Turnitin originaliteitsrapport

<u>Master Thesis</u> Door Stacha Petrovic

Van Thesis Plagiarism Checker (Master Thesis AE _55472_1)

Verwerkt op 26-feb.-2017 6:20 PM CET
ID: 776161428
Woordtelling: 51424

Similariteitsindex	Similariteit per bron	
5%	Internet Sources:	3%
	Publicaties:	4%
	Student Papers:	2%

---

bronnen:

---

1   < 1% match (publicaties)

Advances in Aerospace Guidance Navigation and Control, 2013.

---

2   < 1% match (publicaties)

"Space Engineering", Springer Nature, 2016

---

3   < 1% match (publicaties)

Bergsma, Michiel, and Erwin Mooij. "Application of Taylor Series Integration to Reentry Problems", AIAA Atmospheric Flight Mechanics Conference, 2016.

---

4   < 1% match (Internet vanaf 19-jan.-2014)

http://www.acfr.edu.au/pdfs/training/navSystemsDesign/navigation.pdf

---

5   < 1% match (Internet vanaf 03-mei-2010)

http://www.cs.cmu.edu/afs/cs.cmu.edu/user/cjp/matlab/SC_doc/ACSTheory.pdf

---

6   < 1% match (publicaties)

Indumathi, G., V. P. M. B.A.A. Ananthakirupa, and M. Ramesh. "Architectural Design of 32 Bit Polar Encoder", Circuits and Systems, 2016.

---

7   < 1% match (publicaties)

Jara-Orué, H.M., and B.L.A. Vermeersen. "Tides on Jupiter's moon Ganymede and their relation to its internal structure", Netherlands Journal of Geosciences – Geologie en Mijnbouw, 2016.

---

8   < 1% match (Internet vanaf 30-apr.-2014)

http://www.slideshare.net/NorthWestMax/linked-in-slides-quantum-mechanics

---

9   < 1% match (papers van studenten vanaf 22-jan.-2016)

Cursus: AE4010 Research Methodologies (2015-2016 Q2) _53420_1
Opdracht:
Paper-ID: 622298831

---

10   < 1% match (Internet vanaf 14-feb.-2017)

http://authors.library.caltech.edu/55016/

---

11   < 1% match (Internet vanaf 18-feb.-2017)

http://elib.suub.uni-bremen.de/edocs/00103193-1.pdf

---

12   < 1% match (publicaties)

Methods of Model Based Process Control, 1995.

---

13 < 1% match (publicaties)
Scott, James, and Michael Martini. "High Speed Solution of Spacecraft Trajectory Problems Using Taylor Series Integration", AIAA/AAS Astrodynamics Specialist Conference and Exhibit, 2008.

14 < 1% match (publicaties)
ESS, ROBERT. "Atmospheric effects on Martian aerocapture", 17th Atmospheric Flight Mechanics Conference, 1990.

15 < 1% match (papers van studenten vanaf 21-okt.-2011)
Submitted to University of Adelaide on 2011-10-21

16 < 1% match (Internet vanaf 08-mei-2016)
http://abcm.org.br/app/webroot/anais/cobem/2013/_PDF/1510.pdf

17 < 1% match (publicaties)
Encyclopedia of Applied and Computational Mathematics, 2015.

18 < 1% match (publicaties)
Karp, Ashley C., Barry Nakazono, Joel Benito Manrique, Robert Shotwell, David Vaughan, and George T. Story. "A Hybrid Mars Ascent Vehicle Concept for Low Temperature Storage and Operation", 52nd AIAA/SAE/ASEE Joint Propulsion Conference, 2016.

19 < 1% match (papers van studenten vanaf 14-jun.-2016)
Submitted to Technische Universiteit Delft on 2016-06-14

20 < 1% match (publicaties)
Chern, J.S.. "Optimal trajectory for vertical ascent to geosynchronous Earth orbit", Acta Astronautica, 199307

21 < 1% match (Internet vanaf 16-apr.-2012)
http://audiophile.tam.cornell.edu/~als93/Publications/MeiOckSch99.pdf

22 < 1% match (publicaties)
Kazeminejad, B.. "Huygens' entry and descent through Titan's atmosphere-Methodology and results of the trajectory reconstruction", Planetary and Space Science, 200711

23 < 1% match (Internet vanaf 04-jan.-2015)
http://www.hobartuk.com/sites/all/files/site/cad/amx-10a.dwg

24 < 1% match (publicaties)
Bastante, Juan C., José Vasconcelos, Miguel Hagenfeldt, Luis F. Peñín, João Dinis, and José Rebordão. "Design and development of PROBA-3 rendezvous experiment", Acta Astronautica, 2013.

25 < 1% match (Internet vanaf 08-apr.-2010)
http://www.forward.com.au/products/simgauss/SimGauss.pdf

26 < 1% match (publicaties)
Lin, C.L.. "On the design of fuzzified trajectory shaping guidance law", ISA Transactions, 200904

< 1% match (publicaties)

27   Stenbaek-Nielsen, Hans C., and Peter Jenniskens. "High-Speed Spectrographic Photometry of the Stardust Sample Return Capsule Around Peak Deceleration", Journal of Spacecraft and Rockets, 2010.

28   < 1% match (papers van studenten vanaf 05-apr.-2015)
Submitted to Technische Universiteit Delft on 2015-04-05

29   < 1% match (publicaties)
D. Charlesworth. "Sex differences in fitness and selection for centric fusions between sex-chromosomes and autosomes", Genetics Research, 04/1980

30   < 1% match (papers van studenten vanaf 22-mrt.-2010)
Submitted to Technical University Delft on 2010-03-22

31   < 1% match (publicaties)
Pellegrini, Etienne, Ryan P. Russell, and Vivek Vittaldev. "F and G Taylor series solutions to the Stark and Kepler problems with Sundman transformations", Celestial Mechanics and Dynamical Astronomy, 2014.

32   < 1% match (Internet vanaf 05-mrt.-2014)
http://ksklum.tripod.com/PhysicsNotes.html

33   < 1% match (Internet vanaf 23-mei-2016)

https://aaltodoc.aalto.fi/bitstream/handle/123456789/17842/master_van_Gijlswijk_Rian_2015.pdf?isAllowed=y&sequence=1

34   < 1% match (Internet vanaf 18-mrt.-2010)
http://www.cparity.com/projects/AcmClassification/samples/285863.pdf

35   < 1% match (papers van studenten vanaf 31-aug.-2015)
Submitted to Technische Universiteit Delft on 2015-08-31

36   < 1% match (Internet vanaf 21-sep.-2014)
http://hanspeterschaub.info/Papers/grads/JayMcMahon.pdf

37   < 1% match (Internet vanaf 23-jul.-2010)
http://scholar.lib.vt.edu/theses/available/etd-04282004-071227/unrestricted/dissertation.pdf

38   < 1% match (publicaties)
Cervone, Angelo, Emilio Rapposelli, and Luca d'Agostino. "A Simplified Model for the Evaluation of the DeltaV of Ascent Trajectories", 41st AIAA/ASME/SAE/ASEE Joint Propulsion Conference & Exhibit, 2005.

39   < 1% match (papers van studenten vanaf 30-apr.-2015)
Submitted to Universiti Kebangsaan Malaysia on 2015-04-30

40   < 1% match (Internet vanaf 02-jun.-2009)
http://www.aoe.vt.edu/~cdhall/courses/aoe4065/FDReports/DC04.pdf

41   < 1% match (Internet vanaf 13-jan.-2010)
http://spacewx.com/Docs/AIAA-G-003C-2008_V8%20Review%20Copy_wwv.pdf

42 < 1% match (Internet vanaf 16-mrt.-2016)
http://anorien.warwick.ac.uk/mirrors/CTAN/language/japanese/japanese-otf/vf/utfcgr0-h.vf

43 < 1% match (publicaties)
Bai, Xiaoli, and John L. Junkins. "Modified Chebyshev-Picard Iteration Methods for Orbit Propagation", The Journal of the Astronautical Sciences, 2011.

44 < 1% match (publicaties)
Urrutxua, Hodei, Manuel Sanjurjo-Rivo, and Jesús Peláez. "DROMO propagator revisited", Celestial Mechanics and Dynamical Astronomy, 2015.

45 < 1% match (publicaties)
Dolean, V.. "Convergence analysis of additive Schwarz for the Euler equations", Applied Numerical Mathematics, 200405

46 < 1% match (publicaties)
Shokouhifar, Mohammad, and Ali Jalali. "An evolutionary-based methodology for symbolic simplification of analog circuits using genetic algorithm and simulated annealing", Expert Systems with Applications, 2015.

47 < 1% match ()
http://www.geo-slope.com/NR/rdonlyres/550F6746-85C7-46DF-AF29-2A66B8BFDFCD/0/Manual_Seep3D.pdf

48 < 1% match (Internet vanaf 08-feb.-2017)
http://www.irs.uni-stuttgart.de/publikationen/index.html

49 < 1% match (Internet vanaf 18-dec.-2016)
http://ta.twi.tudelft.nl/users/vuik/numanal/kadar_afst.pdf

50 < 1% match (Internet vanaf 03-nov.-2015)
http://www.readbag.com/man-fsid-cvut-cz-cosmos-printing-ugcad

51 < 1% match (Internet vanaf 18-mei-2010)
http://sport1.uibk.ac.at/mm/refs/bib.html

52 < 1% match (Internet vanaf 18-feb.-2017)
http://sophia.dtp.fmph.uniba.sk/~fecko/kniha/cituje_sa.pdf

53 < 1% match (publicaties)
Thomas J. Böhme, Benjamin Frank. "Hybrid Systems, Optimal Control and Hybrid Vehicles", Springer Nature, 2017

54 < 1% match (publicaties)
Swetz, Keith M. Carey, Elise C. Bundrick. "Clinical Pearls in Palliative Medicine 2012. (CONCISE REVIEW FOR CLINICIANS)(Conference news)", Mayo Clinic Proceedings, Nov 2012 Issue

55 < 1% match (publicaties)
Gregory, David. "Swing Time.", Trailer Boats, July 2000 Issue

56 < 1% match (papers van studenten vanaf 20-jan.-2016)

Cursus: AE4010 Research Methodologies (2015-2016 Q2) _53420_1
Opdracht:
Paper-ID: 622320564

---

57   < 1% match (Internet vanaf 19-feb.-2017)
http://www.dlr.de/dlr/en/desktopdefault.aspx/tabid-10285/year-all/

---

58   < 1% match (Internet vanaf 27-mei-2016)
http://en.wikipedia.org/wiki/Tharsis

---

59   < 1% match (Internet vanaf 16-jan.-2017)
http://elib.dlr.de/102869/

---

60   < 1% match (publicaties)
G. Laan. "Efficiency and implementation of simplicial zero point algorithms", Mathematical Programming, 10/1984

---

61   < 1% match (papers van studenten vanaf 01-jul.-2009)
Submitted to Technical University Delft on 2009-07-01

---

62   < 1% match (Internet vanaf 26-mei-2014)
http://www.ip-super.org/res/Papers/reasoning-service-behavior-VLDB-250607.pdf

---

63   < 1% match (Internet vanaf 26-jan.-2016)
http://www.slideshare.net/arancor411/process-heat-transfer-principles-applications-and-rules-of-thumb

---

64   < 1% match (publicaties)
Whitehead, John. "Mars Ascent Propulsion Trades with Trajectory Analysis", 40th AIAA/ASME/SAE/ASEE Joint Propulsion Conference and Exhibit, 2004.

---

65   < 1% match (publicaties)
Cho, Hancheol, and Adam Yu. "New Approach to Satellite Formation-Keeping: Exact Solution to the Full Nonlinear Problem", Journal of Aerospace Engineering, 2009.

---

66   < 1% match (publicaties)
Paultre, . "Forced Response to Arbitrary Loading in Frequency Domain", Dynamics of Structures Paultre/Dynamics of Structures, 2013.

---

67   < 1% match (publicaties)
Ono, Masahiro, Austin Nicholas, Farah Alibay, and Joseph Parrish. "SMART: A propositional logic-based trade analysis and risk assessment tool for a complex mission", 2015 IEEE Aerospace Conference, 2015.

---

68   < 1% match (Internet vanaf 04-jan.-2017)
https://open.library.ubc.ca/cIRcle/collections/ubctheses/831/items/1.0096546

---

69   < 1% match ()
http://ncnyumc.org/connections/pdf/jun0407.pdf

---

70   < 1% match (publicaties)
IUTAM Symposium on Multiscale Problems in Multibody System Contacts, 2007.

---

71 < 1% match (publicaties)
Applications of Computer Technology to Dynamical Astronomy, 1989.

72 < 1% match (papers van studenten vanaf 20-dec.-2016)
Submitted to Covenant University on 2016-12-20

73 < 1% match (papers van studenten vanaf 21-okt.-2016)
Submitted to University of New South Wales on 2016-10-21

74 < 1% match (Internet vanaf 29-aug.-2010)
http://rwalther.free.fr/Ressources%20et%20cours%20Electronique%20-
%20Audio/(Ebook)Acoustic%20Echo%20Cancellation%20using%20Adaptive%20LMS.pdf

75 < 1% match (publicaties)
Real-Time Integration Methods for Mechanical System Simulation, 1991.

76 < 1% match (publicaties)
K-E Thylwe. Journal of Physics A Mathematical and General, 04/27/2001

77 < 1% match (papers van studenten vanaf 28-sep.-2009)
Submitted to Technical University Delft on 2009-09-28

78 < 1% match (papers van studenten vanaf 22-jun.-2016)
Submitted to Technische Universiteit Delft on 2016-06-22

79 < 1% match (papers van studenten vanaf 14-jun.-2009)
Submitted to Technical University Delft on 2009-06-14

80 < 1% match (papers van studenten vanaf 29-jan.-2013)
Submitted to University of Adelaide on 2013-01-29

81 < 1% match (Internet vanaf 23-apr.-2016)
http://media.proquest.com/media/pq/classic/doc/2834421421/fmt/ai/rep/NPDF?
_s=ADZlatEdwclz6js7dezF9qj4uU4%3D

82 < 1% match (Internet vanaf 17-mrt.-2016)
http://www.tet.com/news/tdy_08062012_2.asp

83 < 1% match (publicaties)
Kumar, K., I. de Pater, and M.R. Showalter. "Mab's orbital motion explained", Icarus, 2015.

84 < 1% match (Internet vanaf 13-jan.-2017)

https://engineering.purdue.edu/AAECourses/aae450/2013/Spring/AAE%20450%20Spring%202013%20Final%20Re

85 < 1% match (Internet vanaf 28-aug.-2016)
https://es.scribd.com/document/85327027/FLACS-Manual

86 < 1% match (Internet vanaf 20-mei-2014)
http://www.lhcsafetyreview.org/docs/black-hole-review.pdf

87 < 1% match (Internet vanaf 01-jun.-2016)
http://e-journal.uajy.ac.id/6285/6/TS513744.pdf

88 < 1% match (Internet vanaf 17-nov.-2016)
https://art.tools.ietf.org/id/draft-ietf-tls-cached-info-20.html

89 < 1% match (publicaties)
Lu, Ping, David Doman, and John Shierman. "Adaptive Terminal Guidance for Hypervelocity Impact in Specified Direction", AIAA Guidance Navigation and Control Conference and Exhibit, 2005.

90 < 1% match (publicaties)
Boukas, Evangelos, and Antonios Gasteratos. "Modeling Regions of Interest on Orbital and Rover Imagery for Planetary Exploration Missions", Cybernetics & Systems, 2016.

91 < 1% match (publicaties)
Carsten Schneider. "Application of unspecified sequences in symbolic summation", Proceedings of the 2006 international symposium on Symbolic and algebraic computation - ISSAC 06 ISSAC 06, 2006

92 < 1% match (publicaties)
"Artificial Neural Networks – ICANN 2009", Springer Nature, 2009

93 < 1% match (publicaties)
W. H. Enright. "Test Results on Initial Value Methods for Non-Stiff Ordinary Differential Equations", SIAM Journal on Numerical Analysis, 1976

94 < 1% match (publicaties)
Moler, . "7. Ordinary Differential Equations", Numerical Computing with MATLAB, 2004.

95 < 1% match (publicaties)
Thomas L. Wilson. "Wave Polarization", Astronomy and Astrophysics Library, 2009

96 < 1% match (publicaties)
Shang, Wei Guo, Jie Tang, Shengjing Ma, . "Impact time guidance law considering autopilot dynamics based on variable coefficients strategy for ", Mathematical Problems in Engineering, Annual 2015 Issue

97 < 1% match (publicaties)
Yi, Huizhi, Qinglin Duan, and T. Warren Liao. "Three improved hybrid metaheuristic algorithms for engineering design optimization", Applied Soft Computing, 2012.

---

werkstuktekst:

Faculty Aerospace, Track Spaceflight Mars Sample Return Ascent Trajectory Analysis using Taylor Series Integration Master Thesis S.D. Petrović T R AMJEACRTSOSRAYMAPNL EA LRYES TISUURSNI NAGS CTEANYTL O R SERIES INTEGRATION MASTER THESIS by S.D. Petrović

> 19**in partial fulfilment of the requirements for the degree of Master of Science in Aerospace Engineering at the Delft University of Technology, to be defended publicly on**

Thursday 23 March 2017. Supervisor: Supervisor abroad Thesis committee: Dr. ir. E. Mooij, Ir. R. Noomen, Dr. ir. R. S. Wilson, Dr. D. Stam, Ir. K. Cowan, TU Delft TU Delft NASA JPL TU Delft (Chair of committee)

> **49TU Delft An electronic version of this thesis is available at http://repository.tudelft.nl/. PREFACE**

Back in December 2013 Warren Gebbett gave a presentation on his work at the Jet Propulsion Laboratory (JPL) in Pasadena, California, USA and the opportunity for a new student to go and perform research at JPL. At this point I sent in my application together with eight other students. Then at the end of December I heard that I was invited for an interview in the first week of January 2014. In this interview it was concluded that I met all the requirements and that I was the perfect candidate to follow Warren up as the next student at JPL with financial backing of Dutch Space (now Airbus Defence and Space, the Netherlands). Financial backing was also provided by the Stichting Prof.dr.ir. H.J. van der Maas Fonds (Aerospace Engineering Faculty, TU Delft) and the Stichting Universiteitsfonds Delft (TU Delft). Communication with JPL was started in February 2014 and in March 2015 it was clear that I would be working for the Mars Program Formulation Office under the supervision of Roby Wilson (Inner Planet Mission Design Group, National Aeronautics and Space Adminis- tration (NASA) JPL). He told me to focus on subjects that dealt with Mars missions. At that point I was doing my internship at DLR Bremen on Lunar rocket ascent and descent, which lasted till June 2015. When I came back to Delft me and my supervisors Erwin Mooij (rockets, trajectories, entry and descent) and Ron Noomen (mission design and orbit analysis) agreed that it would be best to perform a study on these Mars subjects to prepare for my visit to JPL and to formulate proposal thesis topics. The first week at JPL I presented these initial thesis topics to both people from the Inner Planet Mission Design Group and the Mars program formulation office. The next few weeks were spent choosing and refining one of these topics. In the months that followed a literature study was performed to determine the final thesis topic. In the beginning of 2016 I got the green light to start my thesis research and so the journey that would last slightly longer than I expected started. It was not easy, and I quickly realised that I had misjudged the amount of time I would need debug my code. Unfortunately, this was unavoidable. Completely rewriting models and equations happened so often that I lost count. All of this caused a lot of mental strain which was not always easy to deal with. But with the continuous support of my family, my friends, all my supervisors and colleagues I was still able to keep hope and push on. I never gave up. And even though the research was slower than I expected and a lot of compromises had to be made in the end I am happy with the results and the amazing opportunity that was given to me. This is why I would personally like to thank my TU Delft supervisors Ron Noomen and Erwin Mooij, for always keeping faith and sticking with me till the end. They were never afraid to speak their mind which always made me stronger. I would also like to thank my supervisor at JPL, Roby Wilson for the great support and help during my year-long stay in his group. His input always made me critically thing about what I was doing and showed me a slightly different view from the academics that I was used to. During my time abroad I often had some personal struggles as well, which is why I would like to thank my parents, Robert and Linda Petrović for both the financial and mental support that they were able to give me. I would also like to thank the Rutkowski family for hosting me for several periods during my stay in the US and for being my family away from home. And finally, I want to thank all of my friends for always being there for me. Without all of you this thesis research would never have become a reality, which is why I am glad to be able to present the research that I performed the past year in this report. S.D. Petrović Delft, The Netherlands, February 2017 i ABSTRACT Mars Sample Return is a mission concept that has been proposed many times in the past two decades. It is currently still being investigated by different space agencies around the world. The proposal of the Jet Propulsion Laboratory has reached the phase where different elements of this mission concept have been approved and are currently in the design phase. The complete campaign architecture involves three different missions: the first mission will have the objective to

> **67collect samples and** store **them, the second**

mission will then collect these stored

mission will collect the orbiting sample and bring it back to Earth. The first part and the final part are currently secondary objectives for the Mars 2020 and Mars 2022 missions respectively. However, the second mission has not yet reached the approved design phase. An important objective of this second mission is to launch the samples into an orbit around Mars. This can be done using a Mars Ascent Vehicle. For such a mission profile, the ascent trajectory has to be simulated during the pre-design phase. An integrator can be used to simulate a Martian ascent by propagating the state of the vehicle in time. Numerical integration methods can be used for such a simulation, however in recent years another integration method has shown potential in both orbital trajectory problems as well as (re)-entry problems. This method is called Taylor Series Integration (TSI) and is based on the propagation of the state using Taylor series. The effectiveness of this integration method for a Mars ascent scenario is tested against the Runge-Kutta-Fehlberg 7th (8th)-order (RKF7(8)) numerical integrator. Therefore the goal of this research is to determine the speed of convergence, the accuracy and the computational speed of both methods. Using a set of spherical kinetic and dynamic equations in the rotating frame of Mars, taking into account the gravitational acceleration, the aerodynamic drag and the vehicle thrust, a trajectory was computed utiliz- ing both integrators. The TSI method was both more accurate and had a faster speed of convergence, however in this research the computational speed was still slower than the RKF7(8) reference numerical integrator. TSI does, however, show other advantages compared to RKF7(8) as well: far fewer function calls, theo- retically unlimited order (an optimum of order 20 was determined), can readily handle discontinuities, no interpolation is required afterwards resulting in a theoretically unlimited resolution and it shows less varia- tion in the number of evaluations as a function of the error tolerance. A major disadvantage though of TSI is that it can take a long time to set it up since it requires all equations before starting the integration. An analysis on the influence of different parameters on a Mars ascent trajectory was also analysed us- ing TSI. Parameters that influence the trajectory are: launch latitude, launch flight-path angle and launch heading angle. It can be concluded that TSI is a viable integration method for use in ascent problems considering the rapid convergence and the accuracy of the results. ii

NOMENCL ATURE Constant γa Ratio of specific heats µ Mars standard gravitational parameter Ω_M Rotational velocity of Mars g_0 Standard gravitational acceleration at Earth sea level M_a Molecular mass of the Martian atmosphere R_a Molar gas constant R_{MOLA} Mars MOLA reference radius (also R_M) S Reference area 1.35 $4.2828314 \cdot 10^{13} m^3/s^2$ $7.088 \cdot 10^{-5}$ rad/s 9.80665 m/s2 0.04334 kg/mol 8.3144598 m2 kg s−2 K−1 mol−1 3396·103 m 0.091 m2 Greek α Root-finding curve coefficient γ Flight-path angle δ Latitude ∆V Change in velocity ε_T Thrust elevation angle η Numerical approximation η Step multiplication factor λ Inertial longitude ρ Atmospheric density τ Central-body fixed longitude τ Local error tolerance χ Heading angle (or azimuth) φ General Euler angle Φ Increment function ψ_T Thrust azimuth angle ω_P Off-set angle between inertial frame and zero meridian at inertial frame set time Roman a Acceleration a Semi-major axis a Speed of sound vi -

78**rad rad m/** s **rad - - rad** kg/m3 **rad - rad rad - rad rad m/** s2 **m**

m/s NOMENCLATURE aD Drag acceleration ag Gravitational acceleration ai,j RKF coefficients of the previous time-derivatives aT Thrust acceleration bi RKF weights for each point ci RKF step-size weights C Transformation matrix coefficient CD Drag coefficient D Drag f Limit function fn Auxiliary placeholder function gn Auxiliary placeholder function h (Current) step-size h Altitude i Inclination Isp Specific impulse

63**î Unit vector in x-direction ĵ Unit vector in y-direction k** Order derivative **k**

Runge-Kutta increment derivative k̂ Unit vector in z-direction K Maximum order mMAV MAV mass mprop Propellant mass M Mach number n Order pn Polynomial coefficient r Radius r Radius vector Ra∗ Specific gas constant t Time t0 Initial time tO Difference between current time and the inertial frame set time T Thrust vii m/s2 m/s2 - m/s2 - - - - N - - - s m rad s - - - - - - kg kg - - - m - m2 s−2 K−1 s s s N NOMENCLATURE Ta Atmospheric temperature Tn,k Truncation error T Transformation matrix un Auxiliary derivative V

50**Velocity Vx Velocity in x-direction Vy Velocity in y-direction Vz Velocity in z-direction**

V Velocity vector wn,i Auxiliary function x State variable x x-position coordinate xn Auxiliary equation xn nth state variable xi+1 Next data point xi Current data point X Taylor Series coefficient (also Xn) X State vector y y-position coordinate z z-position coordinate viii

23**K - - - m/s m/s m/s m/s - - - m - - - - - - m m**

ABBREVIATIONS AB4 Adams-Bashforth 4th-order AB6 Adams-Bashforth 6th-order ABM4 Adams-Bashforth-Moulton 4th-order ABM12 Adams-Bashforth-Moulton 12th-order BF Body-Fixed

11**DLR German Aerospace Center (Deutsches Zentrum für Luft- und Raumfahrt)**

DOPRIN8(7) Dormand-Prince 8th (7th)-order ESA European Space Agency FPA Flight-Path Angle GLOM Gross Lift-Off Mass GMT Greenwich Mean Time

14**GRAM Global Reference** Atmospheric **Model JPL Jet Propulsion Laboratory**

MAV Mars Ascent Vehicle MCI Mars-Centred Inertial MCMF Mars-Centred Mars-Fixed MOLA Mars Orbiter Laser Altimeter MSR NASA PaGMO RF RHR RKF4(5) RKF5(6) RKF7(8) RKF RK4 RKN12 s/c SC14 SG SNOPT SSTO TSI Tudat Mars Sample Return National Aeronautics and Space Administration Parallel Global Multi-objective Optimizer Frame of Reference Right-hand-rule

25**Runge-Kutta-** Fehlberg **4th** (5th **)-order Runge-Kutta-** Fehlberg 5th (6th **)-order Runge-Kutta-Fehlberg** 7th (8th **)-order Runge-Kutta-Fehlberg**

Runge-Kutta 4th-order Runge-Kutta-Nyström 12th-order Spacecraft Störmer-Cowell 14th-order Shampine-Gordon Sparse Nonlinear Optimizer Single Stage to Orbit Taylor Series Integration TU Delft Astrodynamics Toolbox ix 1 INTRODUCTION Mars Sample Return (MSR) is a mission concept that has

been proposed many times in the past two decades. Today, a large amount of research into this mission is being carried out. Even though it is not yet an official project proposal,

> 82National Aeronautics and Space Administration (NASA) Jet Propulsion Laboratory (JPL) is currently working on

pre-cursor missions to eventually fly an MSR mission. The reason NASA would like to get the samples back to Earth compared to analysing it directly on Mars is the fact that the laboratories on board the current rovers are limited. Should the samples be returned to Earth however, they can be analysed in any laboratory using all available resources. To prepare for such a mission, research is being conducted on different aspects of MSR, such as the ascent trajectory of the Mars Ascent Vehicle (MAV) responsible for transporting soil samples into a Martian orbit. Chapter 2 explains the reasoning behind focusing on the as-cent trajectory of the MSR as opposed to many other research topics that were available. In the field of ascent trajectory simulation, numerical integrators such as Runge-Kutta-Fehlberg (RKF) are often used to simulate the trajectory propagation. However, in recent years, another integration method called Taylor Series Integra- tion (TSI) has shown promising improvements compared to RKF in the field of both orbit trajectory analysis as well as (re-)entry cases, however TSI has not yet been used to simulate an ascent trajectory. This is why TSI is the focus of this thesis research and thus the primary research question reads: "Does the TSI method show similar performance improvements over RKF for the Mars ascent case as was observed for the orbital trajectories and (re-)entry cases?". Chapter 2 also discusses some of the early assumptions, and the required reference systems. To be able to simulate the acent trajectory of the MAV, real-life flight conditions have to be modelled in such a way that the results approximate the real ascent. The models used in this research are described in Chapter 3 and are chosen such that the problem is not too complex but at the same time still represents the real-life scenario well enough to be able to draw meaningful conclusions. As mentioned, different standard integrators are available to be used in such a problem and a selection of those is described in Chapter 4. Out of those methods, the best comparison integration methods turned out to be the RKF methods. From the available RKF methods, Runge-Kutta-Fehlberg 7th (8th)-order (RKF7(8)) was chosen for the final comparison with TSI because it showed the best performance in the ascent case. Because the RKF7(8) method is a standard method that is widely available, the main focus and efforts of this research is put on the TSI method. In Chapter 5 TSI is explained in detail and three different formulations are described: a normal cartesian formulation, a unit vector cartesian formulation and a spherical formulation. Eventually it was opted to use the spherical formulation. Both RKF7(8) and TSI are incorporated into a simulation tool, which is capable of computing the ascent trajectory of the MAV given a number of initial conditions. The complete set-up of the simulation tool is described in Chapter 6 and includes both existing software as well as software that had to be developed. During and after the development of the simulation tool, every aspect of the program had to be verified and validated. As described in Chapter 7, this was done in different steps. Two different simulated ascent cases were used to determine the validity of the outcomes of the simulation program. The same cases were then used for the comparison between RKF7(8) and TSI but also to perform a sensitivity analysis on the trajectory. All the results and the corresponding analyses are described in Chapter 8. From these analyses a number of conclusions can be drawn which are outlined in Chapter 9. During the entire research process, many more questions were formed and many more improvements were envisioned. Unfortunately, not all of these 1 questions could be answered nor all improvements implemented in the scope of this thesis. This is why the same chapter also includes a recommendations section for future research. Hopefully this research is able to reduce the knowledge gap that currently exists concerning the performance of TSI for an ascent trajectory case. 2 BACKGROUND One of the current focal points of JPL is Mars and the exploration of the Martian system 1. There are currently two planned missions to Mars: InSight and Mars 2020. InSight is a mission similar to the Viking and the Phoenix missions, and primarily based on the last one. It includes a lander that will perform experiments on the surface of Mars at a fixed location. Mars 2020 includes the next Mars rover and is based on the Curiosity rover. It has a similar design, but will carry different scientific instruments and will focus more on finding life on Mars. Mars 2020 is part of a larger JPL effort to eventually return samples from the Martian surface back to Earth. This mission concept has been on the drawing board for several years now, but finally seems to be getting closer to approval. This sample return mission, or MSR, will consist of three different systems and is spread out over three different missions as shown by the proposed mission

architecture in Figure 2.1. Mars 2020 is the first step (denoted by 1 in the figure) in this plan where the rover will collect samples from the surface of Mars. These samples will then be put into containment tubes and dropped on the surface of Mars. A second mission will then carry a system that will collect these samples from the surface and deliver them into a Martian orbit (see number 3 in the figure) (Shotwell et al., 2016). Then a third mission, currently proposed as the Mars 2022 orbiter (see number 2 in the figure), will collect the sample containment sphere in orbit and eventually deliver it back to Earth (Woolley et al., 2011). Because of the potential of the MSR mission concept and the amount of research that still has to be done on this subject, the focus was put on this JPL effort when looking at research subjects. Figure 2.1: Proposed MSR campaign architecture (Vaughan et al., 2016). 1All past, current, future and proposed JPL missions: http://www.jpl.nasa.gov/missions/ [Accessed 17 November 2016] 3 2.1. CHOSEN MISSION 2 . 1 .

> 69 **C H O S E N M I S S I O N**

Mars 2020 is an approved mission that is currently in the design phase, and Mars 2022 is a proposed mission that will have to be built to enhance the current communication capabilities between Mars and Earth, which is its primary objective. Therefore it is almost certain that these two missions will fly and it should be noted that for both these missions, MSR is a secondary objective. However, this does not guarantee that the entire MSR endeavour will be realised. This is because there is no official approval yet for the second system that will actually bring the samples from the surface back to Earth. However, this does not mean that no research is being done on the return system. Many designs have been envisioned over the past few years and even now engineers cannot decide on the best design. Shotwell et al. (2016) shows two of the proposed designs that are currently being considered. One design involves a mobile launch system based on the Curiosity rover, which will fetch the samples and then directly put them into the spherical container, as depicted in Figure 2.2a. Once this container is filled, the sphere will be placed on top of the MAV, which the rover will be carrying on its back, and launched into Mars orbit. The second design (Figure 2.2b) uses two systems: one fetch rover that will grab the samples and return them to the lander, and the lander containing the MAV. Once the rover returns with all the samples, the sample sphere will be put on top of the MAV and launched into orbit. There are advantages and disadvantages to both of these systems and more research is being done to determine the best option. What both of these systems have in common though, is that they have to launch the sample sphere into orbit using an MAV. An ascent on another celestial body with an atmosphere has never been attempted before, which makes it a crucial part of the MSR campaign. This is also why in the past several years, many researchers have looked at the MAV ascent trajectory problem. (a) (b) Figure 2.2: Two of the proposed concepts for the retrieval of the samples (a) a rover carrying the MAV powered by a radioisotope thermoelectric generator (left) or solar panels (right), (b) a separate lander and fetch rover (Shotwell et al., 2016) 2.2. PREVIOUS MARS ASCENT RESEARCH Not only JPL, but also many other institutions around the world are working on the Mars ascent problem, for instance

> 57 **the German Aerospace Center (Deutsches Zentrum für Luft- und Raumfahrt) (DLR). A** selection **of** reference **research** is provided **in**

Table 2.1. The papers of 2015 and 2016 are based on currently ongoing research and thus represent intermediate findings. Most of the papers before 2015 focus on an older MAV concept, however, the newer JPL research focuses more on a Single Stage to Orbit (SSTO) design. This is why this thesis research will also be focusing on the ascent of an SSTO MAV. Much of the research that was done used either publicly available simulation software or trajectory simu- lation software that was developed in-house. There are several ways in which an ascent trajectory can be simulated, and these different methods of how to simulate such an ascent can result in different simulation times. Therefore, the methods of ascent trajectory propagation and simulation deserve a closer look. 2.3. RESEARCH FOCUS Table 2.1: Previous and current Mars ascent trajectory research. Author Organisation Research focus Fanning & Pierson (1996) Iowa State University (US) Two-stage MAV max payload Mars Ascent trajectories Desai et al. (1998) NASA Langley and JPL (US) MAV design analysis including configuration, GLOM and ascent trajectory determination Whitehead (2004, 2005) Lawrence Livermore National

Laboratory (US) Optimised MAV trajectory for a GLOM of 100 kg Di Sotto et al. (2007) DEIMOS Engenharia and ESA (Portugal/Europe) MAV ascent trajectory optimisa- tion and orbit characterisation Woolley et al. (2011) JPL (US) Conceptual mission design strategies for the MSR ascent phase Trinidad et al. (2012) Northrop Grumman and JPL (US) MAV baseline design and general system study Dumont (2015) DLR (Germany) Mars ascent trajectory simula- tions Woolley (2015) JPL (US) MAV ascent simulation model for mass and performance estima- tion Benito & Johnson (2016) JPL (US) MAV ascent trajectory optimisa- tion for different MAV configura- tions 2.3. RESEARCH FOCUS To compute an ascent trajectory, the initial state is taken and, using a proper description of the dynamical environment, the state is propagated until a final condition is met. In most of the research mentioned in Table 2.1 this was done using numerical integration methods. However, often it is not mentioned which specific methods were used. When looking at integration methods, there is a number of methods that are widely used for space-related problems, such as the standard integration methods (more information on these standard integration methods can be found in Chapter 4). In recent years however, a method that has not generally been applied to spaceflight problems has seen potential to increase performance, resulting in faster and more accurate results. This method is called Taylor Series Integration (TSI) and was first used in a spaceflight trajectory problem by Montenbruck (1992b). A few years later, a comparison between a higher-order Runge-Kutta method and TSI was performed on orbital trajectory problems by Scott & Martini (2008) showing that the application of TSI to such problems can be very beneficial. Bergsma & Mooij (2016) proved that TSI shows similar improvements compared to Runge- Kutta-Fehlberg for re-entry cases, where TSI was both faster and more accurate. However, TSI has not yet been applied to ascent cases, leaving a very important knowledge gap that has to be filled. This is why the Mars ascent trajectory propagation is a good additional test case to add to the knowledge of TSI performance in space-related problems. TSI will have to be compared to a standard integration method to determine its performance. From the two previous cases (orbital trajectories and re-entry) it is expected that TSI will be faster and more accurate than the standard integration methods. In this thesis, it will be tested whether or not this also holds for ascent cases. Because if this is true, TSI could for instance be used in guidance and navigation systems on board of the MAV during ascent to provide faster and more accurate predictions of the trajec- tory as well as during simulations that are done beforehand. However, before TSI can be tested, an accurate representation of an ascent scenario has to be defined. 2.4. MISSION DEFINITION To analyse the Mars ascent problem, a mission will have to be defined from which the requirements for the simulation tool can be derived. The goal of the mission is to put the sample container sphere in a circular 2.5. MAV DESIGN orbit around Mars. To do this, a vehicle called the MAV is used to ascent from the Martian surface into that circular orbit. The MAV should be able to launch from anywhere on the Martian surface into any direction. This means that the simulation tool has to be capable of using any combination of launch altitude, latitude, longitude, flight-path angle and heading angle. The tool should also be capable of starting the simulation assuming a MAV that is standing still. The vehicle also has a propulsion system, which will be used to pro- vide the thrust acceleration. This thrust is required to overcome the gravitational acceleration of Mars and the aerodynamic drag caused by the Martian atmosphere as well as provide enough energy to put the sam- ple container in a circular orbit once the desired orbital altitude is reached. The MAV should also be able to change directions during flight, which in this research is simulated by a gimballed nozzle. This is required be- cause the circular orbit could have any desired inclination, which also means that the simulation tool should be able to imitate an inclination correction if this is needed. When the simulation tool conforms to all these requirements, the resulting trajectory with the corresponding mission parameters should be a good repre- sentation of an actual Mars ascent in the MSR framework. 2.5. MAV DESIGN For this research it was decided to model the simulation to the currently proposed MAV ascent scenario. Therefore, the most up to date representation of the MAV will be used. Over the past 20 years there have been many different iterations of the MAV. Table 2.2 shows a number of different studies that focused on MAV design based on the requirements of that period. It also describes the eventual chosen designs with the corresponding propellants. Many requirement iterations have changed the concept designs. Some of the earliest concepts were described by Desai et al. (1998); Guernsey (1998); Whitehead (1997) and Stone (1999)2 and were prominently focussed on two-stage liquid-propellant rockets. After a change in mission requirements in the early 2000's the leading baseline design changed to a two-stage solid-propellant rocket (Stephenson, 2002; Stephenson & Willenberg, 2006; Whitehead, 2005). However, the requirements kept changing and the design kept getting better defined by (among others) Mars Program Planning Group (2012); Mungas et al. (2012); Sengupta et al. (2012); Trinidad et al. (2012) resulting in a two-stage liquid-propellant rocket as the updated baseline. The past four years however have again seen a slight change in the design. Most of the MAV work is now

being carried out in-house at JPL. Research is still being performed on the two-stage solid-propellant rockets, but SSTO designs are showing that they have an overall better perfor- mance. Recently presented designs include an SSTO liquid-propellant rocket (Vaughan et al., 2016) and an SSTO hybrid-propellant rocket as presented by Karp et al. (2016). These two papers however focused more on the propulsion systems. Because the mission requirements kept changing leading to many different MAV designs, it can become difficult to keep track of all the differences. Shotwell (2016) provides a good overview of the different designs leading up to the current baseline designs. Unfortunately, not much has been published on the specific flight parameters, such as drag coefficient and reference area, of these new designs. Fortunately, the size requirements of the MAV have not changed (much) between the 2012 studies and the current designs. Therefore these parameters can be approximated by look- ing at some of the previous baseline designs such as the two-stage liquid-propellant rocket shown in Fig- ure 2.3. This concept by Trinidad et al. (2012) had a diameter of approximately 340 mm, resulting in an approxi- mate cross-sectional area of 0.091 m2. Unfortunately, the paper does not mention anything about the drag coefficient. However, Whitehead (2004) provided a general drag coefficient graph

45**as a function of the Mach number for a**

generalized shape very similar to most MAV concepts. Therefore, this simplified relation, shown in Figure 2.4 is used in this thesis research as well. These two parameters are crucial in determining the drag force acting on the MAV during the ascent. This drag force acts on the MAV in what is called the aerodynamic frame. In this research it is assumed that both the

1**angle of attack and the side-slip angle**

are zero, such that the aerodynamic frame and the body frame coincide. One of the problems with trajectories is that different perturbations act differently on the vehicle and are easier to express in different reference frames using different formulations. In the end, however, the state will have to be expressed in one reference frame, which means that the drag force will have to be transformed to another frame. More information on these different frames is provided in Section 3.1. 2 Presentation slides: http://www.slideshare.net/cliffordstone/rice-mars-nov99 [Accessed 25 October 2015] 2.5. MAV DESIGN Table 2.2: Previous MAV studies. Author Researched rocket de- Researched and pro- Requirements year sign(s) posed propellant(s) Whitehead (1997) two-stage rockets (com- parison study) solid and liquid (and gel recommended as well) Guernsey (1998) two-stage rocket 2x liquid 1996 Stone (1999)2 Desai et al. (1998) two-stage rocket 2x liquid two-stage rocket hybrid Stephenson (2002) two-stage rocket (three different designs) 2x solid (best), solid and liquid or hybrid and 2x gel (best) Whitehead (2005) one-, two- and three- stage rockets (variational study) solid and liquid 2000 Stephenson & Willenberg (2006) two-stage rocket 2x solid Sengupta et al. (2012) two-stage rocket 2x liquid Trinidad et al. (2012) 1 to 4 stages (comparison study) two-stage rocket (best) solid, liquid, hybrid and gel (comparison) 2x liq- uid (best) 2012 Mungas et al. (2012) single-stage rocket liquid mono-propellant Mars Program Planning Group (2012) undefined rocket solid and liquid Vaughan et al. (2016) single-stage rocket liquid 2014 Karp et al. (2016) single-stage rocket hybrid Figure 2.3: Baseline design by Trinidad et al. (2012) on which the MAV design used in this research is mainly based. Figure 2.4:

20**Drag coefficient as a function of Mach number**

(Whitehead, 2004) 3 MODELS The MAV and the corresponding ascent can be modelled in many different ways. Depending on the desired accuracy of the final solution compared to the real-life expected trajectory, different perturbations and free- dom in motion can be defined and included, ranging from a low-fidelity (3DOF) to a high-fidelity program (6DOF). In a 3DOF system only the translational motion is taken into account, in 6DOF the rotational motion is also included, which makes it more accurate and more representative of a real-life scenario. The main goal of this thesis is to determine if TSI is an integrator that can be used for ascent cases and de- termine its performance compared to previously established methods. This comparison can be done with a less accurate representation of reality, because

this allows for faster computations. Also, should the model be a 6DOF system, the extra accuracy would not have an added value because of all the errors in the assumptions made. Therefore a 3DOF system will suffice for the purposes of this research. In this 3DOF simulation the MAV system can be modelled as a point-mass system with different accelerations acting on it. The change in position can be described by the kinematic equations and the change in velocity by the corresponding dynamic equations. The change in mass is described by the mass-flow. All these equations are provided in Section 3.1. For this model, three perturbing accelerations will be taken into account: gravity, thrust and at- mospheric drag. The effect of third bodies can be neglected because of the short mission duration. All the accelerations are described in Sections 3.2 to 3.4, respectively. At the end of the integration, an impulsive shot is executed to circularise the orbit as described in Section 3.5. 3.1. STATE AND STATE DERIVATIVES In this research problem, different perturbations are best modelled in different reference frames. One exam- ple is the drag acceleration acting in the aerodynamic frame. However, the integration will be performed in the inertial frame. A collection of all the different frames required is shown in Figure 3.1. This means that the accelerations have to be translated into the inertial frame through the presented frames. One way to do this is by using so-called transformation matrices (T). These transformation matrices are composed of rotations around different axes over an associated angle φ. An alternative would be to use quaternions instead, which make use of complex numbers. However, due to the workings of TSI this was not easily implemented here which is why normal Euler angle transformations were used as the baseline for the rotation equations. Figure 3.2 shows a general rotation around the x-axis. When this is set in a matrix, the transformation over the

70**x-, y- and z-axis** can be defined **in the**

respective transformation matrices as described by Equation (3.1). ⌈ 1

8**0 0** ⌉⌈**cos φ − sin φ**⌉ 0 ⌈ **cos φ sin φ 0** ⌉ Tx **(φ) =** ⌊**0 cos φ sin φ** ⌋ , Ty **(φ) =** ⌊ **0 1 0** ⌋ , Tz **(φ) =** ⌊**− sin φ cos φ 0**⌋ (3.1) 0 − **sin φ cos φ sin φ 0 cos φ 0 0 1**

Besides different reference frames, there are also different coordinate systems in which the state of a vehicle can be expressed. Three of the most used coordinate systems in space flight problems are the cartesian, spherical and keplerian system, and all of those systems (or formulations) are used in this research. This also means that the state should be transformed between these different coordinate sets. In Figure 3.3 it is shown how the position of a point can be described in either spherical coordinates or cartesian coordinates. 8 (a) (b) (c) (d) (e) Figure 3.1: Used reference frames (a) (Mars centred) Inertial frame, (b) (Mars centred) Rotational frame, (c) (Vehicle centred) Vertical frame (Mulder et al., 2013), (d) (Vehicle centred) Body frame (Trinidad et al., 2012) and (e) (Vehicle centred) Propulsion frame (Mooij, 1994). Figure 3.2: General rotation around the x-axis (Mooij, 2013). Figure 3.3: Position expressed either in the cartesian (x,y,z) or in the spherical system (r , δ, λ) (Noomen, 2013a). From this figure the relation can be described between the two different systems. This relation is important to allow for the translation between parameters expressed in one system to the other. More information on the transformations and coordinate system relations can be found in Appendix A. The general model is

1**described in the** Mars centred **inertial frame** in terms **of the**

cartesian state (position and velocity

84**in the x-, y-, and z-direction)** and **the** mass. This **is** done because **the**

cartesian state formulation in itself does not contain any singularities and will be used in the numerical integrator that will be compared to TSI. However, different formulations based on the general model will be used for the propagation tool using TSI as will be described in Chapter 5. The general cartesian notation is shown in Equation (3.2), where mM AV is the mass

37**of the** MAV **and the subscript I refers to the**

inertial frame. This frame is used because the target orbit is a circular orbit around Mars described in the Mars centred inertial frame. To determine the next state, the change in the state parameters over time have to be computed. This is done by taking the time derivatives of the state as described by Equation (3.3). $\begin{bmatrix} xI \\ yI \\ zI \end{bmatrix}$ r = $\begin{bmatrix} VxI \\ VyI \\ VzI \end{bmatrix}$ V = mM AV (3.2) $\dot{x}I = VxI$ $\ddot{x}I = \dot{V}xI = axI$ $\ddot{y}I = VyI$ $\ddot{y}I = \dot{V}yI = ayI$ $\dot{z}I = VzI$ $\ddot{z}I = \dot{V}zI = azI$ $\dot{m}MAV = -\frac{T}{g0Isp}$ (3.3) From this point on, the subscript I is omitted, because the state and the state derivatives are always pre- sented in the inertial frame. If variables have to be presented in any other reference frame the corresponding subscripts will be provided and explained. The accelerations

1**in the** x-, **y- and z-direction**

have three con- tributing components: gravitational acceleration, drag acceleration and thrust acceleration. In this model it is assumed that there is a homogeneous gravitational field acting on the point mass with a direction towards the centre of the inertial frame, which in this case is equal to the centre of Mars. The gravitational acceleration can therefore be directly

22**expressed in the inertial frame. The** drag is assumed **to**

work in the opposite direction

85**of the** MAV **velocity vector,** which **in** turn **is assumed to be** in **the**

positive x-direction of the body frame associated with the MAV

11**(angle of attack and side-slip angle** are assumed to **be**

zero, so the aerodynamic frame is in this research the same as the body frame). Therefore the MAV velocity, or ground velocity assuming there is no wind, is acting through the nose of the ascent ve- hicle. The drag

36**acceleration is** therefore **expressed in the** body **frame.** This effect in **the body frame**

has to be translated into an acceleration in the inertial frame, which can be done using transformation matrices. A tra- ditional transformation matrix uses an Euler angle to rotate a coordinate system in a certain reference frame to another. Using the proper Euler, or rotation, angles, the acceleration

15**in the x-, y- and z-** direction in **the body frame**

can be transformed into acceleration components acting

15**in the x-, y- and z-** direction **of the** iner- tial **frame.**

Once it is expressed in the inertial frame it can be added to the effect caused by the gravitational acceleration. It is also assumed that the nozzle of the MAV engine can be pivoted in two different directions: the y-, and z-direction of the body frame using the thrust elevation angle εT and the thrust azimuth angle ψT . This means that the thrust acceleration can better be expressed in what is called the propulsion frame (see Fig- ure 3.1e). Because the nozzle can move in two directions, or rather pivot over two different angles, the thrust acceleration can be translated to the body frame using these two thrust angles. Once the thrust acceleration is expressed in the body frame, the same transformations that are used for the drag acceleration can be used to determine the thrust acceleration components in the inertial frame. Combining this in one equation then results in the expression for the acceleration vector as shown by Equa- tion (3.4). The subscript G shows that the parameter is a function of the ground velocity, which in this case is equal to the velocity in the rotating frame of the MAV itself because no wind is assumed. T stands for a transformation and the subscript determines the rotation axis. The parameter in the brackets for each trans- formation is the rotation angle, and each vertical line with a letter below it indicates the reference frame ⎛⎛ ⎝aaxy⎞⎠⎞ = ⎛⎝aaxy,,GGrr aavv⎞⎠ + Tz (−ΩM tO + ωP) Tz (−τ) Ty (π + δ | Tz (−χG) Ty (−γG) | ⎜⎜⎝aDr ag ⎞⎠ ⎟⎟ 0 + ... az az,Grav | | R 2 | V | B 0 Tz (−ψT) Ty (−εT) | ⎡ aThrust ⎤⎦ ... | B ⎝ 0 ⎠⎦ (3.4) P 0 The rotations required to go

> 4**from the body frame to the** inertial **frame are** all **a**

function of the current state. Both thrust angles are an input value that is set by the user or the program. −ΩM tO + ωP represents the angle between the rotating frame and the inertial frame and consists of three parts: ΩM is the rotational velocity of Mars around its own axis, tO

> 93**is the difference between the** current **time and the time** at which **the**

inertial frame was set on Mars, and ωP

> 80**is the** angle **between the x-axis of the** inertial **frame and the** zero meridian at **the**

time that the inertial frame was set on Mars. In this thesis the inertial frame is set at the beginning of the simulation (so at t = 0, tO = 0) and is aligned with the rotating frame (so ωP = 0). This means that the two reference frames shown in Figures 3.1a and 3.1b are alligned at the start of the simulation. The transformation from the vertical to the rotating frame requires the latitude and longitude of the MAV, which is a function of the current state. Mooij (1994) provides equations to determine the different rotation angles. These expressions can be used to compute the latitude and longitude as well, as shown by Equation (3.5). r = x2+y2+z2 √ δ=arcsin(z) r xR = cos (ΩM tO) x + sin (ΩM tO) y yR = − sin (ΩM tO) x + cos (ΩM tO) y = atan2(yR) (3.5) τ xR It can be seen that to compute the longitude

> 2**in the rotating frame** (τ) **the position** of **the** MAV in the **rotating frame is**

required. This position is found by transforming the inertial position to the rotational position using ΩM tO as described before. Working out that transformation matrix results in the sines and cosines presented in Equation (3.5). Similarly, the flight-path angle γG and the azimuth angle χG can be described as a function of the current state. For this, some intermediate parameters have to be computed first, which are given by Equation (3.6). Here VI is the

> 32**velocity in the** inertial **frame,** VG **is the velocity in the** rotating **frame and**

Vx,V , Vy,V and Vz,V are the respective x-,

65**y- and z-** velocity **components in the** vertical **frame** as depicted **by**

Figure 3.4. Then the angles can be computed using these parameters as shown by Equation (3.7). In this case,

11**a transformation from the** inertial **frame to the** vertical **frame**

was required to get information on the velocity in the vertical frame. This transformation was written out in sines and cosines in Equation (3.6) resulting in lengthy expressions. √ VI = Vx2+Vy2+Vz2 VG √VI2 + Ω2M (x2 + y2) + 2ΩM (x y − y x) = Vx,V = (Vx + ΩM y) · (sin (δ) sin (ΩM tO)

95**sin (τ) − cos** (ΩM tO) **cos (τ) sin (δ))** + (Vy − ΩM x) · (− **cos (τ)** sin (δ) **sin**

(ΩM tO) − cos (ΩM tO)

76**sin (δ) sin (τ))** + Vz **cos (δ)**

(3.6) Vy,V = (Vx + ΩM y) · (− cos (τ) sin (ΩM tO) − cos (ΩM tO) sin (τ)) + (Vy − ΩM x) · (cos (ΩM tO) cos (τ) − sin (ΩM tO) sin (τ)) Vz,V = (Vx + ΩM y) · (cos (δ) sin (ΩM tO) sin (τ) − cos (ΩM tO) cos (τ) cos (δ)) + (Vy − ΩM x) · (− cos (τ) cos (δ) sin (ΩM tO) − cos (ΩM tO)

76**cos (δ) sin (τ))** − Vz **sin (δ)**

Figure 3.4: Spherical velocity variables in a vertical cartesian frame (Mooij, 1994). = (Vy,V) χG atan2 Vx,V γG =−arcsin(Vz,V) (3.7) VG 3.2. GRAVITY Now that the rotation angles

5**from the body frame to the inertial frame** are known, only **the** thrust angles and **the**

different accelerations have yet to be defined. This is done in the next sections. 3.2. GRAVITY The gravity acceleration is a function of the position of the MAV and the standard gravitational parameter of the planet, in this case Mars. Often, irregularities in the mass distribution of a planet also have to be taken into account (J2,0 etc.), however, because the mission time is very short and the MAV does not even complete one revolution around Mars, these effects can be neglected. Therefore, the gravitational acceleration in the inertial frame can be represented by Equation (3.8). x | | ax,Grav | | −μ r 3 \ay,Grav/ = − μ y | (3.8) az,Grav \ r3 − μ z / | r3 3.3. THRUST The thrust acceleration in the propulsion frame is simply the thrust divided by the current mass of the MAV, which can be written as shown by Equation (3.9). aThrust = T (3.9) mMAV However, this acceleration has

26**to be transformed to the body frame** using **the**

thrust angles as defined in Figure 3.1e. Both the thrust elevation as well as the thrust azimuth angle will be manually optimised to fit the trajectory and then set as a constant value during the entire ascent trajectory. They are thus not a function of the initial conditions but follow from the reference trajectory. The value for the thrust T depends on the chosen MAV design and can be different depending on the reference case. It will be set as a constant value during the ascent. These angles will not be computed, but are set as an input for the program. 3.4. ATMOSPHERIC DRAG The drag acceleration can be represented similarly to the thrust acceleration taking into account that the drag is acting

acceleration can then be written as Equation (3.10). adrag =− D (3.10) mMAV Here the drag is given by Equation (3.11), where the value for VA is in this case equal to VG because it is assumed that there is no wind. D = 1ρVA2SCD 2 (3.11) Also, S is the reference area of the MAV set by the user, ρ is the

Mach number. The Mach number in turn is a function of the speed of sound, which depends on the air temperature as can be seen in Equation (3.12). Here, Ra is the molar gas constant, γa

Martian atmosphere, which according to Ho et al. (2002) can be set to ~1.35. Ma is the molecular mass of the Martian atmosphere and depends on its composition. Williams (2015) mentions a volumetric composition of Mars which results in a mean molecular mass of 0.04334 kg/mol. This value has been used in this research to compute the Specific gas constant Ra ∗ Ṁ = VA a = √γaR∗aTa a where Ra∗ = Ma R a (3.12) The temperature (Ta) changes as a function of the altitude. This means that a temperature model of the atmosphere of Mars will have to be used to approximate these values as the ascent vehicle moves through the atmosphere. This is done through a function fit of an atmospheric model, as is described in Section 3.4.1. The drag coefficient changes with the Mach number. Therefore, this also requires a function fit to be used in the simulation program. This fit is presented in Section 3.4.2. 3.4.1. ATMOSPHERIC DENSITY AND TEMPERATURE FUNCTION FIT Several commercially available Mars atmospheric models are available that can be used to simulate the Mar- tian atmosphere for use in, for instance, trajectory calculations. One of these models is the 2005 Mars-Global Reference Atmospheric Model (GRAM) (Justus, 1990) of which the 2005 version is free to use. This model is of- ten used by the TU Delft and JPL and has also been used by some of the reference research such as Desai et al. (1998) and Trinidad et al. (2012). Using Mars-GRAM 2005, a table containing altitude, latitude and longitude dependent temperature and density data was produced. The altitude range was -0.6 to 320 km MOLA (Mars Orbiter Laser Altimeter) with a step-size of 0.01 km, the latitude and longitude ranges were centred around the example launch site (21.0 N and 74.5 E) with a 10-degree range in each direction and a step-size of 1 ◦◦ degree. The rest of the input parameters were constant and can be seen in Appendix B. The temperature and density data produced are shown in Figures 3.5 and 3.6 respectively for nine latitude and longitude combina- tions including the reference launch site itself. This is therefore a sub-set of the full data volume available in Mars-GRAM 2005. Unfortunately, data tables cannot be used when integrating using TSI, which is why both these data sets had to be fitted with differentiable functions. The temperature data cannot be fitted smoothly with one func- tion. Therefore, depending on the altitude range, a different approximation function is required. This calls for a polynomial function fitting for different sections of the temperature curve. Polynomial functions are easily differentiated and can be set-up with a large approximation precision of the curve section. The condition to be met for a proper fit came from the differences in the temperature-altitude and density- altitude curves, where the maximum difference with respect to the launch site curve was taken. The first requirement for the standard deviation of the polynomial curve fit was then to be (at least) one order lower than this maximum difference. The second requirement is that the maximum difference between the fit and the nominal launch site curve is lower than the initial maximum difference. The temperature-altitude curve was split into 5 sections as roughly visualised in Figure 3.7. The number of sections comes from both the shape of the curves and the requirement for accuracy and maximum order of the polynomial, which is set at 8 because otherwise the polynomial would get too long. This could result in longer run times and under- representation of certain parts of the curve. Also, the number of sections were to be kept at a minimum to avoid having to switch

between too many sections slowing down the computation process. More information on the fitting process is provided in Appendix C. Figure 3.5: Temperature data generated with Mars-GRAM 2005 showing different latitude and longitude combinations. Figure 3.6: Density data generated with Mars-GRAM 2005 showing different latitude and longitude combinations. Figure 3.7: Different temperature curve sections. Each section was fitted with a polynomial function of order n as represented by Equation (3.13). The last section shows a constant temperature, thus the temperature of the launch site curve was chosen to represent this final section, which is equal to 136.5 K. $y = p_0 + p_1x + \cdots + p_{n-1}x_{n-1} + p_nx_n$ (3.13) A low order is preferred, because then the fitted function will be simpler to evaluate and contain fewer terms. However the order has to be high enough to meet the accuracy requirements. To achieve this, the maximum difference between the data curves had to be higher than twice the maximum polynomial standard deviation to obtain approximately 95% accuracy in the fit. Table 3.1 shows the orders that were required and the devi- ations to the launch site temperature-altitude curve. The actual corresponding

12**parameters are** provided **in Table** 3.2. **It should be noted that**

the first few temperature data values were so different from the rest of the curve that it was assumed that this is due to uncertainty in the Mars-GRAM program and were thus treated as outliers. The provided polynomial fit is only directly a function of the altitude, but because it is fit using different latitude and longitude curves it is indirectly also a function of the longitude and latitude variation. The complete polynomial fit for the launch site curve for the temperature is shown in Figure 3.8. Table 3.1: Temperature curve fit data all with respect to the launch site curve (latitude and longitude of the launch site). Section Altitude range [km MOLA] Order Maximum polynomial standard deviation [K] Maximum polynomial difference [K] Maximum data curves difference [K] 1 -0.6 to 5.04 1 0.0312 25.8 0.177 2 5.04 to 35.53 2 0.287 3.90 0.7056 3 35.53 to 75.07 6 0.624 8.00 1.69 4 75.07 to 170.05 > 170.05 8 0.523 6.60 2.45 5 0 0.0 5.5 5.5 Table 3.2: Temperature curve fit parameters (rounded to 3 decimal points) Section p0

87**p1 p2 p3 p4 p5 p6 p7 p8 1** 194.165 **3.**

415 2 222.052 -2.130 0.006 3 -1.167 $\cdot 10^4$ 1.407 $\cdot 10^3$ -68.294 1.733 -0.0243 1.785 $\cdot 10^{-4}$ -5.388 $\cdot 10^{-7}$ 4 2.236 $\cdot 10^5$ -1.523 $\cdot 10^4$ 447.378 -7.405 0.076 -4.862 $\cdot 10^{-4}$ 1.931 $\cdot 10^{-6}$ -4.328 $\cdot 10^{-9}$ 4.1942 $\cdot 10^{-12}$ 5 136.5 The density fit was slightly more difficult because the curves are all very similar and thus result in a higher accuracy requirement for the fit. At first glance it looks like a natural logarithmic function, unfortunately an ordinary exponential did not fit the curve. This is why a more extensive exponential fit was required. The natural logarithm of the data has been plotted in Figure 3.9. With the data represented in the logarithmic domain, again a polynomial function can be fitted. The total fit then satisfies Equation (3.14). $y = \exp(p_0 + p_1x + \cdots + p_{n-1}x_{n-1} + p_nx_n)$ (3.14) The same polynomial requirements as for the temperature curve were enforced for the density curve as well. However, because the polynomial is used in an exponential, some extra requirements are needed to assure the accuracy of the fit. One requirement is that the maximum difference between the final exponential fit and the normal launch site density curve is smaller than the maximum difference between all the data curves. Also, in this case the standard deviation of the difference between the exponential fit and the normal launch site curve had to be within the range of standard deviations of the difference between the different data curves. This meant that even though an 8th-order polynomial fit could be achieved for the natural logarithmic data with the required accuracy, when converted to the exponential fit, the last two requirements were not met. Earlier it was mentioned that an order higher than 8 was not desirable. However, in this case, a single exponential fit could be achieved using a 10th-order polynomial. This fit meant that the density curve did not have to be split up at all, which makes the integration slightly easier. Therefore, it was decided that a 10th-order polynomial was acceptable in this case. The results of the fit are presented in Tables 3.3 and 3.4 and the polynomial and exponential fit curves

74**are shown in Figures 3.10 and 3.11 respectively. Figure 3.**

8: All section fits for the launch site temperature data curve Figure 3.9: Natural logarithmic plot of the density data Table 3.3: Density curve fit data (10th-order polynomial) with respect to the launch site curve (Latitude and longitude of the launch site) Parameter Value Maximum polynomial standard deviation [ln(kg/m3)] 0.0501 Maximum polynomial difference [ln(kg/m3)] 0.160 Maximum natural logarithmic data curves difference [ln(kg/m3)] 0.460 2.826·10−3 Maximum exponential difference with launch site curve [kg/m3] 3.910·10−3 Maximum data curves difference [kg/m3] 1.167·10−4 Standard deviation exponential fit difference [kg/m3] 2.106·10−4 Maximum standard deviation data curves difference [kg/m3] Table 3.4: Density curve fit parameters (rounded to 3 decimal points) p0 p1 p2 p3 p4 p5 -4.172 -0.0962 1.414 ·10−3 -9.604 ·10−5 2.273 ·10−6 -2.884 ·10−8 p6 p7 p8 p9 p10 2.146 ·10−10 -9.620 ·10−13 2.559 ·10−15 -3.724 ·10−18 2.287 ·10−21 Figure 3.10: Polynomial fit for the launch site density data curve. Figure 3.11: Exponential fit for the launch site density data curve. 3.4.2. DRAG COEFFICIENT FUNCTION FIT Similar to the temperature and density curves, the relation between Mach number and drag coefficient, as depicted in Figure 2.4, had to be modelled as a function as well. Again, it cannot be fitted using one single function, but instead had to be modelled by different functions. Fortunately, this curve is already an approx- imation and thus consists of linear elements only. It can be split up into six different sections where the first and last section are constant (CD is 0.2 and 0.3 respectively). Using a similar polynomial fit as before, but now for 1 order only, a linear fit could be made for each of the remaining four sections. The corresponding parameters are shown in Table 3.5 and the curve fit is shown in Figure 3.12. Table 3.5: Drag coefficient curve fit parameters (rounded to 3 decimal points). Section Mach range p0 p1 1 0 to 1 -2.483 ·10−16 0.2 0.0 2 0.5 to 1 0.400 3 1 to 1.3 -0.167 0.567 4 1.3 to 2.5 0.754 -0.142 5 2.5 to 4 > 0.567 -0.0667 6 4 0.3 0.0 3.5. CIRCULARISATION Figure 3.12: All section fits for the drag coefficient - Mach curve. 3 . 5 .

> 68 **C I R C U L A R I S** AT **I O N**

After the integration has been completed (defined by reaching either the desired altitude, mission duration or horizontal flight conditions), a circularisation burn will take place to estimate the required propellant mass to arrive in the desired orbit. This will include the ΔV required to change the

> 1 **flight-path angle,** change **the** in- clination **and** adjust **the**

orbital velocity. A simplified diagram of such a procedure is presented in Figure 3.13. The equations used to perform this instantaneous burn are based on the equations presented by Wakker (2010b). Figure 3.13: Circularisation into the desired orbit (Wakker, 2010b). The first step is to convert the final cartesian state obtained at the end of the integration to Kepler elements. This conversion is a standard conversion that can be found in Wakker (2010b) and requires the standard grav- itational parameter of Mars (μ_M), which is taken to be 4.2828314 km3/s2. Then using the vis-viva equation presented in Equation (3.15) the current orbital velocity of the launch trajectory can be computed. Here the subscript co stands for "current orbit"

> 36 **and a is the semi-major axis of the**

launch trajectory. $\sqrt{}$ Vco = μ_M (2 − 1) (3.15) rco aco Then the required orbital velocity is a simplified representation of the vis-viva equation because it is assumed that the required orbit is always circular, see Equation (3.16). Here d o stands for "desired orbit". 3.5. CIRCULARISATION Vdo = $\sqrt{}$ μ_M rdo (3.16) Because a change in flight-path angle might be required, the current flight-path angle has to be included into the change in velocity as well. For this, the semi-latus rectum (p) of the current orbit has to be computed first and then the cosine

> 1 **of the flight-path angle can be**

computed as shown by Equation (3.17). $p_{co} = a_{co} (1-e_{c2o})$ (p_{co}) $\cos\gamma_{co} = (\sqrt{(p_{cor})_{co}-(1-e^2)})$ (3.17) 2 r_{co} The required ΔV needed to reach the desired orbit can now be computed using Equation (3.18). $\Delta V = \sqrt{V_{d2o} + V_{c2o} - 2V_{do}V_{co} [\cos\gamma_{co} + \cos(i_{do} - i_{co})]}$ (3.18) Finally, the required propellant mass can be computed using Tsiolkovsky's equation as shown by Equation (3.19). m_{prop} $m_{MAV} = \left| 1-e^{\frac{-\Delta V}{I_{sp}g_0}} \right|$ (3.19) 4 NUMERICAL INTEGRATION METHODS Propagation refers to the process of modelling/predicting the manner in which an object (for instance) will progress (in time) from a starting point, usually given an acceleration. Such an acceleration could be the result of a constant force or another kind of perturbation. In orbital computations, numerical integration is often used to model this behaviour, because of the irregularities in the dynamic environment (many perturbations) and the absence of analytical solutions. The solution then provides an estimate of the trajectory and the state of the spacecraft s/c (Hofsteenge, 2013). There are several different types of numerical integrators that all try to predict the Spacecraft (s/c) behaviour in different ways. These types are

88described in Section 4.1. In Section 4.2 the

method is chosen that will be used in this research to compare the performance of the TSI method. Numerical integration methods do have two major drawbacks compared to analytical methods, and that is the accuracy of the produced result and the computational time required to get to that result. Numerical methods come close to the actual result but usually do not reach it. Therefore, there will always be a certain error associated with the answer. One of the most important errors is called the local (and total) truncation error, which is caused by the numerical inaccuracy of the method itself, and are usually the largest errors. A second error can be introduced by the round-off properties of the used computer, which then also depends on the number of evaluations required for each of the integration methods (more evaluations means more round-off errors). These

17round-off errors are due to the fact that computers can only accurately represent a

number up until a certain number of decimal figures. According to Milani & Nobili (1987) there are also specific errors that arise when integrating spaceflight-related problems. Instability errors can occur if the simulated system is chaotic or if the step-size is too large. Another, non-method specific and not necessarily integration related, error source is the mistakes made in the mathematical model representing the physical problem in the simulation. Such errors can occur when forces and disturbances are not (properly) taken into account in the assumptions made to create the mathe- matical model. Fortunately, many of the numerical integration methods already contain functions that approximate some of these errors and then include those approximations to come up with a better solution, or try to min- imize these errors by using multiple approximations. However, errors in the model can remain if the wrong assumptions have been made or wrong numbers have been used. The focus of this chapter will lie on the different numerical integration methods and not the different methods of determining the different errors. 4.1. INTEGRATOR TYPES A number of different numerical integration methods exist, and in this section they will be split based on how they work. In this case they have been split into single-step (Section 4.1.1) and multi-step methods (Section 4.1.2) based on Noomen (2013b). The methods can also be split based on the used step-size; either a fixed step-size that does not change during the integration, or a variable step-size that changes per step (or sometimes even during the step). Finally, the methods can also be categorised by either being explicit or implicit. An explicit method uses the information provided at the start of the integration (the current state x_i and sometimes previous states) to determine the next state x_{i+1}. Whereas an implicit method uses the same information to determine x_{i+1}, but once this first solution is obtained, it uses this next state as an 22 4.2. CHOSEN METHOD FOR COMPARISON approximation of the solution and feeds it back into the method to determine a more accurate solution. This requires iteration. The numerical approximation of the next state can be written as the current state plus the change dur- ing one time-step. This change is defined as the step-size h times the increment function Φ as shown by Equation (4.1). Here, η represents the numerical approximation. $x(t_0 + h) \approx x_0 + h\Phi = \eta(t_0 + h)$ (4.1) 4.1.1. SINGLE

81-**STEP Single-step methods** solely **use the** information at the **current** point **to** determine **the** approximation of **the next**

state. This means that the information of the previous points is neglected and usually not saved (Noomen, 2013b). Examples of the most simple explicit, fixed step-size, single-step methods are Euler, mid-point and Runge-Kutta 4th-order (RK4) (Hofsteenge, 2013). Euler uses the information at the current state to directly compute the estimate of the next state. Mid-point already incorporates an extra estimation where it first computes a point at half the step-size and then uses that information to approximate the next state, and RK4 takes four points into account and takes the weighted average of those to come up with a solution (the start- ing point, two mid-points and a final point). Many methods exists that are based on these simple methods, for instance the mid-point method based high-order extrapolation (a.k.a. DIFEX2) (explicit) method1. Many more methods, however, are based on the RK4 principle such as Runge-Kutta-Nyström (RKN), with variations such as RKN7(6)9) (implicit) (Dormand et al., 1987; Montenbruck, 1992a) and Runge-Kutta- Nyström 12th-order (RKN12) (implicit) (Montenbruck, 1992a) as well as RKF methods such as

25**Runge-Kutta- Fehlberg 4th (5th)-order** (RKF4(5)) and **Runge-Kutta-Fehlberg**

7th (8th)-order (RKF7(8)) (Fehlberg, 1968, 1969). These last two integrators are slightly different since they are still explicit, but use a variable step-size. 4.1.2. MULTI-STEP Compared to the single-step method, the multi-step method does use the information of the previous points to estimate the next state. Usually reaching as far back as the previous three points such as the Adams- Bashforth 4th-order (AB4) method (Noomen, 2013b), this being the only difference between this method and the RK4 method. Extending this method creates the explicit Adams-Bashforth 6th-order (AB6) method. An ex- ample of an implicit, fixed step-size, multi-step method is the Adams-Moulton method. It uses a polynomial to interpolate the function values to approximate the next state (Noomen, 2013b). Explicit and implicit meth- ods can also be combined to create so-called predictor-corrector methods. Here an initial guess is created by the explicit method, which is then fed into the implicit part to correct and improve the estimate. Exam- ples of predictor-corrector methods are Adams-Bashforth-Moulton 4th-order (ABM4) and Adams-Bashforth- Moulton 12th-order (ABM12) (Montenbruck, 1992a; Noomen, 2013b). All previously mentioned multi-step methods use a fixed step-size. Variable step-size methods also exist, such as Shampine-Gordon (SG) (explicit) (Berry, 2004; Meijaard, 1991) and Störmer-Cowell 14th-order (SC14) (predictor-corrector) (Berry, 2004; Ramos & Vigo-Aguiar, 2005). 4.2. CHOSEN METHOD FOR COMPARISON For this research, the focus lies on the creation of a propagator using TSI. This TSI method is completely programmed from scratch, which takes a long time and does not leave time to also completely program an- other integrator. Therefore, it is preferable that the method to which it will be compared does not have to be programmed any more or only needs a few modifications to work. For that reason, it was important to un- derstand which methods were readily available. If a method is already validated and available for use a large amount of time can be saved. Therefore, an inquiry of available methods is made in Section 4.2.1. Another important criterion is that the performance of TSI should be able to be compared to previous study cases. The easiest way of doing that is to see what methods other researchers used to compare TSI to (see Section 4.2.2). The final requirement or the chosen integrator was that it had to be the best performing integrator compared to all the other choices. This ensures that TSI is compared to the best alternative readily available. Using the information from Sections 4.2.1 and 4.2.2 a decision could be made on the final comparison

12**method as** is **described in Section 4.**

2.3. 1Program description: http://elib.zib.de/pub/elib/codelib/difex2/readme [cited 30 October 2015] 4.3. WORKINGS OF RKF 4.2.1. TU DELFT ASTRODYNAMICS TOOLBOX METHODS One of the software tools used in this thesis is TU Delft Astrodynamics Toolbox (Tudat) (see Section 6.1.1 for more information). It is a toolbox that already comes with a number of useful functions. It also contains a

number of pre-programmed validated numerical integrators. A list of the available integration methods is provided in Table 4.1. Table 4.1: Available Tudat integrators (Dirkx et al., 2016). Method Kind of method File Ready for use RK4 explicit, fixed step-size, single-step rungeKutta4Integrator.h Yes RKF4(5) explicit, variable step- size, single-step numericalIntegrator.h & rungeKuttaVariableStep- SizeIntegrator.h & rungeKuttaCoefficients.h/.cpp Yes RKF5(6) explicit, variable step- size, single-step numericalIntegrator.h & rungeKuttaVariableStep- SizeIntegrator.h & rungeKuttaCoefficients.h/.cpp No RKF7(8) explicit, variable step- size, single-step numericalIntegrator.h & rungeKuttaVariableStep- SizeIntegrator.h & rungeKuttaCoefficients.h/.cpp Yes DOPRIN8(7) explicit, variable step- size, single-step numericalIntegrator.h & rungeKuttaVariableStep- SizeIntegrator.h & rungeKuttaCoefficients.h/.cpp Yes Most of these methods are very similar, except RK4, which is a fixed step-size method, and Dormand-Prince 8th (7th)-order (DOPRIN8(7)), which is actually a Runge-Kutta method similar to RKF with an accuracy or- der 8(7) using the formulations as described by Prince & Dormand (1981) according to Weeks & Thrasher (2007). At the time of the research, the Runge-Kutta-Fehlberg 5th (6th)-order (RKF5(6)) method that was pre- programmed into Tudat was not available for use. 4.2.2. METHODS USED IN PREVIOUS RESEARCH Out of the research done on integration methods for space missions, the most relevant is the research that was done using TSI. The research performed by Scott & Martini (2008) focused on orbital trajectories and used RKF8(9) and the researched performed by Bergsma & Mooij (2016) focussed on re-entry cases and used RKF5(6). Unfortunately, RKF8(9) was not available through Tudat and RKF5(6) was out of commission at the time of this thesis research. However, it is good to notice that both researchers used higher-order RKF methods, which encourages the use of similar methods for this research; fortunately similar methods are available. 4.2.3. CHOSEN METHOD Considering the available methods and the methods used in previous TSI research it was determined that the RKF methods and DOPRIN8(7) would be tested in the early phases of the verification process. In this case RK4 would serve as a back-up. During the early verification process RKF4(5), RKF7(8) and DOPRIN8(7) were tested for number of com- putational runs, accuracy, CPU time and robustness. RKF4(5) required many computational runs, was not very accurate, had a high CPU time and sometimes required a step-size that was too small. For that reason it was also not very robust, because at higher tolerances the computation would get stuck with a step-size in the order of 10^{-5} seconds. Something similar was observed for DOPRIN8(7) for high tolerances. It was however very accurate and usually had a low number of computational runs and a low CPU time. However, RKF7(8) was the only one that was robust, accurate, had a low CPU time and a low number of computational runs. It therefore showed the best performance out of the integrators tested for this particular ascent problem, which is why it was chosen as the integration method to which TSI was later compared. 4.3. WORKINGS OF RKF The principle behind RKF is perhaps best explained by first looking at a simpler Runge-Kutta method such as RK4. Noomen (2013b) provides a simple explanation of the workings of RK4 that have been adapted here. 4.3. WORKINGS OF RKF Figure 4.1: Principle of RK4 for a single parameter Noomen (2013b). ΦRK 4 = 1 (k1 + 2k2 + 2k3 + k4) 6 (4.2) In this case, use is made of the general formulation for the numerical approximation as was shown in Equa- tion (4.1). The increment function for RK4 is then presented by Equation (4.2). In this case four points are used to approximate the next state as visualised in Figure 4.1. Runge-Kutta works with the time-derivatives at those points. The derivative at the first point is called k1, at the second point k2 etcetera. These time derivatives are used both in the increment function but also in the process of determining the derivative of the next intermediate point as shown by Equation (4.3).

> 30**k1 = f ′ (t0,** x0**) k2 = f ′ (t0 + h/2,** x0 + **hk1/2) k3 = f ′ (t0 + h/2,** x0 + **hk2/2)** (4.3) **k4 = f ′ (t0 + h,** x0 + **hk3)**

Equations (4.2) and (4.3) can also be generalized as presented by Equation (4.4). In this case bi represents the different weights for each of the points with the corresponding step-size weights ci, n is the order and ai,j are the coefficients of the previous time-derivatives. $\Phi RK = \sum$n bi ki i =1 ( ) (4.4) with ki = f ′ + ci h, x + h n$\sum$−1 t ai,jkj j =1 This general formulation can now also be used to compute the higher-order Runge-Kutta approximations given that ai,j, bi and ci are provided. These sets of numbers can be found in various literature, and different people have come up with different sets. Fehlberg was one of these people. To get a more accurate solution he decided to use two different methods. In the case of this research, RKF7(8) was used, which means that a 7th order and 8th order solution can be computed using

Equation (4.4). The 7th order solution is then used as the method solution and the difference between the 8th and the 7th order solution results in an error estimate which

> 5**can be used to** determine **the** next **step-size.** This **step-size is**

thus computed such as to minimize this error estimate. Therefore the only difference between RKF7(8) and RKF4(5) (for instance) is the coefficient set (ai,j, bi and ci's). 5 TAYLOR SERIES INTEGRATION Taylor Series Integration (TSI) is different from the previously mentioned integrators, because it only uses the information at the current point to predict the state at the next point. However, it does this using higher-order derivatives at that point. These higher-order derivatives have to be obtained

> 20**as a function of the** current **state and the first-order**

state derivatives. When the K th-order derivative is found, a Taylor Series can be set up to determine the next state as a function of the chosen step-size. This is all explained

> 86**and described in more detail in Section** 5 **.1.** In **this** thesis, TSI **is**

the main focus of the analysis, and the performance with respect to RKF7(8) is tested. To able to do this for the given model, a number of equation sets can be identified which have to be written in such a format that it can be used by TSI. These are described, and where needed derived, in Section 5.2. 5.1. GENERAL THEORY TSI has been used to solve ordinary differential equations since the early 1960s (Scott & Martini, 2008). How- ever, the first modern implementation of TSI in a space-trajectory problem was provided by Montenbruck (1992b). Scott & Martini (2008) were able to implement this TSI method into the SNAP trajectory propagator which is the implementation that is used in this thesis. The method is described in Section 5.1.1 and the used step-size method is discussed in Section 5.1.2. This step-size sometimes has to be reduced if a new section in the temperature or the drag coefficient model is reached. A root-finding method is described in Section 5.1.3 that can be used to determine at what step-size the next section is reached. The process of completing one in- tegration step is shown in Figure 5.1 and can be used as a reference throughout

> 40**this chapter. A more detailed** architecture of TSI **is provided in Chapter**

6. Figure 5.1: Simplified flow chart of one integration step for TSI. 5.1.1. WORKINGS OF TSI The formulation used in this thesis is based on the one used by Scott & Martini (2008). TSI uses the current state and its derivatives to determine the next state by computing its Taylor Series described in Equation (5.1). The state vector is represented by X with the corresponding vector for the initial conditions X0. Each of the variables can at any point be represented by xn (t). For a given step-size h and an order $K \in R$ (chosen by the user) the updated state xn (t +h) can be represented by the Taylor Series with n = 1,...,7 in this case (7 variables: three position, three velocity and one mass). The exact solution is obtained by adding $T_{n,K}$, which is the truncation error for the nth variable using K terms. 26 5.1. GENERAL THEORY $x_n(t+h) = \sum_K x^{(nk)}(t)h^k + T_{n,K}$ (5.1) k =0 k ! This particular TSI method uses recurrence relations to determine the kth-order derivatives $(x_n^{(k)})$ and only requires the current state (x_n) and its first derivatives $(x'_n = u_n)$. Each recurrence relation describes a certain operation in the state derivative, such as: multiplication, division, power, exponential, sines and cosines. Equation (5.2) shows the recurrence relations for multiplication

> 17**(w (t) = f (t) g (t**)) and division (w **(t) = f** (t) **g (t**)) respectively **(the**

other operations are described later in this chapter when they are required). For multiplications

$$13W(k) = \sum k \; F(j)\; G(k-j)\; j = 0$$

[ ] For divisions W

$$13(k) = 1\; F(k) - \sum k\; G(j)\; W(k-j)\; (5.2)\; g(t0)\; j = 1\; \text{Both with } W$$

(k) = w(kk)!(t0) F (j) =

$$17f(j)j!(t0)\; \text{and}\; G(k-j) = g((}$$

kk−−j)ji()t!0) , Here, both

$$96f(t)\; \text{and}\; g(t)\; \text{are}\; \text{place-holder}\; \textbf{functions and w}\;(t)\;\textbf{is}$$

called the auxiliary function which replaces the respective operation. Sometimes, the state derivative functions can be rather complex and have many op- erations intertwined. One can then choose to introduce extra variables to simplify the derivatives and reduce the number of operations in that particular derivative. These extra variables together with the expression that they replace then form the auxiliary equations. A random example is shown in Equation (5.3). In the first line the derivative of x4 is written out. In this equation there is an addition and a division. Within this division there is a term with an addition of two multiplications. To simplify this equation it can be chosen to write the term in the brackets as a separate variable called x8 as shown in the second line. x′4 = u4 = 2x1 + (x3x1x+4x2x3) (5.3) x8 = x3x1 + x2x3 Now that this extra variable x8 is introduced, the derivative of the corresponding auxiliary equations has to be described as well (see Equation (5.4), which in turn will use auxiliary functions to determine the kth derivative of x8. In Equation (5.4) the first line shows the new derivative of x4 using the extra variable and the second line shows the derivative of that variable. x′4 = u4 = 2x1 + xx84 x8 = u8 = x′3x1 + x3x1 + x′2x3 + x2x′3 = u3x1 + x3u1 + u2x3 + x2u3 ′ ′ (5.4) At this point it should be emphasised that using auxiliary equations and derivatives is a choice. It is not required to make TSI work. Sometimes it can be easier to introduce auxiliary equations because it makes the problem more comprehensible but this could introduce errors because all auxiliary equations also require auxiliary derivatives, which might not be easy to find. In those cases it could be more useful to just leave the state derivatives as they are and write more auxiliary functions instead. This all depends on the problem and the preference of the user. Now let un(k−1) = xnk for k = 1, . . . , K , then (uk(k−−11))! (k − 1)! , and also k ! ? Xn (k) and n = x n k x n k u k k! ?Un(k).Combining n this results in Equation (5.5). Un

$$13(k-1) = kXn(k) \Rightarrow Xn(k) = Un(k-1)\; k$$

(5.5) The Xn (k) are also known as the Taylor Series coefficients of the nth variable for the kth order derivative (or the quotient in Equation (5.1). Also, Un (k) are the recurrence relation expressions for the nth variable which consist of all the reduced auxiliary functions (W (k)) that form that derivative. 5.1. GENERAL THEORY Using the expression described in Equation (5.5) all Taylor Series coefficients can be found. Then Equa- tion (5.1)

$$12\textbf{can be used to determine the}\; \text{next}\; \textbf{state}\; \text{values}\; \textbf{at}\; \text{time}\; \textbf{t}$$

+ h for the known previous parameter values at time t . The same can then be done to determine the values at the time-step after that using the state values at t + h, etc. Going through all these different time-steps results in the final integration. 5.1.2. STEP-SIZE The step-size used for TSI can be either set as a constant value by the user or determined using a step-size con- troller, giving it the variable step-size

properties, which is done in this thesis. The chosen step-size method was based on the recommendation of both Scott & Martini (2008) and Bergsma (2015) and directly uses the chosen (or preferred) local error tolerance τ to determine the next step size. This method assures that the step-size is small enough such that all variables satisfy the error condition directly. The step-size is deter- mined using a so-called fixed-point iteration performed using Equation (5.6). The initial step-size h_1 can be chosen to be the current step-size h as an easy estimate. Then the iteration is performed over l until a certain required convergence is reached.]) hl+1 = exp K −1 ln |Xn (K −1)|+K ||Xn (K)|hl (1 [τ (5.6) This is then done for all variables and the smallest required step-size is chosen, which is then used to deter- mine the next step-size through $h_{next} = \eta h_{chosen}$, where η is the chosen step multiplication factor which has to be smaller than 1 and has been set at a value of 0.9 in this research. 5.1.3. HANDLING MODEL SECTIONS For both the temperature and drag coefficient models, different sections have been identified with different behaviours. Each of these sections was fitted with a polynomial approximation as described in Section 3.4. Because TSI only uses data available at the current point, it could be that if the step-size is allowed to be too large, the section boundary for one of these models is crossed. At that point, the step-size has to be adjusted such that it stops at the start of the next section. In this way, the next time step will start at the next model sec- tion. This also means that these sections do not necessarily have to be connected, and as a result TSI would be able to handle discontinuities in the model as well, should those occur. This section boundary is found using a root-finding method. The method used in this research is based on the method described in Bergsma (2015). First, the simulation program has to determine if the newly computed parameter (either Mach number or altitude) has indeed moved into the new section. This can be done using the section limits as shown by Equation (5.7) which is called the limit function. From now on, the equations will be written for altitude, but the same expressions hold for the Mach number. f =h−hlimit (5.7) Because this research is based on an ascent trajectory, a new section is reached if the altitude is higher than the lower limit altitude of that section. Therefore, if f > 0 a new section has been reached, and the root-finding method should be used to get the value of f back to zero and thus identify the step-size at which the root of f is located. At the start of the root-finding procedure, two points can be identified. The first point is the starting point of the MAV at the beginning of the integration step and the second point is the proposed end point of the MAV as suggested by the current step-size. If the new section has been reached it means that the point that the MAV is coming from has a limit function value that is below zero, or fFrom < 0. The mission time at that point is given by tFrom. This is the original mission time (toriginal) at the start of the integration and it is therefore the goal to find hreq (required step-size) such that toriginal +hreq results in the final limit function value, fnew = 0. The mission time at the second point (where the MAV was headed when the section was crossed) is given by tTo with the corresponding fTo > 0. This means that fnew and tnew are located somewhere between these original points. The method described by Bergsma (2015) should result in a spiralling motion that (slowly) converges to the root. This means that the root-finding method is always evaluated from the "from" point to the "to" point, irrespective of on which side of the root the next starting point is as depicted by Figure 5.2. 5.1. GENERAL THEORY Figure 5.2: Root-finding method based on Bergsma (2015). The first step is to determine the altitude derivative at the current "from" point, or f˙From. If the Taylor Series coefficients for the altitude (in this case) are available, then the derivative can directly be computed by com- bining Equations (5.1) and (5.5) resulting in Equation (5.8). Otherwise, the derivative has to be comprised of other known derivatives. f˙From ∑K kX (k)hk−1 ≈ (5.8) k =1 f˙F r om Next, the coefficient of the root-finding curve, α, has to be determined. This coefficient is effectively and is computed using Equation (5.9). 2 α = fTo fFrom − fFrom ˙ (tTo −tFrom)2 tTo −tFrom − (5.9) With this coefficient it is possible to compute the new time of the newly computed point. This point is called the new "from" point because no matter if this point is before of after the root of the limit function, that point will be used as the starting or "from" point for the next point computation. This is why this time is called tFrom,new and is computed using Equation (5.10). This equation has a plus sign in front of the square root because the next section has been reached when f > 0.) tFrom,new = tFrom + (−f˙From +√f˙F2rom +4αfFrom 2α (5.10) Now that the new time is known, the step-size

new time. With this new step-size, the Taylor Series coefficients can be used as per Equation (5.1) to determine the new altitude at that new "from" point. This means that, using Equation (5.7) the new limit function value fFrom,new = hFrom,new-hlimit. If this fFrom,new value is zero it means that the section boundary has been found and the newly computed step-size is the step-size that should be used. However, if fFrom,new is on the other side of the root compared to fFrom then the "from" and "to" points have to be updated. The conditions are shown in Equation (5.11). ⌠fTo = fFrom If the signs of fFrom and fFrom,new are opposite ⇒ ⌡ ⌡tTo = tFrom fFrom fFrom,new = ⌊tFrom tFrom,new = ⌠ (5.11) fTo fTo Remains the same = If the signs of fFrom and fFrom,new are the same ⇒ ⌡ ⌡tfTForo=mt=TofFroRme,mneawins the same ⌊tFrom = tFrom,new With this, the root-finding process can be started again at the first step (so starting at Equation (5.8)). However, the condition that fFrom,new should be equal to zero before the root-finding method is satisfied and the new step-size can be accepted is not realistic. This is why two conditions have to be met instead: ≥ the altitude and within a Mach accuracy of 1·10−6. If both these requirements are met, it is assured that the fFrom,new 0 and fFrom,new ≤ 1·10−6. This means that fFrom,new has to be within an accuracy of 1 cm for next section will be used in the next time step calculation, because the section limits are always set at the beginning of the section. 5.2. ASSOCIATED EQUATIONS In order for TSI to be implemented, the state derivatives have to be modelled as a set of functions which are a function of the state only. This can be done in different ways. In this case, three cases were tested: two cartesian cases and one spherical case. For the cartesian cases, the initial conditions first have to be trans- formed into cartesian coordinates. The cartesian equations themselves require reference frame transforma- tions, which can be written in two different ways. The first case is

62described in Section 5.2.1 and the second in Section 5.2.2.

The reason for testing the spherical case is that the initial conditions are provided in spherical coordinates and the intermediate computations can be easily interpreted and checked for errors. However, the equations are highly sensitive to singularities. The spherical equations are described in Section 5.2.3 and already include reference frame transformations. 5.2.1. CARTESIAN EQUATIONS, FIRST CASE The cartesian state and derivatives were initially described in Chapter 3 but are repeated and completed here. The cartesian state is described in Equation (5.12), where m_{MAV} is the mass

37of the MAV and the subscript I refers to the

inertial frame. $\begin{pmatrix} x_I \\ y_I \\ z_I \end{pmatrix}$ r = $\begin{pmatrix} V_{xI} \\ V_{yI} \\ V_{zI} \end{pmatrix}$ V = m_{MAV} (5.12) The corresponding state derivatives are then described by Equation (5.13).

26$\dot{x} = V_x$ $\ddot{x} = \dot{V}_x = a_x$ $\dot{y} = V_y$ $\ddot{y} = \dot{V}_y = a_y$ $\dot{z} = V_z$ $\ddot{z} = \dot{V}_z = a_z$

$\dot{m}_{MAV} = -\dfrac{T}{g_0 I_{sp}}$ (5.13) The accelerations

1in the x-, y- and z-direction

have three contributing components: gravitational accelera- tion, drag and thrust. The gravitational acceleration can be directly expressed in the inertial frame, however the drag is presented in the aerodynamic frame but can directly be expressed in the body frame since there is no wind and the angle of attack is zero. The thrust is expressed in the propulsion frame. Therefore, both the drag and thrust contributions have to be transformed to the inertial frame using transformation matrices. This then results in the expression for the acceleration vector as shown by Equation (5.14). The subscript G shows that the parameter is a function of the ground velocity. $\begin{pmatrix} x \\ y \\ z \end{pmatrix}$ $\begin{pmatrix} a_x \\ a_y \\ a_z \end{pmatrix}$ = $-\mu M 3 \begin{pmatrix} r \end{pmatrix}$ $= \begin{pmatrix} -\mu\mu_M M r r z33 \\ 0 \end{pmatrix}$ + ||||Tz|(−ΩM t0 y | + ωP) |RTz (−τ) Ty (π + δ ||VTz (−χG) Ty) | | − D mM AV 2 (− γG) ||B ||| \ 0 | + ... a z \/ 0 | T]|... Tz (−ψT) Ty (−εT) | B |P | \ m M0 AV || (5.14) /⌡0 It can be seen that this set of equations is a function of the current position and many other parameters. These parameters will all have

to be written as a function of the current state. This can be done by writing them into auxiliary equations, forming extra variables that then also require the auxiliary derivatives. This works for certain parameters, such as the gravity, because these equations are already expressed in the inertial frame. However, the transformation angles are defined in different reference frames, which means that finding the proper auxiliary derivatives can be tricky sometimes. Therefore it was decided to directly write these pa- rameters as auxiliary functions. Each of the auxiliary functions performs one simple algebraic operation and the collection of these auxiliary functions then form the complete set of recurrence relations using the re- currence relations for the simple algebraic operations. For the auxiliary equations, a similar notation will be used as shown by Scott & Martini (2008). Here the equations are denoted by xnumber and the corresponding derivatives xnumber . This notation will also be used for the current state and the corresponding state deriva- ' tives. Accordingly, Equation (5.12) can be written as Equation (5.15) and the corresponding derivatives can be written as presented by Equation (5.16). x1=x x4=ẋ=Vx x2=y x5=ẏ=Vy x7 = mM AV (5.15) x3=z x6=ż=Vz x1′ = x4 x4′ = V̇x = ax = ag,x + aD,x + aT,x x2′ = x5 x5′ = V̇y = ay = ag,y + aD,y + aT,y T x3′ = x6 x6′ = V̇z = az = ag,z + aD,z + aT,z x7′ = ṁM AV = − g0Isp (5.16) In this case the thrust and specific impulse are constant, which means that x7 is constant and any additional ' derivative will be zero. Also, neither one of the thrust angles is a function of the state, which means that aT , in the body frame, is only a function of x7 (also see Equation (5.14)). However, both ag and aD are a function of the position and velocity, where aD is also a function of the MAV mass. Only ag is rewritten using auxiliary equations as mentioned before. GRAVITATIONAL ACCELERATION For the gravitational acceleration two auxiliary equations were required since r = √x2 + y2 + z2. The resulting expressions for the gravitational acceleration are shown in Equation (5.17) with the corresponding auxiliary equations and the derivatives defined in Equation (5.18). ag ,x = −μM xr13 (x21 + x2x21+ x23)3/2 = = − x1 μM x9 ag,y = −μM xr23 (x21 + x2x22+ x23)3/2 = −μM xx29 = (5.17) ag,z = −μM xr33 (x21 + x2x23+ x23)3/2 = = −μM x3 x9 x8 = x12 + x22 + x32 x9 = x83/2 x ' 8 = 2x1x4 + 2x2x5 + 2x3x6 x′9 = 32 xx9x88 ' (5.18) TRANSFORMATION ANGLES The angles required for

all have to be written as a function of the state variables. The required angles are λ, δ, χG and γG . Here λ = τ+ΩM tO −ωP . This means that instead of first transforming to the rotating frame completely and then transforming to the inertial frame, an inertial longitude angle λ can be defined to directly transform to the inertial frame. The first two angles are the longitude and latitude and are spherical coordinates. Thus the relations for these angles can be found using the transformation from the cartesian to the spherical system. However, the angles themselves are not required directly, because they are only used in transformation matrices. These matrices are comprised of the sines and cosines of these angles, which means that a direct relation between the state variables and the sines and cosines of the position angles can be used. These relations can be derived from Figure 3.3 and are described in Equation (5.19) r = x2+y2+z2 √ sinλ √x2+y2 = y cosλ= √x2+y2 x sin δ = z cosδ= √x2+y2 r (5.19) r The corresponding auxiliary functions can then be described by Equation (5.20) using the definitions pro- vided in Equations (5.15) and (5.16). From here on, sine will be shortened to 's' and cosine will be shortened to 'c' for ease of notation. w4,1 = x12 + x22 w4,2 = w4,1?+ x32 w4,5 = sλ = x2 = w 4,4 w4,7 = sδ x3 w 4,3 w4,3 = r =

3 The latitude and longitude could be described using the position vector

> 5**from the body frame to the** vertical **frame is a** function of **the**

ground (underscore 'G') velocity

> 2**in the** rotational **frame.** Since **the position** and velocity **in the** inertial **frame**

are known (current state), the ground velocity (VG) can be written as a function of the inertial velocity (VI). For this, the velocity compo- nents have to be transformed

> 1**from the inertial frame to the** vertical **frame** (which is **the** inverse **of the**

first three transformations described in Equation (5.14)). This transformation is shown in Equation (5.21) and was described in Mooij (1994). This transformation also includes the rotational effect on the velocity due to the rotation of Mars. VV= VyV = −sλ VxV −cλsδ −sλsδ cδ Vx x 0 cλ 0 Vy − 0 × y VzV −cλcδ −sλcδ −sδ Vz ΩM z (5.21) The ground velocity can now be computed as the norm of the vertical velocity vector as shown by Equa- tion (5.22). The transformation matrices disappear because the norm of a transformation matrix is simply 1, which means that VG = VV = VR . Vx 0 x VG=‖VV‖= Vy − 0 × y ‖ Vz ΩM z ‖ Equation (5.22) can be rewritten as Equation (5.23). VG √(Vx + ΩM y)2 + (Vy − ΩM x)2 + Vz2 = The corresponding auxiliary functions are provided in Equation (5.24). w4,9 = Vx + ΩM y = x4 + ΩM x2 w4,10 = Vy − ΩM x = x5 − ΩM x1 w4,11 = w42,9 + w42,10 + x62 w4,12 = VG = w4,11 ? (5.22) (5.23) (5.24) The spherical velocity angles can now be derifed from Figure 3.4 as described by Equation (5.25). These were also provided by Mooij (1994). sinχG √ = VyV Vx2V +Vy2V cosχG √ = V x V Vx2V +Vy2V sin γG = − VzV VG cos γG = √Vx2V +Vy2V VG (5.25) Here, VxV , VyV and VzV are the velocities in the vertical frame. Expressions for these variables can be obtained by rewriting Equation (5.21). This then results in Equation (5.26). VxV = − (Vx + ΩM y) sδcλ − (Vy − ΩM x) sδsλ + Vz cδ VyV = (Vy − ΩM x) cλ − (Vx + ΩM y) sλ VzV = − (Vx + ΩM y) cδcλ − (Vy − ΩM x) cδsλ − Vz sδ The combined auxiliary functions for Equations (5.25) and (5.26) are described in Equation (5.27). w4,13 = −cλsδ = −w4,6w4,7 w4,14 = −sδsλ = −w4,7w4,5 w4,15 = −cδcλ = −w4,8w4,6 w4,16 = −cδsλ = w4,8w4,5 w4,17 = VxV = x6w4,8 + w4,9w4,13 + w4,10w4,14 w4,18 = VyV = w4,10w4,6 − w4,9w4,5 w4,19 = VzV = w4,9w4,15 − x6w4,7 + w4,10w4,16 w4,20=V?x2V +Vy2V =w42,17+w42,18 w4,21= w4,20 w4,22 = sχG = w 4,18

> 6**w4,21 w4,23** = cχG **w4,** 17 = **w4,** 21 **w4,24** = sγG = − **w4,** 19 **w4,** 12 **w4,** 25 = cγG = **w4,** 21 **w4,**

12 (5.26) (5.27) DRAG ACCELERATION The drag acceleration is determined in the body frame by dividing the drag force (D) by the mass of the MAV (x7). The drag force itself is a

> 70**function of** the **position and** velocity. **The equations**

associated with the drag function are described in Equation (5.28) except for the CD equations. The polynomial coefficients for the density equation are provided in Table 3.4 and are represented in Equation (5.28) by Pρ. D = 12ρVG2SCD ρ = ePρ10h10+Pρ9h9+Pρ8h8+Pρ7h7+Pρ6h6+Pρ5h5+Pρ4h4+Pρ3h3+Pρ2h2+Pρ1h+Pρ0 (5.28) h = r −RMOLA The numbering for the drag auxiliary functions starts with 27 because it was added later on and because it used to be an auxiliary equation. The auxiliary functions for the density can then be described by Equa- tion (5.34). w27,1 = h = w4,3 −RMOLA w27,2 = w227,1 w27,3 = w237,1 w27,4 = w247,1 w27,5 = w257,1 w27,6 = w267,1 w27,7 = w277,1 w27,8 = w287,1 w27,9 = w297,1 w27,10 = w2170,1 w27,11 = Pρ10w27,10 + Pρ9w27,9 + . . . + Pρ1w27,1 + Pρ0 w27,12 = ρ = ew27,11 (5.29) Since the

as by Figure 2.4 and it consists of several functions, it has to be split into 6 different sections. Each section has a separate CD − M relation. Before these relations can be described, three additional expressions are required which are described in Equation (5.30). M = VG (5.30) a = √γaR∗aTa a where Ra∗ = Ma R a The corresponding auxiliary functions can be described by Equation (5.31). w27,14 = a = √γaRa∗w27,13 w27,15 = M = w4,12 (5.31) w 27,14 The conditional relations shown in Equation (5.32) describe the different equations that are associated with the different sections of the drag coefficient plot. Here PCDnumber,section are the polynomial fit coefficients as provided in Table 3.5. ⌈0.2, PCD1,2M +PCD0,2, CD = ⌉ ⌊PCD1,3M + PCD0,3, PCD1,4M +PCD0,4, ⌊ PCD1,5M +PCD0,5, ⌊0.3, for

5 (5.32) for 2.5 ≤ M < 4 for M ≥ 4 In this case, the auxiliary function for CD (= w27,16) is any of the conditional relations depending on M (= w27,15). This only leaves the temperature Ta (= w27,13) , which is a function of the altitude h (= w27,1) in km MOLA. But as described in Section 3.4.1, this parameter is split into different sections as well. The equations per section for the temperature are provided in Equation (5.33). Here, PTnumber,section are the polynomial fit coefficients as provided in Table 3.2. ⌈PT 1⌉1h + PT 0,1, PT 3,2h3 + PT 2,2h2 + PT 1,2h + PT 0,2, h6 + PT 5,3h5 + PT 4,3h4 + PT 3,3h3 + . . . Ta = ⌉⌋PT 6,3 . . . + PT 2,3h2 + PT 1,3h + PT 0,3, PT 8,4h8 + PT 7,4h7 + PT 6,4h6 + PT 5,4h5 + ... + PT 4,4h4 + PT 3,4h3 + PT 2,4h2 + PT 1,4h + PT 0,4, ⌊⌋ 136.5, for − 0.6 ≤ h < 5.04 for 5.04 ≤ h < 35.53 for 35.53 ≤ h < 75.07 (5.33) for 75.07 ≤ h < 170.05 for h ≥ 170.05 For both the conditional parameters CD and Ta the required section has to be determined before the evalua- tion of the equations. The drag can now also be described as an auxiliary function as shown by Equation (5.34). w27,17 = VG2 = w42,12 w27,18 = VG2CD = w27,17w27,16 (5.34) w27,19 = D = 1 2 Sw27,18w27,12 THRUST ACCELERATION The only acceleration component still missing is the thrust acceleration. To be able to write the auxiliary functions for the thrust, Equation (5.14) first has to be rewritten such that all the transformations are gath- ered into two transformation matrices (see Equation (5.35)). The transformation matrices are described by Equations (5.36) and (5.37) for TBP and TIB respectively. x1 ⌈ x ⌈ ⌉⌉x′4 ⌈ ⌈−μM 9⌉⌈ − w27,19 ⌉ T x′5′⌋ = −μM x 2 + TIB x7 x 7 x ⌊ ⌊−μM x9 ⌊ 0 ⌊+ TBP ⌊ 0 ⌊⌋6 x3 ⌋⌊⌊⌋ ⌋ 0 ⌋⌋ x9 0 ⌈cψT cεT −sψT cψT sεT ⌉ TBP = ⌊sψT cεT cψT sψT sεT ⌋ −sεT 0 cεT ⌈cλ (−sδcχcγ + cδsγ) − sλsχcγ cλsδsχ − sλcχ cλ (−sδcχsγ − cδcγ) − sλsχsγ⌉ TIB = ⌊sλ (−sδcχcγ + cδsγ) + cλsχcγ sλsδsχ + cλcχ sλ (−sδcχsγ − cδcγ) + cλsχsγ⌋ cδcχcγ + sδsγ −cδsχ cδcχsγ − sδcγ (5.35) (5.36) (5.37) The thrust

can now be found by rewriting the last part of Equation (5.35) to Equation (5.38) using Equation (5.36). ⌈ ⌈ T ⌉ T x7 x7 cψT cεT TBP T ⌊ 0 ⌊ = (5.38) ⌊ ⌋ 0 ⌊ x7 sψT cεT ⌋ − T ⌋ ⌊x7 sεT Equation (5.38) can be expressed as a collection of auxiliary functions as well. These are described in Equa- tion (5.39). w4,26 = cosψT w4,27 = cosεT w4,28 = sinψT w4,29 = sinεT w4,30 = cψT cεT = w4,26w4,27 w4,31 = cεT sψT = w4,27w4,28 w4,32 = 1 x7 w 4,33 = T x7 =Tw4,32 w4,34= cεTcψT =w4,33w4,30 T x7 w4,37 = T x7 cεT sψT = w4,33w4,31 w4,38 = sεT = w4,33w4,29 T x7 (5.39) The thrust accelerations are now written in the body frame, which means that they can be combined with the drag acceleration in the body frame. This is done in Equation (5.40). Here, w4,35 represents the drag acceleration

x-direction in the body frame caused by both the drag and the thrust. ⌈ −x27 ⌉⌈ w4,34 ⌈ w4,34−w4,35 ⌉ w4,36 ⌋ ⌊ ⌊ 0 x7 ⌊+⌊ w4,37 ⌋ = ⌋ w4,37 ⌋ = ⌊ w4,37 ⌋ (5.40) 0 −w4,38 −w4,38 −w4,38 ALL ACCELERATIONS COMBINED At this point the gravity accelerations are known in the inertial frame and the drag and thrust accelerations

4**in the body frame.** To get **the** final **acceleration** vector **in the** inertial **frame,**

the drag and thrust accelerations have to be transformed

4**from the body frame to the** inertial **frame** using TIB resulting **in**

Equation (5.41). Now including the gravity components as well, the (lengthy) expressions for x′4, x5′ and x′6 become Equa- tions (5.42) to (5.44) respectively. ⌈ w 4,36 ⌉ TIB ⌈ w4,37 ⌉ = −w4,38 ⌈ w4,36 (cλ (−sδcχcγ + cδsγ) − sλsχcγ) + w4,37 (cλsδsχ − sλcχ) − w4,38 (cλ (−sδcχsγ − cδcγ) − sλsχsγ) ⌋ w4,36 (sλ (−sδcχcγ + cδsγ) + cλsχcγ) + w4,37 (sλsδsχ + cλcχ) − w4,38 (sλ (−sδcχsγ − cδcγ) + cλsχsγ) w4,36 (cδcχcγ + sδsγ) + w4,37 (−cδsχ) − w4,38 (cδcχsγ − sδcγ) (5.41) x′4 = −μM xx19 + w4,36 (cλ (−sδcχcγ + cδsγ) − sλsχcγ) + ... w4,37 (cλsδsχ − sλcχ) − . . . (5.42) . . . w4,38 (cλ (−sδcχsγ − cδcγ) − sλsχsγ) x′5 = −μM xx29 + w4,36 (sλ (−sδcχcγ + cδsγ) + cλsχcγ) + ... w4,37 (sλsδsχ + cλcχ) − . . . (5.43) . . . w4,38 (sλ (−sδcχsγ − cδcγ) + cλsχsγ) x6′ = −μM xx39 + w4,36 (cδcχcγ + sδsγ) + w4,37 (−cδsχ) − w4,38 (cδcχsγ − sδcγ) (5.44) These equations now have to be written as a collection of auxiliary functions

11**for the x-, y- and z-** direction in **the** inertial **frame.**

The gravitational acceleration can be written as the vector shown by Equation (5.45). ⌈ −μ x1 ⌉ M x9 ⌈ ⌉ ag = −μM x2 w4,39 = ⌈ x9 ⌉ w5,1 ⌊ ⌋ (5.45) ⌊−μM x3 ⌋ w6,1 x9 The auxiliary derivatives (u = x′) can now be defined as the collection of auxiliary functions that describe the different transformations for u4, u5 and u6. These are shown in Equations (5.46) to (5.48) respectively. w4,40 = −sδcχG = −w4,7w4,23 w4,41 cδsγG w4,8w4,24 = = w4,42 = −sλsχG = −w4,5w4,22 w4,43 = −sλcχG = − w4,5w4,23 w4,44 = −cδcγG = −w4,8w4,25 w4,45 w4,40cγG w4,40w4,25 = = w4,46 w4,42cγG w4,42w4,25 = = w4,47 = −w4,13sχG = −w4,13w4,22 w4,48 w4,40sγG w4,40w4,24 = = w4,49 w4,42sγG w4,42w4,24 = = w4,50 cλ (w4,45 + w4,41) + w4,46

6**w4,6 (w4,** 45 + **w4,** 41) + **w4,** 46 = = **w4,** 51 cλ **(w4,** 48 + **w4,** 44) + **w4,** 49 **w4,** 6 **(w4,** 48 + **w4,** 44) + **w4,** 49 = = **w4,** 52 **w4,** 39 + **w4,**

36w4,50 w4,37 (w4,47 w4,43) − w4,38w4,51 = + + u4 w4,52 = w5,2 = cλsχG = w4,6w4,22 w5,3 = sλ(w4,45 + w4,41) + w5,2cγG = w4,5 (w4,45 + w4,41) + w5,2w4,25 w5,4 = −w4,14sχG + cλcχG = −w4,14w4,22 + w4,6w4,23 w5,5 = sλ(w4,48 + w4,44) + w5,2sγG = w4,5 (w4,48 + w4,44) + w5,2w4,24 w5,6 = w5,1 + w4,36w5,3 + w4,37w5,4 − w4,38w5,5 u5 = w5,6 (5.46) (5.47) w6,2 = sδsγG = w4,7w4,24 w6,3 = cδsχG = w4,8w4,22 w6,4 = −sδcγG = −w4,7w4,25 w6,5 = −w4,44cχG + w6,2 = −w4,44w4,23 + w6,2 w6,6 = w4,41cχG + w6,4 = w4,41w4,23 + w6,4 w6,7 = w6,1 + w4,36w6,5 − w4,37w6,3 − w4,38w6,6 u6 = w6,7 (5.48) Also, because auxiliary equations were used for the gravitational acceleration. The corresponding auxiliary derivatives can be written as a collection of auxiliary functions as shown in Equation (5.49). w8,1 = x1x4 w8,2 = x2x5 w8,3 = x3x6 u8 = 2(w8,1 + w8,2 + w8,3) w9,0 = x9u8 w9,1 = w9,0 x8 u9 = 1.5w9,1 Then the equations in Equation (5.50) show the complete set of auxiliary derivatives. u1 = x4 u4 = w4,52 u2 = x5 u5 = w5,6 u3 = x6 u6 = w6,7 u7 = − T g0Isp u8 = 2(w8,1 + w8,2 + w8,3) u9 = 1.5w9,1 (5.49) (5.50) 5.2.2. CARTESIAN EQUATIONS, SECOND CASE The second case (based on a method described by (Bergsma, 2015)) is similar to the first case in terms of set-up of the dynamic equations, the gravitational acceleration, and the drag and thrust accelerations in the body frame. The only difference lies in the formulation of TIB. Instead of using Euler angles and rotation transformation matrices, TIB can be set-up as one single matrix performing a direct transformation. This matrix is described in Equation (5.51), where C stands for transformation matrix coefficient. ⌈C1 TIB =

42⌊**C2 C3 C4 C5 C6 C7**⌋ **C8** (5.51) **C9**

As was seen in Section 5.2.1 the rotation from the vertical to the inertial frame is purely a function

> 1**of the** radius **in the inertial frame (r** or rI) Also, **the**

> 89**transformation from the** body **frame to the** vertical **frame** was purely **a** function **of the**

ground velocity in the vertical frame (VV or VGV). Therefore, if this transformation is to be done in one transformation, the ground velocity in the inertial frame is required (VGI), because it would transform directly to the inertial frame. This means that the coefficients in Equation (5.51) will have to be a function of rI and VGI, where VGI is given by Equation (5.52). $\begin{pmatrix} Vx \\ Vy \\ Vz \end{pmatrix}$ $\begin{bmatrix} x \\ y \\ z \end{bmatrix}$ $\begin{pmatrix} Vx + \Omega M\ y \\ Vy - \Omega Mx \\ Vz \end{pmatrix}$ 0 VGI= − 0 × = (5.52) ΩM z BODY FRAME UNIT VECTORS So when looking from the inertial frame, there are two state vectors: position and ground velocity. The state changes part due to the accelerations in the body frame aB, which is a vector comprised of the drag and thrust accelerations. This vector has the same length and direction

> 40**in the body frame and the inertial frame**

when looking from the inertial frame. Also, any vector is simply a multiplication of the corresponding unit vector. The same holds for the velocity, which means that because the velocity

> 1**in the body frame is defined** through **the**

x-axis of that frame, the unit vector of the ground velocity can be described as the unit vector in the x-direction î as shown by Equation (5.53). This unit vector is also shown in Figure 5.3. î = V G I ‖VGI ‖ (5.53) Figure 5.3: Definition of the body frame unit vectors based on the radius and ground velocity expressed in the inertial frame. Adapted from a figure by Mooij (1994). In Figure 5.3 it can also be seen that the z-axis unit vector kˆ is defined in the same plane as î, VGI and rI perpendicular to î. As a matter of fact, if the flight-path angle is zero degrees, kˆ

> 2**points in the same direction as the** radius **vector.** In that case, **the** y **-axis**

unit vector ĵ completes the unit frame. Since ĵ is perpendicular to the î-kˆ plane, it is also perpendicular to VGI and rI. Now because the radius vector is pointing through the centre of the body away from the centre of the inertial frame, the Right-hand-rule (RHR) gives Equation (5.54). ĵ = rI × VGI ‖rI × VGI ‖ (5.54) Now kˆ can be computed by applying the RHR to î and ĵ, which results in Equation (5.55). kˆ=î×ĵ (5.55) This leaves us with the unit

> 75**vectors in the body frame** defined as **a**

function of the state

> 75**in the inertial frame. THE** TRANSFORMATION **MATRIX**

COEFFICIENTS Now, if the acceleration in the x-direction in the body frame is multiplied with î this axB is redefined in the inertial frame by providing a contribution in the x-, y- and z-direction. A similar multiplication can be done for ayB and ĵ, and azB and kˆ. In matrix form this can be represented by Equation (5.56). [î ĵ kˆ] · $\begin{pmatrix} axB \\ ayB \\ azB \end{pmatrix}$ (5.56) Comparing Equation (5.56) to the expression for the

transformation matrix provided by Equation (5.51) it can be seen that the coefficients can now be expressed as Equation (5.57).

$$42 \begin{pmatrix} \mathbf{C1} \\ \mathbf{C2} \end{pmatrix} = \hat{\imath} \begin{pmatrix} \mathbf{C3} \\ \mathbf{C4} \\ \mathbf{C5} \end{pmatrix} = \hat{\jmath} \begin{pmatrix} \mathbf{C6} \\ \mathbf{C7} \\ \mathbf{C8} \end{pmatrix} = \hat{k} \ \mathbf{C9}$$

Given these definitions, the corresponding auxiliary functions can now be defined. AUXILIARY FUNCTIONS FOR THE SECOND CASE With the transformation matrix now defined, Equation (5.14) can be written as Equation (5.58). x ⎞ ⎛ ax ⎞ ⎞ −μM r 3 ⎞ ⎞ − D ⎞ ⎞ T ⎞ ⎞ ⎞ ⎛ ay ⎞ = −μM y ⎞ mM AV | 0 mM AV az ⎞ | + [î kˆ] · ĵ ⎞ | ⎞ ⎞ / + | ⎝ 0 −μM rz3 ⎞ r 3 ⎠ ⎠ 0 . . . . . . ‖BTz (−ψT) Ty (−εT) |P ⎝ ⎞ ⎞ 0 (5.57) (5.58) In this case, all the auxiliary functions involving the rotation angles and the transformations

6(**w4,5 − w4,** 8, **w4,13 − w4,** 25, **w4,** 40 − **w4,**

52, w5,2 − w5,6 and w6,2 − w6,7) can be discarded and instead new auxiliary functions have to be defined for the new transformation matrix. Please note that the discarded numbers will be reused but now with different function definitions for the second case! The first auxiliary functions can found by rewriting Equation (5.53) as a function of the non-discarded auxiliary functions described in Section 5.2.1 and are shown in Equation (5.59). w4,13 = C1 = Vx + ΩM y = w4,9 VG w4,12 w4,14 = C2 = Vy − ΩM x = w4,10 VG w4,12 w4,15 = C3 = Vz x6 = (5.59) VG w4,12 In case of the auxiliary functions for ĵ first the cross product has to be computed and the norm of that cross product. This is done in Equation (5.60). w4,16 = x6x2 − w4,10x3 w4,17 = w4,9x3 − x6x1 w4,18 = w4,10x1 − w4,9x2 w4,19 = w42,16 + w42,17 + w42

6,**18 w4,** 20 **w4,19 = ? w4,** 21 = C4 = **w4,**

16 w 4,20 w4,22 = C5 = w4,17 w 4,20 (5.60) w4,23 = C6 = w4,18 w 4,20 These auxiliary functions can now be used to describe the kˆ functions as shown by Equation (5.61). In this case, the last auxiliary functions was named 40 because 26 is already taken up by an auxiliary function for the thrust acceleration. w4,24 = C7 = C2C6 −C3C5 = w4,14w4,23 − w4,15w4,22 w4,25 = C8 = C3C4 −C1C6 = w4,15w4,21 − w4,13w4,23 (5.61) w4,40 = C9 = C1C5 −C2C4 = w4,13w4,22 − w4,14w4,21 Then combining everything results in the expression presented in Equation (5.62) for u4, u5 and u6 (u1, u2, u3, u7, u8 and u9 are the same as presented in Equation (5.50)). u4 = w4,39 + w4,36w4,13 + w4,37w4,21 − w4,38w4,24 u5 = w5,1 + w4,36w4,14 + w4,37w4,22 − w4,38w4,25 (5.62) u6 = w6,1 + w4,36w4,15 + w4,37w4,23 − w4,38w4,40 5.2.3. SPHERICAL EQUATIONS For the spherical case, the initial conditions are readily provided. The position comes directly from the chosen launch site and the velocity comes from the initial MAV conditions. These are all defined in the rotating frame as depicted in Figure 5.4 and Equation (5.63). Figure 5.4: Definition of the spherical coordinates (Mooij, 1994). ⎛ r ⎞ R = ⎛ δ ⎞ ⎛ VG ⎞ V = ⎝ γG ⎠ mM AV (5.63) τ χG The different variables are now defined as shown by Equation (5.64) and the derivatives by Equation (5.65). The numbers that were given to the variables are a result of the early derivations. x16 = r = h +RMOLA x12 = δ x11 = τ x1′6 = ṙ x12 = δ˙ ′ x11 = τ˙ ′ x15 = V˙G ′ x14 = γ˙G ′ x13 = χ˙G ′ x15 = VG x14 = γG x7 = mM AV x13 = χG x7′ = ṁM AV = − T g0Isp (5.64) (5.65) The derivatives given in Equation (5.65) are already provided by Mooij (1994) in the rotating frame and include the rotational effects. A simplified form can be seen in Equations (5.66) and (5.67). Because of the availability of this short notation it was decided to perform the TSI for the spherical case in the rotating frame. At the end of the integration the variables can then be transformed to the inertial frame for comparison. x′16 = ṙ = VG sγG x12 = δ˙ = VG c χG c γG ′ r x11 = τ˙ = VGsχGcγG ′ (5.66) r cδ x15 = V˙G = ΩM2 r cδ (sγG cδ − cγG sδcχG) + ′ T cψT cεT − D − μM m m r 2 sγG x14 = γ˙G = 2ΩM cδsχG + VG cγG + Ω2MVGrcδ (cδcγG + sγG sδcχG) + TVGsεmT VμGMr2 cγG ′ r − (5.67) x13 = χ˙ G = 2ΩM sδ − cδscγγGGcχG) + ′ (VG sδ r cγG cδ sχG + ΩM2 r cδsδsχG T sψT cεT + VG cγG VG mcγG The last term in the first two expressions of Equation (5.67) correspond to the acceleration contribution of the gravity. The first expression has a drag component and all three have a thrust component similar to the cartesian cases. The rest of the terms come from the rotational effects. In this case it was decided to define the drag as an auxiliary equation x27, which will be discussed in more detail later in this section. AUXILIARY FUNCTIONS FOR THE SPHERICAL CASE Before the auxiliary functions for Equations (5.66) and (5.67) can be written, some

standard auxiliary func- tions have to be defined first. These include δ, γG and χG. The numbering is based on early derivations and do not correspond to the definitions provided for the cartesian cases! The angle auxiliary functions are defined in Equation (5.68). w4,4 = sδ = sx12 w4,5 = cχG = cx13 w4,6 = cδ = cx12 w4,7 = sγG = sx14 w4,8 = cγG = cx14 w4,9 = sχG = sx13 (5.68) Some expressions for the thrust are also required. These are similar to the equations defined in Section 5.2.1 but are repeated in Equation (5.69) for convenience. w4,26 = cos ψT w4,27 = cos εT w4,28 = sin ψT w4,29 = sin εT w4,32 = 1 (5.69) x7 Now the auxiliary functions can be written for x′11 = u11 till x1′6 = u16 and are shown in Equations (5.70) to (5.75) respectively. w11,0 = VGcγG = x15w4,8 w11,2 w11,1sχG w11,1w4,9 = = w11,1 = w11,0 w11,0 = w11,2 w11,2 = u11 = w11,3 (5.70) r = x16 w11,3 cδ w4,6 w12,1 w11,1cχG w11,1w4,5 = = u12 w12,1 = (5.71) w13,0 = sψT cεT = w4,28w4,27 w13,1 = sγGcχG = w4,7w4,5 w13,2 = ΩM2rcδ = ΩM2x16w4,6 w13,3 = w13,2sδ = w13,2w4,4 w13,4 Tw13,0 = = x7 Tw13,0w4,32 w13,5 = w13,3sχG + w13,4 w13,3w4,9 + w13,4 = w13,6 w13,5 = w = 13,5 VG x15 w13,7 = −2ΩM cδw13,1 + w13,6 = −2ΩM w4,6w13,1 + w13,6 u13 = w13,9 w13,8 = w13,7 w13,7 = cγG w4,8 w13,9 = (2ΩM + w11,3) sδ + w13,8 = (2ΩM + w11,3 4,4 + w13,8) w (5.72) w14,0 = T sεT x7 = Tw4,29w4,32 w14,1 cδcγG w13,1sδ = + = w4,6w4,8 w13,1w4,4 + w14,2 = −μM cγG = −μM w4,8 w14,3 = r 2 = x126 w14,4 = w14,2 w14,3 w14,5 = w14,4 + w13,2w14,1 + w14,0 w14,6 w14,5 = w = 14,5 VG x15 w14,7 = 2ΩM cδsχG + w11,1 + w14,6 = 2ΩM w4,6w4,9 + w11,1 + w14,6 u14 = w14,7 w15,0 = T cψT cεT −D = Tw4,26w4,27 −x27 w15,4 = cγGcχG = w4,8w4,5 w15,1 = w15,0 = w15,0 x7 w15,5 sγGcδ − w15,4sδ w4,7w4,6 w15,4w4,4 = = − mM AV w15,2 = −μM sγG = −μM w4,7 w15,6 = w13,2w15,5 + w15,1 + w15,3 w15,3 = w15,2 w14,3 u15 = w15,6 (5.73) (5.74) w16,1 = VGsγG = x15w4,7 u16 = w16,1 (5.75) DRAG ACCELERATION AS AUXILIARY VARIABLE The drag acceleration for the spherical case is similar to the cartesian case, except this time it was chosen to use auxiliary variables to describe the drag. This means that auxiliary equations have to be defined as is shown in Equation (5.76). Also, this means that auxiliary derivatives are required for each of these equations. Equation (5.77) shows the different derivatives corresponding to the equations. x27 = D = 1ρVG2SCD = 12 2 Sx28x125x29 x28 = ρ = ex30 x29 = CD (5.76) x30 = Pρ10x3101 + Pρ9x931 + Pρ8x381 + Pρ7x371 + Pρ6x361 + Pρ5x351 + Pρ4x431 + Pρ3x331 + Pρ2x321 + Pρ1x31 + Pρ0 x31 = h = r − RMOL A = x16 − RMOL A x27 ′ = 1 ′ 2Sx15(x15(x29x′28+x28x′29)+2x28x29x1′5) x28 x30x28 = ′ x3′0 = x3′1 (10Pρ10x391 + 9Pρ9x381 + 8Pρ8x371 + 7Pρ7x361 + 6Pρ6x351 + . . . (5.77) . . . + 5Pρ5x341 + 4Pρ4x331 + 3Pρ3x321 + 2Pρ2x31 + Pρ1) x3′1 = x1′6 The auxiliary derivatives described in Equation (5.77) can now be used to define the auxiliary functions. These are described in Equations (5.78) to (5.80) for u27, u28 and u30 respectively. In this case u31 = u16. w27,1 = CDρ˙ = x29u28 w27,2 = ρĊD = x28u29 w27,3 = ρCD = x28x29 w27,4 = VG (w27,1 + w27,2) = x15 (w27,1 + w27,2) w27,5 = w27,3V˙G = w27,3u15 u27 = Sw27,6 1 (5.78) w27,6 = VG (w27,4 + w27,5) = x15 (w27,4 + w27,5) 2 w28,1 = u30ρ = u30x28 u28 = w28,1 (5.79) w30,1 = h9 = x931 w30,2 = h8 = x831 w30,3 = h7 = x731 w30,4 = h6 = x631 w30,5 = h5 = x351 w30,6 = h4 = x341 w30,7 = h3 = x331 w30,8 = h2 = x321) w30,9 = u31 (10P ρ10w30,1 + · · · + 3Pρ3w30,8 + 2Pρ2x31 + Pρ1 u30 = w30,9 (5.80) Again, three additional auxiliary equations are required for the CD − M relation (see Figure 2.4) and are pre- sented in Equation (5.81). x32 = M = VG = x15 a x33 x33 = a = √γaR∗aTa = √γaRa∗x34 where R∗a = Ma R a (5.81) x34 = Ta In this case the conditional relations shown in Equation (5.82) are not used directly in the recurrence relations, but only serve as initial conditions. Again, the PCDnumber,section are the polynomial fit coefficients as provided in Table 3.5.
⌠ x29,1 = 0.2, x29,2 = PCD1,2x32 +PCD0,2, x29 = CD = ⎱ ⎰ x29,3 = PCD1,3x32 +PCD0,3, x29,4 = PCD1,4x32 +PCD0,4, ⎪ x29,5 = PCD1,5x32 +PCD0,5, ⎱ x29,6 = 0.3, for 0 ≤ x32 < 0.5 for 0.5 ≤ x32 < 1 for 1 ≤ x32 < 1.3 for 1.3 ≤ x32 < 2.5 (5.82) for 2.5 ≤ x32 < 4 for x32 ≥ 4 The corresponding derivatives that will have to be used for the recurrence relations are presented in Equa- tion (5.83). ⌠x′29,1 = 0, ⎪ ⎰ xx′2′299,,34 == PPCCDD11,,34xx′3′322,, x29,2 = PCD1,2x′32, ′ x2′9 = ⎱ ⎰ x′29,6 x29,5 = PCD1,5x′32, ′ = 0, for 0 ≤ x32 < 0.5 for 0.5 ≤ x32 < 1 for 1 ≤ x32 < 1.3 for 1.3 ≤ x32 < 2.5 for 2.5 ≤ x32 < 4 for x32 ≥ 4 (5.83) The Mach derivative and corresponding speed of sound derivative are then described in Equation (5.84). x′32 = x33x15−x323x15x33 ′ ′ x33 γ2axR33∗a x′34 (5.84) ′ = For the drag coefficient, there are no auxiliary functions required. That is why x in Equation (5.83) can simply ′ be replaced by u. The Mach number and the speed of sound do require auxiliary functions and are described in Equations (5.85) and (5.86) respectively. w32,1 = aV˙G = x33u15 w32,2 = VG à = x15u33 w 32,3 = a2 = x323 w32,4 = 32,1 − w32,2 w32,3 u32 = w32,4 (5.85) w33,1 = Ṫa a = u34 x33 u33 = w33,1 (5.86) The last relations that still need to be described are the ones for the temperature Ta. Similar to CD the cor- responding auxiliary equations, shown by Equation (5.87) are only used for the initial conditions. Again, PTnumber,section are the polynomial fit coefficients as provided in Table 3.2. ⌠ x34,1 = PT 1,1x31 + PT 0,1, for − 0.6 ≤ x31 < 5.04 x34,2 = PT 3,2x331 + PT 2,2x321 + PT 1,2x31 + PT 0,2, for 5.04 ≤ x31 < 35.53 x34 = Ta = ⎱ ⎰ x34,3 = PT 6,3x361 + PT 5,3x351 + PT 4,3x341 + PT 3,3x331 + ... + PT 2,3x321 + PT 1,3x31 + PT 0,3, for

$35.53 \leq x31 < 75.07$ (5.87) $x34,4 = PT\ 8,4x831 + PT\ 7,4x371 + PT\ 6,4x631 + PT\ 5,4x531$ ⎵ x +PT 4,4x341 + PT 3,4x331 + PT 2,4x321 + PT 1,4x31 + PT 0,4, for $75.07 \leq x31 < 170.05$ $34,5 = 136.5$, for $x31$ $170.05 \geq$ The corresponding derivatives that are used for the recurrence relations are presented in Equation (5.88). ⌈x3′4,1=PT1,1x3′1, x34,2=(3PT3,2x321+2PT2,2x31+PT1,2)x′31, ′ ′ x′34 = ⎱⎰ x34,3 = (6PT6,3x351+5PT5,3x341+4PT4,3x331+... x34,4= (8PT8,4x731+7PT7,4x361+6PT6,4x351+5PT5,4x431+... ′ ...+3PT3,3x231+2PT2,3x31+PT1,3)x′31, ⎪⎣x3′4,5 0, = ...+4PT4,4x331+3PT3,4x321+2PT2,4x31+PT1,4)x3′1, for $-0.6 \leq x31 < 5.04$ for $5.04 \leq x31$ < 35.53 for $35.53 \leq x31 < 75.07$ (5.88) for $75.07 \leq x31 < 170.05$ for $x31 \geq 170.05$ All the power relations in Equation (5.88) were already described in auxiliary functions in Equation (5.80). Therefore, depending on the section, the equations presented in Equation (5.88) can be rewritten by simply replacing x′ by u and the auxiliary functions. This is shown in Equation (5.89). ⌈u34,1=PT1,1u31, u34,2= (3PT3,2w30,2+2PT2,2x31)u31+PT1,2u31, u34 = ⎱⎰ u34,3 = (6PT6,3w30,5+5PT5,3w30,4+4PT4,3w30,3+... ...+3PT3,3w30,2+2PT2,3x31)u31+PT1,3u31, u34,4 = (8PT8,4w30,7+7PT7,4w30,6+6PT6,4w30,5+5PT5,4w30,4+... ...+4PT4,4w30,3+3PT3,4w30,2+2PT2,4x31)u31+PT1,4u31, ⎪⎣u34,5 = 0, for $-0.6 \leq x31 < 5.04$ for $5.04 \leq$ $x31 < 35.53$ for $35.53 \leq x31 < 75.07$ (5.89) for $75.07 \leq x31 < 170.05$ for $x31 \geq 170.05$ Now all the information that is required to compute the recurrence relation for the drag is available. Please note that there is a certain order in which these equations have to be evaluated, since they depend on each other. Table 5.1 shows this order. Table 5.1: Order of auxiliary equations and derivatives computations Auxiliary equations Auxiliary derivatives Order xn Order xn Order un Order un 0 x7, x11, x12, x13, x14, x15, x16 4 x32 7 u7, u11, u12, u16 11 u32 1 x31 5 x29 8 u31 12 u29 2 x30, x34 6 x27 9 u30, u34 13 u27 3 x33, x28 10 u28, u33 14 u13, u14, u15 5.2.4. RECURRENCE RELATIONS Now that all the auxiliary functions are known, they can be used to determine the required recurrence rela- tions. To write these concisely and following the same convention as used by Scott & Martini (2008), a number of extra parameters will have to be introduced. The Taylor series coefficients can be written as described by Equation (5.90). Where

> 92**n is the** variable **number and k is the** order **of the**

derivative. $xn(k)\ k!\ ?\ Xn\ (k)$ where $k \geq 1$ (5.90) A similar expression is defined for un and wn as well as the place-holder functions fn and gn which are all shown in Equation (5.91). $Un\ (k)\ ?\ ukn(k!)\ Wn\ (k)\ ?\ wkn(k!)$ $Fn\ (k)\ ?\ fkn(k!)\ Gn\ (k)\ ?\ gk(nk!)$ (5.91) Then remembering that $un = x'n$ and using Equation (5.90) a relation can be described between Xn and Un as shown in Equation (5.92) (Scott & Martini, 2008). un

> $4(k-1) = xn\ (k)\ Un\ (k-1) = kXn\ (k) \Rightarrow \Rightarrow un\ (k-1)\ x(nk)\ (k-1)!\ (k-1)! = \Rightarrow Xn\ (k) =$ $Un\ (k-1)$ (5.92) k

Using these definitions, the auxiliary functions can be written into reduced auxiliary functions W (k) for the three cases. The equations stay the same, but the lower-case letters are replaced by their upper-case counter- parts. A complete list of the reduced auxiliary functions (starting at k 2, or the second derivative u) for the = ′ three cases is presented in Appendix D. As was mentioned before, all the auxiliary functions have been de- fined such that they incorporate one of the following operations: multiplication, division, power, exponential or trigonometric. This has been done because recurrence relations exist for these simple operations as pro- vided by Jorba & Zou (2005). These recurrence relations are written using the definitions from Equation (5.91) and are described in Equations (5.93) to (5.99). for $fn \pm gn \Rightarrow Wn,\pm\ (k) =$ $Fn\ (k)\pm Gn\ (k)$ for $fngn \Rightarrow Wn,mult\ (k) = \sum k\ Fn\ (j)Gn\ (k-j)$ j=0 for $gfnn \Rightarrow Wn,div\ (k) = Gn\ (0)\ Fn\ (k)-$ $j\sum=k1Gn\ (j)Wn,div\ (k\ 1\ [\ ]-j\ )$ for $fn\alpha \Rightarrow Wn,pow$

> $91(k) = 1\ k-1\ \sum\ [\ kFn\ (0)\ j=0\ k\alpha-\ j\ (\alpha\ +1)]\ Fn\ (k$

$-j)Wn,pow\ (j)$ for $efn \Rightarrow Wn,exp$

> $3(k) = 1\ k\sum-\ 1\ (k-j)\ Wn,exp\ (j)\ Fn\ (k-j)\ k\ j=$

0 for cos fn ⇒ Wn,cos

$$3(k) = -1 \sum k \, jW_{n,sin} (k-j) \, F_n (j) \, k \, j=$$

1 for sin fn ⇒ Wn,sin

$$3(k) = 1 \sum k \, jW_{n,cos} (k-j) \, F_n (j) \, k \, j=$$

1 (5.93) (5.94) (5.95) (5.96) (5.97) (5.98) (5.99) Checking the presented equations it can be seen that Equations (5.98) and (5.99) are interdependent. This means that whenever a cosine recurrence relation has to be evaluated, the same recurrence relation for sine (with at least order k − 1) has to be evaluated at the same time (and vice versa). All these equations have been derived by Jorba & Zou (2005) using the general Leibniz rule for the kth derivative of a multiplication as portrayed in Equation (5.100). (fngn)(k) ∑k (k) fn(j)gn(k−j) = j (5.100) j =0 Combining Equation (5.100) and the definition of the reduced derivative for the auxiliary function (see Equa- tion (5.91)) results in Equation (5.101). This equation, when rewritten to include the reduced derivatives for fn and gn, results in Equation (5.94). Wn,mult (k) = 1 ∑k (k) fn(j)gn(k−j) j (5.101) k! j=0 With all the recurrence relations now known, and using the definition of Equation (5.92) the updated state can be described using the Taylor series expansion as described in Equation (5.102). xn (t + h) = ∑K x(nk) (t) hk + Tn,K ∑K Xn (k) hk + Tn,K = with n = 1, . . . , 7 (5.102) k =0 k ! k =0 Here K is the order of the series to which it has to be evaluated and Tn,K is the truncation error. Please note that all the Taylor Series coefficients computed for the auxiliary equations are only needed to determine the Taylor Series coefficients of the state variables (which is why they are auxiliary). 6 SIMULATION TOOL To perform the analysis associated with this thesis, a simulation program has been written. The requirements for the software were that it had to use existing software as much as possible to be able to focus on the thesis problem itself, and the existing software had to be readily available. The tool is comprised of both existing (Section 6.1) and newly developed software (Section 6.2). It is written in C++ and is based on the Tudat struc- ture. The purpose of the software is to simulate the trajectory of the MAV using RKF and TSI. This tool is written such that the performance of both integrators can be compared. The final workings of the complete tool is described in Section 6.3. 6.1. EXISTING SOFTWARE The use of existing software can greatly improve the performance of the final tool and save time as well. An- other important reason to use existing software is that this will make it easier for other people to use and incorporate into their own software as well. The existing software used for this thesis is software that is cur- rently being used by the space department of the TU Delft (Tudat and (in case of Mars-GRAM) by the mission design section at JPL. 6.1.1. TUDAT The TU Delft Astrodynamics Toolbox (Tudat) is, as the name suggests, a toolbox that can be used to solve numerous astrodynamic problems (Dirkx et al., 2016). It is being developed by students and staff of the

83Delft University of Technology. Specifically by the section Astrodynamics and Space missions of the

Aerospace En- gineering faculty. It is programmed in C++ and consists of a number of libraries. These libraries can be called upon by the user to invoke different Tudat functionalities such as standard reference frame transformations or often-used integrators. The available software is completely validated and comes with its own tests to make sure that everything is working properly. It itself uses two external libraries: Eigen and Boost. Both libraries will be discussed in Sections 6.1.2 and 6.1.3 respectively. Figure 6.1 shows Tudat with the different libraries that are used within the Tudat Bundle including the core functions (Tudat Core). In this thesis, the Tudat libraries are used for all standard mathematical and astrodynamic operations as well as constant mathematical parameters. These include (but are not limited to): linear algebra, reference frame transformations for RKF, the use of π, coordinate system transformations, matrix and vector definitions and RKF integrators. Figure 6.1: Tudat structure. 46 6.2. DEVELOPED SOFTWARE 6.1.2. EIGEN Eigen is an external C++ library that was written to perform linear algebra computations 1. The software is free and easy to use, which is why it is widely used by the C++

community and thus also within Tudat 2. Another advantage is that because it does not use any source files but only header files, it does not need to be compiled before using it. The Eigen libraries contain a number of standardized matrices and vectors, each with its own characteristics. An example of an often-used vector is Vector3d (or Eigen::Vector3d), which can, for instance, be used to store the cartesian position of a satellite. Here, the 3 shows that it can store 3 values/parameters and the d shows that these are of the type double. It is mentioned on the Tudat wiki2 that these Eigen vectors and matrices should only be used if required for linear algebra computations. For ordinary storage, the C++ arrays, vectors and matrices should be used to save both storage and computation time. 6.1.3. BOOST Boost is a slightly more complicated set of C++ libraries. Compared to the Eigen library, Boost does first have to be compiled before being able to use all of its functionalities. Fortunately, this compiling is performed by Tudat automatically when setting it up for the first time. Boost is described as an addition to the standard C++ libraries2, thus adding more functionalities 3. Within Tudat, Boost is used to pass free and class functions as an argument to another object and also for dynamic allocation using so-called pointers. Four libraries that are often-used within Tudat are boost::function, boost::bind, boost::shared_ptr and boost::make_shared. The first two libraries are used to pass functions (a function is pointed to by function and called by bind) and the last two are used in case of dynamic allocation (shared_ptr is the pointer and make_shared is the object creator that returns a shared pointer to the created object). 6.1.4. MARS-GRAM Mars-GRAM is a high-fidelity atmospheric model developed by NASA to simulate the global atmospheric conditions on Mars 4 5. The model is based on

41**NASA Ames' Mars General Circulation Model** (for altitudes between **0-80 km) and Mars Thermospheric General Circulation model** (for altitudes above **80 km).**

It can provide density, temperature and pressure data (among other data) with respect to the current altitude, lat- itude and longitude on Mars. Seasonal variations are

2**taken into account in the model** as well, which **is**

why different calender dates will result in different atmospheric compositions. The tool can be used within a simulation tool or as a separate executable. Unfortunately, because it is so detailed (to a level not needed for this thesis), each computation requires a large amount of CPU time. This is also one of the reasons why it was decided to use the stand-alone Mars- GRAM executable to generate a detailed table with atmospheric data as a function of altitude, latitude and longitude that is used to generate a simpler model. Even generating this table required a fair amount of CPU time. The starting altitude was set at -0.6 km MOLA and advanced with a step-size of 0.1 km to 320 km MOLA altitude to cover the entire range that the MAV would have to cover based on the baseline MSR mission profile. Also, the latitude and longitude were varied within 10 degrees from the launch site with a step-size of 1 degree. A Matlab script was written to extract the relevant atmospheric data from the Mars-GRAM output files and write them into a .csv file, thus creating the required atmospheric data table. The atmospheric data in this table was then interpolated to provide an estimate of the atmospheric characteristics at every point along the ascent trajectory, which is required to compute the drag at each time step. Some of the earlier versions of Mars-GRAM are available for free (such as the Mars-GRAM 2005 version used in this thesis), however, the latests versions (such as the Mars-GRAM 2010 version) require a licence agreement. 6.2. DEVELOPED SOFTWARE This section of the software chapter describes the software that either had to be developed around existing software/libraries or had to be developed from scratch (the TSI propagator). Each piece of software is accom- panied by the corresponding software architecture. Every next piece of software then indirectly incorporates the previous architecture through the use of the completed tool. 1More documentation on Eigen can be found on eigen.tuxfamily.org/dox/ [Accessed 8 March 2016] 2 Tudat wiki: tudat.tudelft.nl [Accessed 17 February 2017] 3More documentation on Boost can be found on http://www.boost.org/ [Accessed 8 March 2016] 4Workshop presentation: http://solarsystem.nasa.gov/docs/Utilizing Mars-GRAM.pdf [Accessed 10 December 2015] 5NASA website: https://see.msfc.nasa.gov/model-Marsgram [Accessed 9 March 2016] 6.2.1. RKF PROPAGATOR The RKF (or traditional) propagator architecture is described in Figure 6.2. It starts with the current state, which is then passed on to the state derivative function, created by a user, and is then passed on as input.

The state derivative function is used by the RKF integrators to determine the next state by calling the function a number of times depending on the used method. The integrators that are available through the Tudat li- braries were already mentioned in Table 4.1 including the corresponding files. RK4 can be called by including the rungeKutta4Integrator.h header file, and RKF methods can be called by including the rungeKuttaVari- ableStepSizeIntegrator.h header file. This integration process is repeated until the final condition is met. Within the state derivative function all the state derivatives are updated and stored. The current position is used to update the gravitational acceleration on the MAV, the current mass is used to determine the acceler- ations caused by the thrust and finally the complete state is required to determine the accelerations caused by the drag. Both the drag and thrust accelerations have to be transformed to the inertial frame using the up- dated angles from the current state. The function also computes the current mass flow rate, however since the thrust is constant, this does not change over time. In the state derivative function, all the transformations are governed by pre-developed functions within the Tudat library, which includes the state transformations and the frame transformation

> 5from the body frame to the inertial frame. The transformation from the propulsion frame to the body frame is

however not included in Tudat and had to be written. 6.2.2. TSI PROPAGATOR The TSI propagator has a significantly different architecture compared to that of the traditional propagator as can be seen in Figure 6.3. TSI requires an initial order and step-size to start the integration process. In this thesis it has been decided to keep the order the same throughout the entire integration. The step-size will change during the integration depending on the Taylor series evaluations. The initial state is set as the current state and is fed into the TSI block. Within this block, first the auxiliary equations, derivatives and functions are called, which were set-up for this particular problem. They are evaluated using the current state and already include all the reference frame and coordinate transformations, as well as approximate atmospheric parameter functions. This is required to set-up the recurrence relations, which is where TSI differs from the traditional propagator. Once the auxiliary equations, derivatives and functions have been computed they are used to compute the Taylor series coefficients through the recurrence relations set-up for the thesis problem. These coefficients are then stored for later use and are also passed to the block creating the Taylor series expansion for every state variable thus creating the updated state. The last two coefficients are used to determine the next step-size. This continues until the final integration condition has been met. 6.3. COMPLETED SIMULATION TOOL The final simulation tool combines the integration software in such a way that a complete trajectory prop- agation can be simulated. It consists of different phases: initialisation, initial propagation, integrator burn phase, integrator coast phase, final circularisation and comparison. A brief description of each phase is pro- vided in this section. It should be noted that in the actual program, the TSI propagation (burn and coast phase) supersedes the RKF propagation. Also, three different RKF methods can be chosen: RKF4(5), RKF7(8) and DOPRIN8(7). However, for the final comparison RKF7(8) has been used for the initial propagation as well as the actual RKF propagation. 6.3.1. INITIALISATION During the initialisation phase, all the input values are loaded into the program. This is done by calling the input .txt files and assigning all the proper values to the corresponding parameters. Then, the Mars celestial body class and the MAV class are initialised, providing the Mars atmospheric and planetary data and the MAV system characteristics respectively. At this point, any values that differ from the pre-set default values are updated in both classes. The next step is to convert the initial spherical state into the initial cartesian state and both states are saved in the respective output files for both TSI and RKF. Finally, the StateAndTime class is initialised for TSI. 6.3.2. INITIAL PROPAGATION As will be described in Chapter 7, during the verification process it was found that TSI had trouble dealing with initial conditions close to its singular values during the initial second(s), both for velocity and flight-path angle. This is why it was decided to slightly shift the initial conditions at which the actual integration for TSI would start. Because RKF did not have similar issues in the first second(s), it was chosen to use an RKF method to propagate the first second. For this initial integration, the default integrator is RKF7(8) with a tolerance of $1 \cdot 10^{-15}$. The output of this initial propagation is then used as the initial state for both TSI and RKF for the remainder of the integration. In this way, both methods can still be compared properly. 6.3.3. INTEGRATOR BURN PHASE At the start of the burn phase the final state of the initial propagation is taken as the initial state input for the integrators. During the burn phase, both TSI and RKF are initiated and run until the end of the first burn phase occurs. Within this phase, TSI stops at every altitude and

Mach section of the atmospheric graphs to make sure that no errors occur in the model. Provided that the atmospheric graphs are piecewise continuous, such stops are not required for RKF. The end of the first burn phase can either be based on a set time, a set altitude or the zero flight-path angle requirement. Of course, extra cut-off criteria, such as maximum propellant mass, can be added. During the propagation, all intermediate states are stored in the output files for both methods. 6.3.4. INTEGRATOR COAST PHASE After the first burn phase, the MAV enters a coasting phase. This is simulated by setting the thrust equal to zero. For TSI this can simply be done by setting the thrust in the MAV class to zero within the propagation loop. This means that when the set burn-out condition is met, the class is updated, and the loop is continued. However, it is not possible to do this within one loop for the RKF, because of the manner in which RKF is initialised. In Tudat it is described that in order to use the built-in RKF functions, a separate state derivative class has to be built first. This class is initialised using the initial state classes, and therefore does not change when updating the initial state classes in the loop directly. This means that when the thrust changes to zero, the state derivative class has to be re-initialised, creating the need for a second separate loop. This also means that the integrator class has to be re-initialised as well, which means that a separate integrator has to be defined (one with a different name from the first one). Both the TSI and RKF coasting phases are terminated whenever the desired altitude is reached, a flight-path angle of zero degrees is achieved or after a certain limit time. Again, all the intermediate states are stored in the output files during the coasting phase. 6.3.5. FINAL CIRCULARISATION Once the final condition is met, both the RKF and the TSI final state are used to determine the required final circularisation burn in order to reach the desired final orbit. For this computation, the states are first converted into Kepler elements using the Tudat library. Then the required ΔV is computed

> 12**using the methods described in Section** 3.5. **In this**

case, an instantaneous burn to change the inclination, flight-path angle and orbital velocity to match the desired orbit is assumed. From the ΔV for both methods, it can be determined, using the Tsiolkovsky rocket equation what the required propellant mass is. This then results in a final mass estimate for both the RKF and the TSI propagations. 6.3.6. COMPARISON Finally, the end states of both TSI and RKF are compared to each other. The program provides the numeric difference between the position, velocity and mass as well as the quotient between those values. This quo- tient, or difference fraction, is the TSI state divided by the RKF state and the numeric difference is the TSI state minus the RKF state. These values were used in the verification and validation phase and are used during the thesis analysis as well to show the behaviour of TSI compared to RKF. Figure 6.2: RKF interface architecture Figure 6.3: TSI architecture 7 VERIFICATION AND VALIDATION Verification is the process of determining whether a program meets the requirements or not. Is it working the way it is supposed to? As soon as it works and produces output (verified), these outputs can be compared to other data from which it is known that it is correct. This is called validation. If a program is verified and vali- dated, the outputs should be correct. Fortunately, all the existing software has already been validated, which means that they only still have to be verified to make sure everything is working properly on the simulation computer. Each of the software packages comes with tests which can be used to do this. If all the tests are passed it means that the software is verified for the computer and ready to use. Since Tudat is shipped with Eigen and Boost, the Tudat test files also test these libraries. All the tests were passed, which means that the Tudat, Eigen and Boost libraries are working properly. However, because the integrators are an intricate part of this thesis, it was decided to perform a separate verification for the Runge-Kutta integrators. This verifica- tion is described in Section 7.1. Mars-GRAM also came with its own verification test, where three delivered output files had to be replicated. These verification tests were also successful. TSI had to be verified and validated separately as described in Section 7.2. And finally a complete verification and validation was done on the finished propagation tool. This final verification and validation is described in Section 7.3. 7.1. RK4 AND RKF INTEGRATORS The tutorial page of the Tudat website1 offers two integration tutorials: one involving RK4 and another involv- ing RKF. The objective of the tutorial is to become familiar with the different integration methods available in Tudat. At the same time, a small data table has to be reproduced, which also serves as a verification test. For each of the integrators the same problem was addressed: the computation of the velocity of a falling body after a certain amount of time assuming no drag. The results that had to be reproduced are presented in Table 7.1. Table 7.1: Verification data for the standard integrators End time [s] 1.0 5.0 15.0 25.0 Velocity [m/s] -9.81 -49.05 -147.15 -245.25 The first script was written using the

instructions from the tutorial and is called numericalintegrators.cpp. This script uses the rungeKutta4Integrator.h header file and the RungeKutta4IntegratorXd function from this header file. This function requires three inputs: the state derivative function (problem specific), the initial time and the initial state. Using the .integrateTo extension the end state at a certain end time can be com- puted. This requires the end time and the step-size. For RK4 the step-size is constant. This resulted in the same values as presented in Table 7.1. The second script was used to test the RKF methods. This script is called rungekuttavariable.cpp and uses the rungeKuttaCoefficients.h and rungeKuttaVariableStepSizeIntegrator.h header files. In this case the RungeKuttaVariableStepSizeIntegratorXd function was used which requires the Runge-Kutta coefficients (from 1 Tudat wiki: tudat.tudelft.nl [Accessed 17 February 2017] 52 7.2. TAYLOR SERIES INTEGRATION the respective header file), the state derivative function (problem specific), the initial time, initial state, zero minimum step-size, infinite maximum step-size, relative tolerance and the absolute tolerance as inputs. In this case the integration can be done in individual steps using .performIntegrationStep with the current step- size as the only input. The current step-size is computed in the integration method itself, but in this case it is checked to make sure that the step-size does not take the function beyond the specified end time. The integration steps are then repeated until the end time is met. Running this script again resulted in the same results as presented in Table 7.1. These results, combined with the fact that the test files for these two methods produced no errors, is proof that they are working accordingly. Thus it can be said that the standard integration methods used in this thesis are verified and ready for use in the simulation tool. However, during the development of the trajectory propagation tools, the entire tool (including the integrators) is verified again to determine the performance of the integrators. For the actual propagation tool, the RKF method required the state derivative function associated with this particular ascent problem. The state derivative class was constructed using the state derivative functions as described in this thesis in the cartesian coordinate system based on equations provided by Mooij (1994). The state is used to compute the required transformation angles first and then both the thrust and drag ac- celerations are transformed to the inertial frame. There they are combined with the gravitational acceleration resulting in the accelerations

1**in the** x-, **y- and z-direction.**

This class was tested by itself by inputting certain values which would result in known outcomes. Those simple values by themselves provided the expected re- sults. Later the state derivative function was combined with the RKF integrator in the trajectory propagation tool and verified again as will be described in Section 7.3. 7.2. TAYLOR SERIES INTEGRATION In this section, the verification and validation of the TSI header and source file is described. This was done for each of the blocks shown in Figure 6.3. For the verification it was important that the running of the different classes and header/source files did not produce any obvious errors. For the validation, the results were often checked against manual calculations and expected behaviour, as well as results produced by RKF. Section 7.3 describes the early phases of the verification. This section has been divided into several subsections which showed similar issues during the verification and validation. 7.2.1. AUXILIARY EQUATIONS, DERIVATIVES AND FUNCTIONS As shown by Figure 6.3 the first step in the TSI is to compute the auxiliary equations, derivatives and func- tions. The computations, in theory, are fairly straightforward, however in practice the code can be sensitive to programming errors. During the initial verification of these different classes the outcomes were checked against simple manual solutions. Thus, provided a certain input, the output was known. This was an easy way to identify any coding errors. The early versions of the code required a large number of equations and hence calculations, which all had to be checked. This was initially done with the combination of cartesian coordinates and spherical transformation angles, however, later on more efficient equations were used for separate cartesian and spherical cases. This also made it easier to identify mistakes in the equations and understand certain issues found during the validation process. 7.2.2. TAYLOR SERIES COEFFICIENTS AND UPDATED STATE Once the auxiliary classes were verified, the Taylor series coefficients could be computed using the basic re- currence relations and the full set of recurrence relations. For the Taylor series coefficients it is important that they show a decreasing number series for each of the coefficients. Meaning that each consecutive co- efficient corresponding to a higher derivative should be smaller than the previous one. For example: the coefficient corresponding to the first

derivative has a value of 10 and the coefficient corresponding to the sec- ond derivative has a value of 2. This was the desired outcome that was used for the verification

29**of the full set of recurrence relations. The**

basic recurrence relations were all checked by hand calculations to determine if they indeed worked the way that they were supposed to. The verified basic recurrence relations were used to verify the

29**full set of recurrence relations.** This class consists **of the** equations mentioned in **Appendix**

D. When writing

29**the full set of recurrence relations,**

it is important to abide by the rules that have mentioned in Chapter 5, otherwise errors can be introduced in the code. Also, a number of these recurrence relations were checked by manual calculations. An often recurring problem was that the Taylor series coefficients would show a diverging behaviour where the coefficient would keep increasing to infinity (theoretically). This was often an indication that a writing mistake was made when setting up the full set of recurrence relations. Another easy way to identify mistakes during the validation pro- cess was to compute the next state. Should an error occur in the computation of the Taylor series coefficients, this would usually result in very large increases in the state even if the coefficient itself would be relatively small. Then following the trail back to the largest change in Taylor series coefficient, often a coding or writing mistake could be identified in the full set of recurrence relations. 7.2.3. NEXT STEP-SIZE Fortunately, the step-size class was not a very large class and took less time to verify and validate. Provided a set of Taylor series coefficients, the step-size class could be tested separately from the other files. The verifi- cation process was rather straightforward; the class should be able to provide a step-size estimate given the different input values. Also, when similar sets of Taylor series coefficients were used, the output step-sizes should be rather similar. This was then also part of the validation process. A pure validation process was dif- ficult because there were no reference Taylor series coefficients available, however an estimate for the order of the next step-size could be used. This can however still be improved. Two different step-size determination methods were tested, but in the end it was found that the direct method described by Scott & Martini (2008) and Bergsma (2015) showed more reasonable results and was easier to implement. Also, during the verifi- cation process, the iteration method failed to work properly which led to unusable results and the program failing. However, the direct method worked very well. 7.3. COMPLETE TRAJECTORY PROPAGATION For the complete trajectory propagation it was important to have the integrator method as the only difference between the RKF and the TSI propagation. First the initial values were provided and transformed into the proper coordinate system. These would then be fed into the integrator and at the end of the integration, the results would be printed. The verification and validation of the complete trajectory propagation have been split up into several phases. 7.3.1. PHASE 1: RKF PROPAGATION This was first done for RKF because this method was already validated. Therefore, any mistakes or problems found would lead back to a mistake in the state derivative functions or the model itself. During this phase, a number of problems were identified; certain singularities were found, a limit to the tolerance value was determined for each of the RKF methods (DOPRIN8(7) does not converge fast enough in many cases for a value of $1 \cdot 10^{-15}$) and a mistake in the model was found where the wrong velocity was used to determine the drag acceleration. This meant that some of the model equations had to be rewritten for both RKF and TSI. 7.3.2. PHASE 2: BOTH RKF AND TSI During the second phase of the verification, both RKF and TSI were tested at the same time and the outcomes were compared. Since the RKF method was verified and validated at this point, the outcome of the RKF propagation could be used to verify TSI. This was done using different test cases that focussed on different elements of the code. Each element was tested separately by activating (or deactivating) the gravity, thrust and drag as well as the rotation of Mars. In each case, simple test cases were used to verify the class, such as: vertical drops from a certain altitude, thrusting in a known

direction and flying parabolic trajectories. This was all done in each of the axis directions first and then in arbitrary directions. First the gravity modelling was tested, then the thrust, then a combination of these two, followed by the drag, all in the non-rotating case. The same was then done for the rotating case as well. This was initially done for a zero thrust elevation and thrust azimuth angle, such that the trust was acting directly through the x-axis of the body frame. Also, because in this model the drag is defined to act through the x-axis of the body frame, there should not be any accelerations

1**in the y- and z-direction in the**

body frame. This simplified the problem and meant that some of the equations could be tested without involving all the other equations just yet. During this phase it was discovered by analysing the model equations, that there was an inconsistency in the manner in which the different accelerations were computed. Basically, the rotational angles were being computed using spherical equations which already incorporated the accelerations and then those angles were used to transform the cartesian coordinates and incorporate the accelerations again. Therefore it was decided to rewrite all the equations and actually come up with three different sets: two cartesian models and one spherical model for TSI. The associated equations were already described in Chapter 5. 7.3.3. PHASE 3: THREE DIFFERENT MODELS FOR TSI The third phase consisted of verifying each of these three models in a similar manner as was done during the second phase. During this phase, it was discovered that a starting ground velocity too close to zero would cause singularities in both the cartesian as well as the spherical model. This is why a small initial ground velocity was introduced, which seemed to work for the second cartesian case (unit vector transformations) and the spherical case, however. Unfortunately, this did not work for the first cartesian case (Euler angles) meaning that there were still either coding errors or errors in the model. At this point it was decided to focus on the other two methods instead because of time constraints. It was also found that the initial flight-path angle could not be equal to (or close to) 90 degrees for the spherical case due to singularities in the model. This is why a value of 89 degrees was chosen for the test cases of that particular model. This worked well for gravity and thrust for the spherical and second cartesian case. But not always. 7.3.4. PHASE 4: ONE FINAL TSI MODEL As it turned out, setting an initial ground velocity for the TSI models (cartesian and spherical) was only a temporary solution and did not solve the problem completely. Tests revealed that the initial second is vital for TSI and when the velocity is too small, the Taylor series coefficients tend to increase in value. The solution was to off-set the initial conditions to a later point in the trajectory. This off-set was achieved by having the first second of the trajectory be integrated by RKF7(8) with a set tolerance of $1 \cdot 10^{-15}$ (if the method and the tolerance was not set constant, the results of the integration would be inconsistent). When this was applied to the propagation involving TSI the Taylor series coefficients dropped to an acceptable number again which could then be used to determine the next state. This worked really well for the spherical case, however the second cartesian case still had some bugs (either in the model or in the coding). Again, due to time constraints it was decided to solely continue with the spherical model and use that for the rest of the thesis work. This is the reason that an initial propagation phase was needed as was described in Section 6.3. At the end of the phase, the TSI code was deemed verified, since it showed similar results to the RKF propagation. 7.3.5. PHASE 5: COASTING AND CIRCULARISATION Once the TSI method was verified, the rest of the trajectory propagation could be added. For the coasting phase this meant that the thrust had to be set equal to zero at the point of main engine cut-off. Fortunately, this was a simple update in class value for the TSI method, however this did not work for RKF. Instead it was discovered that because of the manner in which the state derivative class is set-up and used in the integrator itself, the class had to be re-initialized if anything in the state derivative class changed. Therefore, a new inte- gration had to be started for RKF at main engine cut-off. Because the circularisation is treated as an instantaneous burn, this only affected the total propellant mass and velocity of the MAV. Therefore, it could be treated with simple circularisation equations described by Wakker (2010b). The velocity required to reach a perfect desired circular orbit can be computed. Then be- cause the equations are simple and can be solved by hand, the code could be verified (and validated) by using simple cases computed using the program and then compared to the solutions computed by hand. 7.3.6. PHASE 6: TOOL VALIDATION The most important part of this thesis is the comparison between RKF7(8) (from now on simply refered to as the general RKF) and TSI, therefore these two methods should produce results that can be compared and where the only difference is the manner in which they were computed. Provided that the RKF method is verified and

validated and the corresponding model validated (as was done in phase 2), TSI could be validated using RKF with this purpose as mentioned before. However, besides this validation, it should also be checked if these solutions represent a real-life solution to the problem. This means validating both methods against other validated simulation outcomes from other people. Unfortunately, no flight data is available for a Mars ascent case yet. Two JPL sources were found that could potentially be used to validate the software: Woolley (2015) and Benito & Johnson (2016). Woolley used an analytical model to approximate an ascent trajectory for different launch cases and conditions, one of which was an SSTO launcher which was used for validation. Benito on the other hand used a high-fidelity simulation program to simulate the ascent trajectory of some of the newer concept MAV designs numerically. Some of the latest data were provided and were also used for the validation. Table 7.2 gives the validation values associated with the simulated trajectory of the case as presented by Woolley (2015). The left columns show the inputs that were provided and the inputs that were used in the program. The right columns show the results of the simulation as provided and as computed by the program. The thrust angles in that column represent the required thrust angles that were needed as input to get the desired outcome. Table 7.2: Validation case 1: SSTO Woolley (2015). Provided parameters Given Used Provided parameters Given Computed Thrust [kN] 3.56 3.56 Propellant mass [kg] 202.2 202.07 Isp [s] 256 256 Burn-out angle [deg] 6 5.344 Burn Time [s] 142.6 142.5 Circularisation ΔV [m/s] 181 159.6 MAV GLOM [kg] 267.4 267.4 Ascent time [s] 1757 1796.27 Launch altitude [km MOLA] - 0.6 Thrust azimuth [deg] - -0.299 Launch latitude [deg] 0.0 0.0 Thrust elevation [deg] - -0.184 Launch longitude [deg] - 74.5 $1 \cdot 10^{-5}$ Launch ground

22**velocity [km/s]** - Launch **Flight-path angle [deg]** 90 89 Launch **Azimuth [deg]**

90 90 Desired altitude [km MOLA] 390 390 Desired inclination [deg] 45 45 It can be seen that there are some differences which are caused by the fact that the thrust azimuth and eleva- tion angle were unknown and had to be optimised to fit the desired end trajectory by hand. For the purpose of this research however, the validation is to prove that the trajectory computed by the simulation program is indeed a viable trajectory (so representative of a reference trajectory). This is because that is enough to compare the RKF and TSI methods. The plotted trajectories are shown in Figure 7.1. Figure 7.1: Case 1 trajectory as seen from the y-axis. Table 7.3: Differences in state between simulation performed using RKF and using TSI for case 1. State parameter Difference Order of difference x-position [km] less than µm $-5.25 \cdot 10^{-10}$ $-5.55 \cdot 10^{-10}$ y-position [km] less than µm $5.06 \cdot 10^{-10}$ z-position [km] less than µm $-3.86 \cdot 10^{-13}$ x-velocity [km/s] less than nm/s $1.75 \cdot 10^{-13}$ y-velocity [km/s] less than nm/s $4.02 \cdot 10^{-13}$ z-velocity [km/s] less than nm/s $-6.25 \cdot 10^{-13}$ Mass [kg] less than ng For a proper comparison, the final solution of both RKF and TSI have to be representative of the same end state. This means that the difference between the end states of both simulations has to be very small. In Table 7.3 the difference in end state between the two integrators is presented. A similar validation was done using the second case. However, in this case the reference trajectory was pro- vided. The main difference here was that the simulation program is only able to use a constant thrust eleva- tion and azimuth angle at this point. Therefore, the exact profile could not be matched. Benito & Johnson (2016) also included extra perturbing effects creating another difference. However, it was possible to get a tra- jectory close to the one provided, which is enough for all current purposes. Table 7.4 shows the used values and computed parameter values for case 2 (similar to case 1). Again, the trajectory was visualised, however, this time the reference trajectory was also known. In the figures this is represented by a red line. These are shown in Figures 7.2 and 7.3. Table 7.4: Validation case 2: SSTO Benito & Johnson (2016). Provided parameters Given Used Provided parameters Given Computed Thrust [kN] 6.1087 6.1087 Propellant mass [kg] 213.85 213.85 Isp [s] 315.9 315.9 Burn-out angle [deg] - 15.644 Burn Time [s] 99.362 99.362 Circularisation ΔV [m/s] 757.57 594.74 MAV GLOM [kg] 288.96 288.96 Ascent time [s] 949.12 876.09 Launch altitude [km MOLA] -0.8 -0.6 Thrust azimuth [deg] - -0.7273 Launch latitude [deg] 0.0 0.0 Thrust elevation [deg] - -0.674 Launch longitude [deg] 90 $1 \cdot 10^{-6}$ 90 $1 \cdot 10^{-6}$ Launch ground

22**velocity [km/s]** Launch **Flight-path angle [deg]** 90 89 Launch **Azimuth [deg]**

90 90 Desired altitude [km MOLA] 479.19 479.19 Desired inclination [deg] 92.69 92.69 Again, the difference between the outcomes of the two integration methods has to be very small. The differ- ences

are presented in Table 7.5. The difference between the computed (using RKF and TSI) and provided trajectory (red) can mainly be ex- plained by the thrust elevation and thrust azimuth angle curve of the reference trajectory. This curve was derived from the thrust data provided by Benito & Johnson (2016) and is shown in Figure 7.4. Figure 7.2: Case 2 trajectory as seen from the x-axis. Figure 7.3: Case 2 trajectory as seen from the y-axis. Table 7.5: Differences in state between simulation performed using RKF and using TSI for case 2. State parameter Difference Order of difference x-position [km] less than μm $2.26\cdot10^{-10}$ $-3.23\cdot10^{-11}$ y-position [km] less than μm $2.77\cdot10^{-10}$ z-position [km] less than μm $-2.02\cdot10^{-14}$ x-velocity [km/s] less than nm/s $1.46\cdot10^{-13}$ y-velocity [km/s] less than nm/s $1.77\cdot10^{-13}$ z-velocity [km/s] less than nm/s $4.26\cdot10^{-14}$ Mass [kg] less than ng Figure 7.4: Case 2 thrust elevation and azimuth angle curve derived from Benito & Johnson (2016) including the fits that could be used for an updated version of the simulation tool. 8 RESULTS AND ANALYSIS With the program validated, both the ascent towards a circular orbit, with an inclination of 45° i r c in case 1 (as described by Woolley (2015)), and case 2, where the ascent trajectory towards the south pole of Mars with a desired circular orbit with an inclination of 92.69° i r c (as described by Benito & Johnson (2016)), can be used to examine the characteristics of TSI. This examination into the characteristics is performed with TSI as a black box. The investigation is performed in a sensitivity analysis and is very useful to determine the limi- tations and characteristics of this method. In the previous chapter the trajectories from both TSI and RKF were plotted and it could be seen that the results were very similar to the point where the accuracy of TSI was validated. Therefore, in this chapter the performance of TSI will be compared to RKF to determine any advantages and disadvantages. To do this only one variable will be changed every time, which means that from the nominal case only one parameter will be different. In this way the effects can be clearly associated with that particular parameter, which provides a solid base of comparison. The nominal values for both the first and second case are provided in Appendix E. It is also important to have the same cut-off point. This can either be a certain altitude, set end time or zero degrees flight-path angle. Usually the set end time is taken as the cut-off criterion to determine the difference in state. Sometimes the altitude is used in cases where it is interesting to see when a certain altitude is reached and under what conditions when a certain parameter is changed. An example was shown in the validation chapter where certain conditions had to be met at a certain altitude. In some cases it is best to set the zero degrees flight-path angle as a cut-off criterion because either the cut-off time or altitude will never be reached but the effects of a parameter still need to be inves- tigated. This last cut-off criterion is the "fail-safe" for every simulation because this prevents the simulation from going back to the surface and literally crashing the program. In this chapter every section discusses the effect of a different parameter or focusses on one aspect of the simulation. Besides comparing TSI to RKF it will also be compared to the non-rotational case. The nominal case takes the rotating Mars into account, however it is interesting to see what TSI does when the rotation is turned off. For these two cases to be properly compared, again for most parameters, the set end time is the cut-off criterion unless mentioned otherwise. The parameters changed in this sensitivity analysis are: order, error tolerance, launch altitude, launch lati- tude, launch longitude, flight-path angle and heading angle. A multiple-run analysis will also be described where the nominal case is run for a number of times and the CPU time is compared. 8.1. ORDER For most classic integrators the order is set and cannot be changed easily. For RKF for instance, a whole new set of coefficients is required if a different order is desired (hence RKF4(5), RKF5(6), RKF7(8) etc.). However, for TSI changing the order is as simple as updating the maximum order input value. Theoretically this means that an infinite order can be chosen to get the most accurate result. The question is, is this indeed the most optimal. Section 8.1.1 hopes to answer this question by looking at the CPU time. In Section 8.1.2 a range of orders has been run for both cases: once with the nominal conditions of a rotating Mars and once assuming a non-rotating Mars. The effect of the rotating Mars will be investigated. The cut-off criterion for case 1 was 60 8.1. ORDER an end time of 1796 seconds for the data collection run and 789 seconds without live data collection. For case 2 the cut-off criterion was 876 second for both runs. The criterion values were based on the minimum ascent time per order. The ascent time varied slightly depending on the chosen order, but to be able to compare the different orders it was important to choose a criterion such that the cut-off point was the same for all orders. 8.1.1. OPTIMAL ORDER To determine the optimal order an important criterion is that the outcome is accurate, however, what is often more important is that this accurate outcome is computed really fast. This is why, to find the optimum order, the CPU time is the most important criterion. To properly observe the effect of a different order, the nominal case is run for different orders. Thus the only difference will be the maximum order. This can be done for the two presented cases. During real-life simulations a user can choose to either collect all the data during the simulation or only store the final out- comes at the end of the simulation. Storing data during the integration requires a large amount of

computa- tional power and will therefore increase the CPU time required, but it will provide the user with the desired trajectory data. Both runs with and without live data collection have been run for a specific range of orders depending on the case, and the results are shown in Figure 8.1. The first case was run with a maximum order ranging from 5 to 100, and the second case was run from 5 to 31, because of the behaviour observed during the first case runs. The number of function evaluations has been plotted in the same graph on a logarithmic scale. It should be noted that any intersections between the CPU curve and the function evaluations curve is meaningless since they use different axes. (a) (b) (c) (d) Figure 8.1: Order versus CPU time and number of function evaluations (a) case 1 including live data collection, (b) case 2 including live data collection, (c) case 1 without live data collection, (d) case 2 without live data collection. 8.1. ORDER Looking at Figure 8.1a it is clear that the order versus CPU curve can be split up into two phases that have different properties that influence the CPU time. The first phase shows a rapid decrease in the CPU time until order 20 is reached, the second phase continues from that point until the final order is reached. The order has a direct relation to the chosen step-size that TSI computes automatically because the step-size is determined by the error tolerance and the difference between the penultimate Taylor series coefficient and the final one (as described in Section 5.1.2). This means that if these last two coefficients show a large difference, which can happen at lower orders because fewer derivatives are taken into account, the step-size will be lowered; increasing the number of steps. In the first phase, this large number of steps result in a high CPU time as can be seen in each of the graphs. Another factor that affects the CPU time is the number of derivatives taken into account, which is directly related to the order (e.g., 10 as maximum order equals 10 derivatives that are taken into account). At a certain point, an increase in order does not affect the number of steps that much any more (the function evaluation curve flattens out), resulting in a decrease of 3 to 1 steps with every order increase. The result is that the number of steps remains approximately the same but the number of derivatives that are taken into account keeps increasing. Every extra derivative means an extra computation per auxiliary equation/variable. This is what is observed in the second phase. Here the effect of adding more computations on the CPU time becomes clear, showing a corresponding increase in CPU time. This tipping point between the two phases can be observed around the lowest CPU time, where the effect

47**of the number of steps** and **the number of**

derivative computations equal out. This is therefore the optimum order for the TSI method. The two phases can be observed in each of the four graphs, however, because the increase of CPU time, the higher orders were not required and were thus left out in the simulations for case 2. For each of the simulation runs, the two orders which resulted in the lowest CPU time have been recorded in Table 8.1. This was done because the CPU time measurements had a guaranteed accuracy of three decimal figures. Table 8.1: Lowest two orders with corresponding CPU times rounded to four decimal figures for each of the sub-graphs in Figure 8.1. Sub-graph Lowest CPU time Second lowest CPU time Order CPU time [s] Order CPU time [s] a 20 0.0532 21 0.0532 b 25 0.0835 22 0.0837 c 14 0.0075 20 0.0077 d 18 0.0083 20 0.0083 In the table and the graphs it can be seen that the minima are all situated around an order of 20, which is why a maximum order of 20 was set as the nominal value and used for all simulations. It should also be mentioned that the sub-graphs where there is no data collection should be considered more accurate than the figures with live data collection for the purpose of determining the optimum order. However, during an actual simulation at that optimal order, there will most likely be live data collection which is the situation that is shown in the sub-graphs representing the live data collection for each order in Figure 8.1. 8.1.2. COMPARISON WITH NON-ROTATING MARS For the comparison of the rotating and the non-rotating cases with a varying order it is again interesting to look at the differences in CPU time. Figure 8.6 shows these plots for both cases when the trajectory data is not collected during the simulation. It can be seen that for case 1, on average, the non-rotating runs are faster. Also, for a non-rotating Mars the optimum order is 20 as well. The difference in CPU time is better shown by the case 2 run, where the non-rotating curve falls perfectly below the rotating curve. Here, the order 20 has again the second lowest CPU time (the lowest is order 17 with a time difference of 28 microseconds, which is out of the guaranteed accuracy). The time difference between the curves of the rotating and non-rotating Mars can be explained by the fact that if Mars does not rotate, the rotating effects become zero, which reduces the number of operations because many of the recurrence relations will then be equal to zero. 8.1. ORDER (a) (b) Figure 8.2: Order versus CPU time for rotating and non-rotating Mars (a) case 1, (b) case 2. As was mentioned

before, the accuracy of the results is very important. In Figure 8.3 the absolute difference between the nominal end state of RKF before circularisation and TSI for both the rotating and non-rotating Mars are plotted. A trend can be identified where it is shown that an order of 11 or higher will result in a difference of μm in position and nm/s in velocity. Even the lowest order of 5 results in an accuracy of less than 10 cm for position and 10 mm/s for velocity. The spikes in the first case are caused by a slight error resulting from the combination of variables that result at those specific orders. The Taylor series coefficients show that the time steps are similar until approximately 10 seconds. Order 25, 27 and 29 then have a time step resulting in a step end-time of 12.7 - 12.9 seconds, whereas the other orders have a larger time step here. This points to an error in the step-size determination and a combination of variables or state for which the Taylor series coefficients run amok. This unfortunately proves that the simulator is not yet robust enough and could still use some improvements. One improvement that might solve this problem is a change in the manner in which the step-size is computed. The figures also show that the order at which both RKF and TSI show the best correlation is at approximately order 20, which is also the minimum CPU time order. Providing extra proof that an order of 20 is optimal. (a) (b) Figure 8.3: Difference with respect to RKF case for rotating and non-rotating Mars (a) case 1, (b) case 2. 8.2. ERROR TOLERANCE The error tolerance is used to determine the required step-size during the integration. A low error tolerance (say 10^{-5}) is therefore expected to provide an end result that is less accurate than a higher error tolerance (such as 10^{-15}). For the RKF7(8) provided by Tudat the highest accuracy that can be used is 10^{-15}, which is why that accuracy is used as the comparison accuracy. When a range of error tolerances is examined the accuracy of the end state and the speed of convergence can be determined. The speed of convergence follows from the consecutive differences between the end states (that is the difference between every state as a result from the error tolerance in sequence: so the state of 10^{-5} - 10^{-6}, 10^{-6} - 10^{-7} etc.) and the accuracy of the end states comes from the comparison to the nominal case. Both these characteristics will be examined and compared for RKF and TSI, see Section 8.2.1, and for the rotating and non-rotating Mars described in Section 8.2.2. The cut-off criterion for case 1 was 789 seconds and 876 seconds for case 2 and both cases were run without live data collection. 8.2.1. COMPARISON WITH RKF7(8) The speed of convergence is important because it shows the relationship between the different error tol- erances. If the differences are small it means that the change in error tolerance had a small effect. If the consecutive difference is large, it means that the solution has not yet converged. Figure 8.4 shows the consec- utive differences in position and velocity for the two cases for both integration methods. Here it can clearly be seen that TSI converges much faster than RKF. The curve of RKF lies quite a bit higher, which means that the differences are larger, and thus it converges slower than TSI. (a) (b) (c) (d) Figure 8.4: Error tolerance versus consecutive difference between end states before circularisation for TSI and RKF (a)

54**case 1** position, **(b) case 2** position, (c) **case** 1 velocity, **(d) case**

2 velocity. A similar comparison can be done when comparing each of the end states directly to the nominal end state. The result is a nominal absolute difference graph as presented in Figure 8.5 for position and velocity for both cases. Here it can be seen that the accuracy of the position at an error tolerance of 10^{-5} is already in cm for TSI whereas this accuracy is only reached at an error tolerance of 10^{-9} for RKF. For the velocity the TSI accuracy is in the order of 10 μm at this initial error tolerance which is reached at an error tolerance of 10^{-9} using RKF. This means that TSI is, on average, more accurate at any given error tolerance. (a) (b) (c) (d) Figure 8.5: Error tolerance versus nominal absolute difference between end states before circularisation for TSI and RKF (a)

54**case 1** position, **(b) case 2** position, (c) **case** 1 velocity, **(d) case**

2 velocity. Another important observation can be made when looking at the difference in the number of function eval- uations and CPU time for each error tolerance for both RKF and TSI. The values for both cases are shown in Table 8.2. Table 8.2:

46**Number of** function **evaluations and CPU time for**

RKF and TSI. Error Case 1 Case 2 tolerance

46 **Number of evaluations CPU time** [ms] **Number of evaluations CPU time**

[ms] RKF TSI RKF TSI RKF TSI RKF TSI $10-5$ 169 24 0.419 4.703 156 24 0.392 5.353 $10-6$ 182 24 0.432 4.57 182 25 0.49 5.518 $10-7$ 221 26 0.542 4.769 208 27 0.527 5.823 $10-8$ 260 29 0.614 5.272 247 29 0.567 6.354 $10-9$ 312 30 0.823 5.571 286 30 0.709 6.439 $10-10$ 364 32 0.845 5.685 338 33 0.789 7.083 $10-11$ $10-12$ 455 36 1.032 6.227 390 36 0.904 7.454 $10-13$ 546 38 1.277 6.506 546 39 1.231 8.02 $10-14$ 702 41 1.541 7.062 728 41 1.604 8.627 $10-15$ 910 42 1.965 7.227 936 45 2.055 9.538 1222 46 2.666 7.754 1300 47 2.851 9.538 Here one evaluation is defined as running through the derivative equations once. For TSI this means that each time step, one evaluation is performed because all desired derivatives are computed during one com- putational run. In the case of RKF7(8) however, each time step consists of 13 computational runs. Table 8.2 shows that for both cases the number of evaluations of RKF are approximately one order more at the low error tolerances but almost two orders higher at the highest tolerance compared to TSI. This means that a decrease in error tolerance results in a large increase in number of evaluations for RKF but a relatively small increase for TSI. For both cases, the number of evaluations is not even doubled from the lowest to the highest tolerance, whereas the maximum number of evaluations for RKF is 7 to 8 times more. This advantage, however, does not translate well to the CPU time where it can be seen that for most of the error tolerances RKF is an order faster than TSI. This means that one function evaluations for RKF requires less time than the evaluation of a TSI evaluation. The similar increase in CPU time for both integrators is nevertheless still observed. An explanation for the large difference in CPU time could be that TSI was not optimally programmed and thus results in a relatively high CPU time. Further analysis is required. 8.2.2. COMPARISON WITH NON-ROTATING MARS Looking at the rotating Mars and non-rotating Mars runs, a noticeable difference in the CPU time can be observed as shown in Figure 8.6. In both cases the non-rotating Mars run is approximately 0.2 ms faster. This is similar to what was observed with the order runs where the result for an order 20 run is the same as the result for an error tolerance of $10-15$. The real question is, is there a noticeable difference in the convergence speed and the accuracy? This question can be answered by looking at Figures 8.7 and 8.8. Here it can be seen that the rotation of Mars does not have a significant influence on the convergence speed nor the accuracy of the results. (a) (b) Figure 8.6: Error tolerance versus CPU time for rotating and non-rotating Mars (a) case 1, (b) case 2. 8.3. MULTIPLE RUNS (a) (b) Figure 8.7: Error tolerance versus consecutive difference for rotating and non-rotating Mars (a) case 1, (b) case 2. (a) (b) Figure 8.8: Error tolerance versus nominal absolute difference for rotating and non-rotating Mars (a) case 1, (b) case 2. 8.3. MULTIPLE RUNS In Section 8.2.2 it was shown that in this thesis research, the TSI turned out to be slower than RKF for both tested cases. This is contrary to the reference research by Scott & Martini (2008) and Bergsma & Mooij (2016). Therefore, it is a good idea to look at the differences that could occur when measuring CPU time for dif- ferent runs. In this section the nominal case has been run 5000 times in sequence for both RKF and TSI (Section 8.3.1) using a rotating and a non-rotating Mars (Section 8.3.2). If the computer and the CPU mea- surements would be perfect, there should be no difference in CPU time and there would simply be two lines. This is never the case, but that does not mean that the results cannot be compared. Even if the program would be run on another computer, RKF would still be faster than TSI. 8.3.1. COMPARISON WITH RKF7(8) Again the notion that RKF is faster than TSI is confirmed when looking at Figure 8.9, however for both cases there is not one single line per integrator. Instead several outliers and patterns can be observed. 8.3. MULTIPLE RUNS (a) (b) Figure 8.9: Nominal runs versus CPU time comparison between RKF and TSI (a) case 1, (b) case 2. For case 1 it can be seen that there were some periods where the CPU time was slightly higher. This is even clearer for case 2 where two different CPU levels can be distinguished and many random CPU times in be- tween. The CPU time is computed using the internal clock of the computer. It could occur that background programs are run while the integration is running. In such a case, the CPU time computed for that particular integration run can turn out to be higher than expected because the internal clock cannot differentiate be- tween the simulation program and all other programs running at that same instance, as was the case for the second case run. Therefore, even if the same nominal case is run, the CPU time could still be 30% higher or more. 8.3.2. COMPARISON WITH NON-ROTATING MARS The effect of a program running in the background during a simulation run becomes even more clear when looking at the rotating and non-rotating Mars CPU times shown in Figure 8.10. (a) (b) Figure 8.10: Nominal runs versus CPU time for rotating and non-rotating

Mars (a) case 1, (b) case 2. Here it can be seen that, on average, the non-rotating Mars runs are fasters than the rotating Mars runs, which is similar to what has been observed before. However, in both cases there was a spike where the CPU time suddenly increased significantly for the non-rotating runs. This shows, that at that time a program (or pro- cess) was running in the background and clearly affected the performance of the simulation to a large extent. This means that CPU time is not always an accurate representation of the performance of the simulation. As a matter of fact, if the same program would be run on a different computer, the CPU times could be com- pletely different. But it would still be consistent and show the same relationship between the rotating and non-rotating Mars runs as well as RKF and TSI and does therefore not explain why TSI is so much slower compared to RKF. 8.4. LAUNCH ALTITUDE From now on only the rotating runs and the non-rotating runs will be compared. This can be done because TSI has proven to be accurate compared to RKF producing similar results. Thus it is only interesting to look at the effect of the different parameters on the performance of TSI. In this case the effect on CPU time, consecu- tive difference and absolute difference will be investigated as well as the effect on end radius and end velocity before cirularisation. Here the cut-off criterion for case 1 was an end time of 789 seconds and 856 seconds for case 2 and was determined in a similar manner as before. In Figure 8.11 the launch altitude is plotted against the CPU time. This has been done for the altitude range from the candidate altitude of -0.6 km MOLA to the maximum altitude where the MAV could be launched from: 0.5 km MOLA. In this case, no clear pattern can be observed due to the large differences in CPU time. Therefore it can be concluded that the launch altitude in this particular range does not have an effect on the CPU time that can be detected. (a) (b) Figure 8.11: Launch altitude versus CPU time for rotating and non-rotating Mars (a) case 1, (b) case 2. Randomness can also be observed when looking at the consecutive difference for the rotating runs in Fig- ure 8.12. Here the z-position and z-velocity show some slight deviations for case 1, and in the curves of case 2 the rotating states all show deviations. However, it is interesting to notice that no deviations are observed for the non-rotating runs, which means that all the deviations are caused by rotating effects. A similar behaviour can be observed in the nominal absolute difference graphs presented in Figure 8.13. Here case 1 shows al- most no deviations except for the z-position and z-velocity. However, again large deviations can be observed for case 2. These deviations can be caused by the fact that for case 2 the MAV is launched due east but quickly heads south therefore going against the rotation of Mars and creating variations that are not observed for the non-rotating Mars. Another example of the effect of the rotating Mars can be seen in the radius graphs in Figure 8.14. Be- cause in both case 1 and 2 the MAV is launched due east on the equator it is able to get an initial ΔV boost. This increase in velocity compared to the non-rotating Mars results in a final radius that is higher than the non-rotating Mars runs for both case 1 and case 2. It can also be observed that the radius of the final orbit increases with an increase in launch altitude. This could be a direct effect of both the drag and the gravita- tional acceleration, because the atmosphere becomes less dense and the gravitational acceleration becomes smaller the further away from the surface the MAV is, which is independent of the latitude and the longitude. (a) (b) Figure 8.12: Launch altitude versus consecutive difference for rotating and non-rotating Mars (a) case 1, (b) case 2. (a) (b) Figure 8.13: Launch altitude versus nominal absolute difference for rotating and non-rotating Mars (a) case 1, (b) case 2. (a) (b) Figure 8.14: Launch altitude versus orbital radius for rotating and non-rotating Mars (a) case 1, (b) case 2. (a) (b) Figure 8.15: Launch altitude versus velocity for rotating and non-rotating Mars (a) case 1, (b) case 2. In Figure 8.15 the ground and inertial velocity for both cases and both rotating and non-rotating Mars have been plotted. These are the end velocities of the MAV before circularisation. And because an increase in final radius was observed, a decrease in end velocity was expected and this is indeed observed for both cases. When comparing ground velocity it would be expected that the rotating Mars run would show a lower velocity because Mars is actually rotating with the MAV. This effectively lowers the speed at which the MAV travels as observed from the Martian surface. This is indeed seen in both cases. What is also noticeable is that there is fortunately no velocity difference between the inertial and ground velocity for the non-rotating case, which means that the rotation of Mars was indeed zero and the program is working properly. There is, however, a large difference between the ground and inertial velocity for the rotating runs, which is precisely the effective velocity boost from the rotating Mars. Finally it is interesting to notice the difference in end velocities between case 1 and case 2. Case 1 has an inertial velocity which is higher than the ground velocity and also the velocity for the non-rotating run. This is what one would expect for a MAV that is launched due east. However, case 2 also launches due east and shows an inertial velocity for the rotating run which is lower than all other velocities. This can be explained by taking into account that these are all final velocities before the circularisation and recalling the trajectory profile of both cases as presented in Chapter 7. The trajectory of case 1 takes the MAV slightly north after launch, whereas in case 2 the

trajectory goes south and even slightly west. This means that it is moving against the rotation of Mars and thus effectively loses some velocity when compared to both the ground velocity and the velocity in the non-rotating run. This effect was therefore expected. 8.5. LAUNCH LATITUDE For the latitude comparisons, the CPU time, radius and velocity are compared. The cut-off criterion was an end time of 789 seconds for the first case and 783 seconds for the second case. Again, starting with the presentation of the CPU time as shown in Figure 8.16. Even though the curves from case 1 and case 2 might seem different, they show the same behaviour. Starting at the equator and ending on the north pole, both graphs show that the CPU time of the rotating and the non-rotating case converge until they are practically the same on the north pole. This clearly shows the effect of the rotating Mars on both cases, because this effect becomes smaller with an increase in latitude. The difference in trend is explained by the different ascent trajectories. Case 1 is tuned for a north-east trajectory and case 2 is tuned for a south-south-west trajectory. Figure 8.17 shows the radius for each of the launch latitudes for both cases. Again, as expected the launch latitude does not have an effect on the non-rotating runs, however it does seem to have a large effect on the rotating runs. (a) (b) Figure 8.16: Launch latitude versus CPU time for rotating and non-rotating Mars (a) case 1, (b) case 2. (a) (b) Figure 8.17: Launch latitude versus orbital radius for rotating and non-rotating Mars (a) case 1, (b) case 2. Looking at case 1, the end radius shows a sinusoidal effect with the highest end radius at an initial latitude of 20 degrees. It is however interesting that case 2 shows an almost mirror effect, where the maximum end radius is reached using the nominal conditions, but still a sinusoidal behaviour can be identified. The min- imum end radius is reached at 50 degrees and the non-rotating end radius is crossed at a launch latitude of approximately 10 degrees. This sinusoidal effect can only be attributed to the rotating effects which are simi- lar for both case 1 and 2. They still look different though, which is due to the fact that both trajectories are very different. Case 1 launches due east and then north and case 2 launches due east and then south, south-west. This means that the trajectories are practically mirrored, and this is also seen in the radius graphs. These effects are also shown in the velocity curves presented in Figure 8.18. Here the ground velocity shows the expected behaviour and a similar curve is also shown by the inertial velocity. However, for case 1 the inertial velocity starts above the non-rotating velocity and then drops below. This transition happens at the maximum radius, after which the curve slowly recovers towards the non-rotating values. At that crossing point, the MAV ends up due west, however, this effect is maximum at an initial latitude of 50 degrees and becomes smaller again as soon as the latitude reaches the north pole. (a) (b) Figure 8.18: Launch latitude versus velocity for rotating and non-rotating Mars (a) case 1, (b) case 2. 8.6. LAUNCH LONGITUDE For the launch longitude, the cut-off criterion was an end time of 789 seconds for case 1 and 873 seconds for case 2. The longitude (τ) has been varied from 0 to 330 degrees. Both the kinematic equations described in Equation (5.66) and the dynamic equations described in Equation (5.67) do not depend on the longitude. This means that no differences are expected between the different longitudes. In Figure 8.19 the launch longitude is plotted against the CPU time, and even though no two longitudes have the same CPU time there is still no significant difference, because the differences in CPU time can be ex- plained by the arguments presented in Section 8.3. (a) (b) Figure 8.19: Launch longitude versus CPU time for rotating and non-rotating Mars (a) case 1, (b) case 2. It is, however, more interesting to look at the differences in radius and velocity as shown in Figures 8.20 and 8.21 respectively. Here it can be seen that for both the non-rotating as well as the rotating runs there are no noticeable differences for the end radius and end velocity of case 1 (order of 1 m difference). But case 2 does indeed show a difference in both graphs. As a matter of fact depending on the chosen launch longi- tude, the difference in end radius could be as large as 1 km. This difference cannot be explained at this point because there should not be a noticeable difference besides rounding error, since a 1 km difference is rather large for a rounding error. 8.7. FLIGHT-PATH ANGLE (a) (b) Figure 8.20: Launch longitude versus orbital radius for rotating and non-rotating Mars (a) case 1, (b) case 2. (a) (b) Figure 8.21: Launch longitude versus velocity for rotating and non-rotating Mars (a) case 1, (b) case 2. The difference in end radius follows from a difference in velocity. Because the only difference between case 1 and case 2 is the input, these differences can be attributed to rounding errors. It seems that different combi- nations of input values could create cases where the rounding errors can have a large effect. It is interesting to note that RKF showed the same deviations. 8.7. FLIGHT-PATH ANGLE Previously, the cut-off criterion was always set to be a certain end time. However, for the Flight-Path Angle (FPA) it is better to set the cut-off criterion as the point when the simulation reaches the zero degree FPA. This means that not only the final altitude will be different but also the simulation run time. It is expected that an MAV launched at a lower launch FPA will reach the cut-off criterion earlier compared to a high flight-path angle launch. A lower simulation time will equal fewer function evaluations. This is shown in Figure 8.22, where it is clear that for both the rotating and the non-rotating Mars

60**runs the number of function evaluations**

decrease with a decrease in FPA for both cases. It also shows that the

60**number of function evaluations does not differ much** compared to **the**

non-rotating runs, but the two curves do seem to diverge slightly with a higher FPA. This is expected since the simulation run time is higher as well. This divergent behaviour can also be spotted in Figure 8.23, where the curve of the non-rotating runs is always below the rotating runs. These graphs also show

97**that there is a** clear relationship **between** the **number of function evaluations and**

the 8.7. FLIGHT-PATH ANGLE required CPU time. (a) (b) Figure 8.22: Initial flight-path angle versus Function Evaluations for rotating and non-rotating Mars (a) case 1, (b) case 2. (a) (b) Figure 8.23: Function evaluations versus CPU time for rotating and non-rotating Mars (a) case 1, (b) case 2. (a) (b) Figure 8.24: Initial flight-path angle versus end height from launch site for rotating and non-rotating Mars (a) case 1, (b) case 2. As was mentioned in the introduction of this section, the MAV launching at a lower FPA is expected to reach a lower altitude because the zero degree cut-off criterion is reached earlier in time. This is precisely what is shown in Figure 8.24 where the lower FPA only reach an effective altitude of a few metres, whereas the higher FPAs are able to reached an altitude of a few hundred kilometres. This is also due to the fact that besides the FPA nothing else changed in between runs, which in turn means that every run uses the same thrust profile. The nominal thrust profile was set for a FPA of 89 degrees. So if the MAV is set at a much shallower angle at launch it means that this thrust is directed in a more horizontal direction (in comparison) and will therefore have a much larger effect on the horizontal flight profile. Because each of the runs is cut-off at the zero FPA, it is interesting to look at the required propellant mass to reach a circular orbit with the desired inclination at that particular point. Please note that these circularisa- tion burns are assumed to be impulsive and thus only results in an approximation of the required propellant mass. In Figure 8.25 the non-rotating Mars curve is situated above the rotating Mars curve. This indicates that, even though it might not be much, the non-rotating Mars case requires more propellant mass to circu- larise the orbit due to the fact that it is lacking the initial boost of the rotating Mars. (a) (b) Figure 8.25: Initial flight-path angle versus propellant mass required for circularisation and desired inclination change for rotating and non-rotating Mars (a) case 1, (b) case 2. For case 1 the FPA range was from 86.5 till 89.5 degrees, and for case 2 the range was from 83 till 89 degrees. For both cases the minimum required propellant mass to circularise is at a FPA of 89 degrees, which is exactly the nominal case for which the thrust profile was calibrated. It should also be said that the

38**atmospheric drag has** not **been taken into account** for **the**

calculation of the required propellant mass for the circularisation, and it is also treated as an impulsive shot, meaning that all the propellant mass would have to be expelled at once. This is therefore purely theoretical. But it does show the large effect that the launch FPA has on the trajectory. 8.8. HEADING ANGLE The final variable that is investigated is the launch heading angle. This angle defines the direction in which the MAV is pointed at launch. In this thesis research the heading angle is defined to be

55**0 degrees** facing **north, 90 degrees due east, 180 degrees** pointed **south and 270 degrees due west.** Again **the**

cut-off criterion for this angle has been set at a FPA of 0 degrees. Thus the end time and altitude are again left free. This means that the number of function evaluations will also be different as shown in Figure 8.26. (a) (b) Figure 8.26: Heading angle versus Function Evaluations for rotating and non-rotating Mars (a) case 1, (b) case 2. For case 2 it appears that the heading angle does not have any influence on the number of function evalu- ations required for the non-rotating runs. This is expected since the rotation of Mars is not included which negates any arbitrary rotational influences. Therefore, the zero degree FPA will be reached at the same sim- ulation time. However, case 1 shows that sometimes the number of function evaluations is 46 (between 300 and 0 degrees) and sometimes it is 45. That could only happen if the values are close to the boundary values of the allowed precision which determines the step-size. Therefore, an end time difference of 1 µs can result in having to perform one more simulation step. For the rotating cases, the effect of the heading angle can clearly be seen, and are almost opposite between the two cases. This again has to do with the programmed trajectory heading. Case 1 has to go north-east which means that a heading angle in the opposite direction will require extra steps to get the MAV in the right direction, whereas a heading angle that is more directly pointed to the south pole is beneficial for case 2 and thus decreasing the number of evaluations required. In Figure 8.27 the function evaluations are again plotted against the CPU time and here the same trend can be observed as was seen for the FPA results. (a) (b) Figure 8.27: Function evaluations versus CPU time for rotating and non-rotating Mars (a) case 1, (b) case 2. (a) (b) Figure 8.28: Heading angle versus altitude with respect to the MOLA reference for rotating and non-rotating Mars (a) case 1, (b) case 2. Because only the heading angle changes but the rest of the variables remain the same, it would be expected that the reached altitude of the MAV would show a sinusoidal behaviour for both cases. When inspecting Figure 8.28 this behaviour is indeed present in both cases. In fact, the curve of case 1 could effectively be moved as to create the curve at case 2. The non-rotating curves are both crossed due to the fact that the trajectory for both cases result in a flight path that heads due west in the end. It is interesting to note that the maximum altitude that can be reached lies higher in both cases than the nominal case of 90 degrees, but this would be accompanied by a low end velocity. (a) (b) Figure 8.29: Heading angle versus propellant mass required for circularisation and desired inclination change for rotating and non-rotating Mars (a) case 1, (b) case 2. The reached altitude might be higher, but the propellant mass required to reach the desired orbit inclination and circularise the orbit at that altitude might be very high as well. In fact, Figure 8.29 shows that there are only two local minima when it comes to propellant use. And one of these local minima is the global minimum at 90 degrees for which the nominal case was adjusted. These second minima were however a bit unexpected at first. For the first case it lies at 180 degrees and the second case at 270 degrees. When taking a closer look it becomes clear that these are the cases where the MAV is able to get into the same orbit but then passing over the equator from north to south in case 1 and from south to north in case 2. Effectively thus not requiring much energy for the change in inclination. This same behaviour can be seen in the inertial velocity curves for both cases shown in Figure 8.30. (a) (b) Figure 8.30: Heading angle versus velocity for rotating and non-rotating Mars (a) case 1, (b) case 2. Here the ground velocity and non-rotating velocities are an exact mirror of the radius curves and the inertial velocity curve shows two maxima. These velocity maxima do not however correspond exactly to the opti- mal propellant masses found. At this point, it is not entirely clear why this is the case, however, a possible explanation could be that the inclination change also requires some propellant. 9 CONCLUSIONS AND RECOMMENDATIONS During the course of this research, many different aspects of TSI have been investigated. And with each phase new conclusions were drawn, even during the programming and setting up the integrator. All these conclusions are described in Section 9.1. However, with each conclusion also came a sense of what could still be improved and added. All the recommendations that came out of that process are described in Section 9.2. 9.1. CONCLUSIONS The conclusions can be divided up into three different parts. The first part are the conclusions drawn as the TSI method was worked out and programmed. During this phase, some of the advantages and disadvantages of this method became clear and are discussed in Section 9.1.1. Then one of the main aspects of this research is described in Section 9.1.2, where the conclusions are presented with respect to the comparison between TSI and RKF. And finally, a sensitivity analysis was performed on TSI using some of the different variables that the program requires as input. The conclusions from this analysis are presented in Section 9.1.3. 9.1.1. TAYLOR SERIES AS AN INTEGRATOR Setting up the TSI can be difficult, because it requires all the written-out equations for the model. These include the equations to simulate the atmosphere and equations to compute the drag coefficient. All the equations involved also require the first derivatives, and each of these derivatives has to be divided up into auxiliary functions. In this process of setting up all these equations, it is really easy to make mistakes. In this thesis, the basic recurrence relations were simply

programmed into a single class such that each of the recur- rence relations for the auxiliary derivatives and functions could call those basic functions. Even with these simplifications, it can take a long time to set-up this method. Because the auxiliary derivatives and functions are intertwined, adding an extra perturbation for example could be as simple as adding a few more equations, or as difficult as rewriting some of these equations. In this research, the auxiliary functions were determined per derivative and often an already determine auxiliary function could be used in another auxiliary derivative. However, it another method that might work better is to first determine the common derivative parts in each of the auxiliary derivatives, then set-up the auxiliary functions for those parts and use those to determine the rest of the auxiliary functions. This was not done in this simulation program though. It is therefore easier to include as many perturbations as possible when designing the integrator such that during the actual simula- tion one can simply specify the desired perturbations that have to be taken into account. This would make the program itself less problem specific. Once the program was all set-up, a step-size handling procedure had to be included to deal with the differ- ent sections in the temperature and drag coefficient model. This was needed to make sure that TSI did not continue into the next section without changing the section properties. The step-size handling procedure was easily implemented and is able to determine when a next section is reached and change the

> 3**step-size such that the** previous **step ends at the** start of **the next**

section. The atmospheric and drag coefficient models were set-up at the start of the thesis research and at that time a certain number of sections were fitted as to get the best representation of the corresponding model. However, during the development of TSI it became clear that it would be better to have a model with fewer sections. The reason for having more than 1 section 81 9.1. CONCLUSIONS was because this limited the order of the polynomial that had to be fitted. But every time a new section is reached, TSI has to limit a step-size to pass over that boundary. Therefore it would be better to have high- order polynomial fits and fewer sections. Fortunately, this model change can effortlessly be implemented in the simulation program by simply updating the polynomial coefficient matrices, which are an input of the integrator. Because TSI has this ability to readily update the step-size to meet a certain section boundary condition, it means that it can also freely cope with discontinuities. Another advantage is that TSI can theoretically have any desired order. This theoretically unlimited order is however limited by the computer accuracy, because at a certain point the higher orders disappear into the rounding errors. And finally, the Taylor Series coefficients are computed for each time step after which a corresponding step-size is computed to stay within a certain error tolerance. The coefficients are then multiplied by the step-size as per Equation (5.102). This step-size is the maximum step-size that can be used to determine the next state while still staying within the error tolerance, however if a smaller step-size is chosen, it means that any value between the current time step and the next time step can be computed as well (meeting an even higher error tolerance). The current program has the option to save all the Taylor Series coefficients for each time step, which means that even after the simulation has been performed extra data points can be created in between the already chosen time steps by simply choosing a smaller time step as mentioned before. Interpolation is therefore not required and the data can have a theoretically unlimited resolution, which is a major advantage over conventional integrators. 9.1.2. TSI COMPARED TO RKF7(8) The main question that had to be answered in this research was: "Does TSI show similar performance im- provements over RKF for an ascent case as was shown for orbital trajectories and (re-)entry cases?". There are a few different aspects that were compared between TSI and RKF7(8). First, the number of function calls were compared. In this thesis research, a function call is the process of going through the set of equations once. For RKF7(8) this has to be done 13 times in one time step, whereas it only has to be done once per time step for TSI. The effect of this difference was made very clear when both TSI and RKF were run for a number of different error tolerances. Each error tolerance resulted in a different number of time steps in order to stay within the error bounds. For all tested error tolerances, TSI required far fewer function calls than RKF. Not only that, but as the error tolerance became smaller, the number of function calls required for RKF increased rather. This resulted in a maximum number of evaluations required for RKF to be 7 to 8 times more than for the largest error tolerance. This compared to TSI where the difference in number of evaluations between the largest and the smallest error tolerance was less than twice the lowest number of evaluations. The second aspect that was compared was the accuracy. Again using the error tolerance, the accuracy of both methods at different error tolerances could be determined.

For TSI the accuracy of the position and velocity at an error tolerance of 10−5 was 1 cm and 10 μm respectively, whereas RKF was only able to reach these accuracies at an error tolerance of 10−9. This clearly shows that TSI is inherently more accurate even at larger error tolerances. Besides being more accurate, TSI also showed a faster convergence speed compared to RKF, which meant that the smallest tolerances did not show much of a difference. Finally, the CPU time of both methods was compared as well. CPU time is measured by the time it takes the CPU to run through all the computations required for the simulation. In all tested cases TSI turned out to be an entire order slower, on average, than RKF. This is completely opposite to what was expected based on previous research. The RKF7(8) method was provided by the Tudat library and only required a single state derivative file in order to work. TSI on the other hand was programmed in such a manner that is was comprehensible, but in doing so many different program files were created that all had to be called during the integration. Each of those function files contains computations that could be run at the same time, but instead they were run consecutively. Therefore, the CPU time could be decreased by using different threads. One example where this could

> 31**have a** large **impact** is **the** computation **of the Taylor Series**

coefficients. Not all the coefficients are interdependent and could therefore be run in separate threads at the same time. The actual improvement on CPU time that this threading method would have is not yet clear, but is definitely something that should be investigated. 9.1.3. SENSITIVITY ANALYSIS To be able to understand the behaviour of TSI better and to determine the influence of different parame- ters on the ascent trajectory, a sensitivity analysis was performed. In this section, the conclusions will be 9.1. CONCLUSIONS presented per parameter starting with the TSI order. ORDER During the different order runs for both cases it was determined that an order of 20 resulted in the lowest CPU time on average. There were also two different phases identified; one phase before the lowest CPU time showing a rapid decrease and one phase after this lowest CPU time order showing a steady increase. In the first phase, the CPU time was greatly influenced by the high number of time steps that were needed, because of the inaccuracies introduced by the lower orders. The second phase showed that the CPU time was now influenced more by the number of recurrence-relation computations that were introduced with each extra order. Also, the number of time steps decreased only a little between the first order of that phase and the final order of that phase and thus reduced the effect

> 47**of the number of time steps** on **the** CPU **time.**

Finally, it was also shown that from an order of 11 the accuracy of the method did not change that much any more compared to the nominal RKF run. But it still showed the best correlation with the nominal RKF run at an order 20 and since order 20 was also faster, it was better to run the simulations at that order. ERROR TOLERANCE The number of evaluations for TSI varied very little between the different error tolerances. In both cases an error tolerance of 10−5 resulted in 24 evaluations (in that particular simulation run) and an error tolerance of 10−15 resulted in 46 and 47 evaluations for case 1 and 2 respectively. On average this was an increase of 2.2 and 2.3 evaluations per error tolerance. Because the error tolerance has a direct correlation with the number of time steps, it also has a direct influence on the CPU time. An increase in error tolerance, therefore, resulted in an increase in CPU time. Also, the non-rotating Mars runs were faster compared to the rotating Mars runs, which was to be expected since fewer equations had to be performed. It was also found that a lower tolerance resulted in a less accurate final result compared to the higher tol- erance results, which makes sense since a lower error tolerance means that a larger error is accepted making the end results less accurate. MULTIPLE RUNS The multiple nominal runs showed the influence of computer processes on the CPU time of the integration run. This influence of processes running on the background sometimes resulted in a difference in CPU time of 30% or more, even though the exact same case was run. This influence was best shown during a run of the non-rotating Mars scenario where a sudden spike appeared in the CPU data, indicating that another process was active at that time in the background. This does not explain the difference between the RKF and TSI CPU times, but it does show that CPU time is not always a good comparison parameter. LAUNCH ALTITUDE Keeping in mind that a certain variation in CPU times is allowed due to the behaviour observed during the multiple runs, no direct influences of launch altitude on the CPU time could be identified. The

launch altitude did, however, have a direct influence on both the final radius as well as the final velocity of the MAV. An increase in launch altitude resulted in an increase in final radius and a decrease in the final velocity. LAUNCH LATITUDE The CPU time of the launch-latitude runs showed an undeniable influence of the different latitudes. Starting at the equator and ending up at the north pole, the CPU time showed a convergence behaviour of the rotating and the non-rotating Mars curves. Therefore, the closer the MAV launches to the north pole, the smaller the influence of the rotation. The chosen launch latitude also has a large effect on the final radius and velocity due to the rotation of Mars. LAUNCH LONGITUDE Compared to the launch latitude, the launch longitude has virtually no influence at all on the final radius and velocity. This is due to the fact that the rotational influences do not change in the longitudinal direction. 9.2. RECOMMENDATIONS FLIGHT-PATH ANGLE When running the different FPA cases it became clear that the FPA has a large influence on the flight trajectory of the MAV. For these runs the cut-off criterion was a zero degree FPA. With a decrease in launch FPA, that cut-off criterion was reached earlier in flight each time. Up until the point where a difference of 2 degrees could mean reaching the desired orbit or not even reaching an altitude of 1 km. For these runs, the orbit was also circularised (which included an inclination change to the desired inclination) once the cut-off criterion was met. This showed a massive difference in the required propellant mass, because the required orbital velocity became more and more difficult to reach the lower the launch FPA. And because the flight was cut- off earlier, the MAV did not have enough time to change its heading to match the desired inclination which also added to the required propellant mass. HEADING ANGLE Because the other parameters were all left constant in between the launch heading angle runs, a sinusoidal behaviour was observed in the altitude versus launch heading angle curve. And it was interesting to see that for both case 1 and 2 the maximum reachable altitude with those parameters was not at the nominal 90 degree angle. However, these maximum reachable altitudes also required a lot of propellant mass to get into the desired orbit with the desired inclination. Since the nominal case was calibrated for the desired orbit it was expected to require the least amount of propellant mass to reach that orbit, and this indeed turned out to be the case. However, another local minimum propellant option was identified for both case 1 and case 2. They required slightly more propellant but were still a convincing option. These two local minima are the cases where the exact same orbit is reached but then moving in the opposite direction over the equator compared to the nominal trajectory. It was interesting to see that a change in launch heading angle could identify this second local minimum, which also clearly shows that the launch heading angle has a large effect on the launch trajectory. 9.2. RECOMMENDATIONS The recommendations are split into two categories: recommendations to improve the current TSI program (Section 9.2.1) and recommended tests and additions that could be made (Section 9.2.2). The improvements to the current TSI program are needed to try to reduce the CPU time and see if the same performance can be achieved as shown by Bergsma & Mooij (2016); Scott & Martini (2008) and also to try and make the TSI program more robust. The extra tests and additions are either planned processes that could not be executed in the scope of this thesis or processes that were not originally planned but would still be interesting to inves- tigate. 9.2.1. IMPROVEMENTS TO THE CURRENT TSI PROGRAM The list presented in this section suggests some steps that could be taken to maybe improve the performance of the current TSI program. Use threads to perform simultaneous computations The current program does not use any threads at all, instead the computations are all run one after another. However, there are many computations that could be performed in parallel because they are independent. Using threads to perform these simulta- neous computations would theoretically reduce the CPU time. Get rid of obsolete computations and functions Even though most of the obsolete computations and func- tions were removed during the many rewrites of the program, it could be that some of the old code is still present but not used any more. Removing the obsolete code could theoretically reduce the CPU time. Get rid of the separate basic recurrence relations file In the current program, the basic recurrence relations are functions that are written in a separate file and called by the final recurrence relations file. In the reference research, the basic recurrence relations are always described completely worked out in the final recurrence relations. The improvements on the program by getting rid of the basic recurrence relations file and writing them out in the final recurrence file are yet unknown. Use a different root-finding method The current root-finding method was based on the root-finding method used by Bergsma & Mooij (2016), however it could be that another method would work better for the tested scenarios. 9.2. RECOMMENDATIONS Implement a different step-size determination method The step-size determination method worked well for the current program, but it showed some divergence issues when directly applied to the starting conditions. This is why an initial second was integrated using RKF7(8). A different step-size determina- tion method might be able to better estimate the required step-size, although it is unlikely to completely get rid of the early divergence issues in the

Taylor Series coefficients. Use a temperature model with fewer sections but higher-order polynomial fits Because TSI had to stop with the current time step and adjust the step-size accordingly every time a new section was reached, it would be better to use a temperature model with fewer sections. This does mean that function fits with higher-order polynomials are required, but TSI does not seem to have an issue with those. The same goes for the drag coefficient model. Use the coefficients of the previous step to determine the starting xn, un and wn of the next step Currently, only the state is a direct output of one single integration step. However, the Taylor Series coefficients for all the auxiliary equations, derivatives and functions are available. If these are expanded through a Taylor Series, an estimate for all those variables would be directly available at the next integration step and the step of recomputing the auxiliary equations, derivatives and functions through the state could then be skipped. This could reduce the required CPU time. 9.2.2. EXTRA TESTS AND ADDITIONS The tests and additions to the program listed here are suggestions based on the experience gathered during this research and academic interest. Add optimisation Optimisation is something that still needs to be implemented and tested. Use can be made of the Parallel Global Multi-objective Optimizer (PaGMO) library in combination with the Sparse Nonlinear Optimizer (SNOPT) method. Unfortunately, this could not be done during this research. Add extra perturbing accelerations Even though other perturbing effects such as higher-order gravity terms or third-body perturbations could be neglected, it would still be interesting to see what the influence on the trajectory would be. Test the influence of the drag, thrust and gravitational acceleration A more extensive sensitivity analysis could be performed where different scenarios could be tested that include a thrust profile instead of a constant nominal thrust, the actual influence of the drag on the trajectory and what would happen for instance if the gravitational acceleration would be different. Include the thrust angle profile For the complete trajectory analysis and comparison to the numerical anal- ysis currently being performed at JPL it would be interesting to include the thrust angle profile. This could be done in combination with the optimisation to determine the required thrust profile for the optimum trajectory. BIBLIOGRAPHY Akcasu, O. E. & Akcay, I. (2013). Solar time using gps technology. Lonestar Inventions, L.P., Austin, TX, U.S. Patent. Patent application, Docket No. WO 2013070518 A1, filed 2 Nov. 2012.

7Archinal, B. A., Ahearn, M., Bowell, E., Conrad, A., Consolmagno, G. J., Courtin, R., Fukushima, T., Hestroffer, D., Hilton, J. L., Krasinsky, G. A., et al. (2011). Report of the IAU working group on cartographic coordinates and rotational elements: 2009. In Celestial Mechanics and Dynamical Astronomy,

volume 109 (pp. 101– 135). Bachman, N. (2014). NAIF SPICE toolkit hypertext documentation, required reading documents, reference frames. website. [online database], URL: http://naif.jpl.nasa.gov/pub/naif/ [cited 18 December 2015]. Benito, J. & Johnson, B. J. (2016). Trajectory Optimization for a Mars Ascent Vehicle. In AIAA/AAS Astrody- namics Specialist Conference. Bergsma, M. C. W. (2015).

3Application of Taylor Series Integration to Reentry Problems.

Master's thesis, Delft University of Technology. Bergsma, M. C. W. & Mooij, E. (2016).

3Application of Taylor-Series Integration to Reentry Problems

with Wind. In Journal of Guidance, Control, and Dynamics (pp. 2324–2335).: AIAA.

44Berry, M. M. (2004). A Variable-Step Double-Integration Multi-Step Integrator. PhD thesis, Virginia Polytech- nic Institute and State University.

del Rio, J. D. (2008). Generic frame definition Kernel file for ESA planetary missions. [online database], URL:

90http://naif.jpl.nasa.gov/pub/naif/ VEX /kernels/fk/ RSSD0002 .TF

[cited 8 December 2015].

27Desai, P., Braun, R., Engelund, W., Cheatwood, F., & Kangas, J. (1998). Mars Ascent Vehicle Flight Analysis. In 7th AIAA/ASME Joint Thermophysics and Heat Transfer Conference.

Di Sotto, E., Bastante, J. C., & Drai, R. (2007).

24System and GNC concept for RendezVous into elliptical Orbit for Mars Sample Return mission. In AIAA Guidance, Navigation, and Control Conference and Exhibit, Colorado.

Dirkx, D., Kumar, K., Doornbos, E., Mooij, E., & Noomen, R. (2016). TUDAT. [online database], URL: tudat.tudelft.nl [cited 8 March 2016].

39Dormand, J., El-Mikkawy, M., & Prince, P. (1987). Families of Runge-Kutta-Nystrom formulae. In IMA Journal of Numerical Analysis,

volume 7 (pp. 235–250). Dumont, E.

59(2015). Design of a Modular Transportation System for Future Lunar Robotic Missions. In 30th ISTS/

4834th IEPC/6th NSAT Joint Conference, Kobe, Japan.

Duxbury, T.,

10Christensen, P., Smith, D., Neumann, G., Kirk, R., Caplinger, M., Albee, A.,

Seregina, N., Neukum, G., & Archinal, B.

10(2014). The location of Airy-0, the Mars prime meridian reference, from stereo pho- togrammetric processing of THEMIS IR imaging and digital elevation data. In Journal of Geophysical Re- search: Planets,

volume 119 (pp. 2471–2486).: John Wiley & Sons. Fanning, J. & Pierson, B. (1996).

9A model comparison for optimal Mars ascent trajectories. In Engineering Optimization, volume 26 (pp. 271–285).

34Fehlberg, E. (1968). Classical fifth-, sixth-, seventh-, and eighth-order Runge-Kutta formulas with stepsize control. Technical report, Marschall **Space Flight Center,**

NASA. :

53NASA TR R-287. Fehlberg, E. (1969). Low order classical Runge-Kutta formulas with stepwise control.

Technical report, Marschall Space Flight Center, NASA. : NASA TR R-316. 86 BIBLIOGRAPHY

64Guernsey, C. S. (1998). **Mars Ascent Propulsion System (MAPS) Technology Program: Plans and Progress.** In 34th **AIAA/**

ASME/SAE/ASEE Joint Propulsion Conference and Exhibit.

33Ho, C., Golshan, N., & Kliore, A. (2002). Radio Wave Propagation Handbook for Communication **on and Around Mars.** NASA **Jet Propulsion Laboratory,**

Pasadena, CA, 1 edition. handbook JPL Publication 02- 5. Hofsteenge, R. (2013).

52Computational Methods for the Long-Term Propagation of Space Debris Orbits. Mas- ter's **thesis, Delft University of Technology.**

16Jorba, A. & Zou, M. (2005). A Software Package for the Numerical Integration of ODEs by Means of High-Order Taylor Methods. In **Experimental Mathematics,** volume **14 (pp. 99–117).:**

Taylor & Francis.

14Justus, C. (1990). **A Mars Global Reference Atmospheric Model (Mars-GRAM) for Mission Planning and**

Anal- ysis. In

1428th Aerospace Sciences Meeting, Reno, Nevada.

18Karp, A. C., Redmond, M., Nakazono, B., Vaughan, D., Shotwell, R., Story, G., **Jackson, D., & Young, D.** (2016). **Technology development and design of a hybrid Mars ascent vehicle concept.**

48In Aerospace Conference, 2016 IEEE (pp. 1–10).: IEEE.

9Mars Program Planning Group (2012). Summary of the Final Report. [online database], URL: http://www.nasa.gov//sites/default/files/files/ [cited 14 October 2015].

Meijaard, J. (1991).

21A Comparison of Numerical Integration Methods with a View to Fast Simulation of Me- chanical Dynamical Systems. In Real-Time Integration Methods for Mechanical System Simulation,

vol- ume 69 (pp. 329–343).: Springer.

71Milani, A. & Nobili, A. M. (1987). Integration error over very long time spans. In

Celestial Mechanics, volume 43 (pp. 1–34).: Springer.

43Montenbruck, O. (1992a). Numerical integration methods for orbital motion. In Celestial Mechanics and Dynamical Astronomy, volume 53 (pp. 59–69).

31Montenbruck, O. (1992b). Numerical integration of orbital motion using Taylor series. In Spaceflight

mechan- ics (pp. 1217–1231).

61Mooij, E. (1994). The motion of a vehicle in a planetary atmosphere. Delft: Delft University Press. Mooij, E.

(2013). AE3202-week1-lecture1b-Frames. Lecture slides, Delft University of Technology. Course: Flight Dynamics.

58Morton, O. (2003). Mapping Mars: Science, Imagination, and the Birth of a World. Picador, New York,

1 edition. avaiable at JPL library.

28Mulder, J., van Staveren, W., van der Vaart, J., de Weerdt, E., in 't Veld, A., & Mooij, E. (2013). Flight Dynamics, Lecture Notes. Reader, Delft University of

Technology. Course: Flight Dynamics AE3202. Mungas, G., Fisher, D., Vozoff, J., & Villa, M. (2012). NOFBX

73 **Single Stage to Orbit Mars Ascent Vehicle. In Aerospace Conference,** 2012 IEEE (pp. **1–11).: IEEE.**

Noomen, R. (2013a). ae4-878.basics.v4-14. Lecture slides, Delft University of Technology. Course: Mission Geometry and Orbit Design. Noomen, R. (2013b).

56 **ae4-878.** integrators **.v4-** 3. **Lecture slides, Delft University of Technology. [Internal** pub- lication] **Course: Mission Geometry and Orbit Design.**

Prince, P. & Dormand, J.

51 **(1981). High order embedded Runge-Kutta formulae.** In **Journal of Computational and Applied Mathematics,** volume 7 **(pp. 67–75).:**

Elsevier.

72 **Ramos, H. & Vigo-Aguiar, J. (2005). Variable stepsize Störmer-Cowell methods.** In **Mathematical and**

Com- puter Modelling, volume 42 (pp. 837–846). Scott, J. R. & Martini, M. C. (2008).

35 **High speed solution of spacecraft trajectory problems using Taylor series integration. In AIAA/AAS Astrodynamics Specialist Conference and Exhibit,**

Honolulu, Hawaii. BIBLIOGRAPHY Sengupta, A., Kennett, A., Pauken, M., Trinidad, M., & Zabrensky, E. (2012). Systems Engineering and Tech- nology Considerations of a Mars Ascent Vehicle. In 2011 IEEE Aerospace Conference. Shotwell, R. (2016). History of Mars Ascent Vehicle development over the last 20 years. In Aerospace Confer- ence, 2016 IEEE (pp. 1–11).: IEEE. Shotwell, R., Benito, J., Karp, A., & Dankanich, J. (2016). Drivers, developments and options under considera- tion for a Mars ascent vehicle. In Aerospace Conference, 2016 IEEE (pp. 1–14).: IEEE. Stephenson, D. D. (2002). Mars ascent vehicle - Concept development. In 38th Joint Propulsion Conference and Exhibit, Indianapolos, Indiana, volume 4318: AIAA. Stephenson, D. D. & Willenberg, H. J. (2006). Mars ascent vehicle key elements of a Mars Sample Return mission. In Aerospace Conference, 2006 IEEE (pp.1˜1).: IEEE. Trinidad, M., Zabrensky, E., & Sengupta, A. (2012). Mars Ascent Vehicle system studies and baseline concep- tual design. In Aerospace Conference, 2012 IEEE (pp. 1–13).: IEEE. Vaughan, D., Nakazono, B., Karp, A., Shotwell, R., London, A., Mehra, A., & Mechentel, F. (2016). Technology development and design of liquid bi-propellant mars ascent vehicles. In Aerospace Conference, 2016 IEEE (pp. 1–12).: IEEE. Wakker, K. (2010a). Astrodynamics-I Course AE4874, Part 1. Reader, Delft University of Technology. [Internal publication] Course: Astrodynamics I. Wakker, K. (2010b). Astrodynamics-II Course AE4874, part 2. Reader, Delft University of Technology. [Internal publication] Course: Astrodynamics-II. Weeks, M. & Thrasher, S. (2007). Comparison of Fixed and Variable Time Step Trajectory Integration Methods for Cislunar Trajectories. In AAS/AIAA Space Flight Mechanics Meeting. Whitehead, J. (1997). Mars ascent propulsion options for small sample return vehicles. In 33rd AIAA/AS- ME/SAE/ASEE Joint Propulsion Conference and Exhibit: AIAA. Whitehead, J. C. (2004). Mars Ascent Propulsion Trades with Trajectory Analysis. In 40th AIAA/AS- ME/SAE/ASEE Joint Propulsion Conference and Exhibit: American Institude of Aeronautics and Astronau- tics. Whitehead, J. C. (2005). Trajectory analysis and staging trades for smaller Mars ascent vehicles. In Journal of spacecraft and rockets, volume 42 (pp. 1039–1046). Williams, D. (2015). Mars Fact Sheet. website. [online database], URL: http://nssdc.gsfc.nasa.gov/planetary/ factsheet/marsfact.html [cited 1 November 2015]. Woolley, R. C. (2015). A simple analytic model for

estimating Mars Ascent Vehicle mass and performance. In 2015 IEEE Aerospace Conference. Woolley, R. C., Mattingly, R. L., Riedel, J. E., & Sturm, E. J. (2011). Mars Sample Return - Launch and Detection Strategies for Orbital Rendezvous. In AAS/AIAA Astrodynamics Specialist Conference, volume 142: Univelt, Inc. A REFERENCE SYSTEM TRANSFORMATIONS When considering the position, velocity and acceleration of a space vehicle, different reference frames are usually used. Why there are different frames, which frames are required, what they are used for and how to transfer from one frame to the other will be discussed in Appendices A.1 and A.2. In case of, for instance, a car travelling on the road, a certain coordinate system has to be used to measure the velocity in a certain direction. For the car it can be said that a cartesian system is used with x in the direction of motion, z pointing towards the ground and y then pointing to the right if looking in the direction of motion (using the so-called RHR to complete the frame). The speed is then measured and expressed in the x-direction. This works well when a vehicle is travelling in a straight line or on a flat plane, but in orbital mechanics the motion usually has to be described just above or in an orbit around a (spherical) body. Therefore it is important to know which coordinate system is used and how to change between these coordinate systems should the other system become more convenient to use at a certain point. In Appendix A.3 the difference between these coordinate systems and how to transfer from one to the other is explained. A.1. REQUIRED REFERENCE SYSTEMS In the example of the car, the Frame of Reference (RF) is fixed to the Earth. However, when considering two cars driving on the same road, one might like to investigate the difference in velocity between both vehicles. In that case an RF is chosen that is fixed to one of the two cars. This makes it easier to determine the relative velocity of one car with respect to the other. The same can be done for s/c (think of formation flying). There are therefore a number of different RFs that can be used. In this section the required RFs and their application will be presented. A.1.1. MARS-CENTRED INERTIAL RF (I-FRAME) The Mars-Centred Inertial (MCI) reference frame is an RF that is, as the name implies, fixed to the centre of mass of Mars (Mulder et al., 2013). It is a frame in which Mars still rotates with respect to that frame, or in other words it is an inertial frame with respect to Mars. The x-axis is defined through a certain point which can be set by the user. This means that the user can determine his or her own inertial frame. A frame that is often used (and which is based on J2000[1]) is the MARSIAU frame (Bachman, 2014; del Rio, 2008). This is a Mars inertial frame that is translated from the Earth J2000 frame. For the translation of J2000, a point through the mean equator of Mars is specified where the equator ascends through the mean equator of Earth. The x-axis is defined through this point. The z-axis is then defined through the rotational axis of Mars and the y-axis completes the frame through the RHR. The translation is described by Bachman (2014); del Rio (2008) using the J2000 numbers from Archinal et al. (2011). The generic inertial frame is shown in Figure 3.1a. The centre of the frame is represented by the letter 'A'. The MARSIAU frame is required if certain data from outside sources will have to be incorporated into the simulation, however for the actual simulation it was easier and much more straightforward to define a custom Mars inertial reference frame at the start of the simulation. This reference system is very useful if the motion of the s/c with respect to the ground-track is not important. [1]J2000 is an Earth referenced inertial frame that was set through the rotational axis of Earth (z-axis) and through the vernal equinox (x-axis). The reference was set on the Julian date of 2451545.0 or 01-01-2000 at 12:00 Greenwich Mean Time (GMT). 89 A.2. TRANSFORMATION BETWEEN REFERENCE FRAMES A.1.2. MARS-CENTRED MARS-FIXED RF (R-FRAME) The Mars-Centred Mars-Fixed (MCMF) frame is the same as the MCI frame except for one distinct difference. The x-axis of the MCMF frame (x_R) is defined through the 0.5 km wide Airy-0 crater (Morton, 2003), which has a latitude of 5.07829°S (Duxbury et al., 2014). The associated meridian is called the prime meridian. On the 1st of January 2000 at 12:00 the prime meridian of Mars was 176.630° east of the x-axis in the MARSIAU frame (which is important when transforming from the MARSIAU frame to the current MCMF frame)(Archinal et al., 2011; del Rio, 2008). Because the Airy-0 crater is fixed to the surface of Mars and rotates around the Martian rotational axis (in both MCI and MCMF the z-axis), the MCMF frame is also a rotational frame. This frame rotates with Mars and can therefore be very useful for ground observation purposes. In the case of the MAV it is important to take the rotational velocity of Mars into account, since Mars will be rotating underneath the MAV causing rotational effects. A graphical representation of the MCMF frame is provided in Figure 3.1b. Here, Ωt represents the general rotational velocity of the reference body, which is called Ω_M when talking about the rotational velocity of Mars. An angle ω_P is defined as the relative angle between the prime meridian and the x-axis of the MCI frame at the time that the inertial frame was defined. A.1.3. VERTICAL RF (V-FRAME) This next RF can be used to describe the motion of an s/c in orbit around a planet. The centre of the vertical frame is defined in the centre of gravity of the s/c (Mulder et al., 2013) but is not rotationally fixed to it. This means that the frame can rotate with respect to the s/c itself. This is because the x-axis is

set to point to the north pole of Mars. The z-axis is perpendicular to Mars' surface (or points to the centre of Mars assuming Mars is a sphere) and the y-axis then points due east. The x-y plane is tangent to Mars' geoid. Figure 3.1c shows the vertical frame denoted by the letter 'V' at an arbitrary position around Mars compared to the MCI frame. The centre of the frame is defined by the letter 'G'. The vertical frame relates to the Mars rotational frame through the latitude and longitude. A.1.4. BODY-FIXED RF (B-FRAME) The Body-Fixed (BF) frame is fixed to the s/c body and therefore rotates with the s/c as well. The origin of the BF RF can be defined in the centre of symmetry (or the middle of the s/c). The axis orientation can be chosen to be in any direction. In this case the x-axis is defined through the vehicle-centre line and goes through the "nose" of the s/c, the z-axis is defined towards the orbiting body (when assuming an orbiting vehicle) and the y-axis completes the frame through the RHR. An example of a body frame that can be applied to the MAV is provided in Figure 3.1d. The axes of the body frame are denoted by a 'B' in this report. In the thesis problem the body frame relates directly to the vertical frame through the flight-path angle γ and the heading (or azimuth) angle χ, because it is assumed that the x-axis of the body frame coincides with the velocity vector of the s/c. A.1.5. PROPULSION RF (P-FRAME) The propulsion or thrust frame is a body frame where the x-axis is defined by the thrust vector. The y-axis is defined in the same plane as the x and y-axes of the B-frame. Then the z-axis completes the frame through the RHR as depicted in Figure 3.1e. The B-frame and P-frame are related through the thrust elevation gimbal angle ε_T and the thrust azimuth gimbal angle ψ_T . A.2. TRANSFORMATION BETWEEN REFERENCE FRAMES Because different problems and situations are easier to understand and to formulate in different RFs, it is important to understand how to transform from one RF to the other. In this case only transformations for a set point in time, called static transformations, are required for the described thesis problem. The necessary transformations for this thesis problem are described in this section. For a transformation between two RFs, a correlation has to exist. This correlation (orientation) can be described using a set of angles, called the Euler angles, between the different RFs (Mooij, 2013). In a 3- dimensional space, there are three axes. So to get from one RF to the other, a maximum of 3 rotations (trans-lations/transformations) have to take place; one over each axis. However, often fewer rotations are necessary because some of the axes are already properly aligned. Sometimes two reference frames are not directly re- lated through known angles, which means that other (intermediate) RFs will have to be used. In this case more than 3 rotations are possible, but again a maximum of three rotations are needed to go from one RF to A.2. TRANSFORMATION BETWEEN REFERENCE FRAMES an intermediate RF given the known angles. Since a 3-dimensional space is described, 3-dimensional coordinates have to be translated to the next reference frame. For this, cartesian coordinates are used. A rotation around the x-axis can be described by the change in the y- and z-coordinates over an angle φ in the yz-plane as visualized in Figure 3.2. The rotation around the x-axis follows the RHR of rotation, which states that if the thumb of your right hand is pointed in the axis direction, then the rest of your fingers show the direction of positive orientation. Therefore the described rotation is positive. Should the rotation have been in the other direction (clockwise around the x-axis as seen in Figure 3.2), the angle would have been negative. This is very important in transformations, and its usefulness becomes more obvious when more transformations are performed after each other. But first a system of equations is required to transform from the 'i' frame to the 'j' frame. From Figure 3.2 a relation can be described between the coordinates in the 'i' frame and the 'j' frame in matrix form. This relation (convention as by Mooij (2013)) is shown by Equation (A.1). Here, r is the position vector and Tx(φ) represents the x-axis transformation matrix and is the standard matrix to be used if a transformation around an x-axis is performed. The only parameter that changes is the angle.

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix}_j = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\varphi & \sin\varphi \\ 0 & -\sin\varphi & \cos\varphi \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix}_i \Rightarrow r_j = T_x(\varphi) r_i \quad (A.1)$$

This rotation described by Figure 3.2 and Equation (A.1) can also be described for rotations around the y-axis and the z-axis. The complete set of axis transformation matrices is then presented in Equation (3.1). Using the cosine notation 'c' and the sine notation 's' the transformation matrices will become more com-prehensible. Also, because the matrices are orthogonal, the inverse of the matrix is simply the transpose of the matrix. The inverse of a rotation matrix is the rotation in the other direction. The order, or sequence, in which these transformations are performed is very important. This is because the sequence results in a matrix multiplication equation, and from linear algebra it is known that $A \cdot B \neq B \cdot A$. Therefore, if the sequence of transformations is not correct, the resulting RF will not be the desired RF. Also, as mentioned before, it is not always possible or easy to transform from one RF directly into the other. So usually one of the afore mentioned RFs is used as an intermediate RF. It is therefore possible to transform from the P-frame all the way to the I-frame using all the other mentioned RFs with a maximum of three rotations between each frame. For convenience it has been chosen to adopt the convention mentioned by Mulder et al. (2013) and Mooij (2013) to describe the transformations. Therefore the I-frame, R-frame, V-frame,

B-frame and P-frame will be known as FI , FR, FV , FB and FP respectively when used to indicate the RFs in equations. And because these equations involve matrix multiplications, they have to be executed from right to left. Therefore it can be said that the transformation matrix to go from FB → FR is TRB = TRVTVB. This is one of the transformations required for the MAV trajectory. The other two transformations are more straightforward: FP → FB is TBP and FR → FI is TIR. For the TIR matrix , a transition is performed from FR → FI . This is done over the angle ΩM tO (or Ωt tO in the general case as defined in Figure 3.1b). ΩM is the rotational velocity of Mars and tO is the time from the set inertial frame. This is also where ωP is required (not depicted) since this angle describes the position of the prime meridian at the time that the inertial frame was set. Because the angle is defined east of the xI -axis, it has to be subtracted from ΩM tO since ωP is defined positive in the same direction as ΩM tO which is defined to be positive going from xI to xR. The rotation around the z-axis is positive in the same direction according to the RHR for rotation. In this case the rotation is performed in the negative direction around the z-axis with an angle −ΩM tO + ωP . The complete RF transformation with the written out matrix for this rotation is given by Equation (A.2). Please note that the signs of the angles changed in the matrix due to the sine and cosine functions. In Equation (A.2) the vertical lines with the letter designations depict when a certain reference frame is reached. The transformations for TRB and TBP can now be written in a similar manner and are described in Equations (A.3) and (A.4) respectively. ⌈c (ΩM tO − ωP) −s (ΩM tO − ωP) 0 ⌉ TIR = |ITz (−ΩM tO + ωP) |R = ⌊s (ΩM tO − ωP) c (ΩM tO − ωP) 0⌋ (A.2) 0 0 1 TRB = Tz (−τ) Ty (π + δ | Tz (−χ) Ty (−γ) |) | | | R 2 | V | B ⌈cτ (−sδcχcγ + cδsγ) − stsχcγ = ⌊sτ (−sδcχcγ + cδsγ) + ctsχcγ cδcχcγ − cδsγ ctsδsχ − stcχ stsδsχ + ctcχ −cδsχ TBP = |BTz (−ψT) Ty (−εT) |P ⌈cψT ceT = ⌊sψT ceT −seT cτ (−sδcχsγ − cδcγ) − stsχsγ⌉ sτ (−sδcχsγ − cδcγ) + ctsχsγ⌉ (A.3) cδcχsγ − sδcγ −sψT cψT seT ⌉cψT sψT seT ⌋ (A.4) 0 ceT A.3. TRANSFORMATION BETWEEN DIFFERENT COORDINATE SYSTEMS In the previous sections the different RFs and transformations were expressed using the x, y and z-axes. These x, y and z-coordinates belong to the cartesian coordinate system. However, during an integration, simulation or analysis it might be more useful and meaningful to express position (and velocity) in a different coordinate system. Two other systems are the spherical and Keplerian coordinate systems. How to transform back and forth from the cartesian system to these other two systems will be explained in Appendices A.3.1 and A.3.2 respectively and are based on Noomen (2013a). The transformation from the spherical coordinate system to Kepler elements is described in Appendix A.3.3. A.3.1. SPHERICAL AND CARTESIAN Spherical coordinates can be useful to determine the location of a satellite above Mars assuming that Mars is a perfect sphere. In this thesis problem, however, spherical coordinates are used during the MAV ascent sim- ulation. The coordinate relation between spherical and cartesian coordinates is shown in Figure 3.3. Based on this diagram, the relations to go from the spherical system to the cartesian system are derived and are provided in Equation (A.5). Please note that λ ≠ τ except for a transformation to (or from) the R-frame. x = r cos δ cos λ y = r cos δ sin λ (A.5) z = r sin δ The velocities can be obtained by taking the time derivatives of these functions, resulting in the velocity ex- pressions in x-, y- and z-direction given by Equation (A.6). Differentiating those expressions again will result in the accelerations. ẋ = ṙ cos δ cos λ − r δ̇ sin δ cos λ − r λ̇ cos δ sin λ ẏ = ṙ cos δ sin λ − r δ̇ sin δ sin λ + r λ̇ cos δ cos λ (A.6) ż = ṙ sin δ + r δ̇ cos δ The transformation from cartesian coordinates to spherical coordinates can again be derived from Figure 3.3, resulting in the expressions presented in Equation (A.7). r = $\sqrt{x2+y2+z2}$ δ = arcsin (z) λ = arctan (y) r (A.7) x However, the expression for λ only results in a value between −π/2 and π/2. Also, if x = 0, a singularity occurs. To solve this issue, the so-called 'atan2' function can be used. This function incorporates both sine and cosine values to provide a value for λ between 0 and 2π (Noomen, 2013a). λ can then be expressed by Equation (A.8) where atan2 is defined by Figure A.12. λ = atan2(y,x) (A.8) 2Online blog: http://guihaire.com/code/?p=1168 [Accessed 13 November 2015] Figure A.1: atan2 function evaluation conditions. The time derivatives of these functions can be taken to find the velocity expressions. The resulting velocity ex- pressions are shown in Equation (A.9). The left expressions in this equation define the velocity characteristics as often expressed in the spherical coordinate system required for transformations. ṙ = $\frac{xẋ+yẏ+zż}{\sqrt{x2+y2+z2}}$ δ̇ = $\frac{xẋy2 −+ yy ẋ2}{}$ λ̇ = $\frac{rż−zṙ}{√}$ r2 −(z)2 1 r V = $\sqrt{ẋ2+ẏ2+ż2}$ γ=arcsin [ṙ] V (A.9) χ = arccos r δ̇ [ ] V cos γ A.3.2. KEPLERIAN AND CARTESIAN The relation between the Keplerian and cartesian systems is slightly more complex. Figure A.2 shows the Kepler elements: a (semi-major axis), e (eccentricity, not shown but is a ratio that determines the elliptic properties of an orbit), i (inclination), ω (argument of perigee), Ω (right ascension of the ascending node) and θ (true anomaly). Often, however, the mean anomaly M is used instead of the true anomaly. The mean anomaly is defined to be the angle between the semi-major axis and the line between the centre of the orbit and an imaginary point (SM) on the auxiliary circle that can be drawn around the orbit. This point is deter- mined by assuming a constant rotational velocity in this circular orbit with the same orbital time as the actual

(elliptical) orbit. Figure A.2: Definition of the Kepler elements in 2D (left) and 3D (right) (Akcasu & Akcay, 2013; Noomen, 2013a). The transformation from Kepler elements to cartesian coordinates consists of a number of steps starting with the mean anomaly and eccentricity. Both these values have to be used to determine the eccentric anomaly E and in turn the true anomaly θ. The eccentric anomaly is computed by rewriting the first expression of Equation (A.10) into the second expression (Noomen, 2013a). M = E − e · sin E Ei+1 = Ei + M −1−Eie−·ceo·ssEiniEi (A.10) The determination of E is an iterative process. First a reasonable estimate of E has to be provided (it is best to use the value for M for this) and then a new E can be computed. This has to be done until the desired level of precision is reached. Then the true anomaly can be computed using this E as shown by Equation (A.11). θ = 2 · arctan tan (E) √ 11 +− ee [ ] 2 (A.11) With the true anomaly known, the radial distance from the orbiting body to the centre of Mars r can be determined using Equation (A.12) r = a · (1 − e2) 1 + e cos θ (A.12) The cartesian coordinates can now be computed using the expression in Equation (A.13) with the matrix entries provided by Equation (A.14)(Noomen, 2013a). ⌈ x ⌉ l1 ⌈ \y⌉ = ⌊m1 z n1 ⌉ m2⌋ r sinθ l2 (r cosθ) n2 l1 = cos Ω cos ω − sin Ω sin ω cos i l2 = − cos Ω sin ω − sin Ω cos ω cos i m1 = sin Ω cos ω + cos Ω sin ω cos i m2 = − sin Ω sin ω + cos Ω cos ω cos i n1 = sin ω sin i n2 = cos Ω sin i Equation (A.15) then provides the expressions for x, y and z. x = r · [cos Ω cos (ω + θ) − sin Ω sin (ω + θ) cos i] y = r · [sin Ω cos (ω + θ) − cos Ω sin (ω + θ) cos i] z = r sin i sin (ω + θ) (A.13) (A.14) (A.15) For the velocity values, an extra parameter is required. This parameter is called the specific relative angular momentum h. Assuming that the mass of the orbiting body can be neglected with respect to the planet (or other celestial body) it is orbiting, h can be expressed as a function of the standard gravitational parameter of that planet μ, a and e as shown by Equation (A.16). h = √μa · (1 − e2) (A.16) Then with h and the expression from Equation (A.14), the velocities can be expressed by Equation (A.17). ẋ = [−l1 sin θ + l2 · (e + cos θ)] μ h ẏ = [−m1 sin θ + m2 · (e + cos θ)] μ h (A.17) ż = [−n1 sin θ + n2 · (e + cos θ)] μ h It is also possible to convert from the cartesian system back to the Kepler system, however, this requires more steps and intermediate expressions. First, it is convenient to express the position and velocity both in one vector each as shown by Equation (A.18). r=[x y V=[ẋ ẏ z] ż] (A.18) The cross product of these two (r × V) will result in the specific relative angular momentum vector h and the scalar values, or lengths, can then be represented by r =∥r∥, V = ∥V∥ and h = ∥h∥. Also, a vector N can be defined as shown by Equation (A.19)(Noomen, 2013a). ⌈ 0 ⌉ N = \0⌋ × h (A.19) 1 With these parameters set, the first four Kepler elements (a, e, i and Ω) can be computed as shown by Equa-tion (A.20). Here, the value for the eccentricity can be obtained by taking the length of the vector: e = ∥e∥. Also, h(3) is the third value of the specific relative angular momentum vector, so the value in the z-direction. The same principle holds for the values from N. a= (2 − V 2)−1 r μ e = V × h − r μ r (A.20) i = arccos (h(3)) h Ω = atan2 (N(2), N(1)) For the computation of the values for ω and θ, three unit vectors are needed: N̂ = ∥N∥ , ê = ∥e∥ and r̂ = ∥r∥ . N e r This leads to the conditional expressions seen in Equation (A.21). ω = {a−racrccocso(sê(·êN̂·)N̂,) , iofth(N̂er×weis)e·h > 0 θ = {arccos (r̂ · ê) , if (e × r) · h > 0 (A.21) − arccos (r̂ · ê) , otherwise The eccentric anomaly can now be computed using θ and e as shown by Equation (A.22), which is the rewrit- ten form of Equation (A.11). E = 2 · arctan tan (θ) √ 11 −+ ee [ ] 2 (A.22) And finally, the mean anomaly can be found using Equation (A.10)(Noomen, 2013a). A.3.3. SPHERICAL AND KEPLERIAN Often it is required to transform the spherical coordinates into Kepler elements. Wakker (2010a) provides this direct transformation through the relations provided in Equation (A.23). a= r 2− rV 2 μ e = √1 − r V 2 (2 − r V 2) μ μ cos2 γ i = arccos (cos δ sin χ) θ = 2 arctan [√ 11 +− ee tan (E)] [√] (A.23) 2 , where E = arctan a r V sin γ μ a −r ω = sin δ − θ sin (i) Ω = λ − arccos [cos χ] sin (i) Then the spherical coordinates can be computed by rewriting the expressions provided in Equation (A.23) to the spherical coordinates. These rewritten expressions are shown in Equation (A.24). r = a (1 − e cos (E)) , where E = 2 arctan [√ 11 −+ ee tan (θ)] 2 √ ( ) V = μ 2 − 1 r a γ = arcsin e sin E μa] [? r V (A.24) δ = arcsin [sin (ω + θ) sin i] λ = arcsin tan δ − Ω [ ] χ=arccos cos(ω+θ)− sin i [tan(i)] cosδ B MARS-GRAM 2005 INPUT FILE 1 $INPUT 2 LSTFL = 'LIST.txt' 3 OUTFL = 'OUTPUT.txt' 4 TRAJFL = 'TRAJDATA.txt' 5 profile = 'null' 6 WaveFile = 'null' 7 DATADIR = '/home/stachap/MarsGram/binFiles_TUDelft/' 8 GCMDIR = '/home/stachap/MarsGram/binFiles_TUDelft/' 9 IERT = 0 10 IUTC = 1 11 MONTH = 2 12 MDAY = 28 13 MYEAR = 2025 14 NPOS = 32061 15 IHR = 12 16 IMIN = 0 17 SEC = 0.0 18 LonEW = 1 19 Dusttau = 0 20 Dustmin = 0.3 21 Dustmax = 1.0 22 Dustnu = 0.003 23 Dustdiam = 5.0 24 Dustdens = 3000. 25 ALS0 = 0.0 26 ALSDUR = 48. 27 INTENS = 0.0 28 RADMAX = 0.0 29 DUSTLAT = 0.0 30 DUSTLON = 0.0 31 MapYear = 0 32 F107 = 68.0 33 STDL = 0.0 34 NR1 = 1234 35 NVARX = 1 36 NVARY = 0 37 LOGSCALE = 0 38 FLAT = 21 39 FLON = 74.5 40 FHGT = -0.6 41 MOLAhgts = 1 42 hgtasfcm = 0. 43 zoffset = 0. 44 ibougher = 0 45 DELHGT = 0.01 46 DELLAT = 0.0 47 DELLON = 0.0 48 DELTIME = 0.0 49 ΔTEX = 0.0 97 50 profnear = 0.0 51 proffar = 0.0 52 rpscale = 1.0 53 rwscale = 1.0 54 wlscale = 1.0 55 wmscale = 0.0 56 blwinfac = 0.0 57 NMONTE = 1 58 iup = 13 59

WaveA0 = 1.0 60 WaveDate = 0.0 61 WaveA1 = 0.0 62 Wavephi1 = 0.0 63 phi1dot = 0.0 64 WaveA2 = 0.0 65 Wavephi2 = 0.0 66 phi2dot = 0.0 67 WaveA3 = 0.0 68 Wavephi3 = 0.0 69 phi3dot = 0.0 70 iuwave = 0 71 Wscale = 20. 72 corlmin = 0.0 73 ipclat = 1 74 requa = 3396.19 75 rpole = 3376.20 76 idaydata = 1 77 $END 78 79 Explanation of variables: 80 LSTFL = List file name (CON for console listing) 81 OUTFL = Output file name 82 TRAJFL = (Optional) Trajectory input file. File contains time (sec) 83 84 85 86 87 88 89 90 91 92 93 94 95 96 97 98 99 100 101 102 103 104 105 106 107 108 109 110 111 112 113 114 115 116 117 118 119 120 profile = WaveFile = DATADIR GCMDIR IERT relative to start time, height (km), latitude (deg), longitude (deg W if LonEW=0, deg E if LonEW=1, see below) (Optional) auxiliary profile input file name (Optional) file for time-dependent wave coefficient data. See file description under parameter iuwave, below. = Directory for COSPAR data and topographic height data = Directory for GCM binary data files = 1 for time input as Earth-Receive time (ERT) or 0 Mars-event time (MET) IUTC = 1 for time input as Coordinated Universal Time (UTC), or 0 MONTH = MDAY = MYEAR = NPOS = for Terrestrial (Dynamical) Time (TT) (Integer) month of year (Integer) day of month (Integer) year (4-digit; 1970-2069 can be 2-digit) max # positions to evaluate (0 = read data from trajectory input file) IHR = Hour of day (ERT or MET, controlled by IERT and UTC or TT, IMIN SEC LonEW Dusttau Dustmin Dustmax Dustnu = = = = = = = Dustdiam = Dustdens = ALS0 = ALSDUR = INTENS = RADMAX = DUSTLAT = DUSTLON = controlled by IUTC) minute of hour (meaning controlled by IERT and IUTC) seconds of minute (meaning controlled by IERT and IUTC). IHR:IMIN:SEC is time for initial position to be evaluated 0 for input and output West longitudes positive; 1 for East longitudes positive Optical depth of background dust level (no time-developing dust storm, just uniformly mixed dust), 0.1 to 3.0, or use 0 for assumed seasonal variation of background dust Minimum seasonal dust tau if input Dusttau=0 (≥0.1) Maximum seasonal dust tau if input Dusttau=0 (≤1.0) Parameter for vertical distribution of dust density (Haberle et al., J. Geophys. Res., 104, 8957, 1999) Dust particle diameter (micrometers, assumed monodisperse) Dust particle density (kg/m**3) starting Ls value (degrees) for dust storm (0 = none) duration (in Ls degrees) for dust storm (default = 48) dust storm intensity (0.0 - 3.0) max. radius (km) of dust storm (0 or >10000 = global) Latitude (degrees) for center of dust storm Longitude (degrees) (West positive if LonEW=0, or East 121 positive if LonEW = 1) for center of dust storm 122 MapYear = 1 or 2 for TES mapping year 1 or 2 GCM input data, or 0 for 123 Mars-GRAM 2001 GCM input data sets 124 F107 = 10.7 cm solar flux (10**-22 W/cm**2 at 1 AU) 125 NR1 = starting random number (0 < NR1 < 30000) 126 NVARX = x-code for plotable output (1=hgt above MOLA areoid). 127 See file xycodes.txt 128 NVARY = y-code for 3-D plotable output (0 for 2-D plots) 129 LOGSCALE = 0=regular SI units, 1=log-base-10 scale, 2=percentage 130 deviations from COSPAR model, 3=SI units, with density 131 in kg/km**3 (suitable for high altitudes) 132 FLAT = initial latitude (N positive), degrees 133 FLON = initial longitude (West positive if LowEW = 0 or East 134 positive if LonEW = 1), degrees 135 FHGT = initial height (km); ≤-10 means evaluate at surface height; 136 > 3000 km means planeto-centric radius 137 MOLAhgts = 1 for input heights relative to MOLA areoid, otherwise 138 input heights are relative to reference ellipsoid 139 hgtasfcm = height above surface (0-4500 m); use if FHGT ≤ -10. km 140 zoffset = constant height offset (km) for MTGCM data or constant 141 part of Ls-dependent (Bougher) height offset (0.0 means 142 no constant offset). Positive offset increases density, 143 negative offset decreases density. 144 ibougher = 0 for no Ls-dependent (Bougher) height offset term; 1 145 means add Ls-dependent (Bougher) term, -A*Sin(Ls) (km), 146 to constant term (zoffset) [offset amplitude A = 2.5 for 147 MapYear=0 or 0.5 for MapYear > 0]; 2 means use global mean 148 height offset from data file hgtoffst.dat; 3 means use 149 daily average height offset at local position; 4 means 150 use height offset at current time and local position. 151 Value of zoffset is ignored if ibougher = 2, 3, or 4. 152 DELHGT = height increment (km) between steps 153 DELLAT = Latitude increment (deg) between steps (Northward positive) 154 DELLON = Longitude increment (deg) between steps (Westward positive 155 if LonEW = 0, Eastward positive if LonEW = 1) 156 DELTIME = time increment (sec) between steps 157 ΔTEX = adjustment for exospheric temperature (K) 158 profnear = Lat-lon radius (degrees) within which weight for auxiliary 159 profile is 1.0 (Use profnear = 0.0 for no profile input) 160 proffar = Lat-lon radius (degrees) beyond which weight for auxiliary 161 profile is 0.0 162 rpscale = random density perturbation scale factor (0-2) 163 rwscale = random wind perturbation scale factor (≥0) 164 wlscale = scale factor for perturbation wavelengths (0.1-10) 165 wmscale = scale factor for mean winds 166 blwinfac = scale factor for boundary layer slope winds (0 = none) 167 NMONTE = number of Monte Carlo runs 168 iup = 0 for no LIST and graphics output, or unit number for output 169 WaveA0 = Mean term of longitude-dependent wave multiplier for density 170 WaveDate = Julian date for (primary) peak(s) of wave (0 for no traveling 171 component) 172 WaveA1 = Amplitude of wave-1 component of longitude-dependent wave 173 multiplier for density 174 Wavephi1 = Phase of wave-1 component of longitude-dependent wave 175 multiplier (longitude, with West positive if LonEW = 0, 176

East positive if LonEW = 1) 177 phi1dot = Rate of longitude movement (degrees per day) for wave-1 178 component (Westward positive if LonEW = 0, Eastward 179 positive if LonEW = 1) 180 WaveA2 = Amplitude of wave-2 component of longitude-dependent wave 181 multiplier for density 182 Wavephi2 = Phase of wave-2 component of longitude-dependent wave 183 multiplier (longitude, with West positive if LonEW = 0, 184 East positive if LonEW = 1) 185 phi2dot = Rate of longitude movement (degrees per day) for wave-2 186 component (Westward positive if LonEW = 0, Eastward 187 positive if LonEW = 1) 188 WaveA3 = Amplitude of wave-3 component of longitude-dependent wave 189 multiplier for density 190 Wavephi3 = Phase of wave-3 component of longitude-dependent wave 191 multiplier (longitude, with West positive if LonEW = 0, 192 193 phi3dot 194 195 196 iuwave 197 198 199 200 201 Wscale 202 203 corlmin 204 205 206 ipclat 207 208 209 210 = = = = = requa = rpole = idaydata = East positive if LonEW = 1) Rate of longitude movement (degrees per day) for wave-3 component (Westward positive if LonEW = 0, Eastward positive if LonEW = 1) Unit number for (Optional) time-dependent wave coefficient data file "WaveFile" (or 0 for none). WaveFile contains time (sec) relative to start time, and wave model coefficients (WaveA0 thru Wavephi3) from the given time to the next time in the data file. Vertical scale (km) of longitude-dependent wave damping at altitudes below 100 km (10≤Wscale≤10,000 km) minimum relative step size for perturbation updates (0.0-1.0); 0.0 means always update perturbations, x.x means only update perturbations when corlim > x.x 1 for Planeto-centric latitude and height input, 0 for Planeto-graphic latitude and height input Equatorial radius (km) for reference ellipsoid Polar radius (km) for reference ellipsoid 1 for daily max/min data output; 0 for none C ATMOSPHERIC DATA FITTING In Section 3.4.1 the final results of the atmospheric data fits were presented. However, before those results could be obtained, a few iterations of trial-and-error fits were required. These intermediate trials are pre-sented for both the temperature data, in Appendix C.1, as well as for the density data, in Appendix C.2. The fit requirements for the polynomial function were set such that the errors created by the fit were less than the uncertainty of the latitude and longitude dependent data. This way the desired accuracy could still be reached. This uncertainty is caused by the difference in atmospheric data curves for the different latitudes and longitudes. The atmospheric data curve for the latitude and longitude corresponding to the launch site (21.0 ◦N and 74.5 ◦E) was taken as the reference curve and called the launch site curve. The differences be- tween this launch site curve and the 8 other curves were used to define the maximum data curves difference. The polyfit.m function within Matlab was used to fit a polynomial to the data. This function also directly pro- vides a standard deviation of the fit for each data point (in combination with the polyval.m function). One of the requirement was for the maximum standard deviation of the polynomial fit to be one order less than the maximum difference of the data curves with respect to the launch site curve. The second requirements for a proper fit was for the absolute maximum difference between the polynomial fit and the launch site curve to be less than the absolute maximum difference between the data curves and that same launch site curve. C.1. TEMPERATURE Initially, the temperature data curve was split into 6 different sections as portrayed in Figure C.1. The sections were selected on the basis of their individual shapes and the maximum order that was required, where the maximum order for temperature was set at 8. Fewer sections were however always preferred. Unfortunately, the fourth quarter of section 3 caused this section to not meet the maximum standard devia- tion requirement because in that quarter the linear behaviour changes directions slightly and the differences between the different data curves become a lot smaller. Therefore section 3 was split into two different sec- tions resulting in a total of 7 sections as shown in Figure C.2. 101 C.1. TEMPERATURE Figure C.1: Six different temperature curve sections Figure C.2: Seven different temperature curve sections C.2. DENSITY Figure C.3: All section fits for the launch site temperature data curve These sections all met the requirements and thus provided a proper fit to the data. This fit is shown in Fig- ure C.3. However, because fewer sections were preferred, an attempt was made to reduce the number of sections. Because of the inaccuracy in the Mars-GRAM model close to the planet surface, it was decided to delete the outliers near the surface (section 1) and stretch the outcome of the second polynomial fit to the surface instead. Also, it turned out that section 4 and 5 could be combined in such a way that the requirements were still met. This reduced the number of sections from 7 to 5. The final fit was presented in Section 3.4.1 as well as the corresponding errors and polynomial coefficients. C.2. DENSITY For the density curve, because the data represents a logarithmic curve, initially an exponential atmosphere was fit. However, this did not meet any of the requirements for a proper fit. This is why a polynomial fit was attempted for the density curve as well with the same requirement as the temperature fits. The first fit was attempted with two sections as presented in Figure C.4. Figure C.4: Two different density curve sections C.2. DENSITY Figure C.5: Three different density curve sections However, the second section could not meet the requirements because of the initial part. This is why it was split into three sections as presented in Figure C.5. This did meet the error

requirements and resulted in the fit as shown in Figure C.6 with section 1 being the lower part, section 2 the middle and section 3 the upper part. However, in this case the fit for section three resulted in an oscillating behaviour around the actual data which caused the density values to drop below zero as shown in Figure C.7. Since this is unrealistic, an exponential atmospheric fit for only the third section was attempted (also shown in Figure C.7) however, this resulted in the same behaviour as the attempt for the entire curve. Therefore, it was decided to try a different exponential approach which eventually resulted in the fit described in Section 3.4.1. Figure C.8 clearly shows that both the full exponential atmospheric fit and the polynomial fits were not a proper choice here. The figure shows the curved part of the curve. Figure C.6: All section fits for the launch site density data curve C.2. DENSITY Figure C.7: Section 3 fit for the launch site density data curve Figure C.8: Zoom in of the end of section 1 with the polynomial and exponential atmospheric fits D ALL RECURRENCE RELATIONS In this appendix, all the reduced auxiliary functions that form the complete recurrence relations for each of the variables using the basic recurrence relations are provided for all three cases. Each case contains a table with the function definition, the corresponding equation and the basic recurrence relation category. All the reduced auxiliary derivatives are also described. All these expressions are a function of k, where $W(k)$ refers to the kth order reduced derivative. The same holds for $X(k)$ and $U(k)$. All the expressions hold for $k \geq 1$. In the tables, the (k) is neglected to make it more readable. D.1. FIRST CARTESIAN CASE: EULER ANGLES The first Cartesian case requires a large number of reduced auxiliary functions. These are all provided in Table D.1. For each function, the number, the equations and the category is provided. Table D.1: Reduced auxiliary functions for the first Cartesian case Auxiliary function Equation Category $W4,1 = X12 + X22$ Multiplication $W4,2 = W4,1 + X32$ Multiplication $W4,3 = \sqrt{W4,2}$ Power $W4,4 = \sqrt{W4,1}$ Power $W4,5 = X2 W4,4$ Division $W4,6 = X1 W4,4$ Division $W4,7 = X3 W4,3$ Division $W4,8 = W4,4 W4,3$ Division $W4,9 = W4,10 = X4 + \Omega M X2$ Constant Multiplication $X5 - \Omega M X1$ $W4,11 = W42,9 + W42,10 + X62$ Constant Multiplication $W4,12 \sqrt{W4,11}$ Power $=$ Multiplication $W4,13 = -W4,6W4,7$ Multiplication $W4,14 = W4,15 = -W4,7W4,5 -W4,8W4,6$ Multiplication Multiplication $W4,16 = W4,8W4,5$ Multiplication $W4,17 = X6W4,8 +W4,9W4,13 +W4,10W4,14$ Multiplication $W4,18 = W4,10W4,6 -W4,9W4,5$ Multiplication $W4,19 = W4,9W4,15 - X6W4,7 +W4,10W4,16$ Multiplication $W4,20 = W42,17 +W42,18$ Multiplication 106 D.1. FIRST CARTESIAN CASE: EULER ANGLES $W4,21 = \sqrt{W4,20}$ Power $W4,22 = W4,18 W4,21$ Division $W4,23 = W4,17 W4,21$ Division $W4,24 = -W4,19 W4,12$ Division $W4,25 = W4,21$ Division $W4,26 = W4,12$ $W4,27 = c\varepsilon T$ Cosine Cosine $c\psi T$ $W4,28 = s\psi T$ Sine $W4,29 = s\varepsilon T$ Sine $W4,30 = W4,26W4,27$ Multiplication $W4,31 = W4,27W4,28$ Multiplication 1 $W4,32 = $ Division $W4,33 = X7$ $W4,34 = TW4,32$ Constant Multiplication $W4,33W4,30$ Multiplication $W4,35 = X27$ Division $W4,36 = X7$ $W4,34 -W4,35$ Subtraction $W4,37 = W4,33W4,31$ Multiplication $W4,38 = W4,33W4,29$ Multiplication $W4,39 = -\mu M XX19$ Division $W4,40 = -W4,7W4,23$ Multiplication $W4,41 = W4,8W4,24$ Multiplication $W4,42 = -W4,5W4,22$ Multiplication $W4,43 = -W4,5W4,23$ Multiplication $W4,44 = -W4,8W4,25$ Multiplication $W4,45 = W4,40W4,25$ Multiplication $W4,46 = W4,42W4,25$ Multiplication $W4,47 = -W4,13W4,22$ Multiplication $W4,48 = W4,40W4,24$ Multiplication $W4,49 = W4,42W4,24$ Multiplication $= W4,50$ Multiplication $= W4,6 (W4,45 +W4,41)+W4,46$ $W4,51$ Multiplication $= W4,6 (W4,48 +W4,44)+W4,49$ $W4,39 +W4,36W4,50 +W4,37 (W4,47 +W4,43)-W4,38W4,51$ $W4,52$ Multiplication $W5,1 = $ Division $W5,2 = -\mu M XX29$ $W5,3$ Multiplication $W4,6W4,22$ Multiplication $= W4,5 (W4,45 +W4,41)+W5,2W4,25$ $W5,4 = = -W4,14W4,22 +W4,6W4,23$ Multiplication $W4,5 (W4,48 +W4,44)+W5,2W4,24$ $W5,5$ Multiplication $W5,6 = W5,1 +W4,36W5,3 +W4,37W5,4 -W4,38W5,5$ Multiplication $W6,1 = $ Division $W6,2 = -\mu M XX39$ $W4,7W4,24$ Multiplication $W6,3 = W4,8W4,22$ Multiplication $W6,4 = -W4,7W4,25$ Multiplication $W6,5 = -W4,44W4,23 +W6,2$ Multiplication $W6,6 = W4,41W4,23 +W6,4$ Multiplication $W6,7 = W6,1 +W4,36W6,5 -W4,37W6,3 -W4,38W6,6$ Multiplication $W8,1 = X1X4$ Multiplication $W8,2 = X2X5$ Multiplication $W8,3 = X3X6$ Multiplication $W9,0 = X9U8$ Multiplication D.2. SECOND CARTESIAN CASE: UNIT VECTORS $W9,1 = X8$ Division $W27,1 = W9,0 W4,3$ Redefining $W27,2 = W227,1$ Multiplication $W27,3 = W27,4 = W247,1$ Power $W27,5 = $ Power $W237,1$ Power $W27,6 = W257,1 W267,1$ Power $W27,7 = $ Power $W27,8 = W277,1$ $W27,9 = $ Power $W27,10 = W297,1 W287,1$ Power $W2170,1$ Power $W27,11 = P\rho 10W27,10 + P\rho 9W27,9 + . . . + P\rho 1W27,1 + P\rho 0$ Constant Multiplication $W27,12 = eW27,11$ ∫ Exponential PT 1,1W27,1, for $-0.6 \leq h < 5.04$ PT 3,2W27,3 + PT 2,2W27,2 + PT 1,2W27,1, for $5.04 \leq h < 35.53$ PT 6,3W27,6 + PT 5,3W27,5 + PT 4,3W27,4 + . . . $W27,13 = Ta = $ ⎱⎰ . . . + PT 3,3W27,3 + PT 2,3W27,2 + PT 1,3W27,1, for $35.53 \leq h < 75.07$ PT 8,4W27,8 + PT 7,4W27,7 + PT 6,4W27,6 + . . . Constant . . . + PT 5,4W27,5 + PT 4,4W27,4 + PT 3,4W27,3 + . . . Multiplication for $75.07 \leq h < 170.05$ √ ⎱⎰ 0, . . . + PT 2,4W27,2 + PT 1,4W27,1, for $h \geq 170.05$ $W27,14 = $ γa Ra∗W27,13 Power $W27,15 = W4,12 W27,14$ Division ⎰ 0, for $0 \leq M < 0.5$ PCD 1,2W27,15, for $0.5 \leq M < 1$ $W27,16 = CD = $ ⎱⎰ PCD 1,3W27,15, for $1 \leq M < 1.3$ PCD 1,4W27,15, for $1.3 \leq M < 2.5$ Constant PCD 1,5W27,15, for $2.5 \leq M < 4$ Multiplication ⎰⎱ 0, for $M \geq 4$ $W42,12 W27,17 = $ Multiplication $W27,18 = $

W27,17W27,16 Multiplication W27,19 = 12 SW27,18W27,12 Multiplication The complete recurrence relation for each of the auxiliary variables can now be written as a function of the reduced auxiliary functions. These are shown by Equation (D.1) and hold for k ≥ 1. U1 (k) = X 4 (k) = U4 (k − 1) U2 (k) = X 5 (k) = U5 (k − 1) k U3 (k) = X 6 (k) = U6 (k − 1) k k U4 (k) = W4,52 (k) U7 (k) = 0 U5 (k) = W5,6 (k) U8 (k) = 2W8,1 (k) + 2W8,2 (k) + 2W8,3 (k) (D.1) U6 (k) = W6,7 (k) U9 (k) = W9,1 (k) 3 2 These equations are now all a function of the recurrence relations corresponding to the rest of the auxiliary functions, which are all basic recurrence relations as mentioned in Table D.1. D.2. SECOND CARTESIAN CASE: UNIT VECTORS The second Cartesian case is similar to the first one, however fewer reduced auxiliary functions are needed. These are all provided in Table D.2. For each function, the number, the equations and the category is pro- vided. The numbers are not always the same as in the first case, so these should be read as completely new D.2. SECOND CARTESIAN CASE: UNIT VECTORS reduced auxiliary functions. In this case, u4, u5 and u6 are included in the table, because they were defined as recurrence multiplication relations. The complete recurrence relation for each of the auxiliary variables can again be written as a function of the reduced auxiliary functions. These are shown by Equation (D.2). Table D.2: Reduced auxiliary functions for the second Cartesian case Function Equation Category W4,1 = Multiplication X12 + X22 W4,2 = Multiplication = √W4W,14+,2X32 W4,3 Power = W4,4 √W4,1 Power W4,9 = W4,10 = X4 +ΩM X2 X5 −ΩM X1 Constant Multiplication W4,11 = W√4W2,94+,1W142,10 + X62 Constant Multiplication W4,12 Power = Multiplication W4,13 = W4,9 W4,12 Division W4,14 = W4,10 W4,12 Division W4,15 = X6 Division W4,16 = W4,12 W4,10X3 − X6X2 Multiplication W4,17 = X6X1 −W4,9X3 Multiplication W4,18 = W4,9X2 −W4,10X1 Multiplication W4,19 = = W42,16 +W42,17 +W42,18 Multiplication W4,20 √W4,19 Power W4,21 = W4,16 W4,20 Division W4,22 = W4,17 W4,20 Division W4,23 = W4,18 Division W4,24 = W4,20 W4,16W4,23 −W4,15W4,22 Multiplication W4,25 = W4,15W4,21 −W4,13W4,23 Multiplication W4,26 = cψT Cosine W4,27 = cεT Cosine W4,28 = sψT Sine W4,29 = sεT Sine W4,30 = W4,26W4,27 Multiplication W4,31 = W4,27W4,28 Multiplication 1 W4,32 = Division W4,33 = X7 W4,34 = TW4,32 Constant Multiplication W4,33W4,30 Multiplication W4,35 = X27 Division W4,36 = X7 W4,34 −W4,35 Subtraction W4,37 = W4,33W4,31 Multiplication W4,38 = W4,33W4,29 Multiplication W4,39 = −μM XX19 Division W4,40 = W4,13W4,22 −W4,14W4,21 Multiplication U4 = W4,39 +W4,36W4,13 +W4,37W4,21 −W4,38W4,24 Multiplication W5,1 = Division U5 = −μM XX29 W5,1 +W4,36W4,14 +W4,37W4,22 −W4,38W4,25 Multiplication W6,1 = −μM XX39 Division D.2. SECOND CARTESIAN CASE: UNIT VECTORS U6 = W6,1 + W4,36W4,15 + W4,37W4,23 − W4,38W4,40 Multiplication W8,1 = X1 X4 Multiplication W8,2 = X2 X5 Multiplication W8,3 = X3 X6 Multiplication W9,0 = X9U8 Multiplication W9,1 = W9,0 Division W27,1 = X8 W4,3 Redefining W27,2 = Multiplication W227,1 W27,3 = W237,1 Power W27,4 = W27,5 = Power W247,1 W257,1 Power W27,6 = Power W27,7 = W267,1 W27,8 = Power W27,9 = W287,1 W277,1 Power W297,1 Power W27,10 = Power W27,11 = W2170,1 W27,12 = ∫ Exponential Pρ10W27,10 + Pρ9W27,9 + . . . + Pρ1W27,1 + Pρ0 Constant Multiplication eW27,11 PT 1,1W27,1, for −0.6 ≤ h < 5.04 PT 3,2W27,3 + PT 2,2W27,2 + PT 1,2W27,1, for 5.04 ≤ h < 35.53 PT 6,3W27,6 + PT 5,3W27,5 + PT 4,3W27,4 + . . . W27,13 = Ta = . . . + PT 3,3W27,3 + PT 2,3W27,2 PT 1,3W27,1, + for 35.53 ≤ h < 75.07 PT 8,4W27,8 + PT 7,4W27,7 + PT 6,4W27,6 + . . . Constant . . . + PT 5,4W27,5 + PT 4,4W27,4 + PT 3,4W27,3 + . . . Multiplication . . . + PT 2,4W27,2 + PT 1,4W27,1, for 75.07 ≤ h < 170.05 for h 170.05 ≥ √ 0, W27,14 = γa Ra W27,13 ∗ Power W4,12 W27,15 = W27,14 Division W27,17 = W42,12 W27,16 = 0, for 0 ≤ M 0.5 < PCD 1,2W27,15, for 0.5 ≤ M < 1 CD = PCD 1,3W27,15, for 1 ≤ M 1.3 < PCD 1,4W27,15, for 1.3 ≤ M < 2.5 PCD 1,5W27,15, for 2.5 ≤ M < 4 0, for M ≥ 4 Constant Multiplication Multiplication W27,18 = W27,17W27,16 Multiplication W27,19 = 12 SW27,18W27,12 Multiplication U1 (k) = X4 (k) = U4 (k − 1) U2 (k) = X5 (k) = U5 (k − 1) k U3 (k) = X6 (k) = U6 (k − 1) k k U4 (k) = W4,39 (k) +W4,36 (k)W4,13 (k) +W4,37 (k)W4,21 (k) −W4,38 (k)W4,24 (k) U5 (k) = W5,1 (k) +W4,36 (k)W4,14 (k) +W4,37 (k)W4,22 (k) −W4,38 (k)W4,25 (k) (D.2) U6 (k) = W6,1 (k) +W4,36 (k)W4,15 (k) +W4,37 (k)W4,23 (k) −W4,38 (k)W4,40 (k) U7 (k) = 0 U8 (k) = 2W8,1 (k) + 2W8,2 (k) + 2W8,3 (k) U9 (k) = W9,1 (k) 3 2 D.3. SPHERICAL CASE D.3. SPHERICAL CASE Since the Spherical case is very different from the Cartesian cases, it comes with its own auxiliary functions. The reduced versions are shown in Table D.3. In this case, a number of extra auxiliary equations were used, which result in extra recurrence relations. These are all described in Equation (D.3) and hold for k 1. ≥ Table D.3: Reduced auxiliary functions for the Spherical case. Auxiliary function Equation Category W4,4 = sX12 Sine W4,5 = cX13 Cosine W4,6 = cX12 Cosine W4,7 = sX14 Sine W4,8 = cX14 Cosine W4,9 = sX13 Sine W4,26 = cψT Cosine W4,27 = cεT Cosine W4,28 = sψT Sine W4,29 = sεT Sine 1 W4,32 = X7 Division W11,0 = X15W4,8 Multiplication W11,1 = W11,0 Division W11,2 = X16 W11,1W4,9 Multiplication W11,3 = W11,2 W4,6 Division W12,1 = W11,1W4,5 Multiplication W13,0 = W4,28W4,27 Multiplication W13,1 = W4,7W4,5 Multiplication W13,2 = Multiplication W13,3 = Ω2M X16W4,6 W13,2W4,4 Multiplication W13,4 =

TW13,0W4,32 Multiplication W13,5 = W13,3W4,9 +W13,4 Multiplication W13,6 = W13,5 Division W13,7 = X15 −2ΩMW4,6W13,1 +W13,6 Multiplication W13,8 = W13,7 W4,8 Division = + W13,9 (2ΩM W11,3)W4,4 +W13,8 Multiplication W14,0 = TW4,29W4,32 Multiplication W14,1 = W4,6W4,8 +W13,1W4,4 Multiplication W14,2 = W14,3 = −μMW4,8 X126 Constant Multiplication Multiplication W14,4 = W14,2 W14,3 Division W14,5 = W14,4 +W13,2W14,1 +W14,0 Multiplication W14,6 = W14,5 Division W14,7 = X15 W15,0 = TW4,26W4,27 − X27 Multiplication Multiplication 2ΩMW4,6W4,9 +W11,1 +W14,6 W15,1 = W15,0 X7 Division W15,2 = −μMW4,7 Constant Multiplication D.3. SPHERICAL CASE W15,3 = W14,3 Division W15,4 = W15,2 W4,8W4,5 Multiplication W15,5 = W4,7W4,6−W15,4W4,4 Multiplication W15,6 = W13,2W15,5+W15,1+W15,3 Multiplication W16,1 = X15W4,7 Multiplication W27,1 = X29U28 Multiplication W27,2 = X28U29 Multiplication W27,3 = X28X29 Multiplication = W27,4 X15(W27,1+W27,2) Multiplication W27,5 = W27,3U15 Multiplication = W27,6 X15(W27,4+W27,5) Multiplication W28,1 = U30X28 Multiplication W29,1 = $\lceil$ 0, for 0 M 0.5 ≤ < PCD1,2U32, for 0.5≤ M <1 $\rceil$ $\lceil$ PCD1,3U32, for 1≤ M 1.3 < PCD1,4U32, for 1.3≤ M <2.5 $\rceil$ PCD1,5U32, for 2.5≤ M <4 $\lfloor$ 0, for M ≥ 4 X391 Constant Multiplication W30,1 = Power W30,2 = Power W30,3 = X381 W30,4 = Power X361 X371 Power W30,5 = X351 Power W30,6 = Power W30,7 = X341 W30,8 = Multiplication = UX33211(10Pρ10W30,1 +···+3Pρ3W30,8+2Pρ2X31+Pρ1) X331 Power W30,9 Multiplication W32,1 = X33U15 Multiplication W32,2 = X15U33 Multiplication W32,3 = XW32332,1−W32,2 Multiplication W32,4 = W32,3 Division W33,1 = U34 X33 Division $\lceil$ PT1,1U31, for −0.6≤ <5.04 (3PT3,2W30,8+2PT2,2X31)U31+PT1,2U31, for 5.04≤h <35.53 (6PT6,3W30,5+5PT5,3W30,6+4PT4,3W30,7 ... + W34,1 = $\rceil$ $\rceil$ ...+3PT3,3W30,8+2PT2,3X31)U31+PT1,3U31, for 35.53 ≤ h < 75.07 (8PT8,4W30,3+7PT7,4W30,4+6PT6,4W30,5 ... + Multiplication ...+5PT5,4W30,6+4PT4,4W30,7+3PT3,4W30,8+... ...+2PT2,4X31)U31+PT1,4U31, for 75.07 ≤ h < 170.05 $\lfloor$ 0, for h 170.05 ≥ U7 (k) = 0 U11 (k) = W11,3 (k) U12 (k) = W12,1 (k) U13 (k) = W13,9 (k) U14 (k) = W14,7 (k) U15 (k) = W15,6 (k) U16 (k) = W16,1 (k) U27 (k) = SW27,6 (k) 1 2 U28 (k) = W28,1 (k) U29 (k) = W29,1 (k) U30 (k) = W30,9 (k) (D.3) U31 (k) = U16 (k) U32 (k) = W32,4 (k) U33 (k) W33,1 (k) = U34 (k) = W34,1 (k) E NOMINAL CASE INPUT VALUES In this appendix, the input files for the two different cases are given. Table E.1 describes the definition of the input parameters and the units. Table E.1: Description of the input parameters Parameter Meaning desiredOrbitalAltitude This is the altitude in km MOLA that the MAV should reach, so the al- titude of the desired orbit. At the same time is is also a cut-off criteria for the simulation. desiredInclinationDeg This is the desired inclination in degrees of the target orbit. initialAltitude This is the altitude at which the simulation starts, so the launch alti- tude for the MAV in km MOLA. initialLatitudeDeg This is the launch latitude on Mars in degrees at the start of the sim- ulation. initialLongitudeDeg This is the launch longitude on Mars in degrees at the start of the sim- ulation. FlightPathAngleDeg This is the flight-path angle in degrees at the start of the simulation. 90 degrees is straight up and 0 degrees is horizontal. HeadingAngleDeg This is the angle in degrees that defines the heading of the MAV and in the simulation it is defined 0 degrees when pointing north and 90 degrees when pointing east. initialGroundVelocity This is the ground velocity in km/s of the MAV at the start of the sim- ulation. Because both TSI and RKF have to do too many steps if the ground velocity starts at 0 km/s, it is better to give it a small initial velocity. massMAV This is the GLOM in kg of the MAV. thrust This is the nominal thrust in kN of the MAV engine and is a constant throughout the simulation. specificImpulse This is the specific impulse in seconds of the MAV engine and is a constant throughout the simulation. initialBurnTime This is the burn time in seconds of the first burn from the start of the simulation until the coasting phase. constantThrustElevationAngle This is the thrust elevation angle in degrees as defined in the propul- sion frame and is a constant throughout the simulation. constantThrustAzimuthAngle This is the thrust azimuth angle in degrees as defined in the propul- sion frame and is a constant throughout the simulation. maxOrder This is the order of the TSI simulation run, or K . chosenLocalErrorTolerance This is the error tolerance for both TSI and RKF during the actual in- tegration and should not go below 10−15. 113 E.1. CASE 1 chosenStepSize This is the initial step-size in seconds that TSI starts with. It is the default first step-size of the step-size class that is updated every time from then on. setEndTime This is a cut-off criteria and can be chosen to be a certain amount of seconds at which the simulation should stop unless one of the other cut-off criteria is met first. RKFinitiatorTime This is the minimum time in seconds that the initiator step should be run. The initiator step is an RKF run with an error tolerance of 10−15 and stops when it goes beyond the initiator time. This does not mean that it is cut-off at the initiator time exactly. rotatingPlanet If this value is 1, the rotation of Mars is taken into account during the simulation. If it is 0 the simulation is run with a non-rotating Mars such that the inertial frame and the rotating frame are the same. Gravity If this value is 1, the Martian gravitational acceleration is taken into account. If it is 0, then this is neglected. Thrust If this value is 1, the MAV thrust acceleration is taken into account. If it is 0, then this is

neglected. Drag If this value is 1, the drag acceleration is taken into account. If it is 0, then this is neglected. comparison If this value is 1, then the comparison results between TSI and RKF are computed after each integration. If it is 0, then this is not the case. E.1. CASE 1 1 Test case from Woolley 2015 (case 10 SSTO) 2 desiredOrbitalAltitude = 390.0 3 desiredInclinationDeg = 45.0 4 initialAltitude = -0.6 5 initialLatitudeDeg = 0.0 6 initialLongitudeDeg = 74.5 7 FlightPathAngleDeg = 89.0 8 HeadingAngleDeg = 90.0 9 initialGroundVelocity = 0.00001 10 massMAV = 267.4 11 thrust = 3.56 12 specificImpulse = 256 13 initialBurnTime = 142.5 14 constantThrustElevationAngle = -0.184 15 constantThrustAzimuthAngle = -0.299 16 maxOrder = 20 17 chosenLocalErrorTolerance = 1e-15 18 chosenStepSize = 0.01 19 setEndTime = 2000.0 20 RKFinitiatorTime = 1.0 21 rotatingPlanet = 1 22 Gravity = 1 23 Thrust = 1 24 Drag = 1 25 comparison = 1 E.2. CASE 2 E.2. CASE 2 1 Test case from Joel (hybrid) case7_3_2016_v33 2 desiredOrbitalAltitude = 479.19000000064 3 desiredInclinationDeg = 92.6899999999988 4 initialAltitude = -0.6 5 initialLatitudeDeg = 0.0 6 initialLongitudeDeg = 90.0 7 FlightPathAngleDeg = 89.0 8 HeadingAngleDeg = 90.0 9 initialGroundVelocity = 0.000001 10 massMAV = 288.95985303149 11 thrust = 6.01868886452604 12 specificImpulse = 315.9 13 initialBurnTime = 99.361911852794 14 constantThrustElevationAngle = -0.674 15 constantThrustAzimuthAngle = 0.7273 16 maxOrder = 20 17 chosenLocalErrorTolerance = 1e-15 18 chosenStepSize = 0.01 19 setEndTime = 2000.0 20 RKFinitiatorTime = 1.0 21 rotatingPlanet = 1 22 Gravity = 1 23 Thrust = 1 24 Drag = 1 25 comparison = 1 2 4 5 6 7 3.1. STATE AND STATE DERIVATIVES 9 3.1. STATE AND STATE DERIVATIVES 10 3.1. STATE AND STATE DERIVATIVES 11 3.1. STATE AND STATE DERIVATIVES 12 13

23.4. ATMOSPHERIC DRAG 14 3.4. ATMOSPHERIC DRAG

15

23.4. ATMOSPHERIC DRAG 16 3.4. ATMOSPHERIC DRAG

17

23.4. ATMOSPHERIC DRAG 18 3.4. ATMOSPHERIC DRAG

19 20 21 23 24 25 27 28 29 5.2. ASSOCIATED EQUATIONS 30 5.2. ASSOCIATED EQUATIONS 31 5.2. ASSOCIATED EQUATIONS 32 5.2. ASSOCIATED EQUATIONS 33 5.2. ASSOCIATED EQUATIONS 34 5.2. ASSOCIATED EQUATIONS 35 5.2. ASSOCIATED EQUATIONS 36 5.2. ASSOCIATED EQUATIONS 37 5.2. ASSOCIATED EQUATIONS 38 5.2. ASSOCIATED EQUATIONS 39 5.2. ASSOCIATED EQUATIONS 40 5.2. ASSOCIATED EQUATIONS 41 5.2. ASSOCIATED EQUATIONS 42 5.2. ASSOCIATED EQUATIONS 43 5.2. ASSOCIATED EQUATIONS 44 5.2. ASSOCIATED EQUATIONS 45 47 6.3. COMPLETED SIMULATION TOOL 48 6.3. COMPLETED SIMULATION TOOL 49 6.3. COMPLETED SIMULATION TOOL 50 6.3. COMPLETED SIMULATION TOOL 51 53 7.3. COMPLETE TRAJECTORY PROPAGATION 54 7.3. COMPLETE TRAJECTORY PROPAGATION 55 7.3. COMPLETE TRAJECTORY PROPAGATION 56 7.3. COMPLETE TRAJECTORY PROPAGATION 57 7.3. COMPLETE TRAJECTORY PROPAGATION 58 7.3. COMPLETE TRAJECTORY PROPAGATION 59 61 62 63 8.2. ERROR TOLERANCE 64 8.2. ERROR TOLERANCE 65 8.2. ERROR TOLERANCE 66 8.2. ERROR TOLERANCE 67 68 69 8.4. LAUNCH ALTITUDE 70 8.4. LAUNCH ALTITUDE 71 8.5. LAUNCH LATITUDE 72 8.5. LAUNCH LATITUDE 73 8.6. LAUNCH LONGITUDE 74 75 76 8.8. HEADING ANGLE 77 8.8. HEADING ANGLE 78 8.8. HEADING ANGLE 79 8.8. HEADING ANGLE 80 82 83 84 85 87 88 90 91 A.3. TRANSFORMATION BETWEEN DIFFERENT COORDINATE SYSTEMS 92 A.3. TRANSFORMATION BETWEEN DIFFERENT COORDINATE SYSTEMS 93 A.3. TRANSFORMATION BETWEEN DIFFERENT COORDINATE SYSTEMS 94 A.3. TRANSFORMATION BETWEEN DIFFERENT COORDINATE SYSTEMS 95 A.3. TRANSFORMATION BETWEEN DIFFERENT COORDINATE SYSTEMS 96 98 99 100 102 103 104 105 107 108 109 110 111 112 114 115