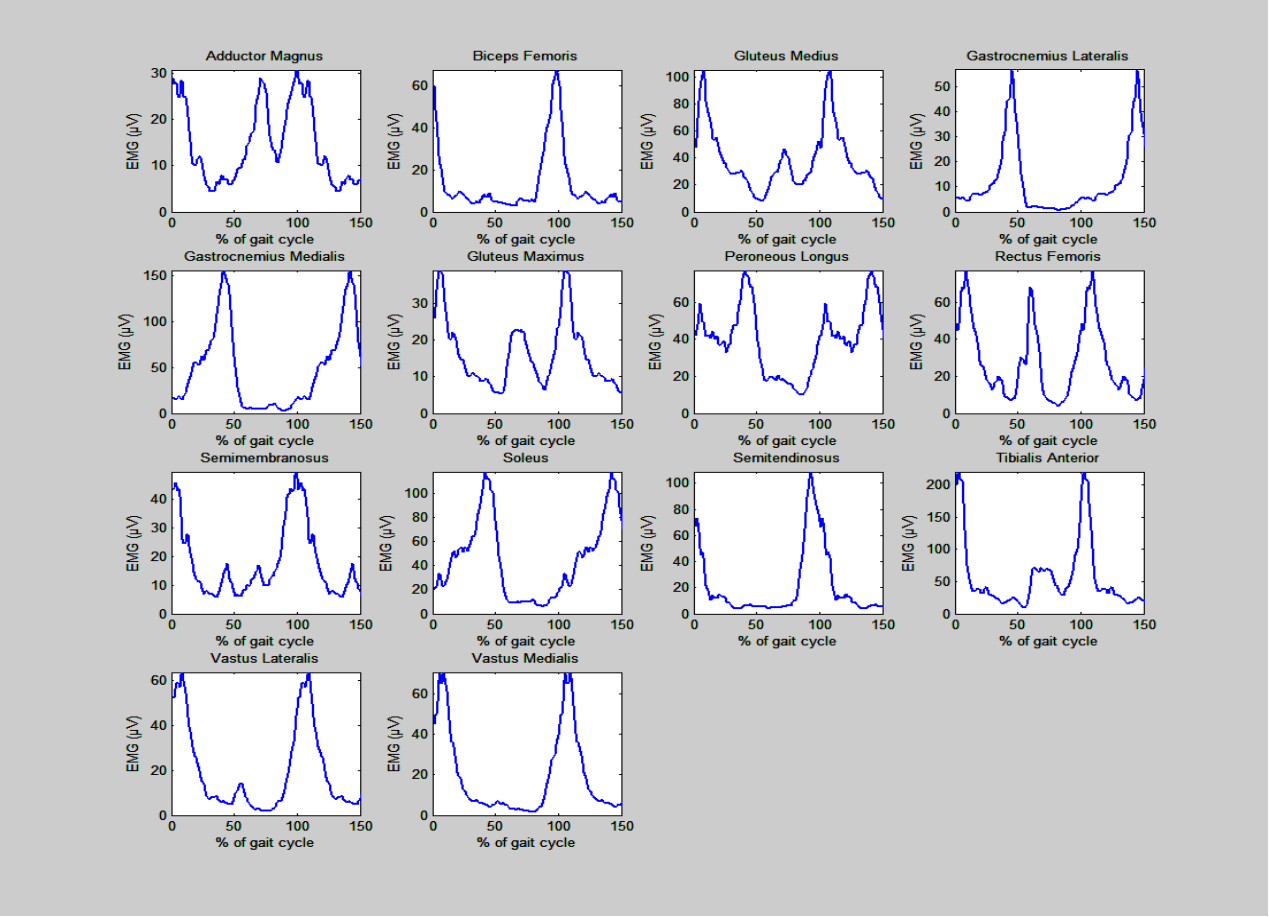
The tracking term tells us that the tracking error was, on average, 0.2 standard deviations which is very good. The effort (mean of cubed muscle controls), multiplied by the Weffort value (here: 20), is also very low which means that muscles are being used very efficiently.

The report output looks like this:

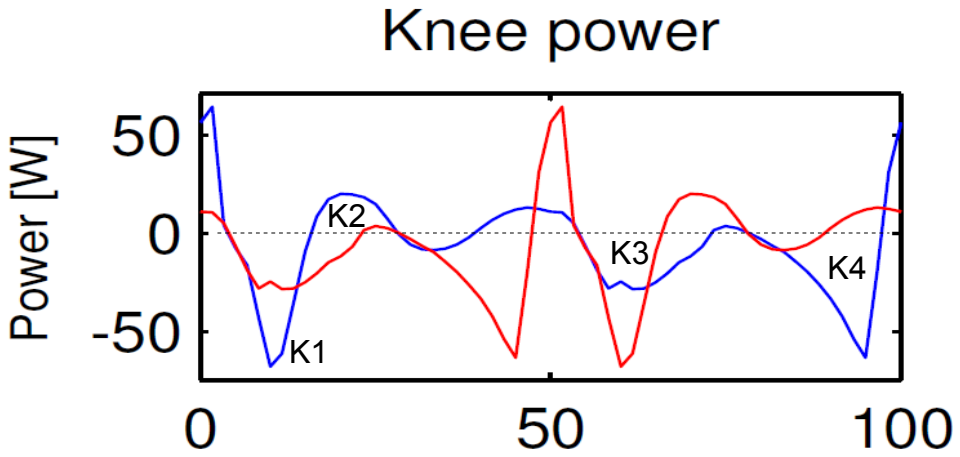


Some observations:

- We only use ground reaction force and joint angles for tracking. We do not use the joint moments (which could be measured in subjects). Nevertheless, perfectly normal joint moment profiles emerge. See, for example [8] for normal joint moment patterns. This suggests that, even though we did not tell the model how to achieve the desired ground reaction forces and joint angles, it did this with a normal coordination pattern. It did not need to do that, mechanically, but the effort term apparently made it do that.
- Further evidence of normal coordination is in the muscle force patterns. We can compare these to human EMG. The EMG graphs below were generated with the supplementary data and software from [9]. Note the horizontal axis is 150% of the gait cycle, to better show the burst around heelstrike (0% and 100%). We see excellent agreement between how the model decided to use its muscles and how humans do it.



- If we zoom in on the knee power graph, we can see the K1-K4 bursts as defined in [12]:



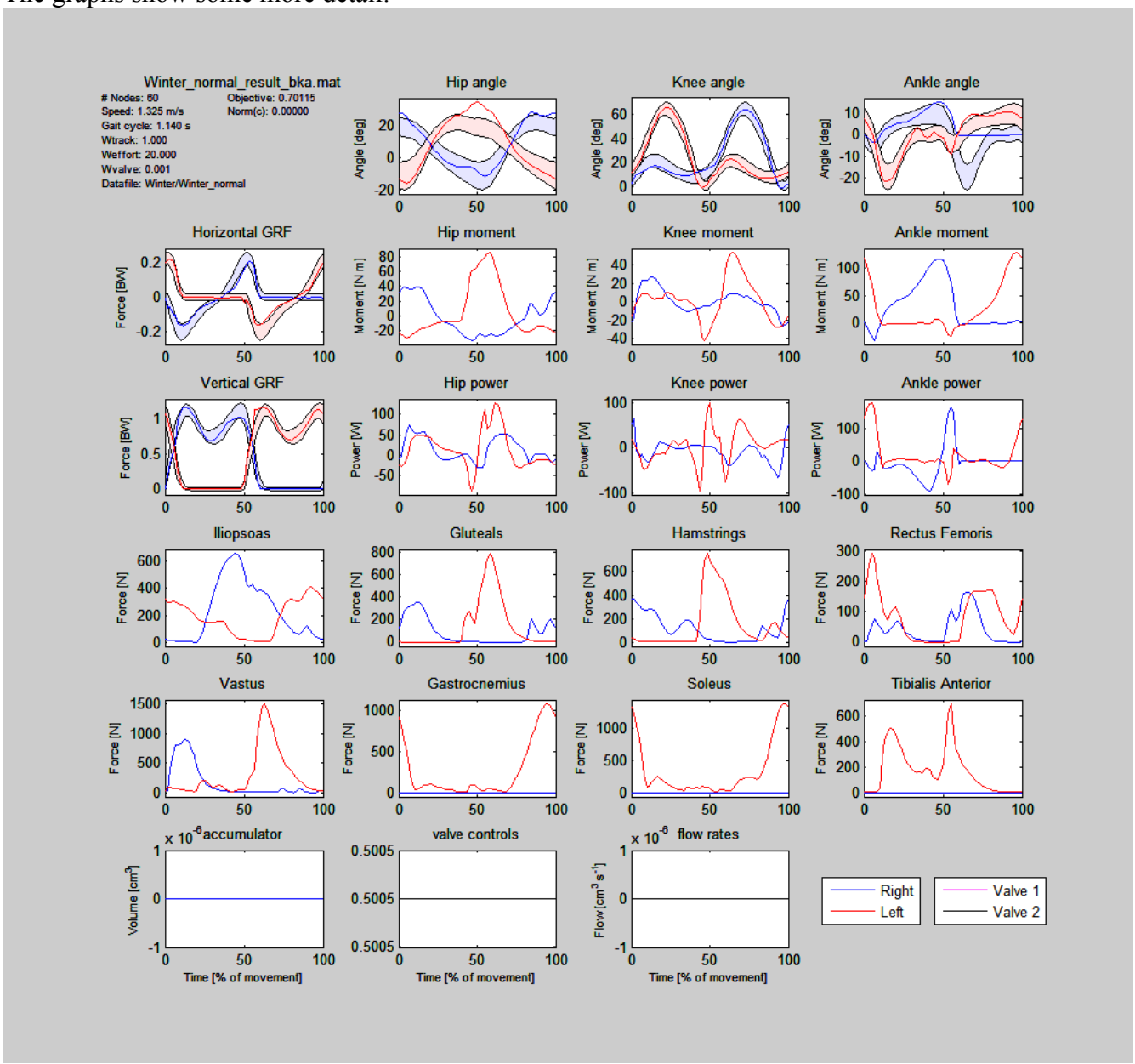
The K bursts are identified in the graph for the right knee (blue curve). K1 is the negative work that we would eventually like to store in the accumulater and re-use to produce the K2 burst, and possibly save some for later also. The K2 burst looks somewhat different from Winter's [12], but there is other human subject data that looks more like our result, for example, http://www.clinicalgaitanalysis.com/data/kinetics.jpg .

We now go to the BKA model. Its optimization result is:

```
1875 -- Normc: 0.000000  Obj:  0.70115 =  0.52961 (track) +  0.17154 (effort) +  0.00000 (valves)
```

It now has some difficulty tracking the normal movement, tracking error is increased more than twofold. Most if this is obviously from the right ankle, the passive device can't control this angle in the swing phase at all. In stance phase, the model can control the right ankle angle via muscles at other joints, which causes effort to increase. This effort increase was apparently larger than the decrease caused by the absence of the three right side ankle muscles which were amputated.

The graphs show some more detail:



Tracking is still quite good (solid curves lie within shaded areas), except hip flexion on the intact (red) side is increased. The model needs to maintain the prescribed speed (a task constraint in the optimal control) and may be making a larger step with the intact leg to compensate for the loss of push-off on

the prosthetic side.

The most remarkable part of this result is that the knee moment on the prosthetic (right) side is much smaller than on the intact side, even though all knee muscles are still intact. For some reason, it must be optimal for the model to walk this way. Below-knee amputee patients do this also [10]. Why? The only structure that couples the knee and ankle is the gastrocnemius, and this is gone in the BKA model. With an intact gastrocnemius, active knee extension stretches the gastrocnemius tendon (during 15% to 45% of the gait cycle). This stored energy is then suddenly released during plantarflexion. This way, power generated by the quadriceps can contribute to ankle function. With a prosthetic foot, you may still have a powerful quadriceps , but the coupling mechanism no longer exists. Indeed, if I run the able-bodied model with a right gastrocnemius that is not a knee flexor anymore, I get similar loss of knee extensor moment, albeit to a lesser degree. This may be part of the answer.

Innovation in prosthetic feet is currently focused on powered active devices. Hugh Herr (iwalk.com), Tom Sugar (springactive.com), and Steve Collins are some people working on these things. If my explanation is correct, such devices may not solve the problem of inhibited knee function in below-knee amputees. Instead, it may be necessary to restore the power transfer from knee to ankle in late stance, so that knee extension becomes useful again to the patient. Power transfer between joints was an active area of research in the 1980s [11]. These days, people do gait analysis instead of functional anatomy, and muscle actions are only described by joint moments, i.e. hypothetical motors inside each joint that act separately. Separately acting joints is also the prevailing paradigm in prosthetic limb engineering, but this can never replace the unique actions of multi-joint muscles. There is modeling work that aims to explain muscle function, but it is rather descriptive and not predictive in a way that it can help design a prosthetic device.

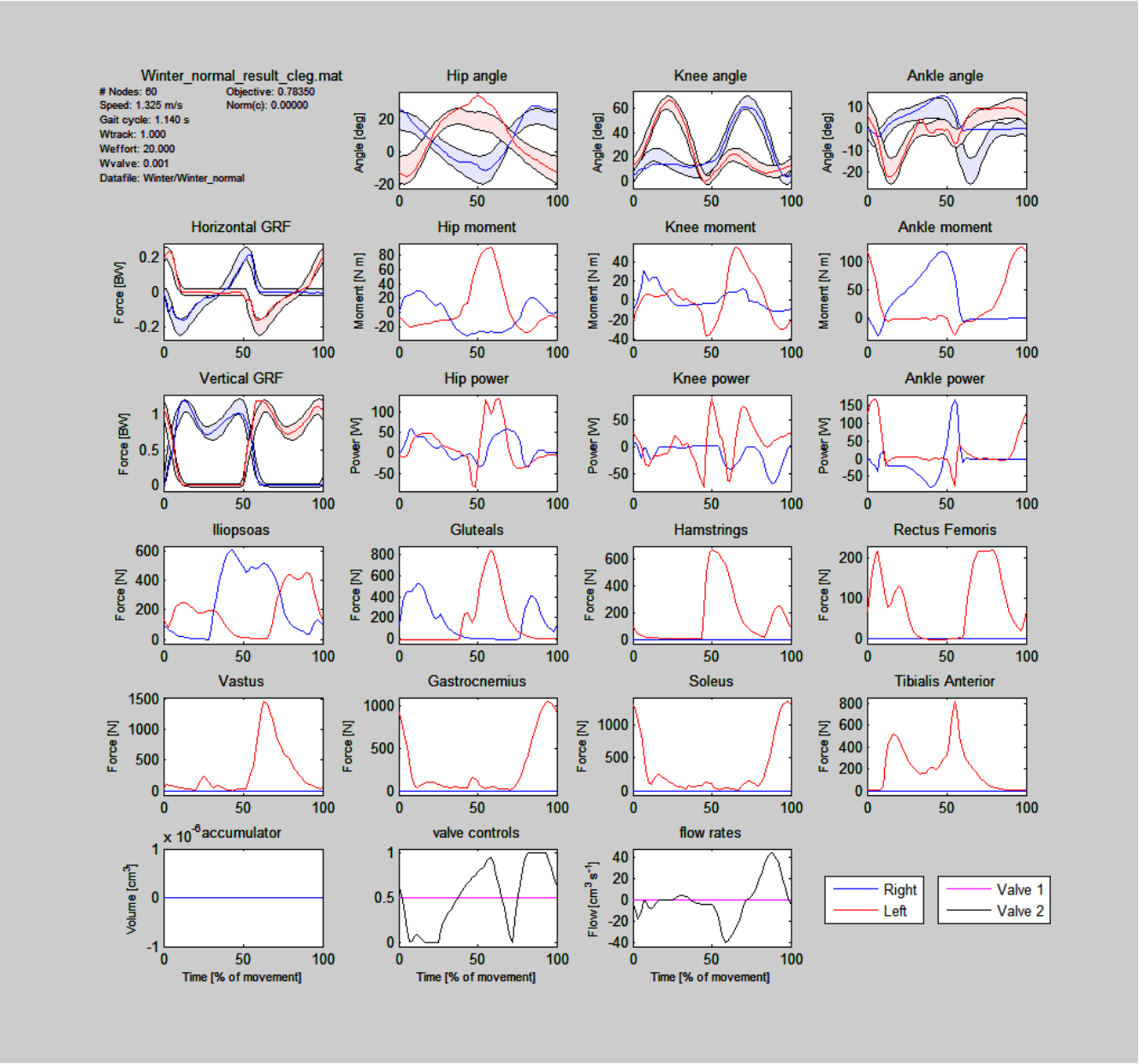
Next is the C-Leg model. We take the BKA model, remove the knee muscles and replace them with a controlled hydraulic damper as in a C-Leg device. Result of optimization:

```
4217 -- Normc: 0.000002  Obj:  0.78350 =  0.56285 (track) +  0.20860 (effort) +  0.01205 (valves)
```

Tracking and effort both got worse again, and now we also see valve action which has a cost of 0.012. We have set the valve cost weighting very low (Wvalve = 0.001), so the model will consider this to be only a secondary criterion in deciding on its optimal control. Tracking and muscle effort should probably always be more important. Tracking is not much worse than in the BKA model. The main difference is a reduction of knee flexion-extension in stance (graphs below). This goes in the same direction as what is seen in patients with C-Legs (and Mauch and Rheo knee), who basically lock the knee at full extension [13-15].

Valve 2 (the controller damper) is mainly closed in stance and open in swing, as expected, and as we previously saw in simulations of only the knee itself [2]. For some reason, the valve wants to close briefly when the knee is at maximal flexion, at about 70% of the gait cycle. It could be that this is unimportant, and only reduces the tracking and effort by a minuscule amount. If we increased Wvalve, the valve might just stay open during swing to reduce its operating cost, with little detriment to the overall system performance in terms of tracking and muscle effort. This needs to be looked at.

Of note is also the knee power curve for the C-Leg (blue curve). There are only negative bursts because a controlled damper can never produce positive power.

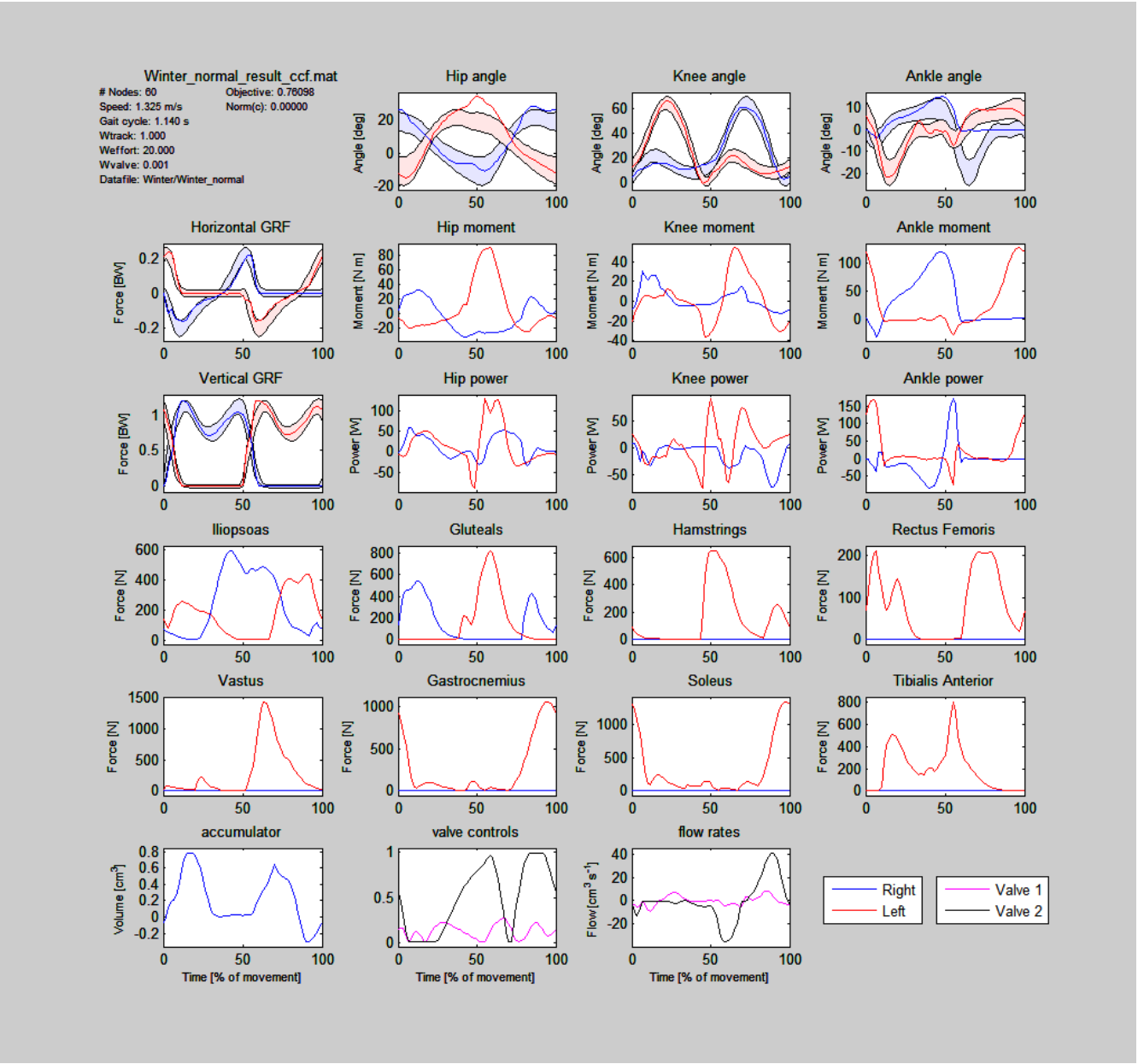


Next is the CCF knee model with controlled energy storage. Result of optimization:

```
4114 -- Normc: 0.000000  Obj:  0.76098 =  0.54796 (track) +  0.19940 (effort) +  0.01363 (valves)
```

The graphs below show that valve 1 chooses to open somewhat, mainly in stance, but not as decisively as in [2]. There is some flow in and out of the accumulator, during 0-25% of the gait cycle, indicating energy storage and return. The pattern is exactly what we expect, but the magnitude is much smaller than expected. The energy return produces only a very small (positive) K2 burst. Overall, the performance is only slightly better than the C-Leg (compare the tracking and effort values in the objective). This is disappointing but understandable if we go back to the BKA result. The BKA result shows that even with a perfectly intact knee musculature, the model prefers to reduce its knee function on the prosthetic side. The K2 burst is already nearly completely gone in the BKA model, so there

seems to be little use for a better knee that helps restore this burst, unless we improve BKA performance first (in models and in patients)!



These are preliminary results and the final word on this has not been said. Here are some limitations of what was done so far:

- The findings cannot yet be considered generalizable. I only used Winter's normal speed data. The package also includes slow and fast speed data. You may also want to use your own data, from other subjects.
- We may need to use a better prosthetic foot model. The torsion spring is nice because we keep the foot geometry the same as in the able-bodied model. This is however, not exactly how a prosthetic foot and ankle work.
- Is there any data out there in which transtibial amputees have better knee function than in [10]?

If so, we might try harder to get better knee function in our BKA model. If not, we should conclude that our model is quite good at predicting patient performance!

- The accumulator stiffness was set at 3 MPa per cubic centimeter, which may not be optimal for this set of gait data and the 7 cc actuator. Still, I do not think that a different value would make the model with CCF knee perform significantly better. This is because the BKA result suggests that even an optimal knee device will not restore normal knee function. At least on this particular gait.
- We could encourage the BKA model (and C-Leg and CCF) to have normal knee function, by tracking human subjects' joint moments in addition to angles and ground reaction forces. The decision on which variables to track is somewhat arbitrary, but I would like to track as few variables as possible. This keeps the model predictive by giving it freedom to choose how it performs the movement task. Ideally, we would want to track nothing at all, but this produces weird gaits (you can try setting Wtrack=0) which are biomechanically possible but may be too hard to control. Tiptoeing is one such solution with very low effort, but clearly not what people choose to do. Early work without tracking [3,16] did not find these weird gaits, but that was when the computational methods were not as far developed. In those days, a very good initial guess was needed to ensure convergence. This initial guess came from human data. The optimization then found a local optimum that was close to the initial guess. Other, better solutions (those weird gaits) were never found. In [16], radically different gaits were not even possible because the system states at 0% and 50% of the gait cycle were not allowed to deviate from human data. The newer implicit methods [4] suffer far less from convergence problems and local optima, so now we are more likely to find the true optimum from the entire solution space of all possible gaits.

"To Do" List

The following items are being considered to be added to the software. Please let me know if anything should be added or removed from this list.

1. Add stability analysis to simulate.m? Perturbations? Floquet analysis?
2. Add Mauch knee to dyn.m (and script.m and initmodel.m)
3. Improve the hydraulic model for the CCF knee in dyn.m, based on actual prototype being developed.
4. Allow optim.m to optimize model parameters, such as accumulator stiffness *k*.
5. in report.m: sync the graphs to right heel strike, for easier left-right comparison (optional)
6. optim.m, implement continuation method to improve convergence.
7. script.m: Always do midpoint after euler for improved accuracy. Needed?
8. report.m: if discretization used was midpoint, report variables at midpoint. This will remove some of the "jitter" in results from midpoint discretization.
9. optim.m: Implement the simultaneous tracking of multiple movement files, to design device that is optimal for a set of movements, rather than a single movement.
10. Implement closed loop control for the valves in simulate.m?
11. 3D modeling (only when NIH grant is funded).

References

[1] Winter, DA. The Biomechanics and Motor Control of Human Gait: Normal, Elderly and Pathological, 2nd Edition, 1991.

[2] van den Bogert AJ, Davis BL, Samorezov S, Smith W. Modeling and Optimal Control of an Energy-Storing Prosthetic Knee. Submitted to ASME Journal of Biomechanical Engineering.

[3] Ackermann M, van den Bogert AJ (2010) Optimality principles for prediction of human gait. J Biomech 43: 1055-1060.

[4] van den Bogert AJ, Blana D, Heinrich D (2011) Implicit methods for efficient musculoskeletal simulation and optimal control. Procedia IUTAM 2: 297-316. Open access at: http://www.sciencedirect.com/science/article/pii/S2210983811000289

[5] Crowninshield RD, Brand RA. A physiologically based criterion of muscle force prediction in locomotion. J Biomech. 1981;14(11):793-801.

[6] Taga G. A model of the neuro-musculo-skeletal system for human locomotion. I. Emergence of basic gait. Biol Cybern. 1995 Jul;73(2):97-111.

[7] Geyer H, Herr H. A muscle-reflex model that encodes principles of legged mechanics produces human walking dynamics and muscle activities. IEEE Trans Neural Syst Rehabil Eng. 2010 Jun;18(3):263-73.

[8] Winter DA. Kinematic and kinetic patterns in human gait: Variability and compensation effects. Hum Mov Sci 3: 51-76, 1974.

[9] Hof AL, Elzinga H, Grimmius W, Halbertsma JPK. Speed dependence of averaged EMG profiles in walking. Gait and Posture 16:78-86, 2002.

[10] Ventura JD, Klute GK, Neptune RR. The effects of prosthetic ankle dorsiflexion and energy return on below-amputee leg loading. Clin Biomech 26: 298-303, 2011. http://www.me.utexas.edu/~neptune/Papers/clinbiomech26(3).pdf

[11] van Ingen Schenau GJ, Bobbert MF, Rozendal RH. The unique action of bi-articular muscles in complex movements. J Anat. 1987 Dec;155:1-5.

[12] Winter, DA. Energy generation and absorption at the ankle and knee during fast, natural and slow cadences. Clinical Orthop Rel Res 175: 147-154, 1983.

[13] Johansson, J. L., Sherrill, D. M., Riley, P. O., Bonato, P., and Herr, H., 2005, "A Clinical Comparison of Variable-Damping and Mechanically Passive Prosthetic Knee Devices," Am. J. of Phys. Med. Rehabil., **84**(8), pp. 563-575.

[14] Orendurff, M. S., Seval, A. D., Klute, G. K., McDowell, M. L., Pecoraro, J. A., and Czerniecki, J. M., 2006. "Gait Efficiency Using the C-Leg," J. Rehabil. Res. Devel., **43**(2), pp. 239-246.

[15] Segal, A. D., Orendurff, M. S., Klute, G. K., McDowell, M. L., Pecoraro, J. A., Shofer, J., and Czerniecki, J. M., 2006, "Kinematic and Kinetic Comparisons of Transfemoral Amputee Gait using C-Leg and Mauch SNS Prosthetic Knees," J. Rehabil. Res. Devel., **43**, pp. 857-870

[16] Anderson FC, Pandy MG. Dynamic optimization of human walking. J Biomech Eng. 2001 Oct;123(5):381-390.