# MONTE CARLO SIMULATION OF A TRIPLE FLYBY CAPTURE AT JUPITER USING PARAMAT

*Darrel J. Conway*

Thinking Systems, Inc.
Tucson, Arizona, USA

*Michael A. Barrucco*

Aerospace Engineer
AIAA member

## ABSTRACT

In 2014, Thinking Systems began work on a parallel processing tool that incorporates the numerical engine from the General Mission Analysis Tool (GMAT) into a system, Paramat, designed to use the processing capabilities of modern, multi-core computer platforms. This paper opens with a brief description of recent changes to Paramat. Paramat is then used to model an orbital capture at Jupiter that uses gravity assists at Callisto, Io, and Ganymede to reduce the orbital insertion costs for the capture. That mission segment is presented as a baseline trajectory for Monte Carlo analysis of the costs of the insertion sequence. Perturbations are applied to the nominal capture trajectory and to parameters related to the course correction maneuvers in order evaluate effect of maneuver modelling errors on the trajectory stability for the capture phase of the mission. Analysis of these data provide insight into the total delta-V costs and margins for the capture phase of the mission, along with an estimate of the orbit determination requirements for the spacecraft state used to plan the capture trajectory.

***Index Terms***— Monte-Carlo, Flight Dynamics, High Performance Computing, Parallel Processing, Simulation

## 1. INTRODUCTION

The General Mission Analysis Tool (GMAT) is a modern, world-class open source tool providing capabilities for spacecraft mission analysis, maneuver planning and optimization, and navigation. The tool is certified for maneuver planning in an operations environment, and is being certified for navigation in operations.

In 2014, Thinking Systems began work on a tool that uses the certified GMAT engine in a parallel processing environment. That tool, Paramat, uses the GMAT numerical engine at its core, and runs multiple GMAT missions simultaneously, accumulating the data generated from each run in a central location and presenting those data to the analysts using the system.

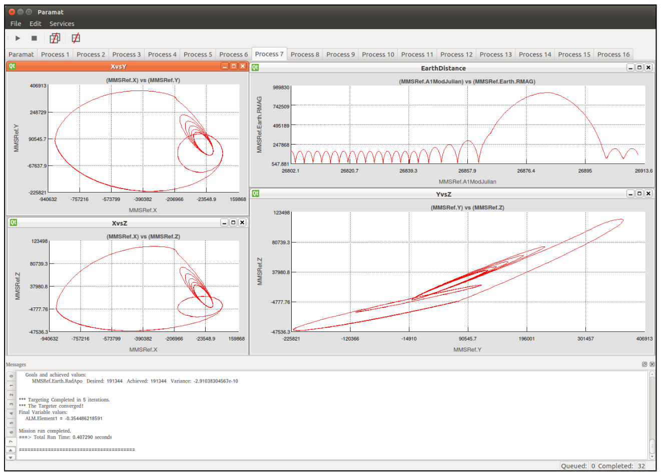
In this paper the Paramat system is described briefly, and then used to perform a piece of analysis for an orbital insertion at Jupiter that uses gravity assists from three of the Galilean moons, as described by Didion and Lynam[1]. The insertion sequence is described, and then two pieces of Monte-Carlo analysis are performed using Paramat. The first analysis examines the stability of the targeted maneuver sequence to see the effects of perturbations of the maneuvers on the insertion sequence. The second analysis perturbs the starting state of the spacecraft to generate an initial estimate of the estimation accuracy needed for the navigation components of the capture.

## 2. THE PARAMAT SYSTEM

Paramat is a prototype tool under development at Thinking Systems. The initial implementation of the tool[2] provided parallel processing capabilities to GMAT by replacing core elements of the system with thread based components. This approach proved to be difficult to maintain, requiring changes to core GMAT classes in order to add thread support, and suffering from issues of thread safety in many GMAT components. Since the initial implementation, Paramat has been reworked to run multiple copies of GMAT as separate processes without changing any GMAT code. The new system drives GMAT through a messaging API, and communicates run results to a central user interface through this custom API. Paramat supports the full functionality of GMAT as released, with no changes to GMAT source code.

Paramat is developed on Linux, and features a functional Linux GUI that is being developed to provide users with a windowed GUI experience similar to that found on the Windows releases of GMAT. The Paramat GUI supports XY parameter plotting for each running GMAT process, along with message logs for each process that capture the GMAT log messages. Thinking Systems plans to add additional features to the Paramat GUI over time so that Paramat users can fully configure their scripts inside of the system. The Paramat view into a GMAT run is shown in Figure 1. Users can toggle the GMAT graphics off in order to improve system performance during parallel processing.

The parallel processing configuration panel, shown in Figure 2, lets users select either a single script to run, or a set of scripts to run. When running multiple scripts, the user can

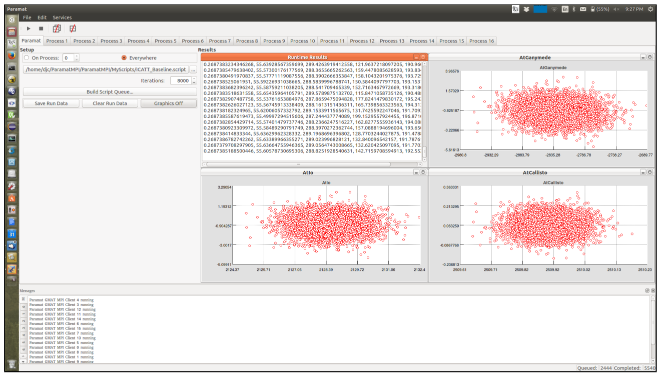


```
Goals and achieved values:
   MMSRef.Earth.RadApo  Desired: 191344  Achieved: 191344  Variance: -2.91038304567e-10


*** Targeting Completed in 5 iterations.
*** The Targeter converged!
Final Variable values:
   ALM.Element1 = -0.354486218591

Mission run completed.
===> Total Run Time: 0.407290 seconds

========================================
```

Queued: 0 Completed: 32

**Fig. 1**. GMAT Running in Paramat

also specify how many iterations should be taken for each script. During Paramat runs, this panel displays data generated from the GMAT processes either graphically or as text. For example, the data presented in the figure is a plot of the B-plane parameters at each of the three Jovian moon encounters for the orbital capture at Jupiter described below. The text panel in the figure contains the total delta-V costs for the capture, along with the altitude of the spacecraft above each moon at periapsis along the triple flyby trajectory. Thinking Systems created a GMAT plugin library that adds functions to the GMAT scripting language so that these data, or any other GMAT calculation parameter, can be selected by the analyst and displayed on the Paramat run panel.



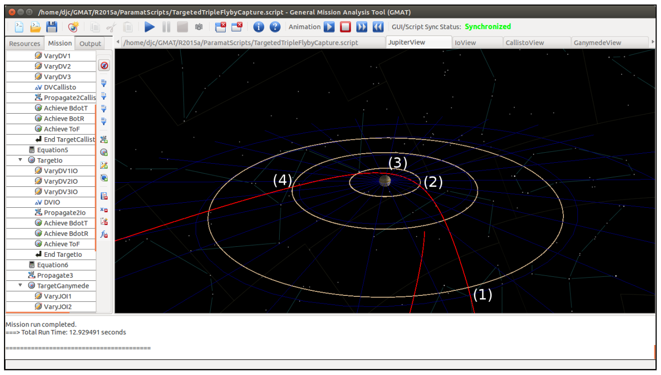
```
Paramat GMAT MPI Client 4 running
Paramat GMAT MPI Client 3 running
Paramat GMAT MPI Client 12 running
Paramat GMAT MPI Client 11 running
Paramat GMAT MPI Client 14 running
Paramat GMAT MPI Client 6 running
Paramat GMAT MPI Client 15 running
Paramat GMAT MPI Client 7 running
Paramat GMAT MPI Client 13 running
Paramat GMAT MPI Client 5 running
Paramat GMAT MPI Client 0 running
Paramat GMAT MPI Client 8 running
Paramat GMAT MPI Client 1 running
Paramat GMAT MPI Client 9 running
```

**Fig. 2**. The Paramat User Interface

The architectural change made to Paramat in 2015 is a stepping stone towards a fully networked implementation of parallel processing using the GMAT numerical engine. The Paramat messaging API is implemented using the industry standard Message Passing Interface (MPI). The use of MPI as the backbone for Paramat's messaging makes networking of the system possible, and enables larger scale parallelization of GMAT processes. Thinking Systems has begun configuration of this networking in order to further exploit the use of many processes for the problems Paramat is designed to address. The first such problem, Monte Carlo analysis using Paramat, is demonstrated using the triple assist Jovian capture example explored in the remainder of this paper.

## 3. A TRIPLE ASSIST JOVIAN CAPTURE

Didion and Lynam have studied a class of orbit insertion trajectories at Jupiter that use the Galilean moons to decrease the energy of the incoming orbit in order to reduce the insertion delta-V for the capture[1]. One such trajectory reduces the maneuver delta-V cost from 768.96 m/s to 268.75 m/s by targeting energy reducing flybys of Callisto, Io, and Ganymede. The insertion trajectory, including the three flybys and the insertion burn at Jupiter periapse, is shown in Figure 3. This trajectory starts from the initial state, expressed in Jupiter-centered Mean-of-J2000 Ecliptic coordinates shown in Table 1.



```
Mission run completed.
===> Total Run Time: 12.929491 seconds

========================================
```

**Fig. 3**. The Triple Assist Trajectory, showing flybys at (1) Callisto, (2) Io, and (4) Ganymede, along with (3) the orbit insertion maneuver at periapsis

**Table 1**. Initial State for the Simulation

| Property | Value |
|---|---|
| Epoch | 2 Feb 2025 21:19:19 UTC |
| X | -4714128.923 km |
| Y | 68943.330 km |
| Z | -60914.940 km |
| $V_x$ | 9.078 km/s |
| $V_y$ | -1.970 km/s |
| $V_z$ | 0.060 km/s |

In order to meet the design goals of the insertion, GMAT was used to target the flybys of each moon sequentially. Three maneuvers were used to perform this targeting: a maneuver at the initial state designed to reach B-plane goals at Callisto, a maneuver at Callisto periapse designed to achieve B-plane goals at Io, and the orbit insertion maneuver at Jupiter periapse, designed to achieve the Ganymede flyby that places the spacecraft in an eccentric initial Jupiter orbit with a period of approximately 200 days. The goal of the insertion for this study is to achieve an initial Jupiter orbit that can then be tuned at the first Jupiter apoapse after estimating the orbit following the Ganymede flyby. Table 2 shows the goals of the targeting used in this study, based on data presented in Didion and Lynam [3]. The B-plane goals at Ganymede were moved further from the moon that used in the reference in an attempt to prevent impacting the surface of the moon when maneuver or navigation errors are taken into account. This change in the targeting parameter at Ganymede increases the magnitude of the Jupiter orbit insertion maneuver slightly above the values found by Didion and Lynam.

**Table 2**. Design Targets for the Flybys

| Moon | UTC Epoch | $\mathbf{B} \cdot \mathbf{T}$ | $\mathbf{B} \cdot \mathbf{R}$ |
|---|---|---|---|
| Callisto | 6 Feb 2025 02:05:20 | 2510 | 0 |
| Io | 7 Feb 2025 07:02:38 | 2130 | 0 |
| Ganymede | 8 Feb 2025 03:54:36 | -2900 | 0 |

The targeting goals specified for the study were were kept relatively loose in the GMAT simulation. The Callisto and Io flyby goals targeted the $\mathbf{B} \cdot \mathbf{T}$ component – lying roughly in the orbital plane of the trajectory – to within 5 km of the goal, and the out of plane $\mathbf{B} \cdot \mathbf{R}$ component to within 10 km of the goal. The time of flight to each flyby was also constrained to achieve the flyby at set epochs, as shown in Table 2. The achieved values for the flyby epoch and maneuver magnitudes are shown in Figure 3.

**Table 3**. Event Epochs and Flybys: Achieved Values

| Event | UTC Epoch | Delta-V |
|---|---|---|
| Initial State | 2 Feb 2025 21:19:19.000 | 0.842 m/s |
| Callisto | 6 Feb 2025 02:05:20.023 | 11.985 m/s |
| Io | 7 Feb 2025 07:02:38.151 | |
| Jupiter | 7 Feb 2025 11:29:26.081 | 267.314 m/s |
| Ganymede | 8 Feb 2025 03:54:36.109 | |

The baseline trajectory represented by these data shows significant fuel savings over a direct insertion into Jupiter orbit. The cost of this insertion phase of the mission is a total delta-V of 280.14 m/s. One feature of the transfer that may be of concern is that the flyby altitudes are low at each moon. The altitudes are 55.5 km at Callisto, 287.2 km at Io – maintained moderately high due to potential volcanic activity – and 280 km at Ganymede. Given these altitudes, it is important to study the effects of maneuver errors and initial state estimation errors on the trajectory. The parallel processing capabilities of Paramat can be exploited to perform those analyses.

## 4. MONTE-CARLO STUDY OF MANEUVER ERRORS

The goal of this study is to determine the ability of the flyby sequence to achieve a Jupiter capture orbit with a period close to 200 days, using three flybys of the Galilean moons as described above, without impacting the surface of any of the moons. The maneuvers were treated as predetermined by the nominal orbit plan in the preceding section. For this study, the maneuver sequence was not retargeted following each burn. The time between the first and second maneuver in the mission timeline is just over 3 days, and between the second and third maneuver, a bit over 1 day 9 hours, with an Io flyby in between. The alternative approach to planning the maneuvers at the start and not retargeting, in practice, would require an operations team to reestimate the state after each maneuver, then replan the next maneuver, validate it, upload the command to the spacecraft on its approach to Jupiter (flying 38 light minutes away from Earth for this trajectory), and then validating a successful upload. Replanning in this scenario seemed to be undesirable unless the maneuver error analysis forced it on the operations plan. Therefore, the stability characteristics of the planned orbit due to maneuver inefficiencies and modelling errors are examined in this section, and the stability due to initial state errors in the next.

The maneuver modeling error case examined here performs runs designed to perturb each of the three maneuvers computed for the trajectory in order to determine the effect of maneuver magnitude dispersions on the flyby altitudes and the final orbit. While the three maneuvers required to achieve the nominal trajectory described above each require that the maneuver be pointed in a very precise direction in order to reach the next flyby target, attitude error modelling issues are not considered in this preliminary study. The maneuvers targeted for the nominal trajectory and used in this study are shown in Table 4. The maneuvers are expressed in GMAT's Jupiter-centered velocity-normal-binormal coordinate system.

**Table 4**. Nominal Targeted Maneuvers

| Maneuver | Delta-V | | |
|---|---|---|---|
| | $V_x$ | $V_y$ | $V_z$ |
| Initial | 0.455 m/s | -0.623 m/s | 0.338 m/s |
| Callisto | 1.028 m/s | -11.698 m/s | -2.394 m/s |
| JOI | -250.918 m/s | 58.076 m/s | 71.582 m/s |

In order to understand the effects of maneuver modelling errors on the stability of the trajectory, the magnitude of each maneuver was varied randomly as follows. The modelling for each maneuver was implemented using a multiplicative thrust scale factor, applied to each nominal maneuver. That scale factor, set to 1.0 for the nominal trajectory described in the previous section, was then perturbed using a Gaussian distribution function.

Paramat has a plugin component built for use perturbing object parameters. The tool uses the robust random number tools introduced in the C++ 2011 standard, and implements both Gaussian and uniform distribution models. Paramat's Gaussian model is built on the standard library's normal distribution class, using the 19937-bit Mersenne twister pseudorandom generator seeded off of random numbers generated in Paramat's controlling interface, which in turn is seeded by the system clock. Paramat also has an option to seed the random number system using a user specified seed value in the implementation. That option was not used in this study. Perturbation values are specified in Paramat by scripting the mean and standard deviation of the perturbation for the Gaussian distribution. (For the uniform distribution, users specify the minimum and maximum values of the distribution.) For this study, the Gaussian distribution used for the thrust scale factor for each maneuver used the values shown in Table 5.

**Table 5**. Thrust Scale Factor Perturbations

| Maneuver | Mean Value | Standard Deviation |
|---|---|---|
| Initial | 0.999 | 0.001 |
| Callisto | 0.995 | 0.005 |
| JOI | 0.995 | 0.005 |

Paramat ran 5000 iterations of the mission using these perturbations. The total wall clock runtime for these iterations was 18 minutes. Because of the sensitivity of the trajectory to perturbations at the manuever points for this model, the flyby most likely to suffer an impact event is the last flyby, at Ganymede. No other impacts were observed in the runs performed for this study. Figure 4 shows the distribution of altitudes at Ganymede periapsis for these runs.

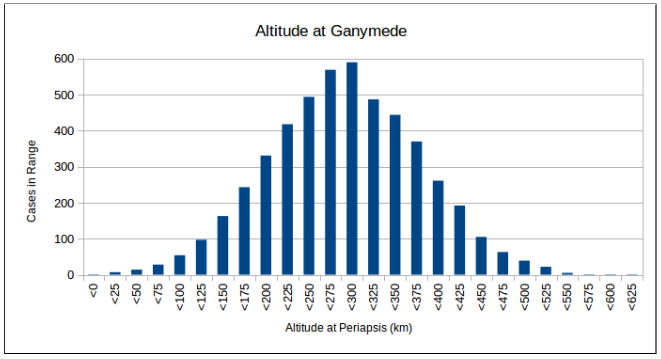


**Fig. 4**. Ganymede Altitude Distribution for Thrust Scale Factor Perturbations

One of the modelled trajectories impacts the surface at Ganymede in this data set. For other 5000 iteration runs performed for this model, the impact count ranged from zero to 4 impacts at Ganymede. No other potential impacts were observed in any run set, and the post-insertion orbit period was stable, ranging from 184.7 days to 230.0 days, with a mean value of 206.2 days for this set, and with nearly identical values for other sets. Table 6 shows the altitude statistics at each periapsis for the data set shown here.

**Table 6**. Data for the Thrust Scale Factor Perturbations

| Body | Mean Alt. | Std. Dev. | Min | Max |
|---|---|---|---|---|
| Callisto | 55.5 km | 0.148 | 55.0 | 56.0 |
| Io | 287.3 km | 2.169 | 279.6 | 295.0 |
| Jupiter | 2.232 $R_J$ | 0.0001 | 2.232 | 2.233 |
| Ganymede | 278.6 km | 88.0 | -9.1 | 606.5 |
| Period | 206.2 days | 6.253 | 184.7 | 230.0 |

Perturbing the thrust scale factor provides an initial analysis of the maneuver fidelity needed to fully study this orbit capture problem. A more thorough analysis of maneuver errors would include perturbations of the burn direction in order to characterize the effects of pointing offsets for the burns. A preliminary check for the effects of pointing offsets produced results similar to those seen with the thrust magnitude study here: with relatively tights control on the maneuver direction, the transfer is stable for most cases of attitude errors, but can result in impacts at Ganymede for the analysis outliers. A more complete analysis is needed before those results can be fully characterized and presented.

## 5. DEFINING THE NAVIGATION REQUIREMENTS

Accurate maneuvering reflects one piece of the error analysis needed to evaluate the requirements for this capture trajectory. The accuracy needed for the estimated state at the start of the trajectory must also be evaluated. Small errors in the initial state are magnified at each flyby. This effect makes evaluation of the state accuracy simple. As long as the spacecraft is captured into orbit at Jupiter, the estimation requirements can be evaluated initially by examining the altitude of the spacecraft at the final flyby, at Ganymede. For the purposes of this analysis, the goal is to minimize the likelihood of impact at Ganymede. In order to develop estimation requirements for this trajectory, a small sampling of possible error envelopes can be run first to determine a rough range for the state estimation requirements, followed by a more refined analysis of the state accuracy requirements.

### 5.1. Coarse Analysis

In order to determine how close the initial state needs to be to its nominal value, Paramat was configured to run seven small Monte Carlo sets of 1000 trajectories each, and the altitude at the Ganymede flyby was examined to determine the viability of the estimation tolerances for each of these cases. The estimation accuracies, expressed as the standard deviation value input for each component of the initial state for the position and velocity used in each of the seven runs are shown in Table 7.The table shows the resulting average altitude at Ganymede, the standard deviation of the altitude envelope, and the number of impacts observed for the 1000 iteration run.

**Table 7**. Coarse Analysis of State Estimation Precision Affects

| Position | Velocity | Altitude | Std. Dev | Impacts |
|---|---|---|---|---|
| 250 m | 5 mm/s | 202.8 km | 724.2 | 385 |
| 250 m | 1 mm/s | 224.8 km | 179.8 | 108 |
| 100 m | 5 mm/s | 218.8 km | 690.7 | 379 |
| 100 m | 1 mm/s | 219.7 km | 133.4 | 56 |
| 50 m | 5 mm/s | 255.4 km | 665.4 | 352 |
| 50 m | 1 mm/s | 214.5 km | 137.4 | 56 |
| 50 m | 0.5 mm/s | 219.9 km | 71.6 | 1 |

As expected, imprecision in the estimated state at the start of the run leads to an increased probability of impact at Ganymede. For the test cases examined here, the likelihood of impact at Ganymede was unacceptable for all but the final case in the table, requiring an estimated state to within 50 km in each position component and to within 0.5 mm/s in each velocity component. That case, and one with a tighter velocity tolerance, are examined next.

### 5.2. Refined Analysis

Starting from the final sample in the preceding section, the next piece of analysis performed examines the effects of errors in the initial state data at each flyby in order to begin to define requirements for the navigation requirements for the Jupiter capture. Paramat was used to perturb the six Cartesian state elements of the initial state in Table 1 using the Gaussian parameters shown in Table 8. Two cases were run for this part of the study. In both cases, the position perturbations of the orbit state were set to center on the nominal state in Table 1 with a 50 meter (standard deviation) Gaussian distribution. In the first set of runs, the velocity distribution had a standard deviation of 0.5 mm/s. The second run tightened the acceptable accuracy of the velocity estimate to 0.25 mm/s.

**Table 8**. Estimated State Accuracies

| State Components | Estimation Standard Deviation |
|---|---|
| Position | 50.0 m |
| Velocity, Case 1 | 0.50 mm/s |
| Velocity, Case 2 | 0.25 mm/s |

Paramat was configured to run 5000 trajectory iterations for each of these distributions. Figure 5 shows the distributions of closest approach altitudes at Ganymede for this pair of runs. The first test, represented by the blue bars in the figure, resulted in six impacts at Ganymede. Tightening the velocity component of the navigation requirement corrected the impact probability, resulting on no impacts, as is seen in red bars in the figure.

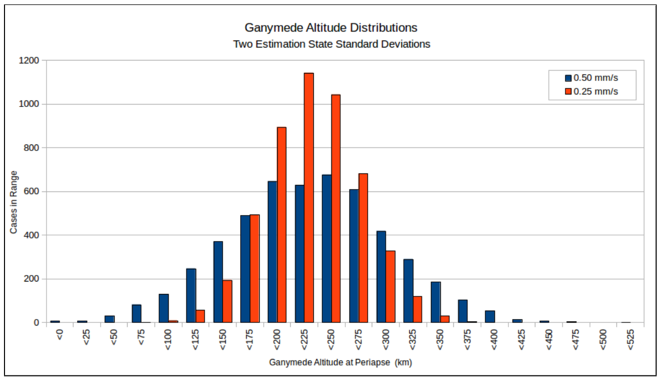


**Fig. 5**. Ganymede Altitude: Loose and Tight Velocity Estimates

Table 9 shows the periapsis distances and orbital period values for the second navigation test case. For this test case, there are no impacts, the target orbital period goal is met, and that altitude at each moon is acceptable for flying the mission if all maneuvers are preformed with the values computed in the targeting run defining the nominal mission.

**Table 9**. Orbit Event Statistics for Navigation Case 2

| Body | Mean Alt. | Std. Dev. | Min | Max |
|---|---|---|---|---|
| Callisto | 55.7 km | 0.084 | 55.4 | 56.0 |
| Io | 290.0 km | 1.184 | 286.1 | 294.1 |
| Jupiter | 2.233 $R_J$ | 6.9 x $10^{-5}$ | 2.233 | 2.233 |
| Ganymede | 218.9 km | 42.466 | 73.1 | 355.7 |
| Period | 199.6 days | 3.275 | 188.3 | 210.1 |

The final set of runs performed for this analysis combined the navigation perturbations with thrust scale factor perturbations to examine the total effect of imprecision in the estimated state and maneuver magnitude on the Jupiter capture trajectory.

## 6. COMBINING MANEUVER AND STATE ERRORS

For the final set of runs for this paper, the thrust scale factor variations in the manuever at the start of the trajectory are replaced by variations in the initial state for the capture trajectory. Variations in the thrust scale factor for the maneuver at Callisto and at the Jupiter periapsis are modeled, and the resulting statistics compiled to determine the stability of the orbit subject to all of the sources of potential error in the maneuver plan. Paramat is again used to perform a 5000 iteration run of the system. The thrust scale factor at the initial state is not perturbed in these runs. Maneuver errors at the initial state burn can be thought of as being incorporated into the error in the initial state at the start of the propagation.

**Table 10**. Initial State and Thrust Scale Factor Perturbations

| Components | Value | Standard Deviation |
|---|---|---|
| Position | Init. State | 50.0 m |
| Velocity | Init. State | 0.25 mm/s |
| Callisto Mnvr TSF | 1.0 | 0.005 |
| Jupiter Mnvr TSF | 1.0 | 0.005 |

Once again, the only potential impact in this set of runs occurs at the Ganymede flyby. One sample case from the 5000 iteration run resulted in an impact at Ganymede, with a periapsis altitude 5.8 km below the surface. The sampling distribution for the spacecraft's altitude at the Ganymede flyby is shown in Figure 6. In the 5000 trajectory sample there are four encounters below 25 km in altitude: the impact mentioned above, and flybys at 7.3 km, 8.1 km and 22.4 km altitude.

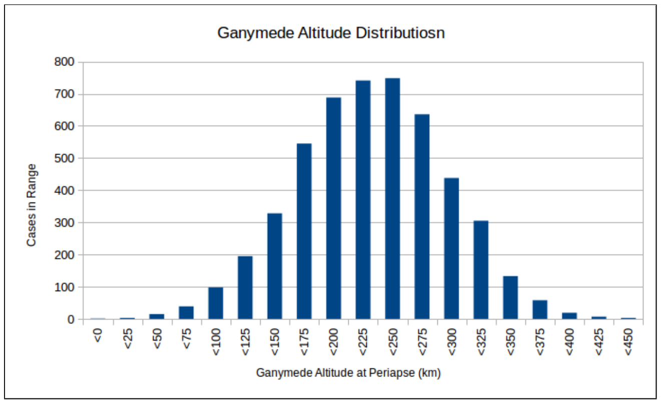


**Fig. 6**. Ganymede Altitude: Trust Scale Factor and Estimation State Errors

Statistics for the flyby events for this model are fall between the results observed for the maneuver error modelling run and for the navigation error run, as can be seen in Table 11. This is expected because the incorporation of the thrust scale factor perturbation into the initial state estimation reduced the total error in the model because of the sensitivity of the subsequent flyby encounters to the gravity assist provided by the Callisto flyby.

**Table 11**. Orbit Event Statistics for the Full Model

| Body | Mean Alt. | Std. Dev. | Min | Max |
|---|---|---|---|---|
| Callisto | 55.7 km | 0.084 | 55.4 | 56.0 |
| Io | 290.0 km | 1.330 | 285.2 | 294.3 |
| Jupiter | 2.233 $R_J$ | 8.0 x $10^{-5}$ | 2.233 | 2.233 |
| Ganymede | 219.4 km | 63.346 | -5.8 | 443.0 |
| Period | 199.6 days | 3.978 | 185.5 | 213.4 |

## 7. CONCLUSIONS

The analysis performed here is preliminary, but does provide guidelines that could be used to specify mission requirements for a Jupiter mission that uses a triple flyby sequence at the Galilean moons to reduce the fuel costs of the capture sequence. Based on the preliminary runs performed here, the navigation requirements for the state estimate for this sequence are tight: each component of the Jupiter-centered spacecraft Cartesian position needs to be determined to within 50 meters of the actual position, with velocity components estimated to with 0.25 mm/s. In addition, the maneuvers need to be modelled to within about 1% of their true values. As these errors grow, the likelihood of an impact at the final encounter becomes unacceptable.

The timing of the maneuver flybys make state estimation from the ground difficult after the initial flyby, at Callisto for the case studied here. For this trajectory, the Io flyby occurs nearly one day and five hours after the Callisto flyby, and the orbit insertion burn is performed less than 4.5 hours later. Precise replanning on the ground of the orbit insertion maneuver following the Io flyby is therefore infeasible. However, if the estimated state and maneuver control can be maintained to the envelopes described above, the capture trajectory is feasible and has a high likelihood of success.

Further options can be explored for this approach to spacecraft capture at Jupiter. For example, if the orbital period following the capture is relaxed, the mission can be retargeted to keep the spacecraft further from Ganymede at the last flyby, further reducing the likelihood of impact at the cost of larger subsequent orbit tuning maneuvers after entering orbit.

## 8. PARAMAT STATUS

The tool used for this study, Paramat, is a system under development at Thinking Systems. Initial work on Paramat, funded through a NASA SBIR contract, used a threading model coded into core components of GMAT to produce a single application running multiple GMAT simulations in parallel. Technical issues with that implementation made the system difficult to maintain because of the changes needed to the core GMAT code base to make the threading approach work.

The current implementation of Paramat is implemented using the released GMAT R2015a code base for Linux and Mac, unchanged from the packaging on SourceForge[4]. Paramat launches GMAT processes as separate executables and collects data from the running processes as scripted by the user. The resulting data is shown to the user as either data on the screen or in plots. All data can be saved for further analysis and other post processing tasks.

The analysis performed in this paper demonstrates the usefulness of Paramat in mission analysis. The section of the paper describing the coarse OD analysis was written to address a question raised on review of the first draft of the paper. Seven runs of the system were performed to generate the data presented there. Each run consisted of 1000 precision orbit propagations of the triple flyby trajectory, accomplished in three and a half minutes of elapsed time from the start of Paramat through the collection of the data. Essentially, in less than half an hour total elapsed time, 7000 trajectories were run, and the resulting data collected, allowing for evaluation of seven different navigation requirements sets.

Paramat is currently an internal tool at Thinking Systems. The system is not yet ready for public distribution. Interested parties are encouraged to contact the authors for additional information.

## 9. ACKNOWLEDGEMENTS

Initial funding for Paramat provided by the NASA Small Business Innovative Research (SBIR) program[5] is gratefully acknowledged. Paramat is built on the code base of the General Mission Analysis Tool[6]. The triple flyby capture used as a sample problem here was studied and documented by Didion and Lynam[1][3]. Any abuses of their work are, of course, the fault of the authors.

## 10. REFERENCES

[1] A.M. Didion and A.E. Lynam, "Impulsive Trajectories from Earth to Callisto-Io-Ganymede Triple Flyby Capture at Jupiter," in *AIAA/AAS Astrodynamics Specialist Conference*. AIAA/AAS, 2014.

[2] D.J. Conway, "Paramat: Parallel Processing with the General Mission Analysis Tool," in *AIAA/AAS Astrodynamics Specialist Conference*. AIAA/AAS, 2015.

[3] A.M. Didion and A.E. Lynam, "Guidance and Navigation of a Callisto-Io-Ganymede Triple Flyby Jovian Capture," in *AIAA/AAS Astrodynamics Specialist Conference*. AIAA/AAS, 2015.

[4] The GMAT Development Team, "GMAT: General Mission Analysis Tool," 2016.

[5] NASA SBIR Program Office, "NASA SBIR and STTR Program," 2016.

[6] The GMAT Development Team, "GMAT Wiki," 2016.