Lambert's problem algorithms

A critical review

Author: Jorge Martínez[1]
Supervisor: Manuel Sanjurjo Rivo[2]

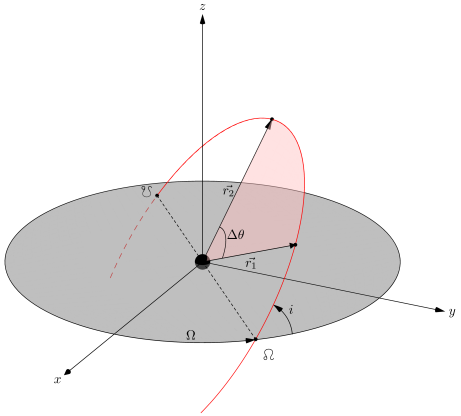
[1]Undegraduate Aerospace Engineer
[2]PhD Aerospace Engineer

Aerospace Engineering Bachelor Thesis
Universidad Carlos III de Madrid

October 4, 2021

What is the Lambert's problem?

Lambert's problem is the Boundary Value Problem (BVP) in the context of the restricted two-body problem dynamics.

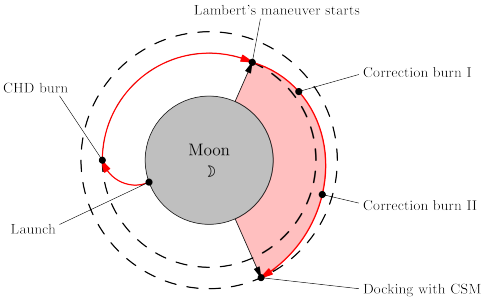


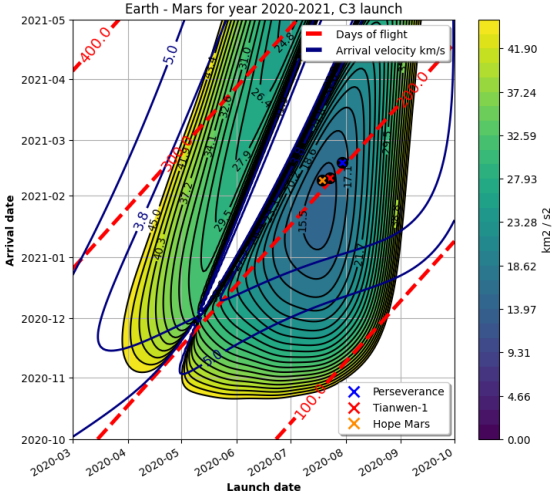
Two-body problem: BVP

$$\ddot{\vec{r}} = -\frac{\mu}{r^3}\vec{r} \qquad \begin{cases} \vec{r}\,(t_1) = \vec{r}_1 \\ \vec{r}\,(t_2) = \vec{r}_2 \end{cases}$$

Solve for the orbit which passes through $\vec{r}_1$ and $\vec{r}_2$ over a finite amount of time $\Delta t = t_2 - t_1$.

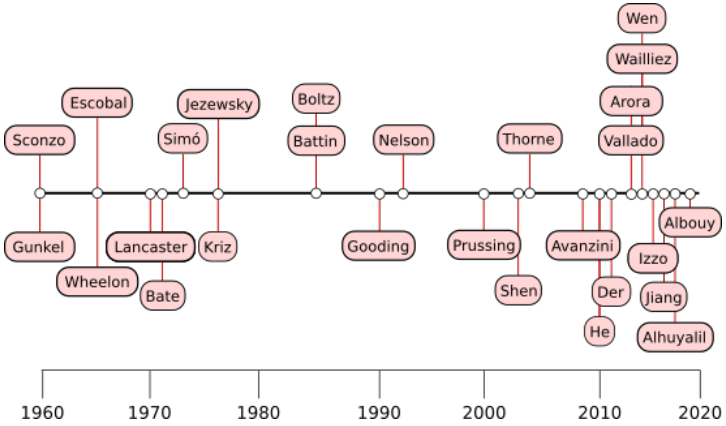
Why do we care about Lambert's problem?





Motivation of this work

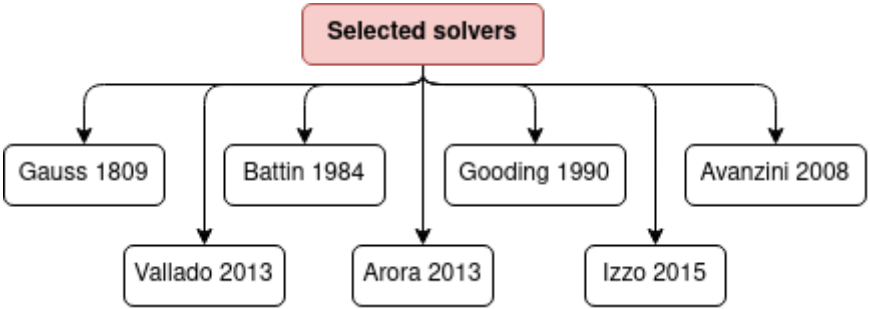
There are lots of algorithms available for solving the problem.



Which one performs the best?

Selected solvers

Selection based on previous works made by Klump 1999, Sangrà 2015 and Martínez 2021.



How can we compare different solvers?

We analyzed and compared the selected solvers by:

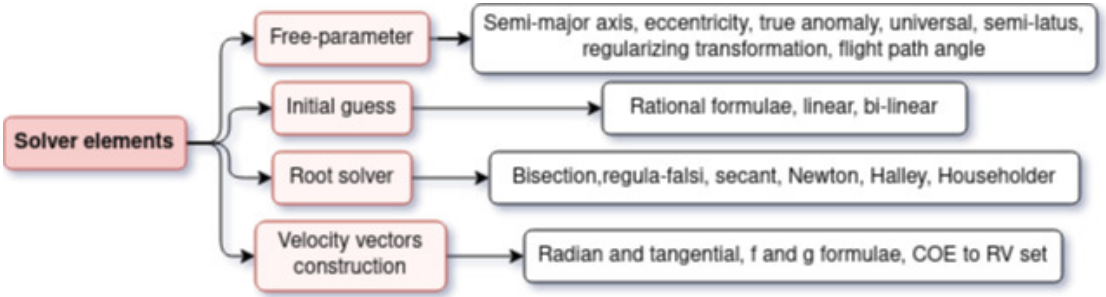
- Identifying a common set of elements in every Lambert's problem algorithm.
- Considering the required number of iterations (NOI).
- Considering the required time per iteration (TPI).
- Considering the total computation time employed (TCT).

A pure Python library was created and all solvers ara available to the public together with the performance comparison:

- Source code: https://github.com/jorgepiloto/lamberthub
- Documentation: https://lamberthub.readthedocs.io/en/latest/

Elements of a Lambert's problem algorithm

Any Lambert's problem algorithm is composed of four basic elements. A classification has been proposed based on the free-parameter.



Selected solvers elements

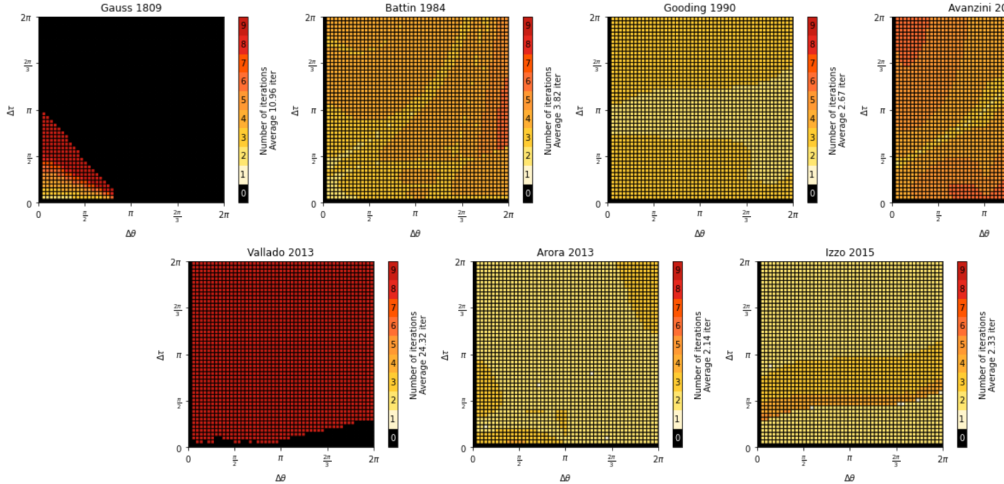
Previously presented elements were evaluated for each one of the selected solvers:

Method	**Element**			
	Free-parameter	**Initial guess**	**Numerical method**	$\vec{v_1}$ **and** $\vec{v_2}$
Gauss 1809	Semi-latus rectum	Rational formulae	Series function approximation	f and g
Battin 1984	Semi-latus rectum	Rational formulae	Series function approximation	f and g
Gooding 1990	Universal	Bi-linear	Halley's method	v_{r_i} and v_{t_i}
Avanzini 2008	Eccentricity	Interval	Regula-falsi	COE to RV
Arora 2013	Universal	Rational formulate	Halley's method	f and g
Vallado 2013	Universal	Interval	Bisection	f and g
Izzo 2015	Universal	Linear	Householder's method	v_{r_i} and v_{t_i}

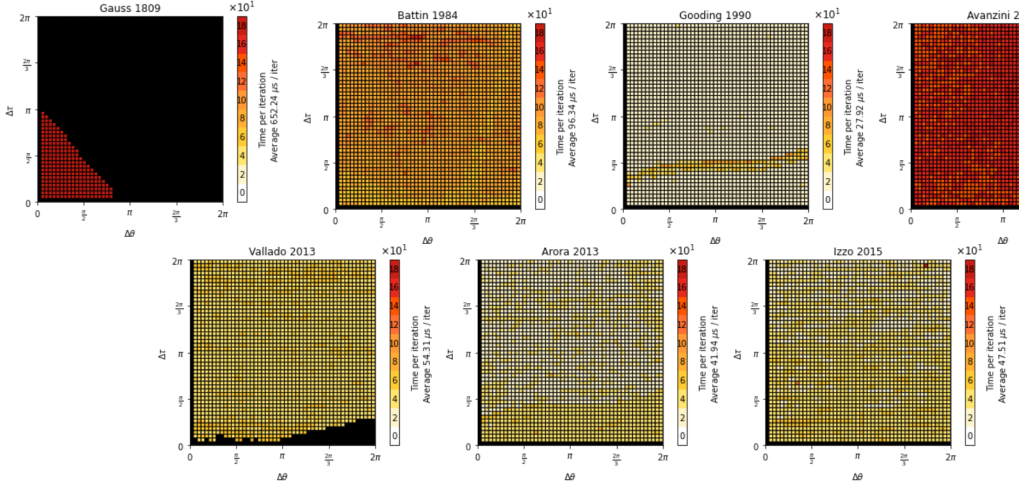
Table: Basic elements of each selected solver.

Universal formulation, rational formulae, high-order methods and f and g functions are seen to be the preferred methods.

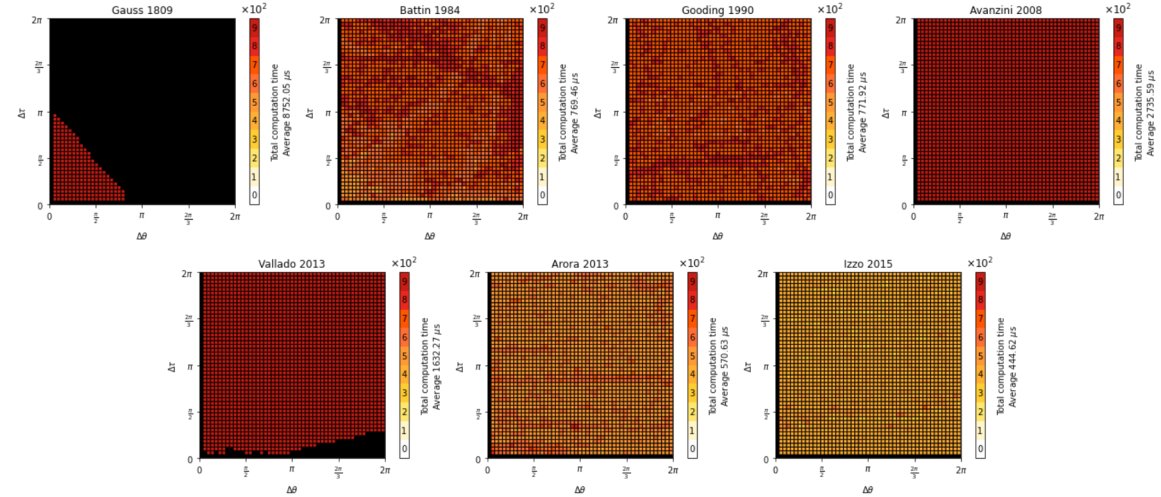
Performance comparison: number of iterations



Performance comparison: time per iteration



Performance comparison: total computation time



Performance results

Method	Performance data			
	Mean NOI	**Mean TPI**	**Mean TCT**	**Mean IW**
Gauss 1809	10.96	652.24 μs/iter	8752.05 μs	81.68 %
Battin 1984	3.82	96.34 μs/iter	769.49 μs	47.83 %
Gooding 1990	2.67	27.92 μs/iter	771.92 μs	9.66 %
Avanzini 2008	4.65	168.64 μs/iter	2735.59 μs	28.67 %
Arora 2013	2.14	41.94 μs/iter	570.63 μs	15.73 %
Vallado 2013	24.32	54.31 μs/iter	1632.27 μs	80.92 %
Izzo 2015	2.33	47.51 μs/iter	444.62 μs	24.90 %

Table: Critical performance data for each solver.

where the iteration workload is given by $\overline{\text{IW}} = (\overline{\text{NOI}} \times \overline{\text{TPI}})/(\overline{\text{TCT}})$

Conclusion

The performance classification as considering the convergence of a solver, the TCT and the iteration workload:

Solver Name	Robustness (x10p)	Multi-rev (x10p)	Mean TCT (x5p)	Mean IW (x1p)	Points	Ranking
Gauss 1809	0	0	1	7	12	7th
Battin 1984	1	0	5	5	30	4th
Gooding 1990	1	1	4	1	41	3rd
Avanzini 2008	1	0	2	4	24	5th
Vallado 2013	0	0	3	6	21	6th
Arora 2013	1	1	6	2	52	2nd
Izzo 2015	1	1	7	3	58	1st

Table: Classification and ranking for the solvers.

Future work

The results presented in this work might be enhanced and expanded in a future by including:

- Error code palette.
- Multi-revolution and parallelization analysis.
- Implement more solvers.
- Memory usage and profiling.

Issues and new features can be tracked via:

- Issue board: https://github.com/jorgepiloto/lamberthub/issues
- Pull requests: https://github.com/jorgepiloto/lamberthub/pulls