AAS 11-263

GPU ACCELERATED LAMBERT SOLUTION METHODS FOR THE ORBITAL TARGETING PROBLEM

Sam Wagner* **and Bong Wie**†

Lamberts problem is concerned with the determination of an orbit that connects two position vectors within a specified time of flight. It must often be solved millions of times, especially when one is conducting global searches for possible gravity assist missions, which requires fast efficient solutions. The orbital targeting problem lends itself well to parallel processing, with each departure and arrival combinations being computationally separate. By using the parallel capabilities of modern Graphics Processing Unit (GPU) technology, it is possible to reduce the total run time of the search program by several orders of magnitude. Three methods, have been implemented on a GPU, with run times up to 1100 times faster, when compared to comparable serial FORTRAN verions, at a maximum of almost 20 million solutions per second. Two example missions, one to asteroid 99942 Apophis and a 200-year Earth to Mars search, have been conducted to evaluate the performance of each method.

INTRODUCTION

Lambert's problem is one of most important problems for initial orbital determination and the orbit targeting problem. It has been studied extensively,[1,2,3,4,5,6] in order to determine efficient and robust solutions. In this paper only methods based on a universal variable approach will be considered. In general, these solutions work for all types of orbits, hyperbolic, parabolic, and elleptical and are very robust. Each of the methods tested can easily be employed for multi-revolution solutions as well.

The initial orbit determination often involves searching millions of launch and arrival combinations using Lambert's solutions. Each of these combinations is completely independent, which means the orbit searching program lends itself well to the use of parallel processing. Using the Portland Groups CUDA FORTRAN capabilities, efficient search program can be developed allowing millions of solutions per second. In this paper, three methods, BMW's universal variable approach,[1] Battin's solution,[2] and Gooding's solution to the Lambert equation[5,6,7] will be developed for an NVIDIA Tesla c2060 GPU. Two example missions, one to asteroid 99942 Apophis, which has been studied extensively by the Iowa State Asteroid Deflection Research Center (ADRC),[8,9,10] and the other for a 200 years search from 1900 to 2100 Earth to Mars transfer. This second example mission was chosen in order to ensure a sufficiently large search for the maximum performance from the GPU. Using GPUs to solve Lambert's problem will help minimize cost, in terms of both run time and computer resources.

*Graduate Research Assistant, Asteroid Deflection Research Center, Aerospace Engineering Department, Iowa State University, Ames, IA.

†Vance Coffman Endowed Chair Professor, Asteroid Deflection Research Center, Aerospace Engineering Department, Iowa State University, Ames, IA.

This paper will first start by introducing the key components of each solution method. The next step will be to describe each example mission and compare the solutions for each method and program version. In total, five searching program versions will be compared. The five versions being compared for each solution are: serial Matlab, parallel Matlab, serial FORTRAN, parallel FORTRAN (using OpenMp), and CUDA FORTRAN. These methods will be used for the two example missions, chosen to represent the types of mission the searching program is being developed for. Finally, the efficiency of each method and program type will be evaluated.

The search program has been developed to be as efficient as possible. It consisted of many functions used to determine the orbit of planets as a function of time, solutions to Kepler's equation, transformation from orbital elements to state vectors, etc. Ephemeris data for Apophis has been obtained using NASA's horizons system, with interpolation done by propagating the orbit using Kepler's equation. A great deal of effort has been put forth to ensure the most efficient and accurate searching program has been developed. Therefore, the total run times for each program version include all the necessary function calls requried to complete the search. The run times for the GPU version only represent the GPU runtime and not the time necessary to transfer the arrays to and from the GPU.

SOLUTIONS TO LAMBERT'S PROBLEM

For any given two position vectors and the time-of-flight (TOF), Lambert's problem is concerned with the determination of an orbit connecting the two position vectors within the specified TOF. For this reason, Lambert's problem is well suited for an initial orbit determination and searching technique. In Lambert's problem, the two position vectors and the TOF are known, but the orbit connecting those two points is unknown. The classical Kepler problem is used to determine a position as a function of time where the initial position and velocity vectors are known. Various solutions so Lambert's problem can be found in the literature.[1,3,2,4] While many solution methods have been proposed, three commonly used methods are the classical universal variable method,[1,3,4] Battin's more recent approach using an alternate geometric transformation than Gauss' original method,[2,3] and the method developed by Lancaster and Blanchard[5] with improvement and additional details provided by R.H. Gooding.[6,7] The search for launch dates often requires Lambert's problem to be solved thousands, often millions of time. For this reason, the object of this section is to outline the most efficient method to solve Lambert's problem.

Lambert Problem Definition

In Lambert's problem, the initial and final radius vectors and time-of-flight are given, respectively, as, $\vec{r}_0$, $\vec{r}$, and Δt. The magnitudes of $\vec{r}_0$ and $\vec{r}$ are denoted as r_0 and r. The following parameters are also needed for the post processing for each solution method.

$$c = \mid \vec{r} - \vec{r}_0 \mid \tag{1}$$

$$s = \frac{r_0 + r + c}{2} \tag{2}$$

A method to determine the transfer angle $\Delta\theta$ without quadrant ambiguity is described below.[4]

The transfer angle $\Delta\theta$ for a prograde orbit is determined as follows:

$$\Delta\theta = \begin{cases} \cos^{-1}\left(\dfrac{\vec{r}_0 \cdot \vec{r}}{rr_0}\right) & \text{if } (\vec{r}_0 \times \vec{r})_k \geq 0 \\ 360^o - \cos^{-1}\left(\dfrac{\vec{r}_0 \cdot \vec{r}}{rr_0}\right) & \text{if } (\vec{r}_0 \times \vec{r})_k < 0 \end{cases} \tag{3}$$

Similarly, the transfer angle for a retrograde orbit is determined as:

$$\Delta\theta = \begin{cases} \cos^{-1}\left(\dfrac{\vec{r}_0 \cdot \vec{r}}{rr_0}\right) & \text{if } (\vec{r}_0 \times \vec{r})_k < 0 \\ 360^o - \cos^{-1}\left(\dfrac{\vec{r}_0 \cdot \vec{r}}{rr_0}\right) & \text{if } (\vec{r}_0 \times \vec{r})_k \geq 0 \end{cases} \tag{4}$$

Universal Variable Method

The basics of the universal variable method are provided in this section. This method is the simplest method to be implemented for the searching program. To guarantee convergence and robustness of the solver, a bisection method is used to solve the universal version of Kepler's equation. If a Newton iteration solution is desired, the required derivatives can be found in Ref. [1]. A method to obtain efficient initial guesses is outlined in Ref. [11]. The transfer angle, $\Delta\theta$, needed for many of the calculations throughout this section is first obtained from the previously outlined method.

$$\sqrt{\mu}\Delta t = x^3 S + A\sqrt{y} \tag{5}$$

$$A = \frac{\sqrt{r_0 r}\sin\Delta\theta}{\sqrt{1-\cos\Delta\theta}} \tag{6}$$

$$y = r_0 + r - A(1 - zS) \tag{7}$$

$$x = \sqrt{\frac{y}{C}} \tag{8}$$

$$C = C(z) = \begin{cases} \frac{1-\cosh\sqrt{-z}}{z} & \text{if } z < 0 \\ \frac{1}{2} & \text{if } z = 0 \\ \frac{1-\cos\sqrt{z}}{z} & \text{if } z > 0 \end{cases} \tag{9}$$

$$S = S(z) = \begin{cases} \frac{\sinh\sqrt{-z}-\sqrt{z}}{\sqrt{(-z)^3}} & \text{if } z < 0 \\ \frac{1}{6} & \text{if } z = 0 \\ \frac{\sqrt{z}-\sin\sqrt{z}}{z} & \text{if } z > 0 \end{cases} \tag{10}$$

In this method, A is a function of geometry given by Eq. (6). C and S are the Stumpff functions of only the variable z given by Eqns. 9 and 10. Kepler's time equation can now be represented as a function of the problem geometry and the universal variable z.

The problem can then be solved using the bisection method, which requires only an upper and lower bound to converge. Multiple revolution solutions can also be found by changing the upper and lower bounds. A lower bound of -4π is the value used in for the Lambert solution in this paper, however, the lower bound should be extended for extremely hyperbolic orbits. The upper bound for the 0-revolution solutions considered in the example mission is then $4\pi^2$. Figure 1 illustrates how the variable z changes with time, as well as typical bounds. After a solution for z has been determined, the Lagrange coefficients can be determined and used to find the initial and final velocity vectors.

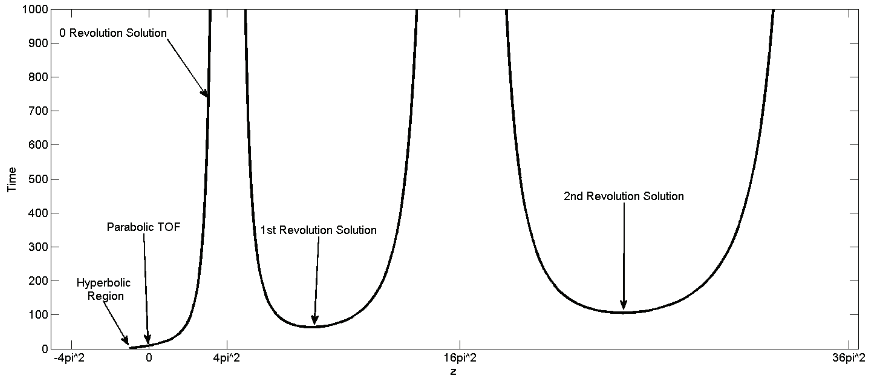

Figure 1. Time of flight versus Z.

Universal Lagrange Coefficients The f and g functions can be converted to a form which is based on universal variables that have already been calculated. Only the velocities are needed, so the $\dot{f}$ is not necessary for this problem.

$$f = 1 - \frac{y}{r_0} \tag{11}$$

$$g = A\sqrt{\frac{y}{\mu}} \tag{12}$$

$$\dot{g} = 1 - \frac{y}{r} \tag{13}$$

The initial and final velocity vectors of the determined orbit are then represented by:

$$\vec{v}_0 = \frac{\vec{r} - f\vec{r}_o}{g} \tag{14}$$

$$\vec{v} = \frac{\dot{r}\vec{r} - \vec{r}_o}{g} \tag{15}$$

Battin's Solution Method

Next we consider Battin's approach first proposed in the 1980's and later published in his book.[2] This solution is similar to the method used by Gauss, but moves the singularity from 180° to 360° and dramatically improves convergence when $\Delta\theta$ is large. It should also be noticed that Battin's solution works for all types of orbits (elliptical, parabolic, and hyperbolic) just as Gauss' original solution does. Throughout this section, a brief outline of Battin's solution will be given. Further details left out can be found in Battin's book.[2]

Lambert's theorem states that the time-of-flight is a function of a, $r_0 + r$, and c. Therefore, the orbit can be transformed to any shape desired as long as the semi-major axis a, $r_0 + r$, and c are held constant. For Battin's formulation[2, 12, 13] the orbit is transformed such that the semi-major axis is perpendicular to the line from $\vec{r}_0$ to $\vec{r}$, which is c by definition. The geometry of the transformed ellipse is shown in Battin's book.[2] From here, the equation for the pericenter radius, r_p, is given below. The details leading up to this function can be found in Ref. [2].

Geometric Transformation of Orbit

$$r_p = a(1 - e_0) = r_{0p} \sec^2 \frac{1}{4} (E - E_0) \tag{16}$$

where e_0 is the eccentricity of the transformed orbit, while r_{0p} is the mean point of the parabolic orbit.[2] The mean point of the parabolic radius r_{0p} is also given by[2]

$$r_{0p} = \sqrt{r_0 r} \left[\cos^2 \left(\frac{\Delta\theta}{4} \right) + \tan^2 (2w) \right] \tag{17}$$

The value of $\tan^2 (2w)$ is needed for the calculation, which will be defined later. Next, Kepler's time of flight equation can be transformed into a cubic equation, which must then be solved. First, y will be defined as

$$y^2 \equiv \frac{m}{(\ell + x)(1 + x)} \tag{18}$$

Kepler's time-of-flight equation can now be represented by the following cubic equation.

$$y^3 - y^2 - \frac{m}{2x} \left(\frac{\tan^{-1} \sqrt{x}}{\sqrt{x}} - \frac{1}{1 + x} \right) = 0 \tag{19}$$

In the above equation, l, m, and x are always positive and are defined below. To calculate l, the following two equations, dependent only on the geometry of the problem, are also necessary.

$$\epsilon = \frac{r - r_0}{r_0} \tag{20}$$

$$\tan^2 (2w) = \frac{\frac{\epsilon^2}{4}}{\sqrt{\frac{r}{r_0}} + \frac{r}{r_0} \left(2 + \sqrt{\frac{r}{r_0}} \right)} \tag{21}$$

Using Eqs. (20) and (21), ℓ can be calculated as follows.

$$\ell = \frac{\sin^2\left(\frac{\Delta\theta}{4}\right) + \tan^2(2w)}{\sin^2\left(\frac{\Delta\theta}{4}\right) + \tan^2(2w) + \cos\left(\frac{\Delta\theta}{2}\right)} \qquad 0^\circ < \Delta\theta \leq 180^\circ \tag{22}$$

$$\ell = \frac{\cos^2\left(\frac{\Delta\theta}{4}\right) + \tan^2(2w) - \cos\left(\frac{\Delta\theta}{2}\right)}{\cos^2\left(\frac{\Delta\theta}{4}\right) + \tan^2(2w)} \qquad 180^\circ < \Delta\theta \leq 360^\circ \tag{23}$$

$$m = \frac{\mu \Delta t^2}{8 r_{op}^3} \tag{24}$$

$$x = \sqrt{\left(\frac{1-\ell}{2}\right)^2 + \frac{m}{y^2}} - \frac{1+\ell}{2} \tag{25}$$

Initial conditions for x that guarantee convergence are:

$$x_0 = \begin{cases} 0 & \text{parabola, hyperbola} \\ \ell & \text{ellipse} \end{cases} \tag{26}$$

The cubic function (Eq. (19)) can now be solved using the following sequential substitution method:

1. An initial estimation of x is given by Eq. (26).
2. Calculate all the values needed for the cubic from Eqs. (20), (21), (22) or (23), and (24).
3. Solve the cubic Eq. (19) for y.
4. Use Eq. (25) to determine a new value for x.
5. Repeat the above 3 steps until x stops changing.

We now have a nearly complete solution algorithm for Lambert's problem, although solving the cubic function is not a trivial matter. Battin[2] next develops a method to flatten the cubic function to improve the convergence rate, as well as a method using continued fractions to determine the largest real positive root of the cubic function. The equations necessary to find the flattening parameters and solve the cubic function are provided below.

The cubic equation, Eq. (19), from the previous section can now be represented in terms of the flattening parameter functions, h_1 and h_2, as follows:

Use of Free Parameter to Flatten Cubic Function

$$y^3 - (1 + h_1)y^2 - h_2 = 0 \tag{27}$$

The flattening parameters can be represented in terms of x, l, and m as

$$h_1 = \frac{(l + x)^2(1 + 3x + \xi)}{(1 + 2x + l)[4x + \xi(3 + x)]} \tag{28}$$

$$h_2 = \frac{m(x - l + \xi)}{(1 + 2x + l)[4x + \xi(3 + x)]} \tag{29}$$

The function $\xi(x)$ needed for the calculation of h_1 and h_2 is calculated from the continued fraction as follows:

$$\xi(x) = \cfrac{8(\sqrt{1+x}+1)}{3 + \cfrac{1}{5 + \eta + \cfrac{\frac{9}{7}\eta}{1 + \cfrac{\frac{16}{63}\eta}{1 + \cfrac{\frac{25}{99}\eta}{1 + \cfrac{\frac{36}{143}\eta}{1 + \ddots}}}}}} \tag{30}$$

where η is defined as

$$\eta = \frac{x}{(\sqrt{1+x}+1)^2} \qquad \text{where} \qquad -1 < \eta < 1 \tag{31}$$

Solving the Cubic Function In this section, a method to determine the largest real root of Eq. (27) is outlined. Using this method a successive algorithm can be used ultimately to determine the f and g functions, given by Ref. [1,3]. Using the Lagrangian f and g functions, the initial and final velocity vectors are then obtained. The first step is to calculate B and u as follows:

$$B = \frac{27h_2}{4(1 + h_1)^3} \tag{32}$$

$$u = \frac{B}{2(\sqrt{1+B}+1)} \tag{33}$$

Also, $K(u)$ is the calculated from the continued fraction

$$K(u) = \cfrac{\frac{1}{3}}{1+\cfrac{\frac{4}{27}u}{1+\cfrac{\frac{8}{27}u}{1+\cfrac{\frac{2}{9}u}{1+\cfrac{\frac{22}{81}u}{1+\ddots}}}}} \tag{34}$$

where the coefficients for the odd and even coefficients of u are generally obtained from the following two equations.

$$\gamma_{2n+1} = \frac{2(3n+2)(6n+1)}{9(4n+1)(4n+3)} \tag{35}$$

$$\gamma_{2n} = \frac{2(3n+1)(6n-1)}{9(4n-1)(4n+1)} \tag{36}$$

The largest positive real root for the cubic equation is calculated as

$$y = \frac{1+h_1}{3}\left(2+\frac{\sqrt{1+B}}{1+2u(K^2(u))}\right) \tag{37}$$

The cubic function (Eq. (27)) can now be solved using the following sequential substitution method.

1. An initial estimation of x is given by Eq. (26).
2. Calculate all the values needed for the flattening parameters Eqs.(28) and (29) from Eqs. (20), (21), (22) or (23), and (24).
3. Calculate $\eta(x)$ from Eqs. (30) and (31).
4. Calculate $K(u)$ using Eqs. (32), (33), and (34)
5. Calculate the solution for the cubic function using Eq. (37) for y.
6. Use Eq. (25) to determine a new value for x.
7. Repeat the above 5 steps until x stops changing.

The next step is to determine the semi-major axis of the orbit. If the semi-major axis is positive, the orbit is elliptical, and the initial and final velocity vectors can be calculated as follows. The hyperbolic and parabolic velocity vectors are calculated in a similar manner.[3]

$$a = \frac{\mu(\Delta t)^2}{16r_{op}^2xy^2} \tag{38}$$

With the semi-major axis is now known, the Lagrange coefficients can be determined in order to find the initial and final velocities of the orbit.

The Lagrange coefficients for the elliptical orbit case can be calculated from the following set of equations.

Determining the Lagrange Coefficients for Elliptical Orbits

$$\beta_e = 2\sin^{-1}\sqrt{\frac{s-c}{2a}} \tag{39}$$

$$\beta_e = -\beta_e \quad \text{If } \Delta\theta > \pi \tag{40}$$

$$a_{min} = \frac{s}{2} \tag{41}$$

$$t_{min} = \sqrt{\frac{a_{min}^3}{\mu}}(\pi - \beta_e + \sin\beta_e) \tag{42}$$

$$\alpha_e = 2\sin^{-1}\sqrt{\frac{s}{2a}} \tag{43}$$

$$\alpha_e = 2\pi - \alpha_e \quad \text{If } \Delta t > t_{min} \tag{44}$$

$$\Delta E = \alpha_e - \beta_e \tag{45}$$

The Lagrangian coefficients can be calculated as follows:

$$f = 1 - \frac{a}{r_0}(1 - \cos\Delta E) \tag{46}$$

$$g = \Delta t - \sqrt{\frac{a^3}{\mu}}(\Delta E - \sin\Delta E) \tag{47}$$

$$\dot{g} = 1 - \frac{a}{r}(1 - \cos\Delta E) \tag{48}$$

With the Lagrangian coefficients now known, the initial and final velocity vectors can be found from Eqs. (14)-(15) respectively. Hyperbolic and parabolic Lagrange equations are necessary for the hyperbolic and parabolic solutions. The details are similar the those outlined above for the eccentric orbits, and can be found in Ref. [3].

Lancaster-Gooding Method

The last method examined in this paper for the use with GPU is commonly referred to as Gooding's solution. Gooding built on work done by Lancaster et al,[5] using a third order Halley's convergence method as well a as complex method to determine initial guesses. These initial guesses are formulated to ensure efficient and robust convergence. The complete details of this solution are too lengthy and complex to include in this paper, but can be found in Refs. [5, 6]. The basic derivation of the equation which must be solved is included in this section.

Lambert's equation, a form of Kepler's time-of-flight equation based only on the chord, semiparameter, and semi-major axis is given by

$$\sqrt{\mu}\Delta t = a^{\frac{3}{2}}[\alpha - \beta - (\sin\alpha - \sin\beta)] \tag{49}$$

where half angle formulas for α and β are given by

$$\sin\frac{\alpha_0}{2} = \sqrt{\frac{s}{2a}} \tag{50}$$

$$\sin\frac{\beta_0}{2} = \sqrt{\frac{s-c}{2a}} \tag{51}$$

The quadrant ambiguities associated with α and β must first be accounted for. The preceding method will allow for the correct α and β to be determined for all four possible solutions. The minimum time of flight is first determined as follows.

$$t_m = \sqrt{\frac{s^3}{8\mu}}(\alpha_m - \beta_m + \sin\beta_m) \tag{52}$$

The subscript m indicates the minimum semi-major axis solution, often known as the minimum energy solution and $\alpha_m = \pi$ and $\beta_m = \beta_0$. A summary of the corrections to α and β four possible arcs is included below, with $0 \leq \theta \leq 2\pi$ and $0 \leq \beta_0 \leq \alpha_0 \leq \pi$.

$$0 \leq \Delta\theta < \pi \qquad \beta = \beta_0 \tag{53}$$

$$\pi \leq \Delta\theta < 2\pi \qquad \beta = -\beta_0 \tag{54}$$

$$\Delta t \leq t_m \qquad \alpha = \alpha_0 \tag{55}$$

$$\Delta t \; t_m \quad \alpha = 2\pi - \alpha_0 \tag{56}$$

Gooding's method is based of solutions to Lambert's equation, as described above. This section is only meant to help illustrate where this solution comes from. Lancaster et al.[5] formulated a universal approach based on Lambert's equation. Gooding then continued this work to determine initial guesses to ensure efficient convergence. Further detail can be found in Refs. [5, 6, 7, 14].

The next step is to evaluate the performance of each algorithm described in this section. Each method and program version will be tested using two example missions contained in the next section. Lastly, the performance aspect of the GPU algorithms will be analyzed.

GPU SOLUTIONS PERFORMANCES

The performance of the entire search program, including the three Lambert solutions, needs to be examined. For this reason, two example missions will be used to evaluate the performance of each version of the search program. However, a majority of the calculations are performed during each Lambert solution, which means the examples are a good performance indicator. The program was originally developed in Matlab and later ported over to FORTRAN and CUDA FORTRAN. Neither the serial or parallel Matlab programs were used for the 200-year Mars mission search due to the low performance of the Matlab programs, when compared to the FORTRAN and CUDA FORTRAN versions. All versions of the program, with the exception of the CUDA FORTRAN version, were run on a standard work station. The details of the workstation can be found in Table 1. The CUDA FORTRAN version was run on a GPU compute cluster containing 4 NVIDIA Tesla c2060 graphics cards. At this time the CUDA FORTRAN searching program only utilizes 1 graphics card. Details of the GPU compute cluster can be found in Table 2. In addition, all FORTRAN versions of the search program, with the exception of the CUDA version, were run using the GNU FORTRAN compiler.

Table 1. Information on the workstation used for the computational efficiency study.

Model	Dell T3500 Workstation
Operating System	Windows Vista Enterprise 64 bit
Processor	Intel(R) Xeon(R) W3520 2.67 GHz -4 core
Memory	6.00 GB 1066 MHz DDR3
Matlab	R2010b-64 bit

Table 2. Information on the GPU workstation used for the computational efficiency study.

Model	Amax ServMax Tesla GPU HPC
Operating System	RHEL v5.4
Processor	2x Intel Xon LGA1366 2.66 GHz -6 core
Memory	32 GB 1333 MHz DDR3
GPU	4x Tesla C2050

Sample Apophis Mission This sample mission is a fictional mission in which Apophis passes through the April 13th, 2029 keyhole, resulting in an impact on April 13th, 2036. This mission is representative of the types of searches that will be performed by the ADRC, and has been studied extensively.[9, 10] Approximately 7 years of launch dates are searched, totalling 2578 days, with a maximum mission length of 1000 days allowed. One-day time steps were used in this search resulting in a total of 2.578 million Lambert solution calls. A porkchop plot of the departure ΔV required is shown in Fig. 2. From this figure, launch dates for the unconstrained intercept mission can be found. Further analysis of the mission will not be presented in this study, but can be found in Ref. [10].

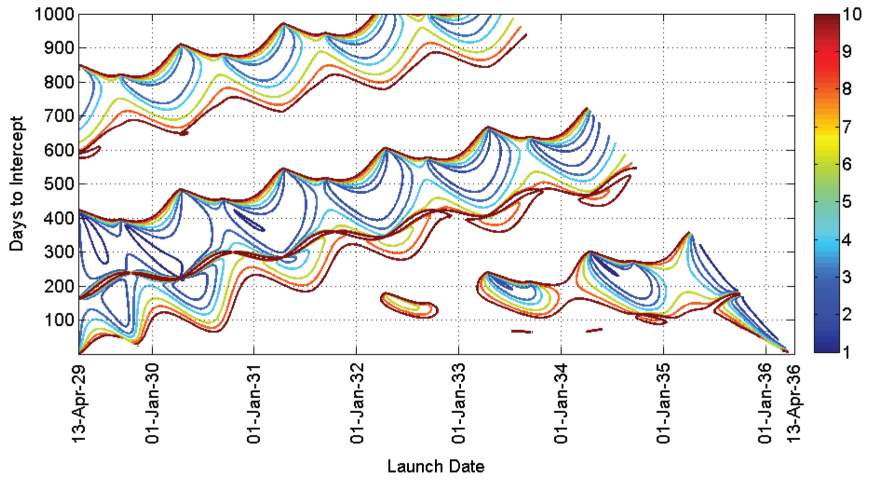

Figure 2. Porkchop plot of departure ΔV for Apophis mission.

The performance of each search program version for the universal variable method, Battin's method, and Gooding method can be found in Tables 3, 4, and 5 respectively.

The universal variable method, the simplest method to implement, has the worst performance for each mission run. The solution employed in this study used a bisection root finding method. It may be possible to increase the performance of this method by using a Newton iteration scheme, using the initial guess method described in Ref. [11]. For this example, mission approximately 930 solutions per second are obtained using the serial Matlab version. The serial FORTRAN version shows significant improvement, at approximetaly 31,000 solutions per second, or 33 times improvement over serial matlab. In comparison, the final CUDA FORTRAN program was able to obtain over 1.67 million solutions per second. The CUDA FORTRAN version of this solution is approximately 54 times faster than the serial FORTRAN version and 1800 times faster than serial Matlab. These results are expected and consistent with improvements found in other studies, often 1 to 2 orders of magnitude improvement in program run times.

Table 3. Comparisons of program version performance for example Apophis mission for Universal Variable Lambert solution.

Program Version	Run Time	Solutions per Second
Serial Matlab	2782.33	929.80
Parallel Matlab	699.93	3,696.08
Serial Fortran	83.68	30,915.39
Parallel Fortran	14.11	183,345.15
GPU Fortran	1.55	1,674,433.66

Battin's method and Gooding's method are often found to achieve comparable results. Both of these methods are two of the most efficient methods developed to date. Table 4 shows the serial Matlab achieved 3,600 solutions per second, or approximately 4 times faster than the comparable universal variable results previously presented. The serial FORTRAN Battin version has approximately the same speed up over the serial Matlab at just over 33 times faster, with 120,000+ solutions per second. For this example the CUDA FORTRAN version got nearly 11 million solutions, or a speed increase nearly 6.5 times faster than the corresponding universal variable method. At 11 million solutions a second the CUDA FORTRAN version is over 91 times faster than the comparable serial FORTRAN version.

Table 4. Comparisons of program version performance for example Apophis mission for Battin's Lambert solution.

Program Version	Run Time (sec)	Solutions per Second
Serial Matlab	713.88	3,623.86
Parallel Matlab	186.03	13,906.36
Serial Fortran	21.42	120,774.98
Parallel Fortran	3.29	786,322.19
GPU Fortran	0.235	10,999,149.66

The last solution method tested is Lancaster's method with Gooding's improvement. While the serial Matlab version of the program runs roughly 33% slower than Battin's method, the GPU version is approximately 24% faster. This method has the least speed-up, of just of 6 times, when comparing the serial FORTRAN and Matlab versions. For this example mission, the Gooding method implemented in CUDA FORTRAN was able to achieve over 13.5 million solutions per second, running nearly 900 times faster than the corresponding serial FORTRAN version.

Table 5. Comparisons of program version performance for example Apophis mission for Lancaster/Gooding Lambert solution method.

Program Version	Run Time	Solutions per Second
Serial Matlab	1064.99	2,420.68
Parallel Matlab	252.59	10,206.26
Serial Fortran	172.61	14,935.40
Parallel Fortran	20.15	127,940.45
GPU Fortran	0.190	13,589,878.76

Overall the speed-ups for each method range from 1 to 3 orders of magnitude. However, as the next example shows further speed increases are possible when larger Lambert searches are performed.

Sample 200-Year Earth to Mars Search The next example mission better illustrate the speed ups possible when utilizing a modern GPU. Each program currently only utilizes one of the four GPU's on the workstation, so speed ups of approximately 4 times over the current program should also be possible.

The Lambert search run in this mission is a 200-year Earth to Mars transfer. As before a 1 day time step out to a maximum of a 1000 day mission is used. This results in nearly 73 million Lambert calls, saturating the GPU much more than the example Apophis mission. The results, presented below, are a significant performance improvement over the Apophis mission results. A porckchop for this mission of the Earth departure V_{inf} from 2000 to 2050 is shown in Fig. 3. For this example mission neither the serial or parallel Matlab programs were run, due the relatively low performance the programs. Results for the Universal Variable, Battin, and Gooding's methods can be found in Tables 6, 7, 8 respectively.

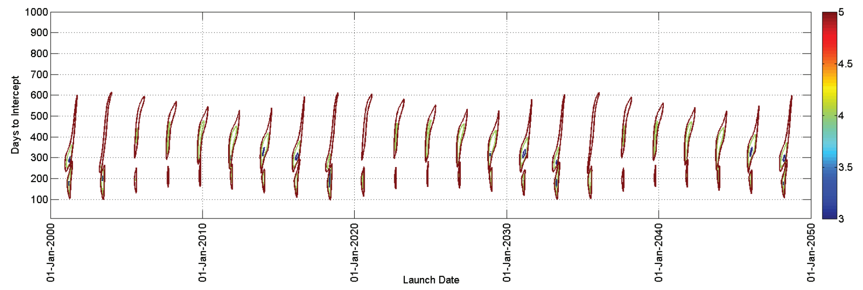

Figure 3. 50-year porkchop plot of departure $V_{\rm inf}$ for Mars mission.

As before, the first method tested was the universal variable method. For this test the serial and parallel FORTRAN versions have similar performance to the previous example. There is, however, a significant performance increase in the GPU version. The CUDA FORTRAN version obtained slightly over 12.6 millions solutions a second, 7.5 times faster that in the Apophis example and over 412 times faster than serial FORTRAN.

Table 6. Comparisons of program version performance for example Mars mission for Universal Variable Lambert solution.

Program Version	Run Time	Solutions per Second
Serial Fortran	2,385.68	30,619.83
Parallel Fortran	364.02	200,673.04
GPU Fortran	5.78	12,632,771.29

In this example mission, Battin's method was slightly faster for the CUDA Fortran version. Using Battin's method, the program was able to achieve 19.7 million solutions per second, while the version using Gooding's method was just over 19 million solutions a second. Both versions were able to perform all 73 million solutions in under 4 seconds. Using Battin's method, a speed up of 180 times over serial FORTRAN was achieved. Amazingly a speed increase over 1100 times was achieved when comparing the serial and CUDA Gooding method programs. In the next section, the best performance from each example will be examined and compared.

Table 7. Comparisons of program version performance for example Mars mission for Battin's Lambert solution.

Program Version	Run Time	Solutions per Second
Serial Fortran	667.23	109480.99
Parallel Fortran	51.58	786322.19
GPU Fortran	3.70	19743079.69

Table 8. Comparisons of program version performance for example Mars mission for Lancaster/Gooding Lambert solution method.

Program Version	Run Time	Solutions per Second
Serial Fortran	4,292.94	17,016.08
Parallel Fortran	638.78	114,357.06
GPU Fortran	3.840	19,023,177.08

GPU Performance Aspects The best performance for each program version from the two example missions is examined in this section. The performance of the final search program is desired, rather than just the performance of each Lambert solver, so no Monte Carlo simulations for each solution have been performed. These solutions have been well covered in literature and are known to converge for nearly all orbit types, making such simulations unnecessary.

The maximum performance, given in speed increases over the original serial Matlab version is shown for the three solution methods are presented in Tables 9, 10, and 11.

Over all, the universal variable method shows the most improvement over the original serial Matlab program versions, showing 4 orders of magnitude better performance, approximately 13,600 times faster. This represents a total of over 12.6 million solutions per second. Using the best performance between the two examples, when run on the GPU, the universal variable method also shows nearly 400 times speed up over the serial FORTRAN version and 63 times speed up over the parallel FORTRAN version. All performance comparisons of the Universal Variable method can be found in Table 9.

Table 9. Performance comparisons versus serial matlab for best case scenarios for the universal variable method from example missions.

Program Version	Efficiency
Serial Matlab	1.00
Parallel Matlab	3.98
Serial Fortran	33.25
Parallel Fortran	197.19
GPU Fortran	13,586.60

The performance characteristics of the different program versions using Battin's solution are shown in Table 10. The CUDA FORTRAN verson using Battin's Lambert solution shows a nearly 5500 times speed up over serial FORTRAN, the least improved of the three methods, but still a 3 orders of magnitude decrease in total run time, with a best performance of 19.7 million solutions per second. Although this method shows the least improvement, it is overall the fastest of the three solutions. This method also shows a 163 time speed-up over the corresponding serial FORTRAN version and a 25 times speed up over parallel FORTRAN.

The last solution method to examine is Gooding's method, shown in Table 11. The Gooding solution method shows the second best improvement when compared with serial Matlab, with run times almost 7,900 times faster. Gooding's solutions also achieved an 1,100 times speed up serial FORTRAN, the most improved in this aspect. The CUDA FORTRAN program version using this solution method was able to achieve a maximum of 19 million solutions per second, approximately

Table 10. Performance comparisons versus serial matlab for best case scenarios for Battin's method from example missions.

Program Version	Efficiency
Serial Matlab	1.00
Parallel Matlab	3.84
Serial Fortran	33.33
Parallel Fortran	390.81
GPU Fortran	5,448.08

3.6% slower than Battin's method.

Table 11. Performance comparisons versus serial matlab for best case scenarios for the Gooding method from example missions.

Program Version	Efficiency
Serial Matlab	1.00
Parallel Matlab	4.22
Serial Fortran	6.17
Parallel Fortran	52.85
GPU Fortran	7,858.61

SUMMARY

A graph of the final best performances for the GPU version of each solution method can be seen in Fig. 4. Each solution method has the same order of magnitude when comparing the number of solutions per second, in the order of tens of millions of solutions per second. This represents speed-ups of 2 to 3 orders of magnitude over the corresponding serial FORTRAN versions. These speed ups will allow for global searching of the minimum Delta V launch windows of more complicated interplanetary missions with multiple gravity assist maneuvers. Utilizing GPUs, it may be possible to find complicated missions, using only brute force methods, faster than methods requiring optimization. This will also help ensure that the global optimal solution is not missed.

CONCLUSION

When comparing Lambert solutions, each method has the same order of magnitude for final solutions per second. All three methods are based on a universal variable approach, meaning they work for all orbit types, hyperbolic, parabolic, and elliptic. Battin's version appears to give the best overall performance for each program version. While mathematically complicated, the implementation of Battin's method is relatively simple. The Universal Variable method is the most intuitive method of the three and is also extremely easy to implement and perhaps the most covered in literature. In comparison, Gooding's method relies on complicated methods to determine initial guess that guarantee convergence and a third-order Halley convergence method. While extremely efficient, Gooding's method is the most difficult method to implement.

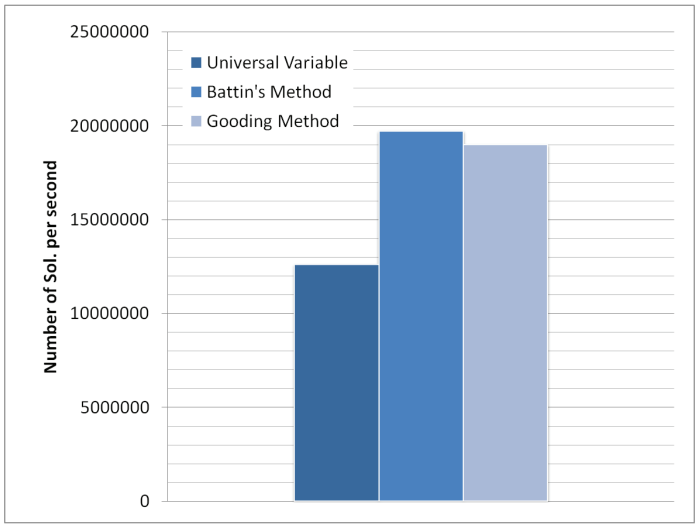

Figure 4. GPU performance comparison for all three methods.

ACKNOWLEDGMENT

This work was supported by the Iowa Space Grant Consortium (ISGC) through a research grant to the Asteroid Deflection Research Center at Iowa State University. The authors would like to thank Dr. Ramanathan Sugumaran (Director, ISGC) for his interest and support of this research work. The authors would also like to thank Brian Kaplinger, graduate research assistant for the Asteroid Deflection Research Center, for his assistance in developing the GPU kernel and driving programs.

REFERENCES

[1] R. Bate, D. Mueller, and J. White, *Fundamentals of Astrodynamics*. 180 Varick Street, New York, NY: Dover Publications, Inc., 1st ed., 1971.

[2] R. Battin, *An Introduction to the Mathematics and Methods of Astrodynamics, Revised Edition*. 1801 Alexadner Bell Drive, Reston, VA: AIAA Educational Series, revised edition ed., 1999.

[3] D. Vallado, *Fundamentals of Astrodynamics and Applications*. 401 Coral Circle, El Segundo, CA: Microcosm Press, 2nd ed., 2004.

[4] H. Curtis, *Orbital Mechanics for Engineering Students*. Linacre House, Jordan Hill, Oxford: Elsevier Butterworth-Heinemann, 1st ed., 2005.

[5] E. Lancaster and R. Blanchard, "A Unified Form of Lambert's Theorem," *NASA Technical Note*, Sept. 1969.

[6] R. Gooding, "On the Solution of Lambert's Orbital Boundary-Value Problem," *Royal Aerospace Establishment*, Apr. 1988.

[7] R. Gooding, "A Procedure for the Solution of Lambert's Orbital Boundary-Value Problem," *Kluwer Academic Publishers*, Jan. 1990.

[8] S. Wagner, A. Pitz, D. Zimmerman, and B. Wie, "Interplanetary Ballistic Missile (IPBM) System Architecture Design for Near-Earth Object Threat Mitigation," *60th International Astronautical Congress*, Vol. IAC-09, Oct. 2009.

[9] S. Wagner and B. Wie, "Minimum ΔV Launch Windows for a Fictive Post-2029 Apophis Deflection/Disruption Mission," *AAS*, Vol. 10, Feb. 2010.

[10] S. Wagner and B. Wie, "Design of Fictive Post-2029 Apophis Intercept Mission for Nuclear Disruption," *to be presented at 2010 AIAA Astrodynamics Specialist Conference, Toronto, Canada*, Aug. 2010.

[11] N. Arora and R. Russell, "GPU Accelerated Multiple Revolution Lambert Solver for Fast Mission Design," *AAS*, Vol. 10.

[12] L. Loechler, “An Elegant Lambert Algorithm for Multiple Revolution Orbits,” Master’s thesis, Massachusetts Institute of Technology, May 1988.

[13] H. Shen and P. Tsiotras, “Using Battin’s Method to Obtain Multiple-Revolution Lambert’s Solutions,” *American Astronautical Society*, Vol. 116, 2003.

[14] J. Prussing and B. Conway, *Orbital Mechanics*. 200 Madison Avenue, New York, New York 10016: Oxford University Press, first ed., 1993.