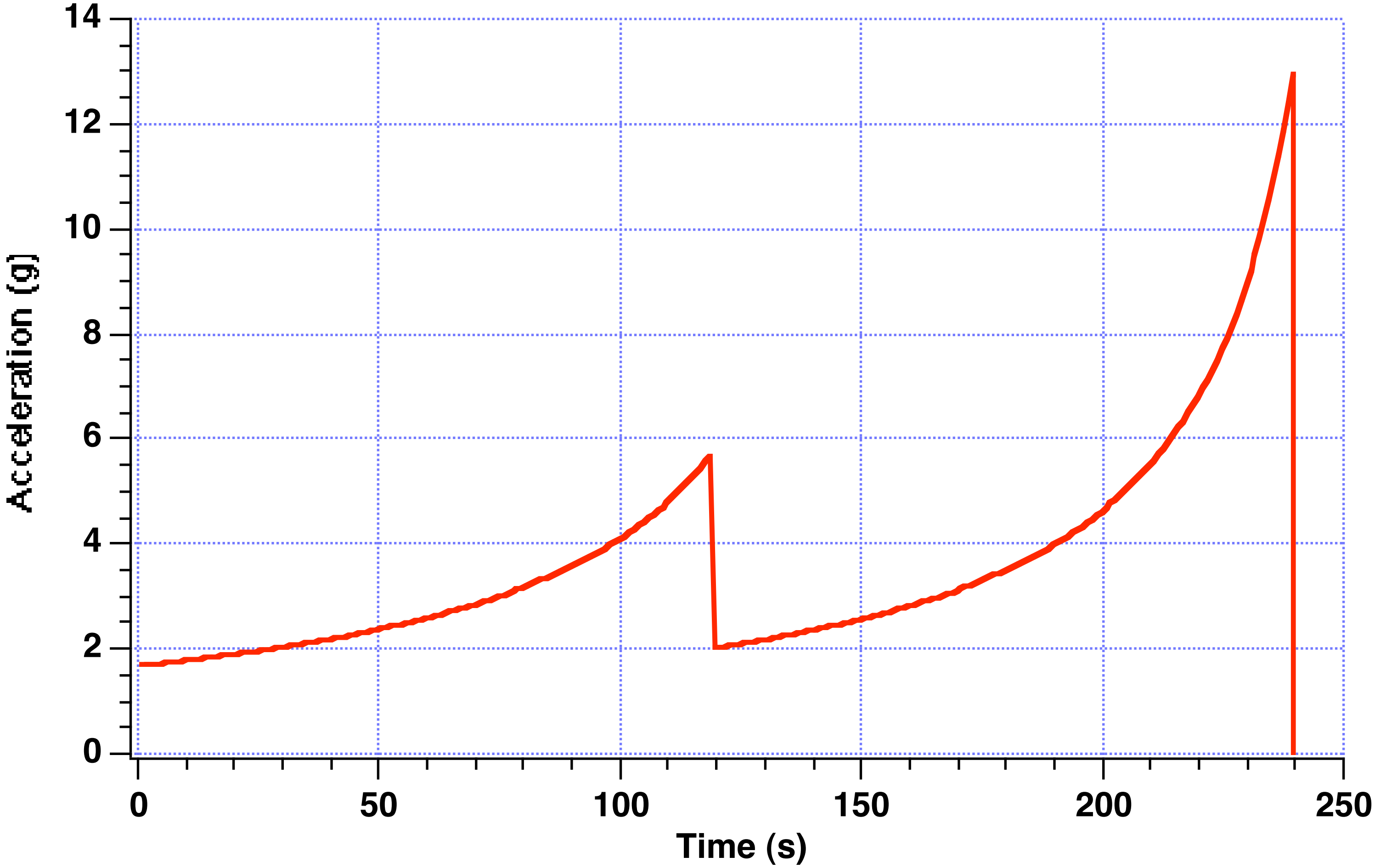
# Boost Phase Filtering Options: Is Simpler Better?

# Overview

- Simple ICBM model
    - Simple guidance equations for flat earth
- Range and angle measurement model
    - Creating pseudo measurements
- 2-State template based Kalman filter
    - Performance and robustness
- 3-State polynomial Kalman filter
    - Performance and comparison with 2-state filter
- Summary

# Simple ICBM Model

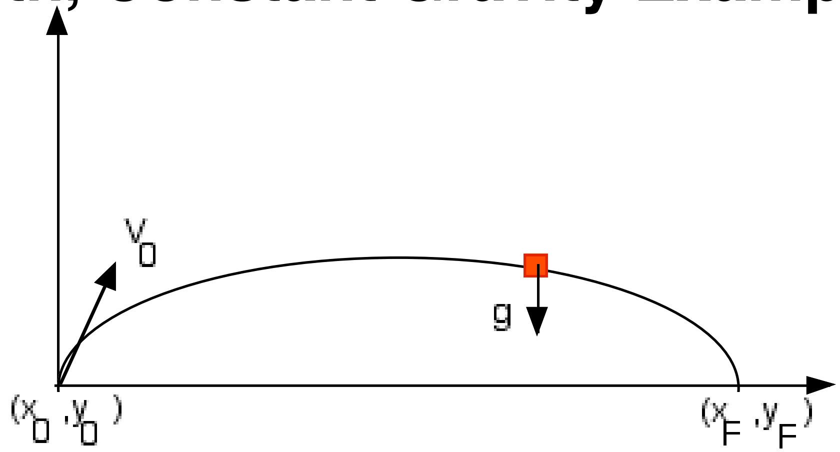
# Acceleration Profile of Generic Two-Stage APS Liquid ICBM



# Two Kalman Filter Design Possibilities

- Two-state template based Kalman filter
  - Magnitude of ICBM acceleration profile is known (template)
  - Estimate ICBM position and velocity based on range angle measurements
- General purpose three-state Kalman filter
  - Assume no a priori information is available
  - Estimate ICBM position, velocity and acceleration based on range and angle measurements only

# How ICBMs Guide to Their Intended Target (Flat Earth, Constant Gravity Example)




**Given Initial ICBM location $x_0$,$y_0$ and destination $x_F$,$y_F$ and desired arrival time $t_F$ we desire initial velocity vector required to hit target at desired arrival time**

**Future location of ICBM can be calculated from High School physics**

$$x_F = x_0 + \dot{x}_0 t_F$$

$$y_F = y_0 + \dot{y}_0 t_F - 0.5 g t_F^2$$

**Solve for initial velocity as if ICBM launched as cannon ball**

$$\dot{x}_0 = \frac{x_F - x_0}{t_F}$$

$$\dot{y}_0 = \frac{y_F - y_0 + 0.5 g t_F^2}{t_F}$$

**Flat earth solution to Lambert's problem**

# Lambert Guidance - 1

**At each instant of time compute desired (Lambert) velocity components**

$$t_{go} = t_F - t$$

$$V_{Lambert_x} = \frac{x_F - x}{t_{go}}$$

$$V_{Lambert_y} = \frac{y_F - y + 0.5 g t_{go}^2}{t_{go}}$$

**Calculate velocity to be gained (Lambert velocity minus current velocity)**

$$\Delta V_x = V_{Lambert_x} - V_x$$

$$\Delta V_y = V_{Lambert_y} - V_y$$

$$\Delta V = \sqrt{\Delta V_x^2 + \Delta V_y^2}$$

**In Lambert guidance we align ICBM thrust vector with velocity to be gained vector**

$$a_{T_x} = \frac{\Delta V_x}{\Delta V} a_T$$

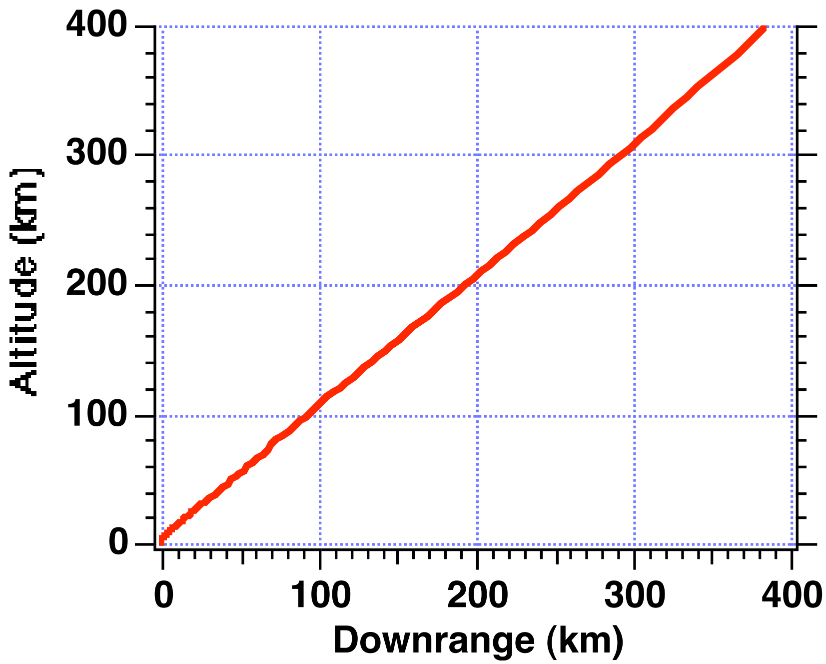
$$a_{T_y} = \frac{\Delta V_y}{\Delta V} a_T$$

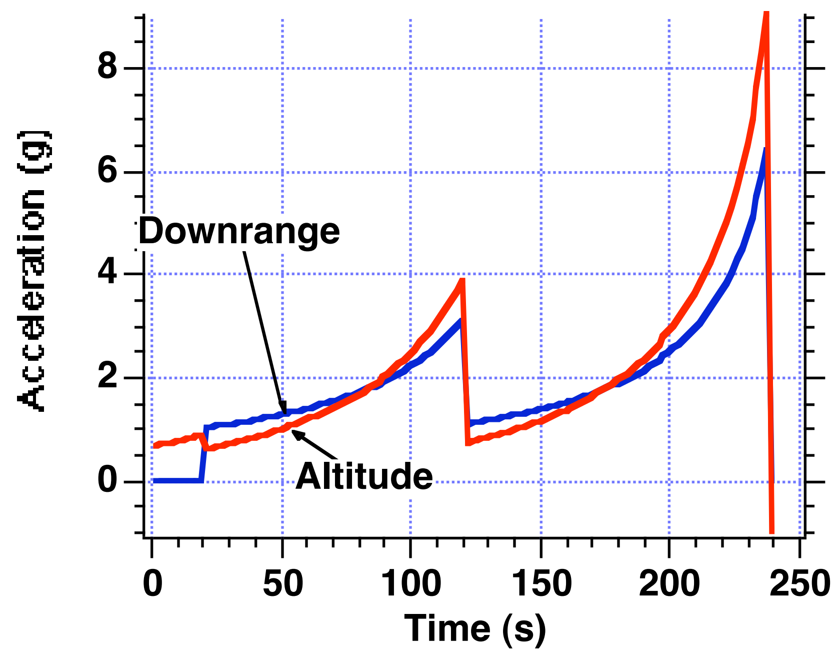
# Lambert Guidance - 2

**Where $a_T$ is the ICBM acceleration magnitude (Thrust/Weight)**

**We iterate at each guidance update until $\Delta V$ goes to zero and then we thrust terminate**

# Trajectory and Acceleration Profile Components of ICBM Boost Phase Portion of 5000 km Trajectory



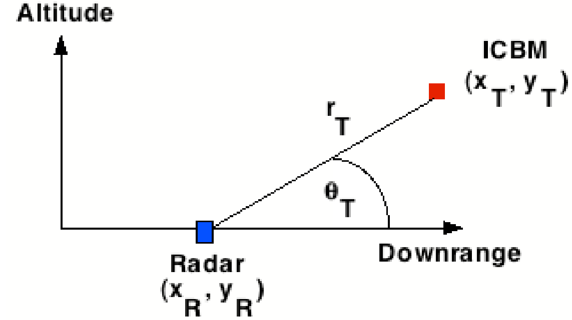
# Measurements and Pseudo Measurements

# Pseudo Measurement Equations

**Actual range and angle from radar to ICBM**

$$r_T = \sqrt{(x_T - x_R)^2 + (y_T - y_R)^2}$$

$$\theta_T = \tan^{-1}\left(\frac{y_T - y_R}{x_T - x_R}\right)$$




**In order to avoid building EKF we will consider pseudo position measurements**

$$x_T^* = r_T^* \cos\theta_T^* + x_R$$

$$y_T^* = r_T^* \sin\theta_T^* + y_R$$

**These pseudo measurements can serve as inputs to 2 decoupled linear Kalman filters**

**One can show (next 2 slides) pseudo measurement variances are given by**

$$\sigma_x^2 = \cos^2\theta_T \sigma_R^2 + r_T^2 \sin^2\theta_T \sigma_\theta^2$$

$$\sigma_y^2 = \sin^2\theta_T \sigma_R^2 + r_T^2 \cos^2\theta_T \sigma_\theta^2$$

# Deriving Variance For Pseudo Measurement Noise-1

**In Cartesian frame model of real world is linear but actual measurements are nonlinear with respect to states. However pseudo measurements are linearly related to states**

**Recall from previous slide**

$$x_T = r\cos\theta + x_R$$

$$y_T = r\sin\theta + y_R$$

**Find total differential from calculus**

$$\Delta x_T = \frac{\partial x_T}{\partial r}\Delta r + \frac{\partial x_T}{\partial \theta}\Delta\theta = \cos\theta\,\Delta r - r\sin\theta\,\Delta\theta$$

$$\Delta y_T = \frac{\partial y_T}{\partial r}\Delta r + \frac{\partial y_T}{\partial \theta}\Delta\theta = \sin\theta\,\Delta r + r\cos\theta\,\Delta\theta$$

**Square both equations**

$$\Delta x_T^2 = \cos^2\theta\,\Delta r^2 - 2r\sin\theta\cos\theta\,\Delta r\Delta\theta + r^2\sin^2\theta\,\Delta\theta^2$$

$$\Delta y_T^2 = \sin^2\theta\,\Delta r^2 + 2r\sin\theta\cos\theta\,\Delta r\Delta\theta + r^2\cos^2\theta\,\Delta\theta^2$$

# Deriving Variance For Pseudo Measurement Noise -2

**Taking expectations of both sides assuming range and angle measurements are not correlated**

$$E(\Delta x_T^2) = \cos^2\theta E(\Delta r^2) + r^2\sin^2\theta E(\Delta\theta^2)$$

$$E(\Delta y_T^2) = \sin^2\theta E(\Delta r^2) + r^2\cos^2\theta E(\Delta\theta^2)$$

**Since**

$$\sigma_{x_T}^2 = E(\Delta x_T^2)$$

$$\sigma_{y_T}^2 = E(\Delta y_T^2)$$

$$\sigma_r^2 = E(\Delta r^2)$$

$$\sigma_\theta^2 = E(\Delta\theta^2)$$

**We can say**

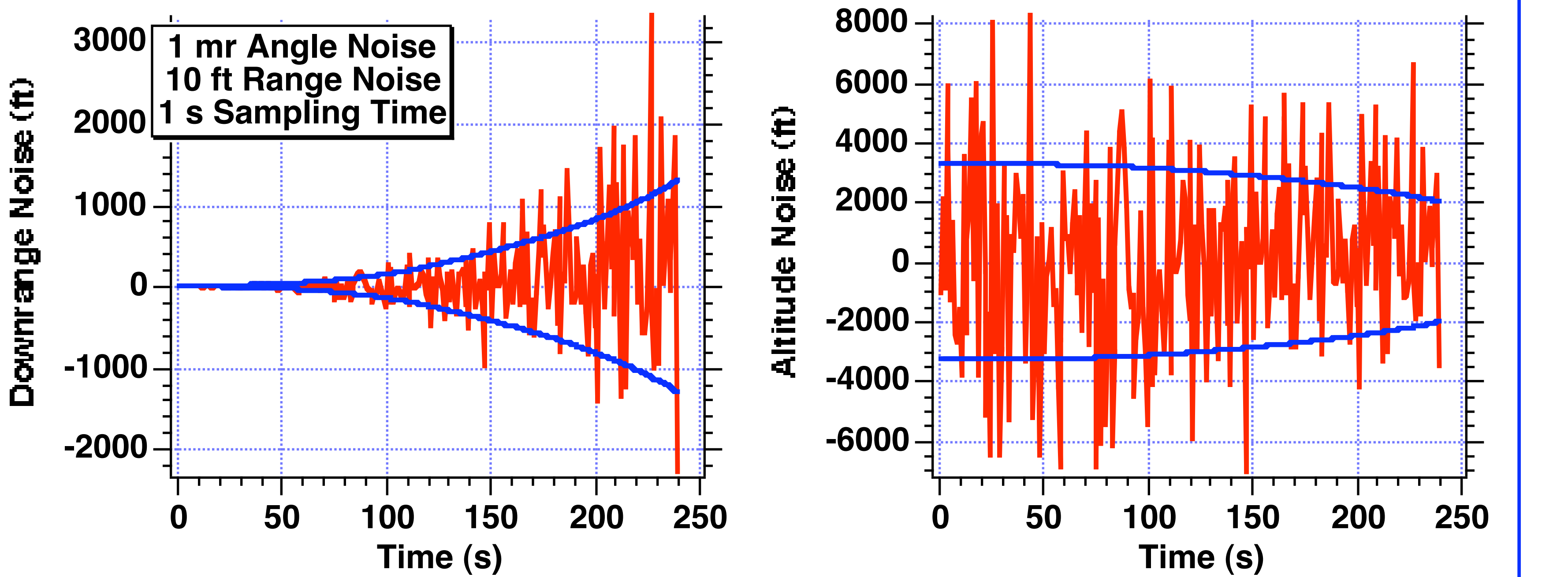
$$\sigma_{x_T}^2 = \cos^2\theta\sigma_r^2 + r^2\sin^2\theta\sigma_\theta^2$$

$$\sigma_{y_T}^2 = \sin^2\theta\sigma_r^2 + r^2\cos^2\theta\sigma_\theta^2$$

***We are pretending noise is on x and y rather than r and $\theta$**

**These pseudo measurement variances can be input to Ricatti equations of 2 decoupled linear Kalman filters**

# Effective Position Noise




**Theoretical calculations for pseudo measurement noise variance appear to be correct**

# 2-State Template Base Kalman Filter

# Two 2-State Template Based Decoupled Linear Polynomial Kalman Filters

## Downrange Filter

$$\mathrm{Res}_{x_k} = x^*_{T_k} - \hat{x}_{T_{k-1}} - \hat{\dot{x}}_{T_{k-1}} T_s - 0.5 a_{Tx_{k-1}} T_s^2$$

$$\hat{x}_{T_k} = \hat{x}_{T_{k-1}} + \hat{\dot{x}}_{T_{k-1}} T_s + 0.5 a_{Tx_{k-1}} T_s^2 + K_{1x_k} \mathrm{Res}_{x_k}$$

$$\hat{\dot{x}}_{T_k} = \hat{\dot{x}}_{T_{k-1}} + a_{Tx_{k-1}} T_s + K_{2x_k} \mathrm{Res}_{x_k}$$

Note that template implies we know magnitude and direction of $a_T$ perfectly

## Altitude Filter

$$\mathrm{Res}_{y_k} = y^*_{T_k} - \hat{y}_{T_{k-1}} - \hat{\dot{y}}_{T_{k-1}} T_s - 0.5 a_{Ty_{k-1}} T_s^2$$

$$\hat{y}_{T_k} = \hat{y}_{T_{k-1}} + \hat{\dot{y}}_{T_{k-1}} T_s + 0.5 a_{Ty_{k-1}} T_s^2 + K_{1y_k} \mathrm{Res}_{y_k}$$

$$\hat{\dot{y}}_{T_k} = \hat{\dot{y}}_{T_{k-1}} + a_{Ty_{k-1}} T_s + K_{2y_k} \mathrm{Res}_{y_k}$$

## Process Noise

$$\mathbf{Q}_k = \Phi_s \begin{bmatrix} \frac{T_s^3}{3} & \frac{T_s^2}{2} \\ \frac{T_s^2}{2} & T_s \end{bmatrix}$$

**We will be adjusting $\Phi_s$ to tune the filters**

# 2-State Template Based Decoupled Linear Polynomial Kalman Filters -1

```
GLOBAL DEFINE
          INCLUDE 'quickdraw.inc'
END
IMPLICIT REAL*8 (A-H)
IMPLICIT REAL*8 (O-Z)
REAL*8 M11,M12,M22,K1,K2
REAL*8 M11P,M12P,M22P,K1P,K2P
LOGICAL QBOOST,QGRAV
INTEGER STEP
OPEN(1,STATUS='UNKNOWN',FILE='DATFIL')
OPEN(2,STATUS='UNKNOWN',FILE='NOISEFIL')
OPEN(3,STATUS='UNKNOWN',FILE='COVFIL')
QBOOST=.TRUE.
QGRAV=.FALSE.
TF=1300.
RT1F=5000*3280.
RT2F=0.
GAMTDEG=89.9
VT=1.
RT1=0.*3280.
RT2=0.
XR=1000.*3280.
YR=0.
VT1=VT*COS(GAMTDEG/57.3)
VT2=VT*SIN(GAMTDEG/57.3)
G=32.2
H=.01
TS=1.
S=0.
SCOUNT=0.
SIGR=10.
SIGTH=.001
PHIN=100.
RT=SQRT((RT1-XR)**2+(RT2-YR)**2)
THET=ATAN2(RT2-YR,RT1-XR)
SIGRT1=SQRT((COS(THET)*SIGR)**2+(RT*SIN(THET)*SIGTH)**2)
SIGRT2=SQRT((SIN(THET)*SIGR)**2+(RT*COS(THET)*SIGTH)**2)
K1=0.
K2=0.
RT1H=0.
VT1H=0.
RT2H=0.
VT2H=0.
T=0.
```

**Desired Destination and Time of Arrival** (TF, RT1F, RT2F)

**Range and angle measurement errors** (SIGR, SIGTH)

**Filter process noise** (PHIN)

**Actual range and angle** (RT, THET)

**Standard deviation of pseudo measurements** (SIGRT1, SIGRT2)

**Filter initialization** (RT1H, VT1H, RT2H, VT2H)

# 2-State Template Based Decoupled Linear Polynomial Kalman Filters -2

**Initial covariance matrix**

**2nd order Runge-Kutta integration of ICBM for boost phase**

**Lambert guidance**

```
      P11=99999999999.
      P12=0.
      P22=99999999999.
      P11P=99999999999.
      P12P=0.
      P22P=99999999999.
      AX=0.
      AY=0.
      XN=0.
      ERR=0.
10    IF(T>240.)GOTO 999
      RT1OLD=RT1
      RT2OLD=RT2
      VT1OLD=VT1
      VT2OLD=VT2
      STEP=1
      GOTO 200
66    STEP=2
      RT1=RT1+H*VT1
      RT2=RT2+H*VT2
      VT1=VT1+H*AT1
      VT2=VT2+H*AT2
      T=T+H
      GOTO 200
55    RT1=(RT1OLD+RT1)/2+.5*H*VT1
      RT2=(RT2OLD+RT2)/2+.5*H*VT2
      VT1=(VT1OLD+VT1)/2+.5*H*AT1
      VT2=(VT2OLD+VT2)/2+.5*H*AT2
      IF(QBOOST)THEN
            TGOLAM=TF-T
            VXLAM=(RT1F-RT1)/TGOLAM
            VYLAM=(RT2F-RT2+16.1*TGOLAM*TGOLAM)/TGOLAM
            DELVX=VXLAM-VT1
            DELVY=VYLAM-VT2
            DELV=SQRT(DELVX**2+DELVY**2)
            IF(TRST>0..AND.DELV>10.)THEN
                  AX=AT*DELVX/DELV
                  AY=AT*DELVY/DELV
            ELSEIF(DELV<10.)THEN
                  TRST=0.
                  QBOOST=.FALSE.
                  AX=0.
                  AY=0.
                  VT1OLD=VXLAM
                  VT2OLD=VYLAM
```

# 2-State Template Based Decoupled Linear Polynomial Kalman Filters -3

```
ELSE
                              QBOOST=.FALSE.
                              AX=0.
                              AY=0.
                              WRITE(9,*)DELV
                              PAUSE
              ENDIF
ENDIF
IF(T<20.)THEN
              AX=0.
              AY=AT
ENDIF
SCOUNT=SCOUNT+H
IF(SCOUNT.LT.(TS-.00001))GOTO 10
SCOUNT=0.
XN=XN+1.
XK1=2.*(2.*XN-1.)/(XN*(XN+1.))
XK2=6./(XN*(XN+1.)*TS)
TS2=TS*TS
TS3=TS2*TS
TS4=TS3*TS
TS5=TS4*TS
RT=SQRT((RT1-XR)**2+(RT2-YR)**2)
THET=ATAN2(RT2-YR,RT1-XR)
SIGRT1=SQRT((COS(THET)*SIGR)**2+(RT*SIN(THET)*SIGTH)**2)
SIGRT2=SQRT((SIN(THET)*SIGR)**2+(RT*COS(THET)*SIGTH)**2)
SIGN2=SIGRT1*SIGRT1
M11=P11+2.*TS*P12+TS2*P22+TS3*PHIN/3.
M12=P12+TS*P22+.5*PHIN*TS2
M22=P22+PHIN*TS
K1=M11/(M11+SIGN2)
K2=M12/(M11+SIGN2)
P11=(1.-K1)*M11
P12=(1.-K1)*M12
P22=-K2*M12+M22
```

**Go straight up for first 20 s**

**Least squares filter gains**

**True angle and range**

**Pseudo measurement standard deviations**

**Ricatti equations for downrange filter**

# 2-State Template Based Decoupled Linear Polynomial Kalman Filters -4

```
SIGN2P=SIGRT2*SIGRT2
M11P=P11P+2.*TS*P12P+TS2*P22P+TS3*PHIN/3.
M12P=P12P+TS*P22P+.5*PHIN*TS2
M22P=P22P+PHIN*TS
K1P=M11P/(M11P+SIGN2P)
K2P=M12P/(M11P+SIGN2P)
P11P=(1.-K1P)*M11P
P12P=(1.-K1P)*M12P
P22P=-K2P*M12P+M22P
CALL GAUSS(THETNOISE,SIGTH)
CALL GAUSS(RTNOISE,SIGR)
THETMEAS=THET+THETNOISE
RTMEAS=RT+RTNOISE
RT1S=RTMEAS*COS(THETMEAS)+XR
RT2S=RTMEAS*SIN(THETMEAS)+YR
IF(XN<10.)THEN
        XK1PZ=XK1
        XK2PZ=XK2
ELSE
        XK1PZ=K1
        XK2PZ=K2
ENDIF
IF(.NOT.QGRAV)THEN
        AT1H=AT1*(1+ERR)
        AT2H=AT2*(1+ERR)
ELSE
        IF(XN<10.)THEN
                AT1H=AT1
                AT2H=AT2
        ELSE
                VTH=SQRT(VT1H**2+VT2H**2)
                ATH=AT*(1.+ERR)
                AT1H=ATH*VT1H/VTH
                AT2H=ATH*VT2H/VTH-G
        ENDIF
ENDIF
```

**Ricatti equations for altitude filter** (SIGN2P … P22P)

**Actual angle and range measurements** (CALL GAUSS … RTMEAS)

**Pseudo measurements** (RT1S, RT2S)

**Use least squares gains for first 10 measurements** (IF(XN<10.) … ENDIF)

**Know direction of acceleration** (AT1H, AT2H)

**Gravity turn assumption for acceleration vector** (VTH … AT2H)

# 2-State Template Based Decoupled Linear Polynomial Kalman Filters -5

```
      RESX=RT1S-RT1H-TS*VT1H-.5*TS2*AT1H
      RT1H=XK1PZ*RESX+RT1H+TS*VT1H+.5*TS2*AT1H
      VT1H=XK2PZ*RESX+VT1H+TS*AT1H
      IF(XN<10.)THEN
            XK1PPZ=XK1
            XK2PPZ=XK2
      ELSE
            XK1PPZ=K1P
            XK2PPZ=K2P
      ENDIF
      RESY=RT2S-RT2H-TS*VT2H-.5*TS2*AT2H
      RT2H=XK1PPZ*RESY+RT2H+TS*VT2H+.5*TS2*AT2H
      VT2H=XK2PPZ*RESY+VT2H+TS*AT2H
      RT1KM=RT1/3280.
      RT2KM=RT2/3280.
      RT1NOISE=RT1-RT1S
      RT2NOISE=RT2-RT2S
      ERRRT1=RT1-RT1H
      ERRVT1=VT1-VT1H
      SP11=SQRT(P11)
      SP22=SQRT(P22)
      ERRRT2=RT2-RT2H
      ERRVT2=VT2-VT2H
      SP11P=SQRT(P11P)
      SP22P=SQRT(P22P)
      WRITE(9,*)T,RT1KM,RT2KM,VT1,VT1H,VT2,VT2H,AT1/G,AT1H/G,
     1      AT2/G,AT2H/G
      WRITE(1,*)T,RT1KM,RT2KM,VT1,VT1H,VT2,VT2H,AT1/G,AT1H/G,
     1      AT2/G,AT2H/G
      WRITE(2,*)T,RT1NOISE,SIGRT1,-SIGRT1,RT2NOISE,SIGRT2,-SIGRT2
      WRITE(3,*)T,ERRRT1,SP11,-SP11,ERRVT1,SP22,-SP22,ERRRT2,SP11P,
     1      -SP11P,ERRVT2,SP22P,-SP22P
      GOTO 10
200   CONTINUE
```

**Downrange Kalman filter** (lines RESX through VT1H)

**Use least squares gains for first 10 measurements** (IF ... ENDIF block)

**Altitude Kalman filter** (lines RESY through VT2H)

**Actual and theoretical errors in estimates** (lines RT1NOISE through SP22P)

# 2-State Template Based Decoupled Linear Polynomial Kalman Filters -6

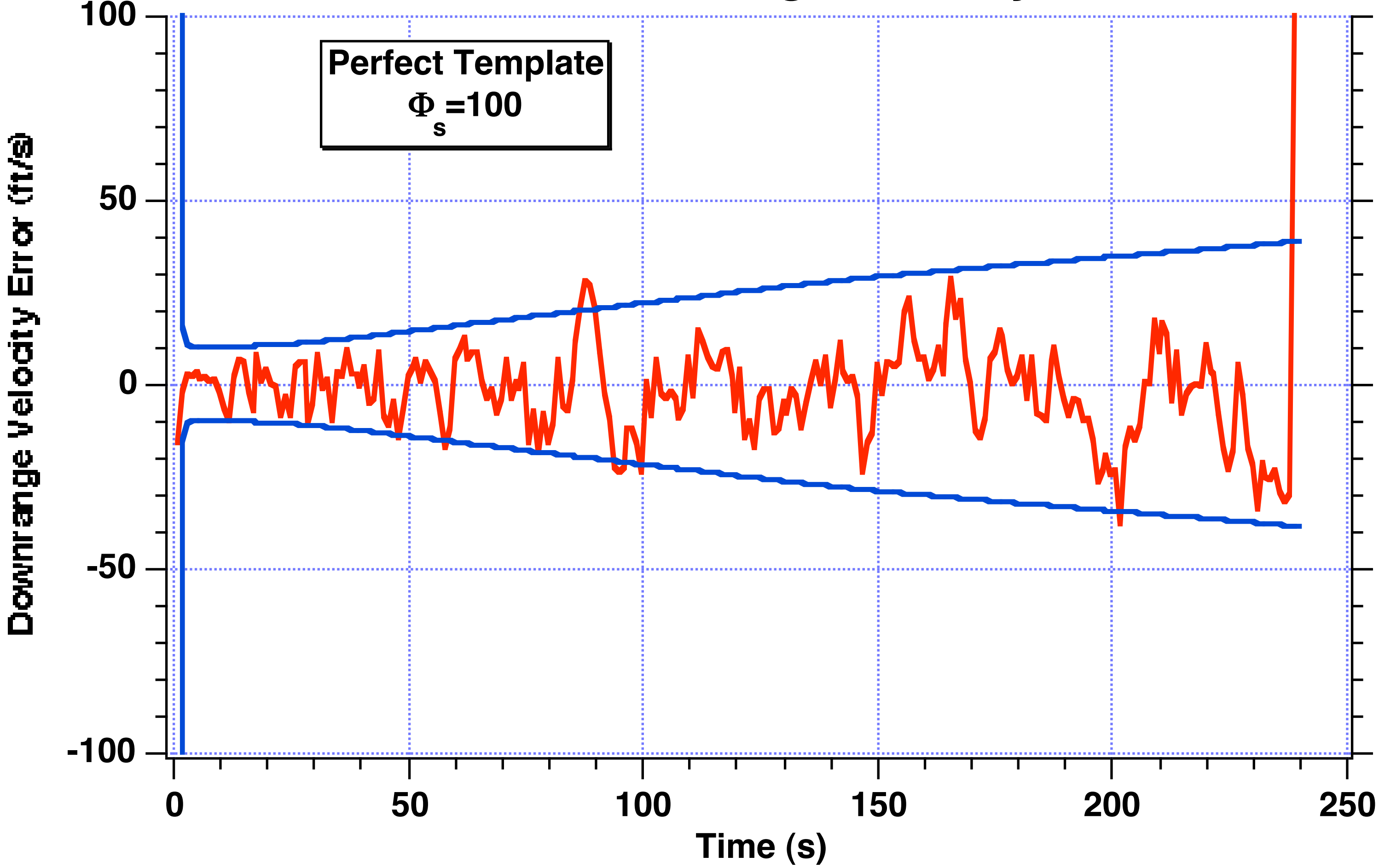
```
        IF(T<120.)THEN
                WGT=-2622*T+440660.
                TRST=725850.
        ELSEIF(T<240.)THEN
                WGT=-642.*T+168120.
                TRST=182250.
        ELSE
                WGT=5500.
                TRST=0.
        ENDIF
        VT=SQRT(VT1**2+VT2**2)
        GAM=ATAN2(VT2,VT1)
        AT=G*TRST/WGT
        AT1=AX
        AT2=-G+AY
        IF(STEP-1)66,66,55
999     CONTINUE
        RT1KM=RT1/3280.
        RT2KM=RT2/3280.
        WRITE(9,*)T,RT1KM,RT2KM,VT1,VT1H,VT2,VT2H,AT1/G,AT1H/G,
  1             AT2/G,AT2H/G
        WRITE(1,*)T,RT1KM,RT2KM,VT1,VT1H,VT2,VT2H,AT1/G,AT1H/G,
  1             AT2/G,AT2H/G
        PAUSE
        CLOSE(1)
        CLOSE(2)
        CLOSE(3)
        END
```

**ICBM** (IF/ELSEIF/ELSE block)

**Acceleration** (AT, AT1, AT2)

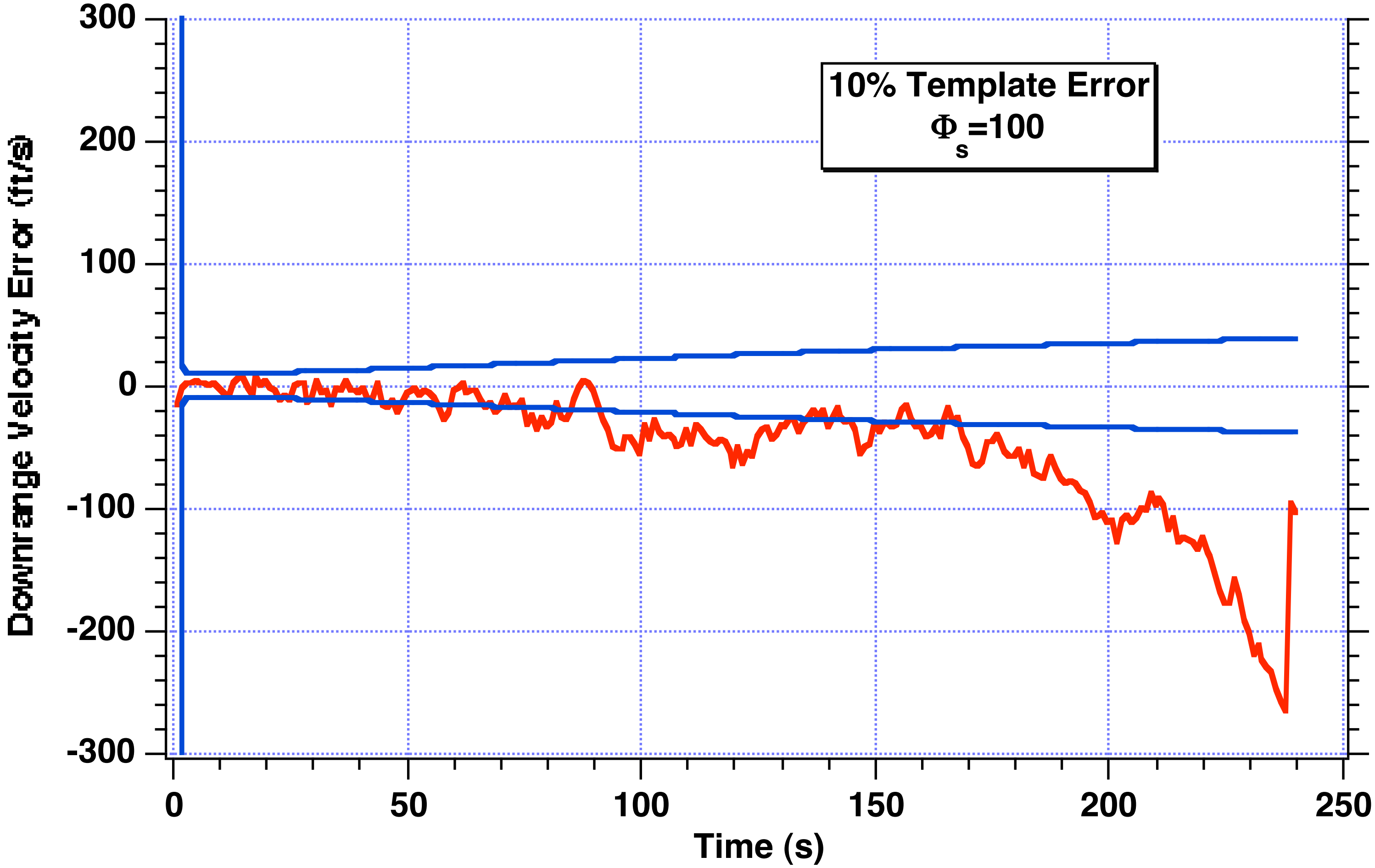
**Differential equations** (IF(T<120.) through IF(STEP-1))

# With Perfect Acceleration Template Two-State Kalman Filter Yields Consistent Downrange Velocity Estimates

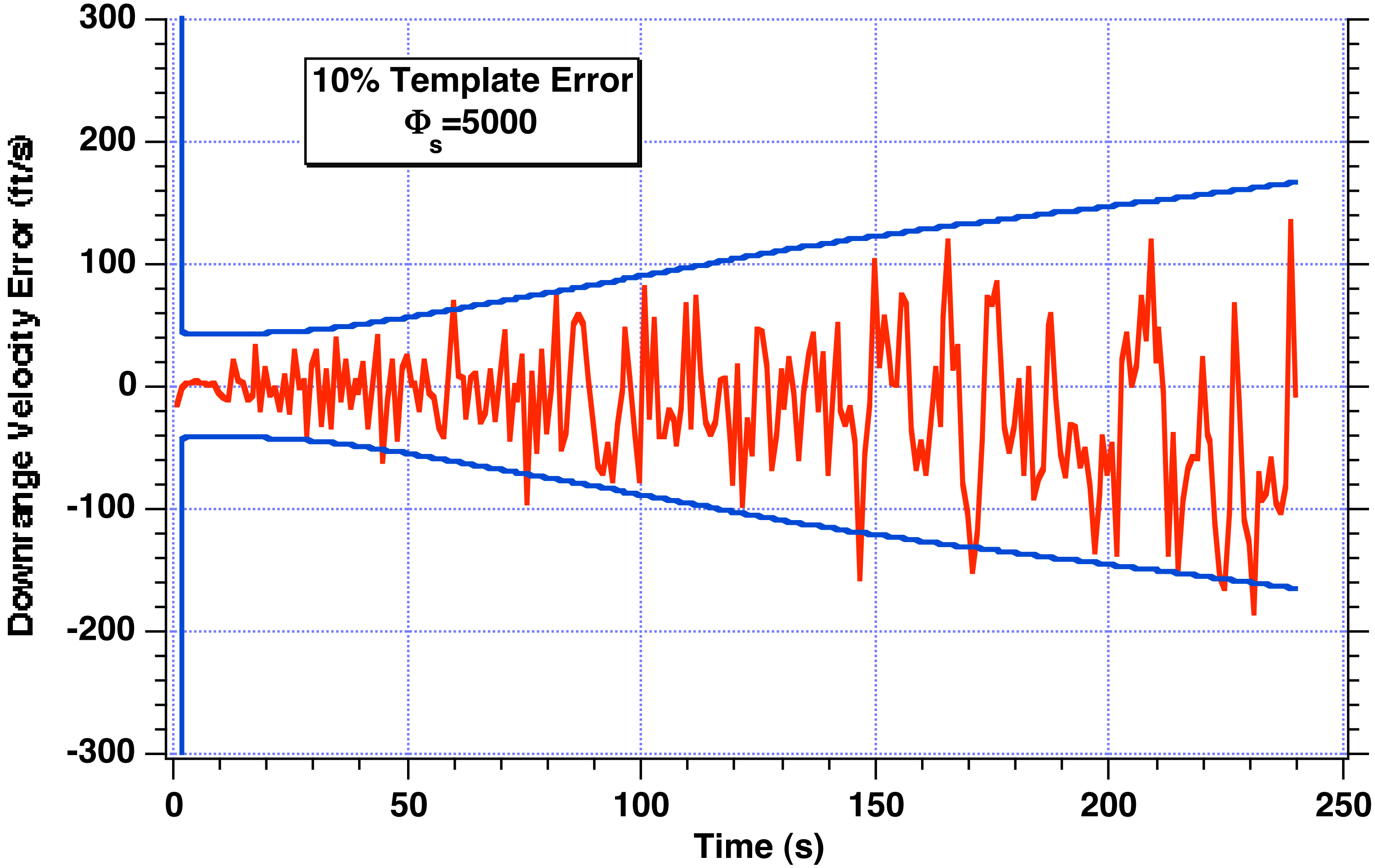



Don't forget we need to know target acceleration magnitude and direction for filter to work

# 10% Acceleration Errors Cause Two-State Template Based Kalman Filter to Diverge



# More Process Noise is Required by Two-State Kalman Filter to Eliminate Divergence When There is Acceleration Error




10% template error means that there is a 10% error in knowledge of ICBM acceleration magnitude

# Gravity Turn Assumption For Template Based Kalman Filter

**Assume we know magnitude of acceleration perfectly but not its direction**

**Gravity turn assumption***

$$a_{T_x} = a_T \frac{\dot{x}_T}{V_T}$$

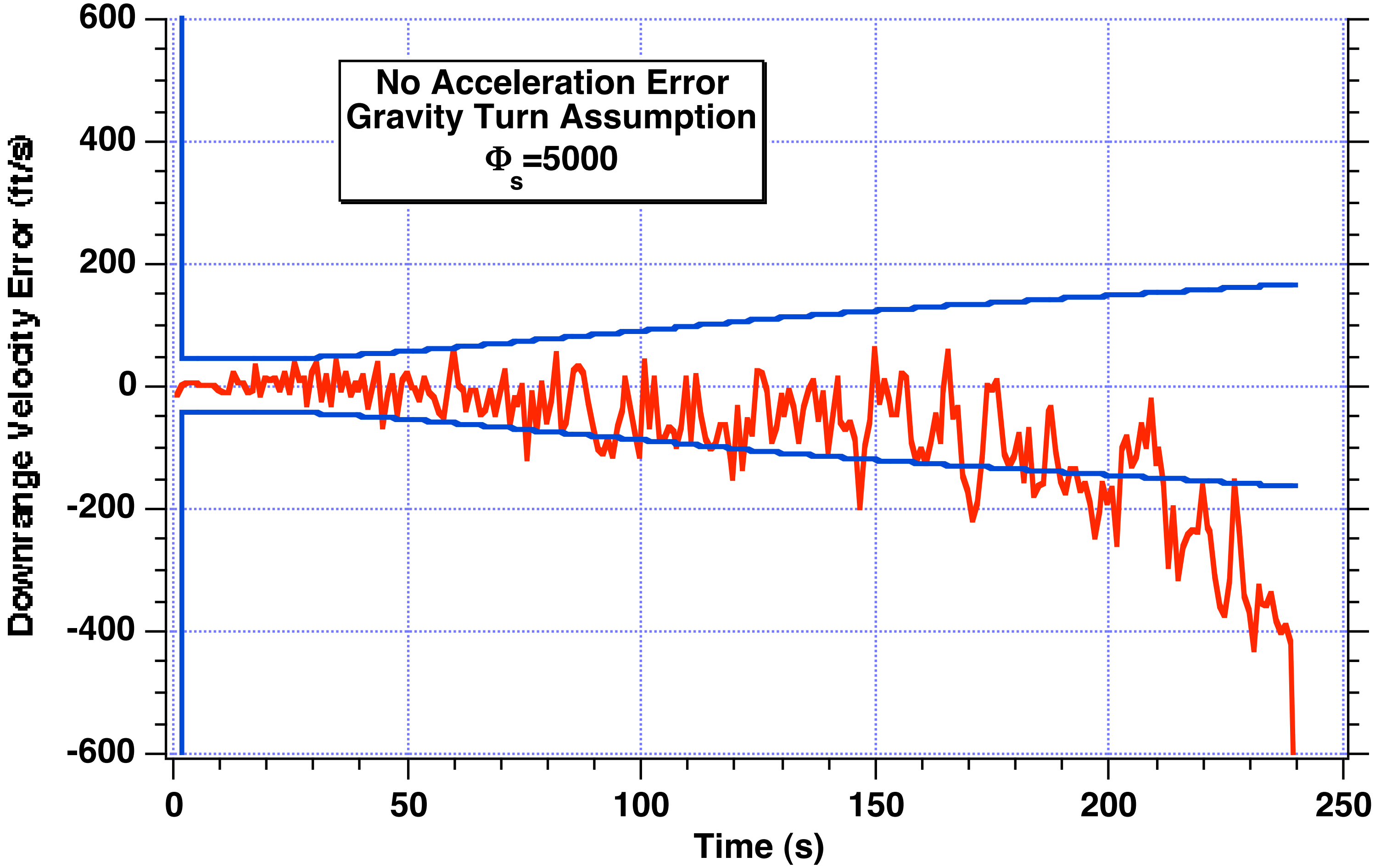
$$a_{T_y} = a_T \frac{\dot{y}_T}{V_T}$$

**Where $a_T$ is acceleration magnitude and $V_T$ is total velocity or**

$$V_T = \sqrt{\dot{x}_T^2 + \dot{y}_T^2}$$

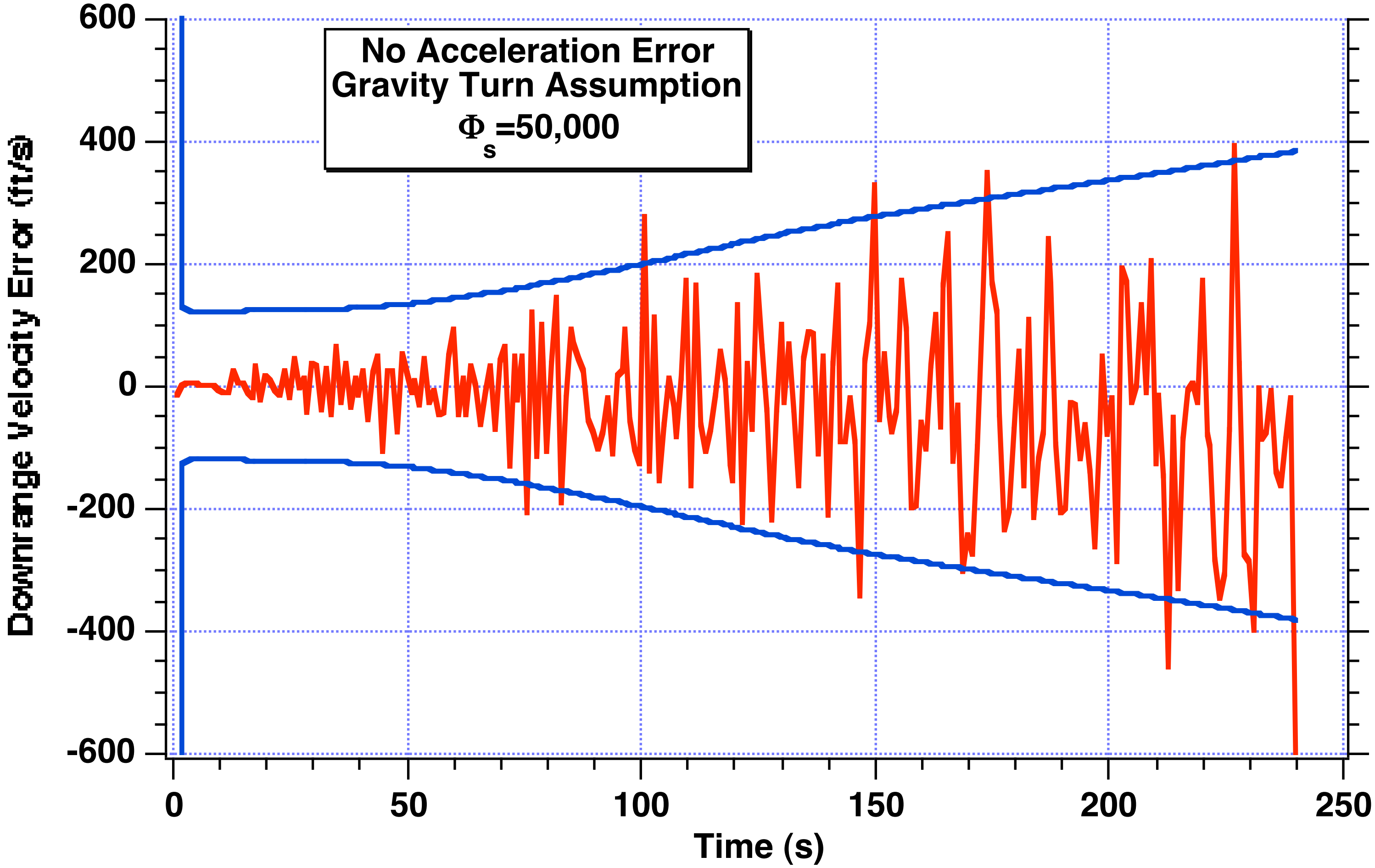
***Recall ICBM is performing Lambert guidance rather than gravity turn in this example**

# Gravity Turn Assumption Causes Two-State Kalman Filter Divergence - Even When Acceleration Magnitude is Know Perfectly




**Gravity turn assumption causes divergence problems in template based Kalman filter because target is actually performing Lambert guidance**

# More Process Noise is Required to Prevent Gravity Turn Two-State Kalman Filter Divergence - Even When ICBM Acceleration Magnitude is Known Perfectly



# 3-State Polynomial Kalman Filter

# 3-State Downrange and Altitude Decoupled Linear Polynomial Kalman Filters - No Template Required

## Downrange Filter

$$\mathrm{Res}_{x_k} = x^*_{T_k} - \hat{x}_{T_{k-1}} - \hat{\dot{x}}_{T_{k-1}} T_s - 0.5\hat{\ddot{x}}_{T_{k-1}} T_s^2$$

$$\hat{x}_{T_k} = \hat{x}_{T_{k-1}} + \hat{\dot{x}}_{T_{k-1}} T_s + 0.5\hat{\ddot{x}}_{T_{k-1}} T_s^2 + K_{1x_k} \mathrm{Res}_{x_k}$$

$$\hat{\dot{x}}_{T_k} = \hat{\dot{x}}_{T_{k-1}} + \hat{\ddot{x}}_{T_{k-1}} T_s + K_{2x_k} \mathrm{Res}_{x_k}$$

$$\hat{\ddot{x}}_{T_k} = \hat{\ddot{x}}_{T_{k-1}} + K_{3x_k} \mathrm{Res}_{x_k}$$

**Note that acceleration is estimated and no a priori information is required**

## Altitude Filter

$$\mathrm{Res}_{y_k} = y^*_{T_k} - \hat{y}_{T_{k-1}} - \hat{\dot{y}}_{T_{k-1}} T_s - 0.5\hat{\ddot{y}}_{T_{k-1}} T_s^2$$

$$\hat{y}_{T_k} = \hat{y}_{T_{k-1}} + \hat{\dot{y}}_{T_{k-1}} T_s + 0.5\hat{\ddot{y}}_{T_{k-1}} T_s^2 + K_{1y_k} \mathrm{Res}_{y_k}$$

$$\hat{\dot{y}}_{T_k} = \hat{\dot{y}}_{T_{k-1}} + \hat{\ddot{y}}_{T_{k-1}} T_s + K_{2y_k} \mathrm{Res}_{y_k}$$

$$\hat{\ddot{y}}_{T_k} = \hat{\ddot{y}}_{T_{k-1}} + K_{3y_k} \mathrm{Res}_{y_k}$$

## Process Noise

$$Q = \Phi_s \begin{bmatrix} \frac{T_s^5}{20} & \frac{T_s^4}{8} & \frac{T_s^3}{6} \\ \frac{T_s^4}{8} & \frac{T_s^3}{3} & \frac{T_s^2}{2} \\ \frac{T_s^3}{6} & \frac{T_s^2}{2} & T_s \end{bmatrix}$$

**We will be adjusting $\Phi_s$ to tune the filter**

# 3-State Decoupled Linear Polynomial Kalman Filters-1

```
GLOBAL DEFINE
        INCLUDE 'quickdraw.inc'
END
IMPLICIT REAL*8 (A-H)
IMPLICIT REAL*8 (O-Z)
REAL*8 M11,M12,M13,M22,M23,M33,K1,K2,K3
REAL*8 M11P,M12P,M13P,M22P,M23P,M33P,K1P,K2P,K3P
LOGICAL QBOOST
INTEGER STEP
OPEN(1,STATUS='UNKNOWN',FILE='DATFIL')
OPEN(2,STATUS='UNKNOWN',FILE='NOISEFIL')
OPEN(3,STATUS='UNKNOWN',FILE='COVFIL')
QBOOST=.TRUE.
TF=1300.
RT1F=5000*3280.
RT2F=0.
GAMTDEG=89.9
VT=1.
RT1=0.
RT2=0.
XR=1000.*3280.
YR=0.
VT1=VT*COS(GAMTDEG/57.3)
VT2=VT*SIN(GAMTDEG/57.3)
G=32.2
H=.01
TS=1.
S=0.
SCOUNT=0.
SIGR=10.
SIGTH=.001
PHIN=60.*G*G/240.
RT=SQRT((RT1-XR)**2+(RT2-YR)**2)
THET=ATAN2(RT2-YR,RT1-XR)
SIGRT1=SQRT((COS(THET)*SIGR)**2+(RT*SIN(THET)*SIGTH)**2)
SIGRT2=SQRT((SIN(THET)*SIGR)**2+(RT*COS(THET)*SIGTH)**2)
K1=0.
K2=0.
K3=0.
RT1H=0.
VT1H=0.
RT2H=0.
VT2H=0.
AT1H=0.
AT2H=0.
T=0.
```

**Desired Destination and Time of Arrival** (TF, RT1F, RT2F)

**Range and angle measurement errors** (SIGR, SIGTH)

**Filter process noise** (PHIN)

**Actual range and angle** (RT, THET)

**Standard deviation of pseudo measurements** (SIGRT1, SIGRT2)

**Filter initialization** (RT1H … AT2H)

# 3-State Decoupled Linear Polynomial Kalman Filters-2

```
        P11=99999999999.
        P12=0.
        P13=0.
        P22=99999999999.
        P23=0.
        P33=99999999999.
        P11P=99999999999.
        P12P=0.
        P13P=0.
        P22P=99999999999.
        P23P=0.
        P33P=99999999999.
        AX=0.
        AY=0.
        XN=0.
10      IF(T>240.)GOTO 999
        RT1OLD=RT1
        RT2OLD=RT2
        VT1OLD=VT1
        VT2OLD=VT2
        STEP=1
        GOTO 200
66      STEP=2
        RT1=RT1+H*VT1
        RT2=RT2+H*VT2
        VT1=VT1+H*AT1
        VT2=VT2+H*AT2
        T=T+H
        GOTO 200
55      RT1=(RT1OLD+RT1)/2+.5*H*VT1
        RT2=(RT2OLD+RT2)/2+.5*H*VT2
        VT1=(VT1OLD+VT1)/2+.5*H*AT1
        VT2=(VT2OLD+VT2)/2+.5*H*AT2
        IF(QBOOST)THEN
                TGOLAM=TF-T
                VXLAM=(RT1F-RT1)/TGOLAM
                VYLAM=(RT2F-RT2+16.1*TGOLAM*TGOLAM)/TGOLAM
                DELVX=VXLAM-VT1
                DELVY=VYLAM-VT2
                DELV=SQRT(DELVX**2+DELVY**2)
                IF(TRST>0..AND.DELV>10.)THEN
                        AX=AT*DELVX/DELV
                        AY=AT*DELVY/DELV
                ELSEIF(DELV<10.)THEN
                        TRST=0.
                        QBOOST=.FALSE.
                        AX=0.
                        AY=0.
                        VT1OLD=VXLAM
                        VT2OLD=VYLAM
```

**Initial covariance matrix** (P11 through P33P)

**2nd order Runge-Kutta integration of ICBM for boost phase** (statement 10 through VT2=(VT2OLD+VT2)/2+.5*H*AT2)

**Lambert guidance** (from IF(QBOOST)THEN onward)

# 3-State Decoupled Linear Polynomial Kalman Filters-3

```
          ELSE
                    QBOOST=.FALSE.
                    AX=0.
                    AY=0.
                    WRITE(9,*)DELV
                    PAUSE
          ENDIF
ENDIF
IF(T<20.)THEN
          AX=0.
          AY=AT
ENDIF
SCOUNT=SCOUNT+H
IF(SCOUNT.LT.(TS-.00001))GOTO 10
SCOUNT=0.
XN=XN+1.
XK1=3*(3*XN*XN-3*XN+2)/(XN*(XN+1)*(XN+2))
XK2=18*(2*XN-1)/(XN*(XN+1)*(XN+2)*TS)
XK3=60/(XN*(XN+1)*(XN+2)*TS*TS)
TS2=TS*TS
TS3=TS2*TS
TS4=TS3*TS
TS5=TS4*TS
RT=SQRT((RT1-XR)**2+(RT2-YR)**2)
THET=ATAN2(RT2-YR,RT1-XR)
SIGRT1=SQRT((COS(THET)*SIGR)**2+(RT*SIN(THET)*SIGTH)**2)
SIGRT2=SQRT((SIN(THET)*SIGR)**2+(RT*COS(THET)*SIGTH)**2)
SIGN2=SIGRT1*SIGRT1
M11=P11+TS*P12+.5*TS2*P13+TS*(P12+TS*P22+.5*TS2*P23)
M11=M11+.5*TS2*(P13+TS*P23+.5*TS2*P33)+TS5*PHIN/20.
M12=P12+TS*P22+.5*TS2*P23+TS*(P13+TS*P23+.5*TS2*P33)+TS4*PHIN/8.
M13=P13+TS*P23+.5*TS2*P33+PHIN*TS3/6.
M22=P22+TS*P23+TS*(P23+TS*P33)+PHIN*TS3/3.
M23=P23+TS*P33+.5*TS2*PHIN
M33=P33+PHIN*TS
BOT=M11+SIGN2
K1=M11/BOT
K2=M12/BOT
K3=M13/BOT
FACT=1.-K1
P11=FACT*M11
P12=FACT*M12
P13=FACT*M13
P22=-K2*M12+M22
P23=-K2*M13+M23
P33=-K3*M13+M33
```

**Go straight up for first 20 s**

**Least squares filter gains**

**True angle and range**

**Pseudo measurement standard deviations**

**Ricatti equations for downrange filter**

# 3-State Decoupled Linear Polynomial Kalman Filters-4

```
          SIGN2P=SIGRT2*SIGRT2
          M11P=P11P+TS*P12P+.5*TS2*P13P+TS*(P12P+TS*P22P+.5*TS2*P23P)
          M11P=M11P+.5*TS2*(P13P+TS*P23P+.5*TS2*P33P)+TS5*PHIN/20.
          M12P=P12P+TS*P22P+.5*TS2*P23P+TS*(P13P+TS*P23P+.5*TS2*P33P)
     1                   +TS4*PHIN/8.
          M13P=P13P+TS*P23P+.5*TS2*P33P+PHIN*TS3/6.
          M22P=P22P+TS*P23P+TS*(P23P+TS*P33P)+PHIN*TS3/3.
          M23P=P23P+TS*P33P+.5*TS2*PHIN
          M33P=P33P+PHIN*TS
          BOTP=M11P+SIGN2P
          K1P=M11P/BOTP
          K2P=M12P/BOTP
          K3P=M13P/BOTP
          FACTP=1.-K1P
          P11P=FACTP*M11P
          P12P=FACTP*M12P
          P13P=FACTP*M13P
          P22P=-K2P*M12P+M22P
          P23P=-K2P*M13P+M23P
          P33P=-K3P*M13P+M33P
```

**Ricatti equations for altitude filter**

```
          CALL GAUSS(THETNOISE,SIGTH)
          CALL GAUSS(RTNOISE,SIGR)
          THETMEAS=THET+THETNOISE
          RTMEAS=RT+RTNOISE
```

**Actual angle and range measurements**

```
          RT1S=RTMEAS*COS(THETMEAS)+XR
          RT2S=RTMEAS*SIN(THETMEAS)+YR
```

**Pseudo measurements**

```
          IF(XN<10.)THEN
                   XK1PZ=XK1
                   XK2PZ=XK2
                   XK3PZ=XK3
          ELSE
                   XK1PZ=K1
                   XK2PZ=K2
                   XK3PZ=K3
          ENDIF
```

**Use least squares gains for first 10 measurements**

```
          RESX=RT1S-RT1H-TS*VT1H-.5*TS2*AT1H
          RT1H=XK1PZ*RESX+RT1H+TS*VT1H+.5*TS2*AT1H
          VT1H=XK2PZ*RESX+VT1H+TS*AT1H
          AT1H=XK3PZ*RESX+AT1H
```

**Downrange Kalman filter**

```
          IF(XN<10.)THEN
                   XK1PPZ=XK1
                   XK2PPZ=XK2
                   XK3PPZ=XK3
          ELSE
                   XK1PPZ=K1P
                   XK2PPZ=K2P
                   XK3PPZ=K3P
          ENDIF
```

**Use least squares gains for first 10 measurements**

# 3-State Decoupled Linear Polynomial Kalman Filters-5

```
      IF(XN<10.)THEN
               XK1PPZ=XK1
               XK2PPZ=XK2
               XK3PPZ=XK3
        ELSE
               XK1PPZ=K1P
               XK2PPZ=K2P
               XK3PPZ=K3P
        ENDIF
        RESY=RT2S-RT2H-TS*VT2H-.5*TS2*AT2H
        RT2H=XK1PPZ*RESY+RT2H+TS*VT2H+.5*TS2*AT2H
        VT2H=XK2PPZ*RESY+VT2H+TS*AT2H
        AT2H=XK3PPZ*RESY+AT2H
        SP33=SQRT(P33)/32.2
        ERRAT1=(AT1-AT1H)/32.2
        SP33P=SQRT(P33P)/32.2
        ERRAT2=(AT2-AT2H)/32.2
        SP33=SQRT(P33)/32.2
        ERRAT1=(AT1-AT1H)/32.2
        SP33P=SQRT(P33P)/32.2
        ERRAT2=(AT2-AT2H)/32.2
        RT1KM=RT1/3280.
        RT2KM=RT2/3280.
        RT1NOISE=RT1-RT1S
        RT2NOISE=RT2-RT2S
        ERRRT1=RT1-RT1H
        ERRVT1=VT1-VT1H
        SP11=SQRT(P11)
        SP22=SQRT(P22)
        ERRRT2=RT2-RT2H
        ERRVT2=VT2-VT2H
        SP11P=SQRT(P11P)
        SP22P=SQRT(P22P)
        WRITE(9,*)T,RT1KM,RT2KM,AT1/G,AT1H/G,AT2/G,AT2H/G
        WRITE(1,*)T,RT1KM,RT2KM,AT1/G,AT1H/G,AT2/G,AT2H/G
        WRITE(2,*)T,RT1NOISE,SIGRT1,-SIGRT1,RT2NOISE,SIGRT2,-SIGRT2
        WRITE(3,*)T,ERRRT1,SP11,-SP11,ERRVT1,SP22,-SP22,ERRAT1,SP33,
     1         -SP33,ERRRT2,SP11P,-SP11P,ERRVT2,SP22P,-SP22P,ERRAT2,
     2         SP33P,-SP33P
        GOTO 10
 200    CONTINUE
```

**Use least squares gains for first 10 measurements**

**Altitude Kalman filter**

**Actual and theoretical errors in estimates**

# 3-State Decoupled Linear Polynomial Kalman Filters-6

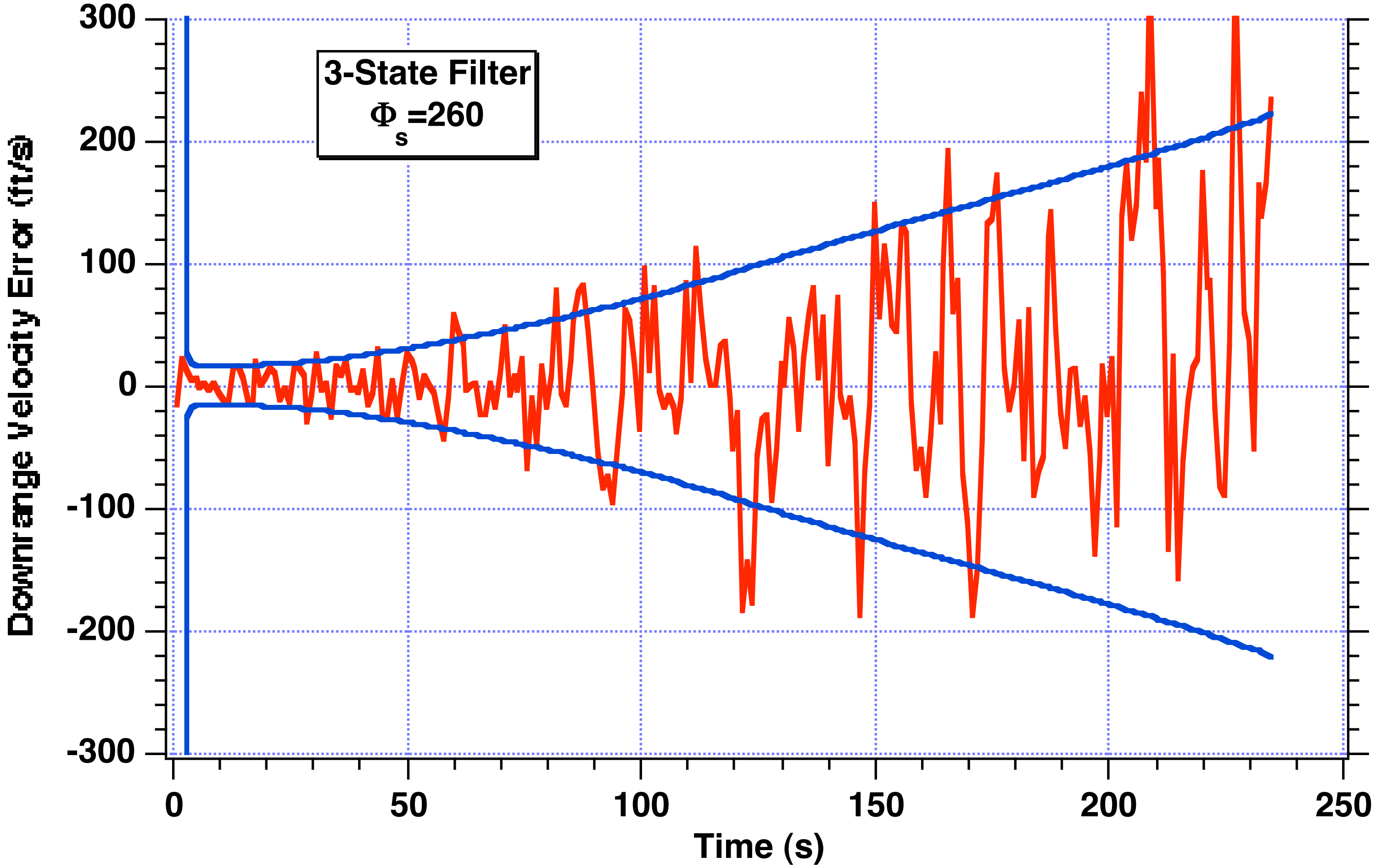
```
        IF(T<120.)THEN
                WGT=-2622*T+440660.
                TRST=725850.
        ELSEIF(T<240.)THEN
                WGT=-642.*T+168120.
                TRST=182250.
        ELSE
                WGT=5500.
                TRST=0.
        ENDIF
        VT=SQRT(VT1**2+VT2**2)
        GAM=ATAN2(VT2,VT1)
        AT=G*TRST/WGT
        AT1=AX
        AT2=-G+AY
        IF(STEP-1)66,66,55
999     CONTINUE
        RT1KM=RT1/3280.
        RT2KM=RT2/3280.
        WRITE(9,*)T,RT1KM,RT2KM,AT1/G,AT1H/G,AT2/G,AT2H/G
        WRITE(1,*)T,RT1KM,RT2KM,AT1/G,AT1H/G,AT2/G,AT2H/G
        PAUSE
        CLOSE(1)
        CLOSE(2)
        CLOSE(3)
        END
```

**ICBM** (IF/ELSEIF/ELSE/ENDIF block)

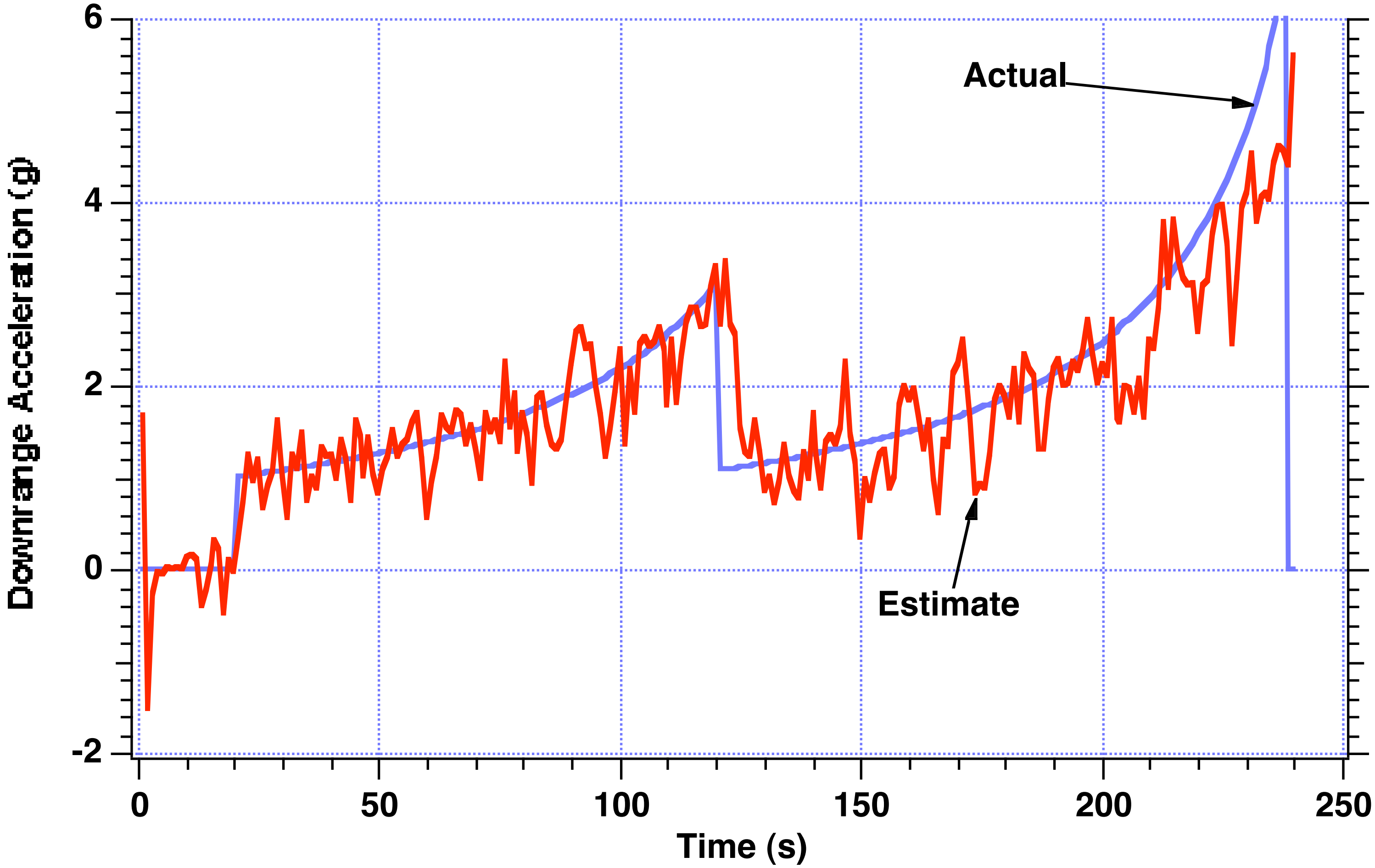
**Acceleration** (AT, AT1, AT2 lines)

**Differential equations** (IF(T<120.) through AT2=-G+AY)

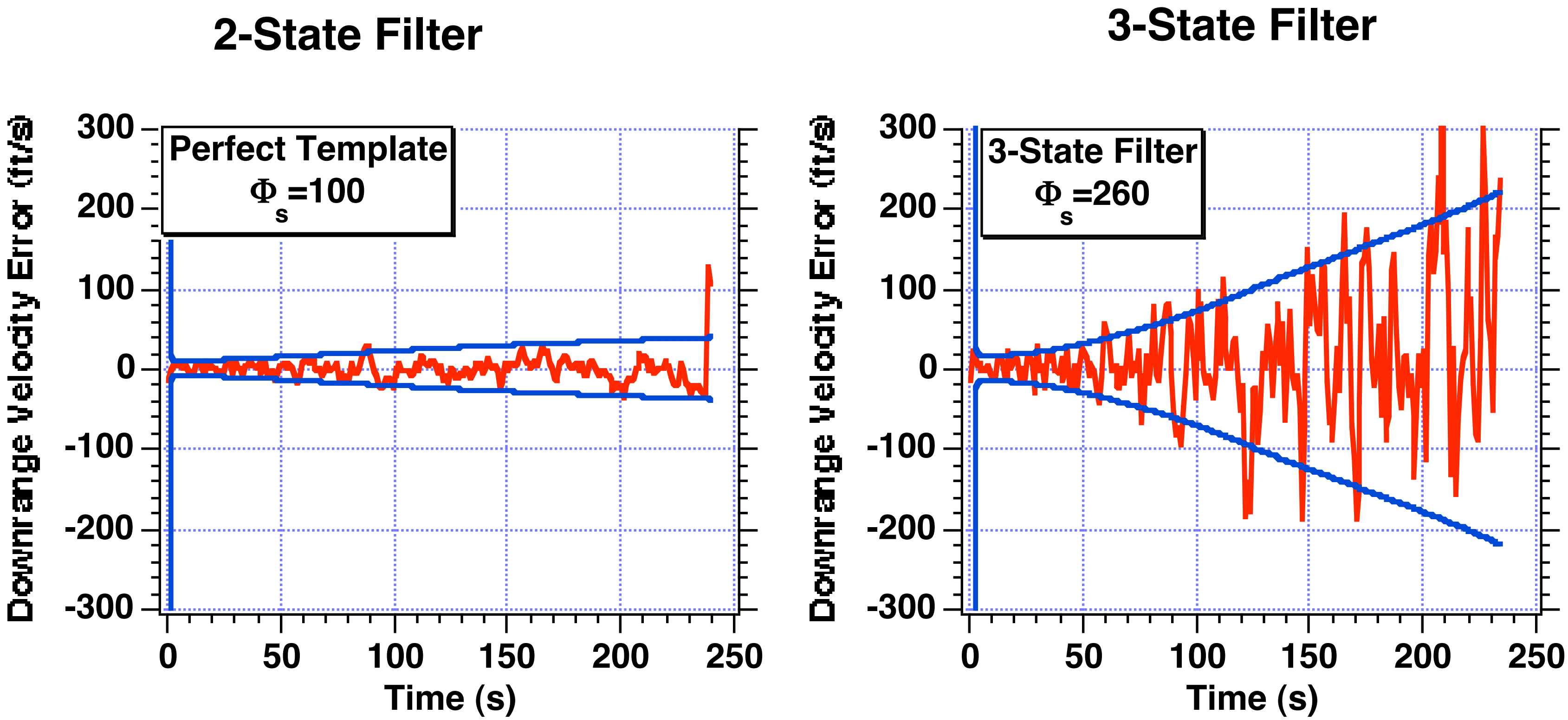
# Three-State Kalman Filter is Tuned so That Errors in Estimates are Consistent



# Nominal Three-State Linear Polynomial Kalman Filter Tracks ICBM Acceleration Quite Well

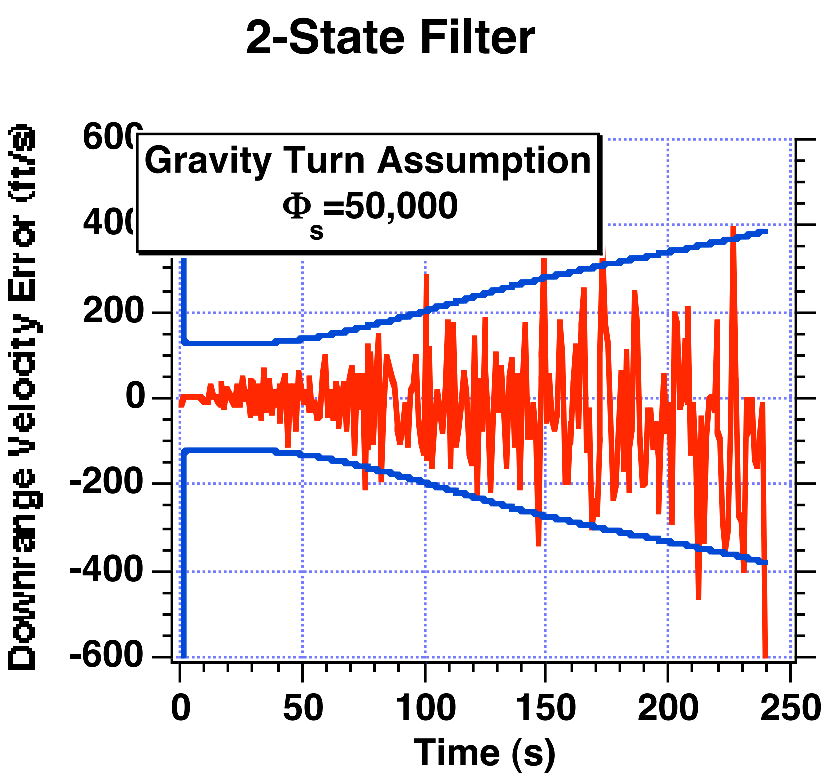


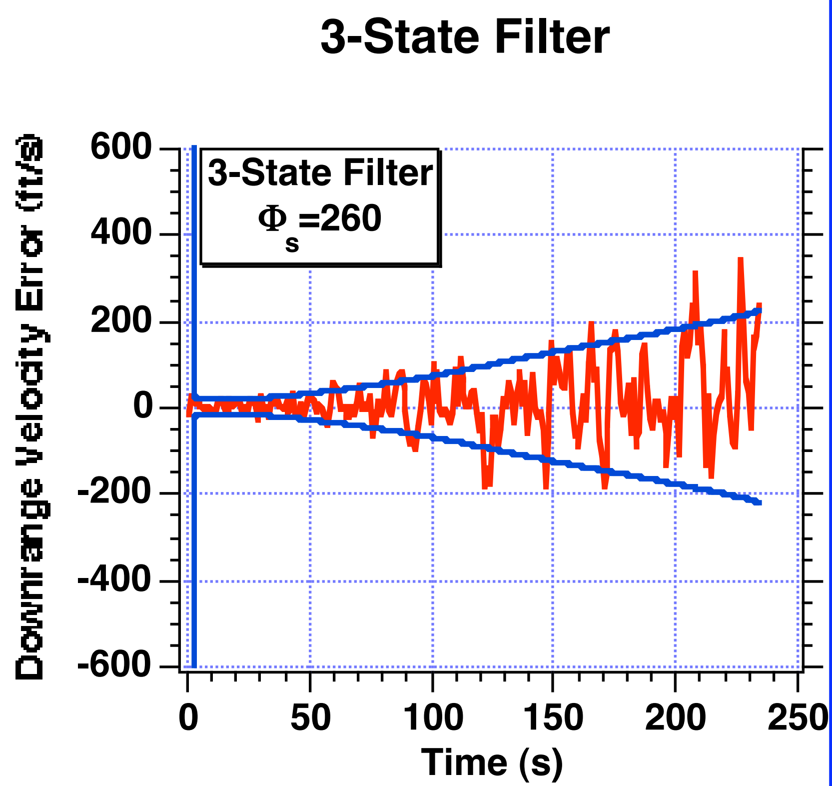
# Filter Comparison - 1




**2-State filter yields better results than 3-State filter if it has a perfect knowledge of the target acceleration magnitude and direction**

# Filter Comparison - 2




**3-State filter yields better results than 2-State filter when realistic errors are considered**