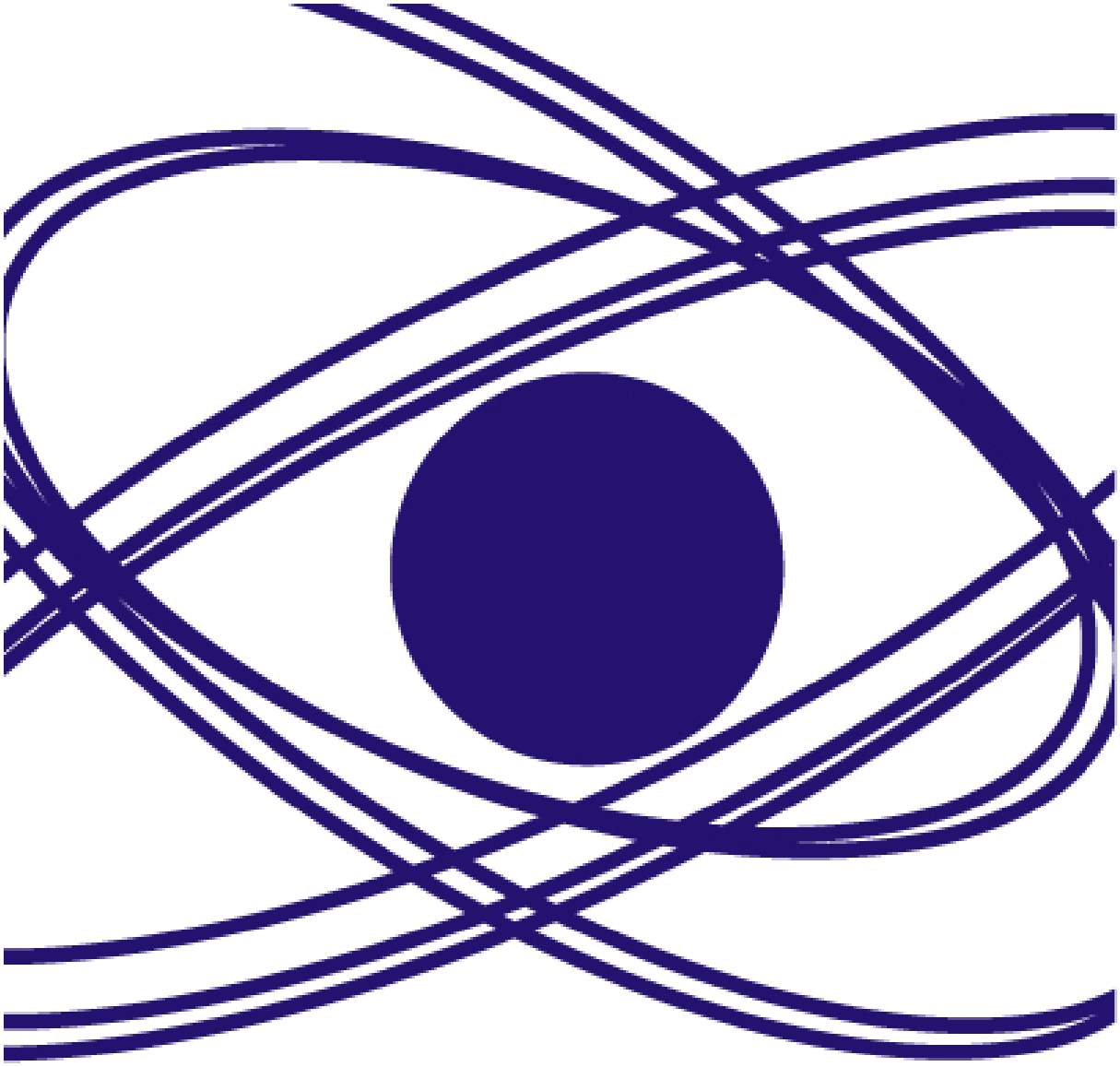


Tapestry

Graphical User Interface for the

Navigation Laboratories, Inc.
GPS Constellation Simulators

Math Models Supplement

Revision V- Jan 2002

TABLE OF CONTENTS

I.1 Mathematical Models

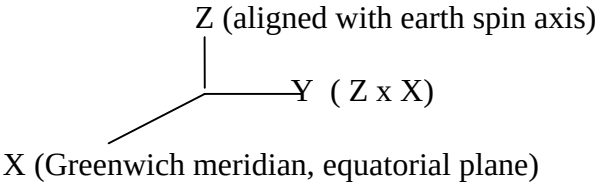
I.1.1 Vehicle Motion model

Coordinate Frames

For terrestrial applications, Tapestry has mechanized the following coordinate systems:

Earth-Centered-Earth-Fixed: (ECEF)

This coordinate system is used for the GPS satellite vectors and is the frame selected for the truth data output. It is defined, consistent with the WGS84 convention as follows:



Geographic (LLA):

This frame is used primarily to communicate with the user through the man machine interface. It uses latitude, longitude, and ellipsoidal altitude. It is a true North pointing frame.

Wander-Azimuth Locally Level Frame (Platform/NED)

The wander azimuth frame is the computational frame used for all terrestrial motion equations. This frame was selected to avoid the singularity associated with a geographic (LLA) frame in which the vertical craft torque rate approaches infinity at the two poles.

The wander frame differs from the LLA frame in that North is not referenced to the earth spin axis (north pole) except at the simulation start time. All terrestrial maneuvers are mechanized in the locally level wander azimuth frame with the wander angle offset computed when needed. This frame generates great circle navigation when "cruising" at a constant wander heading. Note as the vehicle moves due (wander) east, the wander frame rotates clockwise relative to a true north pointing frame. The rotation rate is roughly given by :

$$\omega = \text{Veast}/R\tan(\lambda) \qquad \text{(clockwise if Veast} > 0)$$

See Brockstein and Kuba for the equations that describe the wander azimuth frame.

Earth Centered Inertial (ECI)

For spaceborne applications an Inertial frame is required as an *internal* computational frame. The selected frame is an Earth-Centered-Inertial frame defined such that it aligns exactly with the ECEF frame at the beginning of the GPS week (Sunday midnight). Selecting the ECI frame aligned with ECEF at the start week epoch time simplifies the coordinate frame transformations between the ECI and ECEF frame to a z-axis rotation defined by:

$$C^{E}_{I} = \begin{vmatrix} \cos(\Delta t\,\Omega) & \sin(\Delta t\,\Omega) & 0 \\ -\sin(\Delta t\,\Omega) & \cos(\Delta t\,\Omega) & 0 \\ 0 & 0 & 1 \end{vmatrix}$$

where Δt is the time into simulation week and Ω is earth rate.

Both the geoid gravity model and the state vector propagation function for spaceborne navigation are mechanized in the ECI frame. The transformations between ECI and ECEF for the state vector are given by:

$$P^{ECEF} = C^{E}_{I}\, P^{ECI}$$

$$V^{ECEF} = C^{E}_{I}\, V^{ECI} - \Omega \times P^{ECEF}$$

$$A^{ECEF} = C^{E}_{I}\, A^{ECI} - 2\Omega \times V^{ECEF} - \Omega \times \Omega \times P^{ECEF}$$

$$J^{ECEF} = C^{E}_{I}\, J^{ECI} - 2\Omega \times A^{ECEF} - \Omega \times (\Omega \times V^{ECEF}) - n^{2}V^{ECEF}$$

where $n \approx \mu / (|P|)^{3}$ and μ is the WGS84 gravitational constant.

Local Level Orbit Frame (DXR)

For spaceborne applications a locally level frame is used as a reference for the thrust and attitude maneuvers. The frame is defined as:

$$\begin{aligned} \text{Down Track (D)} &= V^{ECI}/|V^{ECI}| \\ \text{Cross Track (X)} &= D \times R^{ECI}/|R^{ECI}| \\ \text{Radial (R)} &= X \times D \end{aligned}$$

Any space craft thrusting is applied along one of the DXR unit vectors independent of vehicle attitude. Spacecraft Roll, Pitch, Yaw are mechanized relative to the DXR frame.

Keplerian Orbit Frame

Keplerian orbit parameterization are used for two purposes; (a) as an input frame for the initial navigation state, and (b) to mechanize the *Keplerian Orbit* maneuver in the spaceborne script interface. The selected Keplerian parameters are:

T_P	=	time of perigee
a	=	orbit semi-major axis
e	=	orbit eccentricity
i	=	orbit inclination angle
Ω_O	=	longitude of the ascending node
ω	=	argument of perigee

The transformation from the Keplerian to the ECI frame is given by the following source code segment:

Δt = time into simulation.
μ = WGS84 Gravitational constant

```
delta = Δt - Tp;          // time relative to perigee
n     = sqrt(μ/a/a/a);    // mean motion
m     = n*delta;          // mean anomaly
cosw  = cos(ω);
sinw  = sin(ω);
cosi  = cos(i);
sini  = sin(i);
ef    = sqrt( (1.0 - e)*(1.0 + e) );
e2    = e*e;
e3    = e2*e;
e4    = e3*e;
e5    = e4*e;
e6    = e5*e;
```

// Solve for the trig-arguments of true anomaly - accurate to centimeters

```
sinm  = sin(m);
sinm2 = sinm*sinm;
sinm3 = sinm2*sinm;
sinm4 = sinm3*sinm;
sinm5 = sinm4*sinm;
sinm6 = sinm5*sinm;
sinm7 = sinm6*sinm;
cosm  = cos(m);
```

// Power series algorithm taken from Ed Kopitskie/Van Dierendonck

```
// memo on initial  ICD-GPS200 algorithms

 sinv  =

        ef*(
                (1.0 + 3.0*e2 + 5.0*e4 + 7.0*e6)*sinm +
                (2.0*e + 4.0*e3 + 6.0*e5)*sinm*cosm -
                (9.0*e2/2.0 + 85.0*e4/3.0 + 581.0*e6/6.0)*sinm3 -
                (32.0*e3/3.0 + 52.0*e5)*sinm3*cosm  +
                (625.0*e4/24.0 + 29757.0*e6/120.0)*sinm5  +
                324.0*e5*sinm5*cosm/5.0 - 117649.0*e6*sinm7/720.0
        );

cosv  =
        (
                cosm - (2.0*e + 8.0*e3 + 18.0*e5)*sinm2 -
                (9.0*e2 + 25.0*e4 + 49.0*e6)*sinm2*cosm/2.0 +
                (32.0*e3 + 234.0*e5)*sinm4/3.0 +
                (625.0*e4 + 3626.0*e6)*sinm4*cosm/24.0 -
                324.0*e5*sinm6/5.0  -
                117649.0*e6*sinm6*cosm/720.0
        );

//compute the position and velocity of the space craft

        r  = a*(1.0 - e)*(1.0 + e)/(1.0+e*cosv);        // focal radius vector
        vr  = n*a*e*sinv/ef;                            // radial plane velocity
        vd  = n*a*(1.0 + e*cosv)/ef;                    // down track velocity

//transform these parameters into the ECI

        cosu = cosv*cosw - sinv*sinw;
        sinu = cosv*sinw + sinv*cosw;

// ECI

        P^ECI[x] = r*(cosΩ_O *cosu - sinΩ_O *sinu*cosi);
        P^ECI [y] = r*(sinΩ_O *cosu + cosΩ_O *sinu*cosi);
        P^ECI [z] = r*sinu*sini;

        V^ECI [x] = vr*(cosΩ_O *cosu - sinΩ_O *sinu*cosi)
                                - vd*(cosΩ_O *sinu + sinΩ_O *cosu*cosi);
        V^ECI [y] = vr*(sinΩ_O *cosu + cosΩ_O *sinu*cosi)
                                - vd*(sinΩ_O *sinu - cosΩ_O *cosu*cosi);
        V^ECI [z] = vr*sinu*sini  + vd*cosu*sini;
```

The Transformation from the ECI to the Keplerian frame is given by;

$h = P^{ECI} \times V^{ECI}$

$p \cdot v = P^{ECI} \bullet V^{ECI}$ (scalar product)
$r = | P^{ECI} |$
$v = |V^{ECI}|$
$temp = h \times V^{ECI}$

```
if( |h| < 1.0 ) {
        Tp      = 0;
        a       = 0;
        i       = 0;
        e       = 0;
        Ω_O     = 0;
        ω       = 0;
    return;
}
```

$h = h/|h|;$
$a = 1.0/(2.0/r - v*v/\mu);$ // semi major axis
$e = sqrt((1.0 - r/a) (1.0 - r/a) + (p \cdot v)^2 /(\mu a));$ // eccen.
$i = \cos^{-1}(h_Z);$ // inclination

```
if(  |i| < 0.0000001 ) {
        Ω_O  = 0.0;
 } else {
        Ω_O  = atan2(h_X,-h_Y);          // longitude of ascending
node
 }
```

$t_1 = v^2/\mu - 1.0/r;$
$t_2 = p \cdot v /\mu;$

$$RI = \begin{vmatrix} 1 & 0 & 0 \\ 0 & \cos(i) & \sin(i) \\ 0 & -\sin(i) & \cos(i) \end{vmatrix}$$

$$\begin{vmatrix} \cos(\Omega_O) & \sin(\Omega_O) & 0 \end{vmatrix}$$

$$RL = \begin{matrix} -\sin(\Omega_O) & \cos(\Omega_O) & 0 \\ 0 & 0 & 1 \end{matrix}$$

$V_1 = t_1 P^{ECI} - t_2 * V^{ECI}$ // work (temp) vector
$V_2 = RL \cdot V_1$ (matrix multiplication)
$V_1 = RI \cdot V_2$ (matrix multiplication)

ω = atan2(v1[y],v1[x]); // argument of perigee

// Compute the time of perigee for this orbit

$n = \sqrt{\mu/a^3}$; // mean motion

$$RW = \begin{vmatrix} \cos(\omega) & \sin(\omega) & 0 \\ -\sin(\omega) & \cos(\omega) & 0 \\ 0 & 0 & 1 \end{vmatrix}$$

$M = RI \cdot RL$ (matrix multiply)

$C^P_I = RW \cdot M$ (matrix multiply)

$Lx = C^P_I \cdot P^{ECI}$ (matrix multiply)

dnom = (Lx[0] + (a)(e))$\sqrt{1.0 - e^2}$;

E = atan2(Lx[1],dnom);
Tp = Δt - (E - esin(E))/n; // time of perigee

// Catch roundoff

```
if( fabs(Ω_O ) < 0.0000001) {
        Ω_O  = 0;
}
if( fabs(i) < 0.0000001) {
        i = 0;
}
if( fabs(Ω_O ) < 0.0000001) {
        Ω_O  = 0;
}
```

I.1.2 Terrestrial Script Maneuver Implementation.

For terrestrial maneuvers mechanized through the maneuver selection keypad, Tapestry uses the following design philosophy whenever possible:

(a) All maneuvers are built up from an analytic step-jerk representation in both the linear and angular regimes. This feature will be important in high dynamic applications in which code and carrier loop bandwidths/orders are selected based upon specific dynamic assumptions. The alternative use of step acceleration would severely stress not only a second but third order PLL or AFC loops. Step jerk provides a smooth and differentiable acceleration input into the receiver tracking loops.

The step jerk is specified in the wander azimuth frame and can be analytically integrated into acceleration and velocity. As described below, position (obtained from a direction cosine representation) - is not (in general) analytic in an ellipsoidal earth and is obtained by a numerical technique.

Angular motion is computed independent from the linear motion and is also based upon a step angular jerk. This allows the Tapestry user to precisely control the dynamics injected into the receiver under test resulting from motion of the antenna lever arm.

(b) All dynamic parameters (*jerk, acceleration, and velocity*) associated with the maneuvers are specified analytically in the local tangent wander frame with the exception of vehicle position. To compute position, an auto-step sixth order Romberg extrapolation/integration technique was used. This technique, developed by Bolrich and Stoer, results in an algorithm error of less than 0.01 PPM in those cases which can be checked analytically.

The use of analytic representations for the maneuvers allows the Tapestry Vehicle Motion Generator to achieve precise control of not only the linear dynamic state of the vehicle but the angular state as well. In the cases of extreme dynamics (in excess of 5 G's) the auto-step feature allows the Tapestry to partition the maneuver into multiple segments without any sacrifice in accuracy. In such high G situations, the Romberg integrator typically executes 36 function evaluations whereas a fourth order Runge-Kutta would perform only 4 evaluations. This increased operation count, while sacrificing some throughput, provides for an extreme amount of fidelity in the truth data.

(c) In order to synchronize the Tapestry RF outputs with the truth data a dynamic parameter readjustment is often performed. This synchronization is required because the Tapestry generates RF signals derived from

updates to a numeric-controlled-oscillator (NCO) at 500 Hz based on interpolation of 10 Hz polynomial parameters. In addition, the user entered vehicle maneuvers are synchronized such that they always start and stop on an NCO reset epoch. This precise synchronization assures the Tapestry user that no maneuver will persist (in the RF channel) past the time at which the maneuver was supposed to terminate. This is extremely important in situations where the Tapestry provides test data which is to be synchronized with the RF signals. The synchronization of maneuver start and stop times to 10 Hz boundaries often requires the Tapestry to change the parameters (typically the applied jerk) associated with a maneuver. These changes in the values of dynamic parameters are required such that the terminal conditions desired by the user can be achieved.

Note: Vehicle parameter readjustment is in a direction that always decreases the dynamic stress rather than increases it. The operator can be assured that the specified vehicle "maximum" parameters are never exceeded during a maneuver.

All maneuvers are passed through a pre-processing function in which:

(a). The user parameter inputs are validated against the vehicle dynamic constraints.
(b). The maneuver is decomposed into its constituent jerk primitives. A jerk primitive is defined as a time-slice in which the applied jerk is constant. The constant value can be either positive, negative or zero.
(c). The time intervals for each jerk primitive are computed - consistent with the alignment of NCO and maneuver epochs discussed previously. Jerk magnitude may be re-adjusted if required.

Given a decomposed jerk primitive set, and times, the vehicle (wander-azimuth) navigation state is computed schematically as:

J_I = I-th jerk primitive (Wander-NED)
A_I = I-th Acceleration increment (Wander-NED)
V_I = I-th Velocity increment (Wander-NED)
C^E_P = Position direction cosines relating Platform to ECEF

The differential equations for the update of C^E_P is given by:

for(all jerk primitives) {

J_I = re-scaled jerk primitive (I)

$A_I = A_{(I-1)} + \int J_I(t)dt$

$$V_I = V_{I-1} + \iint J_I(t)dt + \int A_{(I-1)}dt$$

$$d/dt\,(C^E_P) = C^E_P\,(\omega^P_{EP}\,x)$$

where the skew symmetric matrix $(\omega^P_{EP}\,x) = \begin{vmatrix} 0 & -\omega_D & \omega_E \\ & 0 & -\omega_N \\ & & 0 \end{vmatrix}$

where

R_M = radius prime meridian (function of C^E_P)
R_V = radius prime vertical (function of C^E_P)
α = wander angle(function of C^E_P)

$$\omega^N_{EP} = V_E\,(\cos(\alpha)\cos(\alpha)/R_V + \sin(\alpha)\sin(\alpha)/R_M) + V_N(\cos(\alpha)\sin(\alpha)\,(1/R_V - 1/R_M)$$

$$\omega^N_{EP} = -V_N\,(\sin(\alpha)\sin(\alpha)/R_V + \cos(\alpha)\cos(\alpha)/R_M) - V_E(\cos(\alpha)\sin(\alpha)\,(1/R_V - 1/R_M)$$

$$\omega^D_{EP} = 0.0$$

} (Increment I for next primitive)

Unfortunately, these differential equations are coupled and non-linear. As mentioned previously, the solution to the position direction cosine update is obtained using a numerical integration routine coded from the IEEE proceeding by Bolrich and Stoer for solving simultaneous coupled non-linear equations. The accuracy of this technique is superb exceeding most 8-th order algorithms.

Overview of modeled maneuvers.

The following presents a brief description of the analytics associated with the maneuver generation algorithm for the Tapestry.

The Cruise Maneuver

The **Cruise** propagates the current vehicle navigation state (based upon the terminal conditions of the maneuver immediately preceding the selection of the **Cruise**) via great-circle navigation for a user specified time interval.

Because of the great circle navigation, a long duration **Cruise**, with a modest (positive) east velocity, would result in a clockwise rotation of the terminal vehicle heading relative to the heading at the initiation of the **Cruise**. For

example, a vehicle at an initial latitude of 36^0N and longitude of 118^0W, moving east at 1000 meters/second for 30 seconds, would have a heading change of about 1.3^0 (south) relative to the initial heading of 90^0.

With the exception of the effects upon vehicle heading described above, the **Cruise** maneuver leaves all other vehicle dynamic conditions (altitude, speed, and attitude) invariant.

The parameter readjustment process quantizes the maneuver duration to lie on a 10 Hz epoch.

The Change Speed Maneuver

Change Speed causes the user vehicle to accelerate (or decelerate) from the terminal speed of the preceding maneuver to the speed selected by the operator. The speed is changed via a three segment process. For example, a speed change from 0.0 meters/second to 100 meters/sec at 10 meters/sec^2 with a maximum linear jerk limit of 100 meters/sec^3 (specified via the Vehicle Dynamic Constraints menu Item) would be executed as follows:

1) The jerk up phase: This phase consists of a positive applied constant jerk, in our example 100 meters/sec^3, for a period sufficient to achieve the desired vehicle acceleration. After achieving the desired acceleration the jerk is instantaneously brought to zero. During this period, the vehicle velocity will increase analytically and the position direction cosine will be propagated in time numerically.
2) The constant acceleration phase: This phase will have zero vehicle jerk and constant acceleration. During this period the vehicle will continue to (analytically) increase in velocity with the position (numerically) propagated forward in time.
3) The jerk down phase: This phase will have an applied negative step jerk which will remove the constant vehicle acceleration. During this period the vehicle will continue to increase in velocity. At the termination of this phase, the time-rate-of-change of velocity will be zero and the final vehicle speed will be precisely as specified by the user. As in the other phases, the position is propagated forward in time.

The time intervals associated with the **Change Speed** maneuver are quantized to lie upon 10 Hz epochs. This may require the reduction of the applied vehicle jerk during phase 1 and phase 3.

In a situation where the specified vehicle acceleration is very large, it may be possible to achieve the desired terminal velocity without achieving the desired terminal acceleration. In this case the second phase is not executed. If the incremental speed change is very small relative to the allowed applied jerk, the

terminal velocity may be achieved in less than 0.2 seconds. This is an error condition and the maneuver is rejected.

The Accelerate maneuver

Accelerate results in a step jerk applied for an interval in order to reach the desired acceleration. During the jerk phase (positive/negative for a positive/negative acceleration change) the vehicle velocity changes quadratically and the position direction cosines are propagated numerically.

The **Accelerate** maneuver execution time is quantized to lie on a 10 Hz epoch. This, in general, results in a slight decrease of the maximum applied jerk.

The Turn Maneuver

Turn allows the user to change the heading of the simulated vehicle while precisely controlling the radial jerk, acceleration, and the presence of banking. The ground track of the **Turn** maneuver closely approximates a circular arc overlaid on the earth at some fixed altitude. The deviation from an exact circular arc is caused by the desired radial acceleration being achieved through the use of a ramp rather than a step.

Nominally the **Turn** is executed in three linear motion and seven angular motion phases. The linear and angular motions are performed simultaneously with the linear motion sequenced as follows:

1) The jerk to constant radial acceleration phase: This phase applies a constant radial jerk perpendicular to the instantaneous vehicle velocity. The jerk persists for an interval sufficient to achieve the acceleration specified by the user. This time interval is quantized to lie upon a 10 Hz NCO reset interval - this may result in a reduction of the applied radial jerk.

During this phase the vehicle speed is maintained as a constant with the instantaneous heading defined as:

$h(t) = \tan^{-1}(V(t)^E/V(t)^N)$; where V is the wander-azimuth locally level velocity at time (t) obtained from the double integration of the applied radial jerk (with the constraint of constant vehicle speed).

2) The constant radial acceleration phase: During this phase the vehicle executes a circular motion with the constant radial acceleration achieved during the previous jerk segment. The applied radial jerk during this phase is zero, however the tangential jerk is not zero (a certain level of jerk is required to swing the acceleration vector throughout its circular motion).

During this phase the vehicle velocity remains constant and the instantaneous heading is given by the same formula as in the first phase.

3) The jerk to zero radial acceleration phase: This phase is the inverse of the first phase and is required to remove the applied radial acceleration.

In addition to the linear dynamic-phases, the **Turn** causes the vehicle bank angle (roll) to change (if specified). The roll angle is computed automatically by setting the terminal value to:

$U = \tan^{-1}(A/G)$, where A is the applied radial acceleration and G is the acceleration due to gravity.

The vehicle is rolled through this bank angle in seven phases as follows:

1) The jerk to constant angular rate phase: During this phase the vehicle attitude is computed by dividing the interval (quantized) into two segments. During the first segment a positive (clockwise) angular jerk is applied (assuming a right-turn) and during the second phase a counter-clockwise angular jerk is applied. These two segments cause the vehicle to achieve a constant roll rate about the nose.

2) The constant angular rate phase: The vehicle continues to roll at a constant rate without applied acceleration or jerk.

3) The jerk to zero rate phase: This phase reverses the sequencing of phase (1) in order to remove the applied vehicle roll rate. At the termination of this phase the vehicle is banked at the desired bank-angle.

4) Constant bank phase: During this phase the vehicle is held at a constant bank angle. The angular rate, acceleration, and jerk are zero.

5), 6), 7). These phases are the inverse of, respectively, phases 3, 2, and 1.

At the termination of the seven attitude phases the vehicle is back in a level configuration. NOTE: To as high a degree as possible the attitude phases are synchronized with the linear phases such that the constant bank phase applies during the entire constant radial acceleration phase.

The **Turn** sequencing described above applies during the nominal **Turn** scenario. Deviations can result if the turn angle is very shallow or the applied acceleration is very large. In such situations the first process to occur is that the constant radial acceleration phase is removed and the applied jerk readjusted - if this suffices the maneuver is executed, otherwise an error window is presented. Similar conditions can occur such that the bank angle must be achieved without a constant angular rate phase.

The Pitch Maneuver

Pitch mechanization and sequencing is very similar to the **Turn** maneuver. This similarity is due to the **Pitch** *being* a **Turn** maneuver in the vertical (rather than the horizontal) plane. The **Pitch** can either be "up" or "down" depending upon the sense of the pitch angle entered by the user.

The instantaneous pitch angle is given by:

$h(t) = \tan^{-1}(-V(t)^D/V_{LEVEL})$, where V_{LEVEL} is the horizontal velocity of the vehicle and $-V^D$ the vertical velocity.

No other attitude axes are affected during the **Pitch** and the vehicle speed is maintained as a constant throughout the maneuver. As with the **Turn**, all segment times are quantized to 10 Hz epochs which may cause the readjustment of applied jerk during some of the phases.

The Climb Maneuver

The **Climb** maneuver is a compound maneuver constructed by serially sequencing a **Pitch**, a **Cruise** and another **Pitch**. During this the **Climb** is executed such that the vehicle maintains a constant speed.

The **Climb**, however, is more complicated than just splicing these three maneuvers together. The complication results from the terminal height at the end of the two **Pitch** phases not being given analytically (it actually is an ERF function which is only computed in tabular form). This lack of an analytical solution results in an iterative process being used, by which the Tapestry tries to determine the duration of the **Pitch** phases based upon a numerical integration of a transcendental function. This process causes the **Climb** to take very slightly more throughput than other maneuvers. At the completion of the iterative process the precise times are obtained for the two **Pitch** phases with the duration of the altitude change "tuned" by the **Cruise** phase (executed during a constant pitch angle which accumulates both an altitude as well as a down track change). The time associated with the cruise phase is, however, in general not quantized to the required 10 Hz time epochs. The subsequent quantization requires a readjustment of the duration of the **Pitch** phases (followed by a readjustment of the **Cruise** phase). This process is, in general, virtually never ending and the Tapestry compromises by executing the **Climb** in such a way so as to achieve a terminal altitude as *close as possible* to that specified by the user. Typically this error is within a few meters.

It is important to note however, that even though the terminal height achieved during the **Climb** is not *exactly* as specified by the user, it is, however, executed exactly! This means that whatever terminal condition was achieved, it was achieved precisely and all truth and GPS data are consistent. What this means is

that the dynamics and pseudoranges associated with the climb are precisely in agreement with the climb as executed by the Tapestry.

Problems with the **Climb** can result when either a very small altitude change is desired (100-200 meters) while pulling large G's, or when a steep pitch angle is specified with insufficient G's being specified.

The specification of a large pitch angle can cause the **Climb** to exceed the terminal height before the pitch can be completed. In general the software will do the best it can..

The Roll Maneuver

The **Roll** maneuver is essentially one half of the bank maneuver described in the **Turn** maneuver.

The **Roll** is executed in three phases as follows:

1) The constant step angular jerk phase: The maximum angular jerk specified in the vehicle definition menu is applied for a sufficient duration followed by a reversed step angular jerk over the same duration which results in a terminal angular rate applied to the vehicle as specified by the user as a maneuver parameter. The jerk duration's are quantized to 10 Hz epochs.
2) The constant rate phase: This phase propagates the vehicle roll angle with constant angular rate and zero angular acceleration and jerk. The duration of this phase is quantized to a 10 Hz epoch.
3) The jerk down phase: This phase is the inverse of phase 1 resulting in zero angular rate and a whole value roll angle equal to that specified by the user as a maneuver parameter.

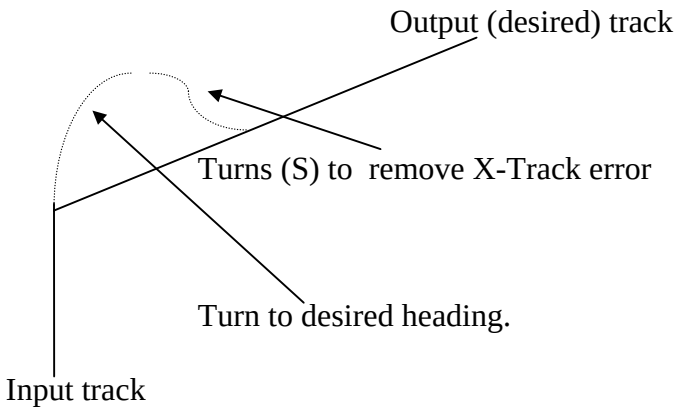
The **Roll** maneuver leaves the vehicle non level. Other maneuvers following the **Roll** will maintain the roll achieved during this maneuver. In some cases a non zero initial roll will cause some maneuvers to be unable to satisfy the user specified terminal conditions.

Waypoint Navigation

Navigating to a specified location seems simple enough, however in an actual vehicle this requires constant/periodic readjustment of vehicle heading to maintain "course". In our simulation environment we have selected to navigate to waypoints using great circle navigation. Great circle navigation is mechanized by maintaining a constant bearing in the wander azimuth frame. Interestingly it can be shown that the drift rate of the wander frame north reference relative to a true north frame (= Ve*TAN(LAT)/RADIUS) is just the correct heading adjustment required to maintain a great circle course.

In our implementation, when a waypoint is specified, the Tapestry computes the range and bearing to the waypoint based on the current position of the simulated vehicle. Most range and bearing algorithms use spherical trigonometry as their basis - this approximation causes a bearing error from a few tenths of degrees near the equator to several degrees near the poles. The Tapestry computes the great circle bearing by using the ECEF representation of the initial and terminal waypoint and forming the difference vector. This vector is transformed into a locally level tangent frame and the bearing angle computed. This bearing is always correct and can differ significantly from the initial bearing computed by a GPS receiver that uses a spherical trigonometry approach. Once the bearing is computed, the Tapestry initiates an appropriate turn to come to the correct course. If a single waypoint has been specified the Tapestry executes a direct-to waypoint.

In a direct-to implementation the initial turn is executed and a trim maneuver is computed. A trim maneuver is required because the Tapestry cannot instantaneously change its heading, thus a sizable cross track distance is accumulated - relative to the desired great circle course - and it must be removed by a succession of right/left turns. The following figure illustrates a direct-to waypoint starting at 0,0 lat/lon and steering to 1,1. The initial turn and following two "s" turns are evident.



I.1.3 Spaceborne Vehicle Motion Model

Gravity Model:

In spaceborne applications, the following source code segment provides the gravitational acceleration in the ECI coordinate frame:

```
#define WGS84_J2    1.082639e-3
#define WGS84_J3    -2.565e-6
#define WGS84_J4    -1.608e-6

        Bessels[0] = WGS84_J2;      default - user entered val overrides
        Bessels[1] = WGS84_J3;      default - user entered val overrides
        Bessels[2] = WGS84_J4;      default - user entered val overrides


        z  =  P_z^ECI;
        r  =  | P^ECI |
        r2 = r*r;
        Z  = z/r;
        Z2 = Z*Z;
        R  = GEOD_SMA_AXIS/r;
        R2 = R*R;
        R3 = R2*R;

        J       =  3Bessels[0]/2;
        H       =  5Bessels[1]/2;
        D       =-35Bessels[2]/8;

        P       = μ(    1.0 + J*R2*(1.0-5.0*Z2) + H*R3*(3.0 - 7.0*Z2)*Z +
                        D*R2*R2*(9.0*Z2*Z2 - 6.0*Z2 + 3.0/7.0)   );

        Gx^ECI  = -P^ECI[x]*P/r/r2;             // ECI gravity vector x component
        Gy^ECI  = -P^ECI[y]*P/r/r2;             // ECI gravity vector y component


        Gz^ECI = - μ z/r/r2
                        (
                                1.0
                                +  J*R2*(3.0-5.0*Z2)
                                +  H*R3*(6.0*z*z-7.0*z*z*z*Z2 - 3.0*r*r/5.0)/r
                                +  D*R3*r*(15.0/7.0 - 10.0*Z2 + 9.0*Z2*Z2)*z
                        );
```

Atmospheric Drag Model:

In spaceborne applications, an atmospheric drag model is provided. The model is of the form:

$$a_I = - C \mathbf{V}_I |V_I|$$

Where C is a constant specified by the user, $\mathbf{V}_I$ is the (vector) inertial velocity and a_I is the inertial acceleration. The drag model is used only for the state vector propagation that includes both the geoid model and thrusting. The drag model is disabled for Keplerian state vector propagation.

State Vector Propagation:

For spaceborne applications, the ECI position and velocity vector state equations are solved numerically using the Bolrich and Stoer differential equation package (section I.1.2). The state vector equations are:

$$\begin{aligned}
d/dt\,(P_X^{ECI}) &= V_X^{ECI}: \\
d/dt\,(P_Y^{ECI}) &= V_Y^{ECI}: \\
d/dt\,(P_Z^{ECI}) &= V_Z^{ECI}: \\
d/dt\,(V_X^{ECI}) &= A_X^{ECI}; \\
d/dt\,(V_Y^{ECI}) &= A_Y^{ECI}; \\
d/dt\,(V_Z^{ECI}) &= A_Z^{ECI};
\end{aligned}$$

$$\text{With } A^{ECI} = G^{ECI} + C^I_O T^O - D\ V^{ECI} \bullet V^{ECI}$$

where C^I_O are the direction cosines transforming O (DXR) to ECI, T^O is the Orbit frame thrust vector, and D is the composite drag coefficient.

I.1.4 GPS Satellite Models

The GPS satellite model is described in ICD_GPS_200 which is mechanized exactly in the simulator. In addition to the standard model the following perturbative effects can be applied:

I.1.5 Selective Availability

Effects from selective availability are modeled as a second order Gauss-Markov process as specified in the memorandum by Studenny and Van Grass, RTCA No. 96-93/SC159-420 which has evolved into the RTCA/DO-208 specified simulator model required for FAA certification testing.

The SA error is computed via the discrete time process:

$$dS/dt = \Phi S + \mathbf{U} w = \begin{vmatrix} \Phi_{11} & \Phi_{12} \\ \Phi_{21} & \Phi_{22} \end{vmatrix} \begin{vmatrix} R \\ V \end{vmatrix} + \begin{vmatrix} U_{11} & U_{12} \\ 0 & U_{22} \end{vmatrix} \begin{vmatrix} w_1 \\ w_2 \end{vmatrix}$$

where S is a column state vector [δR, δV] where δR and δV are the pseudorange and pseudorange-rate errors due to Selective Availability. Φij the state transition matrix elements and Uij the mapping of the white noise processes in state space.

The process noise matrix Q is specified by the four elements

$$Q = \begin{vmatrix} Q_{11} & Q_{12} \\ Q_{21} & Q_{22} \end{vmatrix}$$

which can be uniquely specified, as described in the above mentioned memorandum, by three constants:

$\beta = 1/\sqrt{2}$ the model damping ratio.

σ_r = the pseudorange error standard deviation specified in by the user,

τ = the model correlation time.

Typical values for selective availability are σ_r = 23 meters, τ = 120 seconds. The white noise process, $[w_1, w_2]$ are gaussian. In addition the user specifies the initial value for the applied pseudorange error.

I.1.6 UERE Effects

The generated pseudorange (and rates) are corrupted by un-modeled space, and control segment errors. These errors include un-modeled GPS clock errors, solar pressure on the GPS satellites, etc. The Tapestry incorporates these effects by providing a 2nd order Gauss-Markov model in which the user specifies the lumped range error standard deviation and effective model time constant. The mechanization of the G-M model is identical to that specified in section I.3.1. Typical model values, excluding SA effects which are treated separately, are 4-10 meters with a time constant on the order of several hours.

I.1.7 Troposphere Model

The Tapestry incorporates several Troposphere models. They are all based upon variations of the Chao model and usually only differ at low elevation angles. All models compute the error using four model parameters:

a) The static zenith delay error at sea level. This parameter is either entered by the user or selected from a table of built in models.
b) The user altitude. The Tapestry computes this automatically.
c) The scale height of the atmosphere. This is the denominator of an exponential term to account for decrease of atmosphere as the user altitude increases. This is an entered parameter or taken from a table of built in models.
d) The elevation angle, above the earth horizon, of the GPS satellite. The Tapestry computes this automatically.

The basic model is:

Tropo Delay = (ZenithDelay)/F(α))

The built in models define the following values for the parameters.

Standard Chao:
ZenithDelay: (VerticalDelay)exp(-altitude/ScaleFactor)
VerticalDelay: User entered, typically 2.10 meters.
ScaleFactor: User Entered, typically 7000.0 meters.
F(α): sin(α); α is elevation angle

Rockwell/Collins: (this is the model in the NavcorV, MicroTracker)
h: altitude in Kilometers
α: elevation angle

F(α): sin(α) + 0.000143/(tan(α)+0.0455)
if(h < 1 Km) {
ZenithDelay: 2.5119 - 0.3248*h/1000.0; - 0.022395*h*h.
} else if (h < 9 Km) {
ZenithDelay: -0.1191 + 2.2838 exp(0.1226(1-h))
} else{
ZenithDelay: 0.73736exp(0.1424(9-h))
}

Rockwell/Collins: (this is the model in the 12 Channel Zodiac chip set)
h: altitude in Kilometers
α: elevation angle

F(α): sin(α) + 0.026; if sin(α) > 0, otherwise 0.026
ZenithDelay: 2.4224 exp(-0.00013346*h) h > 0, 2.4224 otherwise.

Trimble TANS model

This model is identical to the Rockwell Jupiter Model.

Litton Aero Products (LAPD).

α: elevation angle
if(sin(α) < 0.026177) {
ZenithDelay: 0.837(VerticalDelay)exp(-altitude/ScaleFactor)
VerticalDelay: 2.208 meters
ScaleFactor: 6900.0 meters
F(α): 1.0
} else {
ZenithDelay: (VerticalDelay)exp(-altitude/ScaleFactor)
VerticalDelay: 2.208 meters
ScaleFactor: 6900.0 meters
F(α): sin(α)
}

I.1.8 Ionosphere Model

There are three Ionosphere models in the Tapestry.

a) The standard model as described by Klobuchar and referred to herein as the ICD-GPS-200 model. This model is governed by the 6 parameters α_1,α_2,α_3, and β_1, β_2, β_3 which are described in the ICD specification. *Note that zero value for all of these parameters does not give a zero ionosphere* !
b) No Ionosphere model. In this case no Ionosphere effects are applied to the output pseudorange and carrier phase data.
c) Shell model. This model is used for spaceborne applications only. In the shell model the Ionospheric group delay (and phase advance) are computed from a circular shell starting at LowerRadius (user entered) meters from the earth center and ending at UpperRadius (user entered) from the earth center. The VerticalDelay (meters) is the total Ionospheric group delay with *NO SLANT RANGE EFFECTS* and is entered by the user (The Tapestry requires the user to do the TEC computation). Given the VerticalDelay, the Tapestry compensates the delay for shell-model slant range and applies the value accordingly based upon line-of-sight from the GPS to the orbiting host vehicle. The effects from the ICD-GPS-200 model are suppressed for spaceborne applications.