CDC8-35835 FR1

**Control Dynamics Company**
Office Park South, Suite 304, 600 Boulevard South, Huntsville, Alabama 35802
(205) 882-2650

December 3, 1986

National Aeronautics and Space Administration
Marshall Space Flight Center, AL 35812

Attention: Mr. Art Calsoni
AP25-G

Dear Mr. Calsoni:

Control Dynamics Company is pleased to submit the enclosed Final Report for the First phase of the Tether Applications under Contract Number NAS8-35835. Work performed subsequent to the initial tasks will be described in a later report. The follow-on tasks include: AMIGA graphics, deployer mechanism simulation, and sets of runs determined by MSFC personnel.

Dr. John R. Glaese
Principal Investigator

JRG/mfr

cc: PD32 25 copies
AT01 1 copy
PS01 5 copies
AS24 2 copies

(NASA-CR-179392) TETHER APPLICATIONS Interim Report (Control Dynamics Co.) 54 p N89-10934
CSCL 22B
Unclas
G3/18 0161079

# TETHER APPLICATIONS

INTERIM REPORT
NOVEMBER 1986

Sponsored By:

George C. Marshall Space Flight Center
Marshall Space Flight Center, Alabama 35812

Under:

Contract No: NAS8-35835

Prepared By:

Control Dynamics Company
600 Boulevard South, Suite 304
Huntsville, AL 35802

Table of Contents

## 1.0 Introduction

### 1.1 Statement of Work

The work involved in the current task is a continuation of work under a previous contract with MSFC. In the first contract a tether simulation was obtained from Analytical and Computational Mathematics (ACM) of Houston, Texas. It was implemented on the MSFC Sigma V computer. At that time several errors were found. These were corrected and documented in the final report for that contract.

The current effort has been extended to include more tasks, but it is the purpose here to discuss only the original statement of work for the continuation of tether analysis.

Task A. For a range of tether lengths, end masses, and orbits define/analyze tether deployment concepts from the Orbiter (tether motion, payout rate, length, tension, end mass position and disturbances) for steady state/dynamic and up/down deployments and from circular/eliptical orbits.

Task B. For a range of tether lengths, end masses, and orbits define/analyze end mass releasing concepts (imparted delta velocity and final orbits) for steady state and dynamic releases taking into account tether and end mass motion before and after release.

Task C. For a range of tether lengths, end masses, and orbits define/analyze tether retrieving or disposing concepts (tether motion, reel-in rate, length, and Orbiter position) for both reusable and disposable tethers.

Task D. Perform special tether analysis tasks.

Task E. Install/update existing tether programs on the MSFC VAX 11/780 computer.

Task F. Submit monthly letter reports and a final report and give a midterm and final review.

### 1.2 Overview of Work

The ACM tether simulation was brought up to speed on the HP 9000 located at Control Dynamics. During the verification process, errors were discovered in the code and corrected. This version was then implemented on the PD VAX 11/780 at MSFC as Version 1.0. A few more code errors were discovered and also corrected. A 4th order Runge-Kutta integration scheme and an initial attempt at constant tension deployment were added to the simulation and delivered as Version 2.0 at MSFC. MSFC implemented formatting changes for the output files and this became Version 2.1. A more refined tension controlled deployment scheme was developed, as were terms for material damping in the tether. These changes comprise Version 3.0 currently existing at MSFC.

A parallel effort to the work above was to update the simulation plotting capabilities on the MSFC Tektronics graphics terminals. A menu type program was developed from which the user can specify types of plots desired, plot limits, graph titles, and scale factors. Walking plot graphs were also developed which depict the tether motion as it deploys.

Control Dynamics gave the midterm presentation on February 7, 1986. This presentation included the work performed on the simulation through Version 2.1 and the status of the graphics.

A complete listing of the code would violate the license agreement between ACM and Control Dynamics, so only those lines originating with or modified by Control Dynamics can be published. Contact Control Dynamics or the Marshall Space Flight Center, Attention C. Rupp PS04, MSFC, AL 35812, regarding the license agreement.

## 2.0 Simulation Changes

### 2.1 Corrections to Simulation

As discussed in the overview, several errors in the original simulation were discovered and corrected. Appendix A shows the original lines of code, the corrections made, and the reasoning behind the changes.

These corrections involved changing integer loop variables which had been previously defined, removal of redundant steps, and corrections of dimensions and units.

### 2.2 Simulation Deletions

For Version 3.0 delivered at MSFC it was decided to delete all lines of code which had been commented out and never used during the first year of this project. Appendix B lists those lines which have been deleted and the subroutines in which they had appeared.

### 2.3 Simulation Additions

During the past year there have also been numerous miscellaneous additions to the simulation not covered under the corrections to the simulation or under the topics of tether tension deployment, damping or the plotting additions. These additions are listed in Appendix C. The main addition involved incorporating a second integration scheme. A 4th Order Runge-Kutta integration routine was incorporated to help in decreasing run times or when it is desired to have a constant step size.

An addition which was requested by MSFC was to have several orbital parameters included in the LGIBLE file output. This change was made in the Subroutine Uprint and the variables are the semi-major axis, the eccentricity, the argument of perigee, the inclination, the ascending node, the true anomaly, the perigee, the apogee, and the longitude of perigee. These terms are all derived from the EVECEL(6) vector calculated in Subroutine Vecele.

Capabilities were also added to Subroutine Deplex to tell the user if the tether is fully deployed or fully retracted. If the tether length has reached either of these limits, then the deployment scheme is switched to constant length deployment and the deployment velocity and acceleration are set to zero.

There were also numerous changes made to the format statements in Subroutine Uprint. MSFC made these changes internally and relayed them to Control Dynamics to implement in the Control Dynamics' subsequent versions of the simulation.

## 2.4 Revised Subroutine Descriptions

| | |
|---|---|
| Tether (Main) | main driver, checks for restart case |
| **Initialization** | |
| Ubegin | initializes flags and constants; thrust maneuvers if required |
| Uinitl | initializes integration variables |
| Rbegin | same as Ubegin for restart case |
| Rinitl | same as Uinitl for restart case |
| Coot | sets mathematical and physical constants |
| Earth | sets zonal and harmonic terms for non-spherical earth |
| Uxtoa | rearranges integration variables for integrator |
| Rotsys | transformation matrix [RI] (inertial to rotating) |
| **Program Control** | |
| Ugen | control for integration loop |
| Timing | time control for thrust maneuvers |
| Switch | thrust maneuver control |
| Ustop | sets stop flag |
| Trackf | follows max and min tension on tether |
| Uprint | routine to print data to output, plot and restart files |
| Vecele | computes true anomaly for output |
| Rwrite | prints data for restart input |
| **Integration Loop** | |
| Uint | driver |
| RK78 | Runge-Kutta-Fehlberg 7(8) routine and 4th order Runge-Kutta routine |
| **Derivative Calculations** | |
| Uder | driver |
| Uatox | places variables in form used to calculate forces |
| Help | sets intermediate values for derivative evaluation |
| Deplex | computes time dependent quantities according to deployment laws |
| Tendpl | computes tether velocity and components of tether acceleration for tension deployment |
| Force calculations: | |
| Kinema | kinematic terms |
| Centra | central force field |
| Bendin | bending stiffness |
| Ginteg | intermediate values used in Bendin |
| Ghalfi | " |
| Thrust | forces due to thrust maneuvers |
| Normal | tether tension |
| Inext | inextensible tether |
| Homoge | homogeneous tether |
| Fulext | fully extensible tether |
| Basmat | called by Inext or Homoge to create intermediate matrix |
| Linequ | called by Inext or Homoge to solve intermediate matrix |
| Pertur | external perturbations |
| Unosph | perturbations due to non-sphericity of earth |

| | |
|---|---|
| Erdpot | standard routine for calculation of higher order zonal and harmonic terms |
| Uthird | third body (sun and moon) perturbations |
| Usolun | intermediate routine to prevent calculating positions more than once every time step |
| Solun | analytical ephemeris of sun and moon |
| Kepequ | solves Kepler equation |
| Ngrav | non-gravitational terms (drag and radiation forces) |
| Vites | velocity with respect to earth's atmosphere |
| Airden | air density from Jaccia model |
| Densit | air density from simple exponential model |
| Solcon | momentum flux due to solar radiation |
| Eclips | sets eclipse flag |
| Rot | transformation matrix [TI] (inertial to tether) |
| Radiat | radiation pressure |

## 2.5 Revised Variables List

| Variable | Common | Variable | Common |
|---|---|---|---|
| * A1 | Atmos | Ipyes | Dripri |
| | Normco | Irad | Force |
| * A2 | Atmos | Iresta | Restar |
| | Normco | Irsav1 | Scount |
| A3 | Atmos | Irsav2 | Scount |
| Alpha | Deplaw | Islack | Slack |
| * B1 | Atmos | Isun | Thirdb |
| | Normco | Iteg | Graton |
| * B2 | Atmos | Ithird | Force |
| | Normco | Iyes | Dripri |
| * B3 | Atmos | Kstep | Restar |
| | Normco | Latt | Atmos |
| Beta | Tespec | Lonela | Tespec |
| Bstiff | Bend | Nactiv(2) | Thrus3 |
| C1 | Normco | Ndeq | Gen |
| C2 | Normco | Nend(2) | Thrus3 |
| Dac | Tdeppa | Nfmas | Traqua |
| Dacini | Tindpa | Nfmis | Traqua |
| Deg | Cbasic | Nodes | Tindpa |
| Direc(2,3) | Thrus2 | Nprint | Dripri |
| Dt | Umesh | Npuls(2) | Thrus1 |
| Dvn | Tdeppa | Nsat(2) | Thrus4 |
| Dvnini | Tindpa | Nstep | Gen |
| Em | Atmos | Nstop | Stopva |
| Epsj2 | Oblate | Nswitc(2) | Thrus3 |
| Fc | Deplaw | Ntess | Oblate |
| Formas | Traqua | Nthrus | Thrus1 |
| Formis | Traqua | Ntype(2) | Thrus2 |
| Fr | Deplaw | Nzon | Oblate |
| Frk | Deplaw | Omq50r | Cdynae |
| Forpro(19) | Projec | Omt50r | Cdynae |
| Gamma | Tespec | P(19) | Cocoef |
| G1 | Bouwei | Param | Gen |
| G2 | Bouwei | Pdurat(2) | Thrus1 |
| Ibend | Bend | Phi | Atmos |
| Idepl | Deplaw | Pi | Cbasic |
| Idurn | Atmos | Pih | Cbasic |
| Idrag | Force | Pinter(2) | Thrus1 |
| Imoon | Thirdb | R | Cart |
| Inosph | Force | Rad | Cbasic |
| Iplot | Dripri | | |
| Iplt | Dripri | | |

* Variables that occur in two common blocks never occur in the same subroutine. They represent two different values. In this sense, they act as dummy variables.

| Variable | Common |
|---|---|
| Relvel(19,3) | Relco |
| Relvqu(19) | Interm |
| Rho | Atmos |
| Rnode(19,3) | Abscoo |
| Rnodea(19) | Abscoo |
| Rshu | Shutco |
| Rshu2 | Shutco |
| Rshutt(3) | Shutco |
| S1(19) | Cocoef |
| S2(19) | Cocoef |
| S3(19) | Cocoef |
| S4(19) | Cocoef |
| Shmass | Emass |
| Shrefc | Radcoe |
| Shucro | Shape |
| Std50r | Cdynae |
| Stepin | Gen |
| Subcro | Shape |
| Sumass | Emass |
| Surefc | Radcoe |
| T | Umesh |
| Talpha(2) | Thrus2 |
| Tau(19,3) | Relco |
| Taumag(19) | Interm |
| Tauqua(19) | Interm |
| Tbeta(2) | Thrus2 |
| Tdepoc | Jdate |
| Tdnow | Jdate |
| Tedens | Tespec |
| Tess(2,36) | Ckufue |
| Tetrad | Shape |
| Tiipl(19) | Interm |
| Tini | Tindpa |
| Tlaini | Actual |
| Tlevel(2) | Thrus2 |

| Variable | Common |
|---|---|
| Tln | Tdeppa |
| Tln1 | Retrap |
| Tln2 | Retrap |
| Tlnin1 | Retrap |
| Tlnin2 | Retrap |
| Tlnini | Tindpa |
| Tplast | Dripri |
| Tprint | Dripri |
| Trefc | Radcoe |
| Tstart(2) | Thrus1 |
| Tstop | Stopva |
| Twopi | Cbasic |
| Vnode(19,3) | Abscoo |
| Vshutt(3) | Shutco |
| X(7) | Cart |
| Xincre | Tindpa |
| Xmass0 | Tindpa |
| Xmass1 | Tindpa |
| Xmass2 | Tdeppa |
| Xmu | Cpoems |
| Xmum | Cpoems |
| Xmus | Cpoems |
| Xnforc(19) | Nforce |
| Y(123) | Umesh |
| Z(123) | Umesh |
| Zl(4) | Jposms |
| Zonal(23) | Ckufue |
| Zs(4) | Jposms |

## 2.6 REVISED COMMON BLODK LIST

/ABSCOO/ RNODE(19,3), RNODEA(19),VNODE(19,3)

RNODE(i,j) = component j of the position $\underline{\hat{r}}_i$ of the node i referred to geocentric inertial reference frame.

RNODEA(i) = $|\underline{\hat{r}}_i|$

VNODE(i,j) = component j of the velocity $d\underline{\hat{r}}_i/dt$ of the node i referred to the geocentric inertial reference frame.

These quantities are computed in the subroutine UATOX.

Used in subroutines: UATOX, UPRINT, CENTRA, PERTUR, NGRAV.

/ACTUAL/ TLAINI

TLAINI = deformed length of the tether at initial time. Input quantity.

Used in subroutines: UBEGIN, HEAD, RBEGIN.

/ATMOS/ PHI, EM, RHO, A1, A2, A3, B1, B2, B3, IDIURN, LAT

Constants and flags for the atmospheric model. Input quantities.

Used in subroutines UINITL, RINITL, UBEGIN, RBEGIN, HEAD, AIRDEN.

/BEND/ BSTIFF, IBEND

Flag and constant for the bending stiffness. Input quantities.

BSTIFF=EJ,E=YOUNG's MODULUS $J = \dfrac{\pi a^4}{4}$

Used in subroutines: UINITL, RINITL, UBEGIN, RBEGIN, HEAD, UDER, BENDIN.

/BOUWEI/ G1, G2

The weights for the extrapolation formulas of the discretization at boundaries. They are set

$$G1 = 1.5, \quad G2 = -0.5$$

in the subroutines UINITL or RINITL.

Used in subroutines: UINITL, RINITL, HELP, BENDIN, NORMAL.

/CART/ X(7), R

X(1-3) = cartesian position vector

X(4-6) = cartesian velocity vector

X(7) = time

R = magnitude of cartesian position vector X(1-3)

These quantities are set in subroutine PERTUR and used locally for the evaluation of the gravitational perturbations.

Used in subroutines: PERTUR, UNOSPH, UTHIRD.

/CBASIC/ PI, TWOPI, PIH, DEG, RAD

Basic mathematical constants.

PI = $\pi$, TWOPI = $2\pi$, PIH = $\pi/2$, DEG = $180/\pi$,

RAD = $\pi/180$.

They are set in subroutine COOT.

Used in subroutines: UINITL, RINITL, COOT, UPRINT, VECELE, KEPEQU, THRUST.

/CDYNAE/ STD50R, OMT50R, OMQ50R †

STD50R, OMT50R, OMQ50R = constants to compute the angle W between Greenwich meridian and mean equinox as a function of the Modified Julian Date in radians.

These quantities are set in subroutine COOT.

Used in subroutines: COOT, UNOSPH.

/CKUFUE/ ZONAL(23), TESS(2,36)

ZONAL(1-23) = zonal coefficients of earth geopotential

TESS(L,1-36) = tesseral coefficients of earth geopotential

TESS(1,L*L-L)/2+M) = C(L,M)

TESS(2,(L*L-L)/2+M) = S(L,M)

These quantities are set in subroutine EARTH.

Used in subroutines: UINITL, RINITL, EARTH, UNOSOPH, ERDPOT.

/COCOEF/ S1(19), S2(19), S3(19), S4(19), P(19)

Constants and coefficients for system of linear equations. Matrix with four non-vanishing diagonals.

Locally used in subroutines: HOMOGE, INEXT, BASMAT.

/CPOEMS/ XMU, XMUM, XMUS

XMU = gravitational constant of the earth

XMUM = gravitational constant of the moon

XMUS = gravitational constant of the sun

These quantities are set in subroutine COOT.

Used in subroutines: UINITL, RINITL, COOT, VECLE, CENTRA, UTHIRD.

† See also: Definition and units in the FORTRAN source listing of the subroutine COOT.

/DEPLAW/ ALPHA, IDEPL, FC, FRK, FR

The flag IDEPL specifies the explicit deployment/retraction law and if

IDEPL

= 1 : $\bar{\ell}(t) = \bar{\ell}(t_0) = \text{const}$

= 2 : $\dot{\bar{\ell}}(t) = \dot{\bar{\ell}}(t_0) = \text{const}$

= 3 : $\ddot{\bar{\ell}}(t) = \ddot{\bar{\ell}}(t_0) = \text{const}$

= 4 : $\dot{\bar{\ell}}(t) = \alpha\bar{\ell}(t)$ . (exponential deployment law)

= 5 : tension deployment law

$t_0$ is the initial time.

ALPHA = $\alpha$ for exponential deployment law.

FC = commanded max tension @ n-1

FRK = tension coefficient

FR = tension @ n-1

Used in subroutines: UINITL, RINITL, UBEGIN, RBEGIN, DEPLEX, UXTOA, TENDPL, RWRITE, UATOX, UDER, UGEN, UPRINT.

/DRIPRI/ NPRINT, IPYES, TPLAST, TPRINT, IYES, IPLOT, IPLT

Output control.

NPRINT, TPRINT: Input quantities.

IPYES : flag set >0 in subroutine UGEN if results of the just performed integration step have to be printed

TPLAST : last print time

IYES : stop flag set in USTOP

IPLOT : counter for plot file

IPLT : counter for walking plot file

Used in subroutines: UBEGIN, RBEGIN, RGEN, UPRINT.

/EMASS/SUMASS, SHMASS

SUMASS = mass of the subsatellite without retracted part of the tether. Input quantity.

SHMASS = mass of the shuttle without retracted part of the tether. Input quantity.

Used in subroutines: RINITL, UBEGIN, RBEGIN, HEAD, DEPLEX, TENDPL.

/FORCE/INSOPH, ITHIRD, IDRAG, IRAD

INOSPH = determines whether or not non-sphericity perturbations are to be computed. Input quantity. (=1, no; =2, yes)

THIRD = determines whether or not third body perturbations are to be computed. Input quantity. (=1, no; =2, yes)

IDRAG = determines whether or not drag perturbations are to be computed. Input quantity.

- =1 : No perturbations due to drag.
- =2 : Perturbations using simple air density model.
- =3 : Perturbations using Jacchia air density model.

IRAD = determines whether or not radiation perturbations are to be computed. Input quantity.

- =1 : No radiation perturbations.
- =2 : Perturbations according to an averaged solar.
- =3 : Perturbations according to a sinusoidally varying solar constant with earth eclipse effects included.

Used in subroutines: UINITL, RINITL, UBEGIN, RBEGIN, HEAD, UDER, PERTUR, NGRAV.

/GEN/ STEPIN, NDEQ, NSTEP, PARAM

STEPIN = initial guess for the integration step. Input quantity.

NDEQ = number of differential equations. Computed in subroutine UGEN

NSTEP = counts the number of integration steps.

PARAM = tolerated truncation error on each integrated quantity. Input.

Used in subroutines: UINITL, RINITL, UBEGIN, RBEGIN, HEAD, UGEN, UDER, UPRINT, RWRITE, USTOP, UINT.

/GRATON/ ITEG

ITEG : flag for integration scheme, 1 variable stepsize 2-4th order Runge - Kutta

Used in subroutines: RBEGIN, RK78, UBEGIN

/INTERM/ TAUQUA(19), TAUMAG(19), TIIP1(18), RELVQU(19)

TAUQUA(i) = $|\underline{\tau}_{i+1/2}|^2$

TAUMAG(i) = $|\underline{\tau}_{i+1/2}|$

TIIP1(i) = $[\underline{\tau}_{i+1/2} \cdot \underline{\tau}_{i+3/2}]$

RELVQU(i) = $|\underline{w}_{i+1/2}|^2$

These quantities are computed in the subroutine HELP and will be used for the evaluation of the $\underline{\tau}_{i+1/2}$ and $\underline{w}_{i+1/2}$ integration variables.

Used in subroutines: UPRINT, TRACKF, HELP, BENDIN, THRUST, HOMOGE, FULEXT, INEXT, BASMAT, DEPLEX, TENDPL, UDER.

/JDATE/ TDEPOC, TDNOW

TDEPOC = is the Julian date at initial time (since Jan. 1, 1950).

TDNOW = is the Julian date in days after each step of propagation.

Used in subroutines: UBEGIN, RBEGIN, UGEN, RWRITE, USTOP, PERTUR.

/JPOSMS/ ZS(4), ZL(4) .

ZS(1-3), ZS(4) = position and distance of the sun referenced to the equatorial system of the earth.

ZL(1-3), ZL(4) = position and distance of the moon referenced to the equatorial system of the earth.

Used in subroutines: UTHIRD, USOLUN, NGRAV, AIRDEN.

/NFORCE/ XNFORC(19)

XNFORC(i) = $N^*_{i+1/2}$ , tether tension magnitude x factor

Used in subroutines: RBEGIN, UPRINT, TRACKF, RWRITE, NORMAL, HOMOGE, FULEXT, INEXT, DPLEX, TENDPL, UDER.

/NORMCO/ A1, A2, B1, B2, B3, C1, C2

Intermediate coefficients used to calculate normal forces. These quantities will be computed in subroutine HELP.

Used in subroutines: HELP, BENDIN, NORMAL, INEXT, BASMAT.

/OBLATE/ EPSJ2, NZON, NTESS

EPSJ2 = $3/2\ J_2\ r_e^2$, computed in subroutine UINITL or RINITL.

where

$J_2$ = second zonal coefficient of earth, set in subroutine EARTH

$r_e$ = equatorial mean radius of earth, set in subroutine EARTH.

NZON = the desired number of zonal terms to be included in non-sphericity model. Input quantity.

NTESS = the desired number of tesseral terms to be included in non-sphericity model. Input quantity.

Used in subroutines: UINITL, RINITL, UBEGIN, HEAD, UNOSPH, RBEGIN.

/PROJEC/ FORPRO(19)

These are scalar products computed in the subroutine NORMAL and used to calculate tether tension.

Locally used in subroutines: NORMAL, INEXT, BASMAT.

/RADCOE/ TREFC, SUREFC, SHREFC

Reflection coefficients for the computation of the radiation pressure. Input quantities.

TREFC = reflection coefficient for the tether

SUREFC = reflection coefficient for the subsatellite; currently not used

SHREFC = reflection coefficient for the shuttle; currently not used

Used in subroutines: UINITL, RINITL, UBEGIN, RBEGIN, NGRAV.

/RELCO/ TAU(19,3) RELVEL(19,3)

These are the integration variables.

TAU(i,j) = component j of the vector $\underline{\tau}_{i+1/2}$

RELVEL(i,j) = component j of the vector $\underline{w}_{i+1/2}$

Used in subroutines: UBEGIN, RBEGIN, UATOX, UXTOA, UDER, UPRINT, RWRITE, HELP, KINEMA, CENTRA, BENDIN, GINTEG, GHALFI, NGRAV, THRUST, NORMAL, DEPLEX, TENDPL, KINMAX, FULEXT, HOMOG.

/RESTAR/ IRESTA, KSTEP

IRESTA = determines whether the case to be run is a new case or a restart case. Input quantity.

KSTEP = number of step on the restart file from which a restart case has to be initialized.

If KSTEP does not coincide with a printed integration step the program will use the next largest printed step for initialization conditions of the restart case.

Used in subroutines: HEAD, RBEGIN.

/RETRAP/ TLNIN1, TLNIN2, TLN1, TLN2

TLNIN1 = undeformed length of that part of the tether which is initially retracted in the subsatellite. Input quantity.

TLNIN2 = undeformed length of that part of the tether which is initially retracted in the shuttle. Input quantity.

TLN1 = undeformed length of that part of the tether which is retracted in the subsatellite at actual time t. Computed in subroutine DEPLEX. Constant in current version.

TLN2 = undeformed length of that part of the tether which is retracted in the shuttle at actual time t. Computed in subroutine DEPLEX.

Used in subroutines: UBEGIN, RBEGIN, HEAD, UPRINT, RWRITE DEPLEX, USTOP, TENDPL, UDER.

/SCOUNT/ IRSAV1, IRSAV2

IRSAV1 = number of rejected integration steps since start of the run.

IRSAV2 = number of rejected integration steps since last printout time.

Used in Subroutines: UPRINT, UINT.

/SHAPE/ TETRAD, SUBCRO, SHUCRO

The geometrical quantities of this COMMON block will be used for the evaluation of air drag and radiation pressure forces.

TETRAD = Radius of tether cross-section. Input quantity. To be given in mm.

SUBCRO = Cross-sectional area of subsatellite. Input quantity. To be given in $m^2$.

SHUCRO = Cross-sectional area of shuttle. Input quantity. To be given in $m^2$.

Used in subroutines: UINITL, RINITL, UBEGIN, RBEGIN, NGRAV.

/SHUTCO/ RSHUTT(3), VSHUTT(3), RSHU2, RSHU

Cartesian coordinates and velocities of the shuttle referred to the geocentric inertial reference frame. RSHUTT and VSHUTT are integration variables.

RSHUTT(3) = $\underline{r}_n(t)$

VSHUTT(3) = $\underline{v}_n(t)$

RSHU2 = $|\underline{r}_n(t)|^2$

RSHU = $|\underline{r}_n(t)|$

RSHU2 and RSHU will be computed in the subroutine UATOX.

Used in subroutines: UBEGIN, RBEGIN, ROTSYS, UATOX, UXTOA, UPRINT, RWRITE, KINEMA, CENTRA, PERTUR, NGRAV, DEPLEX, TENDPL, KINMAX.

/SLACK/ ISLACK

ISLACK is a flag used only in connection with the model for the fully extensible tether. ISLACK is an input quantity and if

ISLACK equal 1 : Negative tether tensions accepted.

ISLACK not equal 1 : Negative tether tensions not accepted, the corresponding normal forces are set zero by the program.

Used in subroutines: UINITL, RINITL, UBEGIN, RBEGIN, HEAD, FULEXT.

/STOPVA/ NSTOP, TSTOP

These are input quantities causing a termination of the execution when the number of integration steps exceeds NSTOP or when the integration time exceeds TSTOP. If negative values are assigned to NSTOP or TSTOP, the corresponding stopping condition is not active.

Used in subroutines: UBEGIN, RBEGIN, USTOP.

/THIRDB/ ISUN, IMOON

Flags to switch on or off the gravitational perturbations due to sun and moon and if

ISUN

= 0 perturbation due to the sun off

= 1 perturbation due to the sun on

IMOON

= 0 perturbation due to the moon off

= 1 perturbation due to the moon on

Used in subroutines: UINITL, RINITL, UBEGIN, RBEGIN, HEAD, UTHIRD.

/THRUS1/ TSTART(2), PDURAT(2), PINTER(2), NTHRUS, NPULS(2)

Parameters to specify thrust maneuvers.

NTHRUS : Number of thrust maneuvers. NTHRUS May be equal 0,1, or 2. Input.

TSTART(J), J=1,2: Start time of thrust maneuvers J. Input.

NPULS(J), J=1,2: Number of pulses of thrust maneuver J. Input.

PDURAT(J), J=1,2: Duration of one pulse of thrust maneuver J. Input.

PINTER(J), J=1,2: Time interval between subsequent pulses of thrust maneuver J. Input.

Used in subroutines: UINITL, RINITL, UBEGIN, RBEGIN, HEAD, TIMING, SWITCH, UDER, THRUST.

/TDEPPA/ TLN, DVN, DAC, XMASS2

Time dependent parameters for deployment or retraction. They will be computed in the subroutine DEPLEX.

TLN = $\bar{\ell}(t)$ : Undeformed tetherlength at the time t.

DVN = $\frac{d\bar{\ell}}{dt}$ : Deployment/ retraction velocity

DAC = $\frac{d^2\bar{\ell}}{dt^2}$ : Deployment/ retraction acceleration

XMASS2 = $m_2(t)$ : Mass of shuttle including retracted part of the tether

Used in subroutines: UPRINT, RWRITE, DEPLEX, HELP, KINEMA, BENDIN, NGRAV, THRUST, NORMAL, HOMOGE, FULEXT, INEXT, UXTOA, USTOP, TENDPL, KINMAX, RBEGIN, UATOX, UBEGIN, UDER.

/TESPEC/ TEDENS, BETA, GAMMA, LONELA

Material tether quantities specified by input.

TEDENS = $\mu$ : Specific mass of the tether (g/m)

LONELA = Flag to choose the model for the longitudinal deformation and if LONELA

= 1 Inextensible tether

= 2 Homogeneous longitudinal deformation

= 3 Fully extensible tether.

BETA = $\beta$ : Longitudinal stiffness

= EF E = Young's modulus ($Nt/m^2$)

F = $r^2\pi$ area of tether cross section ($m^2$)

GAMMA = Damping coefficient

Used in subroutines: UINITL, RINITL, UBEGIN, RBEGIN, HEAD, UDER, UPRINT, DEPLEX, HELP, KINEMA, NGRAV, NORMAL, HOMOGE, FULEXT, TENDPL, KINMAX, UATOX, UGEN.

/THRUS2/ TLEVEL(2), NTYPE(2), TALPHA(2), TBETA(2), DIREC(2,3)

Parameters to specify thrust maneuvers.

TLEVEL(J), J=1,2 : Thrustlevel in Newton. Input.

NTYPE(J), J=1,2 : Specifies type of input giving the thrust direction and if NTYPE(J)

=1 : Direction of the thrust force of maneuver J given by the direction vector DIREC(J,3).

=2 : Direction of the thrust force of maneuver J given by the angles TALPHA(J) and TBETA(J).

TALPHA (J), J=1,2: $\alpha_T$ for thrust maneuver J (see Figure 2.6.1). Input.

TBETA (J), J=1,2: $\beta_T$ for thrust maneuver J (see Figure 2.6.1). Input.

DIREC(J,I), J=1,2, I=1,2,3: Components of the direction of the thrust force of maneuver J referred to the orbital reference frame. Calculated from DIREC1 and DIREC2 in UINITL.

Used in subroutines: UNITL, RINITL, UBEGIN, RBEGIN, HEAD, THRUST.

/THRUS3/ NACTIV(2), NSWITC(2), NEND(2)

Flags and counters for thrust maneuver control.

NACTIV(J), J=1,2: Specifies whether the thruster of maneuver J is active or not and if NACTIV(J)

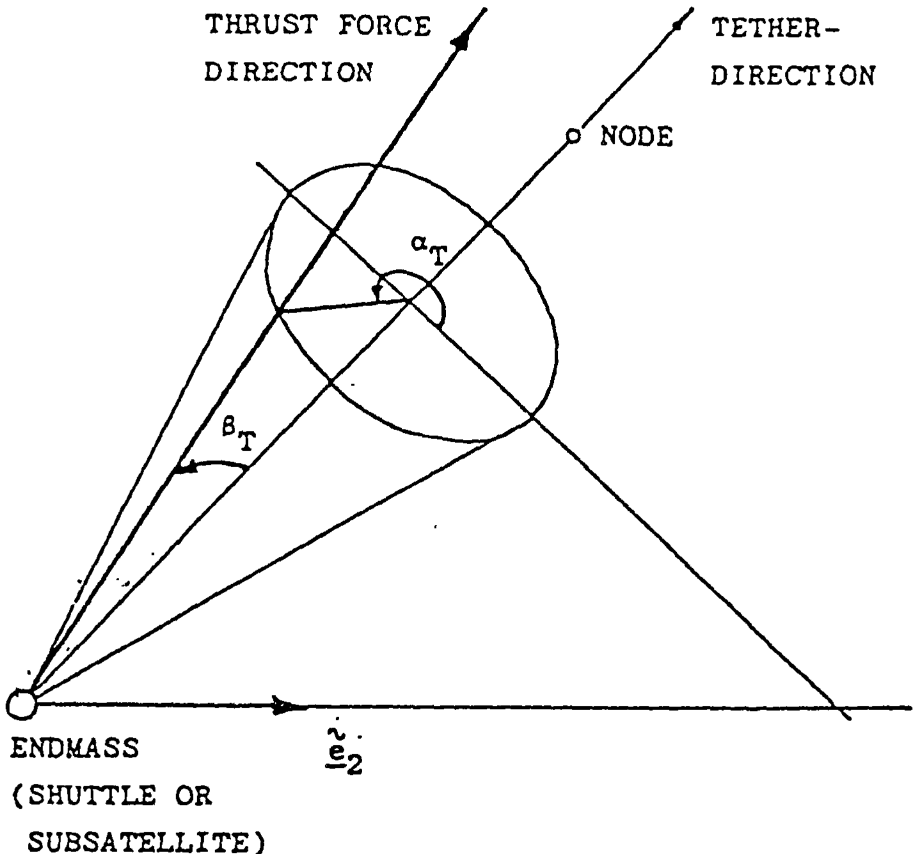
= 0 Thruster not active.

= 1 Thruster active. This flag is set in the subroutine SWITCH.

NSWITC(J) J=1,2: Counts the "switch on" - and "switch off" - actions of maneuver J in order to set the end flag NEND(J). This counter is updated in the subroutine SWITCH.

NEND(J) J=1,2: End flag for the thrust maneuver J. The flag is initialized to 0 in the subroutines UINITL or RINITL and is set equal to 1 in the subroutine SWITCH if the thrust maneuver J has been completed.

Used in subroutines: UINITL, RINITL, UBEGIN, RBEGIN, HEAD, UGEN, TIMING, SWITCH, UDER, THRUST.



THRUST FORCE
DIRECTION
TETHER-
DIRECTION
NODE
$\alpha_T$
$\beta_T$
ENDMASS
(SHUTTLE OR
SUBSATELLITE)
$\underset{\sim}{e}_2$


Figure 2.6.1.

/THRUS4/ NSAT(2)

NSAT(J) J=1,2 : Flag which specifies whether the maneuver J has to be applied on the shuttle or on the subsatellite and if NSAT(J).

= 1 maneuver J to be applied on the shuttle,

= 2 maneuver J to be applied on the subsatellite.

Used in subroutines: UINITL, RINITL, UBEGIN, RBEGIN, HEAD, UDER, THRUST.

/TINDPA/ TINI, TLNINI, DVNINI, DACINI, XINCRE, XMASSO,

XMASS1, NODES

TINI $= t_0$ : Initial time for a given explicit deployment/retraction law. In the current version of the program. TINI is set zero in the subroutine UGEN. (At the same place, the initial value of the integration time T is set zero).

TLNINI $= \bar{\ell}(t_0)$: Initial value of the undeformed deployed tetherlength. Input.

DVNINI $= \dot{\bar{\ell}}(t_0)$: Initial value of the deployment velocity. Input.

DACINI $= \ddot{\bar{\ell}}(t_0)$: Initial value of the deployment acceleration. Input.

NODES $= n$ : The number of spatial nodes including the end-masses. Input.

XINCRE $= h = 1/(n-1)$ : Increment of the spatial discretization. XINCRE is initialized in the subroutine UBEGIN or RBEGIN.

XMASSO : Dummy variable. Not used in the current version of the program.

XMASS1 $= m_1(t)$: Mass of the subsatellite including retracted part of the tether. In the current version of the program which does not treat deployment from the subsatellite, XMASS1 is set equal to SUMASS In the subroutine UBEGIN.

Used in subroutines: UINITL, RINITL, UBEGIN, RBEGIN, HEAD, UGEN, UATOX, UXTOA, UDER, UPRINT, TRACKF, RWRITE, DEPLEX, HELP, KINEMA, CENTRA, BENDIN, GINTEG, GHALFI, PERTUR, NGRAV, THRUST, NORMAL, HOMOGE, FULEXT, INEXT, BASMAT, USTOP, TENDPL, KINMAX.

/TRAQUA/ FORMAS, FORMIS, NFMAS, NFMIS

FORMAS : Maximum tethertension with respect to all nodes and all integration steps between subsequent print times. The quantity is tracked in the subroutine TRACKF.

NFMAS : Node where FORMAS appears. The quantity is tracked in the subroutine TRACKF.

FORMIS : Minimum tethertension with respect to all nodes and all integration steps between subsequent print times. The quantity is tracked in the subroutine TRACKF.

NFMIS : Node where FORMIS appears. The quantity is tracked in the subroutine TRACKF.

Used in subroutines: UPRINT, TRACKF.

/UMESH/ T, DT, Y(123), F(123)

Allocation of arrays for the numerical integration.

T : Time (independent variable).

DT : Stepsize

Y(184) : One dimensional array of integration variables.

F(1472) : Auxiliary array.

Used in subroutines: UGEN, UPRINT, TRACKF, UINT, DEPLEX, TENDPL.

2.7 Revised Input File Example

```
************************************************************************
        Input data for tethered satellites computation
************************************************************************

        Iresta          Flag to determine input for new or restart case
          F             A new run is initialized
          T             A restart case is initialized.  The state of the
                        system is read from the restart file at step>kstep

        Iresta=F,Kstep=4000

************************************************************************

        Inosph          Flag to switch on/off the perturbation due to the
                        nonsphericity of the earth
          1             Off
          2             On
        Nzon            Order of expansion of the earth potential in zonal
                        terms.  Nzon = 0,23
        Ntess           Order of expansion of the earth potential in
                        tesseral terms.  Ntess = 0, ..., 8

        Inosph=1,Nzon=23,Ntess=8

************************************************************************

        Ithird          Flag to switch on/off the perturbations due to
                        third bodies
          1             Off
          2             On
        Isun            Flag to switch on/off the perturbation due to the
                        sun
          0             Off
          1             On
        Imoon           Flag to switch on/off the perturbation due to the
                        moon
          0             Off
          1             On

        Ithird=1,Isun=0,Imoon=0

************************************************************************

        Idrag           Flag to switch on/off the perturbation due to the
                        air drag
          1             Off
          2             On, simple exponential density model
          3             On, sophisticated density model (fit to Jaccia-
                        1973)
        Idiurn          Flag to switch on/off diurnal effects for Idrag=3
          F             Off
          T             On
```

```
Latt            Flag to switch on/off latitudinal effects for
                Idrag=3
  F             Off
  T             On

Idrag=3,Idiurn=T,Latt=T
```

************************************************************************

```
Tetrad          Radius of tether crossection (mm)
Shucro          Shuttle crossection (m**2)
Subcro          Subsatellite crossection (m**2)

Tetrad=.4d0,Shucro=50.d0,Subcro=5.d0
```

************************************************************************

```
Phi             Atmospheric bulge lag angle (deg)
Em              Model exponent
Rho             Atmospheric density at earth surface (kg/m**3)
A1,A2,A3,       Model fit paramters (fit to Jaccia model)
B1,B2,B3

Phi=37.d0,Em=2.75d0,Rho=1.225d0
A1=-.24436605d2,A2=-.14473998d-1,A3=.89553181d3
B1=-.21795610d0,B2=.33153996d-2,B3=-.24610244d2
```

************************************************************************

```
Irad            Flag to switch on/off the perturbation due to the
                radiation pressure
  1             Off
  2             On, solar constant is averaged
  3             On, include variations in solar radiation due to
                earth's orbit
Trefc           Reflection coefficient for tether
Shrefc          Reflection coefficient for shuttle
Surefc          Reflection coefficient for subsatellite

Irad=1,Trefc=.8d0,Shrefc=.8d0,Surefc=.8d0
```

************************************************************************

```
Ibend           Flag to switch on/off the bending stiffness
  1             Bending stiffness neglected
  2             Bending stiffness included
Bstiff          Bending stiffness (kg*km**3/sec**2)
                Bstiff=EJ
                E=Young's modulus
                J= Πa**4/4
                a=tether radius (km)

Ibend=1,Bstiff=5.625d-4
```

************************************************************************

| | |
|---|---|
| Lonela | Specifies model for longitudinal deformation |
| 1 | No longitudinal deformation; tether forces determined by constraint equations |
| 2 | Homogeneous longitudinal deformation; tether forces determined by constraint equations and by Hooke's law for the segment next to the shuttle |
| 3 | Extensible tether; tether forces determined by Hooke's law |
| Beta | Longitudinal elasticity (constant in Hooke's law)<br>Beta=EA<br>E=Young's modulus (Nt/m**2)<br>A=cross sectional area of tether (m**2) |
| Gamma | Damping coefficient |
| Islack | Slackness flag only for extensible tether |
| 1 | Negative tether tensions accepted |
| 2 | Negative tether tensions not accepted but set to zero |

Lonela=1,Beta=5.0d4,Gamma=5.d0,Islack=1

**************************************************************************

| | |
|---|---|
| Nthrus | Number of thrust maneuvers (0,1,2) |
| Tstart(j) | Start time for thrust maneuver (sec), j=1,2 |
| Npuls(j) | Number of pulses for thrust maneuver j |
| Pdurat(j) | Duration of pulse for thrust maneuver j (sec) |
| Pinter(j) | Interval between pulses for maneuver j (sec) |

Nthrus=0,Tstart=100.d0,500.d0,Npuls=3,3
Pdurat=50.d0,50.d0,Pinter=50.d0,0.d0

**************************************************************************

| | |
|---|---|
| Tlevel(j) | Thrustlevel (Nt) of maneuver j |
| Ntype(j) | Specifies type of input giving thrust direction |
| 1 | Thrust direction for maneuvers 1&2 to be given by direction vectors Direc1(3) and Direc2(3) respectively, with respect to the orbital reference frame (normalized automatically) |
| 2 | Thrust direction to be given by angle Talpha(j) and Tbeta(j) in degrees with respect to the tether direction (see Figure 2.7.1) |
| Nsat(j) | Specifies whether maneuver j is done at the shuttle or subsatellite |
| 1 | Maneuver j at the shuttle |
| 2 | Maneuver j at the subsatellite |

Tlevel=100.d0,20.d0,Ntype=2,2
Direc1=0.d0,0.d0,0.d0
Direc2=0.d0,0.d0,0.d0
Talpha=45.d0,0.d0,Tbeta=45.d0,180.d0
Nsat=2,2

**************************************************************************

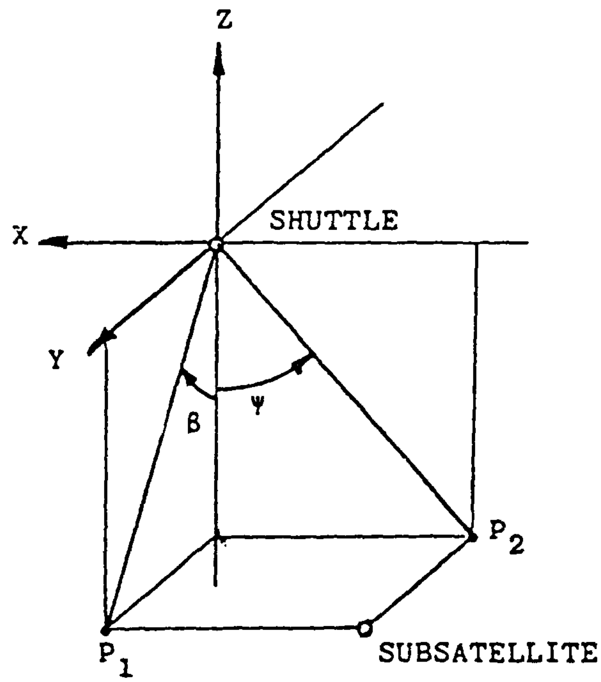



FIGURE 2.7.1

| | | |
|---|---|---|
| X,Y,Z | : | Orbital reference frame |
| $P_1$ | : | Projection of subsatellite to YZ-Plane |
| $P_2$ | : | Projection of subsatellite to XZ-Plane |
| Ψ | : | In-plane angle |
| β | : | Out-of-plane angle |

| | |
|---|---|
| Nodes | Number of spatial nodes including the endmasses (4-30) |
| Stepin | Initial guess for integration time step |
| Param | Tolerated truncation error for integration |

Nodes=6,Stepin=1.d-2,Param=1.d-8

*************************************************************************

| | |
|---|---|
| Idepl | Specifies tether deployment mode |
| 1 | Constant commanded tether length |
| 2 | Constant commanded deployment/retraction velocity |
| 3 | Constant commanded deployment/retraction acceleration |
| 4 | Exponential law: $d\ell/dt = alpha*\ell$ |
| 5 | Tension deployment |
| Alpha | Constant in the exponential law |

Idepl=5,Alpha=1.d-4

*************************************************************************

| | |
|---|---|
| | For tension deployment: |
| Fc | Absolute value of maximum tension at node next to shuttle (Nt) |
| Frk | Slope coefficient for \|tension (Nodes-1)\| <Fc |
| Frini | Initial tension at Nodes-1 (Nt) |

Fc=.02d0,Frk=1.d4,Frini=.02d0

*************************************************************************

| | |
|---|---|
| Tdepoc | Epoch at initialization(days) modified to Julian date: 1950, Jan1, 0hour=0.d0 |
| Tlnini | Undeformed initially deployed tether length (km) |
| Tlaini | Deformed initially deployed tether length (km) |
| Tlnin1 | Undeformed tether length (km) initially stored in the subsatellite |
| Tlnin2 | Undeformed tether length (km) initially stored in the shuttle |
| Dvnini | Initial deploy/retract velocity from shuttle (km/sec) |
| Dacini | Initial deploy/retract acceleration from shuttle (km/sec**2) |

Tdepoc=10400.d0,Tlnini=.1d-1,Tlaini=.1d-1
Tlnin1=0.d0,Tlnin2=19.990d0
Dvnini=.2d-3,Dacini=0.d0

*************************************************************************

| | |
|---|---|
| Rshutt | Cartesian position vector (km) of the shuttle referenced to the earth axis of 1950 |
| Vshutt | Cartesian velocity vector (km/sec) of the shuttle referenced to the earth axis of 1950 |
| Tdirec | Direction vector of the straight tether line referenced to the orbital reference frame with origin at shuttle. (automatically normalized) |
| Omeini | Angular velocity vector (1/sec) of the straight tether line referenced to the orbital reference frame |

```
Rshutt=6671.d0,0.d0,0.d0
Vshutt=0.d0,6.80d0,3.69d0
Tdirec=0.d0,0.d0,-1.d0
Omeini=0.d0,0.d0,0.d0
```

***************************************************************************

| | |
|---|---|
| Tedens | Linear mass density of the tether (kg/km) |
| Shmass | Mass of the shuttle (kg) without tether |
| Sumass | Mass of the subsatellite (kg) without tether |

```
Tedens=.3d0,Shmass=90000.d0,Sumass=180.d0
```

***************************************************************************

| | |
|---|---|
| Iteg | Flag to set integration routine |
| 1 | Variable step size routine |
| 2 | 4th order Runge-Kutta |

```
Iteg=1
```

***************************************************************************

| | |
|---|---|
| Nstop | Stopping condition by step control |
| <=0 | No stopping due to step control |
| > 0 | Stops after Nstop steps |
| Tstop | Stopping condition by time control |
| <=0 | No stopping due to time control |
| > 0 | Stops after Tstop seconds |
| Nprint | Printing condition by step control |
| <=0 | No printing due to step control |
| > 0 | Printing after Nprint steps |
| Tprint | Printing condition by time control |
| <=0 | No printing due to time control |
| > 0 | Printing after Tprint seconds |

```
Nstop=8000,Tstop=10000.d0
Nprint=20,Tprint=-100.d0
```

***************************************************************************

## 3.0 Tension Deployment

### 3.1 Theory

To implement tension controlled deployment as a new option in the tethered satellites simulation, it is first necessary to develop the theory and determine the program modifications necessary. It is most desirable that none of the current options of the program be disturbed and that simulation run speed should not be adversely affected. The original version of the simulation program used the assumption that the undeformed length of tether deployed at any time is known and is constrained to follow a user-defined function of time. As a consequence of this, the tether deployment acceleration couples extensively into the calculation of the time derivatives of the other dynamic variables. Since, the deployment acceleration is constrained, the tension at the deployer is determined and beyond the control of the user. To gain user control of the deployer tension, it is necessary that deployed, undeformed tether length be a dynamic variable. Thus, we must determine another dynamic equation for tether length. Before we do this let us review the dynamical equations of the simulation.

Figure 3.1.1 shows the basic coordinate systems used in the simulation. The equations of motion are solved in the inertial reference frame. The X axis of this reference frame is directed toward the first point of Aries (approximately the position of the sun as seen from the earth at the time of the Vernal Equinox). The Z axis is aligned with the earth spin axis and, thus, points to the celestial north pole. The Y axis is defined to complete a right handed coordinate system. Two other reference frames are used for output by the simulation. Subsatellite positions are defined relative to the shuttle in a local vertical reference frame defined as shown. This frame is oriented with the Z axis aligned with the local vertical position away from the earth. The Y axis points along the orbital angular velocity vector (normal to the orbit plane). The X axis completes the right handed set and points generally along the velocity vector. The output components for the tether nodal vectors are given with respect to a special local vertical reference defined for consistency with current tether analysis convention.

## Simulation Coordinate Systems

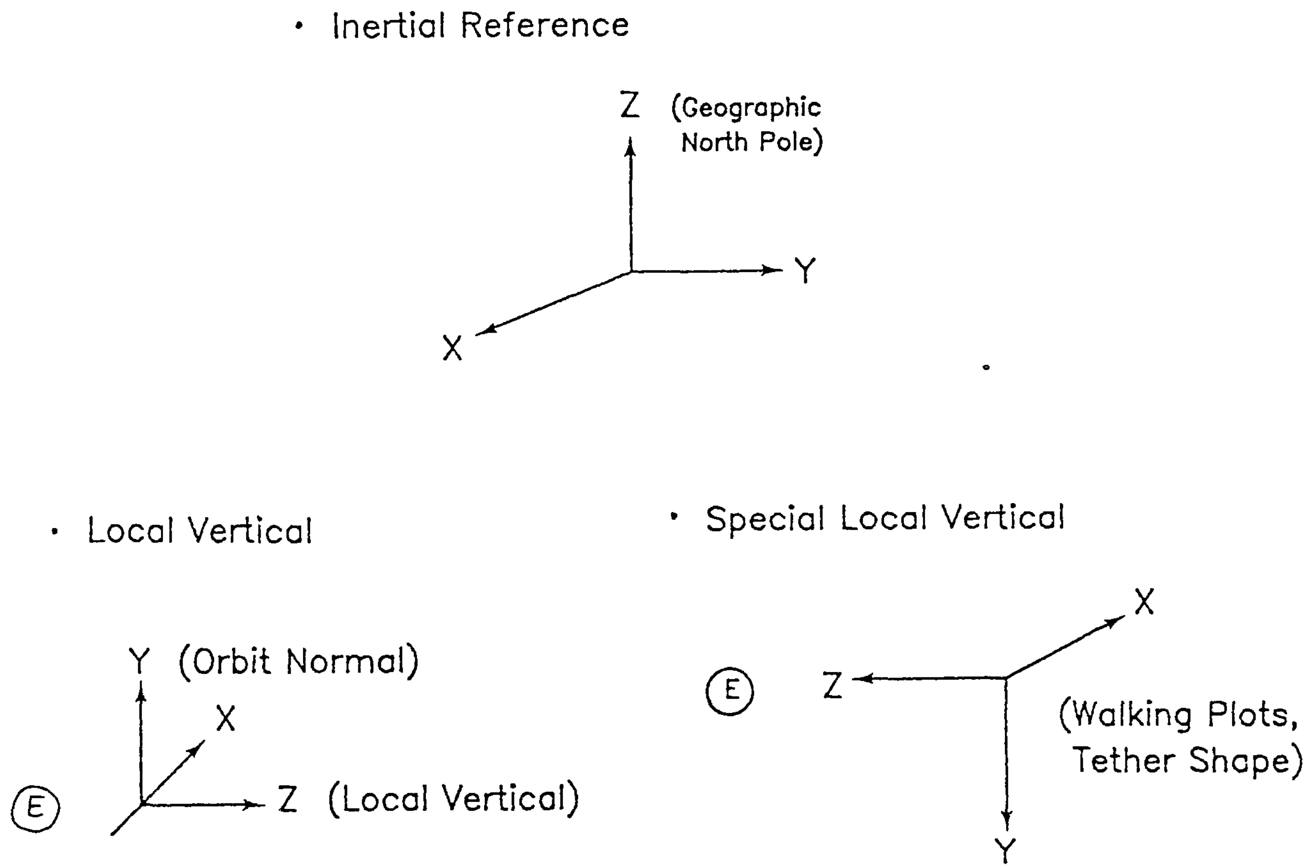
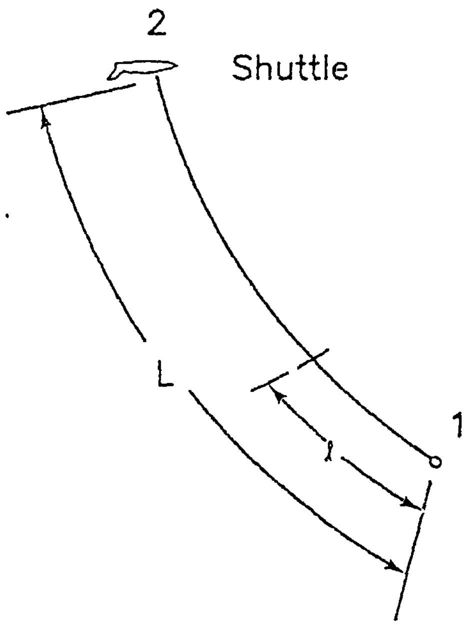



Figure 3.1.1.

These coordinate systems are the only ones of any consequence used by the simulation.

Figure 3.1.2 depicts the geometry of the shuttle-tether-subsatellite system. The shuttle and subsatellite are treated as point masses. The tether is treated as an elastic continuum of nonzero mass. The individual mass elements of the tether are specified by a length($\ell$) measured from the subsatellite. This parameter measures nominal distances along the tether and the value at any mass point corresponds to the unstretched arc length measured from the end. The deformed arc length along the tether will differ from ($\ell$) by the amount of stretch. Other parameters of interest are L, the total deployed (undeformed) length of tether and Lt, the total length of tether available for deployment.

Partial Differential Equation (PDE)
Tethered Satellites Dynamics
Model



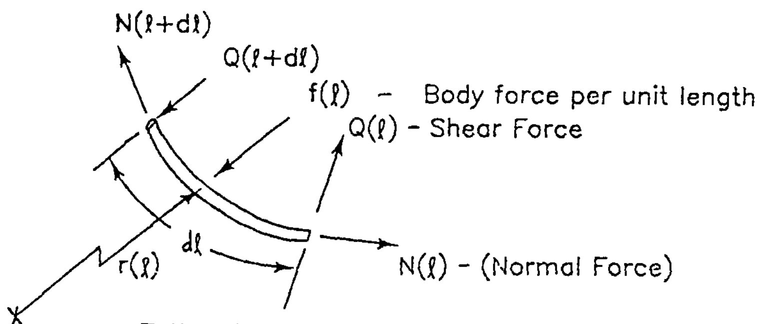

L – Deployed tether length

$L_T$ – Total tether length

ℓ – Mass point on tether measured from satellite

Figure 3.1.2.

Figure 3.1.3 shows a typical element of tether at some position (1) along the tether. The position vector defining the position of ($\ell$) is $r(\ell)$. The forces acting on this element are due to three sources. The first two are material forces due to tether elasticity: tension and shear. The third force is due to the external environmental forces acting on the element. These include earth gravity modelled by spherical and nonspherical earth terms, sun and moon gravity, aerodynamic drag and solar radiation pressure. The form of the tension and shear forces is shown in the figure. The other external forces are based on well-known models and are not discussed. They are explained in the original documentation of this simulation.[1]




Tension: $\underline{N}(\ell) = \beta\left[|\underline{r}'(\ell)| - 1\right]\dfrac{\underline{r}'}{|\underline{r}'|}$ ; $\underline{r}' = \dfrac{\partial \underline{r}}{\partial \ell}$ ; $|\underline{r}'| = \sqrt{\underline{r}'\cdot\underline{r}'}$

Shear: $\underline{Q}(\ell) = -\alpha\dfrac{1}{|\underline{r}'|}\left[\dfrac{1}{|\underline{r}'|}\left(\dfrac{\underline{r}'}{|\underline{r}'|}\right)'\right]' - \alpha\left|\dfrac{1}{|\underline{r}'|}\left(\dfrac{\underline{r}'}{|\underline{r}'|}\right)\right|^2\dfrac{\underline{r}'}{|\underline{r}'|}$

External Forces $f(\ell)$

1. Sperical Earth Gravity
2. Non spherical Earth perturbations
3. Sun & Moon Gravity
4. Aerodynamic Drag
5. Solar Radiation .pressure

Figure 3.1.3.

1. "Dynamics of a System of Two Satellites Connected by a Deployable and Extensible Tether of Finite Mass"; P. Kohler, W. Maag, and R. Wehrli, 1978.

Approximate solutions can be obtained by using appropriate numerical procedures. The partial differential equations appearing in Figure 3.1.4 are written in terms of two independent variables, time and undeformed length. The numerical integration process determines solutions at discrete future times. The spatial variation must also be discretized to obtain the solution numerically. One approach to discretization would be to track the motion of mass points located along the tether at values that are multiples of some fixed distance. The difficulty with this approach is that the number of such points (nodes) required varies with the length of tether deployed. This is inconvenient since the number of equations being integrated would change with the amount of tether deployed. To overcome this, we define a normalized length $(\xi_i) = \ell/L$ and transform the equations of motion to this variable. The new equations of motion are obtained by substitution of the results of the change of variables shown in Figure 3.1.5. If we now select nodal points to be located at the two ends and at points separated by 1/(N-1) for N nodes, we can write the equations of motion of the tether in terms of a constant number of nodes. This makes the job of numerical solution considerably simpler. On the other hand, many more terms are introduced into the equations of motion to account for the variation of the length L. This accounts for the fact that a variable amount of mass is associated with a tether node depending on the length L. Defining new variables T and V allows us to express the equations of motion in first order form. The partial differential equations are converted to difference equations according to the procedures shown in Figure 3.1.6. This process results in the equations summarized in Figure 3.1.7 with auxilliary variables as defined on the bottom of Figure 3.1.7 and Figure 3.1.8.

The set of equations summarized in Figures 3.1.7 and 3.1.8 represents the original form of the simulation as acquired by Control Dynamics under license from ACM/SAI. No material damping was included in this model. Test results suggest that energy dissipation in tether materials is significant. Thus, to model the material damping that is shown to be present, we added viscous damping to the tether tension model. This model is shown in Figure 3.1.9.

A more difficult modification to the simulation was the addition of the capability to simulate tension controlled deployment. The simulation as originally formulated treated the deployed length L as a known function of time specified by the user. From this known function and the dynamic equations for the system, the tension within each segment of the tether is determined. To make the tension controllable it is necessary to make the tether length a dynamic variable. This can be accomplished in several ways. The way chosen minimizes the required changes to the simulation and maintains all existing capabilities. The chosen technique is shown in Figure 3.1.10.

The dynamic equations for the typical tether element are derived in Figure 3.1.4 based on the vector diagram shown in Figure 3.1.3. The motion of the end-masses is defined by the boundary conditions. These boundary conditions are essentially just the dynamic equations for point masses. Body 2 (the shuttle) includes an extra term which looks like a thrusting term due to the flow effect of the deploying tether and results from the same sort of considerations used to derive the dynamical equations of a rocket and is referred to as the rocket term for this reason. If bending stiffness of the tether is nonzero, additional boundary conditions must be satisfied. These are shown at the bottom of Figure 3.1.4.

Dynamics Equations of Motion
Boundary Conditions

$$\mu\, d\ell\, \ddot{\underline{r}}(\ell) = \underline{N}(\ell+d\ell) - \underline{N}(\ell) + \underline{Q}(\ell+d\ell) - \underline{Q}(\ell) + \underline{f}\, d\ell$$

$$= d\ell \frac{\partial}{\partial \ell}\left[\underline{N}(\ell) + \underline{Q}(\ell)\right] + \underline{f}\, d\ell$$

$$\Rightarrow \qquad \mu \ddot{\underline{r}}(\ell) = \frac{\partial}{\partial \ell}\left[N(\ell) + Q(\ell)\right] + \underline{f} \quad : \quad \text{Interior tether point}$$

Boundary Conditions:

$$\left.\begin{aligned} m_1 \ddot{\underline{r}}_1 &= \underline{F}_1 + \left[\underline{N}(\ell) + \underline{Q}(\ell)\right]\Big|_{\ell=0} \\ m_2 \ddot{\underline{r}}_2 &= \underline{F}_2 + \left[\mu \dot{L}^2 \underline{r}'(\ell) - \underline{N}(\ell) - \underline{Q}(\ell)\right]\Big|_{\ell=L} \end{aligned}\right\} \quad \text{End body dynamics}$$

for $\alpha \neq 0$

$$\left.\begin{aligned} \left(\underline{r}'' - \frac{\underline{r}'}{|\underline{r}'|}\frac{\partial}{\partial \ell}|\underline{r}'|\right)\Big|_{\ell=0} &= 0 \\ \left(\underline{r}'' - \frac{\underline{r}'}{|\underline{r}'|}\frac{\partial}{\partial \ell}|\underline{r}'|\right)\Big|_{\ell=L} &= 0 \end{aligned}\right\} \quad \text{Bending moment must vanish at tether ends}$$

Figure 3.1.4.

# DISCRETIZATION

Figure 3.1.5

Define normalized length $\xi = \frac{\ell}{L}$

Change to tether parameter from $\ell$.

$$\underline{r}\,(\ell,t) = \hat{\underline{r}}\,(\xi,t)$$

$$\underline{r}' = \frac{\partial \underline{r}}{\partial \ell} \;;\; \hat{\underline{r}}' = \frac{\partial \hat{\underline{r}}}{\partial \xi}$$

$$\dot{\underline{r}} = \left(\frac{\partial r}{\partial t}\right)_{\ell\ \mathrm{const}} = \left(\frac{\partial \hat{r}}{\partial t}\right)_{\xi\ \mathrm{const}} + \dot{\xi}\ \frac{\partial \hat{r}}{\partial \xi}_{\ t\ \mathrm{const}}$$

$$\dot{\xi} = -\xi \frac{\dot{L}}{L}$$

$$\ddot{\underline{r}} = \ddot{\hat{\underline{r}}} + 2\dot{\hat{r}}'\dot{\xi} + \hat{\underline{r}}''\dot{\xi}^2 + \hat{\underline{r}}'\ddot{\xi}$$

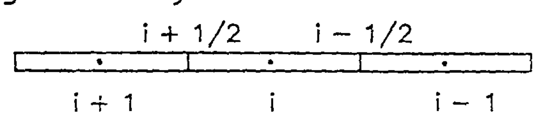
$$\ddot{\xi} = -\xi\frac{\ddot{L}}{L} + 2\xi\left(\frac{\dot{L}}{L}\right)^2$$

Let $\underline{T} = \hat{r}'$

$\underline{V} = \dot{\hat{\underline{r}}}$

Figure 3.1.6

To discretize the spatial equations assume n uniformly distributed nodes including boundary nodes.




$$\ell_i = \frac{i-1}{n-1}\,L \implies \Delta\ell = \frac{L}{n-1}$$

$$\xi_i = \frac{i-1}{n-1} \implies \Delta\xi = \frac{1}{n-1}$$

$$\underline{T}_{i+1/2} = \frac{\hat{\underline{r}}_{i+1} - \hat{\underline{r}}_i}{\Delta\xi} = (n-1)(\hat{\underline{r}}_{i+1} - \hat{\underline{r}}_i)$$

$$\underline{T}_i = 1/2(\underline{T}_{i+1/2} + \underline{T}_{i-1/2})$$

$$\underline{W}_{i+1/2} = \underline{V}_{i+1} - \underline{V}_i$$

# SUMMARY OF DESCRETIZED WORKING EQUATIONS

Figure 3.1.7

$$\frac{d\underline{T}_{i+\frac{1}{2}}}{dt} = (n-1)\,\underline{w}_{i+\frac{1}{2}}\ ;\quad i = 1,\cdots,n-1$$

$$\frac{d\hat{\underline{r}}_n}{dt} = \underline{v}_n$$

$$\frac{d\underline{W}_{1+\frac{1}{2}}}{dt} = a_1\,\underline{N}_{1+\frac{1}{2}} + a_2\,\underline{N}_{2+\frac{1}{2}} + \underline{I}_{1+\frac{1}{2}}$$

$$\frac{d\underline{W}_{i+\frac{1}{2}}}{dt} = b_1\,\underline{N}_{i-\frac{1}{2}} + b_2\,\underline{N}_{i+\frac{1}{2}} + b_3\,\underline{N}_{i+\frac{3}{2}} + \underline{I}_{i+\frac{1}{2}}\ ;\quad i=2,\cdots,n-2$$

$$\frac{d\underline{W}_{n-\frac{1}{2}}}{dt} = c_1\,\underline{N}_{n-\frac{3}{2}} + c_2\,\underline{N}_{n-\frac{1}{2}} + \underline{I}_{n-\frac{1}{2}}$$

$$\frac{d\underline{v}_n}{dt} = \frac{1}{2m_2}\left(\underline{N}_{n-\frac{3}{2}} - 3\underline{N}_{n-\frac{1}{2}}\right) + \underline{I}_n$$

---

$$a_1 = -\left(\frac{3}{2m_1} + \frac{n-1}{\mu L}\right)\ ;\quad b_1 = \frac{n-1}{\mu L}\ ;\quad c_1 = \frac{1}{2m_2} + \frac{n-1}{\mu L}$$

$$a_2 = \frac{1}{2m_1} + \frac{n-1}{\mu L}\ ;\quad b_2 = -2b_1\ ;\quad c_2 = -\left(\frac{3}{2m_2} + \frac{n-1}{\mu L}\right)$$

$$b_3 = b_1$$

Figure 3.1.8

$$\underline{A}_i = -(n-1)\dot{\xi}_i\left(\underline{w}_{i+\frac{1}{2}} + \underline{w}_{i-\frac{1}{2}}\right) - \frac{1}{2}\ddot{\xi}_i\left(\underline{T}_{i+\frac{1}{2}} + \underline{T}_{i-\frac{1}{2}}\right) - (n-1)\dot{\xi}^2\left(\underline{T}_{i+\frac{1}{2}} - \underline{T}_{i-\frac{1}{2}}\right)\ ;\quad i=2,\cdots,n-1$$

$$\dot{\xi}_i = -\xi_i\frac{\dot{L}}{L}\ ;\quad \ddot{\xi}_i = -\xi_i\frac{\ddot{L}}{L} + 2\xi_i\left(\frac{\dot{L}}{L}\right)^2\ ;\quad i = 1,\cdots,n$$

$$\underline{I}_{1+\frac{1}{2}} = \underline{A}_2 + \frac{1}{\mu}\underline{f}_2 - \frac{1}{m_1}\underline{F}_1 + a_1\underline{Q}_1 + a_2\underline{Q}_{2+\frac{1}{2}}$$

$$\underline{I}_{i+\frac{1}{2}} = \underline{A}_{i+1} - \underline{A}_i + \frac{1}{\mu}\left(\underline{f}_{i+1} - \underline{f}_i\right) + b_1\underline{Q}_{i-\frac{1}{2}} + b_2\underline{Q}_{i+\frac{1}{2}} + b_3\underline{Q}_{i+\frac{3}{2}}\ ;\quad i=2,\cdots,n-2$$

$$\underline{I}_{n-\frac{1}{2}} = -\underline{A}_{n-1} + \frac{1}{m_2}\underline{F}_2 - \frac{1}{\mu}\underline{f}_{n-1} + c_1\underline{Q}_{n-\frac{3}{2}} + c_2\underline{Q}_{n-\frac{1}{2}} - \frac{\mu}{2m_2}\frac{\dot{L}^2}{L}\left(\underline{T}_{n-\frac{3}{2}} - 3\underline{T}_{n-\frac{1}{2}}\right)$$

$$\underline{I}_n = \frac{1}{m_2}\underline{F}_2 + c_1\underline{Q}_{n-\frac{3}{2}} + c_2\underline{Q}_{n-\frac{1}{2}} - \frac{\mu}{2m_2}\frac{\dot{L}^2}{L}\left(\underline{T}_{n-\frac{3}{2}} - 3\underline{T}_{n-\frac{1}{2}}\right)$$

$\underline{f}_i$, $\underline{F}_1$ & $\underline{F}_2$ are determined from external force models which depend on $\hat{\underline{r}}_i$, $\underline{r}_1$, $\underline{r}_n$ and derivatives. These are in turn determined from $\underline{r}_n$, $\underline{T}_i$, $\underline{v}_n$, $\underline{w}_i$.

## VISCOUS TETHER DAMPING

$$N = \beta(|\underline{r}'| - 1) + \gamma \frac{\underline{r}' \cdot \dot{\underline{r}}'}{|\underline{r}'|}$$

$$= \beta\left(\frac{|\hat{\underline{r}}'|}{L} - 1\right) + \gamma \frac{\hat{\underline{r}}'}{L^2 |\hat{\underline{r}}'|} (L\dot{\hat{\underline{r}}}' - \dot{L}\hat{\underline{r}}' - \dot{L}\xi\hat{\underline{r}}'')$$

$$\underline{N} = N \frac{\hat{\underline{r}}'}{|\hat{\underline{r}}'|}$$

$$\dot{L} = \frac{\beta |\hat{r}'_{n-1/2}| \left(L|\hat{\underline{r}}'_{n-1/2}| - L^2\right) + L\hat{\underline{r}}'_{n-1/2} \cdot \dot{\hat{\underline{r}}}'_{n-1/2} - T_c L^2 |\underline{r}_{n-1/2}|}{\gamma \left(\hat{\underline{r}}'_{n-1/2} \cdot \hat{\underline{r}}'_{n-1/2} + \xi_{n-1/2}\, \hat{\underline{r}}'_{n-1/2} \cdot \hat{\underline{r}}''_{n-1/2}\right)}$$

Figure 3.1.9

# TENSION CONTROLLED DEPLOYMENT

- Original Sim. Assumed $L, \dot{L}, \ddot{L}$ Constrained

$$\begin{aligned} \dot{\underline{y}} &= \underline{f}_0(\underline{y}) + \ddot{L}\,\underline{f}_1(\underline{y}) \\ L &= L(t) \\ \dot{L} &= \dot{L}(t) \\ \ddot{L} &= \ddot{L}(t) \end{aligned}$$

- Modified Sim.

$$\left.\begin{aligned} \dot{\underline{y}} &= \underline{f}_0(\underline{y}) + \ddot{L}\,\underline{f}_1(\underline{y}) \\ \ddot{L} &= \ddot{L}_0(\underline{y}) + \underline{\ddot{L}}_1^T(\underline{y})\,\dot{\underline{y}} \end{aligned}\right\}\ \text{Fully Extensible, Homogeneous}$$

$$\left.\begin{aligned} &\dot{\underline{y}} = \underline{f}_0(\underline{y}) + \ddot{L}\,\underline{f}_1(\underline{y}) \\ &N_{n-1/2} = N_{n-1/2,0} + \ddot{L}\,N_{n-1/2,1} \Rightarrow \ddot{L} = \frac{F - N_{n-1/2,0}}{N_{n-1/2,1}} \end{aligned}\right\}\ \text{Inextensible}$$

Figure 3.1.10

The details of the development of modifications to the equations of motion of tethered satellites simulation have been discussed in the previous paragraphs. These modifications have been implemented and verified.

## 3.2 Implementation into Simulation

The following additions and changes were made to the program for the constant tension deployment law.

The vectors Y, Z, and F were expanded to allow for the integration of tether deployment acceleration and velocity and the friction force derivative. This change affects common UMESH and was changed in every subroutine in which these terms appear. This increased the dimensions of these variables to 123.

```
        COMMON/DEPLAW/ALPHA,IDEPL,FC,FRK,FR     (added FC,FRK,FR)
```

This common appears in the Main, Subroutine Deplex, Subroutine Rbegin, Subroutine Rinitl, Subroutine Rwrite, Subroutine Tendpl, Subroutine Uatox, Subroutine Ubegin, Subroutine Uder, Subroutine Ugen, Subroutine Uinitl, Subroutine Uprint, and Subroutine Uxtoa.

Subroutine Deplex

```
        GO TO(10,20,30,40,50)IDEPL
 50     CONTINUE
```

Subroutine Kinmax(F0,F1)

```
        IMPLICIT REAL*8(A-H,O-Z)
        COMMON/SHUTCO/RSHUTT(3),VSHUTT(3),RSHU2,RSHU
        COMMON/RELCO/TAU(19,3),RELVEL(19,3)
        COMMON/TDEPPA/TLN,DVN,DAC,XMASS2
        COMMON/TINDPA/DUMMY1(4),XINCRE,DUMMY2(2),NODES
        COMMON/TESPEC/TEDENS,BETA,GAMMA,LONELA
        DIMENSION F(60),F0(60),F1(60)
C
        NMIN1 = NODES-1
        NMIN2 = NODES-2
        N3=3*NODES
        X2=DVN/TLN
        X3=-XINCRE*X2*X2
        DAC=0.D0
1       XM=0.5D0*XINCRE*DAC/TLN
        X1=XM+X3
        XP=X1+X3
C       CONTRIBUTION TO THE DERIVATIVES OF RELVEL(1,I)
        DO 20 I=1,3
        F(I)=XM*TAU(1,I) + XP*TAU(2,I) + X2*(RELVEL(1,I)+RELVEL(2,I))
20      CONTINUE
```

```
C       CONTRIBUTION TO THE DERIVATIVES OF RELVEL(2,I),...,RELVEL(NODES-2,I)
        DO 30 K=2,NMIN2
        FLOAT1=DBLE(1-K)
        FLOAT2=DBLE(K)
        XX1=FLOAT1*(X1+FLOAT1*X3)
        XX2=XM+2.DO*FLOAT1*FLOAT2*X3
        XX3=FLOAT2*(X1+FLOAT2*X3)
        DO 40 I=1,3
        J=3*(K-1)+1
        F(J)=XX1*TAU(K-1,I)+XX2*TAU(K,I)+XX3*TAU(K+1,I)
     $      + X2*(FLOAT1*RELVEL(K-1,I)+RELVEL(K,I)+FLOAT2*RELVEL(K+1,I))
40      CONTINUE
30      CONTINUE
C       CONTRIBUTION TO THE DERIVATIVES OF RELVEL(NODES-1,I)
        FLOAT1=-DBLE(FLOAT(NMIN2))
        XX1=FLOAT1*(X1+FLOAT1*X3)
        XX2=FLOAT1*(X1-FLOAT1*X3)
        DO 50 I=1,3
        J=3*NMIN2+I
        F(J)=XX1*TAU(NMIN2,I)+XX2*TAU(NMIN1,I)+FLAOT1*X2*
     $      (RELVEL(NMIN2,I)+RELVEL(NMIN1,I))
50      CONTINUE
C       ROCKET TERM.  CONTRIBUTES TO THE DERIVATIVES OF RELVEL(NODES-1,I) AND
C       VSHUTT(I)
        XX1=0.5DO*TEDENS*DVN*X2/XMASS2
        DO 60 I=1,3
        XX2 = XX2*(3.DO*TAU(NMIN1,I)-TAU(NMIN2,I))
        J1 = 3*NMIN2+I
        J2 = J1+3
        F(J1) = F(J1)+XX2
        F(J2) = XX2
60      CONTINUE
        IF(DAC .LT. .5DO) THEN
        DO 70 I=1,3*NODES
        FO(I)=F(I)
70      CONTINUE
        DAC=1.ODO
        GOTO 1
        ENDIF
        DO 80 I=1,3*NODES
        F1(I) = F1(I) - FO(I)
80      CONTINUE
        RETURN
        END
```

```
Subroutine Rbegin
        READ(19,*,END=1) INSTEP,TDEPOC,TLNINI,DVNINI,DACDUM,TLNIN1,
     $  TLNIN2,RSHUTT(1),RSHUTT(2),RSHUTT(3),VSHUTT(1),VSHUTT(2),
     $  VSHUTT(3),FRINI
        GOTO(3,4,6,5,7),IDEPL
3       CONTINUE
4       DACINI=0.D0
        GOTO 6
5       CONTINUE
        DACINI=ALPHA*DVNINI
        GO TO 6
7       TLN=TLNINI
        DVN=DVNINI
        DAC=DACDUM
        FR=FRINI
6       CONTINUE
        WRITE(16,*)'FC:',FC,'FRK:',FRK

Subroutine Rinitl
        READ(5,*)FC,FRK,FRINI
        PRINT *,'FC,FRK,FRINI:',FC,FRK,FRINI
        FC = FC/1000.D0
        FRINI = FRINI/1000.D0
        FRK = FRK/1000.D0

Subroutine Rwrite
        WRITE(19,*) NSTEP, TDNOW, TLN, DVN, DAC, TLN1, TLN2,
     $   RSHUTT(1), RSHUTT(2), RSHUTT(3), VSHUTT(1), VSHUTT(2),
     $   VSHUTT(3), FR

Subroutine Tendpl(T,XLODD,XL1DD)
C       THIS SUBROUTINE COMPUTES TIME DEPENDENT PARTS FOR A
C       TENSION DEPLOYMENT LAW.  THE PARAMETERS TO BE COMPUTED ARE
C       DVN:  CONSTRAINED DEPLOYMENT RATE BASED ON TENSION
C       XLODD:  TETHER ACCELERATION EXCLUDING TERMS IN Z
C       XL1DD(1-3):  COEFFICIENTS OF RPDD AT NODE N
C
        IMPLICIT REAL*8(A-H,O-Z)
        DIMENSION RP(3),RPP(3),RPD(3),RPPD(3),XL1DD(3)
        COMMON/TINDPA/TINI,TLNINI,DVNINI,DACINI,XINCRE,XMASS0,XMASS1,
     $  NODES
        COMMON/TDEPPA/TLN,DVN,DAC,XMASS2
        COMMON/DEPLAW/ALPHA,IDEPL,FC,FRK,FR
        COMMON/RETRAP/TLNIN1,TLNIN2,TLN1,TLN2
        COMMON/EMASS/ SUMASS,SHMASS
        COMMON/NFORCE/XNFORC(19)
        COMMON/INTERM/TAUQUA(19),TAUMAG(19),BLABLA(37)
        COMMON/UMESH/TT,DTT,Y(123),F(123)
        COMMON/SHUTCO/RSHUTT(3),VSHUTT(3),RSHU2,RSHU
        COMMON/RELCO/TAU(19,3),RELVEL(19,3)
```

```
C
      NMIN1 = NODES - 1
      NMIN2 = NMIN1 - 1
      NMIN3 = NMIN2 - 1
      XNMIN1 = DBLE(NMIN1)
C     COMPUTE AUXJLLIARY VARIABLES
      XI = 1.DO - .5DO*XINCRE
C     COMPUTE MORE AUXILLIARY VARIABLES
      DO 10 I=1,3
      RP(I) = TAU(NMIN1,I)
      RPD(I) = RELVEL(NMIN1,I)/XINCRE
      RPP(I) = (1.5DO*TAU(NMIN1,I) - 2.0DO*TAU(NMIN2,I) +
     $         .50DO*TAU(NMIN3,I))/XINCRE
10    RPPD(I) = (1.5DO*RELVEL(NMIN2,I) - 2.0DO*RELVEL(NMIN2,I) +
     $         .5DO*RELVEL(NMIN3(I))/XINCRE/XINCRE
      RPMSQ = RP(1)**2 + RP(2)**2 + RP(3)**2
      RPM = DSQRT(RPMSQ)
C     COMPUTE LDOT NUMERATOR TERMS
      XN1 = TLN*RPMSQ
      XN2 = RPM*TLN*TLN
      XN3 = TLN*(RP(1)*RPD(1) + RP(2)*RPD(2) + RP(3)*RPD(3))
      XN4 = FR*XN2
C     COMPUTE LDOT DENOMINATOR TERMS
      XD1 = XI*(RP(1)*RPP(1) + RP(2)*RPP(2) + RP(3)*RPP(3))
      XD2 = RPMSQ
      XNUM = (BETA*(XN1-XN2) + GAMMA*XN3 - XN4)
      XDENOM = GAMMA*(XD1 + XD2)
C     COMPUTE LDOT (DVN) FROM FR
      DVN = XNUM/XDENOM
      IF (DABS(FR) .LT. FC) THEN
      FRDOT = FRK*DVN*TAUMAG(NMIN1)/TLN
      ELSE
      FRDOT = 0.DO
      ENDIF
C     DERIVATIVES OF NUMERATOR AND DENOMINATOR TERMS
      XN1D = XN1*DVN/TLN + 2.DO*XN3
      XN2D = 2.DO*DVN*XN2/TLN + TLN**3*XN3/XN2
      XN3D = DVN*XN3/TLN
      XD1D = 0.DO
      XD2D = 2.DO*XN3/TLN
      DO 20 I=1,3
      XN3D = XN3D + TLN*RPD(I)*RPD(I)
      XD1D = XD1D + XI*RPP(I)*RPD(I) + XI*RP(I)*RPPD(I)
      XL1DD(I) = TLN*RP(I)/(XD1 + XD2)
20    CONTINUE
      XN4D = FR*XN2D + FRDOT*XN2
      XNUMD = BETA*(XN1D-XN2D) + GAMMA*XN3D - XN4D
      XDEND = GAMMA*(XD1D + XD2D)
      XLODD = (XNUMD-XNUM*XDEND/XDENOM)/XDENOM
      RETURN
      END
```

```
Subroutine Uatox
        IF(IDEPL .EQ. 5) THEN
           FR = Y(6*NODES+1)
           TLN = Y(6*NODES+2)
           IF(LONELA .EQ. 1) DVN = Y(6*NODES+3)
           IF(DABS(FR) .GT. FC) FR = SIGN(FC,FR)
        ENDIF

Subroutine Ubegin
        GOTO(3,4,6,5,7)IDEPL
3       DVNINI=0.DO
4       DACINI=0.DO
        GOTO 6
5       DVNINI=ALPHA*TLNINI
        DACINI=ALPHA*DVNINI
        GO TO 6
7       TLN = TLNINI
        DVN = DVNINI
        DAC = DACINI
6       CONTINUE
        WRITE(16,*)'FC:',FC,'FRK:',FRK

Subroutine Uder
        COMMON/DEPLAW/ALPHA,IDEPL,FC,FRK,FR
        COMMON/TDEPPA/TLN,DVN,DAC,XMASS2
        COMMON/RETRAP/TLNIN1,TLNIN2,TLN1,TLN2
        COMMON/INTERM/TAUQUA(19),TAUMAG(19),TIIP1(18),RELVQU(19)
        COMMON/NFORCE/XNFORC(19)
        DIMENSION XNO(19),XN1(19)
        NMIN2 = NODES - 2
        NMIN3 = NODES - 3
        CALL HELP
        IF(IDEPL .EQ. 5 .AND. LONELA .GT. 1) THEN
        CALL TENDPL(T,XLODD,XL1DD)
        ENDIF
C       COMPUTE NORMAL FORCE FOR LDD (DAC) = 0
        DAC = 0.DO
        DO 260 J=1,N3
        JP=J+N3
        Z(JP)=Z(J)+FO(J)
        FP(J)=Z(P)
260     CONTINUE
        CALL NORMAL(Z,F)
        DO 265 J=1,N3
        JP=J+N3
        Z(JP)=Z(JP)+F(J)
        FO(J)=Z(JP)
265     CONTINUE
        DO 270 I=1,NMIN1
        XNO(I)=XNFORC(I)
270     CONTINUE
```

```
C       COMPUTE NORMAL FORCE FOR LDD (DAC) = 1
        DAC=1.D0
        DO 280 J=1,N3
        JP=J+N3
        Z(JP)=FP(J)+F1(J)
        FP(J)=Z(JP)
280     CONTINUE
        CALL NORMAL(Z,F)
        DO 285 J=1,N3
        JP=J+N3
        Z(JP)=Z(JP)+F(J)
        F1(J)=Z(JP)
285     CONTINUE
        DO 290 I=1,NMIN1
        XN1(I)=XNFORC(I)
290     CONTINUE
        IF(LONELA .EQ. 1) THEN
        DAC=(FR/TAUMAG(NMIN1)-XNO(NMIN1))/(XN1(NMIN1)-XNO(NMIN1))
        ELSE
        DAC=XLODD
        XDEN=1.D0
        DO 1005 I=1,3
        XDEN=XDEN-XL1DD(I)*(F1(N3-6+I)-F0(N3-6+I))/XINCRE
        DAC=DAC+XL1DD(I)*F0(N3-6+I)/XINCRE
1005    CONTINUE
        DAC=DAC/XDEN
        ENDIF
        DO 295 J=1,N3
        JP=J+N3
        Z(JP)=F0(J)+DAC*(F1(J)-F0(J))
295     CONTINUE
        DO 300 I=1,NMIN1
        XNFORC(I)=XNO(I)+DAC*(XN1(I)-XNO(I))
300     CONTINUE
C       INTEGRATION VARIABLES FOR TENSION DEPLOYMENT
        Z(6*NODES+1) = FRK*DVN*TAUMAG(NMIN1)/TLN
        Z(6*NODES+2) = DVN
        IF(LONELA .EQ. 1) Z(6*NODES+3) = DAC
        ELSE
C       COMPUTE KINEMATICAL TERMS
        CALL KINEMA(F)
        DO 40 J=1,N3
        JP=J+N3
        Z(JP)=Z(JP)+F(J)
40      CONTINUE
C       THE NORMAL FORCE
        CALL NORMAL(Z,F)
        DO 310 J=1,N3
        JP=J+N3
        Z(JP)=Z(JP)+F(J)
310     CONTINUE
        ENDIF
```

Subroutine Ugen

```
        IF(IDEPL .EQ. 5) THEN
          NDEQ = 6*NODES + 2
          IF(LONELA .EQ. 1) NDEQ = 6*NODES + 3
        ENDIF
```

Subroutine Uinitl

```
        READ(5,*)FC,FRK,FRINI
        PRINT *,'FC,FRK,FRINI:',FC,FRK,FRINI
        FC = FC/1000.D0
        FRINI = FRINI/1000.D0
        FRK = FRK/1000.D0
        IF(IDEPL .EQ. 5) FR = FRINI
```

Subroutine Uxtoa

```
        IF(IDEPL .EQ. 5) THEN
           Y(6*NODES+1) = FR
           Y(6*NODES+2) = TLN
           Y(6*NODES+3) = DVN
        ENDIF
```

## 4.0 Damping at Deployment Point

### 4.1 Theory

While working on the tension deployment scheme, it became apparent that some type of damping would be necessary to reduce the effects of the high frequency modes travelling through the tether. The high frequencies led to extremely small time steps and long computer run times. The damping keeps the basic tether motions while reducing high frequency effects.

The damping terms were incorporated in Subroutine Fulext for nodes 1 through n-1. Subroutine Homoge for node n-1 (current coding then applies damping to the remaining nodes). As no stretch is obtained in the inextensible tether, it was not necessary to add damping.

To obtain the damping equation, it was first necessary to generate a Rayleigh Dissipation Function, F. This term is similar in form to the potential energy equation with $\beta$ replaced with $\gamma$ and derivatives of $\underline{r}'$ used. This yields the equation

$$F = \frac{1}{2}\gamma\left(\dot{\underline{r}}' \cdot \frac{\frac{\partial r}{\partial \zeta}}{\left|\frac{\partial r}{\partial \zeta}\right|}\right)^2$$

Take the partial with respect to $\underline{r}'$ to obtain the damping terms to incorporate in the force equations.

$$\frac{\partial F}{\partial \dot{\underline{r}}'} = \frac{\gamma}{L}\left[\frac{\xi \dot{L}}{L}\,\underline{\tau}'\cdot\underline{\tau} + \dot{\underline{\tau}}\cdot\underline{\tau} - |\underline{\tau}|^2\frac{\dot{L}}{L}\right]\frac{\underline{\tau}}{|\underline{\tau}|^2}$$

In these equations:

$L \rightarrow$ deployed tether length

$\dot{L} \rightarrow$ tether deployment velocity

$\gamma \rightarrow$ damping coefficient

$\xi \rightarrow$ floating tether point

$\underline{\tau} \rightarrow$ position of floating tether point.

4.2 Implementation into Simulation

The following additions and changes were made to the program for the tether damping.

```
COMMON/TESPEC/TEDENS,BETA,GAMMA,LONELA         (GAMMA added in)
```

This change occured in the MAIN, Subroutine Deplex, Subroutine Fulext, Subroutine Head, Subroutine Help, Subroutine Homoge, Subroutine Kinema, Subroutine Kinmax, Subroutine Ngrav, Subroutine Normal, Subroutine Rbegin, Subroutine Rinitl, Subroutine Tendpl, Subroutine Uatox, Subroutine Ubegin, Subroutine Uder, Subroutine Ugen, Subroutine Uinitl, and Subroutine Uprint.

Subroutine Tendpl has previously been entered in the tension deployment section of the report, the damping terms were included.

Subroutine Fulext

```
      COMMON/RELCO/TAU(19,3),RELVEL(19,3)
      NMIN2 = NODES-2
      NMIN3 = NODES-3
      DO 10 K=1,NMIN1
      X=BETA*(X1-1.DO/TAUMAG(K))
      XD = RELVEL(K,1)*TAU(K,1) + RELVEL(K,2)*TAU(K,2) + RELVEL(K,3)*
     $     TAU(K,3)
      IF(K .LT. NMIN1) THEN
         IF(K .EQ. 1) THEN
              TAUSQP = .25DO*(-3.DO*TAUQUA(K) + 4.DO*
     $                 TAUQUA(K+1)-TAUQUA(K+2))/XINCRE
         ELSE
              TAUSQP = .25DO*(TAUQUA(K+1) - TAUQUA(K-1))
     $                 /XINCRE
         ENDIF
      ELSE
              TAUSQP = .25DO*(3.DO*TAUQUA(NMIN1) - 4.DO*
     $                 TAUQUA(NMIN2) + TAUQUA(NMIN3))/XINCRE
      ENDIF
      XD = XD/XINCRE - DVN/TLN*(TAUQUA(K) + DBLE(FLOAT(K)) - .5DO)*
     $     TAUSQP/DBLE(FLOAT(NMIN1)))
      XD = GAMMA*XD/TLN/TAUQUA(K)
      X = X + XD
      IF(ISLACK .EQ. 2 .AND. X .LT. 0.DO) X=0.DO
      XNFORC(K) = X
10    CONTINUE
```

Subroutine Homoge

```
      COMMON/RELCO/TAU(19,3),RELVEL(19,3)
      NMIN3 = NODES-3
CC    ADD DAMPING TO SECTION NEXT TO BODY 2
C
      XD = RELVEL(NMIN1,1)*TAU(NMIN1,1) + RELVEL(NMIN1,2)*
     $     TAU(NMIN1,2) + RELVEL(NMIN1,3)*TAU(NMIN1,3)
      TAUSQP = .25DO*(3.DO*TAUQUA(NMIN1) - 4.DO*TAUQUA(NMIN2) +
```

```
     $          TAUQUA(NMIN3))/XINCRE
        XD = XD/XINCRE - DVN/TLN*(TAUQUA(NMIN1) + (1.D0 -
     $       .5D0/DBLE(NMIN1))*TAUSQP)
        XD = GAMMA*XD/TLN/TAUQUA(NMIN1)
```

Subroutine Rbegin

```
        READ(19,*) LONNEW,TEDENS,BSTIFF,BETA,GAMMA,NODES,SHMASS,SUMASS
        WRITE(16,*) 'LONELA:',LONELA,'BETA:',BETA,'GAMMA',GAMMA,
     $              'ISLACK:',ISLACK
        WRITE(19,*)LONELA,TEDENS,BSTIFF,BETA,GAMMA,NODES,SHMASS,SUMASS
```

Subroutine Rinitl

```
        READ(5,*)LONELA,BETA,GAMMA,ISLACK
        PRINT *,LONELA,BETA,GAMMA,ISLACK
        GAMMA = GAMMA/1000.D0
        READ(19,*)LONNEW,TEDENS,BSTIFF,BETA,GAMMA,NODES,SHMASS,SUMASS
```

Subroutine Ubegin

```
        WRITE(16,*) 'LONELA:',LONELA,'BETA:',BETA,'GAMMA:',GAMMA,
     $              'ISLACK:',ISLACK
        WRITE(19,*)LONELA,TEDENS,BSTIFF,BETA,GAMMA,NODES,SHMASS,SUMASS
```

Subroutine Uinitl

```
        READ(5,*)LONELA,BETA,GAMMA,ISLACK
        PRINT *,'LONELA,BETA,GAMMA,ISLACK:',LONELA,BETA,GAMMA,ISLACK
        GAMMA = GAMMA/1000.D0
```

## 5.0 Tektronics Graphics

With the simulation running on the VAX at MSFC, it became necessary to implement some type of graphics compatible with the Tektronics terminals. A basic scheme was first implemented so that the most fundamental types of graphs could be viewed. Throughout the project this initial code was revised and updated to its present form.(See Appendix-D) The graphics now include such features as a simple menu which returns to the screen after each plot, the ability to set limits on each axis for a plot, the ability to enter labels for the plots, and the ability to plot 'walking-plots'. The 'walking plots' are an attempt to emulate the plots yielded by the Smithsonian simulation. They depict the inplane/out-of-plane motion of the tether as it deploys. The 'walking-plots' include additional features which allow the user to look at every nth point or scale the data as desired. This last option helps the user to better see the deflections in the tether as it deploys.

## 5.1 Examples

The following plots and graphs are examples of the graphics capabilities currently in use at MSFC on their VAX in coordination with the tether simulation. When running the plotting program, the first thing the user sees is a very basic menu (Figure 5.2.1) which lists the variables available and asks what type of plot the user desires. The user can specify either a normal x vs y type plot with up to four y variables, the 'walking-plots', a plot of tension at all the tether nodes (disregarding the shuttle as a node), or no plot at all.

If the user requests the x vs. y type plot, the next question asks how many plots per graph, up to four, are desired. If a number not one through four is input, the program returns to the basic menu. The user inputs the plot per graph number. The program then asks for the x-axis number associated with the x variable desired, and then the corresponding y-axis numbers. Eight character labels for the x and y variables are requested next (Figure 5.2.2), and finally the program gives the current plot limits and asks if any change is desired. If so, then the user inputs the new plot limits. For one plot per graph, new limits are not requested. The plot then appears on the screen, is sent to the thermal printer, disappears, and the menu is returned. Any of the variables listed in the basic menu may be plotted in this manner. (Figure 5.2.3, tether length in km vs. time in sec.; Figure 5.2.4 in-plane and out-of-plane angles in degrees vs. time in sec.; Figure 5.2.5 tether deployment velocity in km/sec vs. time in sec.; Figure 5.2.6 tether deployment acceleration in km/sec**2 vs. time in sec.; Figure 5.2.7 the distance between the shuttle and the subsattelite in km vs. time in sec.; and Figure 5.2.8 the friction force applied at the deployment point in km vs. time in sec.)

For the 'walking-plots' the user needs to know the number of nodes used in the simulation run as this is the first information requested when this type of plot is chosen. The next question asks if the user wants to view a plot of every time point, if not then a number n is requested to look at every nth data set. The first time through the 'walking-plot' section, the code calculates the maximum in-plane and out-of-plane deflections in the tether and where these

```
RUN SPLT2
VARIABLE LIST:
   1     TIME
   2INPL ANG
   3OUTP ANG
   4SHUT-SUB
   5SUBSAT X
   6SUBSAT Y
   7SUBSAT Z
   8 MAX TEN
   9 MIN TEN
  10 SH ANOM
  11SH-EARTH
  12 T-ACCEL
  13   T-VEL
  14   T-LEN
  15      FR
  16 TENSION
  17 TENSION
  18 TENSION
  19 TENSION
  20 TENSION
WHAT TYPE OF PLOT?
1-REGULAR, 2-WALKING, 3-TENSION, 4-NONE
```



Figure 5.2.1

```
VARIABLE LIST:
   1     TIME
   2INPL ANG
   3OUTP ANG
   4SHUT-SUB
   5SUBSAT X
   6SUBSAT Y
   7SUBSAT Z
   8 MAX TEN
   9 MIN TEN
  10 SH ANOM
  11SH-EARTH
  12 T-ACCEL
  13    T-VEL
  14    T-LEN
  15       FR
  16 TENSION
  17 TENSION
  18 TENSION
  19 TENSION
  20 TENSION
WHAT TYPE OF PLOT?
1-REGULAR, 2-WALKING, 3-TENSION, 4-NONE
1
INPUT NUMBER OF PLOTS/GRAPH: 1-4
1
INPUT NUMBER FOR X VARIABLE
1
INPUT NUMBER FOR VARIABLE Y(           1)
14
ONE PLOT/GRAPH
ENTER X-LABEL (8)
TIME
ENTER Y-LABEL (8)
LENGTH
```

ORIGINAL PAGE IS
OF POOR QUALITY

Figure 5.2.2

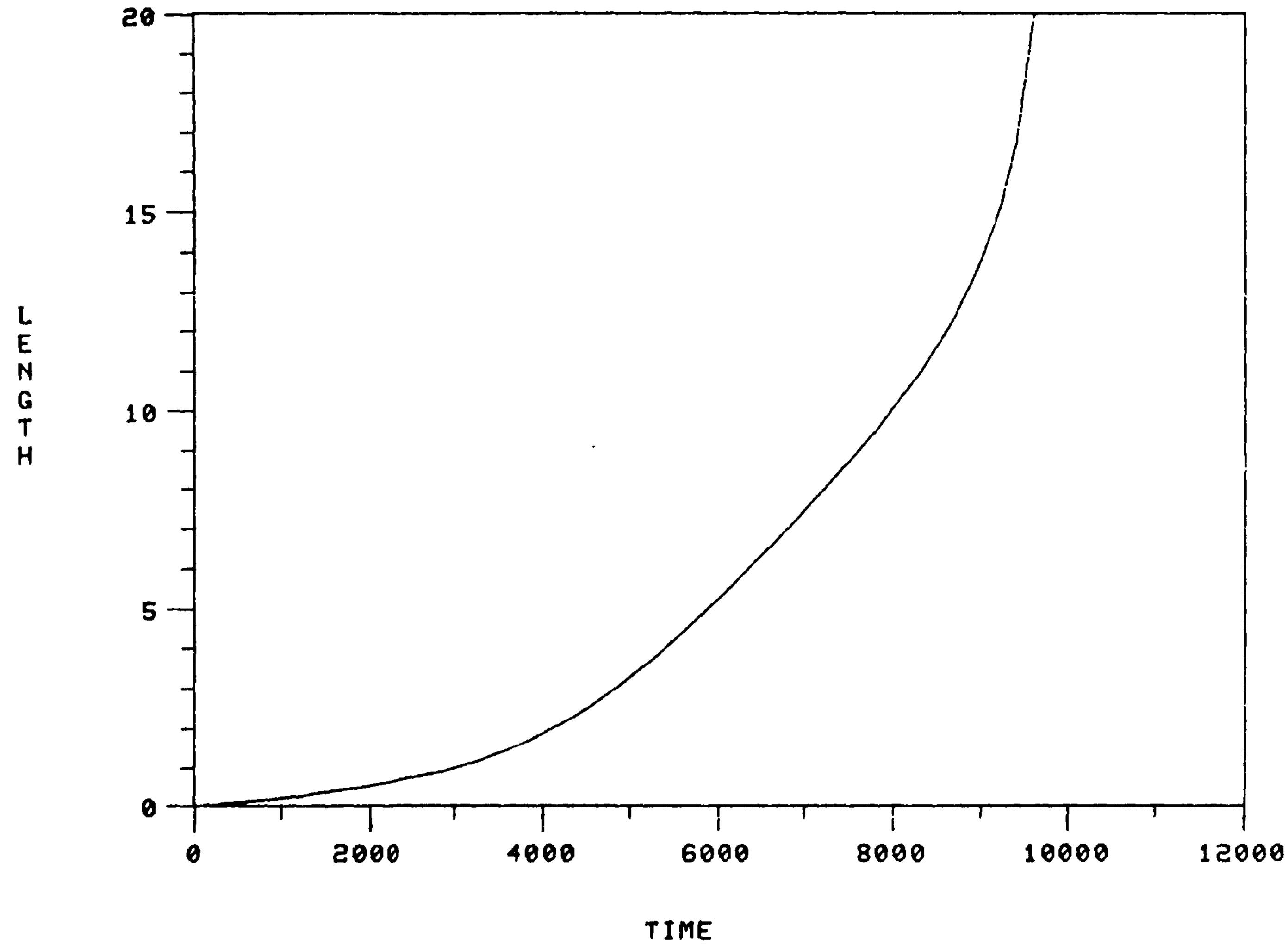
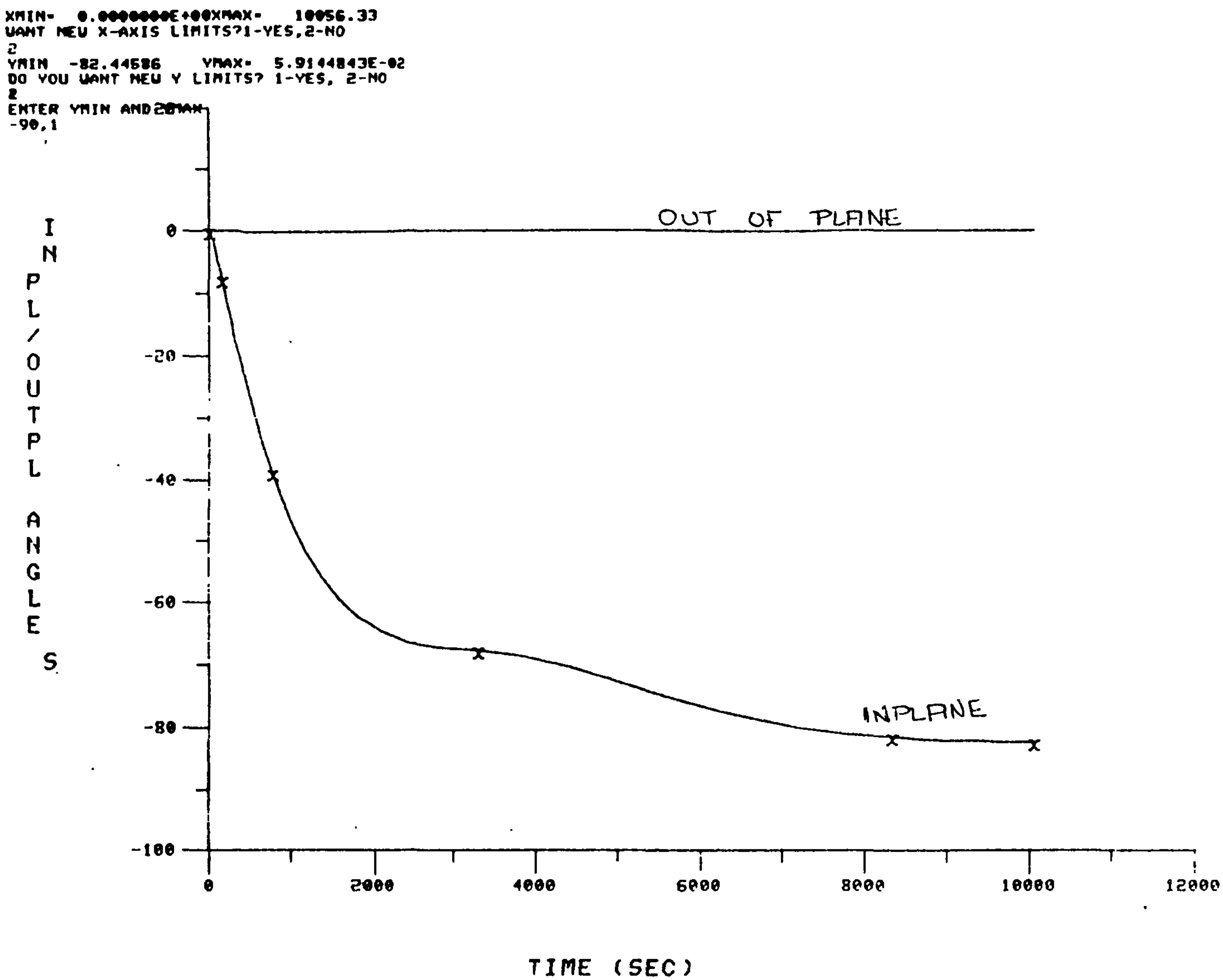



Figure 5.2.3



XMIN= 0.0000000E+00XMAX= 10056.33
WANT NEW X-AXIS LIMITS?1-YES,2-NO
2
YMIN -82.44686 YMAX= 5.9144843E-02
DO YOU WANT NEW Y LIMITS? 1-YES, 2-NO
2
ENTER YMIN AND YMAX
-90,1
IN PL/OUTPL ANGLES
0
-20
-40
-60
-80
-100
0
2000
4000
6000
8000
10000
12000
OUT OF PLANE
INPLANE
TIME (SEC)


Figure 5.2.4

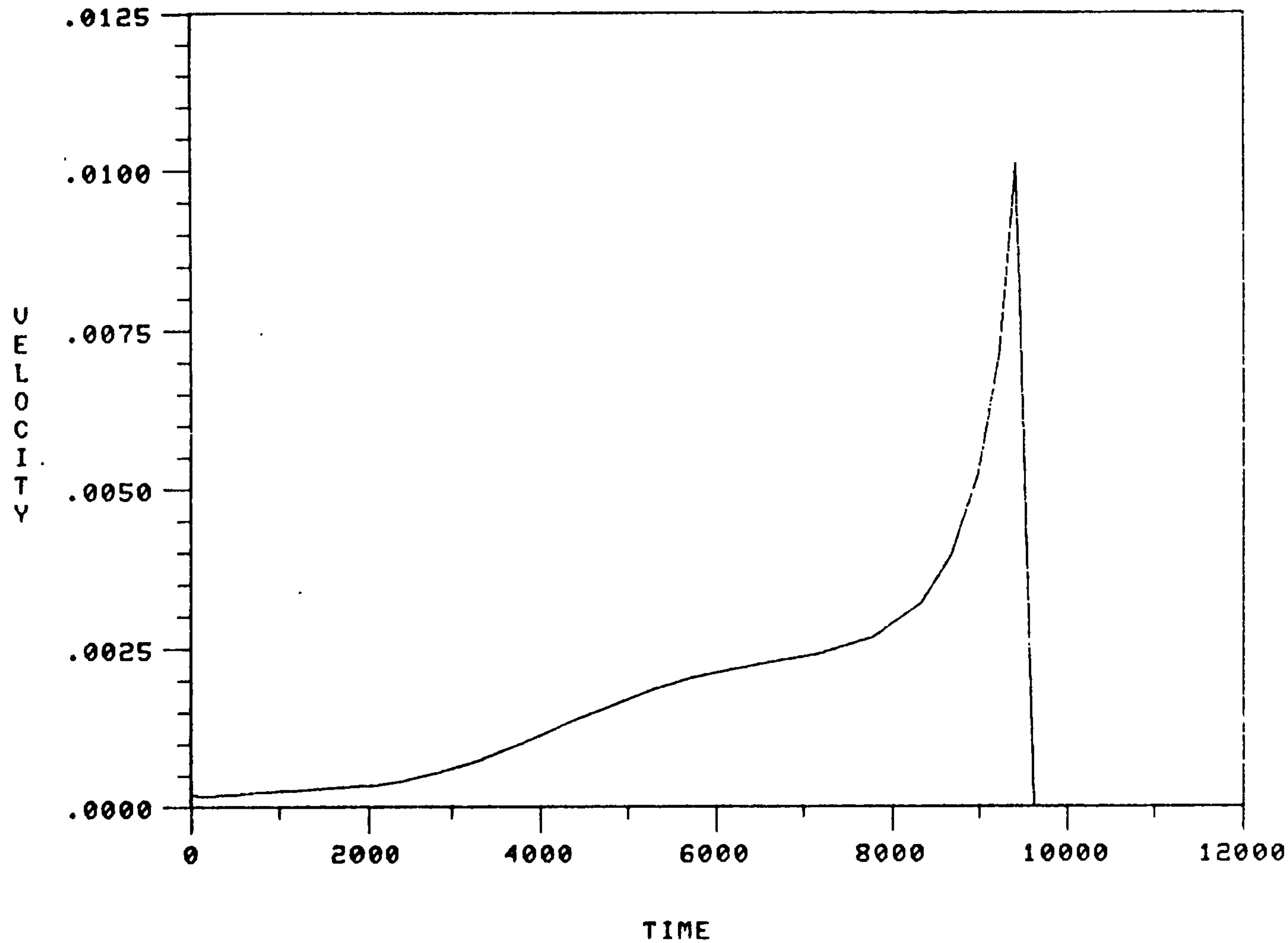

.0125
.0100
.0075
.0050
.0025
.0000
VELOCITY
0
2000
4000
6000
8000
10000
12000
TIME

Figure 5.2.5

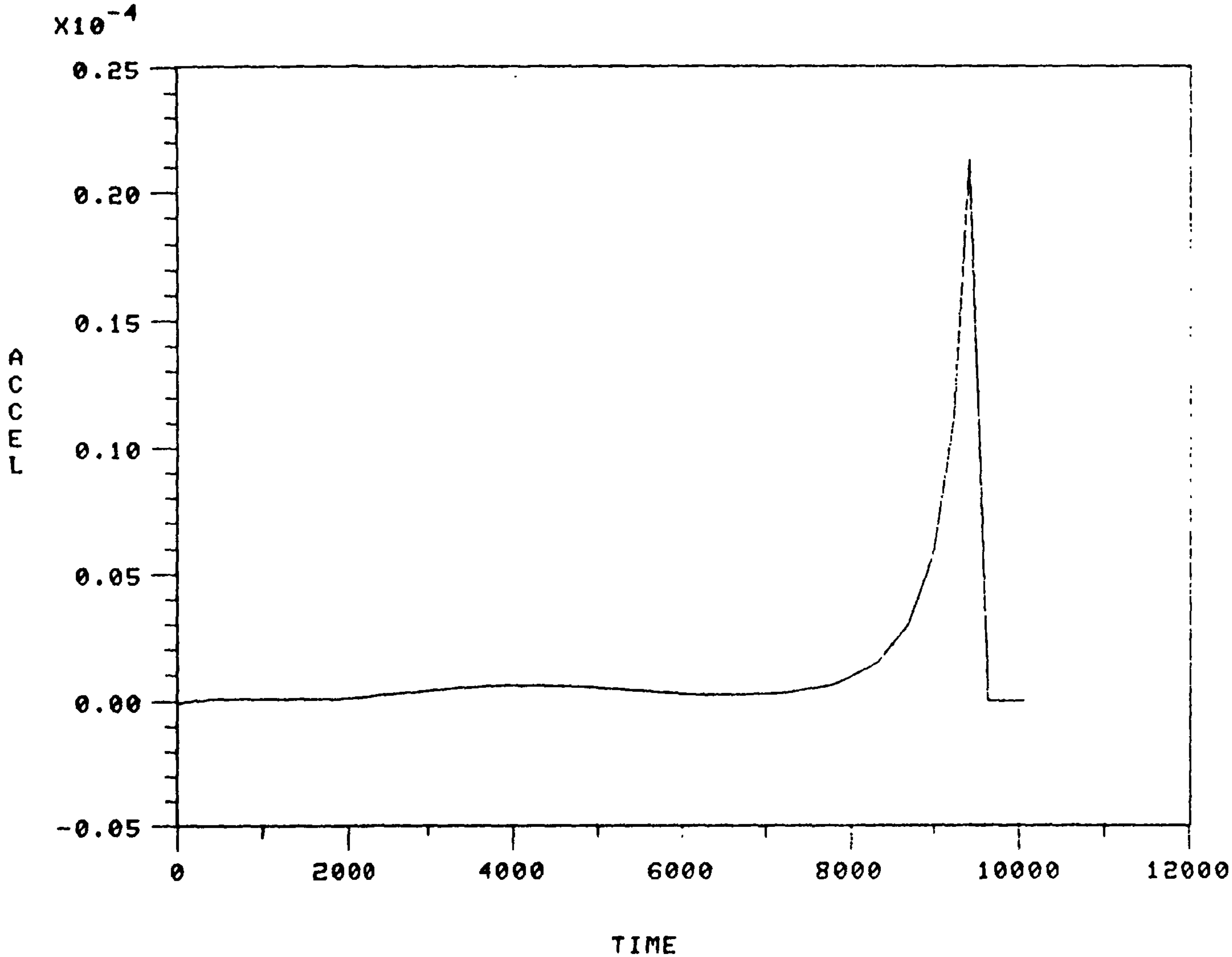

X10^-4
0.25
0.20
0.15
0.10
0.05
0.00
-0.05
ACCEL
0
2000
4000
6000
8000
10000
12000
TIME

Figure 5.2.6

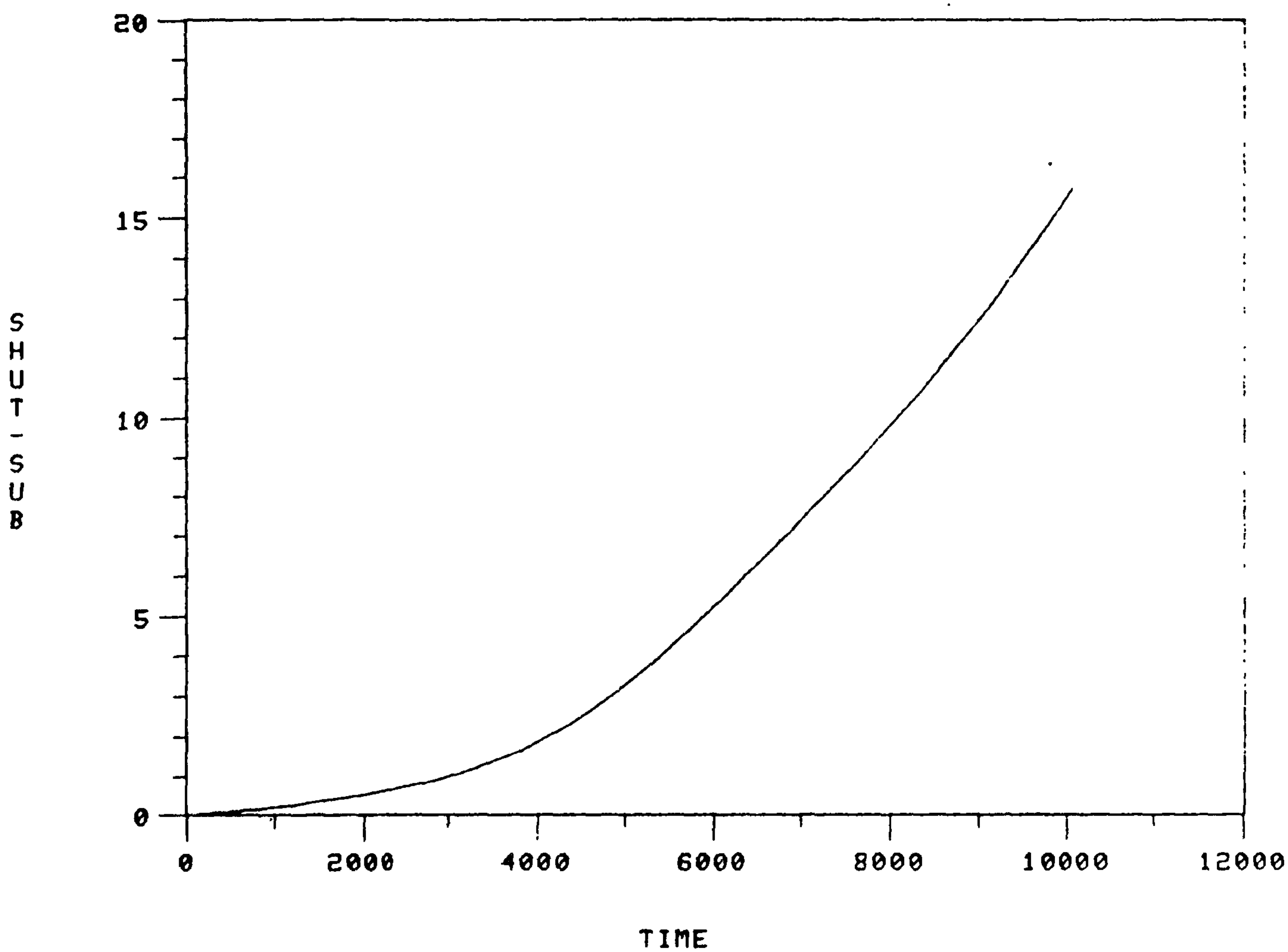



Figure 5.2.7

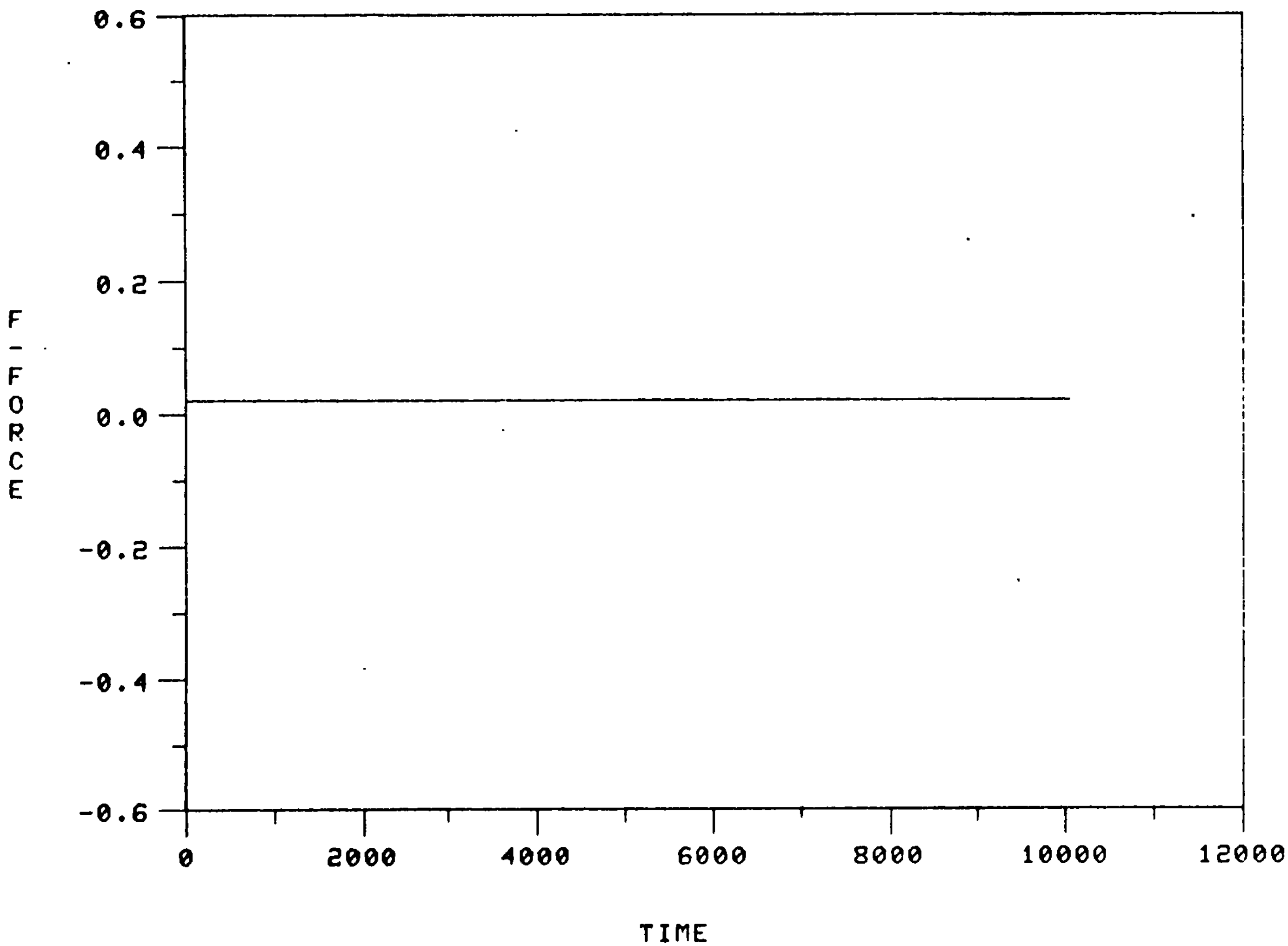

0.6
0.4
0.2
0.0
-0.2
-0.4
-0.6
F-FORCE
0
2000
4000
6000
8000
10000
12000
TIME

Figure 5.2.8

deflections occur. The user can then enter scale factors based on these deflections. The user then specifies an in-plane or an out-of-plane graph. As before, the data is plotted to the screen, sent to the printer, disappears, and the main menu returns. Figure 5.2.9 gives the questions asked when a walking plot is desired. Figure 5.2.10 is the plot associated with the above input, this is the basic plot with the scale factors at magnitude one and every time point plotted. Figure 5.2.11 shows this plot with every third time point plotted. Figure 5.2.12 has all the time points plotted, but the scale factor is .075 versus the previous value of 1.00.

When a tension plot is requested, the program first asks for the base number of variables; i.e. the number of variables above the tension variables in the menu. In the example base menu this number would be 15. The code then plots the remaining tension variables on one graph. When finished, this process also returns to the main menu. Figure 5.2.13 shows the tension for nodes 1 through the node next to the shuttle. These plots were obtained using the sample input file previously given. The markedly different behavior exhibited at the end of the run is accounted for in that the tether became fully deployed at approximately 9100 seconds. At this point the tether switched into the constant length mode for the duration of the run.

```
VARIABLE LIST:
    1     TIME
    2INPL ANG
    3OUTP ANG
    4SHUT-SUB
    5SUBSAT X
    6SUBSAT Y
    7SUBSAT Z
    8 MAX TEN
    9 MIN TEN
   10 SH ANOM
   11SH-EARTH
   12 T-ACCEL
   13   T-VEL
   14   T-LEN
   15      FR
   16 TENSION
   17 TENSION
   18 TENSION
   19 TENSION
   20 TENSION
WHAT TYPE OF PLOT?
1-REGULAR, 2-WALKING, 3-TENSION, 4-NONE
3
ENTER NUMBER OF TETHER NODES
6
DO YOU WANT EVERY TIME POINT? 1-YES,2-NO
1
MAX IN PLANE DEFLECTION =     15.59082
AT THE          51TIME STEP; NODE #          6
MAX OUT OF PLANE DEFLECTION =  2.9296875E-03
AT THE          42TIME STEP; NODE #          4
ENTER XSCAL AND YSCAL
1,1
VARIABLES:
IN-PLANE, OUT-OF-PLANE, RADIAL
ONE PLOT/GRAPH
INPUT X AND Y VARIABLE NUMBER
1,3
```



Figure 5.2.9

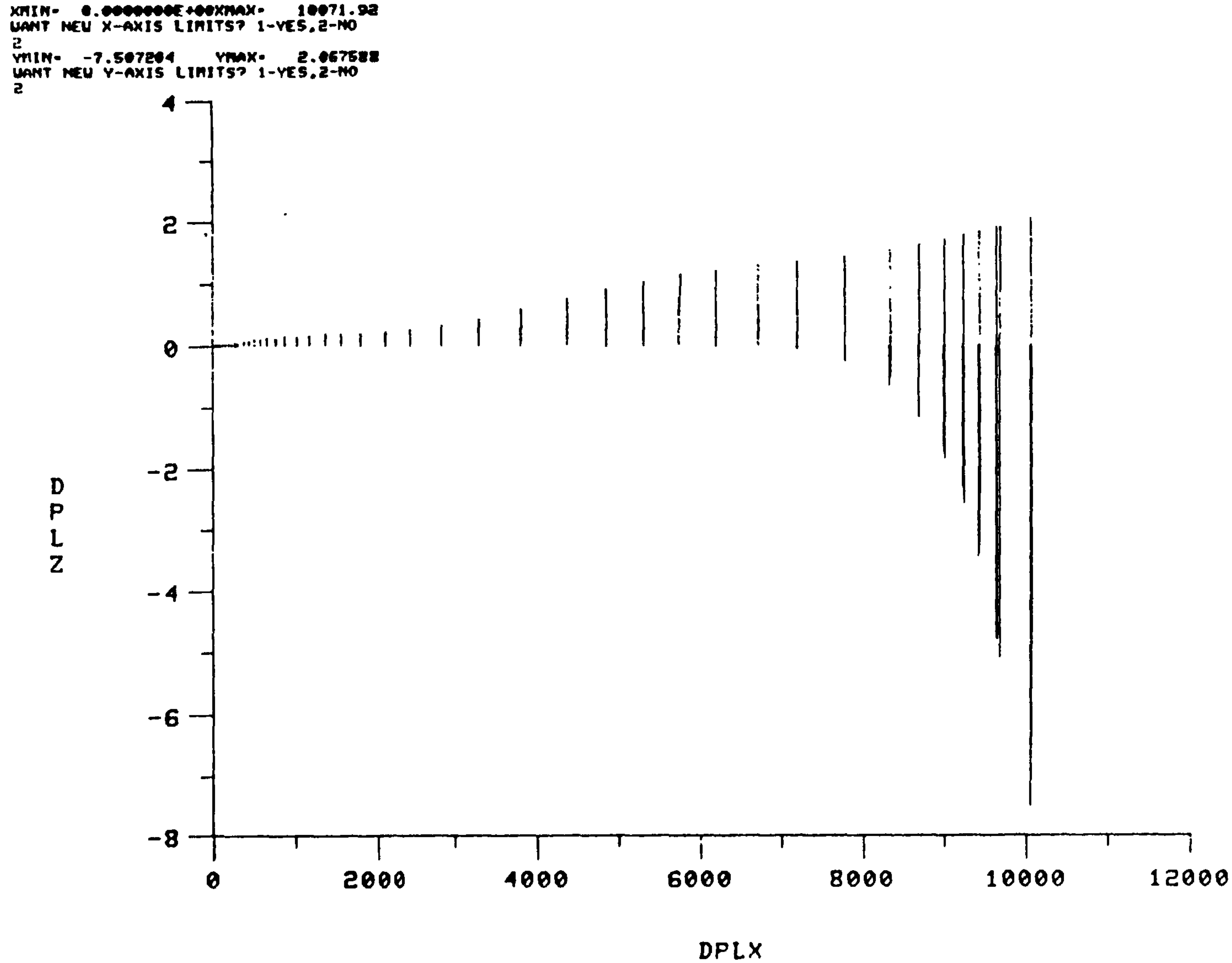

XMIN= 0.00000000E+00XMAX= 10071.92
WANT NEW X-AXIS LIMITS? 1-YES,2-NO
2
YMIN= -7.507204 YMAX= 2.067588
WANT NEW Y-AXIS LIMITS? 1-YES,2-NO
2
4
2
0
-2
-4
-6
-8
0
2000
4000
6000
8000
10000
12000
DPLZ
DPLX

Figure 5.2.10

XMIN= 0.0000000E+00XMAX= 10071.92
WANT NEW X-AXIS LIMITS? 1-YES,2-NO
2
YMIN= -7.507204 YMAX= 2.067588
WANT NEW Y-AXIS LIMITS? 1-YES,2-NO
2

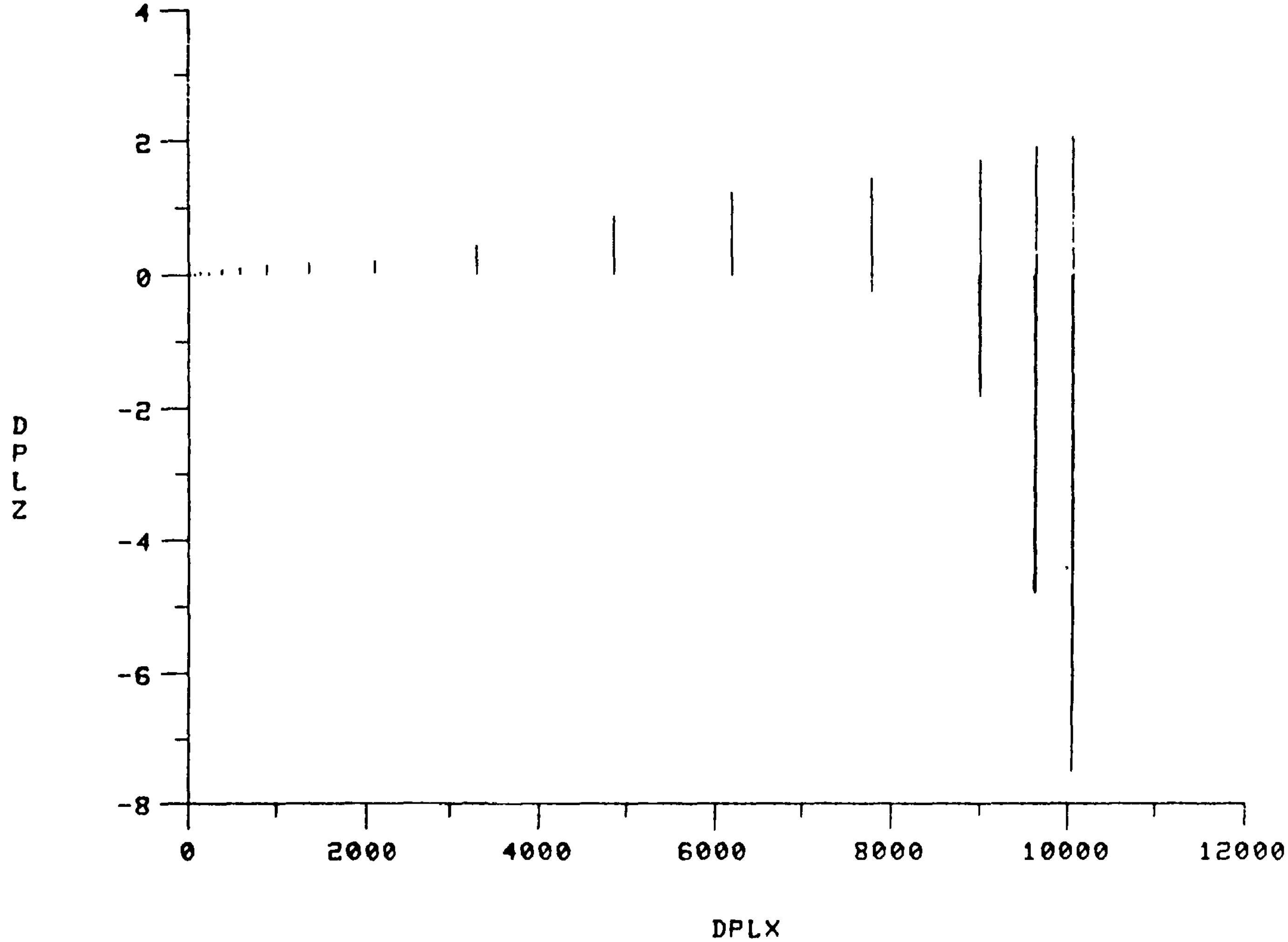



Figure 5.2.11

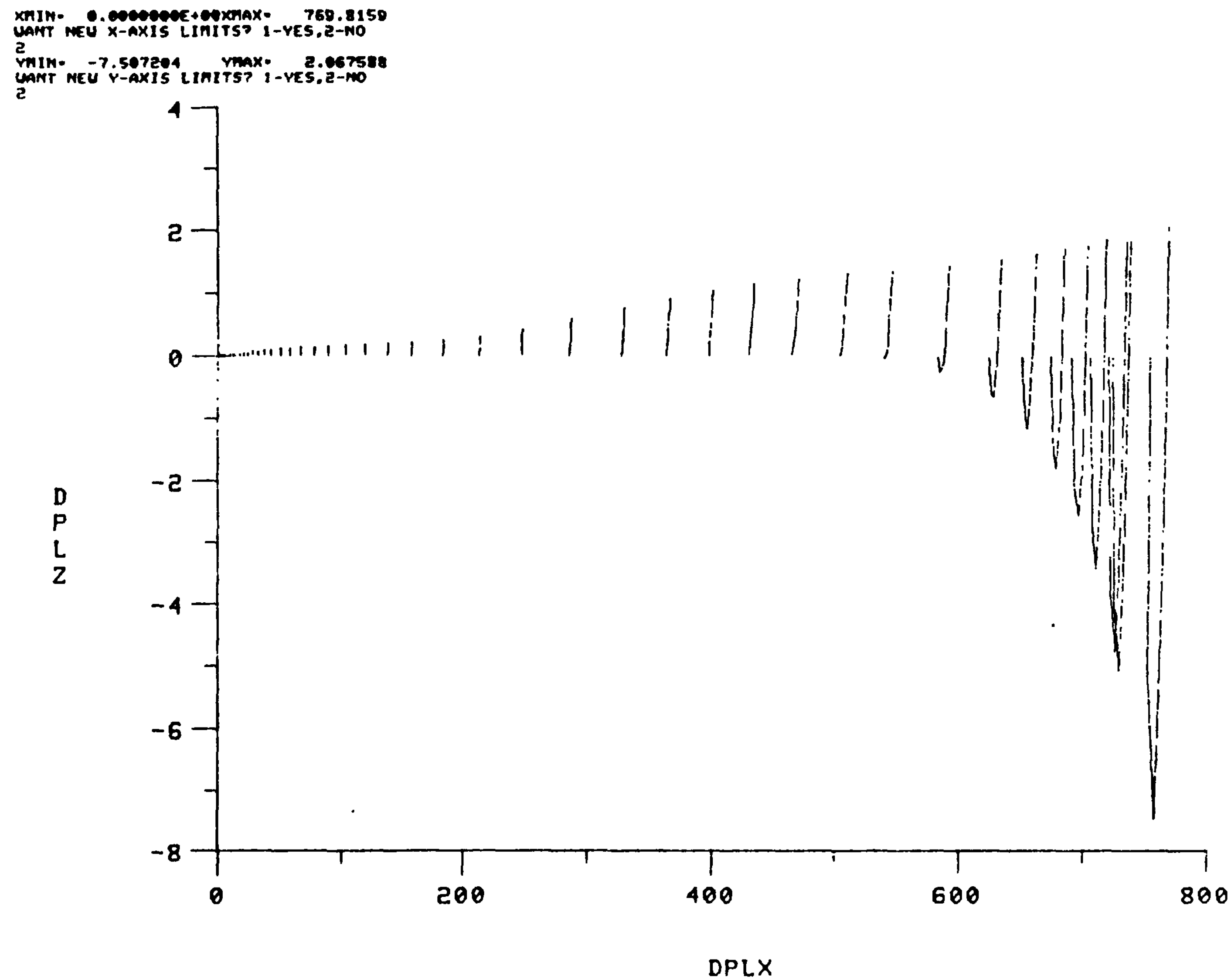

XMIN= 0.0000000E+00XMAX= 769.8159
WANT NEW X-AXIS LIMITS? 1-YES,2-NO
2
YMIN= -7.507204 YMAX= 2.067588
WANT NEW Y-AXIS LIMITS? 1-YES,2-NO
2
4
2
0
-2
-4
-6
-8
DPLZ
0
200
400
600
800
DPLX

Figure 5.2.12

ORIGINAL PAGE IS
OF POOR QUALITY

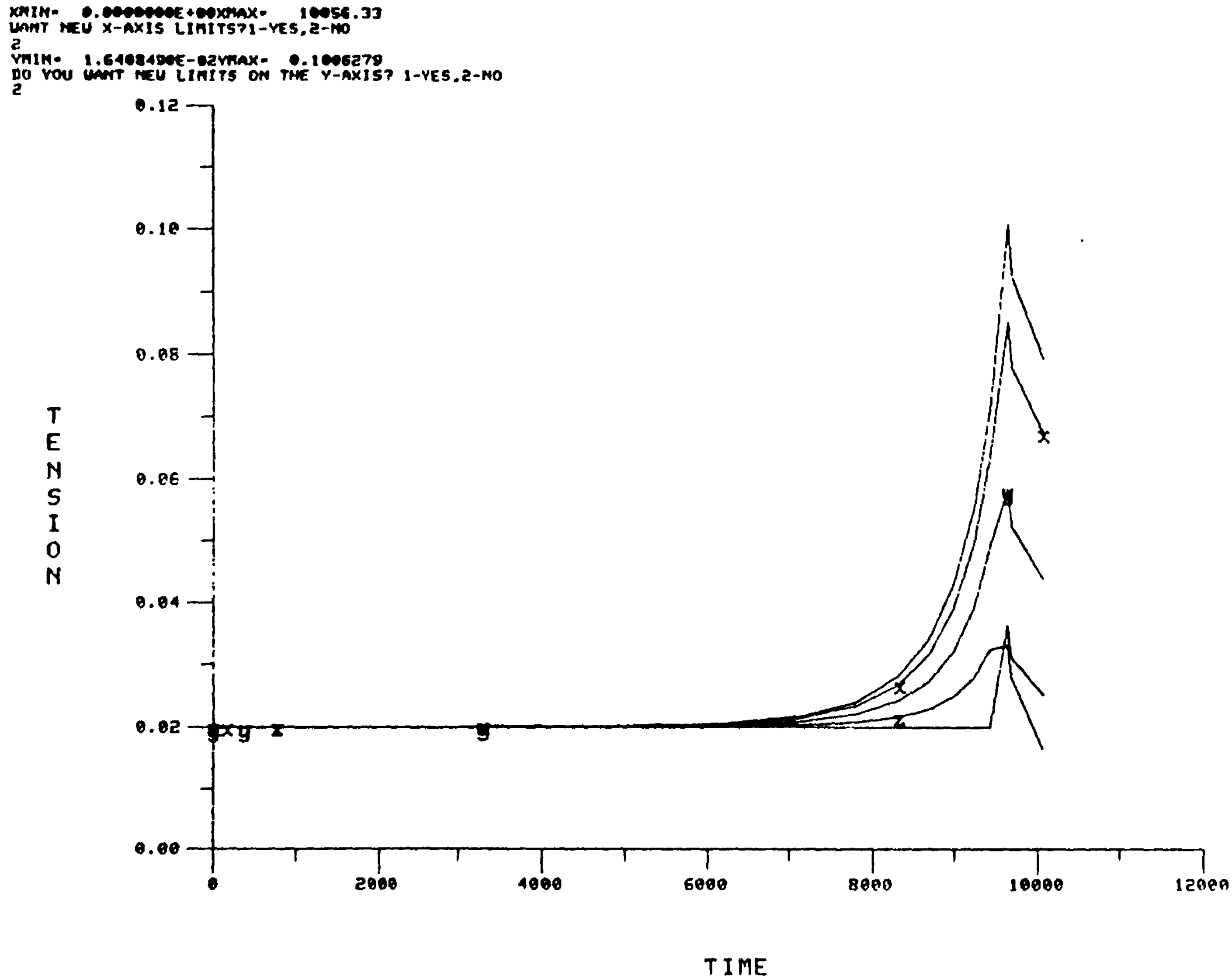



Figure 5.2.13

## 6.0 Conclusions

This report has presented the changes and modifications implemented to the tether simulation to form Version 3.0 as currently in existence at MSFC.

As more changes are made to the simulation at MSFC, they will be documented and submitted as attachments to this report. Other updates to be included at a later time include the SEDS analysis cases, the AMIGA graphics task, the deployer mechanism simulation, and the spinning tether analysis.

# APPENDIX A. SIMULATION CORRECTIONS

Subroutine NGRAV:

Original code

```
      DO 100 J=1,3
         TAUIM1(J) = TAU(J)
         TAU(J) = SNGL(FTAU(I,J))
         TAUI(J) = (TAU(J) + TAUIM1(J))/2.
         LENI = LENI + TAUI(J)*TAUI(J)
         POS(J) = SNGL(FRNOD(I,J))*1000.
 100  CONTINUE
```

Corrected code — The variable J is being used already in the subroutine.

```
      DO 100 JJ=1,3
         TAUIM1(JJ) = TAU(JJ)
         TAU(JJ) = SNGL(FTAU(I,JJ))
         TAUI(JJ) = (TAU(JJ) + TAUIM1(JJ))/2.
         LENI = LENI + TAUI(JJ)*TAUI(JJ)
         POS(JJ) = SNGL(FRNOD(I,JJ))*1000.
 100  CONTINUE
```

Original code — The variable J is being used already in the subroutine.

```
      VIT(J) = SNGL(FVNOD(I,J))*1000.
```

Corrected code

```
      DO 61 JJ=1,3
         VIT(JJ) = SNGL(FVNOD(I,JJ))*1000.
 61   CONTINUE
```

Original code — To avoid math problems.

```
      VN(L) = V(L) - VTAU*TAU(L)/TAUTAU
```

Corrected code

```
      VN(L) = V(L) - VTAU*(TAU(L)/TAUTAU)
```

Original code — Redundant step.

```
      DO 250 L=1,3
 250  VN(L) = VNMAG*VN(L)
```

Corrected code

removed from code

Original code — Incorrect term.

```
      DO 40 L=1,3
         LL = (I-1)*3 + L
         IF(IDR) F(LL) = F(LL) + DBLE(-.5*RHO*CN*AN*VNA*VN(L)/(B*1000.))
         IF(IRA .AND. .NOT. IECL) F(LL) = F(LL) + DBLE(R(L)/(B*1000.))
 40   CONTINUE
```

Corrected code

```
      DO 40 L=1,3
         LL = (I-1)*3 + L
         IF(IDR) F(LL) = F(LL) + DBLE(-.5*RHO*CN*AN*VNMAG*VN(L)/(B*1000.))
         IF(IRA .AND. .NOT. IECL) F(LL) = F(LL) + DBLE(R(L)/(B*1000.))
 40   CONTINUE
```

Original code — Mixed terms.

```
      AN = 2*LENI*SNGL(FTERAD)
```

Corrected code

```
      AN = 2.*LENI*SNGL(FTERAD)
```

Subroutines Uinitl and Rinitl

Original code — Incorrect units.

not originally there

Corrected code

```
      BETA = BETA/1000.D0
```

Subroutines Uprint and Trackf

Original code — Incorrect dimensions.

```
      COMMON/INTERM/TAUQUA(19),TAUMAG(18),BLABLA(37)
```

Corrected code

```
      COMMON/INTERM/TAUQUA(19),TAUMAG(19),BLABLA(37)
```

Subroutine Uthird

Original code — Incorrect dimensions.

```
      DIMENSION P(1)
```

Corrected code

```
      DIMENSION P(7)
```

# APPENDIX B.  SIMULATION DELETIONS

Main:

```
      COMMON/PRAZIS/PREC
      COMMON/TINDPA/TINI,TLNINI,DVNINI,DACINI,XINCRE,XMASS1,NODES
      COMMON/CPOEMS/XMU,WMU,WMUIN,XMUM,XMUS
      COMMON/CONTRO/ICONT
      COMMON/CGEOME/RE,FLAT,FLATSQ,CLIGHT
      COMMON/CSOLAR/SOLPR,ECL,CECL,SECL,XLSUNA,XLSUNB,ECLECC,ECLOM
      COMMON/CONTRL/CNTRL(3)
      COMMON/CDYNAE/STD50,OMT50,OMQ50,OMROT,STD50R,OMT50R,OMQ50R
      COMMON/UMESH/T,DT,Y(992),Z(124)
      PREC = 0.0D0
```

Subroutine Airden:

```
      COMMON/JPOSMS/FILL(36),SUNV(4),LUNVEC(4)
```

Subroutine Bendin

```
      COMMON/TINDPA/DUMMY1(4),XINCRE,DUMMY2,NODES
```

Subroutine Centra

```
      COMMON/CPOEMS/XMU.DUMMY(4)
      COMMON/TINDPA/DUMMY1(4),XINCRE,DUMMY2,NODES
      TYPE *,'IN CENTRA:',RNODEA(K)='
      TYPE *,K,RNODEA(K)
```

Subroutine Coot:

```
      COMMON/CGEOME/RE,FLAT,FLATSQ,CLIGHT
      COMMON/CDYNAE/STD50,OMT50,OMQ50,OMROT,STD50R,OMT50R,OMQ50R
      COMMON/CPOEMS/XMU,WMU,WMUIN,XMUM,XMUS
      COMMON/CSOLAR/SOLPR,ECL,CECL,SECL,XLSUNA,XLSUNB,ECLECC,ECLOM
      RE = MEAN RADIUS OF EQUATOR; FLAT = FLATTENING COEFFICIENT
      REF: GODDARD EARTH MODEL 6
      RE = 6378.144D0
      RE = 6378.156D0
      FLAT = 1.D0/298.257D0
      FLATSQ = FLAT*(2.D0-FLAT)
      CLIGHT = SPEED OF LIGHT. REF: SAO SPEC. REP. 353, 1973.
      CLIGHT = 299792.5D0
      STD50, OMT50, OMQ50 GIVE THE ANGLE BETWEEN GREENWICH MERIDIAN
      AND MEAN EQUINOX (DEG) FROMD = MODIFIED JULIAN DATE (1950)
      ACCORDING TOW = STD50 + OMT50*D + OMQ50*D*D
      STD50R, OMT50R, OMQ50R ARE THE SAME ANGLE EXPRESSED IN RADIANS
      REF: THE ASTRONOMICAL EPHEMERIS 1976, P. 531.
      STD50 = 100.075542d0
      OMT50 = 360.985647335D0
      OMQ50 = .29D-12
      STD50R = STD50*RAD
      OMT50R = OMT50*RAD
      OMQ50R = OMQ50*RAD
      OMROT = MEAN VELOCITY OF ROTATION OF THE EARTH (RAD/SEC)
      OMROT = OMT50R/864.D2
      XMU = CENTRAL EARTH POTENTIAL. REF: SMITHSONIAN STANDARD EARTH III
      AND GODDARD EARTH MODELS 5 AND 6
      XMU = 398601.3D0
      WMU = DSQRT(XMU)
      WMUIN = 1.D0/WMU
```

```
        XMUM, XMUS = MOON, SUN POTENTIAL. REF: JPL DEV. EPHEMERIDES NO. 19
        XMUM = XMU/81.301D0
        XMUS = 132715.0D6
        SOLPR=RADIATION PRESSURE AT MEAN EARTH DISTANCE FROM THE SUN
        IN KG*KM/(M*SEC)**2 REF: NASA TM-X 64627
        SOLPR = 4.51D-9
        ECL, CECL, SECL= INCLINATION OF THE ECLIPTIC AND ITS COS AND SIN.
        XLSUNA, XLSUNB, ECLECC, ECLOM GIVE WITH A PRECISION OF 0.01 DEG
        A VALUE FOR THE LONGITUDE OF THE SUN (LS(RAD)) IN THE ECLIPTI
        FROM: LM=XLSUNA+XLSUNB*DAY; LS=LM+ECLECC*DSIN(LM-ECLOM)
        REF: THE ASTR. EPHEM. 1976, EXTRAPOLATED TO MJD 10000
        ECL = RAD*23.442224D0
        ECL = 0.4090619414299053D0
        CECL = DCOS(ECL)
        SECL = DSIN(ECL)
        XLSUNA = RAD*280.08120D0
        XLSUNB = RAD*.9856473389D0
        ECLECC = 3.343724E-2
        ECLOM = RAD*282.55137D0
Subroutine Deplex
        COMMON/TINDPA/TINI,TLNINI,DVNINI,DAVINI,XINCRE,XMASS1,NODES
Subroutine Fulext
        COMMON/TINDPA/DUMMY1(6),NODES
Subroutine Ghalfi
        COMMON/TINDPA/DUMMY1(4),XINCRE,DUMMY2,NODES
Subroutine Ginteg
        COMMON/TINDPA/DUMMY1(4),XINCRE,DUMMY2,NODES
Subroutine Head
        COMMON/TINDPA/TINI,TLNINI,DVNINI,DACINI,XINCRE,XMASS1,NODES
Subroutine Help
        COMMON/TINDPA/DUMMY1(4),XINCRE,XMASS1,NODES
Subroutine Homoge
        COMMON/TINDPA/DUMMY1(4),XINCRE,DUMMY2,NODES
Subroutine Inext
        COMMON/TINDPA/DUMMY1(4),XINCRE,DUMMY2,NODES
Subroutine Kinema
        COMMON/TINDPA/DUMMY1(4),XINCRE,DUMMY2,NODES
Subroutine Ngrav
        COMMON/PRAZIS/PREC
        COMMON/JPOSMS/FILL(36),FFSO(4),FFLU(4)
        PRE = SNGL(PREC)
        DO 500 I=1,19
 500    TYPE *,'IN NGRAV, FRNOA(',1,')=',FRNOA(I)
Subroutine Normal
        COMMON/TINDPA/DUMMY(4),XINCRE,XMASS1,NODES
        DIMENSION Z(124),F(60)
Subroutine Pertur
        COMMON/TINDPA/DUMMY(7),NODES
        TYPE *,'INPERTUR: K,RNODEA(K)'
        TYPE *,K,RNODEA(K)
```

```
Subroutine Radiat
        COMMON/PRAZIS/PREC
        WRITE(6,1100) AERA,REFCO,CAL,TRANSF`,SCON
        PRE = SNGL(PREC)
        IF(ABS(CAL) .LE. PRE) CAL = 0.0
Subroutine Rbegin
        COMMON/TINDPA/TINI,TLNINI,DVNINI,DACINI,XINCRE,XMASS1,NODES
        COMMON/CONTRL/CNTRL(3)
        READ(5,*)CNTRL
     $  VSHUTT(3),CNTRL(1),CNTRL(2),CNTRL(3)
Subroutine Rinitl
        COMMON/TINDPA/TINI,TLNINI,DVNINI,DACINI,XINCRE,XMASS1,NODES
        COMMON/CPOEMS/XMU,WMU,WMUIN,XMUM,XMUS
        COMMON/CONTRO/ICONT
        ICONT=1
        ZONAL(2) = 0.D0
Subroutine RK78
        SUBROUTINE RK78(IR,T,DT,X,DUM,F1,F2,F3,F4,F5,F6,F7,N,
     $   TOL,DER)
        DIMENSION X(124),DUM(124),F1(124),F2(124),F39124),F4(124),
     $   F5(124),F6(124),F7(124),TOL(124),CH(13),ALF(10)
        PRINT *,'X7 (MUST BE <= 1 TO PASS DT),DT',X7,DT
        PRINT *,'DT ACCEPTED; DT,T:',X9,T
Subroutine ROT
        COMMON/PRAZIS/PREC
        PRE = SNGL(PREC)
Subroutine Rwrite
        COMMON/CONTRL/CNTRL(3)
        COMMON/TINDPA/DUMMY1(4),XINCRE,XMASS1,NODES
     $  VSHUTT(3),CNTRL(1),CNTRL(2),CNTRL(3)
Subroutine Thrust
        COMMON/TINDPA/DUMMY(4),XINCRE,XMASS1,NODES
Subroutine Trackf
        COMMON/UMESH/T,DT,Y(992),Z(124)
        COMMON/TINDPA/TINI,TLNINI,DVNINI,DACINI,XINCRE,XMASS1,NODES
Subroutine Uatox
        COMMON/TINDPA/DUMMY1(4),XINCRE,DUMMY2,NODES
        DIMENSION Y(124)
        TYPE *,'IN UATOX: FOR I = ',I
        TYPE *,'RNODE(N-1,I)=',RNODE(NMIN1,I),
     $          'VNODE(N-1,I)=',VNODE(NMIN1,I)
        TYPE *,'RNODE(',L,',I)=',RNODE(L,I),
     $          'VNODE(',L,',I)=',VNODE(L,I)
        TYPE *,'IN UATOX: K,RNODEA='
        TYPE *,K,RNODEA(K)
Subroutine Ubegin
        COMMON/TINDPA/TINI,TLNINI,DVNINI,DACINI,XINCRE,XMASS1,NODES
        COMMON/CONTROL/CNTRL(3)
        READ(5,*)CNTRL
        WRITE(16,*) 'CNTRL:',CNTRL
```

```
.Subroutine Uder
        COMMON/TINDPA/TINI,TLNINI,DVNINI,DACINI,XINCRE,XMASS1,NODES
        COMMON/PRAXIS/PREC
        DIMENSION Y(124),Z(124),F(60)
        DO 15 K=1,NMIN1
        DO 15 I=1,3
        X=DABS(TAU(K,I))
        IF(X.LT.PREC)TAU(K,I)=0.DO
        X=DABS(RELVEL(K,I))
        IF(X.LT.PREC)RELVEL(K,I)=0.DO
 15     CONTINUE
Subroutine Ugen
        COMMON/UMESH/T,DT,Y(1116)
        COMMON/TINDPA/TINI,TLNINI,DVNINI,DACINI,XINCRE,XMASS1,NODES
Subroutine Uinitl
        COMMON/TINDPA/DUMMY1(4),DUMMY2(2),NODES
        COMMON/CPOEMS/XMU,WMU,WMUIN,XMUM,XMUS
        COMMON/CONTRO/ICONT
        ICONT=1
        ZONAL(2) = 0.DO
Subroutine Uint
        DIMENSION TOL(124)
        COMMON/UMESH/T,DT,A(124),F1(124),F2(124),F3(124),F4(124),
     $  F5(124),F6(124),F7(124),DUM(124)
        CALL RK78(IR,T,DT,A,DUM,F1,F2,F3,F4,F5,F6,F7,NDEQ,TOL,UDER)
Subroutine Unosph
        COMMON/CDYNAE/STD50,OMT50,OMQ50,OMROT,STD50R,OMT50R,OMQ50R
Subroutine Uprint
        COMMON/UMESH/T,DT,Y(992),Z(124)
        COMMON/TINDPA(DUMMY1(4),XINCRE,XMASS1,NODES
        TYPE *,'NSTEP:',NSTEP,'TIME',T
Subroutine Ustop
        CALL TIMCPU(ICP)
        ICP(2) = ICP(2) - ICP(1)
        IF(ICP(2)-5000 .LT. 0) IYES=1
Subroutine Uthird
        COMMON/JPOSMS/FILL(36),ZS(4),ZL(4)
        COMMON/CPOEMS/XMU,WMU,WMUIN,XMUM,XMUS
Subroutine Uxtoa
        COMMON/TINDPA(DUMMY1(4),XINCRE,DUMMY2,NDOES
        DIMENSION Y(124)
Subroutine Vecele
        COMMON/CPEMS/XMU,WMU,WMUIN,XMUM,XMUS
```

# APPENDIX C. ADDITIONS TO SIMULATION

```
Main:                                        Integration additions
        COMMON/GRATON/ITEG
Subroutine Rbegin
        COMMON/GRATON/ITEG
        READ(5,*)ITEG
        PRINT *,ITEG
Subroutine RK78
        COMMON/GRATON/ITEG
        IF(ITEG .EQ. 1) THEN (use variable step size scheme)
        ELSE
        DT6 = DT/6.DO
        DT2 = DT/2.DO
        CALL UDER(T,X,F1)
        T = T + DT2
        DO 200 I=1,N
           DUM (I) = X(I) + DT2*F2(I)
 200    CONTINUE
        CALL UDER(T,DUM,F2)
        DO 210 I=1,N
           DUM(I) = X(I) + DT2*F2(I)
 210    CONTINUE
        CALL UDER(T,DUM,F3)
        T = T + DT2
        DO 220 I=1,N
           DUM(I) = X(I) + DT*F3(I)
 220    CONTINUE
        CALL UDER(T,DUM,F4)
        DO 230 I=1,N
           X(I) = X(I) + DT6*(F1(I) + F4(I) + 2.DO*(F2(I) + F3(I)))
 230    CONTINUE
        ENDIF
Subroutine Ubegin:
        COMMON/GRATON/ITEG
        READ(5,*)ITEG
        PRINT *,ITEG
                                DEPLOYMENT LOGIC
        TL = TLNINI + TLNIN1 + TLNIN2
 999    IF(TLN .GE. TL) THEN
        IF(IDEPL .GT. 1) THEN
        WRITE(6,*)'TETHER HAS BEEN DEPLOYED'
        IDEPL = 1
        DVN = 0.DO
        DAC = 0.DO
        TLNINI = TL
        ENDIF
        TLN = TL
        ENDIF
        IF(TLN .LE. 1.D-5) THEN
        IDEPL = 1
        WRITE(6,*) 'TETHER HAS BEEN RETRACTED'
        ENDIF
```

# APPENDIX D. TEKTRONICS PLOTTING CODE

```
40    CONTINUE
      WRITE(6,*)'INPUT NUMBER OF PLOTS/GRAPH: 1-4'
      READ(5,*) NPLTS
      IF(NPLTS .EQ. 0) GO TO 112
      IF(NPLTS .GE. 5) GO TO 112
      WRITE(6,*)'INPUT NUMBER FOR X VARIABLE'
      READ(5,*) IX
      DO 50 I=1,NPLTS
         WRITE(6,*)'INPUT NUMBER FOR VARIABLE Y(',I,')'
         READ(5,*) IV(I)
50    CONTINUE
      DO 60 I=1,ICOL
         IF(IX .EQ. I) THEN
           DO 70 J=1,IROW
              FLX(J) = X(I,J)
70         CONTINUE
         ENDIF
60    CONTINUE
      DO 75 I=1,ICOL
         IF(IV(1) .EQ. I) THEN
           DO 80 J=1,IROW
              FLY1(J) = X(I,J)
80         CONTINUE
         ENDIF
         IF(IV(2) .EQ. I) THEN
           DO 90 J=1,IROW
              FLY2(J) = X(I,J)
90         CONTINUE
         ENDIF
         IF(IV(3) .EQ. I) THEN
           DO 100 J=1,IROW
              FLY3(J) = X(I,J)
100        CONTINUE
         ENDIF
         IF(IV(4) .EQ. I) THEN
           DO 110 J=1,IROW
              FLY4(J) = X(I,J)
110        CONTINUE
         ENDIF
75    CONTINUE
C
C     GO TO (101,102,103,104) NPLTS
      IF(NPLTS .EQ. 1) GO TO 101
      IF(NPLTS .EQ. 2) GO TO 102
      IF(NPLTS .EQ. 3) GO TO 103
      IF(NPLTS .EQ. 4) GO TO 104
112   WRITE(6,*)'NO PLOT'
      GO TO 1
```

```
C
CC PROGRAM TO READ IN PLOT VARIABLES FROM THE TETHER SIMULATION
CC AND PLOT THEM OUT ON THE NASA VAX.
C
      DIMENSION X(42,2000),FLX(2000),FLY1(2000),FLY2(2000),
     $          FLY3(2000),FLY4(2000),XP(3,7000),FLXP(7000),
     $          FLYP(7000),FLXT(2000),
     $          YDAT(30,2000),Y(40000),XP2(3,7000)
C
     INTEGER IV(42),PLTFLG,FL(42,8),FLABX(20),
     $        FLABY(20),FLP(3,8),FLABXP(8),FLABYP(8),
     $        FLABXT(8),FLABYT(8),FLABX1(8),FLABY1(8)
C
CC READ IN PLOTTING VARIABLES FROM THE TETHER SIMULATION
C
      OPEN(UNIT=1,FILE ='tplot1.dat',FORM='UNFORMATTED',STATUS='OLD')
      OPEN(UNIT=2,FILE = 'tplot2.dat',STATUS='OLD')
      OPEN(UNIT=3,FILE ='pplot1.dat',FORM='UNFORMATTED',STATUS='OLD')
      OPEN(UNIT=4,FILE = 'pplot2.dat',STATUS='OLD')
C
      IP = 0
      IW = 0
      READ(2,5) ICOL,IROW
      DO 10 I=1,ICOL
         READ(2,15) (FL(I,J),J=1,8)
 10   CONTINUE
      DO 20 I=1,IROW
         READ(1) (X(J,I),J=1,ICOL)
 20   CONTINUE
      READ(4,5)ICOL2,IROW2
      DO 210 I=1,ICOL2
         READ(4,15) (FLP(I,J),J=1,8)
210   CONTINUE
      DO 220 I=1,IROW2
         READ(3)(XP(J,I),J=1,ICOL2)
220   CONTINUE
1     WRITE(6,*) 'VARIABLE LIST:'
      DO 30 I=1,ICOL
         WRITE(6,25) I,(FL(I,J),J=1,8)
 30   CONTINUE
      WRITE(6,*)'WHAT TYPE OF PLOT?'
      WRITE(6,*)'1-REGULAR, 2-WALKING, 3-TENSION, 4-NONE'
      READ(5,*)PLTFLG
      IF(PLTFLG .EQ. 2) GO TO 201
      IF(PLTFLG .EQ. 3) GO TO 301
      IF(PLTFLG .EQ. 4) GO TO 202
      DO 40 I=1,ICOL
         IV(I) = 0
```

```
      C
  101 WRITE(6,*)'ONE PLOT/GRAPH'
      WRITE(6,*)'ENTER X-LABEL (8)'
      READ(5,15)(FLABX1(I),I=1,8)
      WRITE(6,*)'ENTER Y-LABEL (8)'
      READ(5,15)(FLABY1(I),I=1,8)
      CALL INITT(960)
      CALL TERM(3,1024)
      CALL CRVPLT(1,IROW,FLX,FLY1,8,FLABX1,8,FLABY1)
      GO TO 1
      C
  102 WRITE(6,*)'TWO PLOTS/GRAPH'
      WRITE(6,*)'ENTER X-LABEL (20)'
      READ(5,35)(FLABX(I),I=1,20)
      WRITE(6,*)'ENTER Y-LABEL (20)'
      READ(5,35)(FLABY(I),I=1,20)
      CALL INITT(960)
      CALL TERM(3,1024)
      CALL CRV22(1,IROW,FLX,FLY1,FLY2,20,FLABX,20,FLABY)
      GO TO 1
      C
  103 WRITE(6,*)'THREE PLOTS/GRAPH'
      WRITE(6,*)'ENTER X-LABEL (20)'
      READ(5,35)(FLABX(I),I=1,20)
      WRITE(6,*)'ENTER Y-LABEL (20)'
      READ(5,35) (FLABY(I),I=1,20)
      CALL INITT(960)
      CALL TERM(3,1024)
      CALL CRV3(1,IROW,FLX,FLY1,FLY2,FLY3,20,FLABX,20,FLABY)
      GO TO 1
      C
  104 WRITE(6,*)'FOUR PLOTS/GRAPH'
      WRITE(6,*)'ENTER X-LABEL (20)'
      READ(5,35)(FLABX(I),I=1,20)
      WRITE(6,*)'ENTER Y-LABEL (20)'
      READ(5,35) (FLABY(I),I=1,20)
      CALL INITT(960)
      CALL TERM(3,1024)
      CALL CRV4(1,IROW,FLX,FLY1,FLY2,FLY3,FLY4,20,FLABX,20,FLABY)
      GO TO 1
      C
      CCC     WALKING POSITION PLOTS
      C
  201 CONTINUE
      IRRW =IROW
  221 WRITE(6,*)'ENTER NUMBER OF TETHER NODES'
      READ(5,*)NODES
      N1 = NODES + 1
```

ORIGINAL PAGE IS
OF POOR QUALITY

```
          WRITE(6,*)'DO YOU WANT EVERY TIME POINT? 1-YES,2-NO'
          READ(5,*)NTPT
          IF(NTPT .NE. 1) THEN
             WRITE(6,*)'ENTER N, FOR EVERY NTH POINT'
             READ(5,*)N
             N2 = (IROW - 1)/N
             DO 501 J=1,3
                 DO 500 I=1,N1
                    XP2(J,I) = XP(J,I)
  500            CONTINUE
  501        CONTINUE
             N4 = 1
             N3 = N
             DO 502 K=1,N2
                 DO 503 J=1,3
                    DO 504 I=1,N1
                       XP2(J,N4*N1+I) = XP(J,N3*N1+I)
  504               CONTINUE
  503            CONTINUE
                 N3 = N3 + N
                 N4 = N4 + 1
  502        CONTINUE
             IRW = N4
             IF(N*N2 .NE. (IRRW-1)) THEN
                DO 505 J=1,3
                   DO 506 I=1,N1
                      XP2(J,N4*N1+I) = XP(J,(IRRW-1)*N1+I)
  506              CONTINUE
  505          CONTINUE
               IRW = N4+1
             ENDIF
             IROW2 = IRW*N1
          ELSE
             DO 507 J=1,3
                 DO 508 I=1,IROW2
                    XP2(J,I) = XP(J,I)
  508            CONTINUE
  507        CONTINUE
          ENDIF
C         WRITE(6,*)'DO YOU WANT TO SCALE DATA ITSELF?1-YES,2-NO'
C         READ(5,*)NSCAL
C         IF(NSCAL .EQ. 1) THEN
C            XX = 0.0
C            YX = 0.0
C            ZX = 0.0
C            DO 400 I=1,IROW
C                XMX = XP2(1,N1*I-1) - XP2(1,N1*I)
C                YMX = XP2(2,N1*I-1) - XP2(2,N1*I)
```

```
C              ZMX = XP2(3,N1*I-1) - XP2(3,N1*I)
C              ZAB = ABS(ZMX)
C              XAB = ABS(XMX)
C              YAB = ABS(YMX)
C              IF(XMX .GT. ABS(XX)) XX = XMX
C              IF(XMX .LT. 0.0) THEN
C                IF(XAB .GT. ABS(XX)) XX = SIGN(XAB,XMX)
C              ENDIF
C              IF(YMX .GT. ABS(YX)) YX = YMX
C              IF(YMX .LT. 0.0) THEN
C                IF(YAB .GT. ABS(YX)) YX = SIGN(YAB,YMX)
C              ENDIF
C              IF(ZMX .GT. ABS(ZX)) ZX = ZMX
C              IF(ZMX .LT. 0.0) THEN
C                IF(ZAB .GT. ABS(ZX)) ZX = SIGN(ZAB,ZMX)
C              ENDIF
C 400         CONTINUE
C           WRITE(6,*)'XX,YX,ZX:',XX,YX,ZX
C             ZX2 = .5*ZX
C             IF(YX .EQ. 0.0) GO TO 401
C             YSCAL = ABS(ZX2/YX)
C             GO TO 402
C 401         YSCAL = 1.0
C 402          CONTINUE
C             XSCAL = ABS(ZX2/XX)
CC
CC          FIND MAXIMUM DEFLECTION IN THE X AND Y AXES
CC
            IF(IW .EQ. 0) THEN
               XX = 0.0
               YX = 0.0
               NINT = 0

               DO 400 I=1,IRPW
                  DO 402 J=2,NODES
                     XMX = XP(1,J+NINT) - XP(1,1+NINT)
                     YMX = XP(2,J+NINT) - XP(2,1+NINT)
                     XABS = ABS(XMX)
                     YABS = ABS(YMX)
                     IF(XABS .GT. XX) THEN
                        XX = XABS
                        IXX = I
                        JX = J
                        XSIGN = XMX
                     ENDIF
                     IF(YABS .GT. YX) THEN
                        YX = YABS
                        IYY = I
                        JY = J
```

```
                         YSIGN = YMX
                      ENDIF
       402          CONTINUE
                    NINT = NINT + N1
       400        CONTINUE
                  WRITE(6,*)'MAX IN PLANE DEFLECTION = ',XSIGN
                  WRITE(6,*)'AT THE',IXX,'TIME STEP, NODE #',JX
                  WRITE(6,*)'MAX OUT OF PLANE DEFLECTION =',YSIGN
                  WRITE(6,*)'AT THE',IYY,'TIME STEP, NODE #',JY
                  IU = 1
                ENDIF
                  N42 = 0
                  N32 = 1
                WRITE(6,*)'ENTER XSCAL AND YSCAL'
                READ(5,*)XSCAL,YSCAL
C               WRITE(6,*)'XSCAL,YSCAL:',XSCAL,YSCAL
*P
                  DO 430 I=1,IROW
                    DO 440 J=1,NODES-1
                      JJ = J+1+N42
C                     XP2(1,JJ) = XSCAL*(XP2(1,JJ) - XP2(1,N1*I)) +
C         $                     XP2(1,N1*I)
C                     XP2(2,JJ) = YSCAL*(XP2(2,JJ) - XP2(2,N1*I)) +
C         $                     XP2(2,N1*I)
                      XP2(1,JJ) = XP2(1,JJ) - XP2(1,N1*I) + XSCAL*XP2(1,N1*I)
                      XP2(2,JJ) = XP2(2,JJ) - XP2(2,N1*I) + YSCAL*XP2(2,N1*I)
       440      CONTINUE
C               WRITE(6,*)'XP2(1,3),XP(1,3):',XP2(1,3),XP(1,3)
                    N42 = N42+N1
                    XP2(1,N32) = XSCAL*XP2(1,N32)
                    XP2(1,N32+NODES) = XSCAL*XP2(1,N32+NODES)
                    XP2(2,N32) = YSCAL*XP2(2,N32)
                    XP2(2,N32+NODES) = YSCAL*XP2(2,N32+NODES)
*P
                    N32 = N32 + N1
       430        CONTINUE
C               ENDIF
                WRITE(6,*)'VARIABLES:'
                WRITE(6,*) 'IN-PLANE, OUT-OF-PLANE, RADIAL'
                WRITE(6,*) 'ONE PLOT/GRAPH'
                WRITE(6,*) 'INPUT X AND Y VARIABLE NUMBER'
                READ(5,*) IXP,IYP
                DO 230 I=1,ICOL2
                  IF(IXP .EQ. I) THEN
                    DO 235 J=1,IROW2
                      FLXP(J) = XP2(I,J)
       235          CONTINUE
                    DO 236 K=1,8
                    FLABXP(K) = FLP(I,K)
       236          CONTINUE
*
```

```
          ENDIF
          IF(IYP .EQ. I) THEN
            DO 240 J=1,IROW2
              FLYP(J) = XP2(I,J)
  240       CONTINUE
            DO 241 K=1,8
            FLABYP(K) = FLP(I,K)
  241       CONTINUE
          ENDIF
  230   CONTINUE
        CALL INITT(960)
        CALL TERM(3,1024)
        CALL CRVP(1,IROW2,FLXP,FLYP,8,FLABXP,8,FLABYP,NODES,IROW)
        IROW = IRRW
        GO TO 1
C
CCC     TENSION PLOTTING CODE
C
  301   CONTINUE
        WRITE(6,*) 'ENTER BASE NUMBER OF VARIABLES'
        READ(5,*)IBASE
        NMIN1 = ICOL - IBASE
        DO 310 J=1,IROW
          FLXT(J) = X(1,J)
  310   CONTINUE
        DO 311 K=1,8
        FLABXT(K) = FL(1,K)
        FLABYT(K) = FL(IBASE+1,K)
  311   CONTINUE
        DO 320 I= IBASE+1,ICOL
          II = I - IBASE
          DO 330 J=1,IROW
            YDAT(II,J) = X(I,J)
  330     CONTINUE
  320   CONTINUE
        DO 340 I=1,NMIN1
          II = I-1
          DO 350 J=1,IROW
            Y(J+II*IROW) = YDAT(I,J)
  350     CONTINUE
  340   CONTINUE
        CALL INITT(960)
        CALL TERM(3,1024)
        CALL TENCRV(1,IROW,NMIN1,FLXT,Y,8,FLABXT,8,FLABYT)
        GO TO 1
  202   WRITE(6,*) 'FINISHED PLOTTING'
C
    5   FORMAT(2I5)
```

```
      FORMAT(8A1)
35    FORMAT(I5,8A1)
36    FORMAT(80A1)
      CLOSE(UNIT=1)
      CLOSE(UNIT=2)
      CLOSE(UNIT=3)
      CLOSE(UNIT=4)
      STOP
      END
```

Subplt.+
4-18-85

```
C
CCC     SUBROUTINE TO PLOT 2 CURVES ON ONE GRAPH
C
        SUBROUTINE CRVE2(IH,IFO,XDATA,YDATA1,YDATA2,LENX,LABX,
     $                   LENY,LABY)
C
        DIMENSION XDATA(2000),YDATA1(2000),YDATA2(2000),
     $            LABX(20),LABY(20)
C
        BMIN = 10000.
        BMAX = -10000.
        AMIN = BMIN
        AMAX = BMAX
        CALL BINITT
        CALL CHRSIZ(2)
        CALL NPTS(IFO)
        CALL XFRM(2)
        CALL YFRM(2)
C
CCC     FIND MINIMUM AND MAXIMUM VALUES
C
        CALL MNMX(YDATA1,BMIN,BMAX)
        CALL MNMX(YDATA2,BMIN,BMAX)
        CALL MNMX(XDATA,AMIN,AMAX)
        WRITE(6,*)'XMIN=',AMIN,'XMAX=',AMAX
        WRITE(6,*)'WANT NEW X-AXIS LIMITS?1-YES,2-NO'
        READ(5,*)ILIX
        IF(ILIX .NE. 1) GO TO 15
        WRITE(6,*)'ENTER XMIN,XMAX'
        READ(5,*)AMIN,AMAX
  15    CALL DLIMX(AMIN,AMAX)
        WRITE(6,*)'YMIN',BMIN,'YMAX=',BMAX
        WRITE(6,*)'DO YOU WANT NEW Y LIMITS? 1-YES, 2-NO'
        READ(5,*)ILIM
        IF(ILIM .NE. 1) GO TO 10
        WRITE(6,*)'ENTER YMIN AND YMAX'
        READ(5,*)BMIN,BMAX
  10    CALL DLIMY(BMIN,BMAX)
        CALL CHRSIZ(3)
        CALL CHECK(XDATA,YDATA2)
        CALL DSPLAY(XDATA,YDATA2)
        IX = (750 - LENX*13)/2 + 150
        IY = (575 - LENY*21)/2 + 125 + LENY*21
        CALL CHRSIZ(2)
        CALL MOVABS(IX,25)
        CALL TSEND
        CALL HLABEL(LENX,LABX)
        CALL SIZES(.25)
```

ORIGINAL PAGE IS
OF POOR QUALITY

```
*P
          READ(5,*)ILIX
          IF(ILIX .NE. 1) GO TO 15
          WRITE(6,*)'ENTER XMIN,XMAX'
          READ(5,*)AMIN,AMAX
  15      CALL DLIMX(AMIN,AMAX)
          WRITE(6,*) 'YMIN=',BMIN,'YMAX=',BMAX
          WRITE(6,*)'DO YOU WANT NEW LIMITS ON THE Y-AXIS? 1-YES,2-NO'
          READ(5,*) ILIM
          IF(ILIM .NE. 1) GO TO 10
          WRITE(6,*)'ENTER Y-MIN AND Y-MAX'
          PEAD(5,*) BMIN,BMAX
  10      CALL DLIMY(BMIN,BMAX)
          CALL CHRSIZ(3)
          CALL CHECK(XDATA,YDATA3)
          CALL DSPLAY(XDATA,YDATA3)
          IX = (750 - LENX*13)/2 + 150
*P
          IY = (575 - LENY*21)/2 + 125 + LENY*21
          CALL CHRSIZ(2)
          CALL MOVABS(IX,25)
          CALL TSEND
          CALL HLABEL(LENX,LABX)
          CALL SIZES(.25)
          CALL SYMBL(120)
          CALL STEPS(10)
          CALL CPLOT(XDATA,YDATA1)
          CALL SYMBL(121)
          CALL STEPS(15)
          CALL CPLOT(XDATA,YDATA2)
          CALL CHRSIZ(2)
C         CALL MOVABS(200,725)
C         CALL HLABEL(LENY,LABY)
          CALL MOVABS(25,IY)
*P
          CALL VLABEL(LENY,LABY)
          CALL CHRSIZ(4)
          GO TO (1,2) IH
  2       CALL FINITT(0,700)
          CALL HOME
          READ(5,*)
          CALL NEWPAG
          GO TO 3
  1       CALL HDCOPY
          CALL NEWPAG
          CALL FINITT(0,700)
  3       RETURN
          END
C
CCC
CCC       SUBROUTINE TO PLOT 4 CURVES ON ONE GRAPH
*
```

ORIGINAL PAGE IS
OF POOR QUALITY

```
            CALL SYMBL(120)
            CALL STEPS(10)
            CALL CPLOT(XDATA,YDATA1)
            CALL CHRSIZ(2)
C           CALL MOVABS(200,725)
C           CALL HLABEL(LENY,LABY)
            CALL MOVABS(25,IY)
            CALL VLABEL(LENY,LABY)
            CALL CHRSIZ(4)
            GO TO (1,2)IH
 2          CALL FINITT(0,700)
            CALL HOME
            READ(5,*)
            CALL NEWPAG
            GO TO 3
 1          CALL HDCOPY
*P
            CALL NEWPAG
            CALL FINITT(0,700)
 3          RETURN
            END
C
CCC
CCC         SUBROUTINE TO PLOT 3 CURVES ON ONE GRAPH
CCC
C
            SUBROUTINE CPV3(IH,IFO,XDATA,YDATA1,YDATA2,YDATA3,LENX,
     $                      LABX,LENY,LABY)
C
            DIMENSION XDATA(2000),YDATA1(2000),YDATA2(2000),YDATA3(2000),
     $               LABX(20),LABY(20)
C
            BMIN = 10000.
*P
            BMAX = -10000.
            AMIN = BMIN
            AMAX = BMAX
            CALL BINITT
            CALL CHRSIZ(2)
            CALL NPTS(IFO)
            CALL XFRM(2)
            CALL YFRM(2)
C
C           FIND MINIMUM AND MAXIMUM VALUES
C
            CALL MNMX(YDATA1,BMIN,BMAX)
            CALL MNMX(YDATA2,BMIN,BMAX)
            CALL MNMX(YDATA3,BMIN,BMAX)
            CALL MNMX(XDATA,AMIN,AMAX)
            WRITE(6,*)'WANT NEW X-AXIS LIMITS?1-YES,2-NO'
*
```

```
      CCC
      C
            SUBROUTINE CRV4(IH,IFO,XDATA,YDATA1,YDATA2,YDATA3,YDATA4,
     $                LENX,LABX,LENY,LABY)
      C
            DIMENSION XDATA(2000),YDATA1(2000),YDATA2(2000),YDATA3(2000),
     $          YDATA4(2000),LABX(20),LABY(20)
      C
            BMIN = 10000.
            BMAX = -10000.
            AMIN = BMIN
            AMAX = BMAX
            CALL BINITT
            CALL CHRSIZ(2)
            CALL NPTS(IFO)
            CALL XFRM(2)
*P
            CALL YFRM(2)
      C
      C     FIND MINIMUM AND MAXIMUM VALUES
      C
            CALL MNMX(YDATA1,BMIN,BMAX)
            CALL MNMX(YDATA2,BMIN,BMAX)
            CALL MNMX(YDATA3,BMIN,BMAX)
            CALL MNMX(YDATA4,BMIN,BMAX)
            CALL MNMX(XDATA,AMIN,AMAX)
            WRITE(6,*)'XMIN=',AMIN,'XMAX=',AMAX
            WRITE(6,*)'WANT NEW X-AXIS LIMITS?1-YES,2-NO'
            READ(5,*)ILIX
            IF(ILIX .NE. 1) GO TO 15
            WRITE(6,*)'ENTER XMIN,XMAX'
            READ(5,*)AMIN,AMAX
       15   CONTINUE
*P
            WRITE(6,*) 'BMIN=',BMIN,'BMAX=',BMAX
            WRITE(6,*)'DO YOU WANT NEW LIMITS ON THE Y-AXIS? 1-YES,2-NO'
            READ(5,*) ILIM
            IF(ILIM .NE. 1) GO TO 10
            WRITE(6,*)'ENTER Y-MIN AND Y-MAX'
            READ(5,*) BMIN,BMAX
       10   CALL DLIMY(BMIN,BMAX)
            CALL CHRSIZ(3)
            CALL CHECK(XDATA,YDATA4)
            CALL DSPLAY(XDATA,YDATA4)
            IX = (750 - LENX*13)/2 + 150
            IY = (575 - LENY*21)/2 + 125 + LENY*21
            CALL CHRSIZ(2)
            CALL MOVABS(IX,25)
            CALL TSEND
            CALL HLABEL(LENX,LABX)
*
```

```
      CALL SIZES(.25)
      CALL SYMBL(120)
      CALL STEPS(10)
      CALL CPLOT(XDATA,YDATA1)
      CALL SYMBL(121)
      CALL STEPS(15)
      CALL CPLOT(XDATA,YDATA2)
      CALL SYMBL(122)
      CALL STEPS(20)
      CALL CPLOT(XDATA,YDATA3)
      CALL CHRSIZ(2)
C     CALL MOVABS(200,725)
C     CALL HLABEL(LENY,LABY)
      CALL MOVABS(25,IY)
      CALL VLABEL(LENY,LABY)
      CALL CHRSIZ(4)
      GO TO (1,2) IH
 2    CALL FINITT(0,700)
      CALL HOME
      READ(5,*)
      CALL NEWPAG
      GO TO 3
 1    CALL HDCOPY
      CALL NEWPAG
      CALL FINITT(0,700)
 3    RETURN
      END
C
CCC
CCC   SUBROUTINE TO TENSION CURVES ON ONE GRAPH
CCC
C
      SUBROUTINE TENCRV(IH,IFO,NMIN1,XDATA,Y,
     $                  LENX,LABX,LENY,LABY)
C
      DIMENSION XDATA(2000),Y(40000),LABX(8),LABY(8),TEMP(2000)

C
      BMIN = 10000.
      BMAX = -10000.
      AMIN = BMIN
      AMAX = BMAX
      CALL BINITT
      CALL CHRSIZ(2)
      CALL NPTS(IFO)
      CALL XFRM(2)
      CALL YFRM(2)
C
```

```
C          FIND MINIMUM AND MAXIMUM VALUES
C
           DO 5 I=1,NMIN1
              II = I -1
              DO 15 J=1,IFO
                 TEMP(J) = Y(J+II*IFO)
 15           CONTINUE
              CALL MNMX(TEMP,BMIN,BMAX)
 5         CONTINUE
           CALL MNMX(XDATA,AMIN,AMAX)
           WPITE(6,*)'XMIN=',AMIN,'XMAX=',AMAK
           WRITE(6,*)'WANT NEW X-AXIS LIMITS?1-YES,2-NO'
           READ(5,*)ILIX
           IF(ILIX .NE. 1) GO TO 8
           WRITE(6,*)'ENTER XMIN,XMAX'
           READ(5,*)AMIN,AMAX
*P
 8         CALL DLIMX(AMIN,AMAX)
           WRITE(6,*) 'YMIN=',BMIN,'YMAX=',BMAX
           WRITE(6,*)'DO YOU WANT NEW LIMITS ON THE Y-AXIS? 1-YES,2-NO'
           READ(5,*) ILIM
           IF(ILIM .NE. 1) GO TO 10
           WRITE(6,*)'ENTER Y-MIN AND Y-MAX'
           READ(5,*) BMIN,BMAX
 10        CALL DLIMY(BMIN,BMAX)
           CALL CHRSIZ(3)
           DO 25 J=1,IFO
              TEMP(J) = Y(J+IFO*(NMIN1-1))
 25        CONTINUE
           CALL CHECK(XDATA,TEMP)
           CALL DSPLAY(XDATA,TEMP)
           IX = (750 - LENX*13)/2 + 150
           IY = (575 - LENY*21)/2 + 125 + LENY*21
*P
           CALL CHRSIZ(2)
           CALL MOVABS(IX,25)
           CALL TSEND
           CALL HLABEL(LENX,LABX)
           CALL SIZES(.25)
           DO 35 I=1,NMIN1-1
              II = I-1
              DO 45 J=1,IFO
                 TEMP(J) = Y(J+II*IFO)
 45           CONTINUE
              CALL CPLOT(XDATA,TEMP)
              CALL SYMBL(119+I)
              CALL STEPS(5+5*I)
 35        CONTINUE
           CALL CHRSIZ(2)
C          CALL MOVABS(200,725)
*
```

```
C      CALL HLABEL(LENY,LABY)
       CALL MOVABS(25,IY)
       CALL VLABEL(LENY,LABY)
       CALL CHRSIZ(4)
       GO TO (1,2) IH
 2     CALL FINITT(0,700)
       CALL HOME
       READ(5,*)
       CALL NEWPAG
       GO TO 3
 1     CALL HDCOPY
       CALL NEWPAG
       CALL FINITT(0,700)
 3     RETURN
       END
       SUBROUTINE CRVP(IH,NPTSS,XARRAY,YARRAY,LENX,LABX,LENY,LABY,
$P
     $               NODES,IROW)
C
C      SUBROUTINE TO PLOT WALKING CURVES ON A GRAPH
C
       DIMENSION XARRAY(7000),YARRAY(7000),LABX(8),LABY(8)
       DIMENSION XTEMP(7000),YTEMP(7000)
C
       AMIN = 10000.
       AMAX = -10000.
       BMIN = AMIN
       BMAX = AMAX
       CALL BINITT
C
       CALL CHRSIZ(2)
       CALL NPTS(NPTSS)
       CALL XFRM(2)
$P
       CALL YFRM(2)
       CALL MNMX(XARRAY,AMIN,AMAX)
       CALL MNMX(YARRAY,BMIN,BMAX)
       WRITE(6,*)'XMIN=',AMIN,'XMAX=',AMAX
       WRITE(6,*)'WANT NEW X-AXIS LIMITS? 1-YES,2-NO'
       READ(5,*)ILIX
       IF(ILIX .NE. 1) GO TO 15
       WRITE(6,*)'ENTER XMIN,XMAX'
       READ(5,*)AMIN,AMAX
 15    CALL DLIMX(AMIN,AMAX)
       WRITE(6,*)'YMIN=',BMIN,'YMAX=',BMAX
       WRITE(6,*)'WANT NEW Y-AXIS LIMITS? 1-YES,2-NO'
       READ(5,*)ILIY
       IF(ILIY .NE. 1) GO TO 20
       WRITE(6,*)'ENTER YMIN,YMAX'
       READ(5,*)BMIN,BMAX
$
```

```
20    CALL DLIMY(BMIN,BMAX)
      DO 25 I=1,NODES
         XTEMP(I) = XARRAY(I)
         YTEMP(I) = YARRAY(I)
25    CONTINUE
      CALL CHECK(XARRAY,YARRAY)
      CALL NPTS(NODES)
C     CALL FRAME
      CALL DSPLAY(XTEMP,YTEMP)
C
      IX = (750 - LENX*13)/2 + 150
      IY = (675 - LENY*21)/2 + 125 + LENY*21
      CALL CHRSIZ(2)
      CALL MOVABS(IX,25)
      CALL TSEND
      CALL HLABEL(LENX,LABX)
*P
C     CALL MOVABS(25,IY)
      CALL SIZES(.25)
      DO 30 I=2,IROW
         DO 35 J=1,NODES
            XTEMP(J) = XARRAY((NODES+1)*(I-1)+J)
            YTEMP(J) = YARRAY((NODES+1)*(I-1)+J)
35       CONTINUE
         CALL CPLOT(XTEMP,YTEMP)
      CALL MOVABS(0,0)
      CALL TSEND
30    CONTINUE
C     CALL TSEND
      CALL MOVABS(25,IY)
      CALL VLABEL(LENY,LABY)
      CALL CHRSIZ(4)
      GO TO (1,2)IH
*P
2     CALL FINITT(0,700)
      CALL HOME
      READ(5,*)
      CALL NEWPAG
      GO TO 3
1     CALL HDCOPY
      CALL NEWPAG
      CALL FINITT(0,700)
3     RETURN
      END
*
No such line exists
*
```