NMBL Application (Needs a Name!)

Requirements for SimTK 1.0

Version 1.0

Clay Anderson, Ayman Habib, Pete Loan,

Allison Arnold, May Liu, Ilse Jonkers, and Scott Delp

First DRAFT, August 2005

Abstract

Here we discuss the requirements for the NMBL application in SimTK 1.0. We outline the intended users, target functionality of the application, major tasks that need to be completed, and how the NMBL Application 1.0 will use SimTK 1.0 resources as they become available.

1 Purpose of this document

This document describes the capabilities of the NMBL Application required to meet the scientific aims of the Simbios Neuromuscular Driving Biological Problem (DBP) (ref to Crouch Grant). The purpose is to establish a common vision shared by the developers and initial users of the NMBL Application. It identifies what parts of this vision will be realized as part of the SimTK 1.0 release and lays out a path for getting there. The tentative release date of SimTK 1.0 is March 1, 2006, with functionality largely solidified by January 1, 2006 and with January and February of 2006 dedicated largely to multiplatform builds, testing, and documentation.

2 Intended users of the NMBL Application

The target users for the NMBL application are the researchers in the Neuromuscular Biomechanics Lab at Stanford (Neuromuscular Driving Biological Problem) or any comparable lab with researchers who would like to generate and analyze subject specific simulations of gait to gain insight into muscle function in normal and pathological gait. In SimTK terms, this group of users fits the description of the "modeler" user group and so will eventually be served by the SimTK Modeling Layer, which is under development [1].

2.1 Subject specific simulation pipeline:

One of the immediate software needs for the NMBL DBP is the ability to generate, validate, and investigate subject-specific simulations. The process is broken into the following steps:

1.1.1. Subject data is obtained from a collaborating clinical facility and pre-processed to have it in common format.

1.1.2. A nominal model (model of a 75 kg male) that has been validated is ***scaled*** to match the anthropometric measurements of the subject.

1.1.3. ***Inverse Kinematics*** is performed on the scaled model using the preprocessed motion capture data to compute joint angles for the scaled model that best reproduce the measured gait kinematics.

1.1.4. ***Residual elimination*** is performed to alter joint angles so that the motion of the model is dynamically consistent with the recorded ground reaction forces.

1.1.5. ***Computed Muscle Control (CMC)*** is performed on the dynamically consistent motion and ground reaction forces to compute the muscle excitations that drive the model through the measured trajectory (tracking).

1.1.6. Analyses are performed on the forward simulation resulting from CMC to quantify muscle function and what-if scenarios and to evaluate treatment options**.**

3 Current status of the NMBL software

The NMBL group currently uses a set of software tools that work together to get the research done but are not integrated into one environment and are generally not extensible to support new research needs. This set contains the following:

- SIMM: Used for initial model creation and for some plotting, visualization and/or animation. SIMM is also used to write the SD/DAST model description file and supporting utilities so that equations of motion can be generated. SIMM is currently distributed by Musculographics, written in C and runs on Windows platform.
- Utilities written by collaborators (Darryl Thelen at University of Wisconsin) to perform steps 2, 3, 4 and 5 of the pipeline in the previous section. The source for these utilities is not available (so far). The executables run on Windows and they come with parameter files to configure them as needed. This code is based on the SIMM dynamics pipeline, written in C and is specifically tied to the model currently in use.
- NMBLTS framework: A set of classes to support running and analyzing forward simulations. These set of classes are accessible through an API that can be invoked from a main program (written in C++) or from a GUI written in Java that is Swing-based. The wrapping of the API to call it from Java is done using Swig. Steps 5, 6, &7 of the pipeline are implemented in this framework.
- Helper applications: SD/FAST which serves as the dynamics engine under both SIMM and NMBLTS, Matlab to do preprocessing and some plotting, C3D to read motion capture data and export it to other formats.

We plan to use NMBLTS as the target platform to integrate the functionality needed by the group since it is object oriented, extensible, under active development and as such offers the highest potential.

4 Functionality

We intend to make available through NMBLTS, all the steps necessary to create, validate and analyze subject specific simulations from start to finish, as well as the ability to experiment with different algorithms to implement these steps or organize them differently. Not all the functionality is promised within the SimTK 1.0 time frame. We'll prioritize our efforts to provide functionality that's most difficult to implement using current tools and help move the research forward for the NMBL. The following sections provide the functionality distilled as a set of use cases and then as a set of tasks that implement these use-cases. The use-cases provide a clear mechanism for defining implementation steps, testing the functionality, and evaluating success/failure.

5 4.1 Use-cases

1. Scale model

- User loads nominal model.
- User defines a set of markers and/or dimensions to use for scaling (from file, or interactively)
- User saves scale parameters to a file.
- User scales model (scaled model visualized with marker data overlaid).
- Write scaled model to file??(This assumes that the model is saved as a file which may not be the case for a dynamic model that uses SD/FAST to implement its equations of motion).

2. Perform Inverse Kinematics

- User defines anatomical and tracking markers (from file, or interactively)
- if static trial is available, allow user to specify anatomical markers that can be used to compute static IK and adjust locations of tracking markers
- User decides whether markers, joint angles, or some weighted combination is to be used for IK's objective function used to solve for kinematics (from file, or interactively)
- Save IK parameters to a file??
- Run IK
- write IK kinematics to a file

3. Run Residual Elimination/Reduction algorithm.

- User adjusts algorithm parameters (from file, or interactively)
- save REA/RRA parameters to a file
- User runs RRA
- write RRA kinematics to a file

4. Run CMC to compute excitations that tracks:

- User adjusts parameters, such as error gains and excitation constraints (from file, or interactively)
- User runs CMC.
- User monitors progress and interrupts if something goes wrong.
- User visualizes kinematics computed from CMC, along with applied GRFs at the COP
- User plots all quantities of interest Pre/Post or save them to file (see plotting use-case).

5. Run a forward simulation.

- User loads model
- User loads muscle excitations from a file.
- User invokes forward simulation.

6. Investigate muscle excitation effects.

- User alters excitation patterns of one or more muscles
- User runs forward simulation and visualizes kinematics
- User saves edited controls to a file
- User saves output variables (kinematics, muscle forces, foot-spring forces) to file
- User plots variables of interest to screen (kinematics, muscle forces), as in plotting use-case.

7. Investigate muscle function using induced accelerations.

- Calculate induced accelerations and induced accelerations per unit force for one or more muscles (and gravity and velocity-related terms as appropriate)
- Visualize induced accelerations by specifying a body position from the simulated motion, constraining the foot, activating one muscle, and integrating forward in time.
- Save output variables (kinematics, muscle forces, foot-spring forces, IAAs) to file.

8. Investigate muscle function using perturbation analysis.

- Perform series of small muscle force perturbations as specified by user
- User configures perturbation analysis: (e.g. constrain other muscle forces to be the same or not, perturb by fixed amount or by a percent, set time intervals for perturbation and integration,and/ or tweak locations or timing of foot-springs)
- allow user to print some pre-defined check to the screen, and to exit analysis if something isn't working
- Visualize effects of perturbation (not sure if this makes sense in the general case?)
- Save output variables (kinematics, muscle forces, foot-spring forces, "influence" and "potential influence") to file
- User plots variables of interest to screen (as in plotting use-case).

9. Perform analyses using a general perturbation tool.

- perform series of perturbations in variables other than muscle force as specified by user
- user option: constrain muscle forces to be the same, or not
- user option: set perturbation size
- user option: set time intervals for perturbation and integration
- allow user to print some pre-defined check to the screen, and to exit analysis if something isn't working
- Visualize effects of perturbation (not sure if this makes sense in the general case?)
- save output variables (kinematics, muscle forces, foot-spring forces, "influence" and "potential influence") to file

10. Plotting.

- User brings up plotting dialog
- User selects a template of plots or interactively:
  - Add new plot windows to the plotting dialog.
  - Add/remove potentially multiple data sets from the plotting dialog.
  - If a user doesn't have a template or starts from a template and modifies it he/she will be prompted to save it to a template on exit.
  - Saved template contains

    - A List of plots, each plot has attributes (name, ranges, datasets, and settings)

11. Compare kinematics.

- User loads two motions (likely one from a file the other from a file or generated live).
- We make available a list of quantities representing differences.
- User selects quantities to plot from the list and plots them using the plotting use-case.
- User can choose to visually diff kinematics by displaying two instances of the model and moving them in tandem using animation like facility. User has control over (colors of the two instances, and visual offsets so that the two instances can be compared side by side).

6 4.2 Tasks

(Items in red will not be completed as part of release 1.0)

- Loading a SIMM model into NMBLTS
  - Create utility to convert SIMM models (joint and muscle files) to a NMBLTS model (XML files). Geometry (e.g., bone files) will be converted into VTK format.
  - Create classes to represent a SIMM model in NMBLTS (serialization and deserialization).
    - Geometry
    - Bodies
    - Joints
    - Coordinates
    - Muscles
    - Wrap Objects
    - Deform Objects
    - Constraint Objects
    - Contact Objects, Force Mats, and Contact Detection
  - Port the existing SIMM Kinematics Engine into NMBLTS
    - Constraints
    - Closed loops
    - Moment arms
  - Dynamics with SDFast
    - Generate an SDFast system description file
    - Implement required SDFast user-supplied methods in NMBLTS (sduforce, sduperr, …)
    - Compile
  - Dynamics with Simbody. When Simbody becomes available, it will replace the SIMM Kinematics Engine and SDFast.

- Scaling a model. Develop methods for scaling the bodies, joints, constraints, and muscles of a model to an individual subject based on experimental kinematics and anthropometric data.
- Inverse Kinematics (IK). Implement methods for solving for the joint angles of a model such that an error criterion that combines a weighted sum of the errors in measured joint angles and marker positions is minimized.
- Residual Reduction Algorithm (RRA). Implement an algorithm for altering kinematics' data for a model so that the kinematics of the model are dynamically consistent with measured external reaction force data.
- Computed Muscle Control.
- GUI
- Plotting
- Visualize contact elements, GRF, COP
- Load and compare kinematics.
- Setup and run CMC
- Setup and run RRA.

7 Development, Build, and Testing

CMake [ref?] will by used as the build utility for NMBLTS 1.0. Doing so will make it possible to use Microsoft Visual Studio as the primary development environment for the Windows machines while at the same time allowing NMBLTS to be compiled for Mac and Linux systems. In addition, using CMake will allow integration with the SimTK Dashboard system (now under development). Several prototypeTesting will be carried out using the CMake and Dashboard system that is currently under development for SimTK.org.

8 Documentation and Tutorials

Documentation will included a hyperlinked API reference manual generated by Doxygen. In addition, a user guide with at least a few basic tutorials will be developed.

9 Installation

We plan to use Java-webstart to install the application. An installer is already in place for the GUI and the required native libraries under http://dev.simtk.org/ahabib/nmblapp.jnlp.

10 What will be accomplished in NMBLTS 1.0

We plan to implement use-cases 1-5, 8 and 10 above for the intended user group in the SimTK 1.0 timeframe. The remaining tasks are available programmatically through an API so there's little urgency in trying to get these into the application. The order of completion of these tasks, however, is dictated by the needs of the research grants of the NMBL DBP. We also plan to use the build-system and the installation utilities as needed to make the software available and accessible to researchers; however the extent to which these are adopted depends on the resources. Also depending on research needs, the functionality may or may-not be delivered in the GUI in NMBLTS 1.0 timeframe, in favor of using a command line utility version.

11 What are we leaving out of NMBLTS 1.0?

GUI functionality that is accessible through programmatic interface (API) will be deferred in favor of actual new functionality needed by the NMBL DBP. In addition, the implementation of the Kinematics Engine will miss on some object types/abstractions that exist today in SIMM but that are not needed for the subject specific workflow.

12 Use of SimTK Infrastructure and Numerical Components

Future releases of the NMBL Application will fully embrace the simulation framework provided by the SimTK modeling layer [ref], in addition to the component-based numerical libraries offered through SimTK. Release 1.0 of the NMBL Application well seek to align itself with SimTK by using the CMake build system, dashboard testing system offered by SimTK.org, and Java Web Start for installation. If possible, NMBL Application 1.0 may also use a small number the numerical libraries that become available through SimTK.org. These may include root solvers, optimizers, and the CVODE integrator. In some instances, through the NMBL Application development effort, a small number of numerical libraries might be contributed to SimTK.org.

13 Risks and unresolved issues

- Will the top-10 potential collaborators happen during the SimTK 1.0 time frame?
- SimTK modeling layer interactions.
- Simbody?
- Papers/conferences/grants and availability of Clay, SimTK 1.0 modeling layer and availability of Ayman, SIMM and availability of Pete.

14 Acknowledgments

This work was funded by the National Institutes of Health through the NIH Roadmap for Medical Research,[1] Grant U54 GM072970. Information on the National Centers for Biomedical Computing can be obtained from http://nihroadmap.nih.gov/bioinformatics.

15 References

1. Crouch Grant
2. The SimTK Modeling Layer
3. The SimTK Multibody Dynamics Toolset.

[1] Information on the National Centers for Biomedical Computing can be obtained from http://nihroadmap.nih.gov/bioinformatics.