Automated Re-entry System using FNPEG

Wyatt R. Johnson, Ph.D.*
Intuitive Machines, Houston, TX, 77058

Ping Lu, Ph.D.†
San Diego State University, San Diego, CA 92182

Susan J. Stachowiak.‡
NASA Johnson Space Center, Houston, TX, 77058

This paper discusses the implementation and simulated performance of the FNPEG (Fully Numerical Predictor-corrector Entry Guidance) algorithm into GNC FSW (Guidance, Navigation, and Control Flight Software) for use in an autonomous re-entry vehicle. A few modifications to FNPEG are discussed that result in computational savings — a change to the state propagator, and a modification to cross-range lateral logic. Finally, some Monte Carlo results are presented using a representative vehicle in both a high-fidelity 6-DOF (degree-of-freedom) sim as well as in a 3-DOF sim for independent validation.

I. Introduction

Intuitive Machines is developing FSW (flight software) for future orbiter and re-entry capability[1] named "ARS" (Automated Re-entry System). Historically, low L/D (lift over drag) re-entry spacecraft for both Earth and Mars entry have used some variant of the Apollo entry guidance algorithm[2]. The Apollo algorithm is lightweight, and has performed well in competition against more complex algorithms, including other predictor-corrector algorithms. [3–5] . The Mars studies have generally found that an optimized reference profile provides better performance in the presence of high model uncertainty than predictor-corrector schemes. However, for Earth missions, atmospheric uncertainty is not as significant. FNPEG[6, 7] provides a predictor-corrector algorithm that does not require the significant preflight tuning and optimization that the Apollo variants do.

Although FNPEG has these nice features, it has not yet been previously flown onboard a vehicle or demonstrated to run on a flight processor. The intended interface between FNPEG and the rest of GNC is that FNPEG should get executed at a 1 Hz rate. In that function call, FNPEG receives an update on the vehicle's state and current acceleration, and computes a new commanded bank angle. The commanded bank angle is the solution to the downrange distance flown problem. This is a nonlinear equation as a function of bank angle, and can be solved for in an iterative fashion (e.g., Brent's method). Each iteration of the root solver involves integrating the equations of motion for a vehicle with lift and drag around a rotating planet.[8]

The ARS FSW is written using CFS (Core Flight Software)[9] as its backbone, and was developed for use and tested on a Freescale A4 processor. CFS executes FSW according to a pre-emptive priority-based scheduler. Considering that this flight processor has a performance of an order of magnitude slower than a typical modern desktop, and that CFS uses time-sliced scheduling, FNPEG needed to both become faster in execution (or equivalently, more efficient with its computations), and needed to be re-architected for CFS integration. This paper will discuss two changes to FNPEG to improve execution time. The first improvement is a replacement of the fixed-step Runge-Kutta 4th-order integrator (i.e., RK4) with a variable-step RK45 method. The second is a simple change to a previously reported improvement to crossrange control[10] (but at a cost of higher CPU usage) that provides the same improvement and requires no additional computational cost.

Finally, 6-DOF (degree-of-freedom) and 3-DOF Monte Carlo performance results are shown for a representative Earth entry vehicle with a mid-range L/D and a full suite of dispersions.

II. State Integrator

The unmodified FNPEG logic used an RK4 method with the Butcher tableau as shown in Table 1, and was later switched to the Dormand-Prince RK45 integrator[11] as shown in Table 2. The RK45 method provides two solutions

*Senior Development Engineer, Intuitive Machines, 3700 Bay Area Blvd Suite 100, Houston, TX, 77058, and AIAA Member.
†Professor, Chair, Department of Aerospace Engineering, AIAA Fellow
‡Engineer, Flight Mechanics and Trajectory Design Branch, 2101 NASA Parkway, Houston, TX, 77058

— a 4th and 5th order accurate solution. If the difference of these two solutions is within a specified error tolerance, than the integration step is accepted. If the error exceeds the tolerance, then the integration step is rejected. In either case, the subsequent integration step size is adjusted to try and drive the error towards the specified error tolerance. In this manner, a step size that was unacceptably large will get smaller, and an overly conservative step size will increase.

0				
$\frac{1}{2}$	$\frac{1}{2}$			
$\frac{1}{2}$	0	$\frac{1}{2}$		
1	0	0	1	
	$\frac{1}{6}$	$\frac{1}{3}$	$\frac{1}{3}$	$\frac{1}{6}$

Table 1: RK4 Butcher tableau

0						
$\frac{1}{5}$	$\frac{1}{5}$					
$\frac{3}{10}$	$\frac{3}{40}$	$\frac{9}{40}$				
$\frac{4}{5}$	$\frac{44}{45}$	$-\frac{56}{15}$	$\frac{32}{9}$			
$\frac{8}{9}$	$\frac{19372}{6561}$	$-\frac{25360}{2187}$	$\frac{64448}{6561}$	$-\frac{212}{729}$		
1	$\frac{9017}{3168}$	$-\frac{355}{33}$	$\frac{46732}{5247}$	$\frac{49}{176}$	$-\frac{5103}{18656}$	
1	$\frac{35}{384}$	0	$\frac{500}{1113}$	$\frac{125}{192}$	$-\frac{2187}{6784}$	$\frac{11}{84}$
	$\frac{35}{384}$	0	$\frac{500}{1113}$	$\frac{125}{192}$	$-\frac{2187}{6784}$	$\frac{11}{84}$	0
	$\frac{5179}{57600}$	0	$\frac{7571}{16695}$	$\frac{393}{640}$	$-\frac{92097}{339200}$	$\frac{187}{2100}$	$\frac{1}{40}$

Table 2: RK45 Dormand-Prince Butcher tableau

Adaptive step-size control is achieved by comparing the local truncation error to a tolerance and making a decision to accept or reject the step. However, this truncation error is a vector quantity. A simple L^2 norm does not suffice, as each state variable may be scaled to different units, and have differing relative impacts as to the final state integration. For this reason, the truncation error is checked for each state variable independently. If any of these checks fail, the integration step is rejected. The step size control uses the worst-case metric for the next step. Figure 1 illustrates the impact of using a constant step size integrator versus an adaptive step size integrator. There are two primary features: first, the integration remains stable, as we ensure that the step size taken is consistent with the desired error tolerance, and secondly, the total number of integration steps is far less (about a 60–75% savings) when using the adaptive step size. The constant step size wastes too much time at high velocities when the dynamic pressure is too low to significantly alter the trajectory. The constant step size also takes too large of a step at low velocities, when heading changes become more sensitive to the bank command.

A. Adaptive Stepsize Control

The RK45 methods provide an embedded local truncation error estimate. This error can be used to adaptively adjust the step size for the next integration step. If the error is much smaller than the desired accuracy, then the step size can be increased. If the error is much greater than the desired accuracy, the step size must be reduced, and the integration step re-attempted.

In each integration step, let $e = ||\bar{e}||$ be a scalar measure of the vectorized local truncation error, η be the desired integration tolerance, δ be the current trial step size, δ_{next} be the suggested step size for the next integration step, ϵ be the desired relative integration tolerance, and γ be the desired absolute integration tolerance. The following procedure is suggested to determine the integration step size:[12]

$$\eta \equiv \epsilon||\bar{x}|| + \gamma \tag{1}$$

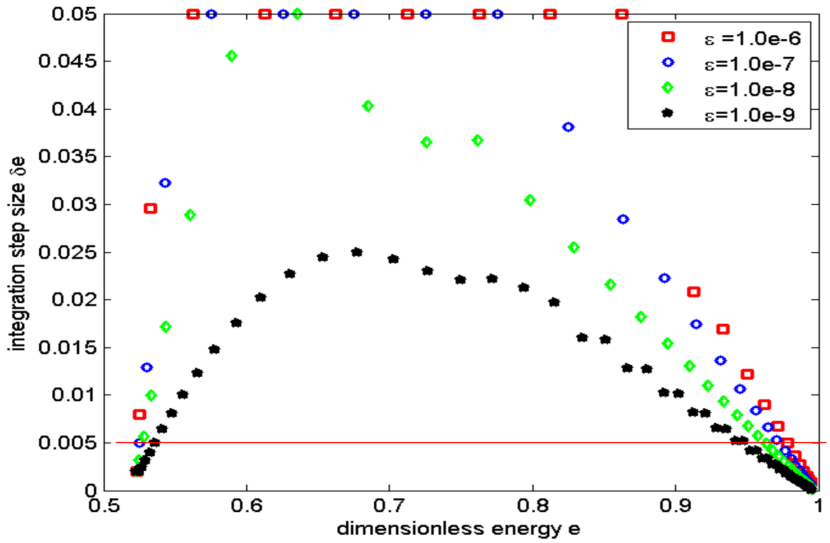


Figure 1: Comparison of fixed step size using RK4 vs adaptive step size using RK45. A fixed step size of 0.005 (in dimensionless energy) results in exceeding error tolerances at entry interface and terminal phases of hypersonic entry. For the region between a dimensionless energy of 0.53 and 0.97, the fixed step size integrator requires too many steps to obtain the same level of accuracy. For all other energies, the fixed step size is insufficient to maintain integration accuracy.

If the integration tolerance is met ($e/\eta \leq 1$), then compute the next integration step size as:

$$\delta_{next} = MIN\left\{SF \cdot \delta\left(\frac{\eta}{e}\right)^{1/5}, 5.0\delta\right\} \tag{2}$$

Otherwise, the integration tolerance is not met ($e/\eta > 1$), and the current integration step is reattempted with a smaller stepsize:

$$\delta = MAX\left\{SF \cdot \delta\left(\frac{\eta}{e}\right)^{1/4}, 0.1\delta\right\} \tag{3}$$

Equations (2) and (3) will expand or reduce the step size according to the magnitude of the local truncation error. To prevent excessive jumps, the $MAX(\cdot,\cdot)$ and $MIN(\cdot,\cdot)$ functions provide bounds on the jumps from step to step. The value SF represents a safety factor to bias the steps to be slightly smaller than the expected value. A failed step has a larger impact to computational throughput than an accepted step which could have been somewhat larger. Thus, the algorithm errs on the side of caution.

ARS uses a variant of this algorithm, where the calculation of the scalarized truncation error e is slightly modified. As written, this scalar error is the L^2 norm of the vector truncation error. This norm is not ideal because it tends to suppress larger errors in one state variable if the others are good. Also, the state variables may not have the same scale, or may not have the same sensitivity to integration error. In the case of FNPEG, the integration state variables are all scaled to the order of 1, which eliminates the second concern. To alleviate the other two concerns, ARS uses a modified approach by assigning each state variable its own relative and absolute integration tolerance, and then uses the worst case error by use of the L^∞ norm.

Define:

$$\zeta = \left|\left|\frac{|e_k|}{\epsilon_k|x_k| + \gamma_k}\right|\right|_\infty \tag{4}$$

In Equation (4), ζ represents a scaled L^∞ norm of e/η. The step size control is then written as:

$$\delta_{next} = MIN\left\{SF \cdot \delta\left(\frac{1}{\zeta}\right)^{1/5}, 5.0\delta\right\} \tag{5}$$

$$\delta = MAX\left\{SF \cdot \delta\left(\frac{1}{\zeta}\right)^{1/4}, 0.1\delta\right\} \tag{6}$$

B. Adaptive Stepsize Tuning

In the second half of the entry, the required step size decreases, as seen in Figure 1. To prevent needless step rejections, an upper bound can be placed on the safety factor term. If each step is barely accepted ($\zeta = 1$), then the

adaptive step size behavior can be modeled by a pair of difference equations. Let E represent the independent variable (dimensionless energy), then:

$$\Delta E_{k+1} = SF \cdot \Delta E_k \tag{7}$$
$$E_{k+1} = E_k + \Delta E_k \tag{8}$$

Equations (7) and (8) have the solution:

$$\Delta E_k = (\Delta E_0) \cdot (SF)^k \tag{9}$$
$$E_k = (\Delta E_0) \frac{(SF)^k - 1}{SF - 1} + E_0 \tag{10}$$
$$= \frac{\Delta E_k - \Delta E_0}{SF - 1} + E_0 \tag{11}$$
$$\Delta E_k = [\Delta E_0 - E_0 (SF - 1)] + E_k (SF - 1) \tag{12}$$

From Eq. (12), we see that the steps in dimensionless energy are bounded by a slope of $SF - 1$. The slope of the blue curve in Figure 1 is about -0.2. This suggests an upper bound of $SF = 0.8$. A larger safety factor will tend to take too large of a step, which would result in rejected integration steps, and increased CPU time. Based on empirical testing (as seen in Figure 2), $SF = 0.6$ tended to work the best.

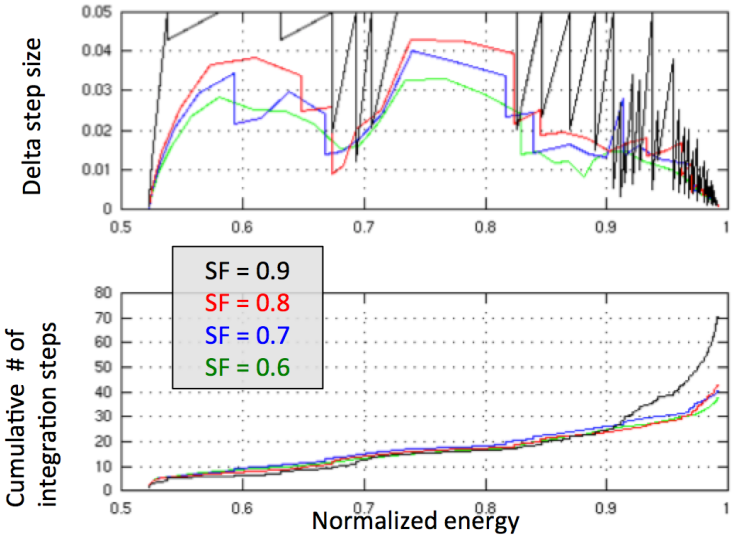


Figure 2: Comparison of different safety factors in integration stepsize control. As predicted, safety factors exceeding $SF = 0.8$ result in numerous rejected steps.

The final part of tuning is to determine the optimal tolerances in which to compute the weighted L^∞ error ζ. Equation (4) has two sets of tolerances : relative tolerances, and absolute tolerances. For this application, all absolute tolerances were set to $\gamma_k = 1 \times 10^{-13}$. If CPU usage were not an issue, using tight relative tolerances would give the best result. The motivation behind the following algorithm is to find the set of relative tolerances that yield the most accurate trajectory while trading in some precision in some of the state variables for better CPU performance:

1. Set all absolute tolerances $\gamma_k = 1 \times 10^{-13}$.
2. Set all relative tolerances $\epsilon_k = 1 \times 10^{-7}$.
3. Flag all relative tolerances ϵ_k as "unconverged".
4. Run a nominal entry to landing simulation with the current set of tolerances.
5. Identify the state variable that had the driving tolerance (caused the most step rejections), and increase it by a factor of 10.
6. Re-run the trajectory simulation. If the simulation results in a higher CPU cost, or provides unacceptable integration accuracy:
   (a) Revert the change.

(b) Mark that relative tolerance ϵ_k as "converged".

7. Repeat step 4 until all tolerances are marked as "converged".

Using the above algorithm, the final set of relative tolerances were determined to be as depicted in Table 3. The states listed on the top required a tight tolerance for integration error control. The states listed towards the bottom could accommodate relatively larger errors without causing as much impact.

C. Results

In summary, the changes to the FNPEG integrator consist of:

1. Switching from an RK4 "3/8" rule to RK45 Dormand-Prince.
2. Using an adaptive step size control that uses a weighted L^∞ norm scheme for integration tolerances.
3. Empirically determining a safety factor that results in the fewest integration steps.
4. Empirically determining a set of relative tolerances that provides the best CPU savings for a given tolerance.

The baseline FNPEG algorithm required 125,141 function calls to RK4 from entry to chute deploy. The modifications resulted in only needing 33,905 function calls to RK45, resulting in a total of a 73% CPU savings across the entire trajectory.

Relative Tolerance	State Variable
1×10^{-7}	radius to center of planet, longitude, range-to-go
1×10^{-6}	latitude, relative flight path azimuth
1×10^{-5}	time
1×10^{-4}	flight path angle

Table 3: Empirically determined relative tolerances that provide best integration accuracy for CPU time spent.

III. Lateral Logic

Early versions of FNPEG used the traditional cross range corridor logic as used by Apollo or the Shuttle. In these algorithms, a cross-range metric was computed indicating how far the vehicle was from the nominal trajectory (which could be represented as an inclination error or a heading error), and then compared to a threshold indicating when to initiate a bank reversal to maintain lateral proximity to the reference trajectory.

Previously, Smith[10] presented an alternate approach to bank reversals that used the pre-existing capabilities of FNPEG's numerical predictor. Rather than have FNPEG account for bank reversals during its forward integrations, guidance would instead assume a constant bank to both the port and starboard sides, and use the final estimated crossranges to determine when to perform a bank reversal. The requirement to analyze both bank directions necessarily dictates an increase to CPU usage, which ARS wishes to avoid. The alterations presented here also have the benefit of removing a slight crossrange bias from the original formulation.

Smith defines two crossrange errors, χ^+ and χ^-, for the final crossrange errors assuming a positive and negative bank direction, respectively. A third metric is defined, shown in Eq. (13), representing the ratio of these quantities.

$$K = \frac{\chi^+}{\chi^-} \tag{13}$$

Smith shows that for n bank reversals, the terminal crossrange error is nominally given by Eq. (14):

$$\chi_f = \frac{\chi^+}{K^n} \tag{14}$$

As n increases, the terminal crossrange error decreases, but never reaches 0. For sufficiently high n, the uncertainties during EDL will render this bias negligible. However, if propellant allocation to bank reversals or transients during bank reversals are issues, then the trajectory designer may wish to limit n to a smaller value.

The crossrange problem can instead be formulated in terms of cross range capability, $\Delta\chi^+$ and $\Delta\chi^-$. These capabilities indicate the ability to change the final crossrange relative to the current crossrange, given by χ_0. We now

define the fractional crossrange capabilities f^+ and f^- as:

$$f^+ \equiv \left| \frac{\chi_0}{\Delta\chi^+} \right| \tag{15}$$

$$f^- \equiv \left| \frac{\chi_0}{\Delta\chi^-} \right| \tag{16}$$

Due to Earth rotation, $f^+ \neq f^-$. However, the crossrange capability that takes the vehicle further away from the centerline is of no use. These quantities can thus be merged into a single value:

$$f = \begin{cases} \left| \frac{\chi_0}{\Delta\chi^+} \right| & \chi_0 < 0 \\ \left| \frac{\chi_0}{\Delta\chi^-} \right| & \chi_0 \geq 0 \end{cases} \tag{17}$$

A value of $f = 0$ indicates the vehicle currently has no crossrange error. A value of $f = 1$ indicates the vehicle has exactly enough estimated control authority to barely reach the target crossrange.

Under the assumption that the crossrange capability is symmetric (i.e., $\Delta\chi^+ = \Delta\chi^-$), then this method is equivalent to Smith's, with the equivalences as shown in Table 4.

Conversions	$K = \frac{1+f}{1-f}$ $f = \frac{K-1}{K+1}$
Crossrange reduction per reversal	$\frac{1}{K} \leftrightarrow \frac{1-f}{1+f}$
Crossrange capability consumed per reversal	$\frac{K}{K+1} \leftrightarrow \frac{1+f}{2}$

Table 4: Crossrange capability equivalences

If a bank reversal were triggered on a threshold of $f = 1$, then the vehicle would make a single bank reversal at an extreme distance, and then attempt to fly back with the remaining flight time. Early in hypersonic entry, the vehicle has large atmospheric and aerodynamic uncertainties relative to its capability to alter crossrange. This extreme threshold would be dangerous since it allows for no error margin. Conversely, at the end of flight, the vehicle has very little crossrange authority, and a low f threshold would needlessly create a crossrange bias at chute deploy. Rather than use a single threshold for the entire flight, the threshold can be scheduled — either as a function of the number of executed bank reversals thus far, or as a function of velocity. Early bank reversals should have a low f threshold, and later bank reversals should have an increasing threshold. The final bank reversal should have a threshold of $f = 1$ to remove any final crossrange bias. An example is depicted in Table 5 and Figure 3. A smaller, more conservative amount of crossrange is allowed before triggering the first bank reversal. Although each subsequent reversal allows for a greater percentage of capability, the absolute capability remaining is greatly reduced. Any predictive errors will have little impact. In Figure 3, the step impulses of the crossrange capability (right hand plot) on the x-axis are due to the switching nature of Eq. (17). Also, if the crossrange capability were being perfectly predicted, the slope of this plot should alternate between ± 1. At entry, where uncertainty is greatest, the predicted crossrange capability is a noisy estimate. As the flight progresses, this estimate improves and generally follows the theoretical ± 1 slope.

Bank reversal	f	Estimated Initial Capability (km)	Stage Reduction	Estimated Remaining Capability (km)
1	0.50	151.8	66.66%	50.6
2	0.65	50.6	78.88%	10.7
3	0.80	10.7	88.88%	1.2
4	1.0	1.2	100.00%	0

Table 5: Estimated crossrange authority from entry to chute deploy as a function of bank reversal. Initial bank reversals use a conversative threshold to ensure sufficient control authority on dispersed cases. Late bank reversals use more of the available control authority.

IV. Monte Carlo Performance

A series of 1000 entry cases was considered, where the dispersions included variations to the vehicle state (prior to a deorbit burn), mass properties, control system effectors, aerodynamics[13, 14], and atmospheric profiles. A few selected dispersions are shown in Table 6.

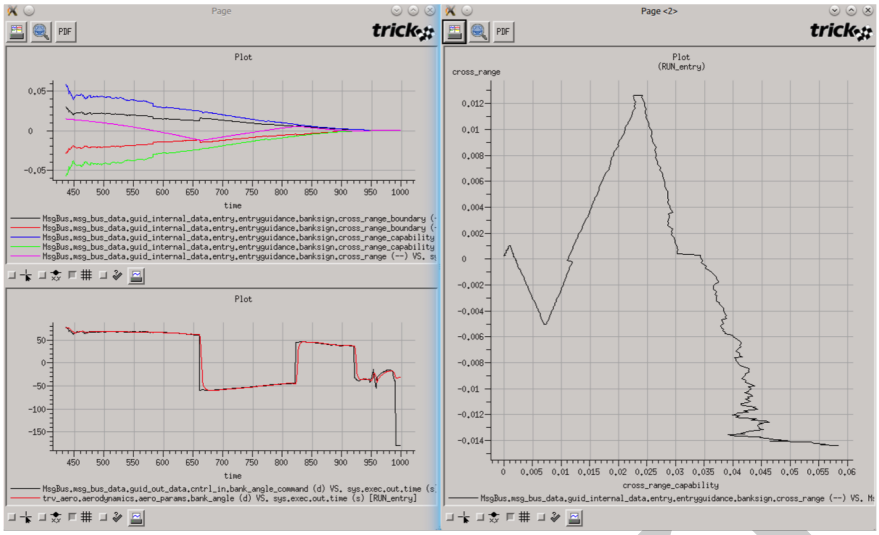


Figure 3: Estimated crossrange authority from entry to chute deploy. The upper-left plot shows the estimated crossrange capability in blue and green. The margined authority is shown in black and red. Finally, the simulated flight profile is shown in magenta. As the magenta curve intersects the margined authority curves, a bank reversal is executed. The bottom-left plot shows the bank angle command (black) co-plotted with the nav estimate of actual bank angle (in red). Finally, the right-hand plot shows the current crossrange angle as a function of estimated crossrange authority.

Parameter	Units	Mean	1σ	Min	Max
Entry Mass	kg	83.34	0.58	81.25	84.87
Entry FPA	deg	-1.17	0.007	-1.18	-1.14
Entry Downrange	km	4443	34	4318	4523
Entry Crossrange	km	-197.2	2.2	-202.3	-189.3
RCS Jet Force	N	4.0	0.2	3.4	4.6
Main Engine Force	N	44.0	2.2	37.2	51.1
Atmosphere Density Multiplier	–	1.0	0.05	.9	1.1
Aerodynamic CA Multiplier	–			-0.1	0.1
Aerodynamic CN Multiplier	–			-0.1	0.1
Aerodynamic CY Multiplier	–			-0.2	0.2
Aerodynamic Cm Bias	–			-0.02	0.02
Aerodynamic Cm Multiplier	–			-0.5	0.5
Aerodynamic Cln Bias	–			-0.0005	0.0005
Aerodynamic Cln Multiplier	–			-1.0	1.0
Aerodynamic Cll Bias	–			-0.0025	0.0025
Aerodynamic Cll Multiplier	–			-0.3	0.3
Aerodynamic Cmq Bias	–			–1.2	0
Aerodynamic Cnr Bias	–			-1.2	0
Aerodynamic Clp Bias	–			-0.5	0

Table 6: A list of selected dispersions for entry guidance performance.

A. Smart Chute Deploy Logic

ARS uses a smart chute deploy logic similar to that of MSL[15]. The logic requires an upper threshold be satisfied for altitude, mach, and dynamic pressure. Once these three conditions are all simultaneously true, then the drogue chute deploy will be triggered by a range trigger, where the range to the target begins to increase. Finally, the drogue chute can also be deployed by a backup trigger. The backup triggers deploy the chute upon any lower threshold being violated for altitude, mach, or dynamic pressure. The backup dynamic pressure trigger also requires the upper altitude constraint to be true, to prevent chute deployment at entry interface.

The supersonic drogue chute has a lower and upper limit on dynamic pressure and an upper limit on Mach for proper inflation. A lower limit on Mach is imposed for vehicle stability, since the aeroshell is unstable at subsonic velocities. A lower limit on altitude is imposed for timeline margin. Finally, the upper limit on altitude is somewhat arbitrary — its purpose is only to prevent chute inflation at entry, as stated earlier.

B. 6-DOF Results

The resulting guidance performance is shown in Figures 4 and 5. Figure 4 shows the conditions of drogue chute deploy relative to the design constraints. Because of the aeroshell instability at low Mach numbers, we prefer to deploy towards the upper Mach constraint, as seen in Figure 4c. Even though the vehicle remains stable, cases that deploy with lower Mach numbers require increased propellant consumption for RCS. Some of the cases trigger the chute deployment immediately upon passing through the upper Mach constraint, which implies the range trigger was immediately active. These cases had already passed the target and were getting further. This behavior is also evident in Figure 4. as the downrange scatter is biased in the positive direction. However, the 99.7% performance is a 1.2 km miss at drogue chute deploy, which is well within the 5 km requirement for a guided parafoil to finish the landing.

Figure 5 shows the bank history for all 1000 cases. Starting from entry (the right-hand side of the plot), the bank profiles remain tightly spread about the nominal up until about 1000 m/s, where a final phase steering command takes over. The stability of the bank profiles indicates FNPEG is handling the dispersions well, and not diverging on extreme cases. The final bank command tends to be 180 deg, as most cases slightly overshoot the target.

C. 3-DOF Results

An initial assessment of FNPEG performance was performed using a 3-DOF simulation. The simulation used an early version of the FNPEG guidance, including an the older predictive lateral guidance, and modeled only a subset of the vehicle aerodynamics. Angle of attack was assumed constant at 55 degrees. Chute deploy for the was controlled by velocity trigger at 530 m/s, and opposed to the range based trigger used in later simulations. A 500-case Monte Carlo simulation was run, with dispersions on the vehicle aerodynamics, mass, and initial conditions. Earth GRAM 2007 was used to disperse the atmosphere and winds. The dispersion levels are shown in Table 7.

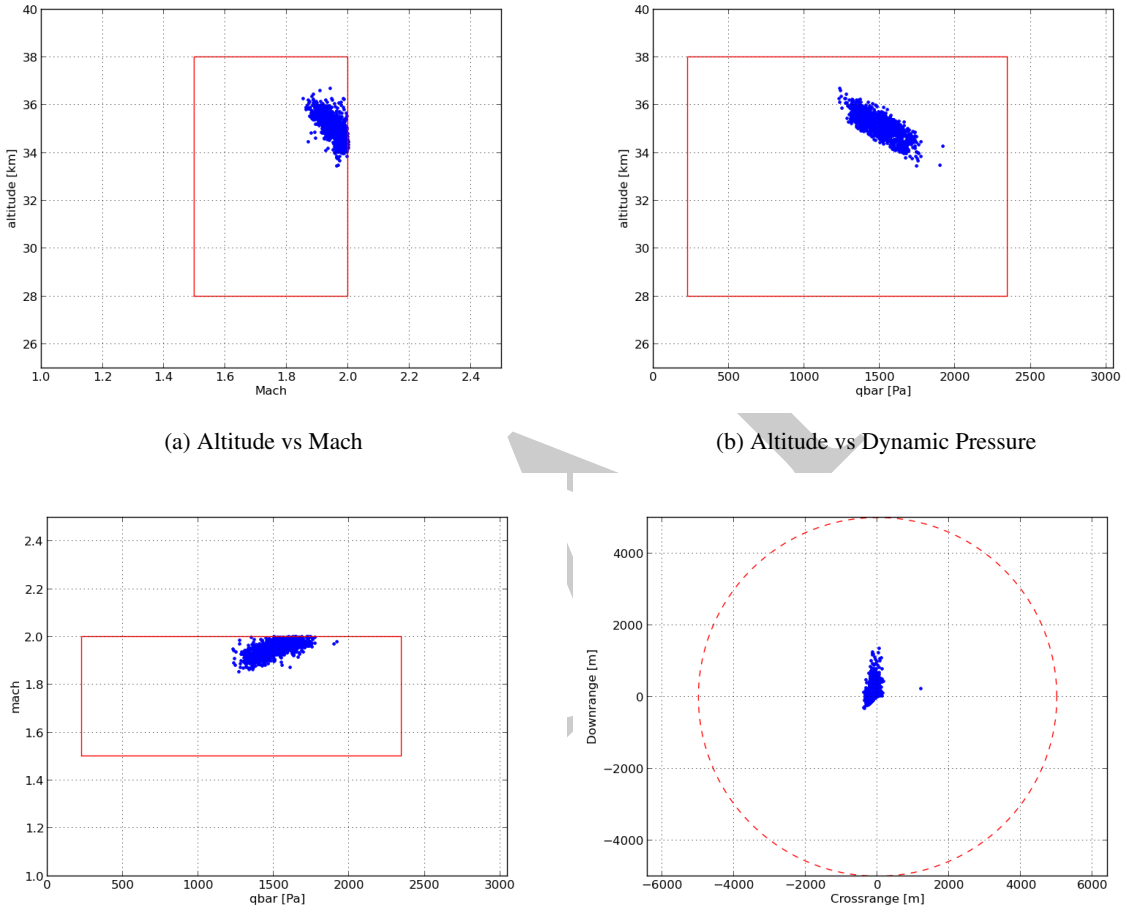
Parameter	Units	Max Dispersion	Distribution
Entry Mass	kg	2%	Normal
Entry FPA	deg	0.01	Normal
Entry Velocity	m/s	30	Normal
Initial Latitude	deg	0.3	Uniform
Initial Longitude	deg	0.3	Uniform
Initial Altitude	m	304.8	Uniform
Relative Azimuth	deg	0.007	Uniform
CL	–	10%	Normal
CD	–	10%	Normal
Winds / Atmosphere	–	GRAM 2007 dispersed	

Table 7: A list of selected dispersions for entry guidance performance using 3-DOF simulator.

Targeting performance was well within the desired 5 km for all of the Monte Carlo runs as shown in Figure 6. Mean miss distances were less than 1 km. Incorporating the improvements to the lateral logic and the deploy trigger could further improve the match with the 6-DOF simulator results.

V. Conclusions

This paper describes the implementation of an advanced entry guidance algorithm called Fully Numerical Predictor-corrector Entry Guidance (FNPEG) for a mid L/D Automated Re-entry System (ARS). It was discovered that using an appropriately tuned variable step-size numerical integration scheme can reduce the computation by up to 60–75%



(a) Altitude vs Mach

(b) Altitude vs Dynamic Pressure

(c) Mach vs Dynamic Pressure

(d) Downrange vs Crossrange

Figure 4: Conditions at supersonic drogue chute deploy.

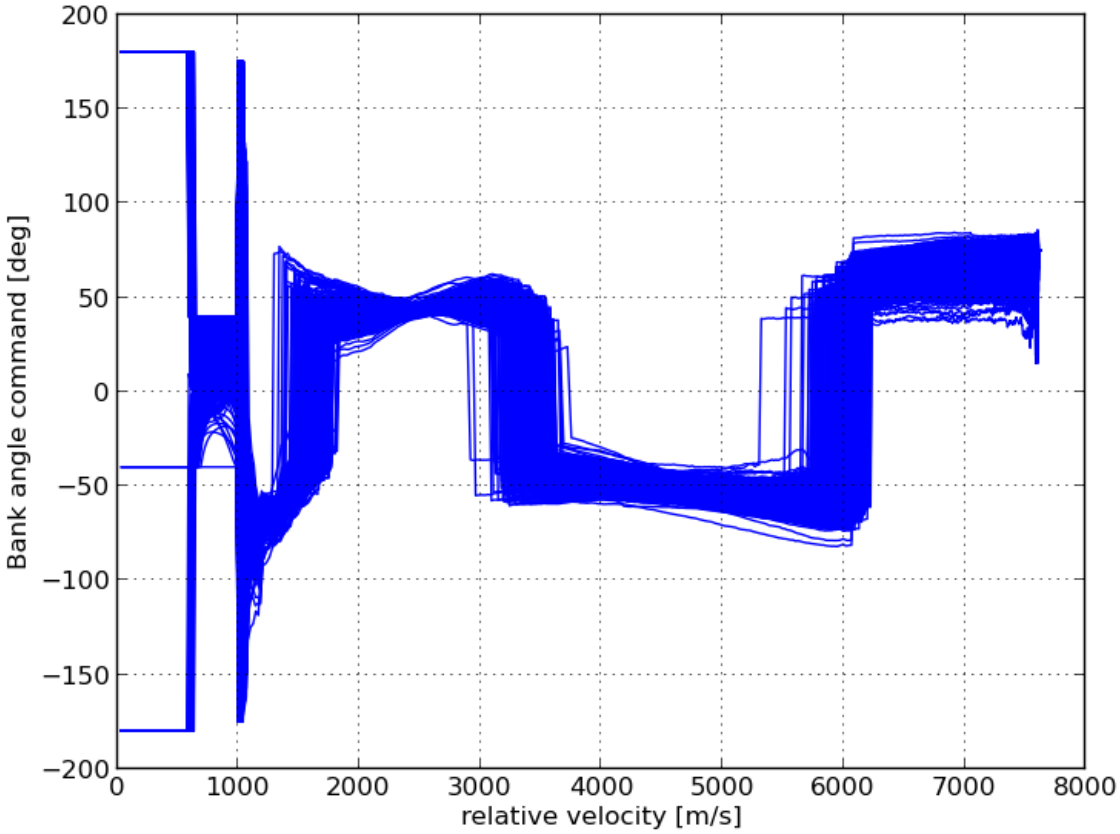


Figure 5: Commanded bank vs relative velocity

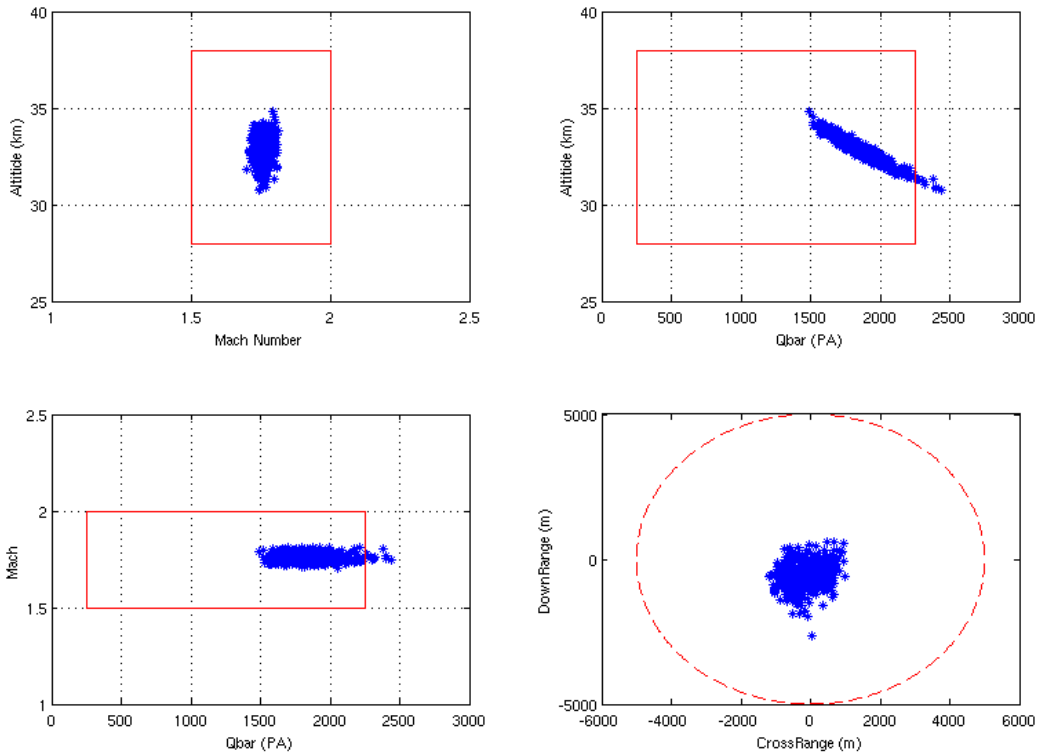


Figure 6: Commanded bank vs relative velocity

as compared to a constant-step-size integration with comparable accuracy. Such a saving is important because of the relatively intensive computational requirements of a numerical predictor-corrector entry guidance algorithm. A simple adaptive integration step size tuning defined by a safety factor appears to be effective. A lateral logic modification allows a user-specified number of bank reversals to be performed without requiring additional trajectory integrations. Monte Carlo simulation results have demonstrated the applicability of FNPEG to the ARS.

The guidance tuning is still in an early phase, and some of the issues can be alleviated. For example, the cases that deploy the chute immediately upon entering the chute deploy box result in a downrange biased landing. If this bias were accounted for, the downrange performance could be improved further. As the ARS system is flown and matures, some of the design margin can be traded for increased performance,, such as propellant saving by using fewer bank reversals, once the aerodynamics of the ARS have been better understood.

References

[1] Stewart, S., Johnson, W., Chomel, C., Tamblyn, S., and Woods, J., "GNC Design for Autonomous Payload Return from ISS," *39th Annual AAS Guidance and Control Conference*, AAS, Breckenridge, CO, 2015.

[2] Morth, R., "Re-entry Guidance for Apollo," R-532, MIT/IL, 1966.

[3] Braun, R. et al., "Mars Surveyor 2001 Atmospheric Flight Team Report," Internal Memorandum MSP-01/98-024, JPL, Pasadena, CA, Feb. 1998.

[4] Mendeck, G. F., "Mars Science Laboratory Entry Guidance," *AIAA Atmospheric Flight Mechanics Conference*, AIAA, Portland, OR, Aug 2011.

[5] Kluever, C., "Entry Guidance Performance for Mars Precision Landing," *Journal of Guidance, Control, and Dynamics*, Vol. 31, No. 6, 2008, pp. 1537–1544.

[6] Lu, P., "Entry Guidance: A Unified Method," *Journal of Guidance, Control, and Dynamics*, Vol. 37, No. 3, 2014, pp. 713–728.

[7] Lu, P., Brunner, C., Stachawiak, S., Mendeck, G., Tigges, M., and Cerimele, C., "Verification of a Fully Numerical Entry Guidance Algorithm," *Journal of Guidance, Control, and Dynamics*, 2016, DOI: 10.2514/1.G000327.

[8] Vinh, N. X., Busemann, A., and Culp, R. D., "Hypersonic and planetary entry flight mechanics," *NASA STI/Recon Technical Report A*, Vol. 81, 1980.

[9] Wilmot, J., "Implications of Responsive Space on the Flight Software Architecture," 2006.

[10] Smith, K. M., "Predictive Lateral Logic for Numerical Entry Guidance Algorithms," *26th AAS/AIAA Space Flight Mechanics Meeting*, AAS/AIAA, Napa, CA, Feb 2016.

[11] Dormand, J. and Prince, P., "A family of embedded Runge-Kutta formulae," *Journal of Computational and Applied Mathematics*, Vol. 6, 1980, pp. 19–26.

[12] Press, W. H., Teukolsky, S. A., Vetterling, W. T., and Flannery, B. P., *Numerical Recipes in FORTRAN; The Art of Scientific Computing*, Cambridge University Press, New York, NY, USA, 2nd ed., 1993.

[13] Robertson, P., "ARS R14-4 Aero Database," Private Communication, NASA Johnson Space Center, Houston, TX, Jan. 2016.

[14] Robertson, P., "ARS Bent Airframe Evaluation," Private Communication, NASA Johnson Space Center, Houston, TX, Jan. 2016.

[15] Way, D., "On the Use of a Range Trigger for the Mars Science Laboratory Entry, Descent, and Landing," *Aerospace Conference, 2011 IEEE*, IEEE, 2011, pp. 1–8.