Efficient Ephemeris Models for Spacecraft Trajectory Simulations on GPUs

Fabian Schrammel[1,2,3], Florian Renk[2], Arya Mazaheri[1] and Felix Wolf[1]

[1] Technische Universität Darmstadt, Germany
[2] European Space Agency (ESA), Darmstadt, Germany
[3] GMV GmbH, Darmstadt, Germany
fabian.schrammel@gmv.com, florian.renk@esa.int,
{mazaheri,wolf}@cs.tu-darmstadt.de

Abstract. When a spacecraft is released into space, its initial condition and future trajectory in terms of position and speed cannot be precisely predicted. To ensure that the object does not violate space debris mitigation or planetary protection standards, such that it causes potential damage or contamination of celestial bodies, spacecraft-mission designers conduct a multitude of simulations to verify the validity of the set of all probable trajectories. Such simulations are usually independent from each other, making them a perfect match for parallelization. The European Space Agency (ESA) developed a GPU-based simulator for this purpose and achieved reasonable speedups in comparison with the established multi-threaded CPU version. However, we noticed that the performance starts to degrade as the spacecraft trajectories diverge in time. Our empirical analysis using GPU profilers showed that the application suffers from poor data locality and high memory traffic. In this paper, we propose an alternative data layout, which increases data locality within thread blocks. Furthermore, we introduce alternative model configurations that lower both algorithmic effort and the number of memory requests without violating accuracy requirements. Our experiments show that our method is able to accelerate the computations up to a factor of 2.6.

Keywords: GPU · simulation · profiling · astrodynamics.

1 Introduction

In space mission design astrodynamics simulations are instrumental in determining the probabilities of spacecraft and space debris trajectories. At the point of release or in-orbit failure, the position and speed of the object as well as the properties (e.g. surface reflectivity) are only known to the mission architects only with a certain precision. Hence, Monte-Carlo simulations containing thousands of object samples are conducted in during the mission preparation phase. Based on the results, the team will choose a nominal separation state or trajectory that satisfies the rules of the current planetary protection and space-debris mitigation guidelines. For instance, during the *BepiColombo* mission [2], currently being flown to Mercury by the European Space Agency (ESA), the upper stage of the Ariane 5 launcher places the mission spacecraft into the desired transfer orbit before release. After the required passivation of the upper stage to prevent further break-up, this object becomes space debris and it is not allowed to impact e.g. Mars or return to near-earth space by a certain probability.

To verify cases like this, the European Space Agency (ESA) has recently designed a tool called *cudajectory* to run such simulations on CUDA-capable devices and achieved a speedup ranging from 1.9× to 10.5× in comparison with established multi-threaded solutions on CPUs [5]. However, we found out that the GPU implementation can benefit from further improvements.

To accelerate such highly parallelizable simulations, we analyze the state of the art and suggest alternative methods to gain performance without violating accuracy requirements. The simulations are typically performed using numerical integration methods, such as the Runge-Kutta-Fehlberg 78 scheme [3]. The use of variable step integration methods is extremely efficient for spacecraft trajectories, since the step size can vary from days in interplanetary space to only seconds, when the object moves very close to a celestial body. In such numerical integration methods, the equations of motions are implemented, and force or acceleration models are used. One of these models is the ephemeris model, which we will focus on in this paper since its performance is often bound by memory. The ephemeris model provides the position of the celestial bodies at a given epoch/time and allows us to derive the gravity field affecting the spacecraft trajectory. For the trajectory of a single object being calculated, the simulation will sequentially go through the ephemeris calculations as the simulated time progresses. However, when different spacecraft or object samples are simulated in parallel, this is no longer the case. Even if the samples have the same initial time, which is not the case in all problems, the integration steps can have different lengths. Thus, each simulated object requires the positions of the celestial objects at a different epochs. In a different problem to be investigated a spacecraft failure shall be simulated along the nominal flight path and thus already the initial epochs of all samples are different. Such initial difference or divergence during the integration process leads to different sets of data requested by the ephemeris routines, overloading the on-chip memory, which indeed results in register-spilling. Therefore, memory bandwidth becomes a bottleneck and decreases the overall performance tremendously.

In this paper, we propose an alternative data layout for the ephemeris data. This new data layout improves data locality. The ephemeris data is restructured from a memory layout optimized for sequential processing to a layout more suitable for parallel processing. We increase the likelihood that the required ephemeris data is available in the caches for several threads running on the GPU, thus preventing threads from stalling. Additionally, we were able to shrink the ephemeris model while maintaining the required accuracy. First of all, some celestial bodies may exert forces small enough to be disregarded. Then, planetary systems can be handled as a single body adding a small error. Lastly, a different type of function can be used to approximate the movement of these bodies.

Originally, Chebyshev polynomials [4, 12] with a degree between 6 to 14 were being used in ephemeris models. However, previous experiments [6, 15] already demonstrated that cubic splines are more memory efficient and straightforward while providing reasonable accuracy. Such optimizations reduce the algorithmic effort and the number of data requests, thus improving data locality. In essence, this paper makes the following major contributions:

- A novel data-locality aware data structure to hold ephemeris model data
- A method for balancing the trade-off between simulation accuracy and speed

In the remainder of the paper, we first provide background on the spacecraft trajectory simulation using GPUs. Then, in Section 3, the effect of alternative model configurations is analyzed to identify further case-dependent optimizations, followed by an evaluation in Section 4. A concise review of related work is presented in Section 5. Finally, we conclude the paper in Section 6.

2 Background on Astrodynamic Simulations

When objects in space are passivated, the point in time, position, and speed are only known with a certain precision. For in-orbit failures, however, the problem is more random as they can occur at any point during the mission. In addition other uncertainties can occur, e.g. the surface reflectivity of an object and thus the solar radiation pressure acting on will depend on the future attitude of the S/C and the properties can also be determined only with a specific accuracy prior to launch. Tiny deviations of the state parameters can lead to a significant difference in the trajectory after years and decades of space travel. Therefore, the Monte-Carlo method [7] is applied beforehand to generate a set of sample states and propagated forward in time to generate their trajectory path. Depending on the case, these samples may be located around an initial guess, as presented in the first picture of Figure 1, or along a specifically planned and controlled trajectory. Once a sufficient amount of such samples (often up to hundreds of thousands) is simulated for an appropriate period, we can produce a meaningful probabilistic result from the predicted trajectories, such as impact probability to a specific celestial body.

2.1 *cudajectory*: ESA Tool for Trajectory Simulations on GPUs

A spacecraft trajectory can be simulated step by step using numerical integration methods. Within each step, the change in position and velocity of the spacecraft is calculated by applying a physics model. ESA developed an in-house tool called *cudajectory* [5], solely designed to simulate the trajectories of a set of initial spacecraft states, a.k.a samples. The tool parallelizes the trajectory simulations by starting one GPU thread per sample, as described in Figure 1. They are numerically integrated until every simulation of a sample reaches the end of a fixed simulation period or collides with a celestial body. The Runge-Kutta method [8] of seventh order is used for step-wise integration, and the eighth

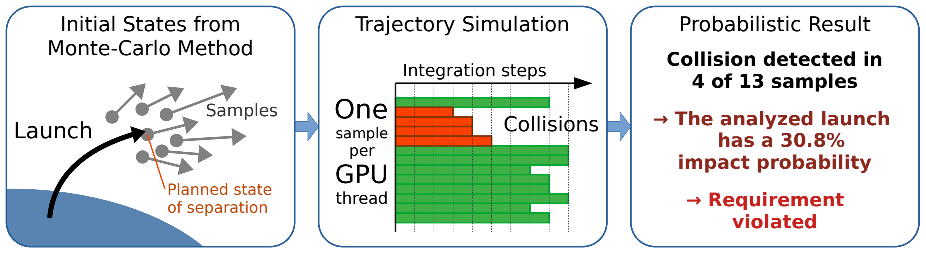
Initial States from Monte-Carlo Method
Launch
Samples
Planned state of separation
Trajectory Simulation
Integration steps
One sample per GPU thread
Collisions
Probabilistic Result
Collision detected in 4 of 13 samples
→ The analyzed launch has a 30.8% impact probability
→ Requirement violated

Fig. 1: Example of the main aspects of a collision analysis of space debris after separation. In a first step samples are generated, which need to be propagated in time in a second step and the results need to be analyzed in a third step. The second part of this process is implemented by cudajectory.

order is added via the Fehlberg method [3] to apply dynamic step-size control. Here, each step contains 13 evaluations of the ordinary differential equation (ODE) on eleven different points in time. Each ODE evaluation applies the physics model, which includes routines to calculate gravitational forces, solar radiation pressure, and collision detection regarding the nearby celestial bodies. All of these calculations require the position of one or more celestial bodies at the current time, retrieved from an ephemeris model.

An ephemeris is a collection of models and values that can describe the position and velocity of astronomical objects over specific periods. Releases from the Jet Propulsion Laboratory (JPL) are known to be the most accurate models nowadays, and the applied data format (*Type 2*) is widely used in the industry [10,11]. These models contain functions of time returning the three-dimensional cartesian position of a body. Chebyshev polynomials are the method of choice for high-precision orbit approximation (See Figure 5) as they are best suited in terms of accuracy, interpolation error, and applicability [12]. For each body, a series of polynomials of fixed interval length and polynomial degree is provided to approximate its orbit over the simulated period. Only the coefficients of each polynomial will be stored in program memory, which are applied during position calculations.

DE432 is the latest release by JPL [4] and serves as a baseline during our research. It covers eleven major celestial bodies and planetary systems of the solar system, where the center of mass (barycenter) is used to include moons.

2.2 State-of-the-Art Performance

Experiments show that the current implementation of cudajectory can be about 10× faster than established multi-threaded CPU solutions on different types of input samples and physics models. However, we noticed that one major performance bottleneck of cudajectory happens for samples at very different points of simulation time [5]. The step sizes applied during the samples for the Bepi-Colombo case range from several seconds to almost nine days. In the main implementation, GPU threads are divided into fixed groups called warps in CUDA terms, each executing in parallel. When the range of the timestamps within a

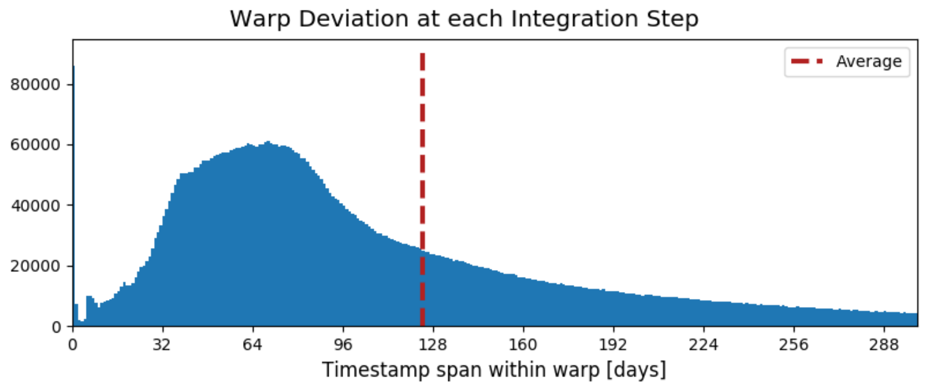

Fig. 2: Distribution of the time difference between the simulation epoch at each integration step within warps of the BepiColombo case example. The maximum range observed difference was 2352 days within a warp. A difference of 0 days indicates that all threads can use the same ephemeris data block.

warp increases due to different start times or dynamic step size control, the threads require different sets of ephemeris polynomials to calculate the position of a specific celestial body. These differences in time can be significant as depicted in Figure 2. If not all data can be made available the whole warp stalls until all of these polynomials have their coefficients data ready. Therefore, when this situation occurs the memory traffic is immensely increased and leads to significant performance degradation.

3 Efficient Ephemeris Formats and Configurations

We analyze the original record-based ephemeris data format on GPUs and propose an alternative data format to improve performance. This format stores the polynomial data in a different order and offers the opportunity to apply cubic splines instead of the current Chebyshev polynomials. Finally, we present additional ways to reduce algorithmic effort, as well as data requirements.

We profiled the performance of the BepiColombo case, running 420,000 threads packed in 13,125 warps on a Tesla V100 using the Nvidia Visual Profiler [14]. The results showed that 95.7% of the memory traffic is linked to local-memory instructions. This is a strong hint to the existence of excessive register spilling, as this memory space can not be utilized manually. Instead, the program automatically includes such instructions to spill and reload the register data. We also identified a high execution efficiency of over 98%, indicating almost no processor idle time. However, such a large fraction of executed instructions is likely linked to register spilling, which reduces the overall efficiency. To alleviate the register spilling, the overall memory traffic must be reduced. Therefore, we apply a different data alignment and memory access pattern, which helps us to improve data locality.

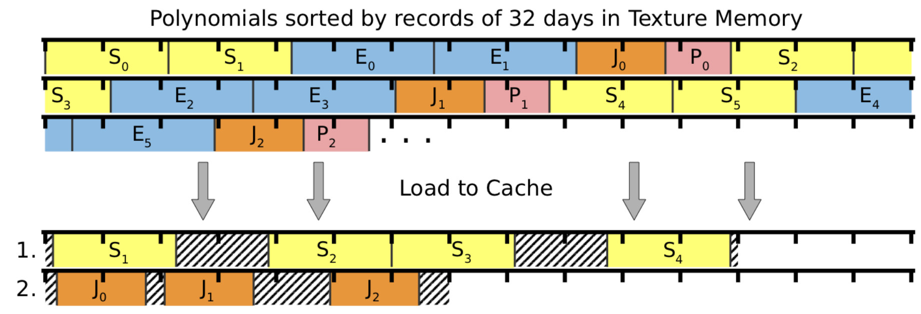

Fig. 3: Two caching examples for DE432 data stored in record-based format, where e.g. the first record spans from S_0 to P_0. We depict data of polynomials for the Sun (S_i), the Earth (E_i), Jupiter (J_i) and Pluto (P_i). Multiple polynomials of the same body are loaded in the cache lines of 128 bytes (between two black ticks) [13]. The loaded but unneeded data in the cache is displayed by diagonal grey stripes.

The data format of DE432 is designed to improve the data locality of single-thread execution. All data required for the position calculations of a specific point in time is collected in one record, which covers 32 days. This method increases the spatial locality, as some coefficients of the latter bodies in the list are pre-cached by requests to earlier bodies. Since the move to the subsequent point in time rarely exceeds 7.0 days, the ephemeris data valid for the previous timestamp will often be reused, which provides temporal locality. Figure 3 describes DE432 data stored in the GPU texture memory. As the data size for each polynomial is often not a multiple of the cache line size, they will not be stored at the start of a cache line. When a warp requests the polynomial coefficients for one specific body covering a specific period, the relevant cache lines are loaded into the on-chip cache. The first request targets four Sun polynomials of 1056 bytes, for which twelve cache lines are loaded, although they would fit into nine. This results in 1536 bytes loaded, which is roughly 45% more than requested. In the second example shown in Figure 3 regarding Jupiter polynomials, 40% more cache lines containing 56% more data than needed are loaded. When all polynomials within the records are applied at some point during the calculations, a fraction of the unneeded data may be used for a different body or point in time. If cached until this point, the data is then immediately available. However, getting a warp instruction ready for execution will generally involve more memory traffic than theoretically necessary. Additionally, a cache overload will replace former cached polynomials, alleviating both spatial and temporal locality effects.

3.1 CUBE: CUdajectory Binary Ephemeris format

An efficient ephemeris data format for massively parallelized cudajectory requires a different view of data locality. Instead of looking at the locality within a single thread, we need to focus on the data that is requested at once by the threads of one warp. By fitting the data's alignment to the access pattern, we can reduce the memory traffic and register spilling.

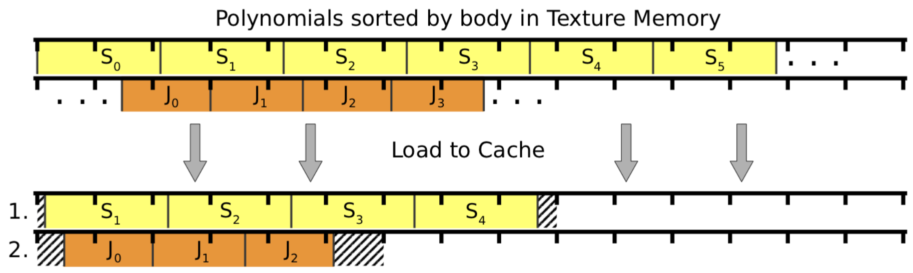

Fig. 4: Same ephemeris model data and example requests as in Figure 3 but with the polynomials sorted first by body and then by time.

The threads within a warp will all perform the position calculation for the same body by design of the software. In case they evaluate the same polynomial, they will also need the same data. For higher timestamp ranges, however, this ranges over multiple polynomials. Therefore, storing the list of polynomials sorted first by the body and then by timestamp will result in a more efficient format than the original record-based approach. This is the exact idea behind the proposed CUBE format.

Figure 4 describes the same data requests as in Figure 3, but now the polynomials are stored in the CUBE format. For the Sun polynomials, we load the minimum necessary number of cache lines, where only 9% more data than needed is included. In the case of Jupiter, still, one more cache line than theoretically necessary is loaded, which is caused by the alignment of the polynomials to the cache lines in texture memory. However, we still perform much better compared to the record-based format because only 20% more cache lines and 33% more data than needed are loaded. Overall, we achieve a 21% reduction of loaded cache lines (From 19 down to 15) by merely changing the alignment of the data, promising a notable performance gain.

Furthermore, we identified a difference in the storage structure of coefficients within each polynomial and the order of accesses by the cudajectory implementation. Thus, there is a chance for additional performance improvement when either the algorithm is adjusted, or the CUBE format is further improved.

3.2 Alternative Models and Configurations

Using cubic splines; Popular types of ephemeris models apply Chebyshev polynomials [10,12] to approximate the position and velocity of celestial bodies. However, Korvenoja et al. [6] showed that ephemeris models for satellite orbits could be computed using cubic splines, achieving high accuracy with significantly lower effort. Moreover, Russell and Arora [15] demonstrated that this technique could also be applied to ephemeris models of celestial bodies. When an alternative model is applied, the deviation in the calculated body positions affects the direction and magnitude of the gravitational pull exerted on the simulated

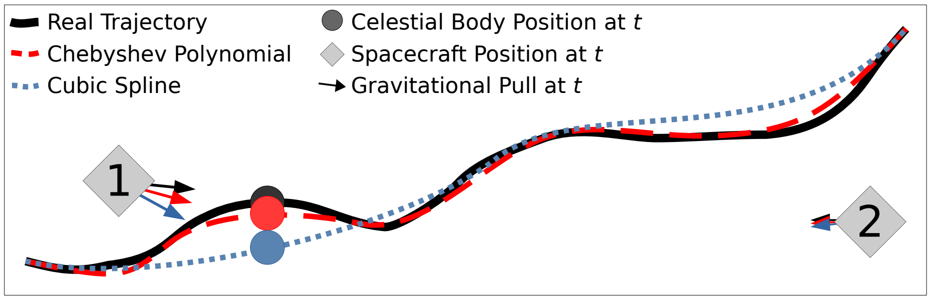
Real Trajectory
Chebyshev Polynomial
Cubic Spline
Celestial Body Position at *t*
Spacecraft Position at *t*
Gravitational Pull at *t*
1
2

Fig. 5: Exaggerated illustration of an arbitrary body's position obtained from trajectory approximations of different accuracy. The gravitational pull on a spacecraft, calculated from this position, is affected by the position deviation introduced by the approximation, especially when close as shown for spacecraft 1.

Celestial Body or Barycentre (BC)	Interval [days]	Polynomial degree N	Per Polynomial values	Per Polynomial bytes	Per 32 Days Record values	Per 32 Days Record bytes
Sun	16	10	33	264	66	528
Mercury	8	13	42	336	168	1 344
Venus	16	9	30	240	60	480
Earth BC	16	12	39	312	78	624
Moon	4	12	39	312	312	2 496
Mars BC	32	10	33	264	33	264
Jupiter BC	32	7	24	192	24	192
Saturn BC	32	6	21	168	21	168
Uranus BC	32	5	18	144	18	144
Neptune BC	32	5	18	144	18	144
Pluto BC	32	5	18	144	18	144

Table 1: Statistics on type 2 polynomials from DE432 with $N+1$ three-dimensional coefficients of double precision floating point type [11] [10].

spacecraft as depicted in figure 5. However, if this effect is small enough (e.g. for spacecraft 2 in the figure), this model may be applied without consequences.

Cubic splines are polynomials of degree three, interpolating between a sequence of knots. When generating an alternative ephemeris model for a specific celestial body, the positions at equally-spaced points in time retrieved from the original DE432 model can be used as knots. By applying a model containing such cubic splines instead of the DE432 Chebyshev polynomials, we are able to improve the efficiency of specific position calculations. Chebyshev polynomials are evaluated by a recursive algorithm, including six to fourteen three-dimensional coefficients (Table 1). On the other hand, a cubic spline is simpler to evaluate and reduces the number of coefficients to four and 96 bytes per polynomial. Additionally, the CUBE format lets us choose the spline interval size for each body independently, as we are not bound to the record's interval anymore. Here, longer intervals increase data reuse both within each specific sample simulation and between different GPU threads.

Furthermore, an ephemeris model using cubic splines turns out to have a very stable deviation compared to the positions retrieved from the original model. The maximum deviation of such a cubic spline model against the original can be calculated statically and later used as accuracy metric to support model selection. Increasing the polynomial interval increases the maximum deviation and decreases the approximation accuracy of the model. However, we improve the reuse of data as fewer different polynomials need to be loaded to cover the same timestamp range. For bodies in the outer solar system, cubic splines seem to be an efficient alternative. While using intervals much longer than set by DE432, we still provide very high accuracy. For bodies closer to the Sun, however, cubic splines are not able to provide more efficient intervals while achieving overall acceptable accuracy levels. This is because of more extreme direction changes in their movement caused by the surrounding close and massive bodies like the Sun and Jupiter. Especially the trajectory of Mercury is heavily perturbed making it very hard to apply cubic splines in an efficient way.

Celestial bodies exclusion; Another method to increase the performance is to exclude a subset of bodies from simulations. An entire celestial body may be excluded from the physics model if its full gravitational effect on the spacecraft state is small enough. This skips the related position calculation and ephemeris data loads within each step and thread, providing a significant performance boost. Although such model modifications are not specific to GPU applications, they are the extreme case of the deviation analysis and, therefore, included in the upcoming experiments.

Planetary systems abstraction; The last optimization method is to abstract planetary systems. In case a planet and its moons are treated as individual bodies by the physics model, they may be abstracted using a fictional body of combined mass at their barycentre, instead. For our applications, this method can be applied for the Earth-Moon system, when the introduced error is small enough to be accepted. We call this method *EMB abstraction* in the rest of this paper.

4 Experimental Results

To assess the impact of the proposed method on the simulation runtime, we execute the disposal analysis for the BepiColombo mission and two additional artificial test cases with different ephemeris models. Here, only the runtime spent on the selected CUDA device is measured. The baseline applies the original DE432 model, and on top of that, the speedup is calculated for runs using alternative ephemeris models. The experiments are executed on Tesla K80, K20XM, and K40M devices as well as on a Tesla V100 in selected cases. The latter is primarily used for detailed investigations of the performance via the Nvidia Visual Profiler because it provides additional insight into the utilization of the device compared to older GPU generations.

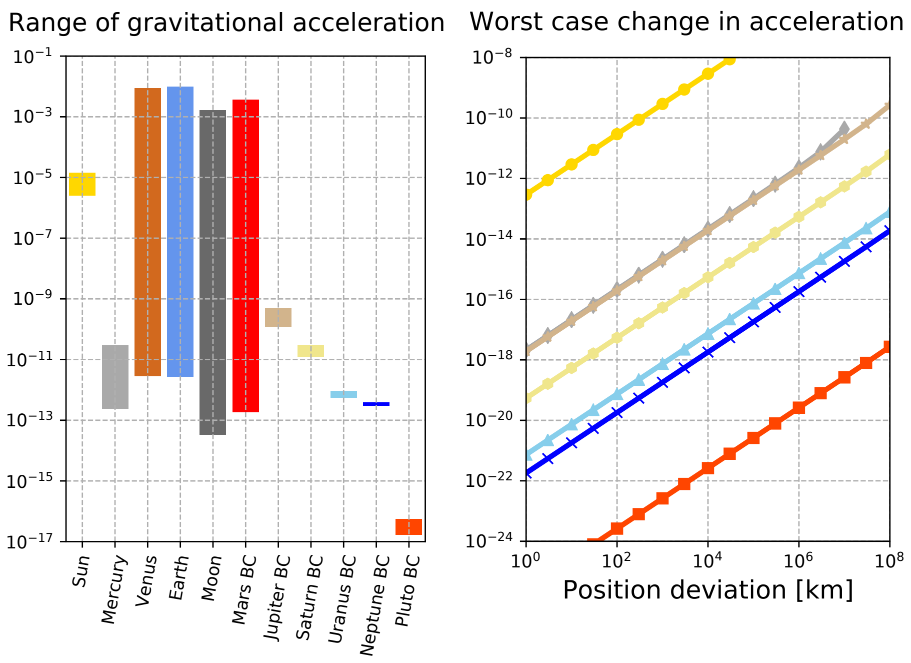

Fig. 6: The analysis of the overall range of gravitational acceleration (left) and worst-case change of acceleration on given position offset (right) per celestial body on a spacecraft between 9.8×10^7 and 2.3×10^8 km from the center of the solar system (Potential distances for the BepiColombo upper stage around the Sun). As Venus, the Earth, the Moon, and Mars orbit within this range, their potential gravitational pull [km/s^2] can be much higher (on close encounter) than for the rest and position deviations would be more critical. For this, these bodies are not included on the right.

4.1 Accuracy Levels of Test Cases

To accelerate position calculations, we pick case-specific ephemeris models without violating accuracy requirements. For this purpose, the accuracy level of the conducted analysis is defined before we select a model for each celestial body. We always opt for simpler models, provided that they guarantee the required accuracy. This way, both the algorithmic complexity as well as the required data size can be reduced.

Regarding the disposal analysis of the BepiColombo upper stage, the model selection will be based on the astrodynamical analysis presented in Figure 6. It requires only a rough guess of the simulated trajectory range and leads to no significant increase in runtime.

The ephemeris model configurations for a range of accuracy levels are presented in Table 2. For each celestial body, either the original polynomials, cubic splines of a specific maximum deviation, or exclusion is selected. The deter-

Accuracy Level	10^{-20}	10^{-15}	10^{-12}	10^{-10}
Sun	DE432	DE432	CS 1	CS 10^2
Mercury	DE432	DE432	DE432	
Venus	DE432	DE432	DE432	DE432
Earth BC	DE432	DE432	DE432	DE432
Moon	DE432	DE432	DE432	DE432
Mars BC	DE432	DE432	DE432	DE432
Jupiter BC	DE432	CS 10^2	CS 10^5	CS 10^7
Saturn BC	DE432	CS 10^4	CS 10^7	
Uranus BC	CS 10	CS 10^6		
Neptune BC	CS 10	CS 10^6		
Pluto BC	CS 10^5			

Table 2: Ephemeris configurations for accuracy levels in km/s^2 regarding the BepiColombo analysis. For each body either the original polynomials (DE432), cubic splines of given maximum deviation in kilometers or exclusion (empty) is selected.

mined collection of polynomials is then stored in the CUBE format supported by cudajectory.

For high accuracy applications at, e.g., 10^{-20} km/s^2, all listed celestial bodies are included as their gravitational effect is of relevant magnitude (see the left plot of Figure 6). Since small position offsets of most bodies already have a worst-case effect larger than 10^{-20} km/s^2, we inherit the original polynomials for these. Only for Uranus, Neptune, and Pluto, cubic splines of lower accuracy are selected. These bodies are always very far away from the spacecraft. Thus, a position deviation of 10 or 10^5 km, respectively, is accepted (see the right plot of Figure 6).

For lower accuracy levels, less accurate cubic splines can be selected for most of the bodies, and some may be excluded entirely when their overall gravitational acceleration is determined to be lower than the level at all times. For instance, Pluto and its moons can safely be excluded when an accuracy level of 10^{-16} or higher is applied, as shown in Figure 6. The model for the lowest level in the table (10^{-10}) includes only six of the eleven bodies. Four of them still require the original accuracy as the spacecraft might have a close encounter, and for the other two, the Sun and Jupiter, cubic splines of quite a high deviation can be selected.

The models for all of the given accuracy levels produce results that can be used to get the first idea of this specific problem, while 10^{-20} and 10^{-15} can also be applied for the final analysis.

Two artificial sample sets on static circular orbits between Earth and Mars are also included to test alternative models not covered by the BepiColombo case. Additionally, we configured their timestamp ranges to stay at zero and 64 days to investigate its effect on the performance. We define an accuracy level at 10^{-11} km/s^2, where cubic spline models at 100 km (Sun), 10^6 km (Venus, Mars, and Jupiter) and 10^8 km (Beyond Jupiter) maximum deviation are applied. Mercury, Uranus, Neptune, and Pluto are excluded from the model if the body

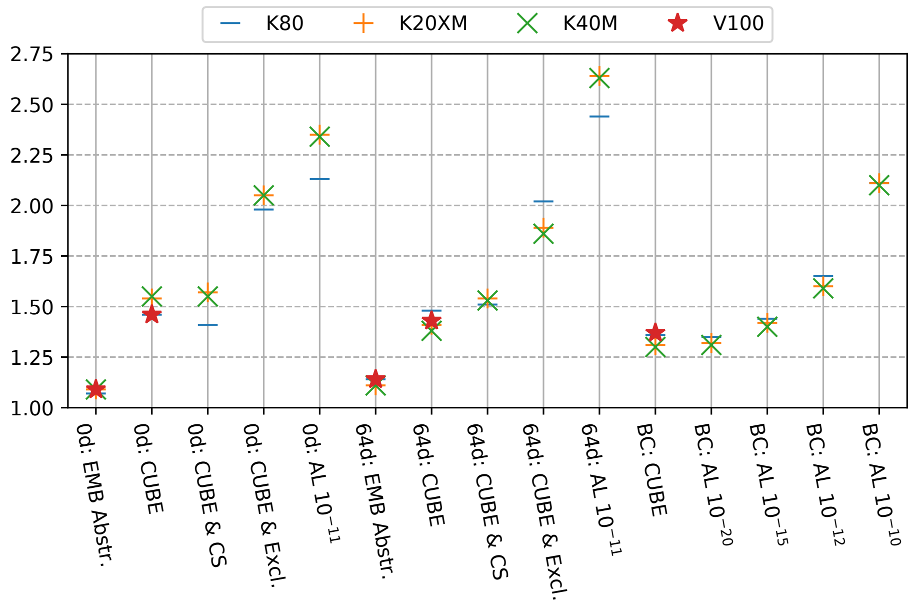

Fig. 7: The speedup of efficient ephemeris models used by the listed cases and configurations. The cases are BepiColombo (BC), zero days (0d), 64 days (64d) timestamp range. The model features include CUBE format, EMB abstraction, cubic splines (CS), and the full exclusion of bodies, which are all included for the accuracy level (AL) configurations.

exclusion feature is enabled. EMB abstraction can safely be applied at every step, as its maximum error is two magnitudes below the selected accuracy level.

When the original model is applied by one of the cases, the total of all position calculations for one point in time requires 2520 bytes of polynomial coefficients. Using the alternative models for the BepiColombo case without EMB abstraction, this data size is reduced by 6% at level 10^{-20} to 48% at level 10^{-10}. However, when EMB abstraction is activated, a reduction of up to 60% can be achieved. The described model for the artificial case can improve this further to 69% reduced memory consumption, while 45% of the position calculations per point in time are skipped entirely.

4.2 Speedup Gained From Ephemeris Model Changes

The BepiColombo analysis applies step size control, which causes an average of 120 days range of timestamps within warps. To identify the performance impact of this timestamp range, we execute the artificial cases without step size control. One case contains samples starting at the same time, and thus the timestamp range will always be zero. Within the second case, however, the samples' start times are equally distributed so that two days are in between every pair of

	DE432	CUBE		Cub. Splines		Exclusion	
Full kernel runtime	10.81 s	7.64 s	-29.3%	7.56 s	-1%	5.99 s	-22%
for integration	0.46 s	0.39 s	-15.2%	0.42 s	+8%	0.42 s	+8%
for physics model	2.22 s	1.74 s	-21.6%	1.85 s	+6%	1.45 s	-17%
for ephemeris model	7.45 s	4.82 s	-35.3%	4.52 s	-6%	3.41 s	-29%
for positions copy	0.52 s	0.43 s	-17.3%	0.46 s	+7%	0.48 s	+12%
Local memory overhead	95.7%	94.7%	-1.0%	95.9%	+1%	94.9%	0
L2 Cache hit rate	38.0%	47.0%	+9.0%	53.0%	+6%	42.0%	-5%
Device memory loads (TiB)	2.508	1.847	-26.4%	1.738	-6%	1.612	-13%
Global memory requests	4.5 G	3.3 G	-26.7%	3.9 G	+18%	2.1 G	-36%
Texture memory requests	90.2 G	90.2 G	0	61.9 G	-31%	56.7 G	-37%
Local memory requests	56.0 G	44.6 G	-20.4%	49.8 G	+12%	31.9 G	-28%

Table 3: Performance profiles of one CUDA kernel call till the first clustering break of the BepiColombo case on a Tesla V100 for DE432 model (baseline) and CUBE format. The third case uses cubic splines on top, while the fourth case excludes Mercury, Saturn, Uranus, Neptune and Pluto on top of the CUBE format.

subsequent samples. This results in a static timestamp range of 64 days within every warp of 32 threads.

The speedup displayed in Figure 7 is observed for the mentioned cases when applying different ephemeris model configurations. All configuration features individually, as well as the fully adapted models, are able to achieve a significant runtime speedup of up to 2.6×, where roughly 62% of execution time is saved. When the data is structured using the CUBE format, a speedup of 1.3× to 1.37× is observed for the BepiColombo case on the tested GPUs. For cases of smaller timestamp ranges, even higher speedup is achieved with over 1.4× and 1.5×, respectively.

The impact of the individual configuration features was tested on artificial cases. Here, the performance gain is similar for both timestamp ranges. However, all features combined are able to speed up the case of 64 days range more than the case of zero-days range. The higher range causes a runtime increase by a factor of 1.13× in the first place, which is reversed by applying the optimizations.

The high accuracy model used for the BepiColombo case already gains a decent performance boost, which is, however, mostly caused by the format change. The included cubic splines do not make a big difference. The less accurate models drive the runtime down to a speedup of 2.1× at the accuracy level of 10^{-10}, which is mainly due to the exclusion of specific bodies.

4.3 Performance Profiles of Ephemeris Model Changes

We profiled the cudajectory using the baseline data format and all the proposed methods. The details are presented in Table 3. The CUBE format reduces the overall runtime by 29.3%, where especially the heavy ephemeris routines are accelerated. Further optimization methods are less affected but still faster by at least 15%. The number of ephemeris data requests is unchanged, but we see a significant reduction of total loads from device memory by over 26%. The only change between the first two runs is the data structure of the texture memory. We can determine that the improved load efficiency results in an increased L2 cache hit rate and fewer reloads of data in both global and local memory. Especially the 12 billion requests saved in local memory have a very beneficial impact on the runtime. Although the local memory overhead was only reduced by 1%, we can state that register spilling is reduced significantly by applying the CUBE format.

When the cubic splines are now applied on top, we observe a further reduction of the ephemeris model runtime. However, the other program sections experience an increase, which results in only a small speedup. This is in line with the observations from Figure 7, where only a small to no speedup is identified for cubic splines on top of the CUBE format. The load requests to texture memory were reduced by 31% as cubic splines are polynomials of lower degree involving fewer coefficients. As the covered intervals are increased as well, we also observe an increase in the L2 cache hit rate. However, the little extra logic in cudajectory to support different types of polynomials introduces additional overhead and thus increases register spilling. This is most likely why we encounter higher rates of local and global memory requests and almost no additional speed.

When specific bodies are excluded entirely from the simulation, a large fraction of the algorithmic effort and data requests are skipped. Thus, the runtime decreases significantly, which makes case-specific model configurations very beneficial. This can be seen in the observed speedup and the presented performance profile for body exclusion. Compared to the CUBE format, 22% of the kernel runtime is saved when 45% of the bodies are excluded. Most of the skipped bodies' orbits are approximated by Chebyshev polynomials of sixth or seventh degree, where, in comparison, fewer coefficients and algorithmic effort are involved. However, with Mercury involving polynomials of degree 14, the most expensive position calculation is also excluded. In total, we perform 37% fewer ephemeris data requests and reduce the device memory traffic by 13%, which subsequently reduces the need for register spilling (28% less local memory requests). As mostly polynomials of shorter intervals are computed, the data reuse and thus the L2 cache hit rate decreases as well.

Although the analyzed configuration features have different impacts on the performance, they were not able to reduce the overall local memory overhead. This states that the pressure on the memory bus due to register spilling is still very high, and a significant amount of time is spent on moving register data back and forth. Additionally, when executing a merged configuration of those cubic splines and exclusions, we get a total runtime per kernel of 5.82 seconds, which is a reduction of 1.73 seconds or 24% against the CUBE format and slightly more than the sum of both individual changes. This draws the conclusion that the changes boost each other and explains the much higher speedups achieved by the lower accuracy level configurations compared to those in the profiles.

5 Related Work

Various astrodynamic simulations utilize the power of GPUs to accelerate their computations. However, we noticed that achieving an efficiency level close to the hardware peak performance necessitates modifications to both the algorithm and data structure. Thus, each problem requires tailor-made optimizations, different from other domains. In the following, we mention similar works in this field.

Russell et al. [16] parallelized the computation of a Mascon model, a high-precision description of the mass distribution of a celestial body for one trajectory simulation. In another work by the same authors [1], they use GPUs to simulate trajectories generated by Lambert's algorithm as an alternative to the Monte-Carlo method. Massari et al. [9], on the other hand, present numerical methods to improve the performance of Monte-Carlo simulations on GPUs, which in theory could also be used in cudajectory. Russell [15] and Korvenja [6] demonstrated that cubic splines and cubic Hermite polynomials produce acceptable ephemeris accuracy while reducing both memory requirements and computation time significantly. Thus, we implemented this concept to reduce the data size of the ephemeris model in cudajectory. To the best of our knowledge, an efficient GPU-specialized ephemeris model for a parallel set of trajectory simulations is not introduced so far.

6 Conclusion and Outlook

Trajectory simulations can benefit significantly from massive parallelization on GPUs. However, with the increase of simulation timestamps within one warp, different ephemeris data need to be loaded into memory, thus causing considerable memory traffic and register spilling. In this paper, we introduced a new data format, called CUBE, which restructures the data to improve data locality. Our experiments showed that by just using the CUBE data format, we could obtain higher speedups of at least 1.3×. Additionally, we noticed that by excluding specific celestial bodies from simulations while losing negligible accuracy, we could reduce the algorithmic complexity and data accesses within the ephemeris model computations and achieve significant speedup. Another approach that yields higher performance with the cost of losing accuracy level is to use cubic splines as an alternative polynomial type. This method decreases the accuracy of the orbit approximation but also simplifies the calculation and reduces the size of the required data. Additionally, adjusting the polynomial intervals affects the data locality within warps of higher timestamp ranges. While the use of cubic splines further improves caching, it also increases the need for register spilling and thus results in only a small runtime improvement on top of the CUBE format. However, in combination with body exclusion, the model changes can boost each other. Thus, cubic splines are a valuable ephemeris model setting. All the proposed optimization methods enabled us to accelerate the trajectory simulations on a real-world scenario between 1.31× - 2.11×, depending on the desired accuracy level.

Acknowledgment. This research has been supported by the European Space Agency, Hessian LOEWE initiative within the Software-Factory 4.0 project, and the German Research Foundation (DFG) through the Program Performance Engineering for Scientific Software. The calculations for this research were conducted on the Lichtenberg high performance computer of the Technical University of Darmstadt and European Space Agency.

References

1. Arora, N., Russell, R.P.: A GPU Accelerated Multiple Revolution Lambert Solver for Fast Mission Design. In: AAS/AIAA Space Flight Mechanics Meeting. vol. 136, pp. 10–198 (2010)
2. ESA and JAXA: BepiColombo: The Europe's first mission to Mercury (2019), `https://sci.esa.int/bepicolombo`
3. Fehlberg, E.: Classical Fifth-, Sixth-, seventh-, and Eight-Order Runge-Kutta Formulas with Stepsize Control. National Aeronautics and Space Administration (1968)
4. Folkner, W.M., Williams, J.G., Boggs, D.H., Park, R.S., Kuchynka, P.: The Planetary and Lunar Ephemerides DE430 and DE431. Interplanetary Network Progress Report **196**, 1–81 (2014)
5. Geda, M.: Massive Parallelization of Trajectory Propagations Using GPUs. Master's thesis, Delft University of Technology (2019)
6. Korvenoja, P., Piché, R.: Efficient Satellite Orbit Approximation. In: In Proc. of 13th International Technical Meeting of the Institute of Navigation Satellite Division. pp. 1930–1937 (2000)
7. Kroese, D.P., Brereton, T., Taimre, T., Botev, Z.I.: Why the Monte Carlo method is so important today. Wiley Interdisciplinary Reviews: Computational Statistics **6**(6), 386–392 (2014)
8. Kutta, W.: Beitrag zur näherungweisen Integration totaler Differentialgleichungen. Z. Math. Phys. **46**, 435–453 (1901)
9. Massari, M., Di Lizia, P., Rasotto, M.: Nonlinear Uncertainty Propagation in Astrodynamics Using Differential Algebra and Graphics Processing Units. Journal of Aerospace Information Systems **14**, 493–503 (2017)
10. N. J. Bachman: SPK Required Reading of NAIF SPICE Toolkit Hypertext Documentation (2017), `http://naif.jpl.nasa.gov/pub/naif/toolkit_docs`
11. NAIF: SPICE Ephemeris Toolkit (2017), `naif.jpl.nasa.gov/naif/toolkit.html`
12. Newhall, X.: Numerical Representation of Planetary Ephemerides. In: Applications of Computer Technology to Dynamical Astronomy. vol. 45, pp. 305–310. Cambridge University Press (1989)
13. Nvidia: Cuda toolkit documentation (2020), `https://docs.nvidia.com/cuda`
14. Nvidia: Visual profiler (2020), `https://developer.nvidia.com/nvidia-visual-profiler`
15. Russell, R., Arora, N.: FIRE: A Fast, Accurate, and Smooth Planetary Body Ephemeris Interpolation System. Celestial Mechanics and Dynamical Astronomy **108**(2), 107–124 (2010)
16. Russell, R.P., Arora, N.: Global Point Mascon Models for Simple, Accurate, and Parallel Gepotential Computation. Journal of Guidance, Control, and Dynamics **35**(5), 1568–1581 (2012)