VICON
MOTION SYSTEMS

Revision history

Created July 1995
Development version of VICON Editor and Modeller software (16 bit)
First update August 1995
Version 1.0 of VICON Editor and Modeller (16 bit) - limited release
Second update February 1996
Production version 2.0 (32 bit), renamed BodyBuilder
Third update April 1996
Production version 2.0 (32 bit), BodyLanguage manual separately bound
Fourth update February 1997
Production version 3.0, BodyLanguage manual re-integrated
Fifth update November 1997
Production version 3.5 Biomechanics edition

Intended Audience

This manual is intended for operators of VICON motion systems who are familiar with Windows NT or Windows 2000 operating software, and with VICON Workstation application software. Manuals for these products are supplied separately

Windows 2000	Microsoft Corp.
Windows NT	Microsoft Corp.
i486, Pentium	Intel Corp.
IBM	International Business Machines Inc.
Acclaim	Acclaim Technologies Inc.
Power Animator	Alias Inc
Kinemation	Wavefront Inc
SoftImage 3D	SoftImage Inc
N-World	Nichimen Graphics
Lightwave 3D	NewTek
Prisms	Side Effects Inc.
VICON Workstation	Oxford Metrics Ltd
BodyBuilder	Oxford Metrics Ltd
BodyLanguage	Oxford Metrics Ltd
DynaCal	Oxford Metrics Ltd

Trademarks

The following are registered trademarks or trademarks with registration pending:

Company Information

Oxford Metrics Ltd 14 Minns Estate West Way Oxford OX2 0JB
tel. +44 1865 261800 fax +44 1865 240527
http://www.vicon.com Manual author: support@vicon.com

INTRODUCTION 09
Overview 09
Licensing 09
Computer Requirements 10
Related Documentation and Technical Support 10
How to Use this Manual 10
Software Installation 11

USING BODYBUILDER 12
The Working Environment 12
Manipulating Data 12
Writing and Running Models 12
Work Flow 13
    Static Trials 14

WORKSPACE WINDOW 15
Opening the Workspace Window 15
Selecting Trajectories 15
Display of Trajectories and Sticks 16
Workspace Definition 17
Rotating the Viewpoint 17
Zooming the Viewpoint 18
Moving the centre of rotation and zoom in the viewplane 19
Changing the centre of rotation and zoom to a
selected point on a trajectory 19
Alternative Display Modes 20
Labels List 20
    Label Mode 21
Replay Control Bar 22

GRAPH EDITOR WINDOW 23
Graph Editor 23
Controlling the Display 23
    Changing the Time Scale 24
    Changing the Vertical Scale 25
    Moving the display laterally without changing scale 25
    Moving the display vertically without changing scale 25
Selecting a Field Range 25
Point Dragging 27

TEXT EDITOR WINDOW 28
Text Editing 28
Opening a Text Editing Window 28
Editing Text Files and Testing Changes 29

FILE MENU COMMANDS	30
File Menu	30
Open	30
Merge	30
Close	31
Save	31
Range to Save	32
Save As	32
Write C3D	33
Axis Changing	34
Write ASCII	35
Options	36
Previous Files	37
Preferences	37
EDIT MENU COMMANDS	39
Edit Menu	39
Undo and Redo	39
Snip	39
Delete	40
Add Point	41
Create New Trajectory	43
Delete Trajectory	44
Delete Model Outputs	44
Defragment Trajectories	44
Delete and Fill	44
Fill Gaps In...	44
Fill All Gaps	45
Maximum Fill Gap	45
Copy Pattern	46
Move Point Mode	47
Simple	47
Weighted A mode	48
Weighted B mode	50
Expand/Reduce mode	50
Filter	51
Filter Characteristics	52
Spike Removal	53
Re-Sampling	53
VIEW MENU COMMANDS	55
View Menu	55
Toolbar	55
Status Bar	55
Rotation Centre	55
Markers	55
Stick Figure	56

Floor 56
Flicker Free 56
Always Refresh 56
Set Frame Rate.. 56

MODEL MENU COMMANDS 57
Model Menu 57
Run Model 57
Run Model for Subject 57
Create ASF 58
Create AMC 58
Create AMC Files... 58
Axis Permutation 59
View Model 60
View Parameters 60
View Last Error 60
Adjust Parameters 60
Subject Settings 60

INTERACTIVE PARAMETER ADJUSTMENT 62
Parameters 62
Starting Interactive Adjustment 62
Parameter List 63
Adjusting Values 63
Finishing IPA 64

WORKSPACE MENU COMMANDS 65
Workspace Menu 65
Select Centre 65
Hide Trajectories 65
Show Trajectories 65
Hide Other Trajectories 65
Hide All Trajectories 65
Show All Trajectories 65
Toggle All Trajectories 65
Toggle Before Display 66
Toggle After Display 66

DESIGNING BIOMECHANICAL MODELS 67
Defining the Problem 67
Dividing the Body into Segments 67
    Approximations and Errors 69
    Modelling the Joints 70
Points, Lines, Planes and Segments 70
    Single Point 71

Two Points	71
Three Points	72
Four Points	72
Choice of Origin and Axes	73
Rotations	74
BODYLANGUAGE	75
Overview	75
Kinematic Modelling with BodyLanguage	75
Before Starting	75
Writing the Model Script	75
Verifying Model Scripts	76
Using Kinetic Data: Terms and Definitions	76
Force	76
Moment	77
Reaction	77
Work and Energy	77
Power	77
Moment of Inertia	78
Applying the Model	78
Displaying the Results	78
OBJECTS	80
Types of Object	80
Numbers	80
Points	80
Segments	80
Rotations	81
Reactions	81
Object Names	81
Limitations of Object Names	81
Reserved Names	82
Object Assignment and Type Definition	83
Inputs	83
Optional Points	84
Outputs	84
Parameters	84
Samples	85
EXPRESSIONS	86
Types of Expression	86
Numbers	86
Points	86
Segments	86
Segment Definition	87
Segment Definition across Hinge Joints	89

VICON
MOTION SYSTEMS

Singularities and Dead Band	89
Anti-Flip	90
Hierarchical Skeletons	93
Rotations of Segments	94
Rotation Definition	94
Helical Vectors	95
Global Rotations	96
Rotations about Fixed or Floating Axes	96
Euler Angles	96
Gimbal Lock	102
KINETIC MODELLING	103
Mass and Inertia Properties of Segments	103
Direct Definition of Mass Properties	103
Definition using a Table	103
Correcting for the Mass of a Prosthesis	104
Reactions, Forces and Moments	105
Force	105
Moment	105
Reaction	105
Force Plate Data	106
Segment (free body) Equations of Motion	107
Connection of other reactions	107
Gravity	108
Linear Acceleration and Angular Velocity and Acceleration	108
Intersegmental Power Flow	108
Units	108
OPERATORS	110
Types of Operator	110
Number Operations	110
Point Operations	111
Segment Operations	112
Rotation Operations (none permitted)	112
Selection Operator	112
FUNCTIONS	113
Types of Function	113
Number Functions	113
Point Functions	113
Segment Functions	115
Rotation Functions	115
Special Functions	115
Spine	116
Length and Stiffness	117
Multiple Segments/Vertebrae Mode	117

Shorthands and Tokens 118
Non-Assigning Statements 118
Conditionality Tests 118
    Dealing with Marker Occlusions 119

MACROS 120
Uses of Macros 120
Text Concatenation Operator 120

OTHER EXPRESSIONS 122
Comments 122
Error Codes 122

EXAMPLE MODEL - FOOT 123
Anatomical Structure 123
    Axis Definitions 123
    Handedness 123
Model Script 123
    Comment Section 123
    Macro "Substitute4" 124
    Optional Points 126
    Segment Definitions 126
    Angle Output Section 132
Left Foot Script 132

EXAMPLE MODEL - KINETIC LEG MODEL 133
Kinematic Principles and Limitations 133
Force Vectors 133
Segment Definitions 134
Definition and Output of Rotations 137

GLOSSARY OF TERMS 140

INDEX 143

Overview

BodyBuilder for Biomechanics is a software package which enables the user to:

- Manipulate data produced by VICON Workstation
- Edit and modify trajectories
- Interpolate broken trajectories
- Create kinematic and kinetic models
- Choose any marker set
- Model body segments and joints
- Calculate angles and moments
- View results on screen
- Output results to file
- Display results in a flexible format
- Create text file output
- Define and save output formats

BodyBuilder for Biomechanics is a powerful set of tools for investigating human movement. Models and outputs can be as simple or complex as required.

The BodyBuilder for Biomechanics package contains two program applications: BodyBuilder and Reporter. BodyBuilder is the data manipulation and modelling element, and Reporter is the flexible graphic output element.

Licensing

It is intended that BodyBuilder for Biomechanics should be installed on the PC Workstation which forms part of each VICON motion capture system. BodyBuilder for Biomechanics is hardware-protected, meaning that you require a dongle (hardware key), issued by Vicon Motion Systems Ltd, plugged into your computer's parallel port, in order to run the program.

Licensed users may wish to run additional copies of BodyBuilder for Biomechanics on other computers. This will require additional dongles, even if those computers are part of the same local area network. For additional dongles, refer to Vicon Motion Systems Ltd or local agent.

Computer Requirements

To run BodyBuilder for Biomechanics, you need a PC with a 486 (minimum) or Pentium (recommended) processor. The better your processor and the faster your clock speed, the faster the program will run. A substantial video memory is advisable, with PCI bus 800¥600 accelerated graphics, and at least 16MB RAM. A large hard disk (recommended minimum 200MB, ideally 1GB)is essential; and a backup device such as a ZIP or DAT drive is strongly recommended. A 32-bit operating system, either Windows NT (recommended) or Windows 95 or 98, is required.

When large .C3D files are opened or generated, the computer may require a very large swap file. This will cause slow running if your hard disk does not have a significant amount of free space (as much as 60MB may be needed). If you are using Windows 95, this may give the appearance of hanging up; Windows NT behaves better in this respect, and is the recommended operating system. Windows NT comes in two versions, Server and Workstation. The Server version, as the name implies, is intended for computers which are the hub of a network, controlling shared resources, while the Workstation version is for computers used for running applications such as BodyBuilder for Biomechanics. Standalone computers should use the Workstation version.

Related Documentation and Technical Support

BodyBuilder for Biomechanics is designed to process data created by the VICON three-dimensional motion capture systems from Vicon Motion Systems. Manuals for these products are supplied separately. Further copies of these manuals, technical advice, software and documentation updates, service and support are available from Vicon Motion Systems, both directly and through their agents and representatives, for a period of one year from the date of the original purchase. After this period, an extended service contract is available in most areas - please refer to Vicon Motion Systems or your local agent.

BodyBuilder for Biomechanics is supplied with online help, the text of which is similar to the manual. Example model script files and data are also supplied with the program.

How to Use this Manual

- This manual contains all the information which you need to use, or learn to use, BodyBuilder for Biomechanics.

- Some chapters are largely explanatory and are recommended for new users.

- Some chapters are largely reference material, describing features, functions and commands in detail.

If you are a new user, you should read this chapter, which contains general information about the software, and then install the software if that has not already been done. You should also read the next chapter, Using BodyBuilder, which is a brief introduction to the BodyBuilder program. The following ten chapters contain detailed reference material relating to BodyBuilder. Keep this section to hand when you start using the program and need to look up information on specific points. The following two chapters, about designing biomechanical models using BodyLanguage, provide a general discussion of the principles of modelling, and should be read by all users. The following seven chapters contain detailed reference material about BodyLanguage.

It is possible to use BodyBuilder for Biomechanics without learning the details of BodyLanguage, or writing models, because example models are supplied. However, the flexibility of being able to write or modify models is the most important feature for many users. Users who do not want to develop their own models may benefit from an understanding of the models they use, so a look at the section on BodyLanguage is recommended. Commented example models are included, and if you are using one of these models, it is a good idea to read through the comments and to gain an idea of what the model does.

If you are upgrading from another version and are already familiar with BodyBuilder and Reporter, you should look at the sections describing the new features (kinetic extensions to BodyLanguage, interactive parameter adjustment, and multiple subject trials).

Software Installation

For all VICON-related software, the top level directory should be

C:\VICON

where C: is the local hard disk drive. Each application program should have a subdirectory in the above top level directory, so BodyBuilder and Reporter will be installed in the directories

C:\VICON\BODYBLD and C:\VICON\REPORTER

respectively. These directories will be created by the installation utility, unless the user chooses otherwise. Certain additional subdirectories will be created for the associated files and example data shipped with the application programs.

The installation utility provides its own instructions, but if any problem should arise with this process, please contact Vicon Motion Systems or your local agent.

The Working Environment

The term "BodyBuilder" refers to the application program which carries out the functions of editing and modifying the original data; applying models; providing facilities for model editing; and displaying 3-dimensional data through the Workspace display window. This manual is concerned with BodyBuilder; Reporter is documented separately.

In common with most Windows applications, when BodyBuilder is launched, a blank window opens with a limited number of menus and commands available. Once a data file is opened, however, more menus and commands become available. A certain number of other files are required in addition to the 3-dimensional data, including model scripts and their parameters, and marker labels with stick-figure links.

Manipulating Data

BodyBuilder can be used for data editing without modelling. The editing functions include filtering, interpolation, trajectory snipping and deleting, trajectory copying and moving, re-sampling of trajectories to a different sample frequency, and relabelling. These functions alone provide a significant range of tools which may be used to improve data which is imperfect (for example, interpolating short gaps, filtering of high-frequency noise, and rectifying errors in the reconstruction and initial labelling). These functions may be regarded as tools for cleaning up previously unsatisfactory data. Often, these functions are used before going on to modelling and data presentation.

The editing functions also allow the user to modify the data by moving points, creating new trajectories, and otherwise making substantial changes to the original data so that the output no longer represents what was measured in certain respects. These functions allow for experiments and for testing of different cases based on a limited initial amount of data. However, the user should be aware of the degree to which the data has departed from the reality of what was actually measured, particularly if the results are to be interpreted as an accurate and objective study of human movement. Even excessive filtering or interpolation can lead to important departures from objectivity in certain cases, and the user must develop a clear sense of the boundary between making minor improvements to the data, and significantly changing the original measurements.

The commands used to open files, control the display of data, and to use the editing functions, are listed in the File, View, Workspace and Edit menus.

Writing and Running Models

The model script is a separate file with the extension .MOD, and it is normally stored in the directory

C:\VICON\MODELS

The script to be used must first be selected (loaded). When BodyBuilder is first launched, no script is loaded; a selection must be made before the modelling-related commands become active. Only one script may be open at any time.

A text editing window is opened when a model script is loaded, allowing changes to be made. Changes take immediate effect, and do not have to be saved before they can be run. A parameter file is used in all but the simplest models, which contains values that may change frequently (eg. the physical measurements of the subject) and are stored this way so that they can be modified without editing the model script itself.

As with the model script, it is necessary to select and open a data file on which to operate; no data file will be opened automatically.

If a data file is open, and a model script loaded, and assuming that they are compatible, the command Run Model may be used to apply the model to the data. BodyBuilder then goes through the data frame by frame, calculating all the new points and other variables defined in the model script. The results are then displayed in the Workspace window, and may be saved (until saved, the newly calculated points are held only in volatile system memory; the original data file on disk is not modified unless the user chooses to overwrite it).

It is possible to make changes to a model script and immediately apply the changed script to some data to see the effect. This allows rapid model development and testing. It is also possible to make changes to the values of parameters and to see the effects of these changes immediately in the workspace, using interactive parameter adjustment.

When a model script has been fully developed, the text editor window need not be displayed; the model can simply be loaded and run without change.

Work Flow

The flexibility of BodyBuilder for Biomechanics allows the following order to be changed according to needs, however this is the normal flow of work when processing data using an existing model:

- Decide the purpose of the measurements, select model and marker set
- Measure physical parameters (if appropriate) of subject, and attach markers
- Collect data using VICON Workstation
- Reconstruct and identify the trajectories, using labels corresponding to those specified in the model script
- Produce C3D file
- Start BodyBuilder and open C3D file

- Use editing tools to clean up the data if necessary
- Load model and check parameters
- Run model and examine results
- Select suitable output format and save
- Start Reporter, load output data and print results

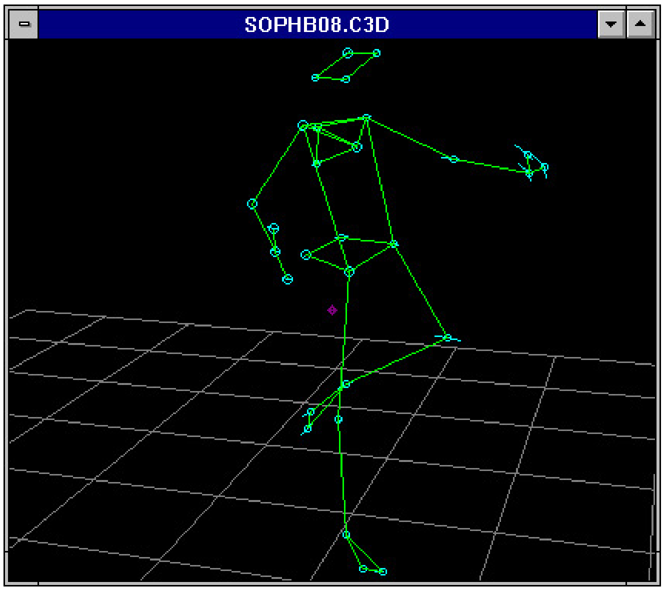
Static Trials

A common variation on the above procedure is the inclusion of a static trial. This is a short data capture during which the subject remains still (or nearly so) and is for the purpose of measuring certain parameters. Users who need to produce Acclaim Skeleton Files use static captures for this purpose. Static trials can also be used together with conditional parts of a model script, to measure certain quantities and write them to the parameter file, so that they are available when the dynamic data is processed. In some cases, variations on the marker set are used for the static trial. An example is the extra heel markers used during the static trial required by VICON Clinical Manager (VCM) and Plug In Gait gait analysis software and Plug In Gait. In that case, the purpose of the static trial is to allow the dynamic gait analysis to take place using the minimum possible marker set.

In VICON Workstation software, it is possible to label data semi-automatically, having first manually labelled a static trial. The C3D (trial data) files produced in this way retain information about the subject name and the marker set used during auto-labelling, and whether the trial was a static trial for auto-labelling purposes.

Opening the Workspace Window

BodyBuilder uses a VICON Workstation Workspace window to display the contents of a .C3D file, to facilitate data editing, and to display the results of kinematic modelling. To open a Workspace window, select File | Open, then select the desired .C3D file. More than one Workspace window may be open simultaneously, but only one may be active at any time. The active window has a highlighted title bar. To make a different window active, place the pointer within it and click the left mouse button. Trajectories in the active window may be selected and edited.



Workstation Workspace windows use true perspective projection, rather than isometric or planar projection. This enhances the realism and usability of the display.

The Workspace window in BodyBuilder does not display objects defined in .WKS or .OBJ files. For clarity, the floor tiling is automatically sized to show the area covered by the motion data.

Selecting Trajectories

Up to 24 trajectories may be selected at a time. Current selections are lost when the current trial is closed.

Most editing features operate on selected trajectories. To select a trajectory, move the pointer close to the chosen trajectory and click the left mouse button. The selected trajectory changes from blue to yellow. Subsequent selections change to red, purple, and dark blue, with this colour cycle repeating for further selections. You cannot select a trajectory until the mouse helper appears - this may take a few moments.

To de-select a trajectory, click the right mouse button (with the pointer anywhere in the window). If several trajectories have been selected, they are de-selected in reverse order.

By default, the Mouse Helper is active. To disable it, use the View | Mouse Helper command.

There is an aid, called the Mouse Helper, to assist selection. When the tip of the pointer arrow is close enough to a trajectory to allow that trajectory to be selected, a box appears on the pointer containing the label and the field number of the point closest to the mouse pointer. If the trajectory is unlabelled, the box contains a dash.

Display of Trajectories and Sticks

3D data can be displayed as

- markers at 3D points in the current field
- sticks connecting 3D points in the current field
- trajectories connecting points before and/or after the current field

Use the View|Markers command to turn the marker display on and off.

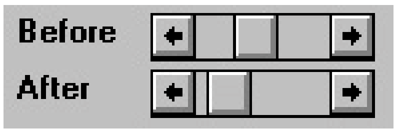
If the user needs to label (or re-label) a trajectory or segment of a trajectory, a .MKR file containing the desired label set must be made available. Use the Model | Subject Settings... command to select a .MKR file with the required label names and stick connections between labelled points. The initial .MKR file available when BodyBuilder is started, is the one selected in the application preferences. This may not be appropriate to the file you wish to edit. When editing a .MKR file, take care to enter the correct number of labels and stick connections.

Some marker set files may contain several different display sets (that is, different sets of sticks). These may be selected from the drop down list above the list of markers at any time.

Stick connections may be defined which connect the points which result from the reconstruction of real marker positions by VICON Workstation, and also the virtual points created by a kinematic model. Sticks are displayed as green lines.

Unlabelled trajectories and points are displayed in white. Labelled trajectories and points are displayed in cyan. Selected trajectories are displayed in a colour which depends on the order of selection. By default, the length of trajectory displayed is one field either side of the current field. The length displayed may be controlled by the user.

Initially, the trajectory length displayed is one field either side of the current field.



Use the Before and After scroll bars in the lower right corner of the main window to control the length of the trajectory display before and after the current field.

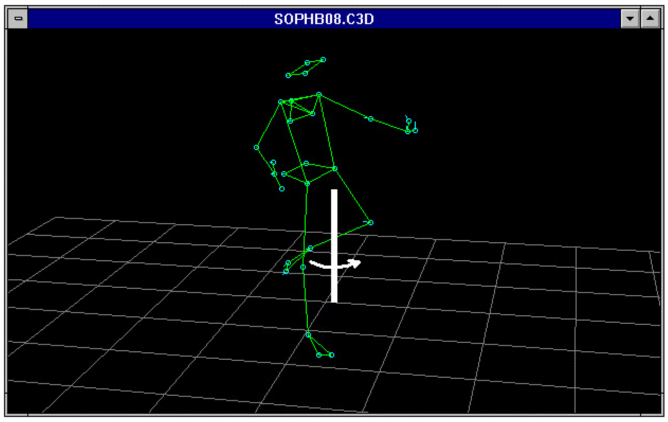
If the Before and After slide bars are moved all the way to the left, no trajectories are displayed in any Workspace window. The markers on the current field may still be displayed, according to the setting of the View | Markers command. In this state, trajectory selection is not possible in the Workspace window.

Workspace Definition

Workspaces are built on an outlined and tiled floor that represents the area or "workspace" where the kinematic data was collected. The size and shape of the outline and tiles are automatically set and are not user-selectable, as they are in VICON Workstation. This simplification has been made in order to avoid a cluttered appearance in the window, and because BodyBuilder is not concerned with kinetic (force) data.

Rotating the Viewpoint

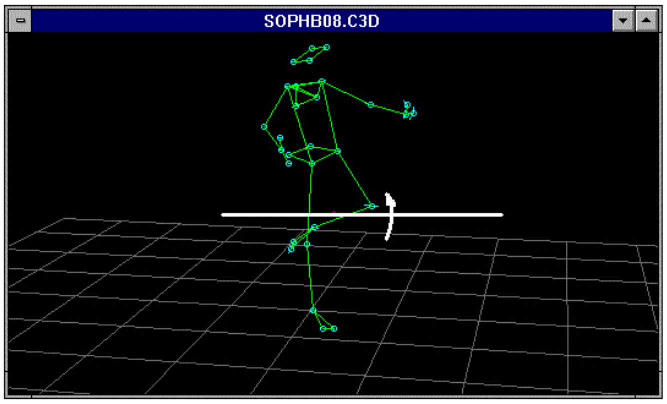
The Workspace is a 3-dimensional, perspective view of the experimental volume. The viewpoint may be changed by rotating the workspace and by moving towards or away from the rotation centre of the workspace to widen or narrow (zoom) the field of view.



It is easiest to select the best point of view after stopping the replay of moving points and sticks.

If the replay is running, you can stop it by pressing the spacebar.

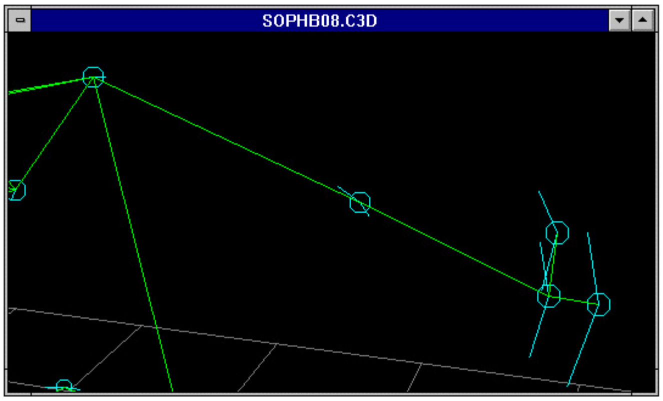
Rotate about a vertical axis through the current centre of the image by positioning the pointer anywhere within the dark area of the Workspace window, holding down the left mouse button, and moving the mouse left or right. The workspace rotates in the direction of mouse movement until the pointer runs off the edge of the screen.



Rotate about a horizontal axis by holding down the left mouse button and moving the mouse up and down. The workspace rotates in the direction of mouse movement. The viewpoint can move through a range which takes it from directly underneath (yellow floor grid) to directly overhead (white grid). In VICON Workstation software, the viewpoint cannot be moved below floor level or over the top of the workspace, since this would turn the view upside down. In BodyBuilder, this potentially confusing point of view is permitted, but should be used with caution.

Horizontal and vertical rotations can be combined in a single mouse movement.

Zooming the Viewpoint



Zoom by positioning the pointer anywhere within the dark area of the Workspace window, hold down the right mouse button, and move the pointer up to zoom-in or down to zoom-out. If the mouse pointer runs off the edge of the screen, release the button, move the pointer back into the screen, and repeat the zoom action. There is no theoretical limit to the range of zoom that can be applied.

Rotations and zooms cannot be made simultaneously.

Moving the centre of rotation and zoom in the viewplane

The current centre of rotation is displayed by default as a small purple diamond. To remove it, use the View | Rotation Centre command.

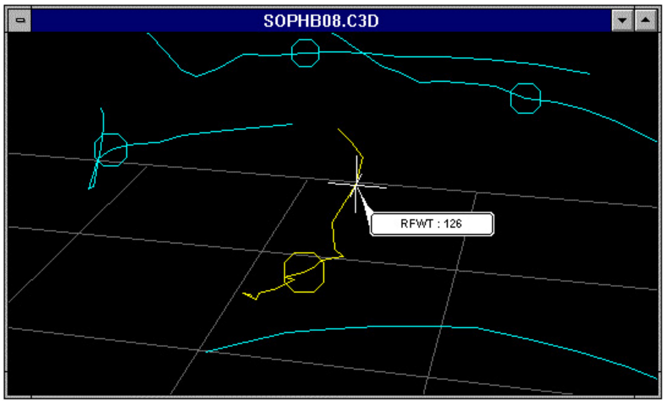
The Workspace rotates around a position called the "centre of rotation". Zoom operates toward or away from it. When a Workspace window is first opened, this is located at the origin of the kinematic coordinate system.

The centre can be moved in the current plane of view. With the pointer anywhere within the dark area of the Workspace window, hold down the Control key and the left mouse button, and move the pointer horizontally and/or vertically. The Workspace view moves in the same direction, creating a new centre of rotation and zoom. Alternatively, you can press both mouse buttons together and move the mouse, to move the Rotation Centre.

A combination of rotation and viewplane movements allows the centre of rotation and zoom to be placed anywhere within the Workspace.

Changing the centre of rotation and zoom to a selected point on a trajectory

The centre of rotation and zoom can be changed to any point on a selected trajectory.



The selected trajectory remains selected after centring. The chosen point on a selected trajectory may not correspond to the current field. In the above example, the selected trajectory is shown yellow; the chosen point for centring is marked by a white cross. You cannot select a trajectory until the mouse helper appears - this may take a few moments depending on your PC.

Shortcut: C key

Hold the pointer steadily at the chosen point and click with the left mouse button to select this point. If the trajectory segment is the only one selected, it changes colour to yellow. If other trajectories are already selected, another colour is used. Use the Workspace | Select Centre command to make the selected point the new rotation centre.

Alternative Display Modes

BodyBuilder can run on a wide range of computers. Some computers have fast processors, some have fast graphics hardware, and others use fast data transfer buses. The optimal display mode setting for Workspace windows depends on the relative performance of these variable resources.

Change the plotting mode by using the View | Flicker Free and View | Always Refresh commands.

Labels List

The .MKR file can be edited using any text editing utility, and is normally found in the \MODELS directory. If the Windows NOTEPAD editor is used, remember to Save any changes, and then update the displayed list by using the Edit | Attach Marker Set... command. It is not necessary to close the NOTEPAD editor between modifications as it does not "lock" the file.

On the right hand side of the main program window, there is space for a list of labels. When a .C3D file is opened, if it was autolabelled, the marker set used for labelling will be loaded. For a manually labelled C3D file, if it is located in a session directory, BodyBuilder reads the SES file and locates the corresponding MKR file.

If there is no such .MKR file, BodyBuilder displays the last label list used in previous work. If this list is unavailable, the label list is blank.

If multiple .C3D files are open simultaneously, the list corresponding to the currently active Workspace window is displayed.

If all Workspace and Graph Editor windows are closed, the list box is blank. If a Text Editor window is active, the list box turns blank temporarily.

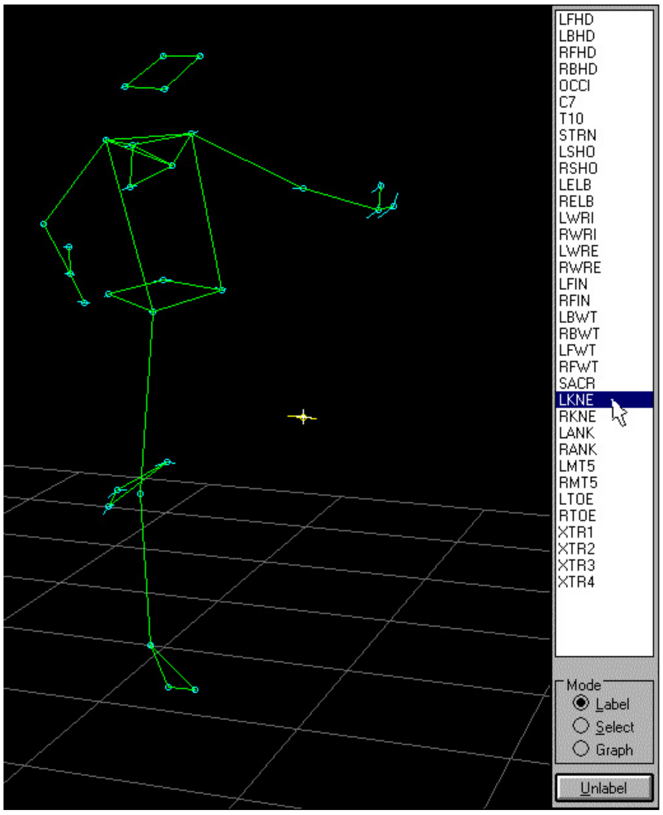
If the list of labels is too long to be displayed in the box, a scroll bar is displayed at the right allowing you to scroll up or down to see further labels.

If you select a different (or edited) .MKR file, the new list appears and is available for use immediately. Different .MKR files may be selected using the Model | Subject Settings command.

The label box has three modes: Label, Select and Graph, chosen by clicking on the relevant button at the bottom of the list.

When in Label mode, you can select a trajectory and click on the appropriate label name to apply that label to the selected trajectory. When in Select mode, you can click on a label name to select the corresponding trajectory. When in Graph mode, you can click on a label name to open a Graph Editor window for that trajectory (or change the current Graph Editor window to the new trajectory).

Label Mode



In BodyBuilder, it is unlikely that you will find it convenient to unlabel and relabel more than one trajectory at a time.

If you want to change the label of a segment, select and unlabel that segment. It will then be displayed in white, and can be given a new label.

To label trajectories, ensure you are in Label mode, then select between one and twenty-four unlabelled trajectories by clicking on each in turn, using the left mouse button. Selected trajectories are highlighted cyclically in yellow, red, purple, and blue. Next, click on the required label for each trajectory, in the same order as the selection. Each trajectory changes to cyan as it is labelled.

If a trajectory is broken, the same label should be applied to each segment.

Relabelling

The label of a trajectory or segment can be changed directly by selecting it and clicking on the revised label in the Labels List.

Select Mode

In Select mode, click on a label in the list to select and highlight the trajectory or segments with that label. Up to 24 trajectories can be selected in this manner.

VICON
MOTION SYSTEMS

De-Selection

De-select trajectories by clicking the right mouse button while pointing anywhere within the Workspace window.

Unlabelling

Use the Unlabel button to remove a label from a selected trajectory. Up to 24 trajectories may be selected and unlabelled in sequence.

Replay Control Bar

Use the Replay Control Bar at the bottom of the screen to control the replay of data.

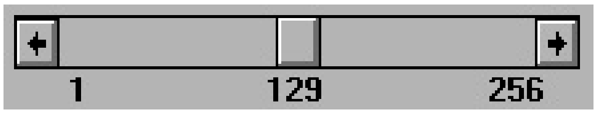
Keyboard shortcuts:
Left: Control-left arrow
Right: Control-right arrow

Use the large Left ◀ and Right ▶ Play buttons on the left of the bar to play (change the current field) Forward and Backward, respectively. Use the View | Real-Time Playback command to control the speed of replay.

Shortcut: Space bar

Use the large ■ Stop button (between Play buttons), to stop the replay.

The numbers under the replay slide bar indicate the first, current, and last fields.



When dragging, try to avoid moving the mouse pointer off the slide bar as the slider returns to its starting position.

Drag the slider on the central field number bar to move to an individual video field. Single-step by clicking on the small arrow buttons at either end of the slide bar.

Graph Editor

The Graph Editor window provides a range of tools allowing the user to alter trajectories, move points, and to create new points and trajectories. The Graph Editor window contains three graphs, which display the X, Y and Z components of the selected trajectory, against time, and makes a variety of tools available for manipulating the data.

In order to use these tools, you must have a .C3D file open. Although it is possible to close the Workspace window and still use the Graph Editor window to operate on the .C3D file, the two windows are designed to be viewed together, vertically tiled within the main program window. There are three methods of opening a Graph Editor window:

- Select the trajectory you want to alter in the active Workspace window. Use the Window | New Graph Editor command.

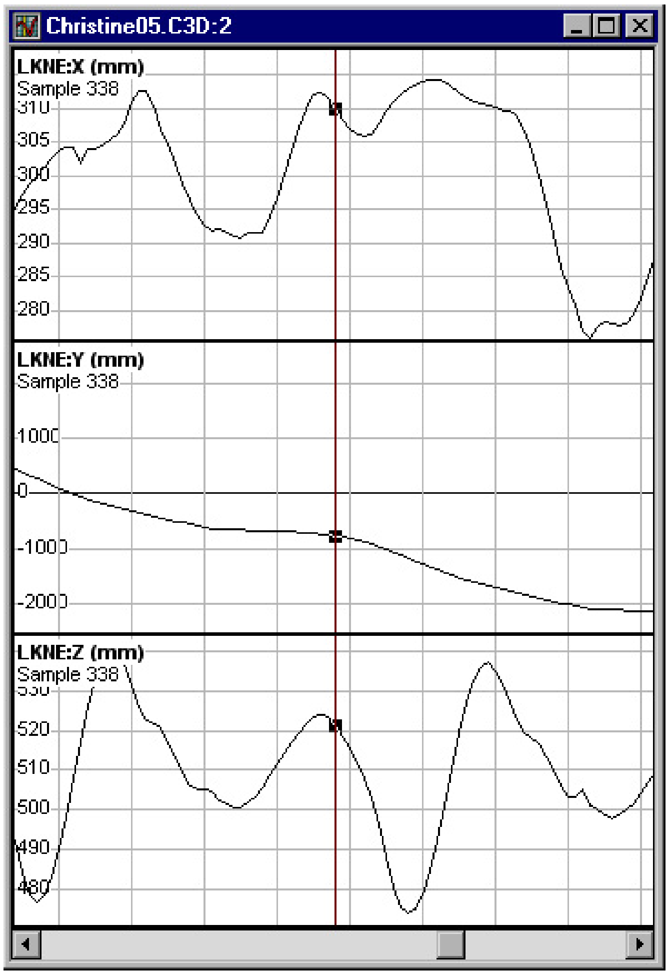
- Select Graph mode at the bottom of the labels list, and then click on the label corresponding to the trajectory you want to alter. A new Graph Editor window opens, displaying the components of the selected trajectory.

- Use the Edit | Create New Trajectory command. A new Graph Editor window opens, with no trajectory or label displayed.

The second method can also be used to change the trajectory displayed in an existing Graph Editor window. If a Graph Editor window is active (with a highlighted title bar), you can use the second method to select an alternative trajectory, which will replace the one currently displayed in the active Graph Editor window. Alternatively, it is also possible to open additional Graph Editor windows, using the Window | New Graph Editor command, but multiple Graph Editor windows which are tiled will be too small to be useful. It is recommended that only one Graph Editor window is used at one time. Any operation which deletes the trajectory displayed in a Graph Editor window will close that window.

Controlling the Display

When a Graph Editor window is opened, the vertical (X, Y or Z component) and horizontal (time) scales are automatically set, with the current field in the centre of the horizontal scale. The current field is indicated by a vertical red cursor, and the value of the component shown by a line, which is interrupted if there is a gap between segments of the same trajectory.



The vertical red cursor indicates the current field, and in each graph the current point is highlighted by a black square.

The top left corner of each graph shows the label of the trajectory (unless it is a newly-created and unlabelled trajectory), the name of the component (X, Y or Z), the units of the vertical scale marks and the current field number.

If the current field number is changed, either by using the replay buttons, the replay slider bar, or by clicking the left mouse button at a different left-right position within the Graph Editor window, the red cursor will move to the appropriate position. Any change in the current field, by any means, will cause the replay slider bar, the Workspace window, and the Graph Editor window, to be updated together.

Changing the Time Scale

Keyboard alternative: G

To change the time scale, position the pointer within the Graph Editor window and press the right mouse button. Slide the pointer to the right to expand the time scale. There is a limit on this expansion to prevent the horizontal space between markers becoming too large. Slide the pointer to the left to contract the time scale. A limit is reached when the display scale is one pixel per field.

The horizontal time scale is the same on all three graphs within a single Graph Editor window whichever component the pointer is positioned over when the change is made. The red cursor remains static while the time scale is changed.

Changing the Vertical Scale

To change the vertical scale, position the pointer within the area displaying the component whose scale you want to change, and press the right mouse button. Slide the pointer up to expand the distance scale, and down to contract it. The change will only affect the component displayed in the section where the pointer was located when the right mouse button was pressed.

When the right mouse button is held down, the up-down and left-right pointer movements may be combined, changing the time scale of all three components and the distance scale of one simultaneously. New users may find it simpler to make one adjustment at a time.

Moving the display laterally without changing scale

If the replay slide bar is used to change the current field outside the range displayed in the Graph Editor window, the graph display will move laterally so that the current field is kept in view.

The keyboard alternative to holding down both mouse buttons is F. The scroll bar is the easiest method to move the display laterally.

It is also possible to move the graph display laterally, either using the scroll bar at the bottom of the window, or by positioning the pointer within the Graph Editor window, holding down both mouse buttons, and moving the pointer left or right. This has the effect of dragging the display. It is possible to continue dragging over the border of the Graph Editor window, thus taking the current field out of view. All three components move together during this adjustment.

Moving the display vertically without changing scale

You can move the display of a single component in the vertical direction by positioning the pointer within the appropriate section of the display, holding down both mouse buttons, and moving the pointer up or down to drag the display.

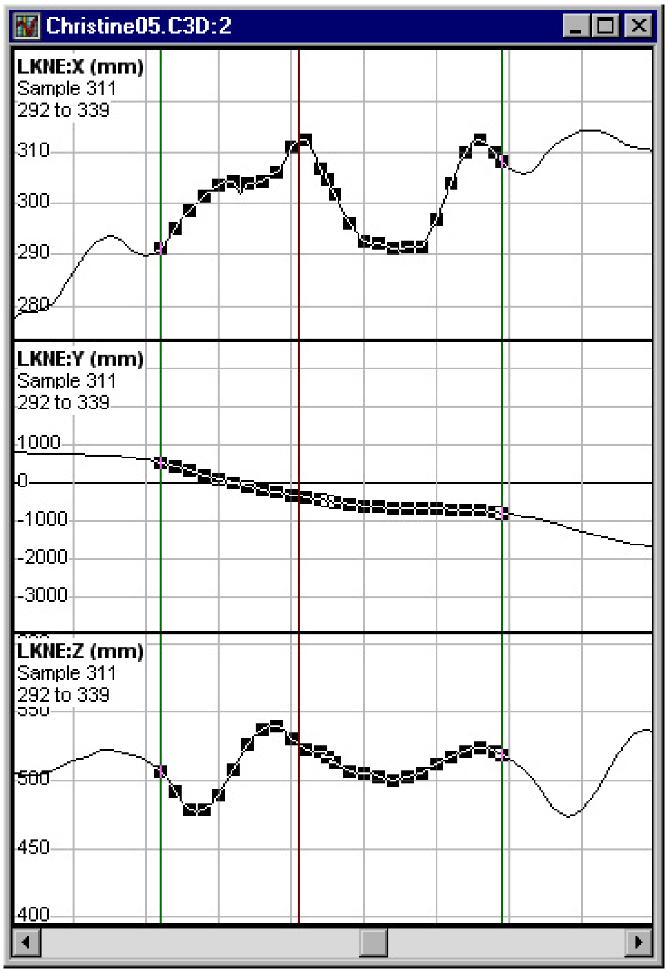
When both mouse buttons are held down, up-down and left-right mouse movements can be made simultaneously, moving all three displays laterally and one display vertically. New users will find it simpler to make one adjustment at a time.

Selecting a Field Range

For those operations which act on a range of fields (rather than the current field), you need to select the start and end points of the range before carrying out the operation.

Select the start of the range by making it the current field by any method (eg.

point and click within the Graph Editor window, or drag the replay slider bar). The red cursor will indicate this point. Then position the pointer within the Graph Editor window at the end of the desired range and left-click while holding down the Control key. A green cursor will appear over the selected end field.



In this example, the current field number (311) is shown above the field numbers and the red cursor. The start and end points of the range (292 and 339) are shown below and with the two green cursors, and points within the range are highlighted by black squares.

You can change the selected end field by repositioning the pointer and repeating the Control-click. The second green cursor will move to the new end field.

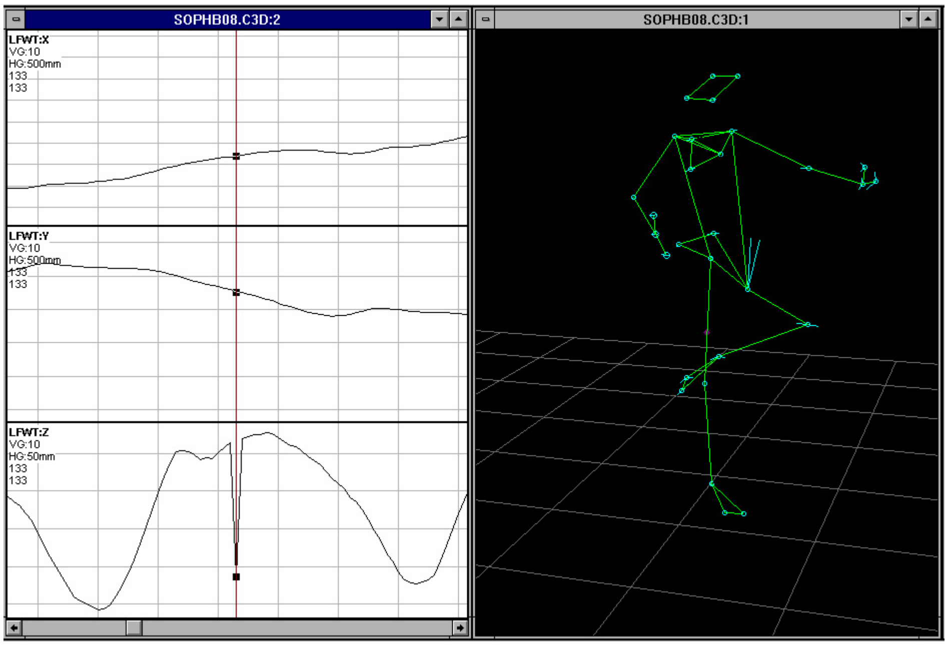
Each field within the range is highlighted by a black point, unless the time scale is such that there is no space to show each point. In this case, a reduced number of points within the range is shown. Use the scale change and lateral drag features to fit the selected range to the size of the window.

Once the start and end fields of the range have been selected, you can change the current field without changing the start and end fields of the range, by dragging the slider bar or by clicking on a highlighted point within the range (on any of the three graphs). If you change the current field, the red cursor will move from the start field of the range, to the new current field. The start of the range will then be indicated by a green cursor. However, if you change the current field by pointing and clicking within the Graph Editor window (other than on a highlighted point), the selected range will be dropped, and the new current field can be used as the start of a different range.

The operations which can be carried out on the points within a range are described in the chapter on Edit Menu commands.

Point Dragging

If the pointer is placed on a highlighted square, that co-ordinate can be dragged by the mouse and dropped at a new value. This has the effect of changing the chosen co-ordinate to a new value. The effect of point dragging is displayed in the associated Workspace window, if one is open.



In this example, the Z component of the point LFWT has been dragged down. The effect this has on the data is displayed in the associated Workspace window, in which the point is shown displaced downwards.

This form of point dragging is the simplest form of data manipulation using the Graph Editor window. More sophisticated tools are available in the Edit Menu.

For more details on point moving operations, see "Move Point Mode".

Text Editing

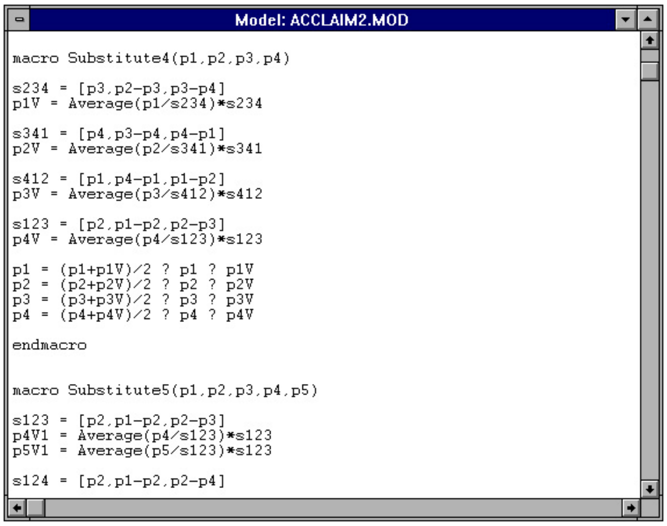
Two types of file - .MOD and .MP - control the action of BodyBuilder's most important function, kinematic modelling. When developing BodyLanguage models and adjusting parameters, it is often convenient to work by a process of repeated trials, with small changes to the model or parameter file tested immediately by application to typical data.

It would be possible to do this by using any standard text editor, such as Notepad, saving the text file after each change, switching to BodyBuilder by Alt-Tab, reloading the modified file, and then testing it. This would be a laborious procedure if it involved more than a few repetitions.

A text editing window has been provided within BodyBuilder so that this procedure can be carried out more efficiently.

Opening a Text Editing Window

In order to open a text editing window, select View Model from the Model menu. If a .MOD file can be found according to the settings for the current subject, it will appear in the text editing window.



```
macro Substitute4(p1,p2,p3,p4)

s234 = [p3,p2-p3,p3-p4]
p1V = Average(p1/s234)*s234

s341 = [p4,p3-p4,p4-p1]
p2V = Average(p2/s341)*s341

s412 = [p1,p4-p1,p1-p2]
p3V = Average(p3/s412)*s412

s123 = [p2,p1-p2,p2-p3]
p4V = Average(p4/s123)*s123

p1 = (p1+p1V)/2 ? p1 ? p1V
p2 = (p2+p2V)/2 ? p2 ? p2V
p3 = (p3+p3V)/2 ? p3 ? p3V
p4 = (p4+p4V)/2 ? p4 ? p4V

endmacro


macro Substitute5(p1,p2,p3,p4,p5)

s123 = [p2,p1-p2,p2-p3]
p4V1 = Average(p4/s123)*s123
p5V1 = Average(p5/s123)*s123

s124 = [p2,p1-p2,p2-p4]
```

The same procedure is used to open a parameter file editing window: select Model | View Parameters and a similar text editing window will open.

MOD, MP and MKR files, as well as AST files, may also be opened in a text window directly by using File Open, and selecting the appropriate file type. In this case, the file is not necessarily associated with a particular trial or subject. In particular, any changes to MKR files can only be made to take effect by saving the file, and opening the Subject Settings dialogue, checking that the correct file MKR file is selected, and clicking the Okay button.

Editing Text Files and Testing Changes

The text editing window works in a similar way to other Windows text editors (eg. Notepad), with Cut, Copy, Paste, Find and Replace commands available in the Edit menu, and Save or Save As... commands in the File menu. Changes can be undone using the Edit | Undo command, but only the most recent change is available with the Undo function. Note that Replace operations are not reversible using the Undo command.

However, it is not necessary to save changes to MOD or MP files before testing them. Simply selecting the Model | Run Model command applies the changed model directly, since BodyBuilder detects that the files have already been loaded.

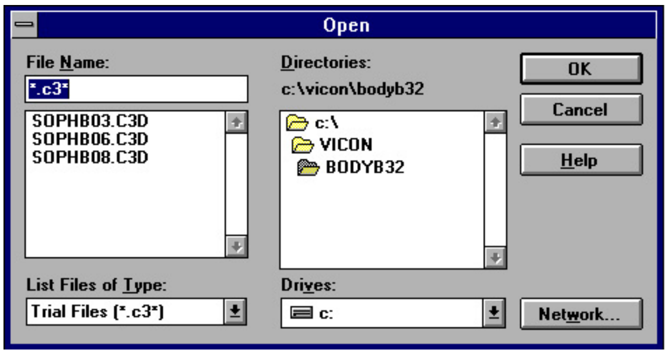
When the text file is in a satisfactory form, it should be saved before closing the editor window. Failure to save changes will result in their being lost when the BodyBuilder main window is closed, however, a warning message will appear if the main window is closed with unsaved changes outstanding. If a long editing and testing session is undertaken, the usual precautions should be taken to save work periodically in case of power failure.

File Menu

The File menu is always available while BodyBuilder is running, although the options within it depend on whether a .C3D file is open. The File menu can be accessed by pointing and clicking on the word File, or by the keyboard alternative Alt-F.

Open

The File | Open command (keyboard alternative: Control-O) opens the File Open browse box:



This default type allows the user to save modified data under the same filename, updating the extension to .C3E, .C3F and so on.

The default file type is .C3* and the desired file can be selected either by clicking on its name and then on the OK button, or double-clicking on the file name.

The File Open toolbar symbol is at the left end of the toolbar. The last four files used are listed in the file menu for quick re-selection.

Merge

If a very long data capture has been carried out, creating a large .TVD file, it may speed processing if it is cut into subsections before reconstruction and labelling. This is because very long 3D files make excessive demands of system memory. A utility is available with Vicon Workstation for this purpose. The result of reconstructing a divided .TVD file is multiple .C3D files, and before kinematic modelling is carried out, it may be necessary to merge these back into a single file. In order to carry out this merging, begin by opening the first .C3D file in the normal way.

Then, use the Merge command. This opens a browse box similar to the Open File browse box. The second .C3D file can be selected from this browser. The contents of this file will be merged with the first. Subsequent files are added in the same way.

Only files which are compatible with each other should be merged, or there may be unpredictable results. In particular, when merging files with different subjects, the trajectories for the subjects should be labelled with different prefixes.

When merging files that contain analogue data, note also that BodyBuilder will only accept analogue data from a single trial. If the currently loaded trial has analogue data, the analogue data from the trial you are trying to merge will be ignored. Otherwise, the new analogue data will be adopted for the merged trial.

Close

The Close command (or its toolbar shortcut) has the effect of closing the active window, whether it is a Workspace, Graph Editor, or Text Editor. If the active window is a Workspace or Graph Editor, and there are other windows displaying the same .C3D file, all will be closed.

Save

The Save command saves the the data in the currect trial, overwriting the C3D file containing the original data. The data is saved in the same format as the original file.

If a limited range of fields has been selected for saving, only the select number of fields will be saved.

A feature introduced in version 2.x of Workstation software.

BodyBuilder saves unlabelled trajectories in the .C3D file. Unlabelled trajectories may result from the use of the Unlabel command, or the Create New Trajectory command.

Multiple segments with the same label will be stored as a single trajectory. This is equivalent to defragmenting the trajectories.

All data resulting from editing and modelling is written to file. Virtual points created by modelling are flagged as such within the working session in which they are created, and can be identified as such and removed using the Delete Model Output command. However, once they are written to file, the flag is lost and modelled points become indistinguishable from any others.

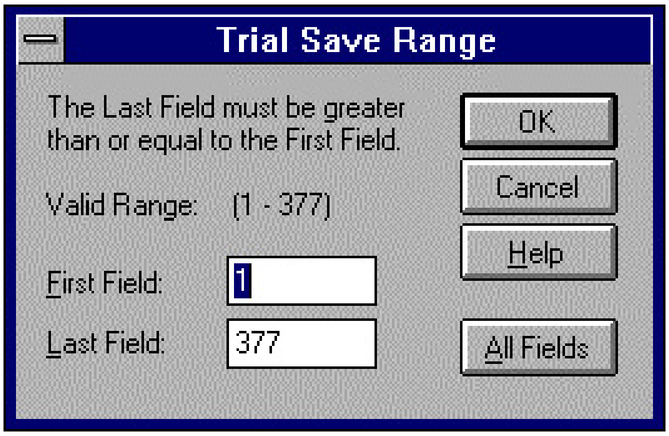
Kinematic angles calculated by modelling are stored as angles, and will be recognised as such when the file is read back in BodyBuilder or Reporter. A new parameter POINT:ANGLES is used to identify them. As with modelled points, they are flagged as model output when first created. (Previous versions of BodyBuilder saved angles in a form indistinguishable from points.)

The units these data are saved in depend on the data type. For more information, see the Save As... section.

Kinetic variables are also saved as trajectories in the file, and marked according to their type. They are recognised when re-loaded in BodyBuilder, and can be deleted by the Delete Model Outputs function.

Range to Save

The Range to Save command opens the Trial Save Range box:


Trial Save Range

The Last Field must be greater than or equal to the First Field.

Valid Range: (1 - 377)

First Field: 1

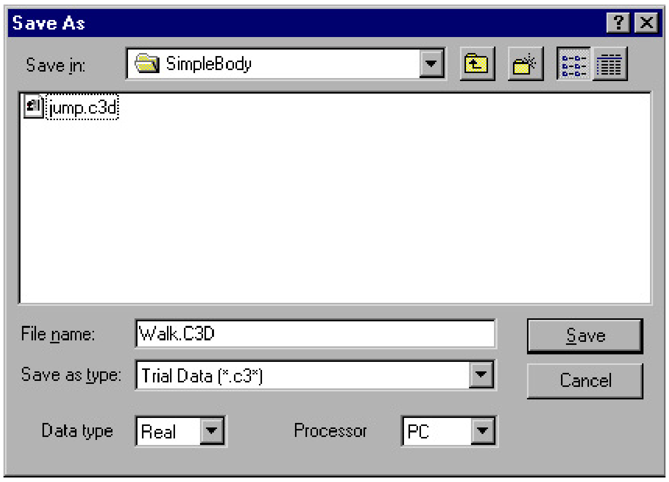
Last Field: 377

By default, the save range is set to the whole length of the file. If, though, only part of the data is of interest, it is possible to select a range of fields which will be written to file when one of the commands Save As, Write C3D or Write ASCII is used.

When a save range is specified, analog data outside that range will not be saved.

Save As

The Save As dialog box appears. By default the name of the original file is prompted. If this is accepted, the original data will be over-written. Any other name will cause a new file to be created with the edited data.


Save As

Save in: SimpleBody

jump.c3d

File name: Walk.C3D

Save as type: Trial Data (*.c3*)

Data type: Real

Processor: PC

It is possible to save edited data under the same name if the extension is changed (eg. to .C3E or .C3F). Note that Windows 95 may not show all file extensions in applications such as Explorer unless requested. If you save your work under a new name, this will appear in the title bar of the current Workspace and Graph Editor windows.

The Range To Save… frame range will be used.

The data can be stored in the file in different formats, according to the C3D file specifications. The data can be saved as Integer data or Real (floating point) data, in formats native to PC (Intel compatible), MIPS (SGI or Sun) or DEC (Vax) formats. Loading and saving files in the PC format will be slightly quicker on PC compatibles computers. The default setting matches that of the original file.

The Real data type takes up twice as much space as the Integer format, but can represent values, especially values calculated as a result of kinetic modelling, with much better precision than the Integer format. In the integer format, some variables can lose precision when the file is saved, and when it is re-opened, graphs of such values will have steps in them.

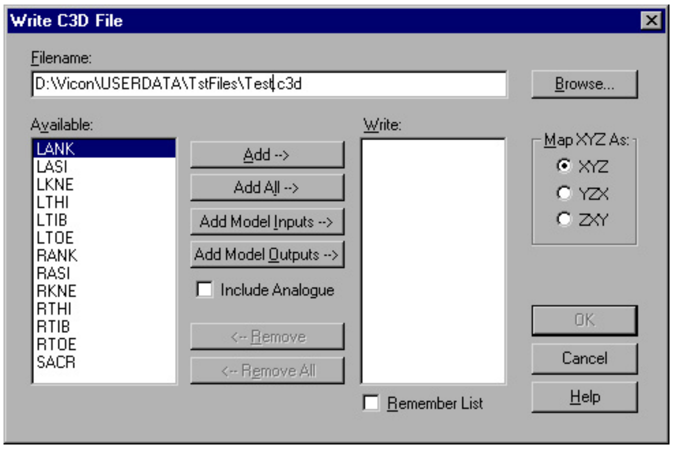
The units for the kinetic data are also dependent on the data type. For Real types, the kinetic data are always saved in the same unts as are used internally, that is N, Nmm, and mW. If the Integer data type is selected, the largest range of values for each kinetic quantity is found, and hence a unit chosen, and the trajectories appropriately scaled, such that the data just fit in the range permitted by the integer type. This gives the best resolution for integer storage, without requiring any further scaling or offsets to each data type. It does mean that the units for each quantity may vary from trial to trial.

Note that some older programs (such as VCM for example) only support one data type. Normally, for historical reasons, this will be Integer DEC format.

Write C3D

Write C3D is similar to the Save As command, in that it allows you to save the data displayed in the Workspace window to disk as a .C3D file. However, Write C3D provides the facility to choose what data is saved.

If a limited range of fields has been selected for saving, this command appears in the menu as Write C3D Range…



The user must select data to write to the output file using the Add (or Add All) and Remove (or Remove All) buttons. Two other buttons are provided: the first, Add Model Inputs, enters the labels of all trajectories in the input file in the Data to Write field; the second, Add Model Outputs, enters the labels of all the trajectories and angles generated by kinematic modelling in the current working session.

At least one label must be entered in the Data to Write list for a .C3D file to be created, and a valid path and file name must be provided.

If the Remember List box is checked, the same write list will be presented the next time the operation is invoked, subject to that data being available. Otherwise the Write list will initially be blank.

Analogue data for the save range will also be saved if you click the corresponding check box.

Axis Changing

Write C3D provides the facility to change the order in which the co-ordinates are written to file. Normally (but not necessarily), VICON Workstation measurements are expressed in a Cartesian frame of reference in which the vertical axis is called Z. Thus, in the .C3D files created under this convention, the first two co-ordinates of a point represent the two horizontal components, and the third, the vertical.

Some applications - such as computer animation packages which follow the convention by which the vertical axis is called Y - may require a different co-ordinate order. The user must select to map the XYZ co-ordinate order to one of three options.

The default option, XYZ, leaves the order unchanged.

The second option, YZX, writes the original second (Y) co-ordinate first; the original third (Z) co-ordinate second; and the original first (X) co-ordinate third.

The result is that the vertical direction in the new file is labelled Y.

The third option, ZXY, writes the original third co-ordinate (Z) first; the original first (X) co-ordinate second; and the original second (Y) co-ordinate third. The result is that the vertical direction in the new file is labelled X.

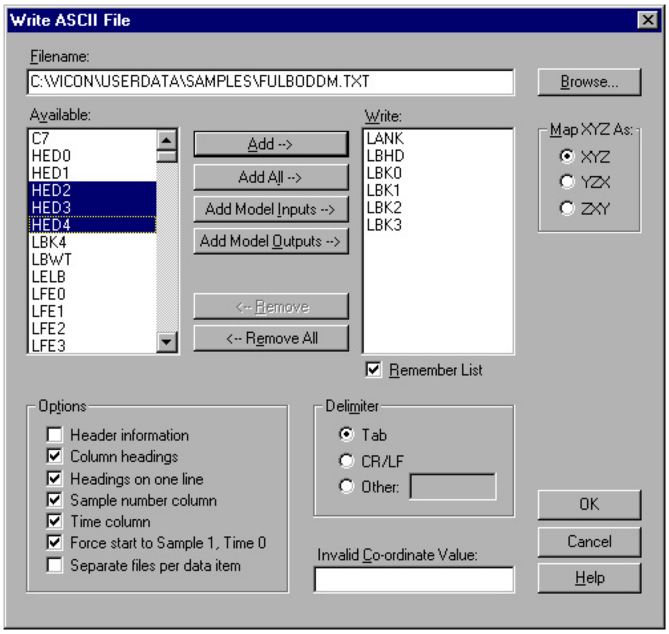
The three possibilities described are not the only possibilities; an exhaustive choice of axis permutations is offered by the Model | Axis Permutation command. However, that command affects model output to .ASF and .AMC files; only Write C3D allows the creation of .C3D files with changed axes. The choice of command depends on the output format needed.

Write ASCII

The Write ASCII command provides a highly flexible means of writing motion data to a text file suitable for display in a spreadsheet or for input to other applications.

If a limited range of fields has been selected for saving, this command appears in the menu as Write ASCII Range...

The Write ASCII command opens a box:


Write ASCII File

Filename: C:\VICON\USERDATA\SAMPLES\FULBODDM.TXT

Available: C7, HED0, HED1, HED2, HED3, HED4, LBK4, LBWT, LELB, LFE0, LFE1, LFE2, LFE3

Write: LANK, LBHD, LBK0, LBK1, LBK2, LBK3

Map XYZ As: [✓ XYZ] YZX ZXY

[✓] Remember List

Options:
- [ ] Header information
- [✓] Column headings
- [✓] Headings on one line
- [✓] Sample number column
- [✓] Time column
- [✓] Force start to Sample 1, Time 0
- [ ] Separate files per data item

Delimiter: [✓ Tab] CR/LF Other:

Invalid Co-ordinate Value:

The user must select data to write to the output file using the Add (or Add All) and Remove (or Remove All) buttons. Two other buttons are provided: the first, Add Model Inputs, enters the labels of all trajectories in the input file in the Data to Write field; the second, Add Model Outputs, enters the labels of all the trajectories and angles generated by kinematic modelling in the current working session. If the Remember List box is checked, the same write list will be presented the next time the operation is invoked, subject to that data being available. Otherwise the Write list will initially be blank.

At least one label must be entered in the Data to Write list for a .C3D file to be created, and a valid path and file name must be provided.

The Delimiter is the character which separates values which are entered on a single line. By default, values are separated by tabs. In this format, the file created will be easily readable by spreadsheet applications.

If another delimiter is required, click on Other, and enter the delimiter in the Other field. Any character may be used apart from combination keystrokes (eg Control-character).

The Invalid Co-ordinate Value field is blank by default. This means that if a co-ordinate does not exist in a given data field (perhaps because the marker was not visible in that field), a space will be left in the output file. Some applications require a value to be entered on every line whether or not a particular marker was in view; this is called the Invalid Co-ordinate value. For example, the value -9999 might be used instead of a blank space, but the value does not have to be a numeral.

The Map XYZ function behaves similarly to the corresponding function in Write C3D described above in the section on Axis Changing.

Options

The Options allow the user to control details of the file format.

Header Information, if checked, causes file header lines to be written containing information required by the 3D Studio computer animation package, and other application software. To maintain 3D Studio compatibility, select Header Information, Column Headings, Sample Number Columns, Time Column, and Force Start to Sample 1, Time 0.

Column Headings causes each column to be headed with the label and component (X, Y or Z). By default the label appears on one line, and the component name on the next line; but if Headings on One Line is selected, the labels and component names are placed on the same line, separated by a colon.

Sample Number Column causes the first entry on each line to be the field (sample) number of the data on that line.

Time Column causes the second entry on each line to be the time of the field in seconds (unless Sample Number is deselected, in which case Time is the first entry on the line).

Force Start to Sample 1, Time 0 writes the sample number and time information such that the first line of data is always sample number 1, at time zero, whether or not the first field in the trajectory save range is actually field number 1 of the .C3D file.

Separate Files per Data Item causes each trajectory to be written to a different file. If this option is selected, a sub-directory is created with the name shown in the Filename box. Each ASCII file is placed in it, with the same name as the trajectory it describes and the extension as chosen in the Filename box.

Previous Files

A list of the four .C3D files most recently opened is available in the File Menu for rapid access.

Preferences

The Preferences command allows the user to control certain aspects of the Undo operation of BodyBuilder.

Certain operations cannot be Undone: these are listed in the Preferences under the General tab. If you check these operations, a warning message will appear when these operations are invoked to remind the user that the Undo command will not be available after the operation. Such warnings allow the user to save their work prior to invoking irreversible operations. However, if the user prefers not to have warning messages, they can be disabled.

The number of operations which can be Undone is user-controllable. However, it is recommended that this number should not be increased much beyond the default value of 10 because it makes substantial demands on system memory.

If the Maximise CPU usage box is checked, the application will remain active when it has no tasks running, rather than lying idle. Depending on system resources, this may allow certain functions to perform a little better, such as data replay and workspace rotation. In most situations, though, this will make no noticeable difference.

The marker size as displayed in the workspace can be changed. The value is in millimetres. This has no effect on any modelling, where the marker size should be accounted for in the modelling script.

The default modelling files listed under the Model tab indicate the file names that are searched for when an "Anonymous" trial (one that has been manually labelled) is loaded, and the folder to be searched for models indicated by all trials.

Acclaim output files (ASF and AMC files) can either be output to the same directory as the trial used to generate them, or a single directory can be specified, where all of the output files are placed. Note that the old files will be overwritten if a new file is generated with the same name. Care may be needed in using this option, for example with manually labelled files which automatically generate the file name ANONYMOUS.ASF for static trials.

Options for the naming of AMC files, and for the ASF files used for the generation of the AMC’s can be selected under the AMC files tab. For trials with just a single subject, the AMC file is normally called <Filename>.AMC, unless the "Use for both single and multiple subject trials" checkbox is checked. In this case the subject name is included too.

Edit Menu

The Edit Menu provides a range of powerful tools for manipulating the data in a .C3D file as displayed in Workspace and Graph Editor windows.

The exact range of options available depends on the types of window open. The full range of options is available when a .C3D file is open in a Workspace Window and a Graph Editor window has also been opened.

Undo and Redo

Keyboard alternatives:
Undo: Control-Z
Redo: Control-A

Most operations in BodyBuilder can be undone, ie. the data can be restored to the condition prior to carrying out the operation. Certain operations are not reversible in this way, and unless they have been disabled in File | Preferences, warning messages will appear before these operations take effect.

If an operation is Undone, it can be redone either by invoking the original command, or by using the Redo command.

Snip

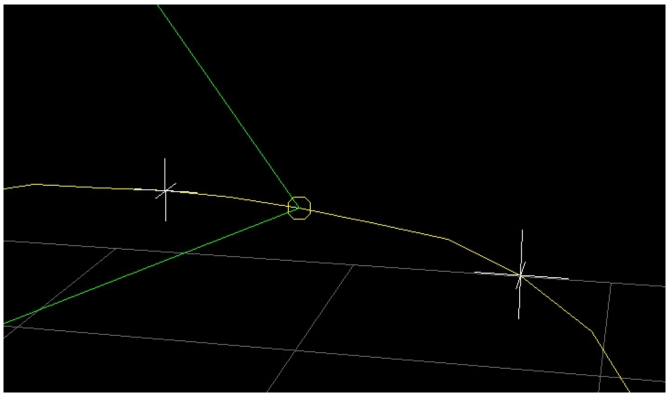
A trajectory consists of a time-series of points which are linked together by having a common label. If you want to separate a trajectory into more than one segment, possibly so that one or other segment can be given a different label, the snipping function allows you to introduce breaks without deleting any points.

To snip a trajectory, select the point at which you want to snip, and choose Snip from the Edit menu. A break is introduced after the selected point (ie. the selected point becomes the last point of the earlier segment, and the next point becomes the first point in the later segment). At this stage, the two segments still share the same label, but they can be selected separately, and can be re-labelled individually.

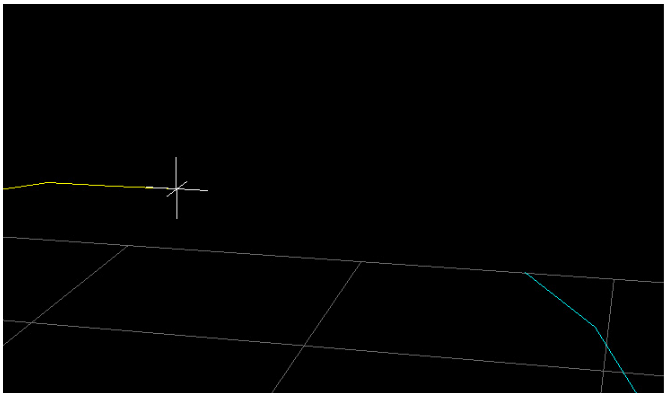
This function may be useful if two nearby trajectories show a "cross-over" error. Both trajectories can be snipped at the cross-over and re-labelled without losing any data.

Selecting a range of two consecutive points is the same as only selecting the first.

It is also possible to snip out a range of points. Select the first point as above, and the second using Control-click. The Snip command deletes the points between the selected end points. The selected points themselves remain undeleted.



In the above example, two points (highlighted by white crosses) define the end points of a range. After the Snip command is invoked:



The end points of the range remain; the points in between are deleted. The earlier of the two trajectory segments remains selected (shown yellow); the later segment is unselected (cyan).

Delete

To delete a single point, position the pointer on it and click the left mouse button. A white cross will appear on the selected point. The trajectory will also change colour, indicating selection. If the wrong point is selected, move to the correct point and left click again.

Keyboard Alternative: - (minus sign)

Select Edit | Delete. A break will appear in the selected trajectory, indicating that the selected point has been removed.

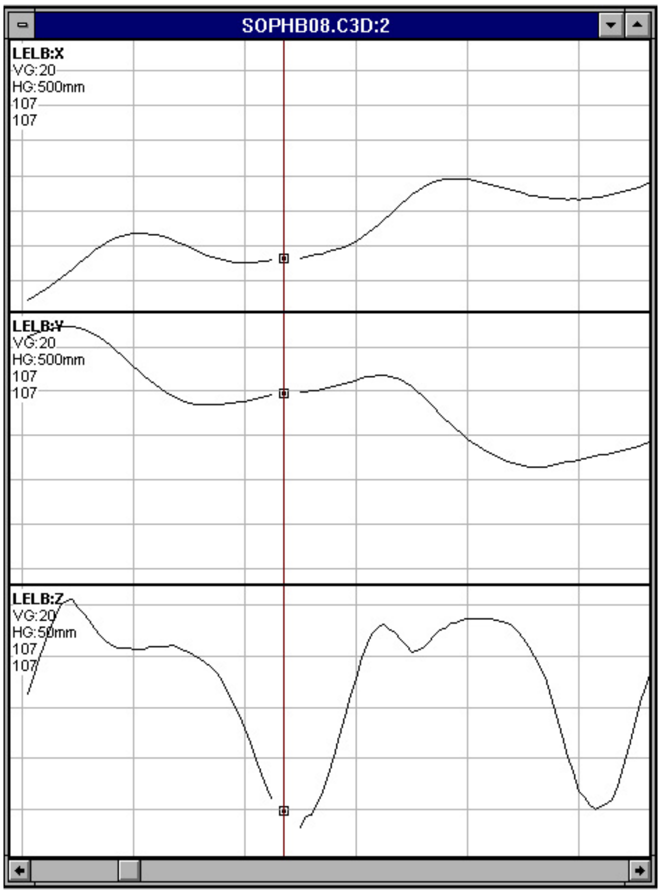
To delete a range of points from a trajectory, select one end of the range as described above. Select the other end by pointing to it, holding down the Control key, and clicking the left mouse button. A second white cross will appear over the selected point. If the second point is incorrectly selected, move to the desired point and Control-click again.

Select Delete from the Edit menu. The two selected points, together with all the points between them, will be removed from the trajectory.

Add Point

This command is available if a Graph Editor window is open and showing a trajectory which has a gap, and if the current field is set within that gap.

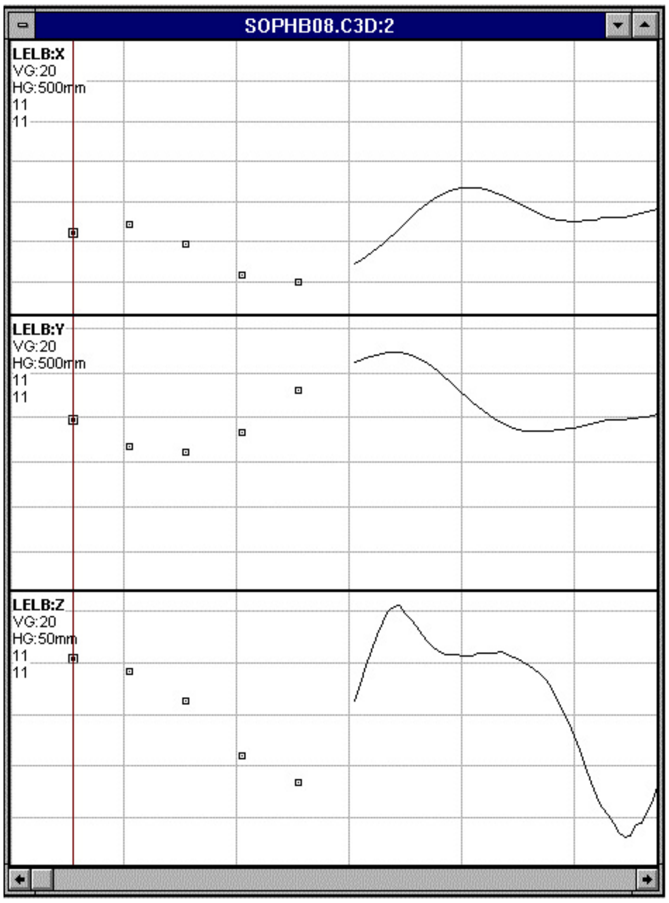
The action of the Add Point command is to create a new point with the label of the selected trajectory, which is placed in the current field by linear interpolation from the rest of the data.



In this example, a Graph Editor window is open, displaying a trajectory LELB in which there is a gap. The red cursor indicates that the current field is set within the gap. The Add Point command has placed a new point within the gap, as shown by the square.

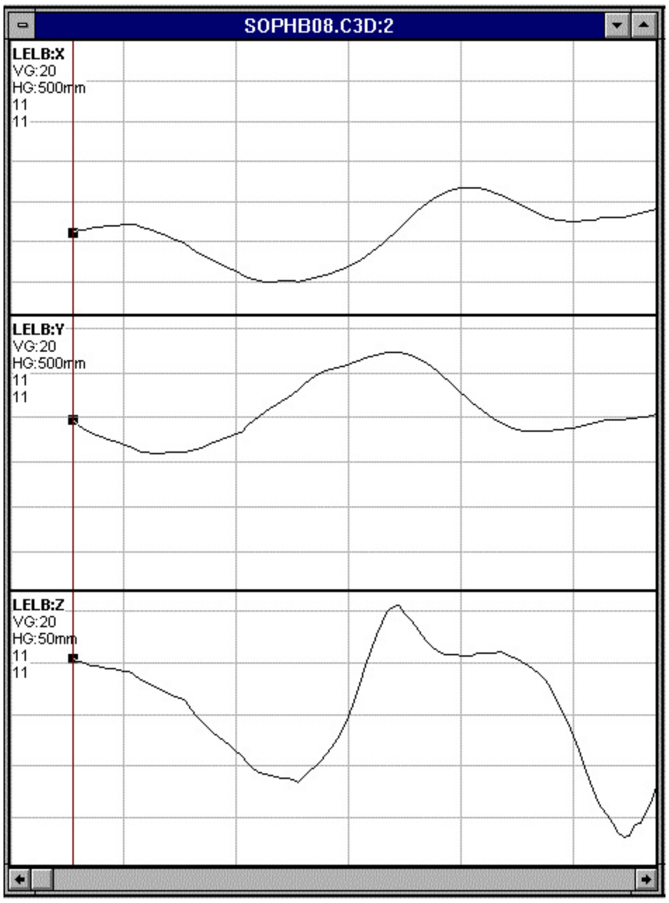
This use of the Add Point feature, depending as it does on linear interpolation, is less useful for filling gaps in trajectories than the Fill Gaps command, which uses more sophisticated interpolation methods.

The Add Point feature may be more useful in extending trajectories which start later or finish earlier than desired. Trajectories cannot be extended beyond their end points by interpolation, so there are two other methods available. One is to write a model script which creates new points by defining the position of the missing marker relative to others which are present in the field range concerned (see BodyLanguage documentation for instructions). The other is to use Add Point to create a new point outside the existing range of data, drag the new point to a good position, and interpolate over the gap. Extra points may be added in the gap to guide the interpolator. This use of Add Point is shown below:



In the above example, the trajectory LELB begins at field 61. It is necessary to create data before that time. The user has placed the red cursor at field 11, making it the current field. The Add Point command has created a new point with the label LELB in field 11. As there is no prior data for linear interpolation, Add Point set the co-ordinates of the new point equal to the values at the nearest end of the trajectory. The user can now drag the new point to a suitable position. Other points have been added and dragged in the gap to guide the interpolator.

The Fill Gap command then invokes the interpolator, and finally the new data can be manipulated with other editing tools until the desired result is obtained.



In the above example, after raising the maximum fill gap limit from the default 10 fields, the Fill Gap command is invoked. The interpolator fills in the missing points. The new data can be filtered and adjusted as necessary.

Create New Trajectory

This command opens a new Graph Editor window showing no data and with no label. It is possible to create data within such a window, and to attach a label to the new data, so as to have the effect of producing a new trajectory. This is an alternative method to using kinematic modelling to create trajectories, but may have advantages in certain situations.

If the Add Point command is invoked in a blank Graph Editor, a new point will be created at the origin (0,0,0). It is possible, for example, to create such a point, move it to a desired place, create another point in a different field, move it somewhere else, and interpolate between the two. This has the effect of creating a new, straight-line trajectory.

Delete Trajectory

The Delete Trajectory command deletes all selected trajectories. The command can be Undone.

Delete Model Outputs

When a model is run, the points created (including points representing kinematic angles) are flagged as modelled points. The Delete Model Output command will delete all such flagged points.

If a file containing modelled points is saved, then re-opened, these flags are lost, and the modelled points are indistinguishable from other points. In this case the Delete Model Output command will have no effect on them.

This operation is not reversible using the Undo command.

Defragment Trajectories

If a trajectory is broken, or has been snipped, into several segments, it will have a fragmented appearance. Selecting one segment will not necessarily select the others with the same label. This can be repaired using the Defragment Trajectories command. All segments with the same label will be stored as a single trajectory. After defragmenting, selecting any segment will select the whole trajectory. Defragment does not invoke any interpolation.

If two segments with the same name overlap, when the defragment command is invoked, some points may be lost from the earlier trajectory.

When a .C3D file is saved (written to disk), defragmentation takes place automatically. This operation is not reversible by the Undo command.

Delete and Fill

Often, the purpose of deleting a point or range of points, is to smooth the trajectory by interpolating over the selected range. BodyBuilder can fill gaps by interpolating from the data either side of the gap. The interpolated data will have no high-frequency components, ie. it will have a smooth appearance. Thus, the delete and fill command amounts to low-pass filtering by a different method.

A range of points within a trajectory can be selected as described above. If the Delete and Fill option is selected from the Edit menu, the selected range of points is cut from the trajectory, and the gap is filled by the interpolator. The keyboard shortcut for this function is the Delete key.

Fill Gaps In…

If a trajectory has been selected, this command appears as, for example, Fill Gaps In LELB. If several trajectories have been selected, the last selected is nominated.

This command invokes an interpolation routine which creates new points in any gap in the selected trajectory which are shorter than the maximum number of fields for interpolation (default 10).

The interpolation routine produces a series of points in the gap(s) which should be a reasonable estimate of the position of the marker during the period it was obscured from view. If the maximum fill gap is set to a much larger value than 10, the interpolated points will not be an accurate estimate of the position of the marker. In this case it may be better to use kinematic modelling to recreate the missing points, if possible.

If a range of points within a trajectory has been selected, only gaps in that range will be filled. In this case, the maximum fill gaps limit does not apply.

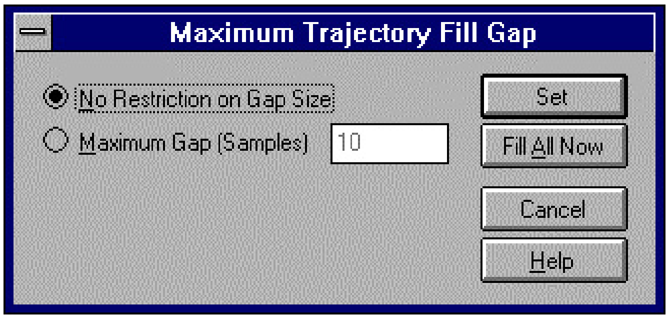
The interpolated segment(s) of trajectory will be smooth. This means that no frequencies above half the sampling rate will be represented. Interpolation is therefore, to some extent, an alternative to filtering; this option is provided by the delete-and-fill command. For more details on filtering, see the section on Filter Characteristics.

Fill All Gaps

This command invokes the interpolation routine for all trajectories, whether selected or not. It operates on any gap shorter than the maximum fill gap.

Maximum Fill Gap

This command opens a box allowing the user to select the maximum number of fields over which the interpolation commands will operate. It is also possible to remove this limit entirely, although this may result in some very long gaps being unrealistically filled.


Maximum Trajectory Fill Gap

[✓ No Restriction on Gap Size]

Maximum Gap (Samples) 10

By default, the maximum is set to 10 fields, because this should result in realistic interpolations. In the example above, the No Restriction option is selected, removing the maximum gap limit.

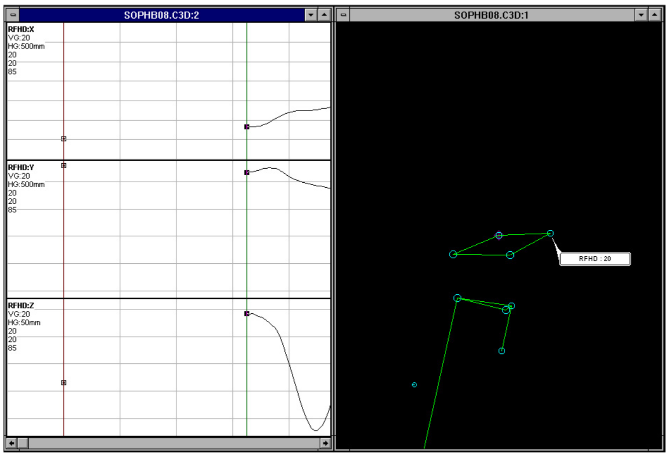
The Set button applies the selected limit and closes the box without invoking the interpolation routine. The Fill All Now button applies the new limit, closes the box, and invokes the Fill All Gaps command.

Copy Pattern

The Copy Pattern command is available only once a Graph Editor window is open, displaying one trajectory, and a Workspace window displaying the same .C3D file data is also open, with a different trajectory selected.

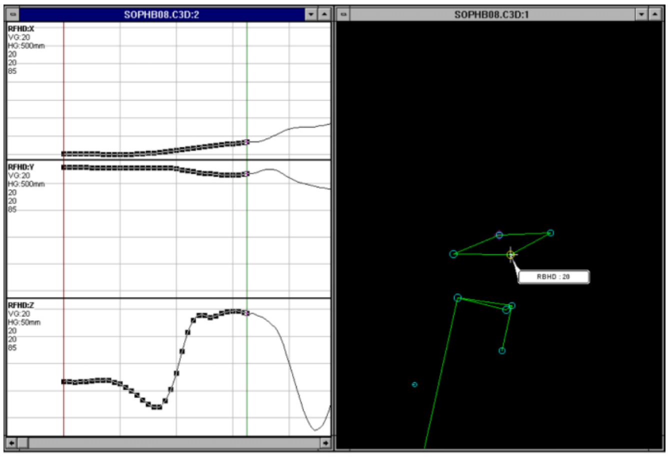
The purpose of Copy Pattern is to fill a gap in one trajectory over a certain range, using another trajectory as a pattern. The trajectory with the gap to be filled should be displayed in the Graph Editor window, with the start and end points of the range to be filled selected. There must be data points at the start and end of the range to be filled: if necessary, one or both of these can be created using Add Point and dragged to suitable places. The selected range is indicated by green cursors, with the red cursor, as usual, indicating the current field.

The Copy Pattern command takes another trajectory as the template for filling the gap highlighted in the Graph Editor window. You can use point-and-click within the Workspace window to select the pattern trajectory. The pattern trajectory should be complete throughout the selected range.



In this example, the Graph Editor window shows the trajectory RFHD, which is missing from the range 20 to 85. The first appearance of this point is in field 85; a point has been added in field 20, and dragged to a realistic position. The two points, highlighted in the Graph Editor window, have been selected as the beginning and end of the range.
Next, the pattern trajectory (RBHD) should be selected in the Workspace window.

When these steps have been completed, click on the Graph Editor window title bar to make it the active window. The Copy Pattern command will then be available in the Edit menu. In the example above, the command would appear as Copy Pattern from RBHD.



Continuing the example above, the trajectory RBHD has been selected in the Workspace, and the Graph Editor window made active once again.

At this point, the Copy Pattern command has been used. A trajectory similar to that of RBHD in the range 20 to 85, has been added to RFHD, and forced to fit the end points of the gap in RFHD.
If necessary, the new points can be further manipulated until satisfactory.

The Copy Pattern function works as follows. For each component (X, Y and Z), the pattern trajectory is analysed and the deviations of the trajectory from a straight line between the end-points of the range are stored. (The function makes no changes to the pattern trajectory).

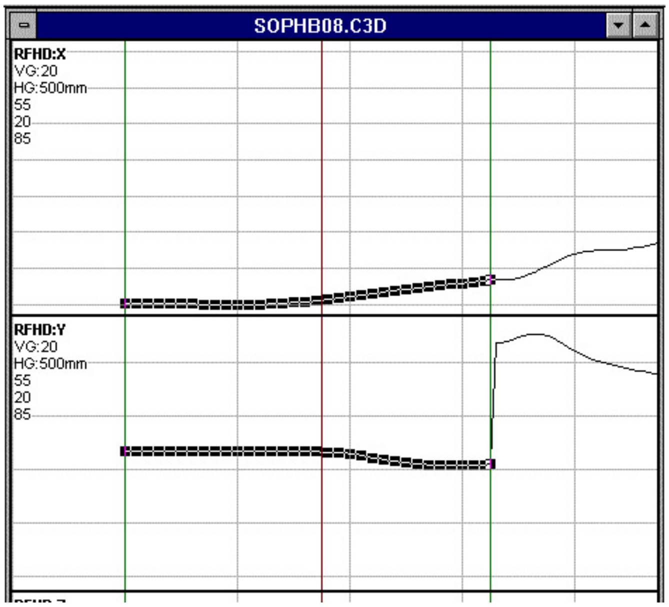
The target trajectory must have two end points, so that a straight line can be drawn between them. The Copy Pattern function then applies the stored deviations to that straight line, creating the new data. In this way, the shape of the new trajectory resembles that of the pattern, but stretched to fit the required end points.

Move Point Mode

Move Point is a useful tool for modifying data in the Graph Editor window. There are four modes, which are selected from a submenu under the Move Point Mode command. Move Point is a mouse dragging operation by which a component of a trajectory can be changed by dragging its highlighted square up or down the graph. If a single field value is selected (ie. a range of one field), the operation is so simple that it is not affected by the selection of any Move Point mode. However, if a range of several points is selected, four choices of mode are available.

Simple

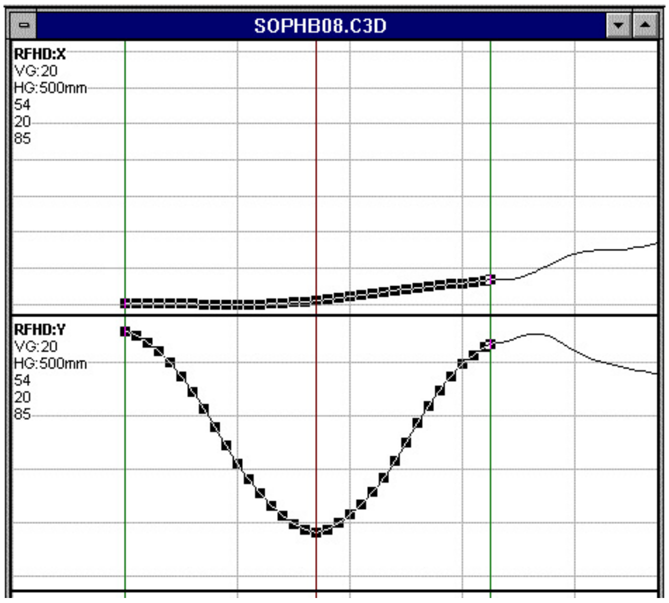
In Simple mode, a range of one component of a trajectory may be dragged using the mouse. The whole of the selected range is dragged up or down the graph together, by an equal amount, with a sharp discontinuity resulting at the end of the range.



In this example, the Y component has been dragged down in simple mode. In simple mode, it makes no difference which point is dragged.
There is a discontinuity at field 85 between the dragged points and those which remain where they were. This would show as a sudden jump in the trajectory.

Weighted A mode

To make a smoother trajectory, you can use Weighted A mode, in which the point that is dragged is moved according to the distance dragged, and the rest of the points in the range are moved by a lesser amount.



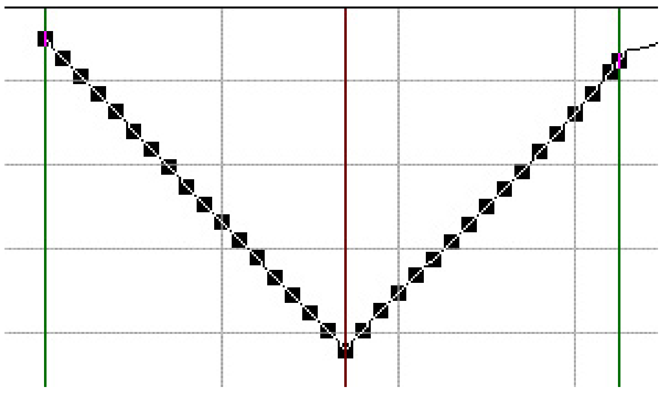
In this example, the selected range of the Y component has been moved in Weighted A mode. The actual point dragged - shown by the red cursor - moves according to the extent of mouse movement. The other points in the range move by a lesser amount, forming a sigmoid curve. The steepness of the sigmoid curve is determined by side-to-side mouse movement. This example shows mid-range steepness.

The function used to create the new values of the modified component is called a sigmoid. Three rules govern this curve:

- The end points remain fixed
- The dragged point moves to the extent determined by dragging
- The mid-points of each wing of the curve move by half that amount

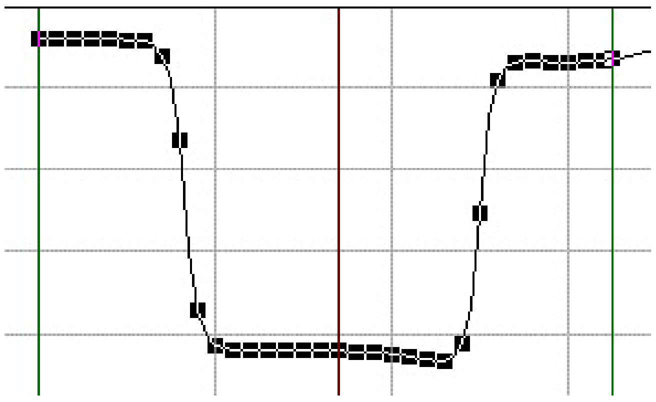
The steepness of the sigmoid curve is determined by lateral mouse movement. At one extreme the shape resulting is a straight-sided triangle fitted through the points determined by the three rules:

This is reached by moving the mouse to the left while dragging the apex point down.



At the other extreme, the shape is something like a top-hat function, fitted through the points determined by the three rules:

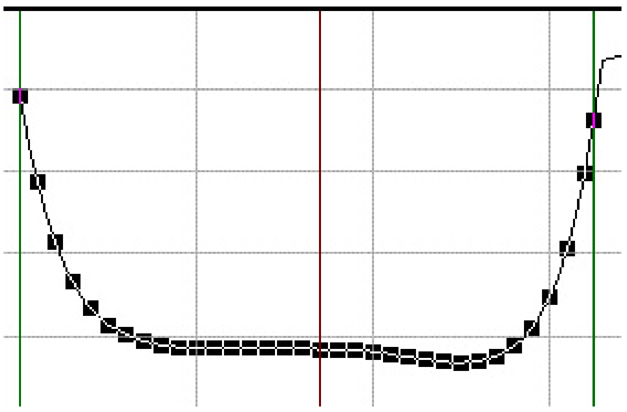
This is reached by moving the mouse to the right while dragging the apex point down.



Weighted B mode

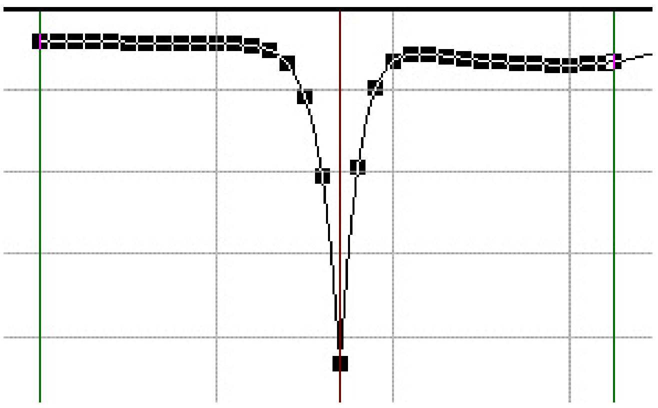
This is similar to Weighted A mode, but follows different rules. The mid-point of each wing of the curve is no longer fixed; instead, it is moved by lateral mouse shift to create either a convex or concave curve:

This effect results from moving the mouse to the left while dragging the apex point down.



The midpoint of the lateral mouse shift range produces a straight sided triangle. The opposite extreme is a convex curve with a spike appearance:

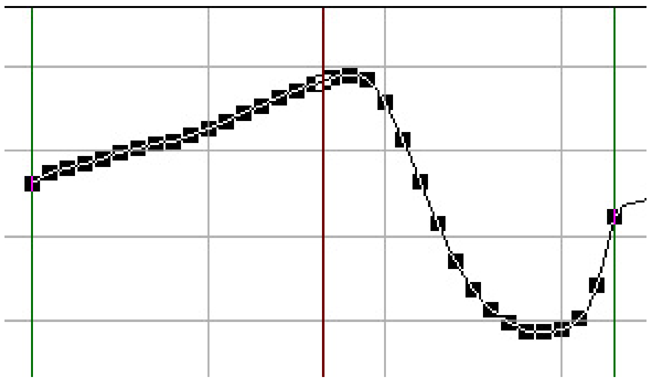
This effect results from moving the mouse to the right while dragging the apex point down.



Expand/Reduce mode

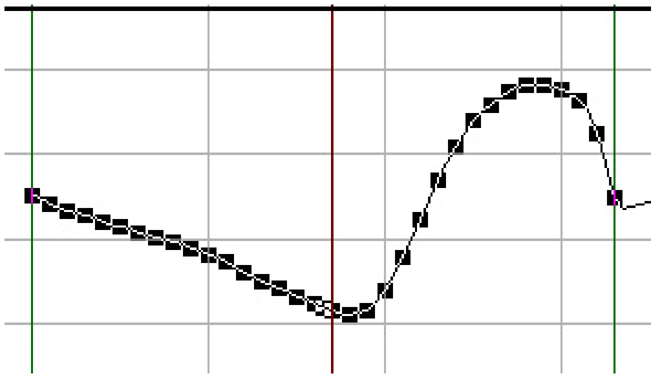
The fourth Move Point mode allows you to exaggerate the curvature (ie. departure from a straight line) of the selected range of points. Once again, the dragged point (indicated by the red cursor) is moved by the amount of up/down drag. All other points are moved by an amount which depends on their deviation from a straight line drawn between the start and end points, and proportional to the amount of mouse drag.

This effect was produced by selecting expand/reduce mode and dragging the apex point (red cursor) upwards.



Opposite mouse movement causes deviation in the opposite direction:

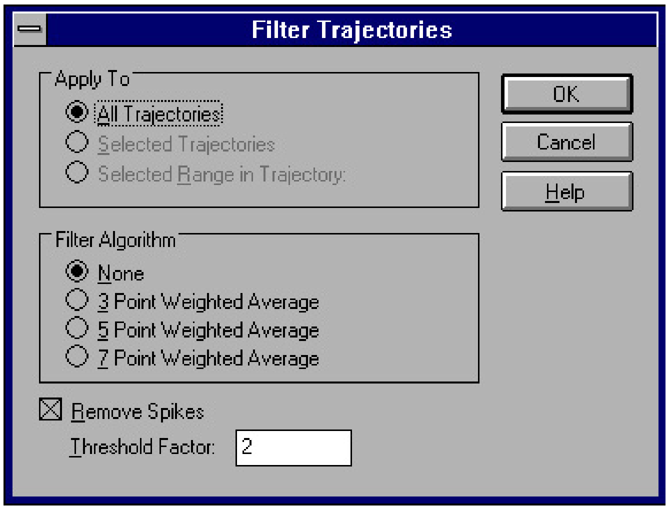
This effect was produced by selecting expand/reduce mode and dragging the apex point (red cursor) downwards.



Filter

Filtering operates either on selected trajectories, or if none are selected, on all trajectories. The effect is to pass only low frequencies, ie. to smooth the trajectories by ironing out sharp features.

The frequency characteristics of the filter function are user-selectable. If you select Filter from the Edit menu, a box appears offering the choice of filtering options.


Filter Trajectories

Apply To
(● All Trajectories) ○ Selected Trajectories ○ Selected Range in Trajectory:

Filter Algorithm
(● None) ○ 3 Point Weighted Average ○ 5 Point Weighted Average ○ 7 Point Weighted Average

☒ Remove Spikes

Threshold Factor: 2

The first option determines which trajectory or trajectories are filtered. It is possible to limit the filter to a selected range of a particular trajectory by selecting the trajectory and defining the range using a Graph Editor window.

Filter Characteristics

The second option determines the filter characteristics. The filter with the highest passband is called "3-point". This filter modifies each co-ordinate according to the function

New Value = (previous sample + 2 ¥ old value + subsequent sample)/4

where "previous sample" and "subsequent sample" are the neighbouring points in the trajectory. This is called a 1-2-1 filter function, and will smooth out trajectories with a sawtooth appearance.

Stronger filters are also available, with 5-point (1-3-4-3-1) and 7-point (1-4-7-8-7-4-1) functions. These substantially reduce the frequency content of the data and should therefore be used with caution. For example, the motion of body parts which naturally have high-frequency components, such as feet or fists, may become blurred by excessive filtering. Repeated use of the 1-2-1 filter will have a similar effect to the use of a stronger filter.

Because filters remove high frequency components, they also have the effect of reducing the excursion of a point from the overall direction of its movement. With repeated heavy filtering, the trajectory of a marker which was moving in a roughly linear path, would ultimately become a straight line. Similarly, if a marker is moving in a circular path, without overall displacement, the effect of filtering will be to reduce the radius of curvature of the path. If repeated, this would ultimately bring the point to a standstill.

If a segment of a trajectory is snipped at both ends and selected, the filter will act only on the selected segment. Repeated filtering will tend to reduce it to a straight line. This is unlike the delete and fill function, which produces a smooth path between the end points, but maintains curvature. Repeated delete and fill operations on the same segment would not produce further changes.

Spike Removal

If "Remove Spikes" is selected from the Filter box, you may choose to remove spikes from the selected trajectory. Spikes occur when the reconstruction process has located a single point which is to one side of the position which would be expected from the surrounding points. Spike removal is a special case of the Delete and Fill function, in which the program automatically identifies such points and carries out a delete and fill on each one.

The threshold setting determines how far a point has to be from those on either side, to be identified as a spike. A higher value leads to fewer points being identified as spikes. Once again, this function should be used with care.

When spike removal is used, the selected trajectory is analysed, and a spike value representing the degree to which each point deviates from a smooth path is calculated. A point which lies directly between its neighbours has the minimum spike value of 1; a point which deviates from this path has a higher spike value. Any point with a spike value higher than the threshold is then deleted and filled. A threshold value close to 1 will, therefore, result in a high proportion of points being deleted and filled. A value of 2 may result in only a few points being removed. The most appropriate value of the threshold can only be determined by trial, beginning with a higher value and reducing it until the desired result is obtained.

Refer to VICON WORKSTATION Users Manual for advice on improving your data.

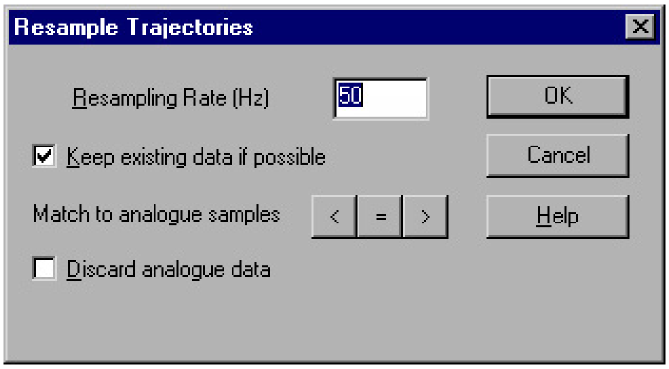
If there are many spikes, it may be better to filter the whole trajectory rather than spike-remove a high proportion of points. If spikes are a frequent problem, it would be a good idea to re-calibrate and see if a better result can be obtained. It may be necessary to re-linearise the cameras. However, if the problem is not excessive, spike removal and filtering will enable you to produce satisfactory results without repeating data capture.

If both filtering and spike removal are selected, spike removal is carried out first.

Re-Sampling

If the desired output is a time-series of point data at time intervals different to that of the input file, the re-sample function allows the user to fit a curve to the data and calculate new points at a user-specified time interval. This function may be used to produce new points at either a higher or lower sample rate than the original sample rate.
If the output field rate is a precise sub-multiple of the original (eg. 10Hz from an original 60Hz), the user may choose to use the original data by discarding the appropriate intermediate points. However, in most cases the re-sampling function works by fitting a curve which is a close approximation to the original data, and then using this to calculate new points. There is, in this case, a minor low-pass filtering effect. Note that resampling at higher rates does not in fact

give you more information. It just increases the amount of data you have (and the size of your file).



If gaps are present in the data, re-sampling will have the effect of invoking the Fill All Gaps command with no maximum gap size limit.

Re-sampling is not reversible using the Undo command - if you are in any doubt about re-sampling, save the data to your hard disk before using this command.

If analogue data is present, buttons will be enabled which will allow you to resample the trajectories at rates compatible with the analogue data (i.e. at rates that wil gaurantee that analogue samples will be coincident with trajectory samples). The < and > decrease and increase the rate. The = button sets the trajectory sampling rate to the same as the analogue. (Be warned that resampling at such high rates may result in huge files. This option is really made available only for users who are doing impact studies.)

If you type in a rate which is not compatible with the analogue data, the analogue data will be discarded (you can choose this option manually if you deliberately want to discard the data). If you want to find a rate which is compatible with what you've typed in, then press the < or > button, and the next closest rate will be found for you, and the analogue data will be retained.

View Menu

The View Menu controls the appearance of certain items in the main and Workspace windows. Most of the commands in the View menu are only available if a Workspace window is active.

Toolbar

The toolbar appears by default (ie. unless you select otherwise). The toolbar is a row of buttons which provide rapid alternatives to menu commands. Each button corresponds to a frequently used command. Not all commands are represented by tools; the View menu commands are not represented.

The tools cannot be changed by the user, only selected for display or otherwise. They are grouped according to the menu to which the related command belongs, and given symbols suggesting function. The File Open tool is commonly used by Windows programs and will be familiar; others are specific to BodyBuilder. The user should gain familiarity with the tools by pointing at them; a device called Tool Tips causes the name of the related command to appear when you point at a tool for longer than one second. If the user prefers not to have the toolbar displayed, the Toolbar command in the View menu toggles the display on or off.

Status Bar

The Status Bar appears at the bottom of the main program window, occasionally displaying helpful messages indicating program function and status.

If the user finds this unhelpful, it can be removed, which allows the working windows to be displayed slightly larger. If the user prefers not to have the status bar displayed, the Status Bar command in the View menu toggles the display on or off.

Rotation Centre

In a Workspace window, the centre of rotation (if selected) is shown by a purple diamond. The axes of Workspace viewpoint rotation pass through this point, and the zoom function acts towards or away from this point.

Like all the View menu commands, this one works as a toggle, meaning that it has two states, and each point-and-click on the command changes the state. If the user prefers not to have the rotation centre displayed, this command can be used to turn off the display. The keyboard alternative is R.

Markers

In the Workspace window, the positions of the markers in the current field are shown by small circles (or, to be precise, octagons).

If Markers are deselected, these will be hidden, leaving the trajectory and stick displays to indicate the motion of points. The keyboard alternative is M.

Stick Figure

The display of sticks joining points is usually helpful, but if the user prefers not to display sticks, they can be deselected. The keyboard alternative is S.

Floor

The 500mm square tile floor appears by default in the plane defined by Z=0. If this is not helpful, it can be deselected by this toggle. The keyboard alternative is W.

Flicker Free

Depending on the hardware (in particular the use of graphics acceleration) it may reduce flickering if the processor takes control of the Workspace window, at the expense of replay speed. Use this toggle to select the display mode which gives the best result on your PC. The keyboard alternative is Control-F.

Always Refresh

Depending on personal preference, the user may choose to refresh the Workspace window after each field is drawn, during a replay, at the expense of replay speed. Use this toggle to select the preferred mode. The keyboard alternative is Control-R.

Set Frame Rate..

The Workspace window may replay every frame of data at the speed determined by the processor and graphics hardware, or can replay the data as if it were captured at different rates. Pressing the Reset button returns the replay rate to the true capture rate.

Model Menu
The model menu commands allow the user to apply kinematic models written in BodyLanguage, and to view the results in the Workspace window. The model menu is also the key to animation-format output functions.

Run Model
The Run Model command runs the BodyLanguage model for the current trial (.C3D file). The sequence of operations is as follows:

- Read .MP and .MOD script files
- Carry out kinematic modelling as defined in the script
- Calculate required outputs
- Update Workspace window to show outputs
- Update parameters (.MP file) if required

The Run Model command causes a model to be run for each subject in the trial, using the model and parameters as specified in the Subject Settings dialogue box.

Although the Workspace window will appear different after the Run Model command is invoked, no new file is created. The model outputs are held in system memory, until the user saves the data by writing to disk. Until the data is saved, the operation can be Undone by selecting the Delete Model Outputs command from the Edit menu.

Note that if required, a new model can be selected, and run, using the outputs from the previous model as inputs to the new model.

If there are any errors in the model script, an error message will appear describing the error, and if appropriate, a text editor window will be opened with the offending line in the script highlighted. See the "View Last Error" command for more details.

Run Model for Subject
If the trial data currently displayed in an active Workspace window has a single subject (performer), or has been labelled manually in VICON Workstation, and there are no label prefixes, the Run Model for Subject command has exactly the same effect as the Run Model command. The subject is identified as "Anonymous" in such cases.

If the trial data currently displayed has been automatically labelled in VICON Workstation (version 2.5 or above) and so has subject-identifier label prefixes, then it is possible to run the model in such a way as only to take effect for one named subject. This is useful if more than one subject appears in the data. The currently named subject is shown at the top of the marker list bar on the right of the Workspace.

Using the Run Model for Subject command allows the use of different models for subjects with different names, appearing in the same trial.

Create ASF

Acclaim Skeleton Files are used by certain computer animation systems. Refer to the documentation provided for those systems.

The Create ASF command is intended for those users who want to produce an Acclaim Skeleton File using a BodyLanguage model and a .C3D file of static data. The specific requirements for this are explained in the section on Acclaim format modelling.

The .ASF file produced is based on a template file with the extension .AST which must match the input data and the model script.

The Create ASF command carries out the same procedure as the Run Model command, with the difference that the line

$Static = 1

is added to the beginning of the .MP file before the combined script is executed. This signifies that the data is a static trial, to be used for creating the skeleton data file. The result of running such a model is a text file with the extension ASF.

The .ASF file created by this command will have the same name as the input .C3D file.

Create AMC

The Create AMC command is used to create an Acclaim motion file using a BodyLanguage model and a .C3D file of motion data. The specific requirements for this are explained in the section on Acclaim format modelling.

The Create AMC command carries out the same procedure as the Run Model command, with the difference that the line

$Static = 0

is added to the beginning of the .MP file before the combined script is executed. The result of running this kind of model is a text output file with the same name as the input .C3D file, and the extension .AMC.

An .ASF file is required before this command can be used.

Create AMC Files...

This command allows the user to process multiple .C3D files and create .AMC files in batch mode. The input files are selected in the Create AMC dialogue.

The .C3* input data files must all be in the same folder. This may be selected by clicking the adjoining Browse button, and selecting one of the required AMC files from the folder. By default, all trials are selected except for those which already have .AMC files.

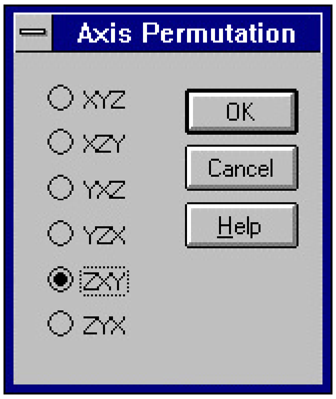
In the case of auto-labelled trials, the ASF file, and model and parameter files used for processing are determined from the subject name, and the marker set file name used during auto-labelling. Where these files cannot be found, or a manually labelled trial is selected, the default files as shown at the bottom of the dialogue will be used. The model and parameter files are the same as those set in the Preferences dialogue, and the ASF file is initially set to be most recently generated one. A different ASF can be selected by clicking the browse button.

When the OK button is pressed, the trials will be processed automatically without any need for further intervention by the operator. One AMC file will be generated for each subject, in each of the selected trials.

Axis Permutation

The Axis Permutation command is used before the Run Model or Create ASF/AMC commands to change the axis convention used from the usual VICON convention (vertical component = Z) to another convention.

The Axis Permutation box appears:


Axis Permutation

( ) XYZ
( ) XZY
( ) YXZ
( ) YZX
(●) ZXY
( ) ZYX

Any global point co-ordinates in the output will be permuted in the way selected. In .AMC files, in which most movements are described relative to other body segments rather than in global co-ordinates, there is a root segment which is specified in global co-ordinates permuted in the way selected.

So, if the data as captured uses the normal VICON convention, it may be (for example) that the subject is facing the X-direction, with the Y-direction on their left, and the Z-direction upward. Then, their root segment in a .AMC file would have a rotation of approximately 0,0,0.

If it is preferred to change to the convention that the subject is facing the Z-direction (out of the screen), with the Y-direction upward, the permutation required is ZXY. The rotation values of the root segment will now be calculated in the order appropriate to the chosen format. For this reason, the Axis Permutation does more than simply re-order global position co-ordinates, like the Map XYZ As... function in the Write C3D command dialog box.

The data will be unchanged if the XYZ permutation is selected. The default selection is the last option used.

View Model

The View Model command will open a text editor window displaying the model script file for the current subject of the active trial. Subsequent invoking of the View Model command brings the model window to the top and opens it if it is minimised.
The window can be closed directly.

View Parameters

The View Parameters command displays the current subject's parameter file in a text editor window, in the same manner as the View Model command.

View Last Error

If the Run Model, Create ASF or Create AMC command generates an error, the View Last Error command opens a text editor window. The line at which the error occurred will be highlighted, and the error message will appear at the bottom of the main application window. If the error was due to a macro call, subsequent selections of View Last Error will toggle the highlighted line between the call to the macro, and the offending line within the macro.

Adjust Parameters

The Adjust Parameters command starts Interactive Parameter Adjustment (IPA) mode. The use of this mode is described in the next section.

Subject Settings

The first field in the Subject Settings box is a drop-down list of subjects for the current trial. Below this are the marker, model and parameter files currently selected for the given subject. These files may be changed using the Browse button beside each field.

When an automatically labelled C3D file is first opened in BodyBuilder, the model and parameter files are assumed to have the same name as the marker set used to perform the auto-labelling. Thus, a subject labelled using the file

LEGS.MKR is assumed to be compatible with the model LEGS.MOD. For a dynamic trial, the parameter file is assumed to have the same name as the subject. For a static trial, if the parameters file is not explicitly named, the basic parameter file for the model is loaded initially (e.g. LEGS.MP) to act as a template for a new parameters file with the subject's name.

Parameters

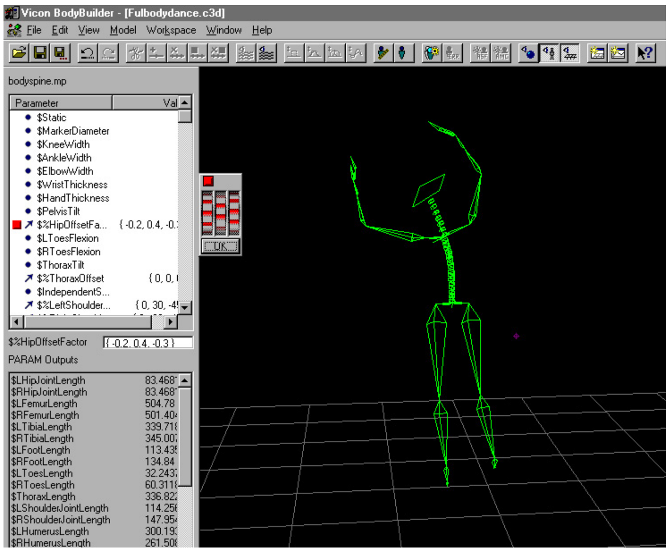
Parameters are numerical values which are part of a model, but which are stored in a parameter file separate from the model script itself, because the numbers are subject to frequent change. For example, they may include values of segment sizes which depend on the individual subject and the exact placement of markers, and so can vary from day to day. The parameter file is added to the model script when the model is run, but stored separately.

Often it is useful to edit the values in the parameter file during a modelling session, in order to change the results in some way. You may want to tilt a segment, or change the location of an internal point (such as a joint centre) in relation to the actual markers. These adjustments can be made interactively, with the effects of the change displayed immediately in the Workspace window. When interactive adjustment is completed, the final values of the selected parameters are entered in the parameter file just as if they had been entered using the keyboard.

Starting Interactive Adjustment

When a C3D file has been selected and the Workspace window is open, interactive parameter adjustment (IPA) mode can be entered by selecting "Adjust Parameters" from the Model menu. The IPA dialog bar will appear on the left of the Workspace.

If either the parameter file or the model file is currently open (ie. a text editor window has been opened, displaying the contents of these files), they will become locked while IPA is active. The files cannot be edited and the text editor windows cannot be closed. This prevents any mismatch arising between the files and the parameter and model text being used by the IPA function. The "Static" check box is also locked for the same reason.


Vicon BodyBuilder - [Fulbodydance.c3d]
File Edit View Model Workspace Window Help
OK

bodyspine.mp

Parameter	Val
$Static	
$MarkerDiameter	
$KneeWidth	
$AnkleWidth	
$ElbowWidth	
$WristThickness	
$HandThickness	
$PelvisTilt	
$%HipOffsetFa...	{ -0.2, 0.4, -0.
$LToesFlexion	
$RToesFlexion	
$ThoraxTilt	
$%ThoraxOffset	{ 0, 0,
$IndependentS...	
$%LeftShoulder...	{ 0, 30, -4

$%HipOffsetFactor { -0.2, 0.4, -0.3 }

PARAM Outputs

$LHipJointLength	83.468
$RHipJointLength	83.468
$LFemurLength	504.78
$RFemurLength	501.40
$LTibiaLength	339.71
$RTibiaLength	345.00
$LFootLength	113.43
$RFootLength	134.84
$LToesLength	32.243
$RToesLength	60.311
$ThoraxLength	336.82
$LShoulderJointLength	114.25
$RShoulderJointLength	147.95
$LHumerusLength	300.19
$RHumerusLength	261.50

Parameter List

The upper part of the IPA dialog bar contains a list of all the parameters found in the parameter file for the current subject in the C3D file (if a multiple subject file is open, only the currently selected subject is affected by changes). Parameter names must begin with the $ symbol to be recognised. The name of the parameter is shown on the left, and the value on the right. The relative width of the name and value columns may be changed by moving the split at the top of the pane.

Each parameter name has a symbol to its left, indicating the type - scalar, point or angle. A scalar has a small circle beside the name. A point (three dimensional quantity) has an arrow. An angle has a mathematical angle symbol. In some cases, a parameter is defined by an expression, rather than a simple definition. These values cannot be adjusted directly, but they are listed anyway with their values, and have no symbol beside the name.

The lower pane contains a display of PARAM outputs that would be appended to the parameter file if the model were run.

Adjusting Values

To adjust a parameter, left-click the parameter name. If that parameter is adjustable, a small thumbwheel window will appear.

A single thumbwheel will appear for scalar values, and a window with three thumbwheels will appear for Point and Angle types. Up to three such windows can exist at once. Further parameters can be adjusted by Control-Left-Clicking other names. The three thumbwheel windows are distinguishable by a coloured lozenge in their top bar. An identical lozenge also appears to the left of the corresponding parameter name.

Values can then be adjusted in two ways. First, the thumbwheels can be ‘grabbed’ by left clicking the mouse, and dragged up, to increase the value and down to decrease it. For Point and Angle thumbwheels, the leftmost wheel controls the X value, the middle wheel the Y value, and the right wheel Z. Pressing the Shift key while dragging in increases the amount the value is changed, and pressing Control reduces it.

Second, the value can be changed by typing in the edit box in the centre of the IPA bar. The parameter that appears here is the one most recently adjusted. When typing, invalid formats (e.g. missing commas for Point definitions) are ignored.

Each time the value is changed, the model is run for the single frame that is displayed in the workspace, and the outputs updated. Changes to the parameter values can be undone or redone by selecting Undo or Redo from the edit menu.

The thumbwheel windows can be closed by clicking the button at the bottom of the window. They also disappear automatically when new parameters are selected for adjustment.

The current frame may be changed during IPA. Model outputs for each frame selected during IPA will remain intact. The trial may also be played (using the forward and back arrow buttons) but play will stop if a parameter value is changed, and the model therefore re-run.

Different C3D Workspace windows may be selected, by clicking them if already displayed, or opening them from the file menu. IPA will load the corresponding models and parameter files. Parameter settings for the previous trial will be remembered and recalled when that trial is re-selected, although the thumbwheel windows will disappear.

In multiple subject trials, the subject may be changed, and the new subject's parameters will be adjustable. The display set may also be changed for the given subject.

Finishing IPA

Once the parameters have been adjusted to the required values, IPA mode may be switched off by pressing the "Finish" button at the bottom of the IPA dialog bar. The bar will disappear, and the final values for the parameters adjusted will be inserted in their appropriate places in the parameter file text. At this stage, the model can be run for the complete static trial, generating the required additional parameters.

Any changes to the parameters file or the trial made during IPA must be saved before closing the trial as the Finish button does not write the new values to disk. If a text editor window is not open for the parameters, the option to save them is offered when the Workspace window is closed.

The Cancel button also exits IPA mode without making any changes to the parameters text, and any trajectories generated by the model during the IPA mode are removed.

The Reset button causes the IPA mode to restart. The thumbwheel windows disappear, the parameter name and output panes are cleared and re-filled by re-reading the parameter file and re-running the model.

Workspace Menu

The commands in the Workspace menu are all concerned with the display of data in the Workspace window.

Select Centre

The centre of rotation and zoom can be changed to any point on a selected trajectory. Hold the pointer steadily at the chosen point and click with the left mouse button to select this point. If the trajectory segment is the only one selected, it changes colour to yellow. If other trajectories are already selected, another colour is used. Use the Select Centre command to make the selected point the new rotation centre.

Hide Trajectories

If you want to turn off the display of certain trajectories, such as phantom points (false reconstructions), select them and use the Hide Trajectories command. When a trajectory is hidden, it cannot be selected, and the mouse helper will not appear, even if the marker outline is still visible.

Show Trajectories

To turn the display of selected trajectories back on, use the Show Trajectories command.

If the hidden trajectory(s) is no longer selected, you can either use the label list to re-select them, or use the Show All Trajectories command.

Hide Other Trajectories

This command hides the display of all trajectories other than those selected. This is useful if you want to concentrate on one or two trajectories with extended Before and After settings, as the other trajectories will clutter the window.

Hide All Trajectories

If you prefer to see only marker outlines, without trajectory lines, the Hide All Trajectories command turns off the display of all trajectory lines.

Show All Trajectories

This command restores the display of all trajectory lines.

Toggle All Trajectories

This command allows you to turn the display of trajectories on and off rapidly. The keyboard alternative T is the most useful way to invoke this command.

Toggle Before Display

This command works the same way as Toggle All Trajectories, but affects only that part of each trajectory prior to the current field. The keyboard alternative B is the most useful way to invoke this command.

Toggle After Display

This command works the same way as Toggle All Trajectories, but affects only that part of each trajectory after the current field. The keyboard alternative A is the most useful way to invoke this command.

Defining the Problem

BodyBuilder for Biomechanics is not an application you simply load and run, like a word processor or a game. It is designed as a tool to enable users to design and carry out complex experiments in biomechanics, and because it allows total flexibility, there is no single, standard way to use it. The user must design their own experiment and way of working.

To derive useful results from your measurements, first it is necessary to have a clear idea of the purpose of the experiment. BodyBuilder for Biomechanics enables the user to model the body as a group of segments, each of which is treated as if it were a rigid body. These segments are linked to each other by joints. The purpose of a measurement, or series of measurements, might be:

- to develop a new model, based on new segment and joint definitions, or a new marker set

- to verify that a particular model is valid for its purpose

- to apply a verified model to a particular subject, to learn something about that subject

In the first case, the user will make extensive use of the BodyLanguage model-writing facility, editing the model and testing the changes to see the effect they have. In the second case, the user will process data using a particular model, and assess the results to see if they are anatomically useful. In the third case, the user aims to learn something about the individual subject or patient, rather than about the model.

The flexibility of BodyBuilder for Biomechanics therefore allows the user to study individual subjects; to study a modelling method; or to develop new methods and test them against real life. The purpose of the work being carried out will affect every aspect of data processing and presentation, and for this reason it is only possible to use the tools available once the purpose has been thoroughly defined.

The process of designing and writing a model is fundamental to the use of BodyBuilder for Biomechanics. Even if the purpose of a particular study is to apply an existing model to a particular subject, it is useful to understand the way in which a model is developed.

Dividing the Body into Segments

Kinematic modelling is not limited to human bodies, and has been applied to animals and machinery; however, the following discussion refers only to human studies

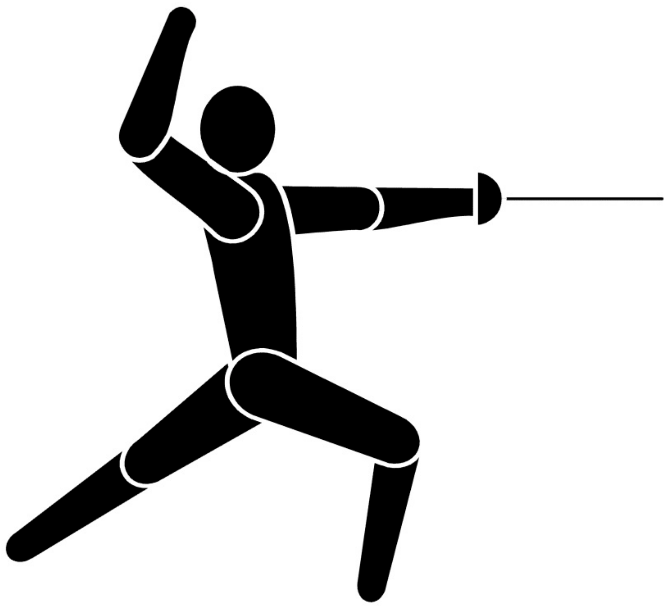
The design of a model begins with a need to describe the human body and its motion for a certain purpose. The purpose may be the study of an abnormality by comparison with normal individuals; the study of physical stresses and the avoidance of injury; the study of mechanical efficiency and its improvement; or the study of a sporting skill and the effects of various kinds of training and coaching. There are limitless possibilities, and each case requires an appropriate model.

The purpose will determine the way in which the body is modelled. The body must be divided into segments, which are assumed to behave as rigid elements, joined in certain ways. The number of segments and the overall complexity of the model will be determined by the purpose of the study. For example, if the purpose is to analyse power transmission at the knee joint in Olympic weightlifters, it is not necessary (and may be impossible) to measure and model the movement of each finger. However, if the purpose is to analyse repetitive movements in typing, each finger may need to be modelled as two or even three segments.

When designing a kinematic model, it may help to begin by making an outline drawing to show the segments and joints.

In the case of this simple 10-segment model, each limb is modelled in two segments, plus the body and head. This may be adequate for a gross description of large movements.

This diagram does not show exactly how each joint is modelled

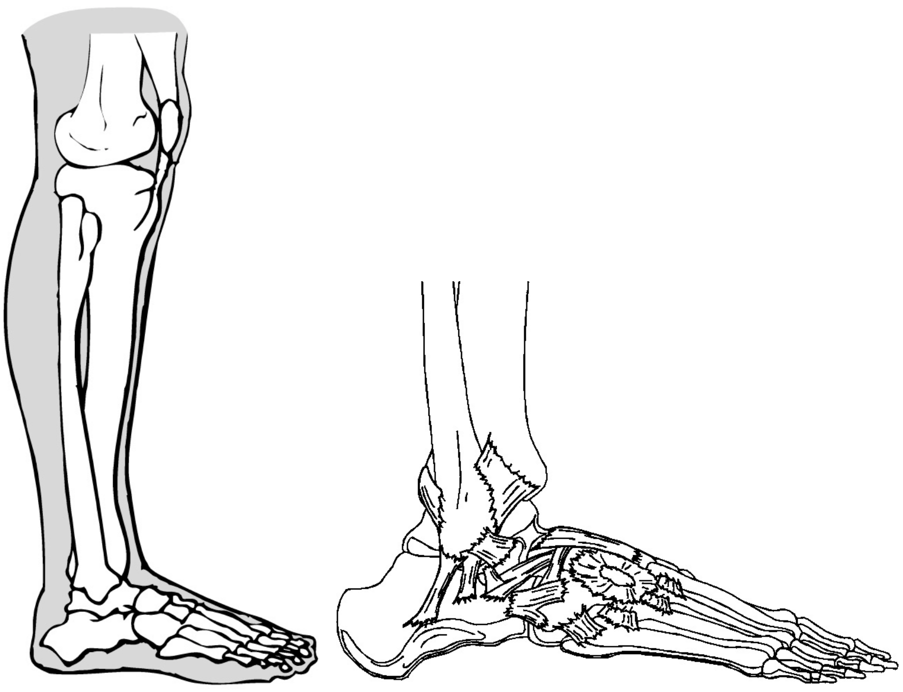


In most biomechanical studies, the division of the body into segments will begin with a study of anatomy. The concept of modelling the body as a number of linked, rigid segments is based on the anatomical fact that the skeleton is made up of rigid bones which are linked by various kinds of joint, whose characteristics are well known. The user must judge to degree to which this approach gives realistic results.

For example, a large bone, such as the femur, will lend itself easily to modelling as a single segment, but there may be some difficulty in finding marker locations which are not prone to substantial amounts of skin movement relative to the underlying bone. A complex body part such as the foot, although presenting fewer skin movement problems, contains many individual bones and compound joints, and will have to be modelled using approximations and simplifications. A body part such as the forearm may be simple to model as a single rigid segment, but it is more difficult to model the relative motion of the radius and ulna. The degree to which a model is realistic will depend on the validity of the approximations and simplifications used, judged within the context of the purpose for which the model is written.

In a more anatomically sophisticated model, it will be necessary to consider the relationship of actual bones to modelled segments; the degree of complexity required; the level of realism needed; what it is possible to measure using markers attached to the skin; and the significance of any kinetic quantities to be derived from the model.

The foot is an example of a body part which may be modelled with various levels of complexity. The degree of anatomical realism required depends on the purpose for which the results are to be used.



Approximations and Errors

Modelling the body as an assembly of rigid bodies linked by joints, and then measuring the motion of those segments using skin markers, is subject to inherent errors and approximations.

Firstly, if it is the aim of the measurement to establish the motion of the skeletal components concerned, there is an error associated with the movement of skin relative to bone. This can be reduced, but not eliminated, by careful choice of marker location. Some experiments may be required to find out the best places to locate markers in order to minimise the effects of skin movement.

Secondly, even if there were no skin movement, there would be some uncertainty involved in the relationship of the marker positions to the underlying skeletal structure and joints. This source of error may be reduced by choosing easily located anatomical points at which the bony structure can be found close to the skin, or by correcting for the uncertainty after the measurement by using adjustable parameters; but it is not possible to eliminate it completely.

Thirdly, the rigid-body concept is an approximation. Even the bones themselves are not perfectly rigid, and the joints contain elastic components such as ligament and cartilege, which can deform slightly under load. These are likely to be insignificant sources of error, though, compared to the movement of soft tissue. Given sensible choices of marker location, soft tissue movement should not be a source of large errors in kinematic studies (concerned with motion), but may be important in kinetic studies (concerned with forces).

Finally, the measurements of marker trajectories are themselves subject to error, and users should be aware of the extent to which these errors may be significant. For example, when calculating the moment about a joint such as the hip, the result is usually the product of a large force and a small distance. Position-measurement errors, although small in absolute terms, may be amplified when such quantities are calculated. Also, the measurement of the rotation of a long, thin segment about its long axis may be more sensitive to small positional errors because of the near-colinearity of the markers on the segment. Techniques to address this error include the use of markers on sticks.

Modelling the Joints

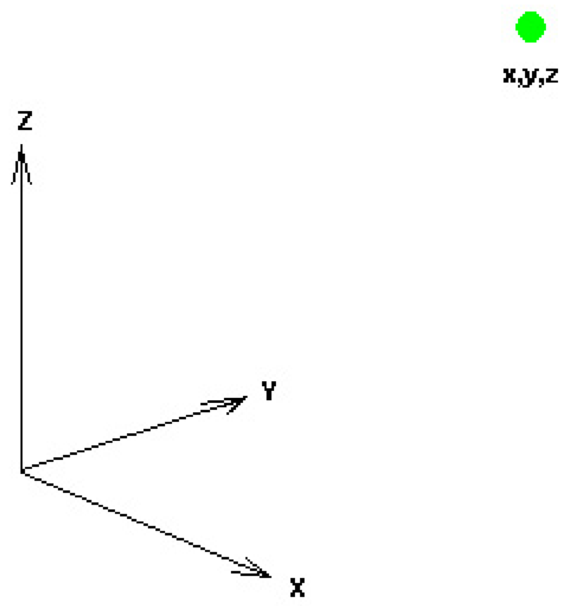
The joints between the segments can be modelled with various degrees of freedom. A joint such as the hip or shoulder, for example, being a ball-and-socket type of joint, can rotate about three axes, and in most kinematic models, it is necessary to measure all these degrees of freedom if the results are to be realistic and useful. The knee and elbow, however, may in some cases be modelled as simple hinge joints with only one degree of freedom. Although this is a simplification, it may be adequately realistic for many purposes. It is necessary to decide how many degrees of freedom to allow each joint, because this determines the number of markers needed to measure the movements concerned.

In most cases, modelling a joint involves estimating the position of an internal location, such as the centre of a ball-and-socket joint, or the axis of a hinge joint, using the measurements of external markers, combined with knowledge of anatomy. This may be as simple as placing a marker on an anatomical landmark and assuming that the internal point can be derived from that marker’s position by adding a certain distance in a specified direction. Alternatively, it may be a more complicated procedure involving several markers and detailed anatomical parameters measured by radiography. Any level of complexity is permitted, and the user must decide how sophisticated the model needs to be, if it is to be suitable for its purpose.

Points, Lines, Planes and Segments

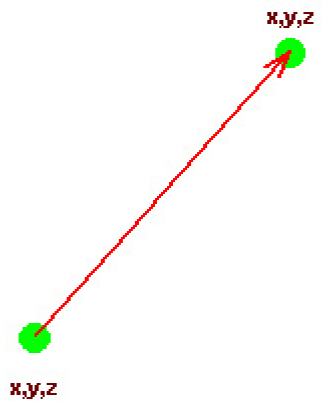
Having decided on the division of the body into segments, and the kind of joints between those segments, the user can begin to write the model and design the marker set which goes with it. At this stage, it is essential to have a clear idea of the number of markers required per segment.

Single Point



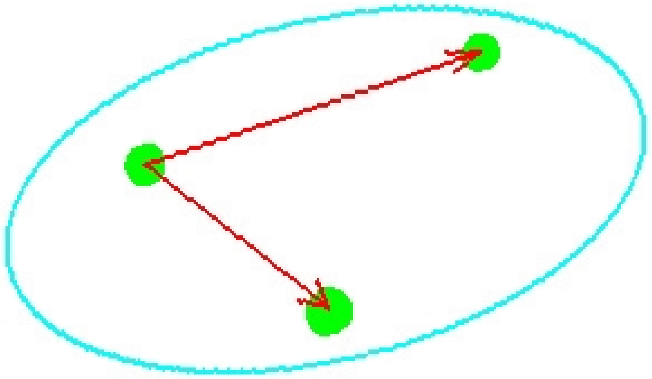
It is clear that the measurement of the position of a single marker (expressed as three co-ordinates, relative to a particular frame of reference), is not sufficient information to fix the location and orientation of a body segment. Even if we assume that we already know the size and shape of the segment, knowing the location of just one point belonging to it is not enough to determine its location and attitude completely.

Two Points



It is also obvious that the positions of two markers (six co-ordinates in total) defines a line. In fact, knowing the positions ot two markers gives us more than just a line; it gives us the length of the line (the distance between the two markers), the sense of direction (think of an arrow rather than a line), and the location of each end of the line, or any other point of interest along it. This, though, is still not enough to fix the location and attitude of a body segment.

Three Points



If we know the positions of three markers which all move together as part of a single body segment (a total of nine co-ordinates), we have enough information to define a plane. Any three points define a plane (unless they are co-linear). However, three points can define more than just a plane; they can define a segment in both position and orientation.

If one of the markers is nominated as the origin; the direction from that origin to another marker is used to define a first axis; a second axis is defined at right angles to the first (in the plane defined by all three markers together); and a third axis is defined at right angles to the plane; then we have defined a complete origin-and-axes system, with its location and attitude fully determined, but only on condition that the three markers are not co-linear.

Four Points

If we have the positions of four markers belonging to one segment, again on the condition that they are not co-linear, we have redundant information, which we can use for other purposes. For example, if one of the markers is temporarily obscured from view, we can use our knowledge of its relationship to the other three, to fill in the gap. This technique is very useful in kinematic modelling, especially in the case of complex motions where markers are often obscured from view. Alternatively, we could use the redundant information to average out errors due to measurement noise, skin movement and so on.

To summarise:

- one marker defines the location of one point, but not of a whole segment
- two markers define a line, with length and direction (ie. an arrow or vector)
- three non-colinear markers define a plane, and also a complete segment (with location and attitude)
- four markers provide extra information which can be useful for interpolating obscured markers or averaging out measurement noise

It is worth remembering that one or more of the points used to determine the location and attitude of a model segment may be derived from a neighbouring segment. In fact, if the positions and attitudes of the neighbouring segments are fully determined, and the degrees of freedom of the joints are defined, it may be possible to determine the location and attitude of a segment without any markers being attached to it. This technique is demonstrated in the model FOOT, supplied with BodyBuilder for Biomechanics.

BodyLanguage allows the user to define a segment using whatever marker locations are suitable in the circumstances. It also allows for the fact that if several markers are attached to one body segment, although in theory they should move together, in fact they will each individually be subject to skin movement and soft tissue deformation.

Choice of Origin and Axes

When defining segments, the model writer is free to choose any point within the segment to be the origin of that segment, and any set of mutually perpendicular directions as the axes. However, there are a few guidelines which simplify these choices.

The majority of models in biomechanics involve linked segments which form a chain, such as Pelvis - Femur - Tibia - Foot, and such models describe the movements of each segment relative to the more proximal segment. For example, a model might be written with the purpose of describing the movement of the tibia with respect to the femur, in terms of rotations about the knee joint.

In such models, it is necessary to define the location of a joint centre and axes of rotation. In the case of the knee, for example, a kinematic model might attempt to describe the movement of the tibia segment in terms of rotation about a knee flexion axis which is fixed in the femur.

In such cases, it is convenient to define the femur segment such that the knee joint centre is the origin of the femur segment. It may also be convenient to define the axes of the femur segment so that one of them coincides with the knee joint axis of rotation.

Generally, it is recommended to write segment definitions so that:

- the origin of the segment is the centre of rotation of the distal joint
- the first axis of the segment coincides with the line between the centres of rotation of the distal and proximal joints linking the segment to its neighbours (ie. it is the long axis of an elongated segment such as a limb segment)
- if compatible with the above, the second axis coincides with the most significant axis of rotation of the distal joint

While these are only guidelines and will not apply in all cases, certain

BodyLanguage functions (especially kinetic functions) assume that they are followed unless otherwise specified. If you decide not to follow these guidelines, pay particular attention to the calculation of force, moment and power quantities in your model.

These guidelines cannot be used in all cases, for example segments, such as the pelvis, which have more than one distal segments attached. The treatment of such segments is illustrated in the example model scripts supplied with BodyBuilder and discussed later in this manual.

Rotations

To describe the attitude of one segment relative to another, it is necessary to choose a mathematical convention for the description. For example, if we choose to model the elbow as a simple hinge joint, knowing that it can only flex within a limited angular range, all we need to choose is the position we call zero, and a sign convention to describe flexion and extension. However, if the model is more complex, we must use a more sophisticated definition. To describe ball-and-socket joints, we must find an unambiguous convention for describing a particular orientation using three angles.

The convention most relevant to biomechanics is called the Euler angle convention, after the mathematician Leonhard Euler. Euler angles correspond best to the concepts of flexion, abduction and rotation which are used in anatomy and biomechanics. The use of this approach to define rotations between segments is illustrated in all the example models supplied. A ball-and-socket joint capable of rotating about three different axes from any given position will be described using three Euler angles; a joint with fewer degrees of freedom (for example, a hinge joint) is equivalent to a ball-and-socket joint in which one or two of the angles is given a fixed value. It is a useful simplification to describe a joint using the minimum number of variables necessary for the model to fulfil its purpose. For example, if it is acceptable to describe the knee as a simple hinge joint with only flexion/extension movement permitted, ie. no valgus/varus or rotation movements, then this will simplify the model and reduce the number of markers required.

The characteristics of Euler angles, their full mathematical definition, and an explanation of the effect known as gimbal lock, are given in textbooks of mathematics and biomechanics, and will not be repeated here.

Overview

BodyLanguage is an interpreted programming language for defining kinematic and kinetic models of complex moving objects, such as the human body. BodyLanguage is designed to be used by non-programmers who have a reasonable grasp of geometry, but not necessarily of vector algebra.

BodyLanguage scripts are read by VICON BodyBuilder and used to process motion data, generating virtual trajectories, kinematic angles, moment and power quantities, and other outputs of interest in Biomechanics.

This section, inevitably, resembles a mathematics textbook; however, every effort has been made to express meaning as clearly as possible and to illustrate each technique with diagrams and examples.

Kinematic Modelling with BodyLanguage

BodyBuilder can generate output files which contain the locations and relative orientations of body and limb segments, calculated on the basis of the user-defined relationships of rotation centres to physical markers.

The user-defined model is contained in a BodyLanguage text file with the extension .MOD, also referred to as a script, in which the relationships between physical and virtual points are stated in general terms. Numerical constants are contained in a separate text file with the extension .MP, so that these parameters can be adjusted without changing the general properties of the model itself.

Before Starting

The first step in using motion capture for computer animation is to decide on the kinematic model to be used. This means choosing the number and type of body segments which are to be defined and measured, and the relationships between them. It also involves deciding the locations of markers to be placed on the body of the subject or patient, whose motion is to be captured; and writing down, in a BodyLanguage script, the relationship between those real points and the kinematic model body segments which are subsequently calculated.

Writing the Model Script

Models are written using BodyLanguage, which has been specially designed to make kinematic and kinetic modelling as clear and concise as possible. Complex mathematical operations can be accomplished with plain, one-word functions. It is perfectly possible, using the information given in this manual, to write a biomechanical model script from the beginning, and several examples of such models are supplied. In addition, Oxford Metrics encourages the exchange of model scripts between users, and will circulate scripts with permission.

However, rather than writing a new model from scratch, it is often quicker to start with an existing model and modify it to suit a new purpose. Often, parts of one model can be copied into another. BodyBuilder makes this easy by providing a model editing window in which changes can be made, and immediately applied to experimental data to see if the intended effect has been obtained.

Verifying Model Scripts

When a model has been written, it is necessary to test it with experimental data to verify that the outputs are suitable for their purpose. This may mean making an assessment of whether the assumptions made about body segments and joints are realistic enough to give meaningful results, and assessing the significance of various kinds of error.

A wide variety of models have been used in clinical gait analysis, the purpose of which is to describe the subject's motion while walking, with sufficient detail and accuracy to provide the basis for diagnostic and treatment decisions. Obviously, any model which is to be used for a purpose of such importance should be thoroughly verified by a properly controlled study, before it is used routinely for assessing patients.

Using Kinetic Data: Terms and Definitions

BodyBuilder version 3.5 implements a range of new extensions to BodyLanguage, which allow segments to be assigned mass and inertia, and which allow moment and power values to be calculated from force plate data, if available. All popular makes of force plate are supported: Kistler, AMTI and Bertec.

The interpretation of kinetic quantities calculated by modelling is beyond the scope of this manual. Also, the laws of motion and the general principles of biomechanics are covered in textbooks and will not be repeated here. However, the following basic terms and definitions, which are used in BodyLanguage, may be useful.

Force

Force is a vector quantity, ie. one which has a magnitude and a direction. Forces acting on a rigid body may be added and subtracted to each other to obtain the net force on the body. All forces may be broken down into three components which are parallel to any given set of mutually perpendicular axes. The total, or net, force acting on a body is the sum of the different forces acting on it at any given time, such as gravity, ground reaction forces, and forces from other bodies linked to the body being considered. If the net force acting on a body is zero, the body will remain at rest or move at a constant velocity. If the net force on a body is non-zero, the body will be accelerated. Some forces can be measured directly (eg. weight, ground reaction forces), others (eg. muscular forces) can be estimated using appropriate kinetic models. The SI unit of force is the Newton (N). Weight is an example of a force; it is the effect of gravity on mass. A mass of 1kg has a weight of about 9.8N. Note that although a force may be fully specified by a magnitude and a direction as far as the strict definition is concerned, the needs of kinetic modelling may require additional information such as the point of application of a force.

Moment

Moment is also a vector quantity. It describes the ability of a force to make a body rotate. It is the (vector) product of the tangential component of force acting on a body, and the distance separating the line of action of the force from a defined reference point. Moment can also be broken down into three components. For example, the vertical component of a moment is the part which tends to make the body rotate around a vertical axis. The SI unit of moment is the Newton-metre (Nm); in biomechanics, the most widely used unit is the Newton-millimetre (Nmm). 1Nm = 1000Nmm. In BodyBuilder, it is assumed that the unit of moment is Nmm.

Reaction

In mechanics, a reaction is a kind of force. When a body presses against another, it experiences an equal and opposite force from that body. This is the reaction. In biomechanics, the term is normally used to describe the force exerted by the ground on the foot (or other body part) in contact with it.
Ground reaction force vectors can be directly measured by force plates. Strictly speaking, a reaction is a three-component force, but the quantity measured by a force plate has an additional three moment components as well, which are measured round a calibrated forceplate centre.
In BodyLanguage, a "Reaction" groups together a force and a moment with the moment reference point, around which the force is deemed to act, as a single convenient unit.

Work and Energy

Work is done when a force moves through a distance. If the movement is in the direction of the force, as in a concentric muscle contraction, the body applying the force is said to have done work (ie. the work done is a positive number). If the movement is against the direction of the force, as in an eccentric contraction, the body applying the force is said to have work done upon it (ie. the work done is a negative number). Work is a scalar quantity and cannot be broken down into components. The SI unit of work is the Newton-metre or Joule (J). Energy is the ability to do work and has the same units. Energy is also a scalar.

Power

Power is the rate of doing work. The power generated by a force acting in a straight line is the product of the force and its velocity. The power of a force causing a rotation is the product of the moment and its angular velocity. Power is a scalar quantity, but sometimes in biomechanics it is treated as a pseudo-vector by calculating power separately for each component of a force or moment. The SI unit of power is the Joule per second (J/s) or Watt (W). In BodyBuilder, the milliwatt is the normal unit (mW). Ground reaction forces, however strong, do not move through a distance, and thus do no work and generate zero power.

Moment of Inertia

Inertia is the tendency of a body to maintain a state of rest or constant motion. It is effectively identical to mass. Moment of inertia is the tendency of a body to maintain a state of rest or constant rotation about a specified axis. It is a scalar quantity which depends on the mass of a body, the way the mass is distributed, and the chosen axis of rotation. The SI unit of moment of inertia is kgm2. Radius of gyration k is a related quantity, defined as follows: if a body of mass M has a certain moment of inertia about a specified axis, a point-like body of mass M placed a distance k from that axis would have the same moment of inertia. Radius of gyration is a distance and in biomechanics is generally measured in millimetres.

The definitions, calculation, measurement, and inter-relationships of these and other quantities are given in textbooks of biomechanics. The user should take care to use consistent sign conventions for quantities such as power, which can be either generated or absorbed by a muscle. When power is transmitted across a joint, one segment does work on the other and it is a matter of convention and preference which is considered to be positive.

Applying the Model

In most cases, the results obtained will depend not only on the anatomical assumptions made, and the model script used, but on the care with which markers are placed on the subject prior to measurement.

Data collection, reconstruction and labelling is a function of VICON Workstation and is described in the documentation supplied with that equipment. The labels used to identify reconstructed trajectories should match those mentioned in the model script.

The editing functions in BodyBuilder allow the data to be manipulated prior to applying the model. For example, short breaks in trajectories can be interpolated; noisy trajectories can be filtered; trajectories can be created, changed and deleted. Clearly these features should be used with care if the results are to reflect real motion rather than manipulation. Users should be aware of the difference between "cleaning up" data and changing it.

Displaying the Results

The output of BodyBuilder may take the form of a text (ASCII) file, listing the locations of various points and segments, and the rotations between certain segments as required. Alternatively, these quantities can be stored in the more compact C3D format, and displayed graphically using Reporter or Polygon Authoring Tool.

Polygon and Reporter allows the user to open several C3D files at once, and to display their contents in graphs. Graphs can be laid out on the page according to preference, and the format can be saved and re-used once it has been defined. Reports can be printed out or saved on disk. Reports may include text as well as graphs and three dimensional visualisations in the workspace.

The results of modelling may also be viewed directly in the Workspace window of BodyBuilder. In many models, virtual points are created for the sole purpose of making segments more easily visible in the Workspace. If the virtual points created are listed in the marker label list file, and stick figure links are defined between them, the segment can be given a natural appearance, which is often enhanced when the display of points is turned off and only the stick figures remain. Virtual markers and display sets listed in the marker file can also be displayed with Polygon. Examples of this technique are given in the sample scripts which are supplied with BodyBuilder for Biomechanics.

Types of Object

BodyLanguage models are built up from 5 types of object:

- numbers
- points
- segments
- rotations
- reactions

Numbers

Numbers are the simplest type of object, from which all other objects are constructed. Numbers can be variable (such as the height of a point) or constant (such as a body measurement like knee width). An important use of constant numbers is for the physical parameters of a measured subject.

Points

Points are locations in space, specified by Cartesian co-ordinates. Each point has three ordered components, corresponding to the X, Y and Z coordinates of the location in space, so a point may also be considered as a vector. The co-ordinates may be specified relative to the overall laboratory frame of reference, in which case the point is described as global, or relative to a particular segment, in which case it is described as local.

There are two kinds of point, real and virtual. Real points, derived from physical markers, are the measured locations of reflective spheres attached to the subject. Virtual points, created by modelling or other data manipulation techniques, have all the properties of real points, but may be in locations where no actual reflective marker existed. Virtual points can be used:

- to specify the location of internal body parts such as joint centres
- to build a mesh of surface points for a more realistic display of the subject
- to re-create, by interpolation or estimation, the location of a real marker which becomes obscured during part of the motion capture

Although a virtual point may not describe the position of a physical marker, the terms "point" and "marker" are often used interchangeably.

Segments

A segment is defined by a group of points, both real and virtual, which move together. A segment can move and rotate with respect to the global frame of reference.

Each segment has a single point which is defined as the segment origin, and three segment axes. The orientation of the segment is described by rotations about the origin, relative to the global frame of reference.

A segment is generally used to describe the position and orientation of a body part, such as the head, the pelvis, or a limb segment, which can be considered as a rigid body which moves relative to neighbouring body segments (although, as human body parts are not perfectly rigid, allowance is made for the fact that markers placed on a body segment may move a little relative to each other).

Rotations

A rotation is a quantity with three ordered components, representing angular movement about 3 specified axes. These ordered angles are called Euler angles.

A rotation may describe the orientation of a single segment with respect to the global frame of reference, or the relative orientation of two segments (whether or not they are linked by a joint).

Reactions

A reaction groups a Force, and a related moment a reference point together, all of which are represented as three dimensional quantities.

Reactions are used to describe the kinetic interactions between segments, and between segments and forceplates. See the kinetic modelling section fopr more details.

Object Names

Every object has a name, composed of up to 256 letters and numbers, plus some additional characters such as % and $. There must be no spaces in an object name. Object names must not start with a number.

The names of real points in the input .C3D file are determined in other parts of the VICON system (after reconstruction, when trajectories are given labels) and must match exactly the names used in the .MOD file. For example, if your BodyLanguage script (.MOD file) uses a point called "KNEE", the input .C3D data file should contain a trajectory named "KNEE". (If the trajectory of the relevant marker in the .C3D file has been given a different label, you can either use BodyBuilder to re-label the trajectory, or edit the script to use the different name.)

BodyLanguage is case-insensitive, meaning that it ignores the difference between upper and lower case characters in an object name. For example, LeftKnee and leftknee are recognised as the same object. To make a long object name more readable, upper case letters may be used in the middle of a name (for example, LeftElbowJointCentre).

Limitations of Object Names

- It is not generally possible to identify the type of an object simply be looking at its name.

- Although in general, object names may have up to 256 characters, there is a limit of sixteen characters on the length of trajectory labels in a .C3D file.

- If you use BodyLanguage to create an object (such as a virtual point) with the intention of saving it in a .C3D file, it should be given a name of no more than sixteen characters.

Reserved Names

The following reserved names cannot be used for objects, since they are reserved for special functions (see section on Functions):

ACOS
ALIGN
AND
ASIN
ATAN
ATAN2
ATTITUDE
AVERAGE
BODYMASS
CHORD
COMP
CONNECT
COS
DEADBAND
DEFINE
DIST
DISTANCETHRESHOLD
ELSE
ELSIF
ENDIF
ENDMACRO
EXIST
FIELD
FORCETHRESHOLD
GRAVITY
IF
LAST
MACRO
MOMENT
NORM
NOT
O (letter "oh")
OPTIONALFORCES
OPTIONALMOMENTS
OPTIONALNUMBERS
OPTIONALPOINTS
OPTIONALPOWERS
OPTIONALREACTIONS

OR
OUTPUT
PARAM
PERP
POWER
REACTION
ROT
SAMPLE
SIN
SINGULAR
SPINE
SQRT
TAN
THEN
VELOCITYTHRESHOLD

Object Assignment and Type Definition

A BodyLanguage model contains assignments by which an object is defined or modified. Every assignment takes the following form:

ObjectName = Expression

where Expression is made up of a combination of other objects, using operators, functions, and brackets ( ). Brackets can be nested ((.(.).).).

The type of an object (number, point, segment, or rotation) is determined by its first assignment in the model, called the "type definition". The type of the expression on the right-hand side of this first assignment fixes the type of the new object on the left-hand side. At every subsequent appearance of an object, its type must remain the same.

New lines have no structural meaning in BodyLanguage and are used only to make the script easier to follow.

Inputs

Since BodyLanguage processes VICON motion data, its variable inputs are real points from VICON three-dimensional data files (.C3D), which move sample-by-sample. Each point used in the model must be referred to by the same name in BodyLanguage as the label attached to it in the .C3D file.

Other fixed inputs, known as "parameters", are constant for an individual subject and may either have been typed in or calculated from a static trial.

• numbers, points, and segments	from parameter file
• points	real or virtual trajectories, read from a C3D file

Optional Points

All objects referenced in a BodyLanguage script must be defined. For an object to be defined, it must either be:

- set to a constant, using one or more numbers
- present in the list of labels of the input C3D file
- calculable from other, previously defined, objects

Any other object found in a script is assumed to be an error (for example a typing mistake) and is flagged when the model is run.

There is one exception to this rule. It is often convenient to write a script that is able to run a different part of a model according to which labelled trajectories are present in the C3D file. Points which may or may not be present in the input file must be listed as optional, using the OptionalPoints() statement. See later section on Non-Assigning Statements.

Objects may be defined but not exist in the current sample (for example if a measured point has not been reconstructed in the current sample).

The OptionalNumbers() statement works in the same way as Optional Points, but declares a number of names as corresponding to scalar values. Other types may also be defined using OptionalForces(), OptionalMoments() etc.

Outputs

The purpose of a BodyLanguage model is to generate outputs:

• numbers, points, and segments	parameter (.MP) file (from static trials)
• virtual points	as .C3D trajectories (eg. joint centres)
• segment rotations	.C3D (Euler angles)
• joint rotations	.C3D (Euler angles)

The outputs, if appropriate, may also be written to .ASF or .AMC file types. The non-assigning statements PARAM() and OUTPUT() are used to generate outputs. (See later section on Non-Assigning Statements).

To optimise BodyLanguage execution speed, the sample-by-sample values of an object are not evaluated unless it either appears directly in an OUTPUT() or PARAM() statement, or are used in the definition of another object which appears in an OUTPUT() or PARAM() statement.

Parameters

A BodyLanguage script is divided into two parts, a parameter section (.MP file) and model section (.MOD file), which are loaded, edited, and saved separately.

The Parameter section is intended to provide subject specific information that is required as input to the model, such as knee width, for example.

When the model is run, the two sections are linked together for execution, with the parameter section heading the model section.

A model can be run with any parameter section which contains the required definitions. Models can also be run with a blank parameter file, but an .mp file must be specified.

BodyLanguage scripts are often divided into two types, "static" and "dynamic", according to the output they generate. Some models however do not need a special static trial.

A static, or calibration, script generates a set of parameter values that are needed for running the main dynamic model. New parameters are appended at the end of the current parameter section. Assignments to existing parameters cause the existing parameter value to be overwritten.

The main dynamic model calculates values required from the moving dynamic trial.

Although both types may be combined in a single script, it may simplify matters if they are kept separate.

Samples

BodyLanguage differs from programming languages in one fundamental respect.

The input to a BodyLanguage model is always a series of sampled data points, equally spaced in time. In a conventional program, there would need to be a "main loop" containing the instructions:

```
set first sample number
read sampled data from file
apply model to data in current sample
test for last sample number
loop back to read next sample
```

With BodyLanguage, this loop is assumed. The model is written simply as it is to be applied to each sample in turn. No explicit loop instructions are needed.

However, it is still possible to create expressions which depend on the sample number (SAMPLE function - see later section on Number Functions), or which use the values of objects from samples other than the current one (sample offset function -see later section on Special Functions).

Types of Expression

An expression is a combination of objects, operators and functions by which a new object is defined. Every assignment statement (by which objects are defined or modified) takes the following form:

ObjectName = Expression

The expression in every assignment in a script must match the type of the object being assigned. (If it is the first assignment of that object, it will define the type of the object). The type of an expression is determined as follows:

Numbers

A number expression can be:

- a number input
- a single integer or real constant (in the model script or parameter list)
- a valid combination of numbers, number operations, and number functions

Points

A point describes a position in space. The same point can be described either:

- in terms of its position relative to the global reference frame of the VICON Workstation measurements
- as a local position relative to the origin of the segment of which it is a member.

Although it is possible to use the same name for a point in both circumstances, it is recommended that, when a local definition of a point is used, its name is prefixed with a "%" character, eg. LSHO (global) and %LSHO (local).

A pair of braces {} is used to construct a point from 3 numbers. A point expression can be:

- a point input
- numbers - {numberA,numberB,numberC}
- a valid combination of points, point operations, and point functions

Segments

A segment is a group of points which remain at fixed distances from one another. As a segment moves, all the points in it move together. A segment can move and rotate with respect to the global VICON Workstation frame of reference. Strictly speaking, a segment expression does not require that the points referred to are absolutely, rigidly fixed relative to each other - since human body parts are not mechanically rigid, there must be some flexibility.

However, segment expressions should include points which move together, at least approximately.

If the global positions of at least 3 points in a segment, (which are not in a straight line) are known, the segment's position (ie. the position of the segment origin), and orientation, can be determined. Any other local points in the segment can then be output as global, virtual points (see section on Operators). This is the method of locating points, such as joint centres, where a real marker cannot be fixed.

A pair of square brackets [ ] is used to construct a segment from a set of points. A segment expression can be:

- a point and two lines in square [ ] brackets

- a valid combination of segments, segment operations, and segment functions

Segment Definition

In a segment definition, one global point defines the segment origin. (If there is no real point at the desired location of the segment origin, the segment is first defined with a different origin, and then moved to the required position).

Up to 4 other global points are used to create two defining lines which together uniquely describe the segment rotation. Rotation of a segment is represented by the orientation of its 3 axes, pointing in mutually perpendicular directions from the segment origin.

The point that defines the segment origin can also be used in the definition of either defining line. Defining lines may also share a common point, so a minimum of 3 global points are sufficient to define a segment.

The global axes X, Y and Z, are established by the VICON Workstation calibration. (Although they can be manipulated in BodyBuilder, for the purposes of this document, they will be taken as fixed). Co-ordinates of global points are an ordered triplet of numbers, in the order X co-ordinate, Y co-ordinate, Z co-ordinate. Segment axes, to avoid confusion with global axes, are referred to as 1, 2 and 3, rather than X, Y and Z. Local co-ordinates are an ordered triplet of numbers, in the order 1-axis, 2-axis, 3-axis. When a segment is defined, it is possible to control which axis has which label, as long as the right-hand rule is followed.

One segment axis is set parallel to, or coincident with, the first defining line (depending whether the segment origin lies on the first defining line).

A second segment axis is in a direction perpendicular both the first and second defining lines. The positive direction of the second segment axis is towards the tip of an imaginary corkscrew whose handle turns, by the shortest angle, from the direction of the second defining line onto the first.

A third segment axis is perpendicular to the first two segment axes, and the positive direction is determined using the right-hand rule.

The order in which the global points are entered in the definition of a segment is critical. If it is intended that:

- the origin is defined by pointI
- the first defining line is (pointK-pointJ)   (from J to K)
- the second defining line is(pointM-pointL) (from L to M)

then the segment definition expression should be:

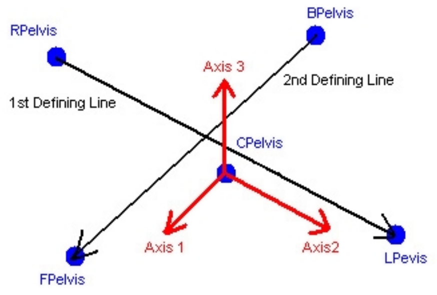
[pointI,(pointK-pointJ),(pointM-pointL),1]

The 1 at the end of the definition expression contains two pieces of information. Its value (1) indicates that the first defining line coincides with (or is parallel to) segment axis 1. Its sign (positive) indicates that the segment axis constructed to be perpendicular to both the first and second defining lines is axis 2 (one more than axis 1). The other axis is, therefore, labelled 3. A token may be substituted for the number as shown in the following table.

The permitted values for this number are:

Value	1st line defines the axis labelled	1stX2nd lines define the axis labelled	Token alternative
1	1	2	xyz
-1	1	3	xzy
2	2	3	yzx
-2	2	1	yxz
3	3	1	zxy
-3	3	2	zyx

Example



The diagram shows the segment defined by

```
Pelvis =[CPelvis,LPelvis-RPelvis,FPelvis-BPelvis,yzx]
```

The virtual point CPelvis is the origin. The first defining line is from RPelvis to LPelvis. The second defining line is from BPelvis to FPelvis. The token yzx means that the axis parallel to the first defining line is Axis 2; Thus Axis 2 is parallel with the first defining line but originates from the origin at CPelvis. Axis 3 is perpendicular to both defining lines, and its direction is determined by the right hand screw rule. Axis 1 is, then, perpendicular to the other two axes, such that the three axes form a right-handed frame. It is therefore not necessarily parallel to the 2nd defining line.

This example uses five points to define a segment, since the defining lines have no common points, and the origin is not part of a defining line definition.

Segment Definition across Hinge Joints

It is often convenient to define one of the axes of a segment to be aligned with the axis of a hinge joint. Using this form of definition limits the degrees of freedom at the joint concerned and allows the segments to be defined using fewer markers than if they were independently defined using separate markers. This approach is justified if it is acceptable for the purpose of the model, to make the simplifying assumption that a joint such as the knee or elbow can only move in flexion. This is equivalent to assuming that there is no valgus/varus movement. Although this technique prevents the two segments moving independently, it is not the same as linking the segments in a hierarchy.

For example, the thigh and lower leg segments on either side of a knee-joint can be defined in terms of just 3 points: hip, knee, and ankle:

```
LFemur=[LKnee,LHip-LKnee,LAnkle-LKnee,-3]
LTibia=[LAnkle,LKnee-LAnkle,LHip-LKnee,-3]
```

where, in this example, LFemur is the left thigh segment and LTibia is the left shank segment. However, two problems arise if the knee hyperextends (goes beyond straight).

First, while the leg is close to straight, the three points used to define the segment are almost co-linear. In the unlikely case that the three points are precisely co-linear, the segment definition is not valid. However, this is so unlikely that it can be disregarded. The real problem is that when the three markers are very close to being co-linear, small variations in the relative positions of these points can cause the direction of the knee axis to swing wildy.

Second, when the knee goes beyond straight, the positive direction of the knee flexion axis reverses and the segments "flip" by 180 degrees.

Singularities and Dead Band

To overcome wild swings in the knee axis when the leg is almost straight, a "deadband" region can be defined in which the direction of the axis is not calculated directly from the segment definition. Instead, the modeller automatically checks the attitudes of the segment on either side of the fields for which the axis lies in the deadband, and creates a smooth transition between them.

If required, this automatic action by the model can be tested by using the special logical function SINGULAR(segmentP). This function is true if segment P is in the dead band. For example, the statement

```
IF SINGULAR(Tibia) DEAD={1,1,1}
```

(followed by an OUTPUT statement) would create a point labelled DEAD at the location 1,1,1 in each field in which the segment Tibia is in the dead band.

The default value for the dead band is 1∞ on either side of straight. For human movement analysis, unless marker positions are adjusted with great care, this value is often too small to eliminate axis swings. A dead band value can be set by including the following assignment anywhere in the script, either as part of the model itself, or more conveniently as a parameter.

```
DeadBand=numberA
```

where numberA is the half-width of the dead band in degrees.

In static trials, if the arms and legs are stretched out, it is possible for the knees and elbows to lie in the dead band all the time, causing the expression to be invalid throughout the trial. In this case it is necessary to set a low value of dead band, or to comment out the line altogether. In static trials, the problem of axis swing is irrelevant, so a dead band is not required.

Anti-Flip

In addition to the problem of defining lines which are almost co-linear, which is addressed using the dead band, there is another potential problem which may arise if (for example) a knee hyperextends slightly (goes past the straight position). Because the shortest angle from the second defining line to the first is now in front of the knee, the knee extension axis will appear to flip around so that it points inward rather than outward. To overcome the flip problem, an optional argument can be added to a segment definition, in the form of an anti-flip line.

To understand the use of the anti-flip line, visualise the 2 segment defining lines projected into a plane to which they are both parallel. Next, project the anti-flip line into the same plane. Now move these three vectors in the plane, without changing their directions, until they originate at the same point.

The segment definition process involves rotating the second defining line, projected into this plane, onto the first, through the shortest angle. The second axis of the segment is defined, using this rotation, by a right-hand screw rule. If, during this rotation, the second defining line passes through the projected anti-flip line, the direction of the second axis is automatically reversed. This has the effect of maintaining the general orientation of the axis, even if the two defining lines move either side of the dead band.

For example, a thigh, or femur, segment is conveniently defined using the lines between hip, knee, and ankle:

LFemur=[LKNE,LHIP-LKNE,LANK-LKNE,zyx]

This works well until the knee becomes hyperextended, as when the ankle moves in front of the hip-knee line. Without an anti-flip line, the direction of the shortest rotation of the second defining line (LANK-LKNE) onto the first (LHIP-LKNE) is then reversed causing the horizontal axis of the segment to flip.

However, if the anti-flip line (LTOE-LKNE) is added:

LFemur=[LKNE,LHIP-LKNE,LANK-LKNE,zyx,LTOE-LKNE]

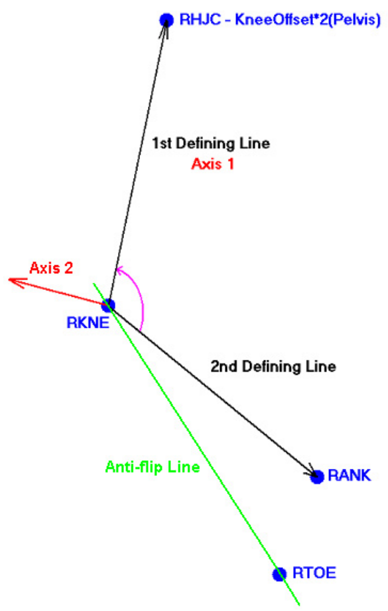
then, if the knee is hyperextended, the second axis rotation passes through the projection of the line (LTOE-LKNE) and the reversal of the horizontal axis directions is automatically negated.

Example

The following segment definition is taken from the model script FULLBODY.MOD which is listed fully in a later section.

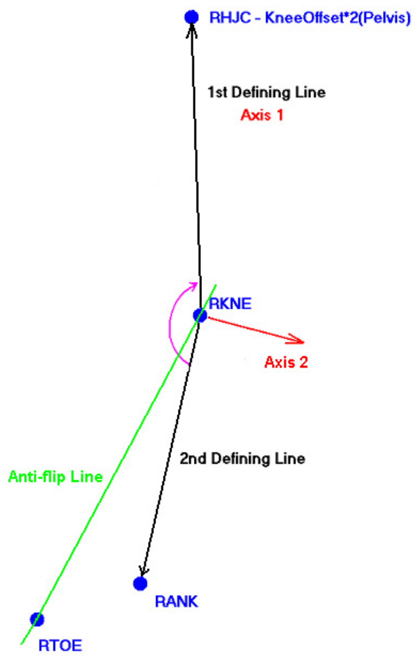
RFemur=[RKNE,RHJC-KneeOffset*2(Pelvis)-
RKNE,RANK-RKNE,zyx,RTOE-RKNE]

The name of the segment defined in this line is RFemur. The expression in square brackets consists of a real point (RKNE) and two defining lines, along with the token zyx and the anti-flip line RTOE-RKNE. The first defining line is from the real point RKNE to the virtual point RHJC-KneeOffset*2(Pelvis), which is to the right of the estimated position of the right hip joint centre, by an amount equal to half the width of the knee joint (the knee offset is a number read from the parameter file). The reason for this complex line definition is that the virtual point RKJC at the knee joint centre has not yet itself been defined, so cannot be used in the simpler definition RHJC-RKJC. The line established instead is closely parallel to RHJC-RKJC but shifted to the right. The second defining line is simply RANK-RKNE, and the anti-flip line is from RKNE to the point RTOE. The token zyx ensures that the first defining line establishes axis 3, and the axis established by the right hand screw rule is axis 2. In the normal situation in which the knee is flexed, the construction is as shown over page.



In this diagram, the plane of the paper is the plane containing the two defining lines; the projection of the point RTOE is also shown. The curved arrow shows the shortest rotation bringing the second defining line onto the position of the first. This causes axis 2 to be aligned out of the paper.

If the knee hyperextends, the construction would be as shown:



In this case, the shortest rotation of the second defining line passes through the anti-flip line (green line), reversing the right-hand screw rule, and therefore causing axis 2 to be aligned into the paper. The anti-flip line will prevent this from occurring and realign axis 2 out from the paper.

If a dead band is defined, then in the case of the two defining lines being co-linear (or close to co-linear), the position of axis 2 will be interpolated from better-defined positions either side of the frames in which the leg is straight.

Hierarchical Skeletons

It is not necessary to specify the interconnections of segments in a kinematic model. Any number of segments may be independently defined using separate sets of three or more markers, and the movement of those segments will be completely independent. In some situations, this may be appropriate and necessary. For example, it is possible to measure the movements of two subjects who are not connected to each other and whose movements are fully independent. Alternatively, it is possible to model two indirectly linked body parts of the same subject (such as the two feet), without modelling all the segments which form the bridge between them.

However, in many models, especially those which use the kinetic functions, it is necessary for segments to be linked in specified ways, corresponding to the actual physical joints between the body parts they represent. In such a model, there is a hierarchy of segments beginning with a root segment, from which chains of other segments branch out. For example, the pelvis may be defined as a root segment, with the legs and upper body represented by chains branching out from it. Each segment in such a chain is called a child segment, and must have a specified parent segment (which may be the root segment, or another segment one level closer to the root). A segment may have only one parent, but may have more than one child. In the example of a pelvis root segment, the left and right thigh segments will both be child segments of the root. No closed loops are permitted.

Any segment definition which does not state the name of a parent segment is interpreted as a root segment definition. There is no limit to the number of root segments which may be defined in one model script (if the hierarchical scheme is not used, every segment is a root segment). The choice of root segment does not correspond to any special anatomical significance. Normally, if a simple diagram is drawn showing the segments used in a particular model, one segment will suggest itself as the root.

The naming of a parent for a particular segment does not itself define the kind of joint between the two; their positions can still be measured independently, and the distance between them is not fixed by their hierarchical relationship. In some cases it may be desirable to limit the degrees of freedom between the parent and the child, but this is controlled entirely by the kinematic part of the segment definition (the choice of origin and defining lines) not by the hierarchy. However, segments which are defined as parent and child will be considered to be physically linked, so that forces can act directly between the two.

Each child segment is defined by an expression which states the name of the parent to which it is attached, and the connection point in the child segment. The segment definition statement syntax is:

segmentS=[pointI,(pointK-pointJ),(pointM-pointL),xyz, pointA , parent , pointC]

where the first five arguments in the expression are the origin, defining lines, token, and antiflip point; parent is the name of the parent segment, which must have been defined earlier in the script; and pointC is the connection point in the child, defined in global space. If the connection point is not specified, the connection will be at the origin of the parent segment.

A hierarchical definition may be made by reference to a previously defined segment, using an assignment statement which adds a parent and attachment point to an existing "free" segment

segmentS=[segmentS,parent,pointC]

This statement would replace the previous definition of segmentS by extending it to include the parent and attachment point. Alternatively, a previous segment definition may be copied, with the addition of a parent and attachment point:

segmentS=[segmentR,parent,pointC]

The component function is described later

The attachment point of a segment may be accessed using the component function with the value 4:

Attachment=segmentS(4) or 4(segmentS)

The IF ELSE block is described later

The definition of segment parents may not be made inside IF ELSE blocks, preventing the definition of a hierarchy that may change from one frame to another.

Rotations of Segments

A rotation refers to either:

- the orientation of a segment relative to the Workstation global space
- the orientation of one segment relative to another (joint angles)

Unless otherwise specified, rotations are saved as 3 Cardan (Euler) angles.

Rotation Definition

A special pair of brackets <> is used to define a rotation. A rotation expression can be:

- one segment in <> brackets, plus a token, eg. <segmentP,xyz>

- two segments in <> brackets, plus a token, eg. <segmentC,segmentP,xyz>

If one segment and a token are enclosed in the <> brackets, the segment rotation is global. If two segments are enclosed in the <> brackets, the rotation is relative between the segments.

The first segment in the <> brackets is the child, or moving segment, whose orientation is to be described relative to the second segment (or global frame of reference). The second segment, if present, is the parent or fixed segment.

In addition to the segment(s), a rotation definition includes a token of the form "xyz", which defines the order in which the angles are calculated. If the token is omitted, the default is "xyz".

The rotation is expressed as angles in degress about the fixed axes of the second segment (or global frame if no second segment is supplied). The order of the rotations is the same as that given by the axis token.

For a given rotation between two segments, there are twelve possible sets of Euler angles. BodyLanguage specifically performs a "fixed axis" calculation where the three angles are calculated in the given order around the fixed axes of the parent segment.
However manipulation of the parameters passed to this routine allows any of the floating axes possibilities to be used. This is described below.

Helical Vectors

An alternative to Euler angles is the Helical Vector representation. If the integer 0 (zero) is placed in the rotation definition instead of the token, a helical vector is calculated.

The Helical axis is an alternative to the description of joint orientation or movement in terms of an ordered sequence of three rotations. This axis is defined by a unit direction vector 'N', about which one segment embedded co-ordinate system is thought to be rotated through the helical, scalar rotation 'theta' radians, in order to reach its current attitude with reference to another segment embedded co-ordinate system, these two segment systems comprising the joint under investigation.

Herman Woltring proposed (see ref. below), for the purposes of describing a particular attitude, that this vector and rotation could be decomposed into orthogonal components in either body segment's co-ordinate system, through the definition of an attitude `vector', THETA = theta * N.
[Woltring, H.J. (1994) 3-D attitude representation of human joints: A standardization proposal. J. Biomechanics 27, 1399-1414.]

'THETA' here is a scaled vector created through the multiplication of the direction vector of the helical axis with the magnitude of the rotation about this axis. This vector ends up having a length of 'theta' (in radians) and a unit direction vector of 'N'.

This attitude vector 'THETA' is the output of the 'helical' function within BodyBuilder, given in units of radians.

This option is provided primarily for users who will make use of helical vectors in their own software.

Global Rotations

A global rotation is a special case where one of the segments is aligned with the global frame of reference. In any expression involving segments, the global unit segment

[{0,0,0},{1,0,0},{0,0,1},xyz]

can be indicated by the numeral 1. For example, the expression

<Pelvis,1>

defines the global rotation of Pelvis, with angles measured about global axes in the order xyz. This is equivalent to the expressions

<Pelvis,xyz> and <Pelvis,1,xyz>

Rotations about Fixed or Floating Axes

In computer graphics animation, it is usual to express rotations about axes in the fixed/parent segment. This is the convention in BodyLanguage, as illustrated above.

In some applications, including biomechanics and clinical gait analysis, Euler angles are normally expressed as rotations about the moving axes of the child or floating segment. To find these angles using the fixed angle calculation in BodyBuilder, you can do the following :-

1. reverse the segment order, ie. <segmentP,segmentC>
2. negate the rotation

For example,
CG: < LeftShank, LeftThigh,yxz>
Bio: -< LeftThigh, LeftShank,yxz>

The first example gives the fixed axis values, and the second gives floating axis values from the Thigh to the Shank, which gives quite different results. For further information about the underlying mathematics of this operation, please refer to the document EulerAngles.rtf.

Euler Angles

Segments and Rotations

First, it should be clear that segments are defined by local three dimensional coodinate systems, or axis systems.

To find the angle between two such segments (that is, a joint angle) the first stage is to find the rotational transformation matrix which represents the rotation from one segment's orientation to the other's orientation. In fact this is quite simple. The orientation of a segment is in fact exactly the same as the representation of the coordinate system i.e. the normalised directions of the three axes in global space. The origin of the coordinate system can be ignored, since we are interested only in the relative orientations of the two segments, and not their global positions.

If we denote the first segment as the parent, and the second as the child, then the rotational transformation matrix from the parent to the child is given by

M = PT.C

[Note that for pure rotation matrices, a transpose of the matrix is equivalent to the matrix inverse. For example, if a matrix represents a simple rotation of N degrees round a given axis, then the inverse (and the transpose) represents a rotation of -N degrees around the same axis]

If this is not entirely clear, then don't worry too much. The main thing is that the matrix M represents what may be a fairly complicated rotation which transforms points from the parent segment, to equivalent points in the child segment.

Calculating Inter-frame Angles

The matrix M represents the rotation from one segment to another. However the representation is very difficult to interpret physically. Several angle notation methods can be used to represent the same three dimensional rotation in a more intuitive way.

Using a "fixed-axis" notation for instance, all the angles are defined as occurring about the axes of the fixed parent reference frame, which stays in the same orientation on the application of each of the rotations.

An alternative to the fixed-axis notation is to use Euler angles. When using Euler angles, all the angles are defined as occurring about the axes of a moving frame, which starts out with the same orientation as the original frame, but whose orientation changes on the application of each successive rotation.

In both cases a frame is moving from the parent orientation to the child orientation in (for instance) three steps, leading to two intermediate frame orientations between initial (parent) and final (child) orientations. The difference is that fixed angles are always measured about the axes of the fixed parent frame, whereas Euler angles are calculated about the axes of the moving intermediate frame at each step.

To calculate angles using either of the above methods, you need to decompose the single rotation matrix into three separate rotation matrices that represent rotations round simple known axes. The order of the decomposition can be chosen in a number of different ways, all of which result in the generation of different values for three angles. It is crucial in the understanding of "Fixed-axis" or "Euler" angles that you know which rotations about which axis each of the angles represents.

Each of the "Fixed-axis" and "Euler" notations has twelve possible available angle sequences, representing the six possible combinations of X, Y and Z (XYZ, ZYX etc.), plus the six possible combinations involving a repeated rotation about the same axis (XYX, ZYZ etc.). Any of these (24 options if we consider just the "Fixed-axis" and "Euler" notations) can be used to represent the same three dimensional rotational transformation matrix. You can only correctly re-constitute the matrix (and thus get the correct rotation) if you know the three axes that the three angles are rotating around.

The method for finding the three angles for a given matrix is the same for all angle configurations. Basically you start with the algebraic representations of your rotations round simple axes, and compose them together to give an algebraic representation of the composite matrix M. Then you inspect your algebraic representation of M to find entries that you can combine to give you simple expressions related to individual angles. This gives you formulae for the angles that you want, which you can then apply to the actual numbers you find in any given example of M that you have.

Thus, for example, you may choose to calculate the angles based on the order Y, X then Z on a fixed axis system. That is, rotate the parent segment round the Y axis of the parent first, to give an intermediate axis sytem I1. Then rotate I1 round the X axis of the parent segment to give I2, then finally rotate I2 round the Z axis, again of the parent segment, to arrive at the child segment orientation. Given that X, Y and Z are simple rotation matrices round the given axes, the combined rotation matrix M can be found by doing the appropriate matrix pre-multiplications

M = Z.X.Y.I

where I is the identity matrix. Thus you take the Identity matrix I, representing the starting position, and apply the Y rotation matrix (in fact a reduntant operation, since Y.I = Y). Then you apply the X rotation to the result, then the Z rotation to the result of that. This will give a matrix M which contains lots of terms for the sin and cosine of the three angles that you are interested in. By combining some of these terms and applying some inverse trigonmetric functions, you can extract the angles you want.

See for example Graphics Gems II p320 for more details of this method. [Ed. J. Arvo, Acedemic Press, 1991, ISBN 0-12-064481-9]
Also, Craig, J.J. (1989). Introduction to Robotics, Mechanics and Control. Reading, Mass. Addison-Wesley. Chapter 2, gives an excellent overview of rotation matrix and angular notation calculations.

Euler Angles

As discussed above, as well as being able to choose the ordering of your axes round which you rotate, you can also choose to make your rotations around a moving system using Euler angles, rather than the 'fixed' axes of the parent segment. That is, you can make the three rotations round the axes of the intermediate segment as it is rotating.
To derive these three (different) angles from the composite matrix M you just need to apply the same technique as above.

Take the rotation order as before, YXZ but this time on an Euler axis system. This leads to a rotation of the parent segment round the Y axis of the parent first, to give an intermediate axis sytem I1. Then a rotation of I1 round the X axis of I1 to give I2, then finally a rotation of I2 round the Z axis of I2, to arrive at the child segment orientation.
[NB: Note the difference between this description and that previously given for the fixed-axis notation.].

Looking at the matrix description of this, the Y axis rotation is no problem at all, it's just a simple rotation round the Y axis of the parent, since it is the first one to be applied. But if you then premultiply the result by the X axis rotation matrix, you are going to rotate round the parent X axis. To get the X rotation round the X axis of the intermediate axis system, you also have to rotate the X axis rotation matrix round the Y axis, so that the X rotation axis coincides with the X axis of the intermediate axis system. i.e. Y.X. The same argument applies to the Z rotation, only this must be premultiplied by both the Y and X rotation matrices, to get the Z rotation axis to line up with the second intermediate axis system. This gives Y.X.Z. Note that when you are applying the Y and X rotations to the Z rotation, applying the result will also perform the X and Y rotations on the original parent axis system. Thus you get

M = Y.X.Z.I

Again, you can solve this algebraicly and combine terms and solve to give you your required angles.

Fixed vs. Euler Axes

Well, I am sure that you have noticed that the difference in the decomposition formulae between the fixed and Euler notations is simply that the order of the rotations is reversed. What does that mean in practice ? What it means is that the formulae for calculating angles for a fixed frame notation in a particular order, is exactly the same as the formulae for an Euler frame notation, but for angles in the opposite order. So for an Euler notation with rotations in the order Y, X, Z, and for a fixed frame with rotations in the order Z, X, Y, the decomposition for both is

M = Y.X.Z.I

This means that in BodyBuilder, which finds fixed frame angles in any order, you can find the Euler angles by asking for the fixed frame angles in the opposite order to the actual order you want.

Similarly, if you have an Euler notation algorithm that solves the angles for any order of application, you can get a fixed frame solution for a given application order by asking for the Euler frame solution in the opposite order.

Call this Rule 1

What it *doesn't* mean is that you can just ask for the angles in the order you want from a fixed frame solution, then just swap them around to give you your Euler solution. The angles that are calculated for fixed and Euler notations are quite different. If you have already got angles that were calculated using a fixed frame, and you want to know the equivalent using an Euler frame, the most straightforward way to do this is to re-compose the complete rotation matrix M using a fixed frame, and then decompose again using an Euler frame. Though you could do all the maths and work out a more direct solution.

Parent and Child

Up to now, we've been considering the rotation of the parent frame onto the child frame, but we can also swap them, and consider the rotation of the child onto the parent. Of course the child to parent transformation is just the inverse transformation of parent to child, so it is just MT.

Also it is directly intuitive to realise that, if you are using a Euler reference frame, the three rotations from the child to parent, are just the same as from parent to child, but in the opposite order, with each rotation being in the opposite direction (that is, the inverse rotation). So, given that

M = Y.X.Z.I

then the inverse child to parent transformation is given by

MT = ZT.XT.YT.I

This is true whether you have a fixed or a Euler reference frame for your axes of rotations.

What this means, is that you can get the same result from an Euler angle calculation, if you swap the two segments, invert the angle order, and negate the resulting angles (i.e. invert the simplified rotation matrices).

Call this Rule 2

It doesn't seem very useful at this stage, but it can be handy.

Euler Angles and BodyLanguage

As stated above, the BodyLanguage function for calculating angles between segments uses a fixed coordinate system for the angles that it calculates. You give the child segment first, then the parent segment, then an axis order token.

That is, if you write the expression

Angle = <Thigh, Pelvis, XYZ>

what you get for Angle(1) is the X rotation from the Pelvis to the Thigh round the X axis of the Pelvis, Angle(2) is the rotation from the first intermediate axis system to the Thigh round the Y axis of the Pelvis, and Angle(3) is the remaining rotation from the second intermediate axis system to the Thigh round the Z axis of the Pelvis.

If you want to apply Rule 2 on it's own, you can, and get the same result

Angle = -<Pelvis, Thigh, ZYX>

except that of course the three angles will be in the opposite order (Angle(1) is the Z rotation, (3) the X rotation).

You can apply Rule 1 to get Euler reference frame angles out. Say, as is typical with biomechanical gait models, you want the order YXZ with an Euler frame. You can apply Rule 1 to make the fixed frame algorithm give you that, just by inverting the rotation order. So instead of writing

Angle = <Thigh, Pelvis, YXZ>

which would give you the fixed axes, you should simply invert the angle order token, to give

Angle = <Thigh, Pelvis, ZXY>

which will give you the three Euler angles you want. But it gives you them in the opposite order than you would normally like - that is it gives you the Z angle in Angle(1), and the Y angle in Angle(3). Of course this is easy to rectify with another little bit of code.

But here is where Rule 2 comes in useful. You can then apply Rule 2 to get the same result in terms of the values for the angles, but because Rule 2 swaps the order of the rotations again, you get the angles in the right places in the Angle variable. Take the last expression, swap the segments, swap the rotation order, and negate the results, and you get

Angle = -<Pelvis, Thigh, YXZ>

and that gives exactly the same numbers, but now the rotation round the Y axis is in Angle(1), and the rotation round Z is in Angle(3).

Now this expression looks a bit like a Euler reference frame calculation, that takes the parent segment first, and then the child, but gets the sign of the angles wrong ! This has been a major source of confusion, it seems. But now you can see that it isn't.

Keeping in mind exactly what BodyLanguage gives you (fixed frame rotations in any order) and with the careful application of Rule 1 and Rule 2, you should be able to extract whatever useful angles you would like.

There is an accompanying little BodyLanguage model [Angles.mod] which illustrates these things. Basically it constructs three segments: a parent, one child which is rotated from the parent round a fixed axis system of the parent, and a second child rotated round a Euler system of the intermediate axes systems. Then it uses the angle function, as described above, to calculate the correct angles between the parent and the two children, applying the two rules to get the results that you expect. You can run the model with any data file you like.

Gimbal Lock

The most widely-used form for recording the rotation between two segments is as 3 Euler (Cardan) angles. The primary rotation angle (normally flexion) is defined about one axis (side-to-side, or lateral), a secondary rotation angle (normally abduction/adduction) is defined about a second axis (forward, or anterior), and a third rotation angle is defined about the remaining axis (nominally vertical). This is the method used by BodyBuilder.

However, Euler angles suffer from one well-known problem. If any of the rotation angles becomes close to 90 degrees, for example lifting the arm to point directly sideways (shoulder abduction about a forward axis), the other two axes of rotation become aligned with one another, making it impossible to distinguish them from one another (shoulder flexion and rotation). This is known as gimbal lock.

True gimbal lock is rare, arising only when two axes are perfectly aligned. A similar effect occurs when a segment spins through more than one revolution. If the range of all three Euler angles is +/-180∞, there are two possible sets of Euler angles which describe a given orientation. This ambiguity may cause switching between one solution and the other, resulting in sudden discontinuities. This situation resembles gimbal lock.

To avoid this, one of the angles must be restricted to +/-90∞. In BodyLanguage, the second angle is chosen to be restricted in this way. This restriction applies to static trials.

In dynamic trials, if a segment rotates continuously, BodyLanguage prevents discontinuities by keeping an internal record of rotations in a form which does not suffer gimbal lock.

Mass and Inertia Properties of Segments

Mass and inertia can be assigned to each segment for the purposes of calculating forces and moments. This can be done in two ways: either directly, by supplying numerical values in a definition expression; or indirectly, by referring to a table of typical values.

Direct Definition of Mass Properties

These can be made in the same expression as the basic kinematic and hierarchical definitions, or added as an extension to a previously defined segment. The mass properties must the last part of the definition, whether the whole definition is made in one expression, or in parts. The syntax for extending a previously defined segment is:

```
segmentS=[segmentS,SegmentMass,CentreOfMassPoint,Inertia]
```

where segmentS is the segment name
SegmentMass is a scalar quantity expressing the mass of the segment (in kg)

CentreOfMassPoint is the location of the centre of mass of the segment, in the local co-ordinate system of that segment

Inertia is defined using three terms, like a point, which correspond to the components of the moment of inertia

Definition using a Table

The inertial properties can be defined using a table of anthropometric data based on standard values for the mass and inertial properties of various body segments. In most cases this will be a close enough estimate to give useful data. The table must be included in the model script file before any reference is made to it in segment definitions.

Each entry in the table must include a name and four numbers. The numbers are:

1. the segment mass as a proportion of total body mass
2. the location of the centre of mass as a proportion of the length of the pricipal (long) axis of the segment
3. transverse radius of gyration around the centre of mass
4. longitudinal radius of gyration around the centre of mass

the radii of gyration being quoted as a proportion of the length of the segment. Note that it is assumed that the segment definition follows the convention that the origin is at the distal end, with the first (principal) axis being the length of the segment (origin of segment to attachment point in proximal segment). The centre of mass is assumed to lie on this axis (which is a good approximation for human limb segments) and the three length numbers are all expressed as a proportion of the segment length.

The table must begin with the word AnthropometricData and end with the word EndAnthropometricData

Example:

```
AnthropometricData
DefaultFemur 0.1 0.567 0.323 0
DefaultTibia 0.0465 0.567 0.302 0
DefaultFoot 0.0145 0.5 0.475 0
EndAnthropometricData
```

In this example, the numbers on each line are taken from published anthropometric data (Biomechanics and Motor Control of Human Movement, David A. Winter, publ. John Wiley, p. 56).

The total body mass must be given, either in the model script or (preferably, since it will be different for each subject) in the parameter file, using the name BODYMASS, in a line such as

```
BODYMASS = 75 {*kilos*}
```

The longitudinal radius of gyration is used for the inertia calculations around the principal (long) axis, and the transverse value, around the other two axes. Reference to the entry values are made by the entry name. The segment length defaults to the distance from the segment origin, to the attachment point on the parent segment, a distance which may vary slightly from frame to frame because of skin movement, measurement noise, or inappropriate joint centre estimation, but which will in most cases be nearly constant.

A fixed length may be defined using a value from the parameter file:

```
segmentS=[segmentS,EntryName<,SegmentLength>]
```

where EntryName is the reference to the table (eg. DefaultFemur in the example above), and SegmentLength is an expression for the segment length (expressed in the units in which the original measurements were made, normally millimetres).

Correcting for the Mass of a Prosthesis

The tabulated values in the example above refer to the mass of a normal complete body. If one or more segments have been replaced by a prosthesis which has a mass greater than the natural segment it replaces, the BodyMass needs to be corrected to allow for this. The value required is an estimate of what the body mass of the subject would be if they had all natural body segments - the equivalent intact body mass.

For example, if the subject has one artificial foot which has a mass of 2kg, and the total body mass of the subject with the prosthesis is 62kg, then the body mass without the prosthesis is 60kg. From the above table of normal proportions, this represents (1 - 0.0145) of the total mass of the body if the subject had a natural foot. To calculate the equivalent intact body mass, the actual body mass (60kg) without the prosthesis should be divided by this figure, giving 60/0.9855 = 60.88kg. The formula in the general case is

BodyMass (equivalent intact body mass)
= (Total Mass - Mass of Prosthesis)/(1 - Normalised Segment Mass)

where the normalised segment mass is the proportion of normal body mass which is attributable to the segment replaced by the prosthesis.

The segment in the model which represents the prosthesis, should then be defined with the specific mass of that prosthesis.

Reactions, Forces and Moments

Reaction, Force and Moment are kinetic objects which can be defined using BodyLanguage assignment statements.

Force

Force objects have identical properties to points. They can be plotted and saved as points, using OUTPUT() statements, or stored as parameters using PARAM() statements. They can be defined from named or constant numbers in the same way as points:

```
forceF = {numberA,numberB,numberC}
```

Forces may be split into components in the same way as points

```
component1 = 1(forceF)or forceF(1)
```

Moment

Moments, like forces, have identical properties to points. They can be plotted and saved as points, using OUTPUT() statements, or stored as parameters using PARAM() statements. They can be defined from named or constant numbers in the same way as points:

```
momentM = {numberD,numberE,numberF}
```

Moments can be split into components like forces or points.

Reaction

Reaction is a combination of force, moment, and point of application:

```
reactionR = |forceF,momentM,pointP|
```

where pointP is the point (local or global) at which the reaction is applied. A reaction may be split into its components as follows:

```
pointP = 1(reactionR) or reactionR(1)
forceF = 2(reactionR) or reactionR(2)
momentM = 3(reactionR) or reactionR(3)
```

Forces, moments and reactions can all be transformed between global and local co-ordinate frames of reference using the * and / operators. By convention local names should begin with the % sign.

A function allows a reaction to be referred to a new point:

reactionR = REFER(reactionR,pointP}

Reactions can be added and subtracted. The result is referred to the point referenced by the first reaction specified. Reactions can be negated in sign. Reactions can also be added to the parameters list

PARAM(<. . .,>reactionR<,. . .>)

and they can be output as three trajectories

OUTPUT(<. . .,>reactionR<,. . .>)

The three trajectory labels are generated from the reaction name, with the suffixes F, M and P applied to the force, moment and point components respectively, and the force and moment trajectories are marked accordingly. The moment values are converted to Nm when they are output, making them less distracting in the workspace view by limiting the amplitude of their motions.

Individual forces and moments are distinguishable from points only by their names within the model. There are no constraints applied to their use, other than those normally applied to points. If a force is OUTPUT independently from a reaction, it appears as a point in the workspace, and in the C3D file if saved.

Force Plate Data

Force plate data, if present in the input C3D file, is automatically integrated by a kinetic model. When a REACTION function is called (see below), each of the force plates is deemed to be in contact with one, and only one, segment, for each frame in the trial.

An automatic internal test is made for each segment that is part of a hierarchy for which reactions need to be solved. The automatic algorithm uses the following parameters to determine which segments defined in the script are connected to each forceplate.

For any segment to be attached to a force plate, the magnitude of the force must exceed that of the ForceThreshold, which may be set to any scalar value. The default value is 10 N.

A segment origin or attachment point must be perpendicularly above the plate to be considered, at a distance less then the DistanceThreshold, which may also be set to any scalar value. This has a default of 200 mm.

The closest end of the segment to the plate must also have a velocity less then the VelocityThreshold. The default VelocityThreshold is set to 2000 mms-1. Note that if this value is too low, it can cause the ends of the forceplate data to be cut off from the REACTION calculations, which may lead to subtle changes in graphs of kinetic variables. Great care must be taken in setting this value. It is deliberatley set to a relatively high level just for this reason. It most cases, where both feet are modelled, this value need not be changed.

If you find that these default values are not appropriate for your models or testing procedure, then they can be overidden in the model script using an assignment statement such as

```
ForceThreshold = 5
```

The segment deemed to be in contact with the plate is the one which has the shortest perpendicular distance from origin or attachment point to the plate, and the lowest velocity. Where two segments have the same shortest distance, the segment with the closest second end (either origin or attachment point) is taken, preferring those ends above the plate.

The force plate data is available in the script as reaction types. They are named and numbered consecutively as ForcePlate1, ForcePlate2 etc.

Note that if the automatic connection of forceplates is found to be inappropriate in your particular application, it can be disabled by setting one of the threshold values very high. If required, reactions from the forceplate can be ‘connected’ to selected segments using the CONNECT function described below.

Segment (free body) Equations of Motion

A single reaction function solves the equations of motion of a segment, taking into account all reactions applied to it by its child-segments in the hierarchy, as well as segment mass distribution, its motion, and gravity. The result of the function is the reaction applied to the segment, at its attachment to its parent, which achieves dynamic equilibrium.

```
ReactionR =     REACTION(segmentS)
```

where segmentS is the segment in dynamic equilibrium.

The reaction can be referred to a different point in the segment by providing a point, which is assumed to be local to the segment, as a second argument to the function.

```
ReactionR =     REACTION(segmentS, PointP)
```

Connection of other reactions

```
CONNECT(Segment,Reaction,ScalarTest)
```

This function allows the application of the Reaction expression to the given

Segment. Once this function has been called, the reaction function is held internally with the segment name, and included in the list of reactions applied to the segment when a REACTION function is called for the segment (or any parent segment, for whom this segment's reaction is required to be calculated). The ScalarTest parameter is a scalar expression which allows selective connection of the reaction. If ScalarTest evaluates to zero for a particular frame, then the Reaction is not applied to the Segment, otherwise it is.

Gravity

Gravity is specified internally in a point format

GRAVITY = {0,0,9810}

This defines gravity as equivalent to an upwards acceleration of 9810 mms-2, assuming that the VICON lab (global) space was calibrated with Z up. This value can be changed anywhere within the script. Following REACTION results will be correspondingly affected.

Changing the value of the acceleration due to gravity does not change the motion of the segments, which is driven only by the measurements of the trajectories of markers on the subject. Reducing the gravity specification will not make the subject "moonwalk" or "spacewalk". Although the function would be useful for kinetic accuracy in the event of data being captured in space, it is actually provided to cater for situations in which the Z axis is not up.

Linear Acceleration and Angular Velocity and Acceleration

To calculate segment reactions, values of linear velocity and acceleration are calculated using the segments position and orientation over a period of 0.06 seconds, centred on the current frame. This gives a reasonable time period during which the segments move by an amount significantly greater than the noise or errors in the positions of the markers. This sampling period is fixed. The sampling period used for the calculation of angular velocities for the powers is 0.04 seconds.

Intersegmental Power Flow

A new number function is defined to calculate power flow between segments

powerI = POWER(segmentA,segmentB)

where segmentA is the segment to which power (+ve or -ve) flows
segmentB is the segment from which power (+ve or -ve) flows.

Units

The following units are assumed within the calculations :

Mass	kilograms	kg
Distance	millimetres	mm
Force Newtons	Newtons	N
Moment	Newton millimetres	N.mm

Inertia	kilogram.millimetre.millimetre	kg.mm2
Power	milliwatts	mW

So if, for example, the force plate moment data is supplied in the C3D file in Newton metres, it should be converted to N.mm by a line such as

ForcePlate1 = |ForcePlate1(1),
ForcePlate1(2)*1000, ForcePlate1(3)|

This line will not affect the reactions automatically applied from the force plates. Instead, automatic application must be disabled, by setting either the ForceThreshold to a very high value (at least 100*BodyMass for example), or the DistanceThreshold to zero. Then the reaction from the force plate can be applied to the segment using the CONNECT function.

If such corrections to the incoming data are not made, then the output values should be converted as necessary. Note that in calculating the moment due to the angular acceleration, the moment is divided by 1000 internally, to convert correctly to N.mm, given that the inertia is supplied in kg.mm2.

If using tables, the units the inertia are in depends on the units of the length and mass of the segment, as supplied by BodyMass and the segment definition of the origin and attachment point or the supplied length expression.

Types of Operator

Operators are symbols used to combine, select, or compare objects in an expression.

There are three types of operator: arithmetic, selective, and logical.

Arithmetic and selective operators can be used together in an arithmetic expression.

The arithmetic operators are +, -, *, and /. They can be used to combine objects of the same or of different types. The type of an arithmetic operation is defined by the type of its result.

The selection operator is ?. It is used in an arithmetic expression between clauses (sub-expressions) of the same type.

The logical operators AND, OR, NOT, <, >, and = are used to compare numbers, or number expressions, within logical expressions. Logical expressions are used only in Conditionality Tests.

In an expression, clauses are evaluated according to the order of precedence:

- (used to indicate negation)
/,*
-, +
>=, <=, >, <
<>, ==
NOT, OR, AND
?

Number Operations

Numbers can be added (+), subtracted (-), multiplied (*), or divided (/) in an arithmetic expression. The result is always a number and the normal arithmetic precedents apply.

Numbers can be compared in a logical expression. The result of a logical expression is true or false:

numberA AND numberB	is true if numberA and numberB are both true (non-zero)
numberA OR numberB	is true if either numberA or numberB is true (non-zero)
numberA<numberB	is true if numberA is less than number B
numberA>numberB	is true if numberA is greater than numberB
numberA==numberB	is true if numberA is equal to numberB

Paired combinations are permitted, in either order;

<= or =<	less than or equal to
>= or =>	greater than or equal to
<> or ><	not equal to

Point Operations

Points can be added (+) and subtracted (-):

(pointI+pointJ)/2	mid-point of pointI and pointJ
(pointJ-pointI)	line from pointI to pointJ.

Examples:

Point addition and subtraction:

```
FPelvis=(LFWT+RFWT)/2
```

defines the virtual point FPelvis as the mid-point of the line joining the real markers LFWT and RFWT.

```
LFemur=[LKJC,LHJC-LKJC,LAJC-LKJC,zyx,LTOE-LKJC]
```

defines the segment LFemur using the lines LHJC-LKJC and LAJC-LKJC as defining lines, and LTOE-LKJC as the anti-flip line.

Number comparison:

```
If
   $Static==1
   $#Child#Length=DIST(O(Child),O(Parent))
   PARAM($#Child#Length)
EndIf
```

If the variable $Static is set to 1 in the parameter file, the parameter $#Child#Length is defined. For example, a parameter such as "$TibiaLength" might result from this definition. The PARAM command carries an implicit AVERAGE function, meaning that a single mean value of $ChildLength would be written to the parameter file. The EndIf line indicates the end of the conditional section. Not all possibilities result in actions: if both $VariableBoneLengths and $Static are both false (set to a value other than 1), no definition is implemented. Points can also be scaled by numbers. For example, pointI*numberA scales the position of the point I.

Example:

```
%LHJC=PelvisDiameter*$%HipOffsetFactor
```

In this example, the local point %LHJC is defined by the vector $%HipOffsetFactor (from the parameter file) multiplied by the number

PelvisDiameter (whose value is calculated in the preceding line).

The operators * and / have a special meanings in point operations. They are used to transform a point between its global version in the Workstation calibrated space and its "local" version in a segment, and vice versa;

pointI = %pointI*segmentP	converts local %pointI in segmentP to global pointI
%pointI = pointI/segmentP	converts global pointI into local %pointI in segmentP

Note that it is the operators in the expression which define the action, not the % character in the name of the point object.

Segment Operations
Points can be added to (+) and subtracted from (-) segments. The result is a segment in a new position but with the same rotation.

segmentP = segmentP+pointI	segmentP is moved to O(segmentP)pointI where O(segmentP) is the origin of segmentP.

Rotation Operations (none permitted)
Because of the nature of Euler angles, rotations cannot be combined by any of the arithmetic operators. For example, two rotations cannot be added together.

Selection Operator
The selection operator ? is used within an arithmetic expression to separate alternative clauses of the same type. The expression is set equal to the first clause which is defined in the current field (sample);

pointI = pointJ ? pointK ? ? pointL	pointI is set to pointJ, if defined, otherwise pointK, if defined, otherwise pointL

The selection ? operator has a lower precedence in an expression than any of the arithmetic operators.

Types of Function

A number of special functions and constructions may used in an expression. They are defined according to the type of their result.

Number Functions

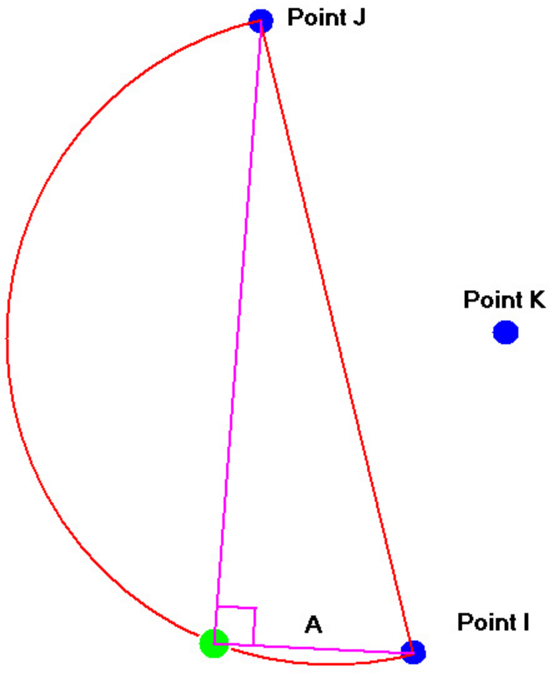
1(pointI),2(pointI),3(pointI) or	Components of pointI
pointI(1),..(2),..(3)	Components of pointI
n(rotationq) or rotationq(n)	nth angle component of rotationq
COMP(pointI,pointJ)	Component of pointI in direction of pointJ
DIST(pointI,pointJ)	Distance between pointI and pointJ (either point can be omitted)
SIN(Angle)	Sine of the angle (in degrees)
COS(Angle)	Cosine of angle (in degrees)
TAN(Angle)	Tangent of the angle (in degrees). Angle can be any value except ±90∞, or other value of which the tangent is infinite.
ASIN(Value)	Arc-sine function. Value must be between -1 and +1. Returns angle in the range -90∞ to +90∞.
ACOS(Value)	Arc-cosine function. Value must be between -1 and +1. Returns angle in the range 0∞ to +180∞.
ATAN(Value)	Arc-tangent function. Value may be any number. Returns angle in the range -90∞ to +90∞.
ATAN2(x,y)	Arc-tangent function. X-value is the base of the triangle concerned, Y-value is the perpendicular. Zero values not permitted.
SQRT(Value)	Square root function. Value may be any number greater than zero. Returns the positive square root.

Point Functions

O(segmentP), or 0(segmentP)	Point at origin of segmentP

1(segmentP), 2(segmentP), 3(segmentP), or segmentP(1), ..(2), ..(3)	Directions of axes of segmentP
NORM(pointI,pointJ,pointK)	Direction perpendicular to plane of 3 points I,J,K
PERP(pointI,pointJ,pointK)	Point at perpendicular bisector from point I to line J-K
CHORD(numberA,pointI, pointJ,pointK)	Point at distance A from I in plane IJK forming a right angle between I and J on the opposite side of IJ from K

The CHORD function may be useful in locating the centre of the knee joint given the hip joint centre and a marker on the mid-thigh and lateral condyle of the knee. This method is used in VICON Clinical Manager and Plug in Gait gait analysis software.



In the illustration above, the green point is created using the CHORD function. The line between points I and J is used as the diameter of a circle. The plane of the circle is the plane of this line and point K. Any point on the red semicircle will fulfill the right-angle criterion, and the distance A specifies the point indicated. The distance A is thus used in the construction as the chord of a circle, hence the name of the function.

Segment Functions

ATTITUDE(segmentP)	Move segmentP to the global origin {0,0,0} keeping the same rotation (attitude)
ROT(segmentP,pointI,q)	Rotate segmentP about direction of pointI by q degrees
ALIGN(segmentP,p,q, pointI)	Rotate segmentP about axis p (=1,2,3) to align projections of axis q (=1,2,3) and pointI in plane perpendicular to p

Rotation Functions

rotationq(-n)	Change the sign of the nth angle (n=1,2,3) in rotationq
<numberA,numberB,numberC>	Define a rotation from three angles (in degrees). This function has the same syntax as rotation definition

Special Functions

LAST()is a special function used to recall the most recent arithmetic value of an object from the current or any previous field. If the object has never previously been defined, it remains undefined.

LAST(pointI)	pointI from the last occasion it was defined

EXIST() is a special function used in a logical expression to establish whether or not an object (number, point, segment, or rotation) is defined in the current field.

EXIST(pointI)	true if pointI is defined in current field

EXISTATALL() is used in a logical expression to establish whether or not an object (number, point, segment, or rotation) is defined in at least one field during the trial. More than one object may be supplied to the function, in which case all such objects must exist for at least one frame during the trial.

EXISTATALL(pointI, pointJ)	true if both I and J are ever defined

EXISTALWAYS() is used to determine whether an object or objects have values for every frame in the trial.

AVERAGE() is a special function which can be applied to any type of object (number, point, segment, or rotation) , but which is evaluated in a different way from all other functions.

Normally, each assigned expression is evaluated once per field through the full set of data to which the model is applied. However, any expression containing an AVERAGE() function is evaluated only once for all fields.

AVERAGE(numberA)	Average numberA over all fields
AVERAGE(pointI)	Average pointI over all fields
AVERAGE(segmentP)	Average segmentP over all fields
AVERAGE(rotationq)	Average rotationq over all fields

For technical reasons, the AVERAGE() function is not allowed inside IF blocks. There is more information about this in the file "AverageModifications.txt"

MIN() MAX() STDDEV() functions calculate the minimum, maximum and standard deviation (n-1) values for scalar values over the whole trial. These functions are not allowed inside IF blocks either.

[ ] is a special "field offset" post-fix function used to find the final value of an expression in any field, relative to the current one.

segmentP[-1]	segmentP in previous field

The "sample offset" function has many uses, including the creation of special filters.

FIELD or SAMPLE, FIRSTSAMPLE and LASTSAMPLE are special predefined values that can be used in the script to provide the current, first and last sample numbers in the trial that is being processed. These are fixed values, and cannot be changed.

Spine

SPINE()is a special segment function for creating new segments based on the position and orientation of existing ones. As the name implies, it is designed for creating vertebrae between the segments Pelvis, Thorax and Head. The general form is

SPINE(<length,><direction,>SegmentL,<location,
>SegmentU<,stiffnessL><,stiffnessU>)

The parameters in <> brackets are optional. The default values listed below are used if they are omitted:

SegmentL	is the lower segment at one end of the spine
SegmentU	is the upper segment at the other end
length	is the total distance along the curve of the spine between segmentL and segmentU.

The default length of the spine is the length of the cubic base spline between the end segments.

direction	is a local line (axis or local point) within both segmentL and segmentU indicating the local directions of the ends of the spine. The same local direction is used for new segments generated within the spine. Default is the Z or 3 axis.
location	is the relative distance (0 - 1) along the spine from SegmentL to SegmentU where the origin of the new segment will be located. Default is 0.5

stiffnessL and stiffnessU set the flexibility at either end of the spine. The stiffness of any point on the spine is a linear proportion of the values set for either end. Higher values mean that the spine is stiffer in the direction of its curve, and the orientation of its created segments remain closer to the orientation of the end segments. Lower values mean that the spine is more flexible and its direction and segment orientation tend towards a straight line connecting the end segments. Default stiffness is 1.

Length and Stiffness

The SPINE parameters length and stiffness are interdependent. A spine of unspecified length, with identical stiffness set at either end, is generated with the same stiffness along its length. As the stiffness is increased, the spine grows longer, eventually forming a loop between the ends. If the stiffness is decreased, the spine grows shorter, forming a straight line between the ends when stiffness is 0.

If length is specified, the absolute stiffness cannot be controlled independently. In this situation, the specified stiffness parameters indicate the relative stiffness at either end.

Multiple Segments/Vertebrae Mode

The SPINE function has a special mode which can be used to generate a series of vertebrae simultaneously:

SPINE(SegmentL,<location1>,Segment1,<location2>,Segment2,...,<locationN>,SegmentN,SegmentU>)

where

Segment1,Segment2,...,SegmentN	are new segments created along the spine

<location1>,<location2>,...,<locationN>
are the optional relative distances of the new segments along the spine. Locations always appear before the segment to which they apply.

The optional parameters length, direction and stiffness can be included and have the same effect as in the previous mode.

This mode of the SPINE function is used on its own, rather than as an expression in an assignment. It does not appear on the right of a = sign.

Shorthands and Tokens

When 0 (numeral zero) is used in place of a point, it means {0,0,0}. If it is a local point, it means the origin of the segment. If it is a global point, it means the global origin.

When 1 (numeral one) is used in place of a segment, it means:

[0,{1,0,0},{0,0,1},xyz]	the global segment

The following tokens are predefined to the numbers indicated:

```
xyz = 1
yzx = 2
zxy = 3
xzy = -1
yxz = -2
zyx = -3
```

Their values may be changed, but they may not be assigned as points, segments or rotations.

Non-Assigning Statements

There are three statement lines which do not make assignments.

OUTPUT(pointl,rotationq,..)

Create output records of pointl and rotationq for all the samples for which they exist. Output records are displayed in the 3D Workspace and can be saved.

PARAM(numberA,pointl,segmentP,rotationq)

Append assignments for numberA,pointl,segmentP and rotationq to parameter section of current BodyLanguage script. Parameter sections can be saved.

OptionalPoints(pontl,pointJ,..)

Listed points may be referenced in an expression even if they do not exist in the input C3D file. Any expression derived a non-existent optional point is not evaluated. There are also OptionalNumbers, OptionalForces etc. functions too.

Conditionality Tests

BodyLanguage allows user-specified assignments to be made if, and only if, the current value of a logical expression is TRUE or FALSE.

The form of a Conditionality Test is:

```
IF logicalexpression (THEN) assignment...
(ELSIF logicalexpression...) (ELSE assignment...) ENDIF
```

where brackets indicate optional elements of the test.

Dealing with Marker Occlusions

The primary use of Conditionality is to create virtual points in place of real points that are temporarily missing, for example due to marker occlusions.

For example;

```
PelvisOrigin=(RHIP+LHIP)/2
LmRHip=LHIP-RHIP
PelvisABDO=[PelvisOrgin,LmRHip,ABDO-PelvisOrigin,2]
%SacrumAvABDO=AVERAGE(SACR/PelvisABDO)
IF EXIST(SACR) ELSE SACR=%SacrumAvABDO*PelvisABDO ENDIF
Pelvis=[PelvisOrgin,LmRHip,PelvisOrigin-SACR,2]
```

In this example, the segment Pelvis is normally defined in terms of three measured points, LHIP, RHIP, and SACR. However, if point SACR (sacrum, on the back of the pelvis) is occluded, it is automatically re-created using its local offset, %SacrumABDO, in an alternative definition of Pelvis based on measured point ABDO (abdomen, on the front of the pelvis). The same method can easily be extended so that there are several alternative definitions are used for a segment, according to what which measured points are available.

Uses of Macros

In a BodyLanguage script, many separate sections can end up repeating the same basic set of instructions. For example, the procedure for filling gaps in a group of related points may be identical but for the names of the points involved.

In order to make scripts shorter, easier to read, and more reliable, BodyLanguage supports the defining and calling of macros. A macro works like a subroutine in a compiled program. Once defined, a macro can be called as many times as required, thereby eliminating repetitions of text. Only when the script is executed, is each macro call automatically replaced by the lines in the macro definition.

In order to make a macro applicable in different situations, it can be defined using a set of dummy variables or macro parameters. These parameters are listed at the start of the macro definition. When the macro is executed, the parameters are replaced by a matching set of real variable names.

A macro definition, which must occur earlier in the script than any corresponding macro calls, starts with:

```
define macro    <macroname>(parameter1, parameter2, ...)
```

(although the word define is optional), and ends with:

```
endmacro
```

where:

<macroname>	is any unique, valid variable name

A macro call is simply:

```
<macroname>  (variableA, variableB, ...)
```

where the number and type of variables in the call must match the number and type of parameters in the definition. A macro call can constitute a whole line of script, or can appear within an expression, as long as the substitution results in valid BodyLanguage syntax.

Text Concatenation Operator

To make macros more powerful, a special text concatenation operator, #, is available for use within macro definitions (but nowhere else in a BodyLanguage script). Dummy variables within a macro definition can be constructed of concatenations of parameters and other strings of characters.

For example:

```
$#Child#Length=DIST(O(Child),O(Parent))
```

where: Child is a parameter of the macro definition of which this assignment is a part.

There are several examples of macro definition and text concatenation in the example script FULLBODY.MOD, which is listed and explained in a later section.

Comments

Comments are encouraged. Each comment string starts with characters {* and ends with characters *}. Comments can be "nested", ie. a complete section of script can be commented out although it includes comments. This requires strict matching of the {* and *} symbols in pairs. If an unmatched comment symbol causes a model script to fail to run, an error message will indicate the cause of failure, and the cursor will be placed at the first unmatched {*symbol.

Error Codes

Every time a BodyLanguage script is run, it first checks the parameters and model for errors.

If an error is found, a dialog box displays the error message. The line containing the error is highlighted and diplayed in the appropriate text window. If necessary, the text editor window will be opened, and the text will be scrolled to make the line visible. The error message also appears in the status bar at the bottom of the main application window. If you change windows of scroll the text, pressing the "show last error" button will re-display the offending line, and the error message.

If the cause is in a macro expansion, pressing the "show last error" button, or selecting the menu command, will display the line in the macro where the error occured. Pressing the button further will alternate between the two lines, to allow you to check both the macro code, and the arguments you are passing to the macro.

Anatomical Structure

The model FOOT divides the foot into several segments with limited degrees of freedom between them. The first segment defined is the tibia. Beneath this, and related to it by a simple hinge joint about an intermalleolar axis, is the talus segment. The position and orientation of the talus is deduced entirely from the segments around it - no markers are attached to it directly. The calcaneus is defined below the talus and related to it by a simple hinge joint, the subtalar joint, about an axis which is defined by the placement of a marker on the instep. The central part of the foot is represented by a midfoot segment, and the toes by a hallux segment.

Axis Definitions

The axes of rotation for the tibia-talus joint (ankle flexion joint) and the subtalar joint, in this model, are not derived from any assumed relationship between exterior landmarks and interior anatomy. The locations of these axes are determined entirely by marker placement. It is assumed, for example, that the ankle flexion joint axis lies between the malleoli, and it is suggested that the two markers labelled LMAL and MMAL are placed on these exterior landmarks, however, if they are placed elsewhere, the axis will be located on a line between them. Thus, the user has the freedom to define the location of this axis by placing the markers where they choose on the subject.

This contrasts with the assumptions made in the kinematic model used in VICON Clinical Manager, for example, where it is assumed that the markers will be placed on stated landmarks (such as the anterior superior iliac spines) and a theory of anatomy (based on cadaver studies) is used to derive the locations of the hip joint centres. In such a model, misplacement of the markers will invalidate the assumptions, and lead to erroneous interior points being calculated. In FOOT, there are no such assumptions, and if the user has their own opinion about the location of the axes concerned, they can place the markers accordingly. Users of models like this should be aware of the implications for comparability of data from different workers.

Handedness

This model has two equivalent sections, one for the left foot, dealing with all the markers whose labels begins with L, and one for the right foot, dealing with the markers whose labels begin with R. The two sections are very similar, but not quite identical - the corresponding segments on either side cannot be mirror images of each other, because of the requirement that all segment axes form a right handed system.

Model Script

Comment Section

The following comment section describes the markers used.

```
{*VICON BodyLanguage*}
{*copyright Oxford Metrics 1997*}
```

{*Foot.MOD, for use with Foot.MP paramter file*}

{*This a sample file for BodyBuilder for Biomechanics. It is supplied to illustrate some of the range of modelling techniques made possible by this program. It should NOT be used for clinical or research purposes without independent verification. Oxford Metrics accepts no liability for its performance.*}

{*The model uses a total of 28 markers:

Lateral and Medial Epicondyles of the Knee	RLEPC, RMEPC, LLEPC, LMEPC
Tibial Tuberosity	L/RTTUB
Shin	L/RSHN1, L/RSHN2
Lateral and Medial Malleoli	L/RMMAL, L/RLMAL
Lateral and Medial Calcaneus	L/RMCAL, L/RLCAL
Proximal and Distal 2nd and 5th metatarsals	L/R P/D2MT,P/D5MT
Hallux	L/RHLX

Medial epicondyle and medial malleolus markers are required only for static trials, however if they are present, they will be used.

1) collect static and motion trials, using above markers
2) set $static = 1 in .MP file and process static trial
3) adjust rotation parameters in .MP file until foot posture is as measured
4) set $static = 0 in .MP file and process motion trial
5) write output angles (ANKA,STLA,MDFA,HLXA) to C3D or ASCII file*}

Macro "Substitute4"

{*Start of macro section*}
{*======================*}

macro Substitute4(p1,p2,p3,p4)

This line begins the definition of the macro named Substitute4.

s234 = [p3,p2-p3,p3-p4]

This line defines the segment "s234", whose origin is at p3, whose Axis 1 is parallel to the line from p3 to p2, and whose second defining line is the line from p4 to p3. This segment will exist in every frame in which the points p2, p3 and p4 are all present.

```
p1V = Average(p1/s234)*s234
```

The above line defines the global point p1V in the following way. First, the expression in () brackets is evaluated in every frame for which it is valid. "p1/s234" is the local position of p1 relative to the segment s234. Therefore, the expression is valid in each frame in which all four points are present. Then, the average of this throughout the trial is evaluated, resulting in a single averaged local point (ie. not a separate value for each frame). Finally, this is converted to a global point by the *s234 operator, and this is evaluated in each frame, as the segment s234 is not fixed. The result is a new point, p1V, which is fixed relative to segment s234, and exists in every frame in which s234 exists. The averaging process acts as a kind of filter, in which the movement of p1 is smoothed by relating it to p2, p3 and p4. The result is that p1V will behave more smoothly than p1. Also, p1V will exist in frames in which p1 itself does not exist. This is the key feature of the operation.

```
s341 = [p4,p3-p4,p4-p1]
p2V = Average(p2/s341)*s341

s412 = [p1,p4-p1,p1-p2]
p3V = Average(p3/s412)*s412

s123 = [p2,p1-p2,p2-p3]
p4V = Average(p4/s123)*s123

p1 = (p1+p1V)/2 ? p1 ? p1V
p2 = (p2+p2V)/2 ? p2 ? p2V
p3 = (p3+p3V)/2 ? p3 ? p3V
p4 = (p4+p4V)/2 ? p4 ? p4V
endmacro
```

The last four expressions redefine p1 - p4 using the selection operator. In a frame in which p1 and p1V both exist (ie. all four original points exist), p1 is redefined to be the mid-point of the line joining p1 and p1V. This will have some filtering effect on the movement of p1, while allowing it to remain reasonably true to the original measurements. If p1V does not exist in a particular frame (it exists only in frame in which p2, p3 and p4 exist), the measured value of p1 is left unchanged. If p1 does not exist, but p1V does exist (ie. points p2, p3 and p4 only are present), then p1V is used to fill the gap in p1. This operation, therefore, modifies p1 by both smoothing its trajectory, and extending it into frames in which p1 originally did not exist. The result is that there will be a point p1 in any frame in which at least three of the four original points existed. In the following three lines, the same redefining expressions act on the other points. The result is that all four points are smoothed, based on the presumption that they are fixed to the same body segment and so move together; and gaps in the trajectories are filled in, unless more than one marker is missing, in which case there is insufficient data to reconstruct the segment.

```
{* End of macro section *}
```

Optional Points

```
{*Points which may not be present in every trial*}

OptionalPoints(RMEPC,RMMAL,RP5MT,RTTUB,RSHN2)
OptionalPoints(LMEPC,LMMAL,LP5MT,LTTUB,LSHN2)
```

Segment Definitions

```
{* Tibia segment *}
{* ============= *}
{* This segment uses 7 markers in static trials (L/MEPC,TTUB,SHN1/2,L/MMAL), with medial markers (MEPC and MMAL) removed for motion trials*}

{*Minimise dropouts of RLEPC and RLMAL*}
SUBSTITUTE4(RLEPC,RTTUB,RSHN1,RSHN2)
SUBSTITUTE4(RLMAL,RTTUB,RSHN1,RSHN2)

{*Create a dummy segment using most visible tibia markers, present in all trials*}
DummyTibia = [RLMAL,RLEPC-RLMAL,(RLMAL+RLEPC)/2-RSHN1]
```

The segment "DummyTibia" is defined with its origin at the marker RLMAL, and uses in its definition the points RLEPC and RSHN1. These markers will remain in position throughout the experiment, including static and dynamic tests. The segment DummyTibia itself is not interesting in biomechanical terms, but is used to derive some local points as follows:

```
{*For static trial in which medial markers are present, save key anatomical points and Tibia scale as parameters*}
If $Static == 1
        {*Find mid-points of knee and ankle*}
        RKJC = ((RLEPC+RMEPC)/2)
        RAJC = ((RLMAL+RMMAL)/2)
        $%RKJC = RKJC/DummyTibia
        $%RAJC = RAJC/DummyTibia
        $%RMMAL = RMMAL/DummyTibia
        $RTibiaScale = DIST(RKJC,RAJC)/100
        PARAM($%RKJC,$%RAJC,$%RMMAL,$RTibiaScale)
EndIf
```

The segment "DummyTibia" is used to obtain the local co-ordinates of the marker MMAL and the virtual points KJC and AJC, so that when the markers MEPC and MMAL are removed for dynamic trials, we can re-create their positions relative to the remaining markers. The If expression tests whether the parameter $Static has been set to 1. If so, that is a sign that the trial is a static one, and the virtual points are calculated and written (by the PARAM command) to the parameter file. If the test is not satisfied, because $Static is not set to 1, the section is not acted on.

```
{*Create global positions of key anatomical points from parameters and dummy tibia segment*}
If (Exist(RMMAL) AND $Static == 0) Then
        RAJC = ((RLMAL+RMMAL)/2)
Else
        RAJC = $%RAJC*RDummyTibia
        RMMAL = $%RMMAL*RDummyTibia
EndIf
If (Exist(RMEPC) AND $Static == 0) Then
        RKJC = ((RLEPC+RMEPC)/2)
Else
        RKJC = $%RKJC*RDummyTibia
EndIf
```

In this part, the marker RMMAL and virtual points RAJC and RKJC are recreated from the parameters calculated earlier.

```
{*Define Tibia segment with third axis aligned with long axis of tibia*}
RTibia = [RAJC,RKJC-RAJC,RMMAL-RLMAL,zxy]
```

In this definition, the segment origin is at the ankle centre, and the first defining line is the line from the ankle centre to the knee centre. This line (because of the token zxy) becomes the third axis. The second defining line is from the lateral to the medial malleolus. Because of the token, this means the second axis of the tibia section is in the plane defined by the first axis and the line RMMAL-RLMAL, and is positive in the lateral-to-medial direction (inward). This segment is of genuine interest, unlike DummyTibia, which is not referred to again.

```
{*Draw Tibia, using 16 vertices, scaled to length of Tibia*}
RTIB1 = $RTibiaScale*{10,0,95}*RTibia
RTIB2 = $RTibiaScale*{-4,-10,100}*RTibia
RTIB3 = $RTibiaScale*{-4,0,100}*RTibia
RTIB4 = $RTibiaScale*{-4,10,100}*RTibia
RTIB5 = $RTibiaScale*{3,0,80}*RTibia
RTIB6 = $RTibiaScale*{0,-8,80}*RTibia
RTIB7 = $RTibiaScale*{-3,0,80}*RTibia
RTIB8 = $RTibiaScale*{0,3,80}*RTibia
RTIB9 = $RTibiaScale*{3,0,20}*RTibia
RTB10 = $RTibiaScale*{0,-6,20}*RTibia
RTB11 = $RTibiaScale*{-3,0,20}*RTibia
RTB12 = $RTibiaScale*{0,3,20}*RTibia
RTB13 = $RTibiaScale*{10,0,10}*RTibia
RTB14 = $RTibiaScale*{0,-10,0}*RTibia
RTB15 = $RTibiaScale*{-5,0,10}*RTibia
```

```
RTB16 = $RTibiaScale*{0,10,4}*RTibia

OUTPUT(RKJC,RAJC)
OUTPUT(RTIB1,RTIB2,RTIB3,RTIB4,RTIB5,
RTIB6,RTIB7,RTIB8)
OUTPUT(RTIB9,RTB10,RTB11,RTB12,RTB13,
RTB14,RTB15,RTB16)
```

This section defines sixteen new virtual points for the purpose of drawing a simulacrum of the tibia in the Workspace display. The points will be joined together when a suitable MKR file is selected with the required links.

```
{*Draw y-axis of Tibia*}
RAKJ1 = RAJC+100*2(Tibia)
RAKJ2 = RAJC-100*2(Tibia)
OUTPUT(RAKJ1,RAKJ2)
```

In this section, two points are defined to illustrate the second axis of the tibia section. Note that the above two sections contained OUTPUT commands, which means that they actually generate new data points in the C3D file.

```
RTibia2=[RAJC,RLMAL-RMMAL,RAJC-RKJC,yxz]
```

In this segment definition, the intermalleolar axis is used as the first defining line, and becomes the second segment axis. This means that the second defining line (knee to ankle) is not normally parallel to the third axis, and the knee joint does not therefore lie on an axis. However, despite this difference, this segment cannot rotate relative to the segment RTibia. RTibia2 will be used later for defining the ankle flexion.

```
{* Talus *}
{* ===== *}
{* This segment uses 3 real (RL/RMMAL,RP2MT) and one virtual (RCCAL)
markers*}

{*Find mid-point of Calcaneous*}
RCCAL = (RLCAL+RMCAL)/2

{*Minimise dropouts of LEPC and LMAL*}
SUBSTITUTE4(RLMAL,RMMAL,RCCAL,RP2MT)

{* Draw Malleolar axes along y-axis of Tibia*}
RMALC= (RLMAL+RMMAL)/2
RMAL1 = RMALC+(RLMAL-RMMAL)
RMAL2 = RMALC-(RLMAL-RMMAL)
OUTPUT(RMAL1,RMAL2)
```

In this section, two points are defined which illustrate the inter-malleolar axis. This axis is not parallel to the tibia lateral axis.

```
{*Define Upper Talus segment with lateral(y) axis along inter-malleolar (RLMAL-RMMAL) line*}
 RTalusU = [RAJC,RMMAL-RLMAL,RP2MT-RCCAL,yzx]
```

The Upper Talus segment is defined with its origin at the same point as that of the tibia, and with the intermalleolar line as its second axis.

```
If $Static ==1
        {*Save Upper Talus scale as parameter*}
        $RTalusScale = ((DIST(RLMAL,RMMAL)-$MarkerDiameter)/1.5)/100
        PARAM($RTalusScale)
EndIf
```

The TalusScale parameter is one-hundredth of the distance between the lateral and medial malleolus markers.

```
{*Reposition RIGHT Upper Talus, maintaining attitude*}
%RSTJ = {0,0,-20*$RTalusScale}
RSTJ = %RSTJ*RTalusU
RTalusU = RSTJ+Attitude(RTalusU)
RTalusU=ROT(RTalusU,1(RTalusU),-$TalusRotation)
```

The TalusScale parameter is used to shift the segment "down". It is then rotated by a variable parameter TalusRotation.

```
{*Draw Talus, using 8 scaled vertices*}
RTAL2 = $RTalusScale*{30,30,45}*RTalusU
RTAL1 = $RTalusScale*{30,-30,45}*RTalusU
RTAL4 = $RTalusScale*{-30,-30,80}*RTalusU
RTAL3 = $RTalusScale*{-30,30,80}*RTalusU
RTAL6 = $RTalusScale*{65,50,-50}*RTalusU
RTAL5 = $RTalusScale*{65,-50,-50}*RTalusU
RTAL8 = $RTalusScale*{-65,-50,0}*RTalusU
RTAL7 = $RTalusScale*{-65,50,0}*RTalusU
OUTPUT(RTAL1,RTAL2,RTAL3,RTAL4,RTAL5,
RTAL6,RTAL7,RTAL8)
```

This section creates virtual points which are used to draw a box shape to represent the talus in the Workspace window.

```
{*Define Lower Talus segment with anterior(x) axis along subtalar (RP2MT-RCCAL) line*}
RTalusL = [RSTJ,RP2MT-RCCAL,RMMAL-RLMAL,xzy]

{*Draw Subtalar axis along x-axis of Lower Talus*}
RSTJ1 = RSTJ+100*1(TalusL)
RSTJ2 = RSTJ-100*1(TalusL)
OUTPUT(RSTJ1,RSTJ2)
```

This section defines another segment which has a different origin and orientation to Upper Talus, but which does not move relative to Upper Talus.

```
{* Calcaneus *}
{* ========= *}
{* This segment has 3 real (RL/RMCAL,RP2MT) markers*}

If $Static ==1
        {*Save Calcaneus scale as parameter*}
        $RCalcaneusScale = ((DIST(RP2MT,RCCAL)$MarkerDiameter)/3)/100
        PARAM($RCalcaneusScale)
EndIf

{*Define Calcaneus segment with anterior(x) axis along subtalar (P2MT-CCAL)
line*}
RCalcaneus = [RCCAL,RMCAL-RLCAL,RP2MT-RCCAL,yzx]
```

This segment is defined with its origin at the point RCCAL (mid-point of calcaneus markers).

```
{*Apply rotation offset parameters*}
Calcaneus = ROT(Calcaneus,2(Calcaneus),$CalcaneusFlexion)
Calcaneus = ROT(Calcaneus,3(Calcaneus),$CalcaneusAbduction)
Calcaneus = ROT(Calcaneus,1(Calcaneus),$CalcaneusInversion)

{*Draw Calcaneus, using 8 scaled vertices*}
RCAL2 = $RCalcaneusScale*{210,45,10}*RCalcaneus
RCAL1 = $RCalcaneusScale*{210,-45,10}*RCalcaneus
RCAL4 = $RCalcaneusScale*{-50,
-55,110}*RCalcaneus
RCAL3 = $RCalcaneusScale*{-50,55,110}*RCalcaneus
RCAL6 = $RCalcaneusScale*{235,55,-45}*RCalcaneus
RCAL5 = $RCalcaneusScale*{235,-55,
-45}*RCalcaneus
RCAL8 = $RCalcaneusScale*{-90,-95,
-15}*RCalcaneus
RCAL7 = $RCalcaneusScale*{-90,95,-30}*RCalcaneus
OUTPUT(RCAL1,RCAL2,RCAL3,RCAL4,RCAL5,
RCAL6,RCAL7,RCAL8)

{* Midfoot *}
{* ======= *}
{* This segment has 3 real (RD5MT,RP/RD2MT) markers*}

{*Define Midfoot segment with lateral (y) axis along metatarsal (RD5MT-
RD2MT) line*}
RMidfoot = [RD2MT,RD2MT-RP2MT,RD5MT-RD2MT,xzy]
```

```
If $Static == 1
          {*Save Midfoot scale as parameter*}
          $RMidfootScale = DIST(RD2MT,RP2MT)/100
          PARAM($RMidfootScale)
EndIf

{*Draw Midfoot, using 8 scaled vertices*}
RMDF2 = $RMidfootScale*{8,-84,0}*RMidfoot
RMDF1 = $RMidfootScale*{8,16,0}*RMidfoot
RMDF4 = $RMidfootScale*{8,-84,15}*RMidfoot
RMDF3 = $RMidfootScale*{8,16,20}*RMidfoot
RMDF6 = $RMidfootScale*{-112,-111,-74}*RMidfoot
RMDF5 = $RMidfootScale*{-112,-11,-74}*RMidfoot
RMDF8 = $RMidfootScale*{-112,-111,-49}*RMidfoot
RMDF7 = $RMidfootScale*{-112,-11,-44}*RMidfoot

RMTJ1 = (RD5MT+RD2MT)/2+$RMidfootScale*(RD5MT-RD2MT)
RMTJ2 = (RD5MT+RD2MT)/2-$RMidfootScale*(RD5MT-RD2MT)

OUTPUT(RMDF1,RMDF2,RMDF3,RMDF4,RMDF5,RMDF6,RMDF7,RMDF8)

{*Draw Metatarsal joint axis*}
OUTPUT(RMTJ1,RMTJ2)


{* Hallux *}
{* ====== *}
{* This segment has 1 real (RHLX) marker*}

{*Apply toe width offset*}
RD1MT=RD2MT-$ToeWidth*2(RMidfoot)

{*Define Hallux segment with lateral (y) axis aligned with Midfoot y-axis*}
RHallux=[RHLX,-2(RMidfoot),RHLX-RD1MT,yxz]

If $Static ==1
          {*Save Hallux scale as parameter*}
          $RHalluxScale = DIST(RHLX,RD2MT)/150
          PARAM($RHalluxScale)
EndIf

{*Reposition Hallux, maintaining attitude*}
%RDHLX = {-5*$RHalluxScale,0,0}
RDHLX = %RDHLX*RHallux
RHallux = RDHLX+Attitude(RHallux)


{*Draw Hallux, using 8 scaled vertices*}
RHLX2 = $RHalluxScale*{-20,70,40}*RHallux
```

```
RHLX1 = $RHalluxScale*{-15,-20,0}*RHallux
RHLX4 = $RHalluxScale*{-15,10,80}*RHallux
RHLX3 = $RHalluxScale*{-20,70,80}*RHallux
RHLX6 = $RHalluxScale*{-30,70,40}*RHallux
RHLX5 = $RHalluxScale*{-20,-20,-20}*RHallux
RHLX8 = $RHalluxScale*{-40,-10,80}*RHallux
RHLX7 = $RHalluxScale*{-40,70,80}*RHallux
OUTPUT(RDHLX,RHLX1,RHLX2,RHLX3,RHLX4,RHLX5,
RHLX6,RHLX7,RHLX8)
```

Angle Output Section

```
{* Angle Outputs *}
{* ============= *}

{*Calculate joint angles, using Grood&Suntay sequence -
flexion,abduction,rotation*}
RANKA = <RTalusU,RTibia2,yxz>
RSTLA = <RCalcaneus,RTalusL,yxz>
RMDFA = <RMidfoot,RCalcaneus,yxz>
RHLXA = <RHallux,RMidfoot,yxz>

{*Output angles for plotting and saving*}
OUTPUT(RANKA,RSTLA,RMDFA,RHLXA)
```

The use of the segment RTibia2 in the definition of RANKA means that the two segments in the angle definition have a common axis, the intermalleolar axis, so the relative rotation of the two segments is forced to be around this axis. Only one component of RANKA should have a significantly non-zero value, as a result.

Left Foot Script

The rest of the script is a duplicate of the above, with the replacement of L (for Left) prefixes in all labels, and with some defining lines changed to maintain right-handed systems. For example, the tibia segment definition is:

```
LTibia = [LAJC,LKJC-LAJC,LLMAL-LMMAL,zxy]
```

The order of points in the second defining line being the reverse of that in the corresponding right foot definition.

Kinematic Principles and Limitations

The kinematic parts of this model use a minimal marker set to model the pelvis and lower extremities using seven segments.

The pelvis segment is defined using three markers, one on each anterior superior iliac spine, and one on the back at the same height as the posterior iliac spines. A simple hip centre estimation model generates hip centres relative to the pelvis segment.

The thigh segments are defined using the knee and mid-thigh markers plus the estimated hip centres. The knee marker should be placed on the lateral epicondyle, and the mid-thigh marker should be placed on the thigh, just below the hand swing level, and in the plane containing the hip centre and knee flexion axis.

The tibial segment is defined in a similar way, with an ankle marker on the lateral malleolus, and a mid-tibia marker placed in the plane defined by the knee centre and ankle flexion axis.

The foot, although technically a segment, does not have rotations about three axes fully determined. Only one additional marker is used to define the foot segment, meaning that ankle flexion and rotation can be measured, but ab/adduction (pronation/supination in foot terminology) are not measured.

This marker set is widely used in clinical gait analysis, and similar modelling methods are used in VICON Clinical Manager (VCM) and Plug in Gait gait analysis software. However, there are important differences. The model within VCM is fixed and cannot be adjusted, while this model can be edited. VCM modifies the timebase of the data to match the gait cycle, which this model does not do. In this process of timebase modifying, VCM interpolates between data points, which this model does not. Also, VCM uses a static trial to determine certain parameters, while this model is only able to process walking data. This model will not, therefore, replicate the results of VCM precisely.

In addition, it should be noted that while the models used in gait analysis software products such as VCM are fixed, the model described here is open to editing. Also, while the model in VCM has been verified by widespread testing and use, any reproduction of similar methods in BodyLanguage will not be so well verified. The intention of this model is to illustrate modelling methods, and to provide a starting point for further developments. It should not be used for clinical gait analysis.

Force Vectors

```
{*VICON BodyLanguage (tm)*}

{*For use only with BodyBuilder Version 3.5, or higher*}
```

```
{* Show the force plates *}
if EXIST( ForcePlate1 )
        Force1 = ForcePlate1(1)
        Moment1 = ForcePlate1(2)
        Centre1 = ForcePlate1(3)
```

The quantity ForcePlate1 is a Reaction. If it exists (ie. there is data from a force plate), the quantities Force1 , Moment1 and Centre1 are defined equal to the components of the reaction.

```
        if ( ABS ( Force1 ) > 10 )
        Point1 = Centre1 + {-Moment1(2)/Force1(3),
                Moment1(1)/Force1(3), -Centre1(3) }
        else
                Point1 = Centre1
        endif
```

Point1 is defined as the centre of the force plate plus an amount equal to the centre of pressure location relative to the centre of the plate - that is, Point1 is the centre of pressure.

```
        Force1 = Force1 + Point1
        OUTPUT ( Point1, Force1, Centre1 , Moment1 )
endif

{*
if EXIST( ForcePlate2 )
        Force2 = ForcePlate2(1)
        Moment2 = ForcePlate2(2)
        Centre2 = ForcePlate2(3)
        if ( ABS ( Force2 ) > 5 )
                Point2 = Centre2 + { -Moment2(2)/Force2(3),
Moment2(1)/Force2(3), -Centre2(3) }
        else
                Point2 = Centre2
        endif
        Force2 = Force2 + Point2
        OUTPUT ( Point2, Force2, Centre2 )
endif
*}
```

Segment Definitions

```
{*Pelvis*}
{*======*}

PELO=(LASI+RASI)/2

        Pelvis=[PELO,LASI-RASI,PELO-SACR,yzx]
```

The Pelvis segment is defined using the markers LASI , RASI and SACR if all three are present. The right-to-left direction between the ASIS markers is the second axis of the segment. The plane of all three markers is the plane containing the first and second axes, the first axis pointing in an anterior direction.

```
{*Hip Joint Centres are positioned within the
Pelvis segment by user-set offsets, scaled
by inter-ASIS distance*}

InterASISDist=DIST(LASI,RASI)
LHJC=(InterASISDist*$%LHipOffsetFactor)*Pelvis
RHJC=(InterASISDist*$%RHipOffsetFactor)*Pelvis
```

The HipOffsetFactor parameter can be set by the user according to the anatomical model they prefer. It can be seen that in this case, the crucial biomechanical definition is made by setting a parameter, rather than by re-positioning markers as in the foot model.

```
{*Pelvic Tilt can be adjusted*}
Pelvis=ROT(Pelvis,2(Pelvis),$PelvisTilt)

{* Add its mass etc. *}
Pelvis = [Pelvis, BodyMass*0.142, {0,0,0}, {0,0,0}]

{*Femora*}
{*======*}

KneeOffset=($MarkerDiameter+$KneeWidth)/2

   LKJC=CHORD(KneeOffset,LKNE,LHJC,LTHI)
   RKJC=CHORD(KneeOffset,RKNE,RHJC,RTHI)

   LFemur=[LKJC,LHJC-LKJC,LTHI-LKJC,zxy]
   RFemur=[RKJC,RHJC-RKJC,RKJC-RTHI,zxy]
```

The femur segment definition uses the CHORD function to locate the knee centre. This virtual point is then used as the origin in the femur segment definition.

```
{*Femoral Rotation can be adjusted*}

LFemur=ROT(LFemur,3(LFemur),$LFemurRotation)
RFemur=ROT(RFemur,3(RFemur),$RFemurRotation)
```

The attitude of the femur segment can be adjusted using a thigh rotation parameter, allowing the user to compensate for mis-positioning of the thigh marker.

```
LFemurLength=DIST(LKJC,LHJC)
RFemurLength=DIST(RKJC,RHJC)

{* Add the hierarchical and inertial characteristics *}
LFemur=[LFemur, Pelvis, LHJC,
         0.1*BodyMass, LFemurLength*{0,0,0.567},
         LFemurLength*LFemurLength*{0,0.292,0.292}
         *0.1*BodyMass ]
RFemur=[RFemur, Pelvis, RHJC, 0.1*BodyMass,
RFemurLength*{0,0,0.567},
RFemurLength*RFemurLength*{0,0.292,0.292}
         *0.1*BodyMass ]

{*LFCoM = LFemurLength*{0,0,0.567} * LFemur
OUTPUT( LFCom )*}

{*Tibiae*}
{*======*}

AnkleOffset=($MarkerDiameter+$AnkleWidth)/2

   LAJC=CHORD(AnkleOffset,LANK,LKJC,LTIB)
   RAJC=CHORD(AnkleOffset,RANK,RKJC,RTIB)

   LTibia=[LAJC,LKJC-LAJC,LTIB-LAJC,zxy]
   RTibia=[RAJC,RKJC-RAJC,RAJC-RTIB,zxy]
```

The tibial segments are defined similarly to the femur segments.

```
{*Tibial Rotation can be adjusted*}

LTibia=ROT(LTibia,3(LTibia),$LTibiaRotation)
RTibia=ROT(RTibia,3(RTibia),$RTibiaRotation)

LTibiaLength=DIST(LAJC,LKJC)
RTibiaLength=DIST(RAJC,RKJC)

{* Add the hierarchical and inertial characteristics *}
LTibia=[LTibia, LFemur, LKJC,
0.0465*BodyMass, LTibiaLength*{0,0,0.567},
LTibiaLength*LTibiaLength*{0,0.279,0.279}
         *0.0465*BodyMass ]
RTibia=[RTibia, RFemur, RKJC, 0.0465*BodyMass, RTibiaLength*{0,0,0.567},
RTibiaLength*RTibiaLength*{0,0.279,0.279}
         *0.0465*BodyMass ]

{*Feet*}
{*====*}
```

```
LFoot=[LTOE,LTOE-LAJC,LKJC-LAJC,xyz]
RFoot=[RTOE,RTOE-RAJC,RKJC-RAJC,xyz]
```

The foot segments are defined using the line between the knee and ankle joint centres, with the result that the foot segment does not have three rotational degrees of freedom relative to the tibia.

```
{*Apply Static Trial Offsets*}

LFoot=ROT(LFoot,2(LFoot),-$LStaticPlantarFlexion)
RFoot=ROT(RFoot,2(RFoot),-$RStaticPlantarFlexion)

{* Add the hierarchical and inertial characteristics *}
LFoot=[LFoot, LTibia, LAJC, 0.0145*BodyMass, DIST(LTOE,LAJC)*{0.5,0,0},
DIST(LTOE,LAJC)*DIST(LTOE,LAJC)*{0,0.226,0.226}*
0.0145*BodyMass]
RFoot=[RFoot, RTibia, RAJC, 0.0145*BodyMass, DIST(RTOE,RAJC)*{0.5,0,0},
DIST(RTOE,RAJC)*DIST(RTOE,RAJC)*{0,0.226,0.226}*
0.0145*BodyMass]
```

Definition and Output of Rotations

```
{*Angles*}
{*======*}

PELA=<Pelvis,1,yxz>
PELA=<1(PELA),2(PELA),3(PELA)>
LHPA=<Pelvis,LFemur,yxz>
LHPA=<1(LHPA),2(LHPA),3(LHPA)>
RHPA=<Pelvis,RFemur,yxz>
RHPA=<1(RHPA),-2(RHPA),-3(RHPA)>
LKNA=<LFemur,LTibia,yxz>
LKNA=<-1(LKNA),2(LKNA),3(LKNA)>
RKNA=<RFemur,RTibia,yxz>
RKNA=<-1(RKNA),-2(RKNA),-3(RKNA)>
LANA=<LTibia,LFoot,yxz>
LANA=<1(LANA),-2(LANA),3(LANA)>
RANA=<RTibia,RFoot,yxz>
RANA=<1(RANA),-2(RANA),-3(RANA)>

OUTPUT(PELA,LHPA,RHPA,LKNA,RKNA,LANA,RANA)

{*Segment Axes*}
{*============*}
PEL1=PELO+50*1(Pelvis)
PEL2=PELO+50*2(Pelvis)
PEL3=PELO+100*3(Pelvis)

LFEO=O(LFemur)
```

```
LFE1=LFE0+50*1(LFemur)
LFE2=LFE0+50*2(LFemur)
LFE3=LFE0+LFemurLength*3(LFemur)
RFE0=0(RFemur)
RFE1=RFE0+50*1(RFemur)
RFE2=RFE0+50*2(RFemur)
RFE3=RFE0+RFemurLength*3(RFemur)

LTI0=0(LTibia)
LTI1=LTI0+50*1(LTibia)
LTI2=LTI0+50*2(LTibia)
LTI3=LTI0+LTibiaLength*3(LTibia)
RTI0=0(RTibia)
RTI1=RTI0+50*1(RTibia)
RTI2=RTI0+50*2(RTibia)
RTI3=RTI0+RTibiaLength*3(RTibia)

LFO0=0(LFoot)
LFO1=LFO0+50*1(LFoot)
LFO2=LFO0+50*2(LFoot)
LFO3=LFO0+100*3(LFoot)
RFO0=0(RFoot)
RFO1=RFO0+50*1(RFoot)
RFO2=RFO0+50*2(RFoot)
RFO3=RFO0+100*3(RFoot)

{*Virtual Points Output*}
{*=====================*}
OUTPUT(PEL0,PEL1,PEL2,PEL3,LHJC,RHJC)
OUTPUT(LFE0,LFE1,LFE2,LFE3,RFE0,RFE1,RFE2,RFE3,LKJC,RKJC)
OUTPUT(LTI0,LTI1,LTI2,LTI3,RTI0,RTI1,RTI2,RTI3,LAJC,RAJC)
OUTPUT(LFO0,LFO1,LFO2,LFO3,RFO0,RFO1,RFO2,RFO3)
```

These virtual points are defined to make the segments visible in the Workspace window, when a suitable MKR file is selected.

```
r3 = REACTION( LFemur )

FP1 = | ForcePlate1(1), ForcePlate1(2), ForcePlate1(3) |
CONNECT( LFoot, FP1, 1 )
        {* ABS( ForcePlate1(1) ) > 5 ) *}

%r2 = REACTION( LFoot )
r2 = %r2 * LFoot

Force = r2(1)
Moment = r2(2)
Centre = (r2(3) - { Moment(2)/Force(3), -Moment(1)/Force(3), 0 })
Force = Force + Centre
```

```
M2 = %r2(2)/(BodyMass)

output( Force, Centre, M2 )

P2 = { POWER( Pelvis, LFemur )/BodyMass, POWER( LFemur, LTibia
)/BodyMass,POWER( LTibia, LFoot )/BodyMass }
output( p2 )
```

Acclaim
File formats .AST, .ASF and .AMC were devised by Acclaim Corp. for the description of kinematic models and motion data.

BodyLanguage
The language in which kinematic models for BodyBuilder are written.

Browse box
Also called a Browser. A small window which allows rapid search of directory trees to find particular files.

Cartesian frame of reference
A system of three perpendicular axes which meet at an origin. The position of any point relative to these axes can be specified by an ordered triplet of numbers.

Current field
The field displayed in the Workspace window. The number of the current field is shown under the replay slide bar. May be referred to as Current Frame or Current Sample.

Dongle
A hardware key which must be plugged into a computer before BodyBuilder can be run. Supplied to licensed users by Vicon Motion Systems.

Field
A time-sample of kinematic data. In standard video terminology, a "frame" is composed of one even- and one odd-numbered field. In kinematic modelling, as in cine film, the distinction between even and odd fields is lost, and the word "frame" is often used interchangeably with "field".

Gait analysis
The study of human movement for medical purposes.

Global point
A point which is defined in terms of the VICON reference axes, which are constant throughout a trial.

Interpolation
Filling a gap in a trajectory by inventing new points. Small gaps in smooth trajectories may be interpolated reliably. Large gaps and trajectories which are not smooth are more difficult to interpolate, and the results may be unrealistic. Interpolation should be used with caution.

Inverse Kinematics
Deducing the movements of segments from the desired overall result rather than measured motion. For example, deducing the movements of leg segments from the requirement that the feet do not move relative to the floor. This is an inverse problem without a unique solution.

Kinematic angles
Numerical description of the angular relationship between connected body segments of a kinematic model. Stored as three numbers.

Kinematic model
A numerical description of a moving object, composed of segments and links.

Label
A name by which a point or trajectory is identified. Limited to 16 characters.

Local point
A point defined in terms of the axes of a particular segment, which is itself able to move during the trial.

Marker
A reflective ball fixed to the performer. Also called a real marker, or physical marker, to distinguish between trajectories which result from actual measurements, and virtual ones created by modelling.

Operating system
Software which controls the operation of a computer. BodyBuilder requires a 32-bit operating system, which may be either Windows NT (preferred) or Windows 95.

Passband
The range of frequencies allowed through a filter. Motion at a nearly constant speed has low frequency components; motion involving high accelerations has high frequency components which may fall outside the passband of a filter.

Point
A location in space, specified by 3D co-ordinates. A trajectory (or segment of a trajectory) consists of a time-series of points. A point is stored in a .C3D file as three spatial co-ordinates and a residual, identified by a label. Points may represent the measured positions of real markers, or may be virtual (created by modelling). "Point" and "marker" are often used interchangeably.

Pointer
Screen symbol (by default, an arrow) which is controlled by moving the mouse.

Reconstruction
The calculation of the position of a marker by a VICON system.

Script
A set of BodyLanguage definitions and functions which can be run by BodyBuilder. Scripts are contained in .MOD files.

Segment
In a kinematic model, body parts are represented by segments, which are assumed to be rigid elements linked by joints. Segments have both position and

orientation. (The term "trajectory segment" is sometimes used to describe a short or interrupted marker trajectory.)

Spike
A point which lies to one side of an otherwise smooth trajectory. Spikes can occur as a result of a mis-resconstruction in a single field, perhaps caused by video noise or marker obscuring.

Static trial
A short capture during which the performer stands still. Used for creating Acclaim skeleton files and for deriving subject-specific parameters.

Stick
A line connecting two points as displayed in the Workspace window. Defined in the current .MKR file, sticks are a useful graphic aid, but have no significance above that of the points they link.

System memory
The working memory (RAM) of a computer. Data and programs in current use are held in system memory, and will be lost if the computer is shut down before they are written to disk.

Toggle
A command, button or key which causes a particular display, function or feature to change between its two possible states.

Trajectory
The path through space followed by a marker, whether real or virtual. Stored in a .C3D file as a time-series of points with the same label. May consist of several trajectory segments separated by gaps, or a single, uninterrrupted trajectory, depending on marker visibility. Displayed in a Workspace window as a line through the position of the marker in the current field.

Trial
A single data capture. The word also refers to the .C3D file resulting from a data capture.

Virtual point
A point produced by kinematic modelling, in a position where no real marker existed during motion capture. Virtual points may represent internal features such as joint centres, where it is impossible to place a real marker.

VICON
MOTION SYSTEMS

INDEX

A
Acclaim skeleton file 58
add point 41

B
batch mode 58

C
clinical gait analysis 76, 96, 133
copy pattern 46
create new trajectory 23, 31, 43

D
defragment 31, 44
delete model output 31, 44, 57
dongle 09, 140

E
error code 122
expand/reduce 50

F
fill gap 44-45
filter characteristics 52

G
gimbal lock 102
global axes 87, 96
global rotation 96

H
hardware key 09
helical vector 95
hinge joint 70, 74, 89, 123

J
joint centre 62, 73, 84

K
kinematic model 16, 68, 75, 141

L
labels list 20-22

M
merge 30
MKR file 16, 20, 28
mouse helper 15, 16, 19, 64
move point 47, 50

N

numerical constants	75

O

object names	81
occlusions	119
ordered triplet	87

P

point dragging	27
preferences	37

R

range to save	32
real points	75, 80-81, 119
relabelling	21
replay	17, 22, 25, 56
re-sampling	53
reserved names	82
right-hand rule	87-88
rotation centre	55
run model	57

S

segment definition	87
simple mode	47
snip	39
spike	53
spine	117
stick figure	56

T

text concatenation	120
tiled floor	17
time scale	24
toolbar	55

U

unlabel	22

V

VICON Clinical Manager	123, 133

W

weighted A mode	48
weighted B mode	50
Windows	15
write ASCII	35
write C3D	33

Z

zoom 18-19