Brief description of the programs and their function

The *Anz/Telio/Show3d* package is intended to used for research as well as the clinical evaluation of human locomotion. The three programs work in concert with one another to provide a wide array to tools.

Anz

Anz carries the analysis of measured marker trajectories to produce estimates of spatial human limb kinematics. It can also combine this information with force plate data to evaluate the net joint resultant loads and power transfers. Muscle EMG data may also be processed. The program is command line based meaning that you must type in the commands to make it do anything. When running under VMS on a VAX, it can be used repeatedly via command files to yield a production environment for clinical gait evaluation. The result is an almost "handsoff" evaluation of normal and most abnormal human gaits. You can carry out a few or many of the analyses available. There many general kinematics and kinetics procedures are included as well as a number of gait specific evaluations. In a number of evaluations the research nature of the program is apparent because several methods are provided to do the same thing. When the user types the command to execute an analysis, the program checks whether the needed information for an analysis is present and if so carries out the calculation. Otherwise, a error message (hopefully informative) is displayed. The results of the analyses are stored in a single binary format file. The contents of this file can be examined from within *Anz* or *Telio*, which is used to graphical display the information. Results may also be exported in text format so that other programs can further process the information. The commands used to produce the analysis results are stored in the *Anz* binary file when it is saved so that at a later date the user(s) may be able to see the exact calculation process that was used to produce the information. Further details can be found in the *Anz* Introduction document.

Telio

Telio (based upon the word teleology) is a command line driven program to graphically display the results of *Anz* analyses. It can also read and display several data file formats native to the VICON marker tracking system. The program has a very flexible display format so that screen displays and/or printouts of many types can be produced. Two basic display types are available: graphs and stripcharts. Graphs exist as independent sections on the screen or page whereas stripcharts must have a common x axis. Graphs are better for presenting dissimilar information together whereas stripcharts work well with time series data such as EMG, force plate, and joint angle data together. Data may be read in from a number of files at once and numerous graphs or stripcharts containing a number of data sets can be displayed at once. Doing this can take a number of typed commands, so a macro facility has been included in *Telio*. This allows a series of commands to be played back with only a couple of keystrokes. This is especially useful for repetitive displays or printouts such as clinical gait. This can be combined with a command file in VMS to produce printouts of data without any interaction between user and program. The program is both an exploratory tool to look at data and a production to tool to dump data into printed form quickly. A number of gait specific features have been included to make *Telio* more functional in the clinical gait context. Presently there is no manual for *Telio*, but a

command summary document is available in the documentation. An overview will be written eventually. The best approach to seeing how *Telio* operates is to look over the macro and command files along with the printouts and then play around with it using the sample data files. NOTE: Telio uses the DI3000 graphics subroutine library from Precision Visuals Inc. to provide device independent graphics capabilities. This package must be installed for Telio to operate. The source code must be modified if another graphics package is to be used.

Show3d

Show3d generates three dimensional views of the human locomotion data and marker trajectories analyzed by *Anz*. It is a command line driven program which can produce views of the analyzed motion from any direction. The body segments are represented by parallelapipeds to emphasize nonsagittal plane motions. Segments may be represented with or without hidden lines removed. The program uses a hidden line removal algorithm derived to handle situations where segments interpenetrate one another. A number of customization options are available including the display of the images on screen and/or via printed form. This program provides useful information for rapid evaluation of a set of data and the results are impressive to the layperson.. However, it was never developed to the level that was originally intended so there may be some lingering bugs and/or design faults. NOTE: This program also uses the DI3000 graphics package

What's included here...

The *Analyze* gait analysis and display package consists of three programs:

1) *Analyze*, usually referred to as *Anz*, which carries out the analysis of the marker trajectories, force plate data, and EMG data.
2) *Telio* which produces graphs of the results of Anz analyses
3) *Show3d* whcih displays 3d images of the measured motions

The code is written in VAX Fortran and was orginally developed to run under VMS. The VAX version of this package contains not only the source code, but also the executable images of the programs. These have been built with the DEC Module Management System (which is similar to the UNIX make program) and have the maximum number of frames of data set to 150 marker frames and 1500 electrical data frames (force plate and EMG). A sample marker, force plate, and EMG data file from the VICON motion system produced using AMASS is included for testing with the VAX tape release of the system. In addition some gait data that has been either completely or partially analyzed is included for experimenting with in all three programs.

The Macintosh version of this package contains only the source files along with the rest of the text files. The executables and binary data files are not included. *Telio* and *Show3d* were never intended for anything but a VAX while *Anz* was built using Language Systems Fortran on a Macintosh as an experiment. Exchanging binary data between the Mac and VAX is no fun and as a result none is included. I did it, but only by first reading the data files with *Anz* on the VAX and saving a .anz file which I transfered to the Mac. *Anz* can work on a Mac but it needs serious changes to work well there. The use of Macintosh disks to distribute this package is exclusively for convenience.

Same goes for the PC disk version. The same files as are in the Macintosh version are included. Unfortunately, the file names have had to be truncated to 8 characters because of the silly file limitations of DOS. Anz has never been built on a PC. See further notes on the PC disks about the file name changes.

Organization

The files that are part of the *Anz* gait analysis and display software package are separated into three parts corresponding to the three programs. These files are contained in the Anz, Telio, and Show3d subdirectories.

What the file's are...

There is a MANIFEST file with a listing of the files in the subdirectories.

The file extensions indicate the following:

.for	the fortran source code files
.inc	the include files for the fortran source- mainly common blocks & constants
.mms	the Module Management System files to direct building of the programs
.anz	data files containing the results of Anz analyses- it's a binary file
.mac	macro files for use with Telio to automate graph generation

TERMS OF USE FOR ANZ, TELIO & SHOW3D

This program is copyrighted but may be freely distributed and various sections used and/or modified by others under the following restrictions

1) When this code or some part is used, proper attribution to the developers of this code- Dwight Meglan and the Gait Analysis Lab of The Ohio State University- must be given.

2) If you make changes to the code and redistribute it subsequently, these changes must be clearly described and marked. In addition, modified versions of these programs must not be given the same names.

3) This code must be provided free of charge if it is redistributed, except for recovery of the cost of distribution.

4) NO part of this code may be included in any commercial software. It may NOT be provided as an addon to a commercial product by the manufacturer or distributor, even if no charge is made for it. Licensing to distribute in this manner may be arranged by contacting Dwight Meglan.

5) No profit should be derived from the use of this code. If it is used in a clinical context, then cost recovery fees may be charged by the user(s).

WARRANTY - THERE IS NONE !!

I, and The Ohio State University, take NO responsiblity for the accuracy of the numerical calculations in this code. I accept NO liability for any difficulties resulting from the use of the results of this code.

SORRY for the heavy handed language but in this crazy world of finger pointing lawsuits, you just never know what will happen...

This code is UNSUPPORTED. It has been used extensively in both clinical and research contexts and I have tried to find all the bugs, but with something this large there are bound to be some latent problems. I am interested in hearing about uses of the code by others, improvements, or problems identified. I cannot help with debugging or teaching others about what is in here. This is intended to be used as a teaching tool for those interested in studying human locomotion kinematics and kinetics primarily, not as a finished clinical product. In reality it is being used as such and I am confident that it carries out this role well, but I do not have the time or desire to support this set of programs as would be required for a clinical product .

Comments can be sent to: Dwight Meglan, meglan@mayo.edu

Address should be valid until at least Fall 1992. I would prefer email contact because of the potential for people calling me inspite of all the disclaimers above.

ANZ Command Summary

[]	indicates default value
**	indicates default active option
(#)	indicates number required with option

Section	Command	Option(s)
AVeraGe	POSition	
	ANGle	
	REFerence	/FILe=filename file with reference position data
CYCle	EVEent	
	STAtistics	/use ANKle marker to calculate step stats [no]
		/use HEEl marker to calculate step stats [no]
		/use TOE marker to calculate step stats [no]
EMG	NAMe	
	FILter	/apply hanning WINdow to data **
		/No WiNdowing of data
		/LOW frequency cutoff (# hz)
		/Low frequency cutoff SLope (# %/hz)
		/HIGh frequency cutoff (# hz)
		/High frequency cutoff SLope (# %/hz)
		/NOTch frequency center (# hz)
		/Notch WIdth (# hz)
		/Notch frequency side SLope (# %/hz)
	RECtify	
	INTegrate	/WINdow width (# msec) [50.0 msec]
	MoveWinAvg	/WINdow width (# msec) [50.0 msec]
		/use COSine taper on window [no]
	THReshold	/PERcent (# %) [20.0 %]
FoRCe	CONdition	/NO Zeroing of force plate data [no]
		/variation allowed in IDLe region (# % fz) [5.0 %]
		/minimum # of FRaMes in idle region (#) [10]
		/FZ Minimum plate active threshold (# N) [10]
		/fz in IDle region Maximum level (# N) [10]

ALIgn /manually assign Right FooT plates (#) [none]
/manually assign Left FooT plates (#) [none]
/frame of Heel Strike on plate 1 (#) [none]
/frame of Heel Strike on plate 2 (#) [none]
/frame of Heel Strike on plate 3 (#) [none]
/first force plate hit by SECond heelstrike [no]
/FZ Minimum active theshold level (# N) [5.0]
/Foot Strike Threshold (# % cycle) [10.0]
/change in fz TRIgger for heelstrike (# % fz) [10.0]
/CHAnge in fz level detection of heelstrike **
/ABSolute value of fz level detection of heelstrike [no]
/MZ moment Zero on plate 1 [no]
/use 2ND foot heelstrike & GRL data to find frame offset [no]

CLIp force/motion data

Center of PRessure

Local Center Pressure

WREnch

INTegral /estimate Body Center of Mass path [no]
/set inital bcm VeLocity of Zero [no]

REConstructed grl

Pitch Moment Arm

Pitch Wrench Arm

Center of Pressure Velocity

FILter /apply hanning WINdow to data **
/No WiNdowing of data
/LOW frequency cutoff (# hz)
/Low frequency cutoff SLope (# %/hz)
/HIGh frequency cutoff (# hz)
/High frequency cutoff SLope (# %/hz)
/NOTch frequency center (# hz)
/Notch WIdth (# hz)
/Notch frequency side SLope (# %/hz)

HIStory

JoiNT Jcs ANgle /use REFerence joint angles [no]
/modify ankle Ankle FLexion (# deg) [no, -6.0]
/No Ankle Flexion modification **
/use SHoulder angle Polar definition [no]
/Extrapolation THreshold using REF (# deg) [10.0]
/FILe=filename file with reference angle data
/INTegrate joint angle velocities to get joint angles

Jcs Vel Accel /use SIMple differentiation of JCS angles [no]
/GeneralizedCrossValidation **
/DynamicProgrammingFilter [no]
/NOIse added % of signal range (# %) [0.5]
/NO Noise added to signal for dpf
/VARiability threshold for no smoothing (# %) [1.0]

Euler ANgle /modify ankle Ankle FLexion (# deg) [no, -6.0]
/No Ankle Flexion modification **
/use SHoulder angle Polar definition [no]

Euler Vel Acc /use SIMple differentiation of JCS angles [no]
/GeneralizedCrossValidation **
/DynamicProgrammingFilter [no]
/NOIse added % of signal range (# %) [0.5]
/NO Noise added to signal for dpf
/VARiability threshold for no smoothing (# %) [1.0]

LOAd /use FULl body calculation scheme [no]
/INTerpolate bad frames [no]
/No INterpolation [no]
/interpolation POLynomial order (#) [3]
/OVErride the bad GRL indicators near a gait event **
/NOVerride of bad GRL indicators

WRench

POWer

TRaNslation

ScrewANgle

MomentARm

WrenchARm

FILter /apply hanning WINdow to data **
/No WiNdowing of data
/LOW frequency cutoff (# hz)
/Low frequency cutoff SLope (# %/hz)
/HIGh frequency cutoff (# hz)
/High frequency cutoff SLope (# %/hz)
/NOTch frequency center (# hz)
/Notch WIdth (# hz)
/Notch frequency side SLope (# %/hz)

Filterable joint quantities: JAN JVA LOA POW

EXternal Moment

WoRK

LoadANgle

CENter /Finite SCrew axis [no]
/Instataneous SCrew axis **
/intersect screw with PLaNe= filename **
/use SPHerical motion assumption [no]

AverageCeNter /limit centers to CLIpping volume= filename
/use SEQuential motion frames for joint centers **
/use PERmutations of all available motion frames [no]

MaRKer INTerpolate /POLynomial order [3]
/EXTrapolation polynomial order [3]
ADJust

SMOoth /VelocityAndAcceleration [no]
/GeneralizedCrossValidation **
/DynamicProgrammingFilter [no]
/NOIse added % of signal range (#) [0.5 %]
/NO Noise added to signal for dpf
/VARiability threshold for no smoothing (#) [1.0%]

FILter /apply hanning WINdow to data **
/No WiNdowing of data
/LOW frequency cutoff (# hz)
/Low frequency cutoff SLope (# %/hz)
/HIGh frequency cutoff (# hz)
/High frequency cutoff SLope (# %/hz)
/NOTch frequency center (# hz)
/Notch WIdth (# hz)
/Notch frequency side SLope (# %/hz)

REGularize /use gait CYCle for start & end frames [no]
/STArt frame (#) [1]
/END frame (#) [number of motion frames]
/PoiNTs on interval (#) [100]
/interpolation POLynomial (#) [3]
/PAD points (#) [0]

OPTion GENeral LOGfile
UPDate display
NOUpdate
INTerpolating polynomial order

FoRCe number of idle FRaMes
IDLe region variation
plate active TRIgger value of fz
HeelStrike Threshold

REAd
- /TR3 override file extension check: read TR3 file format
- /TRU override file extension check: read TRU file format
- /C3D override file extension check: read C3D file format
- /ANZ override file extension check: read ANZ file format
- /GL1 override file extension check: read old GLS file format
- /GL2 override file extension check: read new GLS file format
- /FPD override file extension check: read FPD file format
- /NCZ file marker data uses new Coord system [no]
- /OGN force plate data uses old amplifier gains [no]
- /ZER zero force plate data using zeros in file [no]
- /NOC don't convert GLS2 force plate data with cal matrix [no]
- /BEG start frame for reading GLS1 and GLS2 data [1]
- /DRZ resolution decrease for reading GLS1/GLS2 (#) [no, 10]
- /UPP data in C3D file is for upper extremity study [no]
- /NOH upper extremity data does not contain head markers [no]
- /BYT reverse byte order of dat in ANZ file [no]
- /NOF don't use the calculation flags stored in the ANZ file [no]
- /NMN dont't override marker names in file [no]
- /TDP data in C3D file is for time-distance parameter calculation only [no]

SAVe
- /TRU save marker/angle data in TRU file format
- /GLS save force plate/EMG data in GLS2 file format
- /ANZ save all possible data in ANZ file format **
- /NOF don't save the calculation flags [no]

SEGment

PARameters
- /ZeroMassInertia [no]
- /ZeroNonSagittal inertias [no]
- /ZerINertia only [no]
- /STArt frame (#) [1]
- /NUMber of frames (#) [number of motion frames]
- /force use of ADUlt bsp formulas [no]
- /force use of CHIld bsp formulas [no]

MaRKer diagnostics
- /ANGle between markers rather than distance

POSition
- /use anatomic REFerence correction [no]

VelAndAccel
- /use yxz EULer angle derivatives **
- /use rotation MATrix derivatives [no]
- /use old method for ROTation angle derivs [no]
- /SMOoth the rotation matrix components [no]
- /use MaRKer vel/accel [no]
- /use RIGid body correction with MRK [no]
- /generate DIAgnostics with MRK [no]
- /GeneralizedCrossValidation **
- /DynamicProgrammingFilter [no]
- /NOIse added % of signal range (#) [0.5 %]
- /NO Noise added to signal for dpf
- /VARiability threshold for no smoothing (#) [1.0%]

ENeRgy

FiniteSCrew /use WALdron's method **
/use WOLtring's method [no]
/do ABSsolute calculation **
/do RELative calculation [no]

InstantSCrew

BodyCenMass /calculate bcm VELocity also

MOMentum

ANGle /INTegrate the segment rotational acceleration

FILter /apply hanning WINdow to data **
/No WiNdowing of data
/LOW frequency cutoff (# hz)
/Low frequency cutoff SLope (# %/hz)
/HIGh frequency cutoff (# hz)
/High frequency cutoff SLope (# %/hz)
/NOTch frequency center (# hz)
/Notch WIdth (# hz)
/Notch frequency side SLope (# %/hz)

Filterable segment quantities: TVL RVL POS VAA
TAC RAC

STAtus GENeral
PATient
MaRKer
SEGment <none>
INErtia
FoRCe <none>
AVerGe
JoiNT
AVeraGe <none>
POSition
ANGle JCS
EULer
CYCle
EMG
LIMit

Data Export Commands

```
EXPort    AVeraGe      POSition/orientation
/IGOr                      Jcs ANgle
/NOHeader                  anatomic REFerence correction
/INC=#

EXPort    gait CYCle

EXPort    EMG

EXPort    FoRCe        GroundReactionLoad { plt #, comb (/ANI), ft ([/GCS],/LCS) }
                                            (/NWT,/NWH,/NWL)
                                            /LBS
                           CenterPRessure      { plt #, comb, ft }
                           LocalCenterPressure
                           WREnch              { plt #, comb, ft }
                           INtegral1st         { plt #, comb, ft }
                           INtegral2nd         { plt #, comb, ft }
                           StrikeIndex
                           PitchMomentArm      (/GCS,[/PRO],/DIS)
                           PitchWrenchArm      (/GCS,[/PRO],/DIS)
                           CenterPressureVelocity
EXPort    HIStory
EXPort    JoiNT        JcsANgle
                           JcsVelocity&Acceleration    /NOR
                           EulerANgle
                           EulerVelocity&Acceleration  /NOR
                           LOAd      (/NWT,/NWH,/NWL)
                                     /LBS
                                     ([/JCS],/GCS,/PRO,/DIS)
                           LegLoaD   (/NWT,/NWH,/NWL)
                                     /LBS
                                     ([/JCS],/GCS)
                           WREnch
                           POWer     (/NWT,/NWH,/NWL)
                                     /LBS
```

```
                    LegPoWer    (/NWT,/NWH,/NWL)
                                /LBS

                    FiniteSCrew

                    InstantaneousSCrew

                    ScrewANgle  ([/JCS],/PRO,/DIS)

                    TRaNslation     (/JCS,[/GCS],/PRO,/DIS)

                    MOmentArm       (/GCS,[/PRO],/DIS)

                    WRenchArm       (/GCS,[/PRO],/DIS)

EXPort  MaRKer  POSition    ([/GCS],/LCS)
                    DIStance
                    VelocityAndAccel

EXPort  SEGment PaRaMeter   /ANImation

                    POSition    /MATrix
                                /ANImation

                    VELocity        ([/GCS],/LCS)

                    ACCeleration    ([/GCS],/LCS)

                    ENeRgy

                    FiniteSCrew

                    InstantaneousSCrew

                    MOMentum

                    BodyCenterMass

                    BodyCenterMassVelocity

                    ANGle

                    SWIvel3D

EXPort  REConstructed   GroundReactionLoad  { com, ft ([/GCS],/LCS) }

                        CenterPRessure      { com, ft ([/GCS],/LCS) }

                        StrikeIndex
```

Telio Command Summary

Commands for graph and stripchart display modes in Telio

Coordinate System names:

Acronym	Description
GCS	Lab global coordinate system
LCS	Segment local coordinate system
DIS	Distal segment local coordinate system (applies to joint quantities)
PRO	Proximal segment local coordinate system (applies to joint quantities)
JCS	Joint coordinate system
EUL	Fixed eular angle sequence

Summary of variable names:

Acronym	Description	Options
mpn	marker position	
mvl	marker velocity	
mac	marker acceleration	
spn	segment position	
smp	segment marker position	
smd	segment intermarker distances	
svl	segment velocity	(/GCS or /LCS)
sac	segment acceleration	(/GCS or /LCS)
ske	segment kinetic energy	
spe	segment potential energy	
ste	segment total energy	
sis	segment instantaneous kinematic screw	
sfs	segment finite kinematic screw	
smm	segment momentum	
san	segment angle to GCS	
bke	body kinetic energy	
bpe	body potential energy	
bte	body total energy	
bcm	body center of mass position	
bcv	body center of mass velocity	
bmm	body momentum	
jan	joint angle	(/EUL or /JCS)
jvl	joint velocity	(/EUL or /JCS)
jac	joint acceleration	(/EUL or /JCS)
jld	joint load	(/PRO, /DIS, /JCS, or /GCS)
jll	joint load- leg summed	(/JCS or /GCS)
jwr	joint wrench	
jpw	joint power transfer	
jpl	joint power transfer- leg summed	
jis	joint instantaneous kinematic screw	
jfs	joint finite kinematic screw	

jtr	joint translation	(/PRO, /DIS, /JCS, or /GCS)
jsa	joint finite screw angle	(/PRO, /DIS, or /JCS)
jma	joint moment arm	(/PRO, /DIS, or /GCS)
jwa	joint wrench arm	(/PRO, /DIS, or /GCS)
fpl	ground reaction forces on force plates	
fft	ground reaction forces on feet	
fcb	ground reaction forces on combined feet	
fcp	center of pressure of GRL of force plates	
fcf	center of pressure of GRL for feet	(/LCS or /GCS)
fcc	center of pressure of GRL for combined feet	
fwp	force wrench for force plates	
fwf	force wrench for feet	
fwc	force wrench for combined feet	
fp1	first integral of GRL upon force plates	
ff1	first integral of GRL upon feet	
fc1	first integral of GRL upon combined feet	
fp2	second integral of GRL upon force plates	
ff2	second integral of GRL upon feet	
fc2	second integral of GRL upon combined feet	
fsi	strike index for feet	
pma	pitching moment arm	(/PLV, /TRK, or /GCS)
pwa	pitching wrench arm	(/PLV, /TRK, or /GCS)
fcv	velocity of center of pressure in GCS	
rft	reconstructed ground reaction forces on feet	(/GCS or /LCS)
rcb	reconstructed ground reaction forces on combined feet	
rcf	reconstructed center of pressure of GRL for feet	(/GCS or /LCS)
rcc	reconstructed center of pressure of GRL for combined feet	
rsi	reconstructed strike index for feet	
emg	emg channel signal	
mln*	muscle length	
mvl*	muscle velocity	
mor*	muscle origin position	(/LCS or /GCS)
min*	muscle insertion position	(/LCS or /GCS)
mla*	muscle line of action	(/LCS or /GCS)
mem*	muscle/IEMG measure	

* not implemented yet...

tim	time (seconds)
fra	frame number
per	percent of gait cycle or data set

Command parameter summary:

[] indicates parameters for graphing only
{} indicates parameters for 3d view only

Joint names:	rak	right ankle	lak	left ankle
	rkn	right knee	lkn	left knee
	rhp	right hip	lhp	left hip
	pvl	pelvis-lab		
	pvt	pelvis-trunk		
	nec	neck		
	rsh	right shoulder	lsh	left shoulder
	rel	right elbow	lel	left elbow
	rwr	right wrist	lwr	left wrist
Segment names:	rft	right foot	lft	left foot
	rcf	right calf	lcf	left calf
	rth	right thigh	lcf	left calf
	plv	pelvis		
	trk	trunk		
	hed	head		
	rua	right upper arm	lua	left upper arm
	rla	right lower arm	lla	left lower arm
	rhd	right hand	lhd	left hand

Acronym	1st param	2nd param	# graph param	# 3dvu param	
mpn	mrk#	[axis(x,y,z)]	2	1	
mvl	mrk#	[axis(x,y,z)]	2	1	
mac	mrk#	[axis(x,y,z)]	2	1	
spn	SegName	[axis(x,y,z)]	2	1	
[smp	SegName	mkr# axis(x,y,z)	3	-	]
[smd	SegName	dist#	2	-	]
svl	SegName	[axis(x,y,z,rx,ry,rz)] {type(rot,trn)}	2	2	
sac	SegName	[axis(x,y,z,rx,ry,rz)] {type(rot,trn)}	2	2	
[ske	SegName	comp(rot,tran,tot)	1	-	]
[spe	SegName		1	-	]
[ste	SegName		1	-	]
sis	SegName	[type(x,y,z,px,py,pz,rot,trn)]	2	1	
sfs	SegName	[type(x,y,z,px,py,pz,rot,trn)]	2	1	
smm	SegName	[axis(x,y,z,rx,ry,rz,rm,tm)] {type(rot,trn)}	2	2	
[san	SegName	axis(flex,abad,inex)	2	-	]
[bke	comp(rot,tran,tot)		1	-	]
[bpe			0	-	]
[bte			0	-	]
bcm	[axis(x,y,z)]		1	0	
bcv	[axis(x,y,z)]		1	0	
bmm	[axis(x,y,z,rx,ry,rz,rmag,tmag)] {type(rot,trn)}		1	1	

[jan	JntName	axis(fle,abd,int)	2			-	]
[jan/eul	JntName	axis(x,y,z)	2			-	]
[jvl	JntName	axis(fle,abd,int)	2			-	]
[jvl/eul	JntName	axis(x,y,z)	2			-	]
[jac	JntName	axis(fle,abd,int)	2			-	]
[jac/eul	JntName	axis(x,y,z)	2			-	]
jld/gcs,pro,dis	JntName	[axis(fx,fy,fz,mx,my,mz)] {type(rot,trn)}	2			2	
jld/jcs	JntName	[axis(mlf,apf,cdf,fem,aam,iem)]					
[jll/gcs	side(rt,lf)	axis(fx,fy,fz,mx,my,mz)	2			-	]
[jll/jcs	side(rt,lf)	axis(mlf,apf,cdf,fem,aam,iem)		2	-	]	
jwr	JntName	[type(x,y,z,px,py,pz,mom,frc)]			2	1	
[jpw	JntName		1			-	]
[jpl	side(rt,lf)		1			-	]
jis	JntName	[type(x,y,z,px,py,pz,rot,trn)]	2		1		
jfs	JntName	[type(x,y,z,px,py,pz,rot,trn)]	2		1		
[jtr/gcs,pro,dis	JntName	axis(x,y,x,mag)	2			-	]
[jtr/jcs	JntName	axis(mlt,apt,cdt,mag)	2			-	]
[jsa/pro,dis	JntName	axis(x,y,z)	2			-	]
[jsa/jcs	JntName	axis(fle,abd,int)	2			-	]
[jma	JntName	axis(x,y,z,mag)	2			-	]
[jwa	JntName	axis(x,y,z,mag)	2			-	]

fpl	Plate#	[axis(fx,fy,fz,mx,my,mz)] {type(mom,frc)}	2			2	
fft	side(rt,lf)	[axis(fx,fy,fz,mx,my,mz)] {type(mom,frc)}	2			2	
fcb	[axis(fx,fy,fz,mx,my,mz)] {type(mom,frc)}		1			1	
fcp	Plate#	[axis(x,y,z)]	2			1	
fcf	side(rt,lf)	[axis(x,y,z)]	2			1	
fcc	[axis(x,y,z)]		1			0	
fwp	Plate#	[type(x,y,z,px,py,pz,mom,frc)]	2		1		
fwf	side(rt,lf)	[type(x,y,z,px,py,pz,mom,frc)]	2		1		
fwc	[type(x,y,z,px,py,pz,mom,frc)]		1			0	
[fp1	Plate#	type(fx,fy,fz,mz)	2			-	]
[ff1	side(rt,lf)	type(fx,fy,fz,mz)	2			-	]
[fc1	type(fx,fy,fx,mz)]		1			-	]
[fsi	side(rt,lf)		1			-	]
[pma	axis(x,y,z,mag)		1			-	]
[pwa	axis(x,y,z,mag)		1			-	]
fcv	side(rt,lf)	[axis(x,y,z)]	2			1	

[rft	side(rt,lf)	axis(fx,fy,fz,mx,my,mz)	2	-	]
[rcb	axis(fx,fy,fz,mx,my,mz)		1	-	]
[rcf	side(rt,lf)	axis(x,y,z)	2	-	]
[rcc	axis(x,y,z)		1	-	]
[[rsi	side(rt,lf)		1	-	]
[emg	channel#		1	-	]
[mln	MuscleName		1	-	]
[mvl	MuscleName		1	-	]
mor	MuscleName	[axis(x,y,z)]	2	1	
min	MuscleName	[axis(x,y,z)]	2	1	
mla	MuscleName	[axis(x,y,z)]	2	1	
mem	MuscleName		1	1	

Overall Graph command form:

GRAph/#/SAM/DFX

:without DFX	xvar{/OPT} {p1 p2} yvar{/OPT} {p1 p2} yvar {p1...
:with DFX	xvar{/OPT} {p1 p2} yvar{/OPT} {p1 p2} xvar {p1 p2} yvar ...

Graph command options

/#	number of graphs that will appear on screen/page
/SAM	data sets in graph will use the same x axis
/DFX	data sets will consist of independent x,y pairs

Graph variable options {/OPT}

/NGX	Negate the x axis values of this variable's data set
/NGY	Negate the y axis "
/NOA	Don't align sides on this data set even if OPT GRA ALI ON has been set. This is a very special case which is used to handle the nomal joint angle data stored in NORM_ANG.ANZ which has been extracted from the original OSU OUT normal joint angle data set. These angles are stored with the right and left sides already time aligned in the gait cycle while every other data set that will be used here will have to have Telio align it's sides. This is a nasty little kludge but it works (see ANGLES.MAC, macro 4 for an example of this option being used).
/YTI	Applies to EMG's only. Inserts the stored EMG channel name into the graph YTItle - especially useful for making EMG strip charts
/NON	Don't normalize data for this variable even though normalization is set. This is used with normal joint moment and power data from the literature since it is in the ANZ files in normalized form already— it does not need to be normalized again...

/NON Don't normalize this quantity. This is mainly included so data which has already been normalized and stored in an ANZ file can be plotted on the same graph with data which requires normalization. Primary application is for normal moment, power and ground reaction load data sets taken from the literature and imported into ANALYZE. These are usually already normalized and must not be normalized a second.

IMPORTANT time saver:

GRA PRE will display the same set of data defined previously for a graph or set of graphs but reflecting most of the changes made in the graph options. This way you can taylor the graph(s) such as labels and the axis scales and then redisplay the graph without having to type the command line(s) required to display the graph(s). Also, works great for getting the display the way you want it and then changing the settings to print the graph and use the GRA PRE command. Then turn the print option off again.

Graph Display Options

General command form: OPT GRA/# cmnd switch/data
/# : indicates which graph number is being modified (default is 1)

Graph option commands: (values in [] are the default settings where applicable)

RES Reset all the options of the graph to their default values
CYC Clip data to gait cycle [OFF]
RAA Remove average angle (standing position joint angle) [OFF]
PRI Copy graph(s) being displayed into print file [OFF]
SCR Turn off/on the terminal screen (useful for fast printing a graph or when running Telio in batch mode to prevent the log file from becoming huge. [ON]
LGD Display legend in graph [ON]
MLG Display legend using manually input legend text [OFF]
EVE Display gait events [ON]
PCT Display percent cycle line rather marking gait events on the data points. This option only applies when the x axis of a graph is set to percent gait cycle. [ON]
ALI Align sides. Rearrange the data from the side of the second heelstrike such that when the data is graphed, it looks as if the two sets have the same time scale rather than being offset by a half a cycle. [ON]
LIN Display graph data sets with lines between the data points [ON]
SYM Display the graph data points with symbols [ON]
GRL Display the graph label either manual or automatic versions [ON]
MGL Display the manually input graph label. The automatic graph label is the text in the patient name field of the ANZ file. [OFF]
NWT Normalize forces, moments, powers to body weight [OFF]
NWH " " " " " " " *height [OFF]
NWL " " " " " " " *leglength [OFF]
XZL Display the x axis zero line [ON]
YZL Display the y axis zero line [ON]
PFZ Make the fz value of the GRL positive [ON]
SAS Use simple autoscaling for determining the extents of the graph axes. This will make the scales +/- 10% of the min/max of the x and y data. The normal scaling method tries to be fancy and find increments for the axes that display nicely, but it blows it sometimes. Thus, this option provides a way out if it goes wrong. [OFF]
FGE Display only the first data set's gait events. Normally the gait events for every data will be displayed, but for different gait runs with different timings of the events the graph can become quite messy. Thus, this allows the graph to be cleaner by excluding information. [OFF]
MSC ON/OFF Use manually defined scale limits for the graph [OFF]
XMIN XMAX YMIN YMAX Set the scale limits [0 100 0 1]
DEF Default file number from which data will be taken for the graph [1]

LGP Legend position 1-9 1: Bottom left 9: Top Right [1]
DIV Number of divisions for the x and y axes [5,5]
FON Number of font being used for text in graph [1]
MLT Input manual legend text
MGT Input manual graph name text
TIT Input graph title Use *** to make entry contain no text
SUB Input graph subtitle
XTI Input x axis title
YTI Input y axis title

NOTE: A special graph form may be invoked to print out the time-distance parameters in the space normally occupied by a graph. It can be used by stating GRA BLA in which case a blank graph area is produced. When this command has been given it may be followed by these keywords: Indiscriminate use of the following commands, especially TD1 and TD2 on the same graph, will make a mess.

TD1 First set of time distance parameters written to graph area (these are the distance measures)
TD2 Second set of time-distance parameters written to graph area (these are the time parameters)
TIT Title of graph will be inserted in graph area
SUB Subtitle inserted
NAM Name of patient placed in graph area
DAT Date of analysis placed in graph area
FIL Name of data file containing data for graph will be written to graph area

Overall StripCHart command form

sch/# xvar {p1 p2} yvar {p1 p2} yvar {p1 p2} ...

Each of the graphs in the strip chart is defined using the same command syntax that is used with the GRAph command.

IMPORTANT time saver:

SCH PRE will display the same set of data defined previously for a graph or set of graphs but reflecting most of the changes made in the graph/chart options. This way you can taylor the graph(s) such as labels and the axis scales and then redisplay the graph without having to type the command line(s) required to display the graph(s). Also, works great for getting the display the way you want it and then changing the settings to print the graph and use the GRA PRE command. Then turn the print option off again.

Stripchart option commands

NOTE: Each individual graph in the stripchart is actually controlled by the options that apply to the graphs. To customize the appearance of the individual sections of the graphs, the OPT GRA commands must be used. The OPT SCH commands affect the overall appearance of the stripchart.

EVE Display the gait event lines on the stripchart [ON]
TIT Set the title for the stripchart
LAB Show the stripchart label that appears in the lower left corner [ON]
MLB Use the manually set stripchart label rather than the automatic label which is set to the patient name in the ANZ file ON/[OFF] or Label text
MNX Use/set the x axis of the stripchart to a manually defined range of values ON/[OFF] or xmin xmax
SAS Set the autoscaling of the stripchart x axis to be the simple autoscaling described in the graph options
PRI Print the stripchart [OFF]
SCR Turn the terminal display [ON]/OFF
SUB Set the stripchart subtitle

XTI Set the x axis title
XDV Set the number of divisions on the x axis scale [5]
FON Set the font used for the strip chart text [1]

STAtus commands

There are a bunch of status commands which can be used to check what a graph or stripchart's settings currently are and to look at what data will be used in the plots.

Show3D Command Summary

Command	Param1	Param2	Param3
REAd	ANZ filename		
3DSingle	**SIDe** **TOP** **FROnt** **ISometric1** **ISometric2** **MANual**	**MaRKer** **SEGment** **AVeraGe**	Segment Name/**RT**/**LF**/**ALL**
3DMultiple	**SIDe** **TOP** **FROnt** **ISometric1** **ISometric2** **MANual** view direction	**MaRKer** **SEGment** **AVeraGe**	Segment Name/**RT**/**LF**/**ALL**
OPTion	**WIRe**/**NOWire** frame of segment **CENter**/**NOCenter** display of segment **ERAse**/**NOErase** previous frames **STEp**/**ANImation** through data frames **LCS**/**NOLcs** display segment LCS **FACe**/**NOFace** remove hidden faces of segment **HIDe**/**NOHide** remove segment hidden lines **CSY**/**NoCoordinateSystem** show lab GCS **PRInt**/**NOPrint** send screen image to disk file **ManualWiNdow**/**AutomaticManualWindow** use manually defined view window **ManualVieW** define the manual view direction **SeGmentCentered**/**NoSG** **Segment Name** center view on segment LCS center **STArt** frame # **INCrement** # between frame increment		
STAtus			

ANZ Calculation Dependencies

Command		Required Previous Commands
AVG	POS	SEG POS
	ANG	JNT JAN and/or JNT EAN
	REF	AVG POS
CYC	EVE	REA ANZ or TR3 or TRU file
	STA	SEG POS CYC EVE
EMG	NAM	REA GLS1 or GLS2 or FPD file with emg data
	FIL	REA GLS1 or GLS2 or FPD file with emg data
	REC	REA GLS1 or GLS2 or FPD file with emg data
	INT	REA GLS1 or GLS2 or FPD file with emg data
	MWA	REA GLS1 or GLS2 or FPD file with emg data
FRC	CON	REA GLS1 or GLS2 or FPD file with force plate data
	ALI	FRC CON CYC EVE
	CLI	ALI
	CPR	FRC CON FRC ALI if center of pressure under individual feet is desired
	LCP	FRC ALI FRC CPR SEG POS SEG PRM
	WRE	FRC CPR FRC ALI if GRL wrench under individual feet is desired
	INT	FRC CON FRC ALI if GRL integral under individual feet is desired
	REC	JNT LOA

FRC	PMA	FRC CLI SEG BCM FRC CPR
	PWA	FRC CLI SEG BCM FRC WRE
	CPV	FRC LCP SEG VAA
	FIL	REA GLS1 or GLS2 or FPD file with force plate data
JNT	JAN	SEG POS
	JVA	JNT JAN SEG ANG SEG VAA
	EAN	SEG POS
	EVA	JNT EAN SEG VAA
	LOA	SEG POS SEG VAA SEG PRM JNT JAN FRC LCP if GRL data is to be used in calculation
	WRE	JNT LOA
	POW	JNT LOA JNT VAA
	TRN	SEG PRM SEG POS JNT JAN
	SAN	SEG FSC JNT JAN
	MAR	JNT LOA
	WAR	JNT WRE

MRK	INT	REA marker data from ANZ or TR3 or TRU file
	ADJ	MRK INT
	SMO	MRK ADJ
	FIL	REA marker data from ANZ or TR3 or TRU file
	REG	MRK ADJ CYC EVE if gait cycle data is to be used for regularizing
SEG	PRM	MRK ADJ
	MRK	SEG POS
	POS	MRK ADJ AVG REF if SEG POS/REF is given
	VAA	SEG POS MRK SMO/VAA if SEG VAA/MRK is given SEG MRK if SEG VAA/MRK is given
	ENR	SEG PRM SEG VAA
	FSC	SEG POS
	ISC	SEG VAA
	BCM	SEG PRM SEG POS
	MOM	SEG PRM SEG VAA
	ANG	SEG POS

ANZ Site Specific Data

Site/System Specific data required by ANZ

Motion system Coordinate system (lab GCS)

Minor adjustment may be need in the marker coordinates as they are read in from the file. More than probably just a rotation of CS's and/or add/subtract a distance to/from one or more component. ANZ's internal CS assumes a right hand system where x is in the front/back direction y medial/lateral and z vertically upward. The center of the CS is on the surface of the floor between the force plates. NOTE: There is a correction applied to the TR3 file marker data in the present ANZ code because OSU's VICON was setup years ago with the x direction medial/lateral and the y direction front/back plus the center of the CS is 19 inches above the floor !!! (I didn't do it- I only work there :-) Look at the code in read_tr3_file which is contained in file.for...

Force plate plate:

Placement/orientation relative to lab GCS

Change data in anz_init_constants.inc for the variable PltLoc. The plate position/orientation relative to the lab GCS is stored here as a 4x4 matrix for each plate. The initial 3x3 is the rotation matrix of the plate internal CS relative to the lab GCS. The last column is the position of the plate CS relative to the lab GCS.

Amplifier gains and/or calibration matrix

Change data in anz_init_constants.inc for the variable CalMatrix This is a 6x6 calibration matrix for each force plate. The channel gains as well as their excitation voltages are incorporated into the matrix. Thus, the raw force plate voltages are all that are needed as input to these matrices. These matrices are used in the routine that reads GLS2 files (the OSU electrical data file format). There is provision to insert normalized calibration matrices into CalMatrix and the amp gains into NewAmpGains. Basically this section will probably need a rewrite for the specific site. NOTE: FPD files already incorporate calibrated force plate data (I think so at least) and these are already supported by ANZ. This provides a way around having to rewrite the force plate reading section of ANZ.

Physical size

The width and length of the plates must be changed in anz_init_ constants.inc. The relevent variable is PltSize. These are only used in TELIO to allow the representation of the force plates in the 3D views.

Marker system

Right now only 2 marker sets are supported in ANZ: 1) The OSU full body marker set (21 markers) and 2) An upper body only marker set incorporating only the right arm which was part of a preliminary research project at OSU. Changing the marker system used is not a task for the weak of heart but is definitely possible. A separate write up on how to do this will be completed in the future. By the way TELIO is quite marker system independent and should not require any changes at all to work with another marker system. (No changes were made when the upper extremity system was added!)

ANALYZE source file/Routine name Cross Reference

Source File Name	Routine Name
Anz.for	Anz SetOptions FlipLogical GetPaitientInfo
Anz_File.for	Read_anz_file WriteStatus General_itor Force_itor Segment_itor Cycle_itor SwapInteger SwapIntegerArray SwapLogical SwapLogicalArray Read_1dim_array Read_2dim_array Read_3dim_array Read_4dim_array Zero_motion_flags Zero_force_flags Write_anz_file CountNumVarSaved IncVarCnt General_rtoi Force_rtoi Segment_rtoi Cycle_rtoi Write_1dim_array Write_2dim_array Write_3dim_array Write_4dim_array Find_scale_factor_1d Find_scale_factor_2d Find_scale_factor_3d Find_scale_factor_4d
Anz_init_anl.for	InitFullBodyAnalysis InitUpperExtremAnalysis
Anz_jnt_ang_ref.for	CalcJCSAnglesRef FindJntRefAng JCSAnglesRefAng RotMattoJcsAng JCSAngtoRotMat ReadSetRefAng CalcAnatSegPosRef
Average.for	Do_AVG_calc
Char.for	ParseLine MakeUpperCase MakeReal (f)

	RemoveCMNDOpt
Cycle.for	Do_CYC_calc InputGaitEvents CheckGaitPattern (f) CalcGaitStatistics
Digital.for	DigFilterDataSet FilterEMG FilterFRC BuildFilter Realft Four1 Swapr
Emg.for	Do_EMG_calc RectifyEMG IntegrateEMG MoveWinAvgEMG
Export.for	ExportData ExportAVG ExportCYC ExportREC ExportEMG GetSegNum (f) GetJntNum (f) GetMrkNum (f) GetCSType (f) OpenTextFile (f)
ExportFrc.for	ExportFrc ConvGRL ExpFrcGRL PrintGRLHeader ExpFrcCPR ExpFrcLCP ExpFrcWre ExpFrcSI ExpFrcInt1 ExpFrcInt2 ExpFrcPma ExpFrcPwa ExpFrcCpv
ExportJnt.for	ExportJnt ConvertNormJntLoad ConvertNormJntPower ExpJntJCSAng ExpJntJCSVAA ExpJntEulAng ExpJntEulVAA ExpJntLoad PrintJntLoadUnits PrintIgorJntLoad ExpJntLegLoad PrintIgorLegLoad ExpJntWre

	ExpJntPower
	PrintJntPowerUnits
	ExpJntLegPower
	ExpJntFinScrew
	ExpJntInsScrew
	ExpJntScrewAng
	ExpJntTrans
	ExpJntMomArm
	ExpJntWreArm
	ExpJntAngRefCorAng
ExportMrk.for	ExportMrk
ExportSeg.for	ExportSeg
	BuildFileNames
	ExportSwivel3D
	ExpSegPos
	ExpSegVel
	ExpSegAcc
	ExpSegAng
	ExpSegEnergy
	ExpSegFinScrew
	ExpSegInsScrew
	ExpSegMom
	ExpBodyCM
File.for	Read_file
	Get_file_name (f)
	Read_c3d_file
	AssignFullAnlMrkName
	Read_tr3_file
	Read_tru_file
	Read_fpd_file
	Read_gls_file
	Read_gls2_file
	ReadGls2Bytes
	Save_file
	Save_gls2_file
	CalcScaleFactor
	WriteGls2Bytes
	FillGls2Header
	Save_tru_file
Filter.for	Filter
	FiniteDif
	DPF *
	SWITCH *
	MACRO *
	SIGMA *
	TDYN *
	INVSP *
	SPOFA *
	SDOT (f) *
	SPOSL *
	SAXPY *
	TRACE *
	GCVSPL **
	BASIS **

PREP	**
SPLC (f)	**
BANDET	**
BANSOL	**
SEARCH	**

* Authors: Hank Busby & David Trujillo
** Author: Herman Woltring

Force.for	Do_FRC_calc
	ConditionFrcData
	AlignFrcMtnData
	SetTrigFrm
	SetFtPlt
	ClipFrcMtnData
	Swap
	CalcCntrPres
	CalcCntrPresLCS
	CalcFrcWrench
	CalcFrcIntegral
	IntegrateData
	ReconstructGRL
	CalcPitchMomentArm
	CalcPitchWrenchArm
	CalcCprVel
Joint.for	Do_JNT_calc
	CalcJCSAngles
	CalcJCSAnglesSimple
	HandleAngleError
	InterpAngles
	CalcJCSVelAcc
	CalcEulVelAccYXZ
	AngFilterStatus
	CalcEULAngles
	CalcEULVelAcc
	CalcJntLoad
	CalcAllJntLoads
	CalcAJntLoad
	LoadLCStoLCS
	LoadGCStoLCS
	LoadLCStoGCS
	LoadGCStoJCS
	CalcJntWrench
	CalcJntPower
	DigFilterJnt
	CalcJntTranslation
	CalcJntScrewAngle
	CalcJntMomentArm
	CalcJntWrenchArm
Marker.for	Do_MRK_calc
	InterpolateMrkData
	Bad_marker (f)
	Find_interp_frm
	AdjustMrkData
	Lagrange_Interp
	SmoothMrkData

	DigFilterMrkData
	RegularizeMrkData
Math_sub.for	Gauss_Elim
	MatInv (f)
	VecChangeCS
	PntChangeCS
	DistPtPlane
	DistLineLine
	DistPtLine
	VecDot
	VecMag
	VecSub
	VecX
	VecNorm
	VecLth
	PntGCStoLCS
	VecGCStoLCS
	VecLCS1toLCS2
	MatTran
	MatMult
	MinLineXPnts
	EulerAngles
	EulerAngles2
	ShldrAngles
	JCSAngles
	SwivelAngles
	TriCen
	Line_Xsect
	PerpenPtLine
Segment.for	Do_SEG_Calc
	CalcSegParams
	CalcFullBodySegShape
	CalcFullBodyMassInertia
	CalcFullBodySegEnds
	CalcFullBodySegPos
	CalcLCSFoot
	CalcLCS1
	CalcLCSCalf
	CalcLCSPelvis
	CalcLCSTrunk
	CalcLCSHead
	HandleSegPosError
	InterpSegPos
	CalcSegFiniteScrews
	CalcFSAWaldron
	CalcFSAWoltring
	CalcSegInstantScrews
	CalcInstantScrewAxis
	CalcSegEnergy
	CalcMrkDiagnostics
	CalcBodyCM
	CalcSegMomentum
	CalcSegGCSAngles
	DigFilterSeg
SegVelAcc.for	CalcSegVelAcc

	RotMatDeriv
	MakeOrthoNorm
	OldEulAngDeriv
	YXZEulAngDeriv
	CalcSegRotVelorAcc
	SegFilterStatus1
	SegFilterStatus2
	MrkVAAtoSegVAA
	FormSolveLSEqtn
Status.for	DisplayStatus
	DisplaySegment
	DisplayCycle
	DisplayAverage
Upper.for	CalcUpperBodySegPos
	CalcLCSPelvis2
	CalcLCSTrunk2
	CalcForeArmLCS
	CalcLCSHead2
	CalcUpperBodySegEnds
	CalcUpperBodySegShape

VAX/MAC ANZ Source Code Conversion

Conversion of ANALYZE source code between VAX and Macintosh

Code which must be modified for conversion between VAX and Macintosh versions of ANZ. All lines are marked in the code as follows:

Vax specific code line ends in ! VAX

Macintosh specific code line ends in ! MAC

Just comment out the lines of code not applicable. In otherwards,comment out the VAX lines when working on the macintosh and the MAC lines when working on the VAX.

Location of machine specific code

File name	Subroutine Name
anz_file.for	read_anz_file
anz_file.for	write_anz_file
export.for	OpenTextFile
file.for	read_file
file.for	save_file

NOTE the code incompatibilities have to do exclusively with the handling of input/output files.

Updating ANZ Source Code

Steps required to add a variable to ANZ

NOTE: Before modifying any of ANZ's source code **make a back up !!!!**

1) Add variable to desired common block

   Ex: BCMVel added to common /Segment/ in anz_seg.inc

2) Add calculation flag structure

   Ex: SEG_FLG.BCMVel added to structure /segment_flag/ in anz_structures.inc

3) Add indication of calculation status to status routine in status.for

4) Update structure /All_Flags/ in anz_file.inc Primarily this is the comment about the # of bytes in the flag structure to which the variable has been added.

5) Add pointer location for new variable to structure /anz_layout/ which is contained in anz_file.inc Make sure that it is added at the end of the list of pointernames

   Ex: SEG_BCMVel is added after FLAGS

6) Update MaxVar in write_anz_file stored in anz_file.for MaxVar is the maximum number of quantities that are possible to be stored in the ANZ file.

   NOTE: Right now there is only room in the structure /anz_layout/ for 128 separate variables and ~108 are being used now. The restriction comes from using 512 byte records for reading and writing which can, as a result, only contain 128 integer*4 numbers and a single record at present is being used to store the pointer structure. If more than 128 variables are to be stored than some additional code must be added to ANZ to handle more than a single record holding the variable location/scaling pointers...

   Ex: Increase MaxVar from 107 to 108

7) Update the routine write_anz_file by adding the code to save the new variable

   Ex: Added code to write BCMVel immediately following code to write BodyCM. The code was essentially a copy of that to write out BodyCM just modified to save the new variable.

8) Noting position of new variable writing code add a corresponding increment to CountNumVarSaved in anz_file.for This allows a display count of the progress of the file saving process.

   Ex: Added the following line right after the line of BodyCM

   if (SEG_FLG.BCMVel) Call IncVarCnt(108,VarNum(NumVar+1),NumVar)

   NOTE: That 108 is the location of the new variable pointer in structure /anz_layout/

9) Increase the size of AnzVarName to size of MaxVar. AnzVarName is contained in common /Names/ which is in anz_names.inc

Ex: Increase from 107 to 108

10) Add AnzVarName initialization text to anz_init_constants.inc at the end of the last data statement in the file. This is used to display the name of the variable being written/read.

Ex: Add 'Body CM Velocity ' after 'Calculation Flags '

NOTE that the variable name must be 20 characters long.

11) Add code to read_anz_file to read new variable from ANZ file

Ex: Add code to read BCMVel immediately following the code to read BodyCM

12) Update code at front of read_anz_file which is used to convert between least significant byte to most significant byte computer storage methods for the calculation FLAGS. This is rather messy since it is dependent upon how the calculation flags were ordered in the structure /All_Flags/ and the fact that only the logical*4 and integer*4 quantities in the structures are to have their bytes reversed if the ANZ file is moved between computers.

Ex: Since BCMVel was added to SEG_FLG which is in the middle of the FLAGS structure the position of last logical flag of EMG_FLG was increased by 1. The numbers within the code at 'if (ptr.FLAGS .ne. 0) then' were changed. Thus,

```
call SwapLogicalArray( 55,FLAGS.array )
do i= 66, 70 ! EMG flags
i= FLAGS.array(71) ! EMG.FLG.NumProcOp
call SwapInteger( i,FLAGS.array(71) )
```

changed to

```
call SwapLogicalArray( 56,FLAGS.array )
do i= 67, 71 ! EMG flags
i= FLAGS.array(72) ! EMG.FLG.NumProcOp
call SwapInteger( i,FLAGS.array(72) )
```

This is very messy and must be done with utmost caution. Moving things around too much will make previous ANZ file's calculation FLAGS be read in incorrectly. The data will still be read from ANZ file but some of the nice save features such as the operations on the EMG data and indications as to smoothing/filtering of various data will be hopelessly scrambled.

12) Update the zero_motion_flags or zero_force_flags, which ever is appropriate, in file.for.

Ex: Add SEG_FLG.BCMVel= .false. to zero_motion_flags

13) Add export variable code to appropriate export routine...

Ex: Add ExportBCMVel to ExportSeg.for

Updating Telio Source Code

Updating TELIO to display new variables (graphing functions only):

1) Add pointer location for new variable to structure /anz_layout/ which is contained in tel_anz_file.inc Make sure that it is added at the end of the list of pointer names

   Ex: BCMVel is added after FLAGS

2) Add the variable name text to and increase the size of the array AnzVarName which is defined in tel_names.inc and is initialized in tel_init.inc.

   Ex: Add the text 'Body Cntr Mass Vel ' to tel_init.inc
   Increase AnzVarName(107) in tel_names.inc to 108

3) Add variable name/number to tel_const.inc If you do not add this at the end of the list of numbers than all the numbers must be shifted after the one you insert.

   Ex: Add parameter BCV_DATA= 21
   Shift all variable numbers up by 1 starting BMM_DATA on

If numbers are shifted, then the routines which identify a command to a specific class of data must be updated. The classes are:

Class	Subroutine
Marker	MrkCmnd
Segment	SegCmnd
Joint	JntCmnd
Force	FrcCmnd
Reconstructed	RecCmnd
EMG	
Average	
Muscle	MslCmnd
Time	TimeCmnd

These routines are in the file buildgraph.for

4) Add the 3 letter acronym for the new variable to the listing of acronyms in the array VarNames. The order of the variable names MUST be the same as the variable name/numbers in tel_const.inc. The acronyms for VarNames are initialized by a data statement in the file tel_init.inc Also, the dimension of the VarNames array must increased in tel_names.inc

   Ex: Add 'BCV' immediately following 'BCM' in tel_init.inc
   Increase the dimension of VarNames from 69 to 70

5) Modify the flags structure of the variable class so that there is an indicator of whether it has been read. These structures are defined in the file tel_struct.inc

   Ex: Add .BCMVel to the SegmentFlag structure

NOTE: The ANZ file contains a copy of the flags and thus the structures here must be identical to those used in ANZ if these flags are to be read properly.

6) Modify the code in the subroutine ZeroFlags, contained in the file tel_file.for, so that the new flag position is zeroed properly.

Ex: Increased the do loop zeroing the FD.SF.Flag array from 11 to 12

7) Modifiy the code in ReadANZFileParam, contained in the file tel_anz_file.for, so that the flags are read properly and if there are any initialization variables that must be read. If only a change in the flags as occured, then no change to this code should be necessary because the changes in the flag structures in step 4 should takecare of things. The reading of the flag structure from the ANZ file accomplishes this task...

Ex: No modification of ReadANZFileParam was needed to add BCMVel

8) There is an error trap (format statement # 100) in AssignPlotData that displays all the variable names if a mistake is made in the command line when a variable name should be given. This write statement has the total number of variable names incorporated into it. Thus, when the dimension of the array VarNames is increased, the number of variables printed out by this statement must be increased to this same value. AssignPlotData is in buildgraph.for

Ex: Increase k=1, 69 to k=1, 70 for this statement

9) Update the arrays GrNumPrm and TdNumPrm defined in tel_graph.inc and tel_3dvu.inc respectively and initialized in tel_init.inc These arrays contain the number of command line parameters that must appear after the variable name for the data to be correctly defined for graphing or 3d display. The dimension of the arrays must be increased to be the same as the dimension of VarNames. The order of data elements in the initialization data statement must be the same as the VarNames 3 letter acronyms.

Ex: Add ,1 to the GrNumPrm initialization immediately following the number of paramters for the BCM variable and add ,0 to TdNumPrm in the same place. Increase the dimension of these arrays from 69 to 70. NOTE: The GrNumPrm value is 1 because the component (x,y,z) of the Body CM Velocity must be given immediately following the BCV variable name on the command line. In the case of TdNumPrm, no furhter information is needed since the velocity is drawn as a vector originating from the BCM point.

10) Add code in the respective Assign data routine for the class of variable added to make it possible to read data in from ANZ file. The routines corresponding to the data classes are:

Class	Subroutine
Marker	AssignMrkData
Segment	AssignSegData
Joint	AssignJntData
Force	AssignFrcData
Reconstructed	AssignRecData
EMG	AssignEmgData
Average	
Muscle	AssignMslData
Time	AssignTimeData

These routines are in the file buildgraph.for The number and type of parameters passed into the routine ReadFileData are critical to assuring proper reading of the

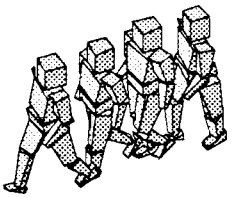
data, so be careful here. Otherwise the program will more than probably read something in from the file but as soon it gets graphed you'll see that it is gibberish. (ReadFileData routine is in tel_file.for) Be careful about the number of command line parameters needed by variable. These are defined in GrNumPrm

Ex: Add code to AssignSegData to read BCV immediately following the code for reading BCM. Use BCM code as a prototype for BCV. In fact in this case the two sections are identical.

11) Add the code which reads the data from the ANZ file. This is added to the routine ReadANZFileData which is in tel_anz_file.for This code is very dependent upon the exact shape of the array in which the data is stored in the ANZ file. Look at the explaination with the routine for more details. Typically it is best to use a variable whose array shape is similar to the new variable as a prototype for the new code.

Ex: Add code to ReadANZFileData to read BCV immediately following the code for reading BCM. Use BCM code as a prototype for BCV.

12) Add variable to status display so that user can tell whether the new variable is present in the ANZ file.

ANALYZE 

An Overview

What is ANALYZE ?

ANALYZE is an analysis package for processing motion, force plate and EMG data from a motion measurement system which can generate a number of quantities that are used for diagnosis and research of human locomotion and locomotion disorders as well as other activities such as chair raising or upper extremity motion. In fact, at present *ANALYZE* will generate over 100 different quantities. The input data required to use *ANALYZE* are 3 dimensional marker trajectories, force plate ground reaction load data and/or muscle electromyography data. The previous senence uses 'and/or' because *ANALYZE* is capable of performing either motion, ground reaction, or EMG analyses alone or in combination with one another. This enables marker data to be crunched to get purely kinematic information, such as joint angles, velocities, and accelerations without having to know force plate data (obviously that data isn't needed for such an analysis). It also enables ground reaction force data to be crunched to get such purely load related information as center of pressure and ground impulse. But, since all this information is required to find joint moments and powers (in reality this is only true for the double support phase of gait- more on that later), *ANALYZE* has been set up in a flexible fashion such that the tools which need to be available to get these quantities are also available for use in studies which have little or nothing to do with gait. For example, the force plate section of *ANALYZE* can be used to produce center of pressure information on a person's balance as they stand for a period of time on the force plate(s). The EMG section can be used to perform signal processing tasks such as digital filtering, integration, and rectification. The results can be displayed in a multitude of ways in *TELIO*, the graphics postprocessing program that is a companion to *ANALYZE*. *TELIO* is discussed in detail in its own documentation. In addition to full body motion, *ANALYZE* has been configured for studying upper extremity motion in which only one arm and the trunk and head are present. Another example is that rigid body arrays can be tested using the motion section of *ANALYZE*. In the end, the result is that one package can be used to study

a large number of motion, EMG, and ground load protocols without having to create special programs for every little variation that one comes up with, as has been done in the past.

In addition, a consistent and extendible file format has been developed to store the analysis results in a machine independent format. In this way, the results will be usable on any future machines which a lab may purchase or decide to use for other reasons. This also facilitates the sharing of data between researchers. The format is extendible in that if future researchers come up with other analyses to add to *ANALYZE* (which is relatively easy to do and will most definitely happen) the resulting files will still be compatible with previously written program code used to read *ANALYZE* files. Up till now every time another analysis method was developed another file format was created and more programs needed to be written to use the files. The result has been a proliferation of files, each of a unique, idiosyncratic type, which a programmer needed to wrestle with for a while. Well, no more!

Future additions to *ANALYZE* have been planned for by the very way in which it has been designed internally. Basically, the analysis code is based upon the principles of the mechanics of three dimensional rigid bodies. These analyses do not contain any simplifications which prevent their use in future studies (actually there is one dealing with the inertia matrix of a body segment. More on that in a later section.), as is the case with software which is oriented toward sagittal plane motions only and hence contains two dimensional motion assumptions. To do this, *ANALYZE* has been structured to perform its analysis upon only 3D rigid body segments (e.g., the foot, calf, and thigh) which are connected by joints which do not have any constrained degrees of freedom. In otherwards, the segments are free to move relative to one another (3 translations and 3 rotations- the definition of these is discussed later in the section about coordinate systems). In fact, whether the whole body, only the upper body, or only the lower body is present becomes immaterial to the analysis routines as long as the segment connections (i.e., joints) have been defined properly.

For this type of flexibility to take place, the first thing that *ANALYZE* does is use the marker data to calculate the position and orientation of the body segments in space in each frame of data. In addition, the joint position relative to these segments must be defined (in the segment's local coordinate system) and the segment mass and inertia calculated from anthropometric data. (For mathematical purists out there, yes the joint positions are assumed based upon the anthropometric data, BUT the kinematic screws of the relative motion between segments can be calculated in the subsequent analyses and could be used to find the "joint axis". This may be a future enhancement, but don' t bug me about it.) This is the only part of *ANALYZE* which is marker set specific and thus these routines are the only ones that need to be modified if a different marker set is used. In fact, *ANALYZE* has been set up such

that it can support a number of marker sets by using option switches in its commands. (See the later section about how to modify *ANALYZE* for your own marker set.) A better solution would be to have the marker set to segment calculations in a parameter file which would be read and interpreted by *ANALYZE*. The calculations would essentially be mini programs in some compact language. This is very desirable, but is path fraught with pitfalls, traps, and headaches (e.g., what should *ANALYZE* do if the person who wrote the marker to segment calculation did it so wrong that divide by 0's occur ?) and deadlines needed to be met. Thus, it is not part of *ANALYZE* at present but it is definitely a future enhancement (once again, don' t bug me about when it will be done- I' m not a computer science major so I need to learn how write a language interpreter, etc...). Once the marker to segment calculations have been performed the only other thing that *ANALYZE* needs to know is how many segments there are, which ones they are, and how they are connected together.

What does ANALYZE recognize ?

At present, *ANALYZE* is set up to recognize up to 15 unique body segments and 15 unique joints. It is 'up to' because, as pointed out previously, the program has been set up in flexible manner so that not all segments and joints need be included in the analyses. However, *care must be taken at this point.* If you choose to analyze less than the full body, then the analysis must be set up correctly. For example, the joint angles of the shoulder can't be calculated from the position and orientation of the thigh and calf. *ANALYZE* is not set up to be used by just anyone. The user must understand the principles of mechanics to use and/or modify the program properly.

The body segments which *ANALYZE* recognizes and their abbreviations are:

Right Foot	rtft	Left Foot	lfft
Right Calf	rtcf	Left Calf	lfcf
Right Thigh	rtth	Left Thigh	lfth
Pelvis	pelv		
Trunk	trnk		
Right Hand	rthd	Left Hand	lfhd
Right Forearm	rtla	Left Forearm	lfla
Right Upperarm	rtua	Left Upperarm	lfua
Head	head		

The joints which *ANALYZE* recognizes are:

Right Ankle	rtak	Left Ankle	lfak
Right Knee	rtkn	Left Knee	lfkn
Right Hip	rthp	Left Hip	lfhp
Pelvis-LAB GCS	pvlb		
Pelvis-Trunk	pvtk		
Right Wrist	rtwr	Left Wrist	lfwr
Right Elbow	rtel	Left Elbow	lfel
Right Shoulder	rtsh	Left Shoulder	lfsh
Neck	neck		

Note that the pvlb joint is only used to compute the absolute pelvis angle and thus can be thought of as an imaginary joint since there is no such anatomic joint. This joint is the only one in which the joint angles are calculated relative the global coordinate system (GCS). In all other joints, the angles are defined as those between the segment distal to the joint relative to the proximal segment.

Other joints could be added later on, but the arrays within the program have been set up with these limits (15 segments, 15 joints) in mind. The file format is set up in such a way that other segments and joints could be added and the files could still be read by subsequent programs, such as *TELIO*, without having to rewrite the file reading code. However, the program which reads such a file must have provision for larger arrays and a way to use this extra data. *TELIO* is set up to handle the 15 segments and 15 joints defined here at a maximum. (As an aside, *TELIO* can read in up to 6 different ANZ result files and plot any combination of their contents in graphs or display the contents in 3D. So, even though there is a segment and joint limit, it can still do alot.)

What can ANALYZE calculate ?

The quantities that *ANALYZE* calculates can be grouped into several categories: marker based information, gait cycle time-distance information, segment based information, joint based information,force plate based information, and EMG information. Each of these is describedin the following sections.

Marker based information

Marker position: Marker position data read in from a file can be checked for gaps and interpolated/extrapolated using a selectable order polynomial. Marker data can then be smoothed if desired with any of three methods. A generalized cross validated smoothing polynomial (GCV) (Woltring, 1986) which automatically adapts to the

data, a dynamic programming filter (DPF) (Dohrman, 1988) which is also adaptive, and digital filtering with selectable high, low and notch filter frequencies. NOTE: *ANALYZE* has the capability to interpolate the marker data onto any specified interval with any number samples created on that interval. This is convienient for making gait data appear at regular intervals of the gait cycle, for instance every 2% of the cycle.

Marker velocity and acceleration: The first and second derivatives of the marker trajectories can be calculated using either the GCV, DPF, or finite difference methods.

Marker diagnostics: Used to test the accuracy, resolution, and repeatability of the motion measurement system. This is essential in determining limits of confidence in the analysis results. Two measures are provided: 1) The intermarker distances between markers on the same segment in each frame of motion data and 2) the position of the segment markers relative to the segment local coordinate system. These show how the markers are moving relative to one another within a segment. Ideally there should be no movement if the markers were rigidly attached to a rigid body but in reality there is motion due to such factors as skin and muscle motion and vibration of the sticks on which some markers are placed. This information is useful in determining the reliability of the calculation of the segment position and orientation.

Segment based information

Segment Parameters: Estimates of segment mass, inertia relative to the principal axes of inertia, and joint positions relative to segment center of mass based upon published experimentally derived regression equations. Segment local coordinate system is assumed to be aligned with the principal axes of inertia (the assumption alluded to previously- this may be eliminated by a more rigorous technique (Kaleps, 1984) which requires the measurement of the 3D locations of some 83 points!) and thus the products of inertia are 0. Thus, inertia is expressed as 3 vector rather than a 3x3 matrix. Separate databases are used for male, female, and children (Jensen, 1989).

Segment Position: Segment position and orientation relative to the lab global coordinate system (GCS). Expressed in terms of the position of the segment mass center and the orientation of the segment principal moment of inertia axes via a 3x3 rotation matrix. Note that since a rotation matrix is used there is no built in assumption as to a type euler angle, or whatever, system that is used to express the rotations. This is most useful for subsequent mathematical analyses, but causes difficulty in interpretation.

Average segment position: Average position and orientation of the segment in all motion data. Useful for looking at average standing posture as well as studying and correcting for biases due to marker placement on the segments.

Segment angle: The yaw-pitch-roll euler angles of each segment relative to the GCS. In this way, the angle of the foot relative to ground may be studied, for instance. This provides a method to interpret the 3x3 rotation matrix, though not the best since euler angles are rather obscure at times also.

Segment velocity and acceleration: Segment velocities and accelerations, both translational and rotational, relative to the lab GCS and the segment local coordinate system (LCS). This information is useful in interpreting kinematic data since it is dependent upon the first and second derivatives of motion and thus shows changes in motion much more clearly. These derivatives are computed using either the GCV, DPF, or finite difference methods. Note that fully three dimensional kinematic calculations are used in computing these quantities.

Segment energy: The rotational, translational, and total kinetic energies of each segment are available for each frame of motion data. Potential energy of the segment center of mass is defined relative to the GCS. Total energy of segment which is a scalar summation of the total kinetic and potential energies of the segment.

Body energy: Same as for individual segments except that they are the sum of all the kinetic, potential, and total energies of all the segments represented in the marker set.

Segment momentum: The linear and angular momentums of each segment in the segment's LCS. In addition to these vectorial quantities, the magnitudes of both momentums are also calculated.

Segment global screw: The screw axis defining the motion of the segment in the GCS from one frame to the next. This is an instantaneous screw since the position of the segment in the previous frame is used as the reference for the calculation of the screw. Note that the screw is expressed in the GCS rather than the LCS of the segment. A separate calculation of the instantaneous screw of each joint (the motion of the distal segment relative to the proximal segment) is also carried out. In addition, the finite screw axis for the same motions is also calculated for comparison/research purposes. Both the relative screw for the segments's motion form one frame to the next as well as the absolute screw for the kinematic state of the segment relative to the GCS can be calculated. Several calculation methods are present for calculating the finite screw

since studies have shown that some methods are more sensitive to errors due to noisy data than are other methods [Fenton, 1989].

Joint based information

Joint angle: Joint angles are calculated relative to the joint coordinate system (JCS) and the proximal segment LCS. The JCS is a nonorthogonal coordinate system which more closely resembles the way in which orthopaedists describe joint angles than do other euler angle sequences. The JCS for each joint is slightly different and as a result they should be presented with diagrams and mathematical formulae to give each joint a precise definition. The concept of this coordinate system is based upon a paper by Suntay and Grood (1986). Their coordinate system was defined only for the knee, but here it has been generalized to the 15 joints of the body which *ANALYZE* recognizes. In addtion, standard euler yaw-pitch-roll angles have been included for completeness sake since some previous studies have used this convention. Note that the sagittal plane angles (flexion-extension) are very similar for both coordinate systems and in fact have been analyzed theoretically to show that they do not deviate greatly until a fair amount of non-sagittal angulation occurs. The non-sagittal angles (in/external, ab/adduction, pro/supination, etc.) are a great deal different from one another, however.

Average joint angles: Average joint angle over the period of the motion data. This is useful for looking at biases in the calculated joint angles resulting from the placement of the markers upon the body segments.

Joint velocity and acceleration: The joint velocities and accelerations in both the JCS and proximal segment LCS's are calculated in either of two ways. The kinematically correct way utilizing the absolute rotational velocities and accelerations relative to the lab GCS of the segments on either side of a joint or by numerically differentiating the individual joint angle components using the GCV, DPF, or finite difference methods. Note that this second technique, though used by many previous researchers, is not correct for calculating the true joint velocities and accelerations. The error is not particularly significant unless the joint motion has significant components in planes other than the sagittal which the case with pelvic motion especially and in hip motion to a lesser degree.

Joint reaction load: The joint reaction loads (forces and moments) are computed at the current joint position and segment kinematic state. Note that the joint reaction loads (i.e., 3 moments and 3 forces) are calculated since all calculations are carried out in full 3D with no assumptions as to the type of motion being studied. A number of

previous software packages have limitations in this respect for either they assume sagittal plane motion or they only calculate joint moments. The loads are given relative to the JCS, the proximal segment LCS, the distal segment LCS and the lab GCS. All four are calculated to enable comparison with other previous studies since there has not been uniformity in the way in which the joint loads have been expressed in the literature published to date. The loads are always given in terms of the load exerted by the proximal segment upon the distal segment across the joint. The scheme used to calculate the joint loads is very flexible for it is based upon the concept of chained calculations. Depending upon the activity, the phase in the activity, and the degree to which the ground reaction loads are known, an appropriate joint load calculation method is used. For example, if the reaction loads under the stance leg during single stance are known then a sequence of calculations which moves from the support foot up to the hip and then starts at the swing foot and moves up to the hip of the swing leg also is used to calculate the leg joint reaction loads. The arms, neck and trunk joint reaction loads are calculated by progressing from distal to proximal across the arms and then the trunk. If, however, the ground reaction loads under the support foot are not known then a calculation sequence which progresses up the swing leg, along both arms, down the neck, then to the trunk, and then lastly down the support leg is utilized. Note that the resulting calculated joint reaction loads in the support leg will, without a doubt, be different from those calculated using the ground reaction if it were known. This is primarily due to inadequate determination of the mass/inertia properties and joint locations of the segments further up the calculation chain. Any inaccuracies will be cumulative due to chaining of the calculations, however at least an estimate is available whereas normally none would be calculated in other software packages. The calculation scheme will change throughout the data set depending upon what the round contact/valid force plate data combination is at a given data frame. A number of schemes have been developed for full body joint reaction load calculation. To do only upper body for example, a connectivity structure, which guides the calculation sequence, would have to be defined within *ANALYZE*. Lastly, if at any frame of motion data the reaction loads of a joint cannot be calculated (the joints of both legs during double stance of human gait when neither foot's ground reaction load is known), then an array indicating the validity of a joint's reaction load calculation is updated and later in *TELIO* this data is automatically skipped over when it is graphed or displayed in 3D since it is invalid.

Support load: The vectorial summation of the joint reaction loads of the joints (ankle,knee,hip) in the individual legs. This is of interest in studying the process of gait (See Winter, 1987).

Joint reaction wrench: The joint reaction load may also be expressed in terms of a load wrench, which like its kinematic analog, the instantaneous screw, is an invariant quantity in terms of the coordinate system in which it is expressed. This may prove useful as a way to compare the joint reaction loads calculated by different programs and/or labs. It may also be useful in interpreting the joint reaction loads in a more succinct and consistent manner. This is needed at times because the resulting joint reaction load patterns in 3D can be quite complex. Previous studies have looked primarily at the motion moments (saggital plane joint moments consisting of both active and passive moments) and have ignored the structural moments (such as the ab/adduction resultant moments and an/posterior force in the knee). These may prove to be important in areas such as compensation of gait where a patient may be trying to minimize the structural moment to protect a specific structure (e.g., minimizing abduction moments at the knee to minimize strain on the medial collateral ligament). Anyway, the capability is here for the using.

Joint screw: The screw defining the motion of the distal segment relative to the proximal segment across the joint from one motion frame to the next. This is an instantaneous screw using the positions and velocities of the distal and proximal segments. Note that the distal segment position/velocity is computed in the proximal segment LCS in each frame in the process of the screw calculation, so even though the proximal segment may be moving the screw is still calculated properly. This screw defines the relative motion of the segments across the joint. The literature indicates that the calculation of screws is very sensitive to errors in position data and thus unless the segment position and orientation is determined using redundant markers (more than the minimum of 3 per segment), the resulting screws may not be very useful. I have not yet had a chance to check this out. This calculation could also be applied to data gathered from an instrumented spatial linkage (ISL) (another future project...) which would presumably give better results. Lastly, finite screw calculations have been included for comparison/research purposes.

Joint Screw Angle: Essentially the direction vector of the finite screw axis of the joint multiplied by the rotation angle about the axis and projected onto the proximal segment, distal segment, or JCS coordinate systems.

Joint Translation: The translation across the joint based upon the assumption that joint position relative to the proximal and distal coordinate systems is fixed. If not then the point indicated by the two segment LCS's is not the same and hence the difference between the two points is the calculated joint translation. At present, this is used as a

diagnostic measure to check the assumption that the ankle, knee, hip, elbow, and shoulder markers are being placed near the axis of rotation of the joint.

Joint power transfer: The flow of power across the joints is calculated using the joint rotational velocities and the joint reaction moments. This transfer consists of both active (contracting muscles) and passive (resistance of the ligaments, capsule, passive elasticity of the muscles) power lumped into a single quantity.

Joint Reaction Load Moment Arms: The moment arms of each joint reaction loadmoment component relative to the joint location is calculated using the joint reaction forces in the other planes. For instance, the moment arm of the x moment component is calculated using the y and z components of the resultant joint force.

Joint Wrench Arm: The perpendicular distance from the joint reaction load wrench axis to the joint location is calculated.

Force Plate information

Individual and combined foot ground reaction load: The force place data is summed and split such that the loads under each individual foot and their combination is computed. The only time these are different is during the double support phase of gait.

Individual and combined foot centers of pressure: The center of pressure (COP) of the ground reaction loads measured under each foot as well as the center of pressure from the combined feet. The only time these two are different is during the double support phase of gait.

Center of mass via integration: The location of the center of mass in 3D of the entire body is calculated by integrating the ground reaction force twice and scaling it by the body mass. This may be compared directly with the center of mass calculated using the estimated segment masses and center of mass locations.

Ground reaction wrench: Wrench of the ground reaction load for the individual feet and the combined foot loads.

Center of pressure relative to foot LCS: Position of the ground reaction COP during the contact phase of each foot relative to the foot' s LCS.

Strike index: The position of the COP relative to the total anterior/posterior distance along the foot and the position of the foot's LCS. This requires that the foot length

be calculated. At present *ANALYZE* uses the toe and heel markers of the whole body marker set to do this calculation.

Individual and combined foot ground reaction impulse: The impulse of the foot upon the ground.

Reconstructed ground reactions: Using the segment accelerations combined with their mass and inertia as well as geometric configuration, the ground reaction loads, both magnitude and location, are calculated.

Pitching moment arm: The perpendicular distance from the line formed by the ground reaction force passing through the center of pressure to the location of the body center of mass. This can be thought of as a rough estimate of what makes the body pitch forward/backward, side to side, and twist.

Wrench pitching arm: The perpendicular distance from the line formed by the ground reaction load wrench axis to the location of the body center of mass.

EMG's

Filtering: Digital filtering with high, low, and notch filter capabilites.

Rectification: Rectify the signal

Integration: Integrate the signal

Moving Window Average: Perform a moving window average with a user specified window size.

Threhold: When combined with several of the above signal processing operations this can provide a way of displaying when a muscle is or is not active in an on/off fashion.

Comment on numerical capabilities

At present *ANALYZE* contains three different ways to filter data. One method is a generalized cross validation routine utilizing splines developed by Herman Woltring which is configurable from within *ANALYZE*. Another generalized cross validation routine from Henry Busby of The Ohio State University which does not use splines is also included. Lastly, a general digital filter routine can be used in which its upper and lower cutoff frequencies can be set as well as have a notch defined. This routine can have the slope of the

cutoffs and notch specified so that the user may create their own specific filter functions. It also contains a Hanning window function which may be used or turned off. Any of these filter operations may be chained if so desired. Several methods have been provided because experiments have shown merit and inadequacies in all of them. It was felt that the user should have final say on what is used by being able to experiment with their data since it may have unique properties which make it work better with one method or another. Most quantities within *ANALYZE* are filterable.

How is ANALYZE presently implemented ?

In its present implementation, *ANALYZE* is not being used optimally, but at the same time practical, useful results are being generated. Currently, *ANALYZE* runs on a whole body marker set consisting of 21 markers, an upper body marker set consisting of 9 markers which includes the head, trunk, pelvis, right upper arm, and right lower arm, and a simple 6 marker set for measuring foot motion only. The full body marker set does not allow the determination of true 3D position and orientation of some of the segments because in some cases markers are shared by two segments. In these cases, the markers are assumed to sit at the joint "center", or on the axis of rotation, and thus should be sharable between segments since the common center would not move relative to the two segment LCS's. A primary reason for this marker set being used is that it has proved to be a very practical arrangement for day to day, and often several times a day, use upon patients with various gait pathologies. Since the marker data is being collected upon a VICON system, minimization of technician time in sorting and marker identification errors is very important. With the introduction of AMASS, the new 3D marker trajectory reconstruction software of Adtech and distributed by Oxford Metrics, this point is becoming less important. The marker set is an effective compromise between measurement of motion which is clinically useful yet still can be used upon all types of pathological motions without the markers being confused. Machine independency of code and data files has been extensively tested by producing a version of *ANALYZE* on the Macintosh using the same FORTRAN source code used for the VAX version. In addition, data files produced by the two programs have been interchanged successfully.

Marker data can presently be read in from VICON TR3 files, AMASS C3D files, and an older file format used at Ohio State called TRU files. Force plate data is read in from C3D and FPD files as well as from from two file formats used at Ohio State- GLS and GLS2 files. The results can be saved in either *ANALYZE* format (.ANZ) or the older TRU format. Results can also be dumped in text form to allow export to other packages such as statistics and presentation graphics. The number of file formats supported will be expanded in the future depending upon how widely *ANALYZE* is used.

The Future

At present, the public release of ANALYZE is projected for the fall of 1991. Full documentation including derivation of the algorithms used in the FORTRAN code as well as a user's guide will be available. The source code will also be made available. The only charge will be to cover the costs of reproduction. The intention in releasing ANALYZE, as well as TELIO, in its entirety is to provide other research groups with a resource which may be studied, used and/or modified for their specific needs. As such, the packages are not intended to be a commercial product and thus there will be no support. It will be user supported software— you use it, you support it!

NOTE: This project is intended to make 3D motion analysis capabilities widely available to all human motion researchers. It is not to be used by commercial ventures to generate a product loosely based upon this work and then charge ridiculous fees for the resulting application. Thus, legalise will be included in the release version which will hopefully allow those for whom it is intended to use it freely and yet will prevent those who represent the antithesis of this project from obtaining financial gain.

References

1 Busby, H. R. and D. M. Trujillo (1985). *Numerical Experiments with a Differentiation Filter.* J Biomech Eng 107: 293-299.

2 Dohrmann, C. R., H. R. Busby, et al. (1988). *Smoothing Noisy Data using Dynamic Programming and Generalized Cross-Validation.* J. Biomech. Eng. 110: 37-41.

3 Fenton, R. G. and X. Shi (1989). *Comparison of Methods for Determining Screw Parameters of Finite Rigid Body Motion from Initial and Final Position data.* Advances in Design Automation-1989, Montreal,Quebec, ASME, 433-439.

4 Grood, E. S. and W. J. Suntay (1983). *A Joint Coordinate System for the Clinical Description of Three-Dimensional Motions: Application to the Knee.* J Biomech Eng 105: 136-144.

5 Jensen, R. K. (1989). *Changes in Segment Inertia Proportions Between 4 and 20 Years.* J Biomech 22(6/7): 529-536.

6 Kaleps, I., C. E. Clauser, et al. (1984). *Investigation into the Mass Distribution Properties of the Human Body and its Segments.* Ergonomics 27(12): 1225-1237.

7 McConville, J. T., T. D. Churchill, et al. (1980). *Anthropometric Relationships of Body and Body Segment Moments of Inertia.* United States Air Force Aerospace Medical Research Laboratory, AFAMRL-TR-80-119.

8 Woltring, H. J. (1986). *A Fortran package for generalized, Cross-validatory Spline Smoothing and Differentiation.* Adv. Eng. Software 8(2): 104-113.

9 Young, J. M., R. F. Chandler, et al. (1983). *Anthropometric and Mass Distribution Characteristics of the Adult Female.* Federal Aviation Administration, FAA-AM-83-16.